Physica 22D (1986) 187–204
North-Holland, Amsterdam

THE IMMUNE SYSTEM, ADAPTATION, AND MACHINE LEARNING

J. Doyne FARMER and Norman H. PACKARD*
The Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

and

Alan S. PERELSON
Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

The immune system is capable of learning, memory, and pattern recognition. By employing genetic operators on a time scale fast enough to observe experimentally, the immune system is able to recognize novel shapes without preprogramming. Here we describe a dynamical model for the immune system that is based on the network hypothesis of Jerne, and is simple enough to simulate on a computer. This model has a strong similarity to an approach to learning and artificial intelligence introduced by Holland, called the classifier system. We demonstrate that simple versions of the classifier system can be cast as a nonlinear dynamical system, and explore the analogy between the immune and classifier systems in detail. Through this comparison we hope to gain insight into the way they perform specific tasks, and to suggest new approaches that might be of value in learning systems.

1. Introduction

The immune system is a highly evolved biological system whose function is to identify and eliminate foreign material. In order to do this, it must be able to distinguish between foreign molecules (or *antigens*) and the molecules that constitute self. A prerequisite for the performance of this task is a powerful capability for learning, memory, and pattern recognition. In order to accomplish this, the immune system employs genetic mechanisms for change similar to those used in biological evolution. In the immune system, however, these processes function on a time scale that can be as short as a few days, making the immune system an ideal candidate for the study and modeling of adaptive processes.

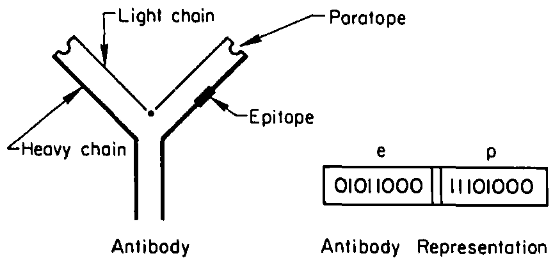
Motivated by the experimentally determined properties and existing theories of the immune system, in particular the network hypothesis of Niels Jerne [1–3], we have developed a model for the immune system that is easily simulated on a computer. This model turns out to have many features in common with an approach to machine learning and artificial intelligence introduced by Holland [4], called the classifier system. Our central purpose in this paper is to compare these two systems. In doing so we hope to give some perspective on the classifier and immune systems and provide insight into how both of them function.

A unique aspect of both the immune and classifier systems is that their equations of motion evolve in time. In typical studies in dynamical systems theory the dimension and the components of the state vector $(x_1, x_2, \ldots, x_N)$ are fixed. In contrast, while the list of variables in the immune or classifier systems is always finite, its composition varies with time. As components are created or destroyed the differential equations describing the dynamics change, and both the dimension N and composition of the state vector changes. Systems that allow new variables to be triggered into action were first examined by Rössler [5, 6] in the

*Permanent Address: The Institute for Advanced Study, Princeton, NJ 08540, USA.

context of autocatalytic networks. Of course, it is possible to embed such a system in an infinite dimensional space, and view the dynamics as fixed in time. We find it more useful, though, to construct an algorithm that generates the appropriate dynamical equations in the lowest possible dimensional state space, and study the dynamics in this context. In addition to the immediate application to the immune and classifier systems, we believe that this general approach may be useful to describe many other complex dynamical systems in which adaptation and evolution occur.

We begin the discussion with a brief introduction to relevant aspects of the real immune system, and then go on to present our model. We then briefly review Holland's classifier system, compare it to the immune system, write it in the form of a nonlinear dynamical system, and make some general remarks concerning the connection between nonlinear dynamics and learning systems in artificial intelligence and biology.



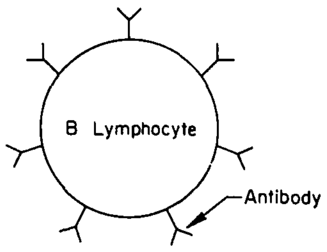


Fig. 1. A schematic representation of the structure of an antibody, an antibody as we represent it in our model, and a B-lymphyocyte with antibodies on its surface that act as antigen detectors.

2. A brief summary of the properties of the immune system

Given the rich chemical environment present in a highly evolved organism, it is inevitable that foreign organisms will attempt to invade in an effort to make use of these resources. To counteract this, in vertebrates the immune system has evolved to identify and dispose of foreign material. This is done in part by antibody molecules that tag foreign material and mark it for eventual removal by lymphocytes, phagocytic cells, and the complement system. A typical mammal such as a mouse or a human is thought to contain on the order of 10^7–10^8 different antibody types, each with its own unique chemical composition. The specialized portion of the antibody molecule used for identifying other molecules is called the *antibody combining region* or *paratope* (see fig. 1). The sequence of amino acids making up the paratope determines its shape and hence the set of other molecules that it can react with. If the shape of an foreign antigen molecule matches that of the paratope, the antibody can attach itself to the antigen, leading to its eventual demise. The regions on any molecule (antigen or antibody) that the paratopes can attach to are called *epitopes* (see fig. 1).

The set of all paratopes is like a very large collection of keys, and the set of all possible epitopes a very large collection of locks*. If the organism is to survive, it must be able to produce a key to open any lock. In fact, the number of

*The antibody combining region is generally formed by a groove or pocket in the 3-dimensional structure of the antibody. Thus, viewing a paratope as a lock might be more appropriate, and this is a convention sometimes found in the immunology literature. The important point is that keys and locks are complementary structures, just as paratopes and epitopes are complementary, and in this sense either may be considered as lock or key. Since many paratopes are generated to recognize a given epitope, we find it more convenient to think of paratopes as analogous to keys.

possible locks is so large that it is not possible to keep on hand a key for every possibility; nor is it even possible to store a separate blueprint for manufacturing each possible key so that it can be made on demand. The number of possible molecular shapes is so large that there is simply not enough DNA in a cell to do this. Instead, the DNA contains a large number of useful building blocks, which can be combined in different ways to make a large number of useful "master keys", each capable of opening an entire class of locks. Almost any lock can be opened, yet the specificities of the master keys may overlap, so that one lock may also be opened by many different keys. Perelson and Oster [7], in an idealized calculation, show that if antibody types (master keys) are made at random, the probability that a random epitope is not recognized is vanishingly small in systems with a realistic number of different antibody types. In fact, in experiments the number of antibodies that recognize a given epitope ranges from a few thousand to none at all. It seems likely that the segments of DNA that combine to produce new antibodies are not random, having undergone considerable evolutionary selection. Higher order control mechanisms may also influence the range of antibody types produced, causing additional nonrandomness.

A central difficulty of employing a very efficient recognition procedure is that antibodies may also recognize and destroy the tissue of the organism in which they reside. To prevent this, the immune system must either (1) block the production of antibodies that react with the molecules of the host organism; or alternatively (2) eliminate or suppress them once they are produced. This is called the *self – not-self recognition problem*. It is widely thought that plan (1) above is unfeasible, for the following reason: Any *a priori* information as to which antibodies recognize self would have to be coded for in the genes. However, since an animal gets genes from both its mother and father, the genes derived from its mother would recognize the molecules derived from its father as foreign, and *visa versa*. This is a strong argument that the prevention of self destruction must operate through plan (2), i.e. regulation rather than blockage of the production of self-destructive types.

One means of accomplishing regulation, suggested by Jerne [1–3], is called the *idiotypic network* theory. Because antibodies, like other molecules, have epitopes they are capable of being recognized by other antibodies. Different antibody types can be distinguished and regulated just as though they were antigens. An antibody type is thought to be enhanced when its paratopes recognize other types, and suppressed if its epitopes are recognized. Even if self-destructive antibody types are produced, their numbers can be regulated by other antibodies which recognize them and cause their destruction. There is experimental evidence indicating that such recognition processes can be extended to more than one level, with A recognizing B, which recognizes C, etc., allowing for the possibility of complex reaction networks forming. There are many ideas regarding how the elimination of self-destructive types is actually accomplished [8–11], including the possibility of a "learning phase" during the gestation of an embryo. (Note that different antibody types may have many epitopes in common. An epitope that is unique to a given antibody type is called an *idiotope*, hence the name idiotypic network for any scheme of regulation that works through the recognition of idiotopes).

Antibodies are manufactured by special cells knows as *B-lymphocytes*. Each lymphocyte has about 10^5 antibodies attached to its surface, with identical paratopes that serve as sensors to detect the presence of an epitope that this antibody type can respond to. When the appropriate epitope is detected, the lymphocyte is stimulated to reproduce more lymphocytes (i.e. to *clone*) and also to secrete free antibodies. This process of amplifying only those lymphocytes that produce a useful antibody type is called *clonal selection* and is illustrated in fig. 2.

The diversity of the immune system is maintained through the replacement of roughly five percent of the B-lymphocytes every day by new

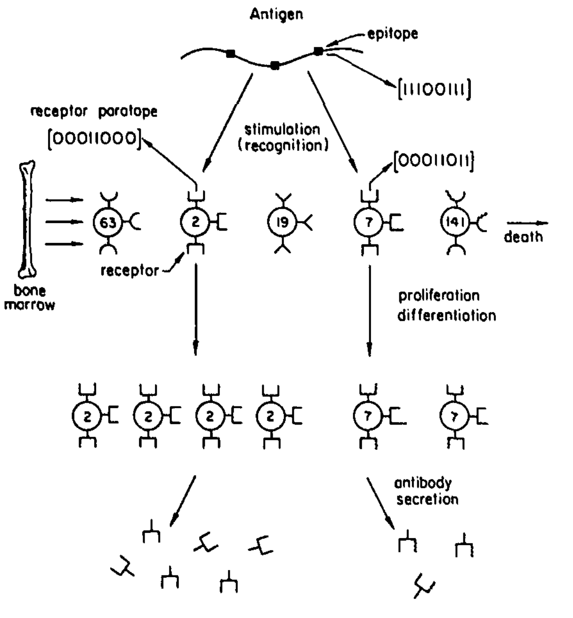


Fig. 2. A schematic illustration of clonal selection. At the top, a foreign molecule (i.e. an antigen) is represented as having 3 epitopes, each represented by the binary string 111000111. The bone marrow, shown on the left, pumps a diverse set of B-lymphocytes into the system, each of which has a large number of identical antibodies attached to it that serve as receptors. We show five different antibody types, labeled 63, 2, 19, 7, and 141, each of whose antibody types have a unique shape. Lymphocytes of type 2 have receptors with a paratope 00011000 which is perfectly complementary to the epitope of the antigen, indicating a strong reaction. Lymphocytes of type 7 have receptors that are partially complementary to the epitope (matching in 6 of 8 positions), indicating a weak reaction, and the remaining types match so poorly that they effectively do not match at all, and therefore do not react. Upon exposure to the antigen lymphocytes of types 2 and 7 are stimulated to proliferate. They also differentiate into a state in which they secrete free antibodies. The amount secreted is proportional to the degree of matching between the receptor and the antigen, so that lymphocytes of type 2 are strongly stimulated, whereas lymphocytes of type 7 are only weakly stimulated. Lymphocytes of type 63, 19, and 141 are not stimulated at all, and eventually die and leave the system.

lymphocytes generated in the bone marrow. In a human this amounts to approximately 10^6 lymphocytes produced every second. The number of different antibody types generated in the bone marrow during this process is not known, but is thought to be extremely large. The diversity comes about through the reshuffling of the host DNA that codes for the antibody genes. Within different B-lymphocytes the section of DNA that is expressed to create antibodies is generally different. Thus as cells are created within the bone marrow they are destined to make different antibodies. Although the distribution of antibody types is highly diverse, it is not necessarily totally random, since the basic building blocks are largely derived from a fixed set, as explained below.

We now describe the combinatorial process for generating diversity in more specific terms. Antibody molecules are composed of two polypeptide chains, a heavy chain and a light chain (see fig. 1). These chains are made independently, and it is thought that any light chain can combine with any heavy chain to produce an antibody. Thus 10^4 heavy chains and 10^4 light chains can combine to give 10^8 different antibodies, at least in principle. This combinatorial efficiency is also used at the level of the genes. The heavy chain is coded for by 4 separate genes, a V gene, a D gene, a J gene and a C gene. The light chain is coded for by 3 separate genes (V, J, C). Within each B-lymphocyte both the light and heavy chain are assembled by picking one gene from the library of V genes; then sequentially joining that gene to a gene picked from each of the other libraries (e.g. the D library). Thus again there is a huge combinatorial amplification in the number of different antibody types that can be formed from a small number of gene libraries, each containing a limited number of genes. Besides this combinatorial diversity, other mechanisms are employed to generate further diversity [11]. For example, errors are introduced at the boundaries between the gene segments when they are joined together, and point mutations accumulate within the gene libraries.

In addition to the production of new cells in the bone marrow, additional diversity is generated during the reproduction of B-lymphocytes when they are stimulated by recognizing an epitope. The mutation rate for antibody genes during this process is thought to be vastly greater than that for non-antibody genes [12]. It seems quite likely that this high mutation rate is an intentional mechanism designed to generate diversity, in order to provide the immune system with a means of rapidly generating new antibody types centered about a given type. For example, if a lymphocyte is stimulated to reproduce because its antibody type weakly matches something, replication errors may provide a means of finding types that match more closely, and thus react more strongly.

Later we will see that similar ideas have been adopted in classifier systems to try and improve the performance of algorithms. In this context such a procedure for generating diversity with combinatorial efficiency by mimicking the process of genetic reproduction is called a "genetic operator" or "genetic algorithm". As a means of creating new types with a high probability of being useful, genetic operators are believed to be vastly more efficient than random generation of new elements [24].

3. The model

The description of the immune system given above is greatly simplified. We have left out many important elements, such as T-lymphocytes and macrophages, in order to concentrate on the elements that contain the essence of the idiotypic network. In our model we make further idealizations, with the goal of producing a system that is simple enough to simulate on a computer but that still contains enough realism to embody characteristic properties of the network. We should say, though, that there are many different network theories for the immune system. It is our intent to simulate several of them, in order to determine which of them produce reasonable behavior and which do not. For simplicity, in this article we focus attention on a version of our model that we consider to be promising, discussing only a few possible variations. A more detailed discussion, along with simulation results, will appear elsewhere [13].

The first major simplification that we make is to represent the sequence of amino acids specifying the chemical properties of the epitope and paratope as binary strings. The simplest way of thinking about this representation is to view our antibodies as being composed of two amino acids, **0** and **1**. One could also parse the binary strings in such a way that a sequence of five binary numbers corresponds to an amino acid. In this way all twenty amino acids could be represented. We believe this level of sophistication is unnecessary. We also make the simplification that each antigen and each antibody type has only one epitope*. In reality an antigen or antibody may contain many different epitopes, with multiple copies of each (see fig. 2). Later versions of our model will include this added level of complexity. An antibody is thus represented as a pair of strings (p, e), with p denoting the paratope string and e the epitope string. We find allowed reactions between different antibodies and between antibodies and antigens by searching for complementary matches between strings. Since two molecules need not be exactly complementary in order to react with each other, we do not require exact matching between the paratope and epitope. Furthermore, to model the fact that two molecules may react in more than one way, strings are also allowed to match in any possible alignment. To be more specific, given an epitope string of length l_e and a paratope string of l_p, we define a matching threshold $s \leq \min(l_e, l_p)$, representing a threshold below which two antibodies will not react at all. Letting $e_i(n)$ denote the value of the nth bit of the ith epitope string, $p_j(n)$ denote the value of the nth bit of the jth paratope string, and denoting the exclusive or operation (which corresponds to complementary matching) by $\wedge$, the matrix of matching specificities m_{ij} is given by

$$m_{ij} = \sum_k G\Big(\sum_n e_i(n+k) \wedge p_j(n) - s + 1\Big), \qquad (1)$$

*One kind of interaction between antibodies that is not yet incorporated into the model described here is that a paratope may also function as an epitope, i.e. it can serve as a label to allow recognition by other paratopes. This type of interaction is dominant in a model due to Hoffmann [14].

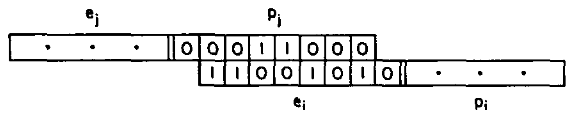


Fig. 3. Procedure used for computing partial matches. In this example $l_e = l_p = 8$ and $s = 6$. Epitopes and paratopes can match in all possible shifted alignments, though if $s = 6$, only alignments with $-2 \le k \le 2$ need be examined. Illustrated is the case $k = -1$ so that $e_i(n-1)$ is compared with $p_j(n)$. For the example shown, it is easy to determine that $G(x) = 1$ for $k = -1$ and $G(x) = 0$ for all other values of k, hence $m_{ij} = 1$.

where $G(x) = x$ for $x > 0$ and $G(x) = 0$ otherwise. The sum over n ranges over all possible positions on the epitope and paratope; the sum over k allows the epitope to be shifted with respect to the paratope. G measures the strength of a possible reaction between the epitope and the paratope. For a given alignment, i.e. value of k, G is zero if less than s bits are complementary, whereas if s or more bits are complementary, $G = 1 + \delta$, where δ is the number of complementary bits in excess of the threshold s (see fig. 3). If matches occur at more than one alignment, we sum their strength in eq. (1) to take into account the fact that molecules may be able to interact in more than one way, and thus react more strongly because they spend more time together than molecules that can interact in only one alignment.

When a lymphocyte recognizes an epitope, it can be stimulated to divide and also to produce free antibodies. The outcome of this process is to make both more free antibodies and more lymphocytes with the same antibody type attached to their surface. To simplify matters, in our model we lump together free antibodies with antibodies attached to cells, and keep track only of the total number of antibodies of a given type i in terms of the concentration variable x_i. The number of lymphocytes is thus dealt with only implicitly.

To model stimulation let us first take a very microscopic view and examine what happens when two different antibodies interact. In this interaction we assume that the paratope on one antibody recognizes the epitope on the other antibody. Assume that the outcome of this interaction is that the antibody with the paratope reproduces some fixed number of times, while, with some fixed probability, the antibody with the epitope is eliminated. The degree to which one antibody reproduces and the other "dies" is controlled by the degree of complementarity between the paratope and the epitope. Our model can be asymmetric with respect to antibody interaction, and is reminiscent of Jerne's [1-3] model and a later model of Richter [15, 16].

For many purposes it is possible to avoid simulating the microscopic dynamics in terms of differential equations for the concentrations. This is possible providing the system is well mixed and sufficiently large that the number of interactions needed to produce a significant change in the concentration of any particular type of antibody is large. Letting there be N antibody types, with concentrations $\{x_1, x_2, \ldots, x_N\}$, and n antigens with concentrations $\{y_l, \ldots, y_n\}$, the microscopic assumptions made above lead to the following set of differential equations:

$$\dot{x}_i = c\left[\sum_{j=1}^{N} m_{ji}x_ix_j - k_1\sum_{j=1}^{N} m_{ij}x_ix_j + \sum_{j=1}^{n} m_{ji}x_iy_j\right] - k_2x_i. \qquad (2)$$

The first term represents the stimulation of the paratope of an antibody of type i by the epitope of an antibody of type j. The second term represents the suppression of antibody of type i when its epitope is recognized by the paratope of type j. The form of these terms is dictated by the fact that the probability of a collision between an antibody of type i and an antibody of type j is proportional to x_ix_j. The parameter c is a rate constant that depends on the number of collisions per unit time and the rate of antibody production stimulated by a collision. The match specificities m_{ij} take into account what reactions occur and how strongly. Recall that in defining m_{ij} the convention was taken that the first index refers to the epitope, and the second to the paratope. The constant k_1 represents a possible inequality between stimulation and suppression. When $m_{ij} = m_{ji}$ the interactions between paratopes and epitopes become symmetric and our model is reminiscent of one developed by Hoffmann [17]. This system is driven by the presence of antigens, represented by the third term. To model the full immune response we must also introduce equations to model the antigen concentrations y_j, which may change in time as the antigens grow or are eliminated. The final term models the tendency of cells to die in the absence of any interactions, at a rate determined by k_2. We have explored several schemes for k_2. Thus far, the one that seems to work best is to vary k_2 in such a way as to keep the total concentration of the system at a constant value. The alternative is to hold k_2 a constant. We are currently exploring other more realistic approaches, but for the purposes of this paper the precise nature of the damping term is unimportant.

An essential element of our model is that the list of antigen and antibody types is dynamic, changing as new types are added or removed. Thus N and n in eq. (2) change with time, but on a time scale that is slow compared with the changes that occur in the x_i. In our simulations eq. (2) is integrated for a period of time, then the composition of the system is examined and updated as needed. To perform this updating we place a minimum threshold on all concentrations, so that a variable and all of its reactions is eliminated when the concentration drops below the threshold. This simulates the elimination of an antigen or the death of the last cell expressing a given antibody type. This is an important property, for new space is needed in a finite animal (or computer memory) to allow for the addition of new antibody types.

In our model we generate new antibody types in any one of a variety of ways. Mimicking the processes that occur during the reproduction of real lymphocytes, from time to time we apply genetic operators to the paratope and epitope strings, such as crossover, inversion, and point mutation. *Crossover* is implemented by randomly selecting two antibody types, randomly choosing a position within the two strings, and interchanging the pieces on one side of the chosen position in order to produce two new types. Paratopes and epitopes are crossed over separately. *Point mutation* is implemented by randomly changing one of the bits in a given string, and *inversion* is implemented by inverting a randomly chosen segment of the string. Random choices are weighted according to concentration whenever appropriate.

Since our model concerns itself with variations of the antibody types themselves, the processes described above are closer to a model of genetic changes that occur during cloning than they are to production of new types in the bone marrow through reshuffling of the gene library. Since the antibodies in the system reflect the gene library anyway, this difference is probably not essential for the first stages of simulation. A crude alternative, just for comparison, is to simply generate new antibodies *de novo* using a random sequence generator.

Antigens can be generated by a variety of mechanisms, either randomly or by design. The same antigen can be repeatedly presented to the system to see if the system learns to eliminate it more efficiently after each exposure. After the system learns to cope with one antigen we can present a panoply of random antigens and see if the system then forgets what it has already learned. The number of antigens that the system is simultaneously exposed to can be varied, as can the rate at which new antigens are presented to the system.

The ability of the system to learn and to effectively eliminate antigens depends upon the detailed regulatory rules governing the operation of the system. In Jerne's network hypothesis internal recognition events between antibodies play a crucial regulatory role. Even without the stimulation of external antigens, experiments performed in germ-free environments show that there is considerable activity in the immune system of mice. In order to examine this we performed preliminary simulations of systems without external antigens and without the time-dependent introduction of new antibody types. We find, not surprisingly, that antibodies whose paratopes match epitopes are amplified at the expense of other antibodies. If $k_1 = 1$ (equal stimulation and suppression) and $k_2 > 0$ then every antibody type eventually dies due to the damping term. Letting $k_1 < 1$ favors the formation of reaction loops, since all the numbers of a loop can gain concentration and thereby fight the damping term. As N increases, so does the number and length of the loops. The robust properties of the loops allow the system to remember certain states even when the system is disturbed by the introduction of new types. New antibodies that are introduced into the system are retained and their concentration is increased if they recognize other molecules in the system, either internal or external. Antibodies that do not recognize other elements in the system are eventually washed out. Thus besides immunological memory, our system also exhibits "immunological forgetting", in which molecular shapes that are not useful over some reasonable time scale are eliminated. In the real immune system it is thought that a large majority of antibody types are eliminated before ever encountering a complementary antigen.

In the real immune system, antigens are sometimes remembered for periods that in some cases are comparable to the lifespan of the organism. The exact mechanisms for this are not known. One hypothesis suggests that the antigen itself (or some partially degraded form), is sequestered in lymph nodes and other organs, and periodically detected by the immune system, thereby reinforcing memory in a brute force fashion through constant exposure to the antigen. Because antigens are potentially dangerous this strategy is highly risky, and would seem unlikely to be capable of withstanding the test of evolution. Another hypothesis for memory is that B-lymphocytes that have reacted to an antigen simply go into a dormant state lasting for decades, waiting for the possibility of a recurrence of the same antigen. It is known that B-lymphocytes can go into a resting or "memory" state for periods of weeks or possibly months, but it is not known if they can survive

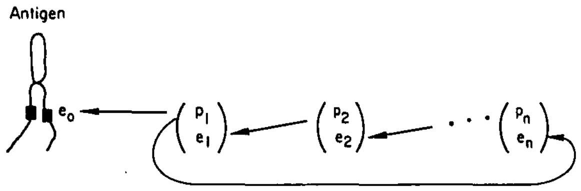


Fig. 4. The formation of a cycle allows the antigen with epitope e_0 to be "remembered". Arrows denote recognition via our string matching algorithm. Paratope p_i recognizes epitope e_{i-1} for $i = 1,2,\ldots,n$. To form a cycle, we assume that by chance p_1 recognizes e_n in addition to e_0. Thus e_n must resemble the antigen e_0. If the antigen is eliminated, the existence of the cycle can maintain the concentration of Ab_1, an antibody that specifically recognizes the antigen.

for periods of years without being stimulated to divide. Stimulation could come from sequestered antigen or by idiotypic network interactions. Our model suggests the following third alternative for memory by means of the idiotypic network. In the presence of an antigen, the concentrations of all antibodies that recognize it will be increased. Let us call this set of antibodies Ab_1. Further, because of our recognition rules, the concentrations of all antibodies that recognize epitopes on Ab_1 antibodies will also increase. Let us call this set of antibodies Ab_2. Continuing in this way, let Ab_n antibodies have paratopes which recognize the epitopes of Ab_{n-1} antibodies. If Ab_n is like the original antigen, it will be recognized by Ab_1, forming a cycle or autocatalytic loop of order n, and even in the absence of antigen the shape of the antigen is remembered in Ab_n. (See fig. 4.) Note that if paratopes are also assumed to function as epitopes (and chemically it is hard to see how this would not be the case), then the even values of n are guaranteed to resemble the antigen. Since the matching process is not perfect, though, the best match is not necessarily obtained for $n = 2$, allowing for the possibility of higher order cycles. Cycles have been studied in other models of immunological networks by Hiernaux [18] and Seghers [19], and in models of evolution by Kauffman [20] and Eigen and Schuster [21].

4. A brief summary of the classifier system

Our simplified network model of the immune system is closely analogous to the classifier system, which we now briefly describe. The classifier system represents a new method of problem solving and learning in artificial intelligence, and has been successfully employed in diverse applications such as gas pipe line control [22] and poker [23]. Approaches to the classifier system are as diverse as theories of the immune system. Different means of implementing the classifier system are still being tried, and there are many variants of the basic idea. For definiteness we will summarize the classifier system more or less as described by Holland [4], and comment on possible variations only when they are relevant.

It is convenient to think of the classifier system as a black box whose purpose is to perform a computation or control function *without* having to be explicitly programmed for the specifics of a given task. An example of a task for which a classifier system has been successfully employed is control of a one-dimensional gas pipeline, as shown in fig. 5b. (See Goldberg [22].) In this example, inputs to the classifier system come from pressure sensors along the pipeline, and the system can respond by turning any one of several compressors on or off. The objective for this problem is to overcome surges in supply and demand by tuning the operation of the compressors. The classifier system that Goldberg implemented contained no preprogrammed information about pipeline control. Starting the classifier rules with random values, the system was able to learn this task well enough that it eventually was capable of performing it as well as a human operator.

A schematic illustration of the internal elements of a classifier system is given in fig. 5a. The system can be divided into two parts, a set of *rules* or

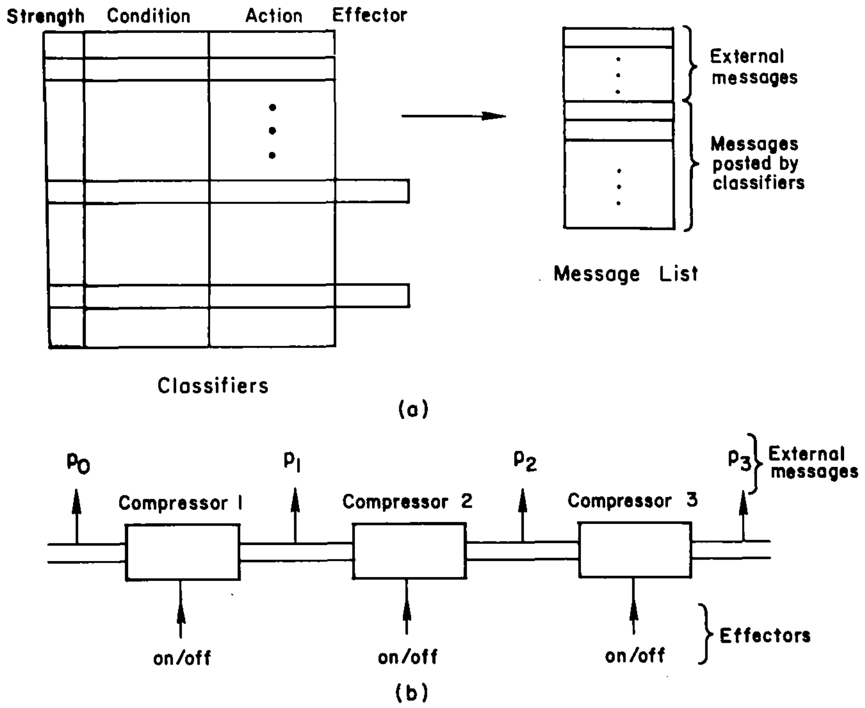


Fig. 5. A schematic illustration of the classifier system, as discussed in the text. (a) Illustrates the components of the classifier system, and (b) shows a typical example that the classifier system has been applied to, in this case the control of a one-dimensional gas pipeline (cf. [22]).

classifiers, and a *message list*. The rules contain information giving possible responses of the system, while the message list contains the inputs from the external environment and provides a forum for the rules to communicate and interact with each other. The messages on the list are by convention binary strings of a fixed size, i.e., they consist of a fixed number of the characters **0** and **1**. At least one of the messages on the list is provided by the external environment. In the example given above, some of the messages correspond to the readings of the pressure sensors, encoded in binary form. The other messages on the list are supplied internally by the rules themselves. The list of messages is dynamic, changing as the external inputs change or as the rules place new internal messages. Through the process of reading and posting messages the rules "process" information, in a manner described below.

The classifier rules consist of multiple parts: one or more *conditions*, an *action*, and a *strength*. The conditions and actions are strings whose length is the same as that of the messages on the message list. The conditions allow the classifier to "read" the message list by searching for matches of the conditions against the messages posted on the list. When a match is found, if certain criteria are met a rule is allowed to post its action part as a new message on the list. Some rules have a special role as *effectors*, meaning that their action parts cause external outputs. In the pipeline example, this might be the address of a compressor to be activated. Typically, though, when a message is posted it is simply read by other rules. The posting of an external message, say from a pressure sensor, generates a series of internal messages posted by different rules until a message from an effector is eventually posted, causing a response such as the activation of a compressor. This chain of events is like a computation. In fact, it can be shown that a classifier system can implement an arbitrary set of boolean functions.

The *strength* is a number associated with each rule which is designed to indicate its value to the system. This forms the basis of learning. If a rule helps bring about useful responses, it gains strength, and otherwise it loses it. Strong rules are given more influence than weak rules in determining the overall response of the system. The problem in allocating strength properly is that the actions of a rule are often indirect. A rule may emit a message triggering a chain of internal responses, which eventually lead to a valuable external output, but it may be very difficult to determine which messages were useful as opposed to irrelevant or harmful. In the classifier system the value of a rule is established by means of a mock economy, in which the rules "pay" each other for information. The usefulness of the information is ultimately measured by the performance of the system based on its external outputs. When a useful output takes place, the rules that accomplished the output are "paid" and strength is passed back via the economy. This procedure is sometimes called the *bucket brigade*.

To introduce the individual parts of the classifier system and motivate their function, our discussion up until now has been schematic. To make things concrete, we will now explain in more detail how the classifier system actually functions. We have thus far neglected to mention that, in contrast to the binary messages on the list, the rules are actually built out of three symbols, **0**, **1**, and #. The # functions as a "wild card", matching with anything. A match is defined to occur only when the rule matches the message on every symbol. If the condition contains many #'s, though, the effect is the same as if partial matches were allowed. Such nonspecific rules are general purpose, since they will match to many different input messages. In some implementations of the classifier system the rules are also allowed to have #'s in their action parts as well, interpreted as a *pass-through*; when a match occurs, if there is a # in the action part the corresponding bit of the message on the list is passed through to the next message, so that it is a combination of pieces from the action part of the rule and pieces derived from the previous message. For simplicity, in our discussion here we will assume that pass-through is not implemented, and that the action parts consist of a fixed string of ones and zeros. This implies that all the rules posted on the list (except for external messages) correspond directly to the action part of at least one of the rules. In general, classifier systems can have rules with more than one condition part. For simplicity, however, we will discuss the case in which each rule has only one condition.

In order to distinguish between different types of information, one of the bit positions in a message can be used as a *tag*. This can be an explicit delineation, for example, the convention might be taken that if the first bit of the message is **1** it is an external message, and if it is **0** it is an internal message. Tags also evolve implicitly, so that the system may use the presence of a value in a certain position to trigger a certain class of responses. Typically it is difficult for an external observer to determine what messages actually do; their meaning is only in reference to the other parts of the system.

When a rule finds a match with a message already on the list, it must bid to place its action part on the message list on the next step. There are many possibilities for how this can be done. We follow Holland [4], and assume that the rule makes a bid proportional to the product of its strength x_i and the specificity of the match m_{ij}. For the classifier system, m_{ij} is defined as the number of non #'s; if the match is not perfect (i.e. if all non # positions do not agree), $m_{ij} = 0$. Since there is only space on the message list for a finite number of messages, the decision about which rules get to post messages is determined through a competitive bidding procedure in which only the highest bidders are allowed to post messages. If a rule is allowed to post a message, it must *pay* the rule that supplied the message that it matched to. This payment is equal to the bid. Whenever a response is made to the external environment, the system is rewarded according to the desirability of the response. Again, we follow Holland and assume that all classifiers with messages currently active on the list are paid equally by the external environment whenever a desirable response occurs. In the pipeline example, whenever a compressor is activated, the active classifiers will gain strength according to the extent to which the adjustment brings supply closer to demand. In order to be active, however, these classifiers depend on other rules to emit messages that they can match to. They must pay these rules, which in turn must pay other rules, etc. The result is that *all* of the rules participating to bring a given result about receive payment, thus allocating credit for indirect actions. It is this process that we earlier referred to as the bucket brigade.

Finally, in order to ensure that useless rules decline in strength, all rules are taxed according to their strength. Rules that do not post messages that other classifiers read will steadily decline in strength, which in turn gives them less influence in determining the responses of the system.

If the rules are started out with random strengths, the classifier system will initially make random decisions. As it accumulates experience, however, the strength of the rules will reallocate itself according to their usefulness. Strong rules will make more successful bids, thus gaining more decision-making power and influence on the behavior of the system as a whole. This is how the system learns.

Of course, unless the best rules are already contained in the system, its behavior will not be optimal. One solution would be to randomly introduce new rules to the system, and let them compete with the old rules. This process is, however, extremely slow and inefficient, since most of the time random rules are worthless, and the amount of time needed for the system to improve is enormous. To solve this problem, the set of rules present in the system is allowed to change with time in a manner that mimics biological evolution, by modifying the rules *via* the same kind of genetic transformations that occur in DNA. Rather than starting from scratch, new solutions are strongly based on old solutions. To accomplish this, classifiers are randomly selected according to their strength, and allowed to "breed" with each other, through the application of genetic transformations such as cross-over, mutation, and inversion to their strings. (These operations are precisely the same as those used in the immune model.) The weakest rules are eliminated from the system and replaced by the offspring of the strong rules. These new offspring are then given a chance to let their strengths change in competition with the old rules. Once the system has had time to reallocate strength, the process is repeated. Through this procedure new solutions evolve that are superior to those originally present. As discussed by Holland [24], the most efficient approach is to use cross-over much more commonly than the other genetic operators, resulting in a rate of generation of useful solutions which is much faster than random or random-walk generation. The use of genetic operators to produce new solutions give the classifier system a capability that is analogous to creativity.

The beauty of the classifier system is that, at least in principle, the input-output interface is the only part which must be hand-tailored for a specific problem. In practice, however, the problems that have been solved using classifier systems are still quite simple. Also, a considerable amount of human discretion must be used in setting the parameters of the system to achieve optimal performance. Nevertheless, the classifier system presents one of the best current alternatives for an algorithm that can accomplish general purpose learning and creative problem solving.

5. Equations of motion for the classifier system

To make the analogy between the classifier and immune systems clearer, we write the classifier system in the form of a dynamical system. The resulting equation of motion is similar in form to that of our immune system model. Let x_i^t denote the strength of the ith rule at time t. (The ordering of the rules is arbitrary. Also, since the superscripts almost always have the value t, they will be omitted unless we need them for emphasis.) Let m_{ij} be the specificity of the match between the condition part of classifier i and a message posted by classifier j, i.e. the first index refers to the condition and the second to the action. Matches can also occur with external messages, which we will assign indices ranging from $N+1$ to $n+N$. Note that this notion of specificity is slightly different than that used in the immune system model. Here, m_{ij} is nonzero if all non-#'s in the ith condition match the corresponding symbols of the jth message, in which case m_{ij} is the number of non-#'s.

To formulate equations for the redistribution of strength, we must explicitly write down the amount of strength that is gained or lost by a given rule. Strength can be gained due to payments either by other rules or the environment, and it can be lost either by bid payments to other rules or by taxation. Fixing attention on the ith rule, the size of the bid made for a nonzero match with the ith message is $m_{ij}x_i$. It will lose strength, however, only if the bid is accepted, i.e. if the bid exceeds the bid threshold T*. This can be taken into account by multiplying by a Heaviside function, defined as $H(x)=1$, for $x>0$, and $H(x)=0$ otherwise. The only possibilities for making a match at any given moment are with messages on the list. To take the action of the list into account, we introduce a variable f_j, which is 1 of the jth classifier has its action posted on the message list, and 0 otherwise. Finally, in the event that a classifier matches several messages and has more than one bid exceeding the threshold, we adopt the convention that it simply pays all of the classifiers that posted the matched messages a fixed fraction c of the full amount of each bid. The change in strength of the ith classifier due to bid payments

*The bidding threshold T is not constant, but actually varies with time in order to keep the number of messages on the list constant as different rules become active and the bids change. The performance or scoring function P is also a function of time. In addition, if the set of rules present changes due to genetic operators, or as the environmental messages change, m_{ij} is a function of time.

can be written

$$\Delta x_i = -c \sum_{j=1}^{N+n} m_{ij} x_i f_j H(m_{ij} x_i - T).$$

The sum is taken to $N+n$ because the condition may match messages from the n external messages as well as messages posted by the N classifier rules.

Likewise, a rule's strength is increased when it receives payments from other classifiers. These payments are proportional to the strength of the bids of others, and can only occur if the ith classifier has its message posted on the list. The amount of strength gained is the size of the bid of the jth classifier, $m_{ji} x_j$, multiplied by f_i to take account whether i has its message on the list, multiplied by a Heaviside function to take into account whether the bid is accepted. This must be summed over all other classifiers. The result is

$$\Delta x_i = c \sum_{j}^{N} m_{ji} x_j f_i H(m_{ji} x_j - T).$$

There are other mechanisms for the strength to change. A rule can lose strength from taxation, which we will assume is proportional to strength. It can gain strength through direct payment from the performance function, P, if its message is active on the list.

When we combine all of the above effects, the final equation is

$$\begin{aligned}\Delta x_i &= x_i^{t+1} - x_i^t \\ &= c\left[\sum_{j=1}^{N} m_{ji} x_j f_i H(m_{ji} x_j - T) \right. \\ &\quad \left. - \sum_{j=1}^{N} m_{ij} x_i f_j H(m_{ij} x_i - T)\right] \\ &\quad + f_i P^t - k_2 x_i. \end{aligned} \tag{3}$$

The first term takes into account payment received from messages posted on the previous time step. The second term takes into account payments to other classifiers for successful bids. The third term is the external driving term, coming from payoffs by the performance function (again, we assume here that every classifier with an active message gets paid off), and the final damping term is due to taxation. The damping coefficient in eq. (2) was taken to be time dependent in order to control the overall mass of the system. This same mechanism may be employed in eq. (3) to control the average value of the classifier strengths. The action of the message list is contained in the terms of the form $f_j H(\ )$. f_j can be computed recursively using the formula

$$f_i^{t+1} = H\left(\sum_{j=1}^{N+n} f_j^t H(m_{ij} x_i^t - T^t) \right), \tag{4}$$

which simply states that in order to post a message at the next time step a classifier must make at least one successful bid. (The H outside the sum is to treat the case in which more than one bid is successful.)

6. A comparison of the classifier and immune systems

There is a very close analogy between our model of the immune system and the classifier system. This is not a coincidence. Although Holland did not have the immune system specifically in mind, he made extensive use of biological principles in designing the classifier system. We have summarized some aspects of the analogy between the immune and classifier systems in table I*.

In deriving the above equations of motion, we have used a version of the classifier system similar to the one discussed by Holland [4]. Comparing eq. (3) to eq. (2), it is clear that the main difference between the two systems is the nature of the nonlinearity in the two equations. In particular, the quadratic nonlinearities of eq. (2) are replaced

*There also exists an analogy between the immune system and neural networks [25–27].

Table I
A comparison of the major components of a classifier system and an immune system

Classifier system vs.	Immune system
classifier	antibody type
condition	epitope
action	paratope
strength	concentration
specificity	specificity
tax	dissipation (loss) term; eg. the $-k_2 x_i$ term in eq. (2).
payoff	antigen reduction
external message	antigen
effector	
message list	all paratopes and antigens, together with competition terms
economy	concentration update rule
genetic operators to generate new classifiers	genetic operators to generate new anti-body types
performance function	rate of antigen removal

by the more nonlinear $f_j H(\ )$ terms of eq. (3), which are a manifestation of the finite message list. The form of eq. (4), however, is highly dependent upon the means of selection for placement on the list. For a classifier system to function properly, it is important that it be capable of maintaining useful elements while still being able to assimilate new elements *via* the genetic algorithm. (This property has been called *gracefulness* by Holland.) The harsh nonlinearity caused by the finite message list sometimes makes this difficult, since a classifier which is barely strong enough to get on the list but crucial to some operation may have a difficult time surviving. To provide diversity and prevent stagnation a stochastic element is sometimes added to the bidding process [23].

As a point of comparison, suppose that the selection is done entirely stochastically. There are many ways to implement this. For definiteness, suppose that when a match occurs a classifier posts its action on the message list with a probability proportional to the strength of the rule that posted the message it matches. Furthermore, assume that the bidding constant c is small, so that the change in strength on any one time step is small. Also assume that the environmental messages change gradually, so that m_{ij}, T, and P also change slowly. With these assumptions the random selection procedure can be averaged over many time steps, resulting in the following deterministic equation of motion:

$$\dot{x}_i = c\left[\sum_{j=1}^{N} m_{ji} x_i x_j - \sum_{j=1}^{N+n} m_{ij} x_i x_j + x_i P\right] - k_2 x_i. \tag{5}$$

Comparing eq. (5) to eq. (2), we see that, except for the third term, these two equations are identical. The difference is relatively insignificant, having to do with the nature of the driving term.

Adding noise to the bidding procedure as done above softens the harsh nonlinearity and eq. (5) shows that the resulting equation has the same quadratic nonlinearities found in the immune system. In certain circumstances, the use of deterministic equations of this type may be a faster and more direct solution than detailed simulation of a stochastic process. (Note that the same dichotomy exists in the immune system; although we have chosen to represent the immune system as a differential equation, we could have simulated the interaction of individual antibodies, which would have lead to a system more similar to the classifier system.)

Although there are several easily discernible differences between the immune and classifier systems, for example the algorithm used to compute the specificity matrix m_{ij}, most of these differences are minor. One principal difference between the two systems is in their interaction with the external environment. For the classifier system there is a clear separation between inputs and outputs. For the immune system, however, this separation is not so clear. All information about the external environment is obtained from the antigens, which play a dual role as external messages and effectors, causing a driving term which is in much the same form as that of the internal interactions of the system. In fact, an antigen is more like an external condition than an external message, since it has a strength associated with it and it enters the equations of motion with the same sign as the conditions.

Another pronounced difference lies in the system of message passing used in the classifier system. The message list serves two purposes: First, it provides a mechanism for implementing competition between the rules by restricting matching. Although this feature is not explicitly present in the immune model, the same effect is achieved through the nonlinear competition terms in eq. (2), evidenced by the similarity to the implementation of the classifier system represented in eq. (5). The second purpose of the message passing system is to provide a means of computation, as discussed below.

In the classifier system computation may be performed even when the strengths remain fixed in time. In this case the classifier system becomes a "production system", which amounts to a stimulus-response lookup table, implementing boolean functions to make connections between its inputs and outputs. In many implementations, the system effectively evolves to a fixed point in the space of classifier strengths. This comes about either because the strengths of the important classifiers are unperturbed by the new rules generated by the genetic algorithm, or because the system is designed to have the action of the genetic algorithm being gradually quenched, i.e. as the system gets better and better, fewer and fewer new classifiers are added. Another possibility that seems to have not been exploited very much is that computation and dynamic adaptation can be accomplished through the time evolution of the strengths.

The variation of concentration, which is the analogous quantity to strength, is in fact the basic mechanism through which computation is achieved in the immune system. Without the dynamic aspect of concentration the immune system would be incapable of functioning, since the ability to eliminate an antigen depends crucially on the ability to generate large increases in relevant antibody concentrations. The immune system model thus suggests enhancements for the design of a classifier system that could allow the classifier system to make more computational use of strength dynamics.

Another major effect of the influence of the environmental interactions has to do with the continuity of the dynamics. Because the immune system consists of a very large number of individual lymphocytes and antibodies, and since the concentrations of antigens change fairly continuously, for most purposes we can model the dynamics with a differential equation. For the classifier system, however, this depends very much on the parameters of the system and the rate at which the external messages change. If the external messages change slowly, the contents of the message list will also change slowly, and a differential equation will provide a good model. Alternatively, when stochastic bidding schemes are used, we can write an equation for the time averaged quantities. In either case the close analogy to the immune system is preserved. But in the case that we use a deterministic bidding scheme as in eq. (3) and allow the external message list and/or the payoff function to change significantly on successive iterations, the contents of the message list can change on a rapid time scale even if the bidding constant c is very small. The net result is that some properties of the classifier system (e.g., the function f_j) change on a time scale that is significantly faster than that on which the strength changes, and we must consider the classifier system as a mapping rather than a continuous system*.

In one sense it is an accident that there is any similarity at all between the immune and classifier systems. The classifier system does not have to

*For the cases where the strengths/concentrations are changing slowly compared to the composition of the message list, it becomes advantageous to store the match matrix m_{ij} through a linked list instead of recomputing the matches every time step. This technique gives a considerable speed advantage in our immune system simulations.

satisfy any of the constraints of nature, since ultimately its only function is to provide a general purpose scheme for adaptive problem solving. In its creation it was hand-tailored by Holland for this purpose. In another sense, though, the similarity is no coincidence, since although Holland did not have the immune system specifically in mind, he was strongly guided by biological principles.

Eqs. (2), (3), and (5) are all of the basic form

$$\Delta x_i = \text{internal interactions} + \text{driving} - \text{damping}, \tag{6}$$

where the precise form of these terms depends on exactly how the internal interactions, driving and damping take place. This is also the basic form of other systems occurring in biology, such as the equations underlying coupled autocatalytic reactions, or the Lotke-Volterra equations for population dynamics, which had a strong influence on Holland. Nonetheless it is interesting that the analogy between the immune and classifier systems is so strong, considering that they were conceived independently. The fact that there are so many apparently accidental correspondences between the two systems points out the extent to which artificial intelligence can benefit from the study of natural systems, and suggests that there may be certain universal approaches to the design of efficient learning systems.

A feature of both the immune and classifier systems which is not apparent from an inspection of the equations of motion is the action of genetic operators, which cause the form of the equations to change in time, on a time scale which is slow compared with the time scale for strength or concentration to equalize. This seems to be a universal feature – when new elements are introduced, the system needs some time to test them to determine whether or not they are useful. On one hand, if the introduction of new elements is done too frequently, the system may be unstable, but on the other hand, if it is not done often enough, the system may be slow to respond to its environment. The generation of new "solutions" act in precisely the same manner in both systems. The end result in either case is to provide a form of creativity, giving a capability to cope with unforeseen circumstances. In the immune system this is an essential feature, without which an organism would die. In both systems, the use of genetic operators such as crossover provides a much more powerful and efficient means of generating diversity than simple random variation [4, 24]. The general concept of dynamical systems in which the rules change according to the state of the variables is not commonly studied, and presents a new challenge to the theory of dynamical systems.

7. Concluding comments

We have presented a new network model for the immune system. Although it was not designed with this in mind, it turns out to embody many of the features of the classifier system. Our model consists of a set of differential equations, together with a threshold to remove useless antibody types, and genetic operators to introduce new ones. We have also presented a version of Holland's classifier system, casting it in the form of a dynamical system. As we have shown, the equations of motion for the two systems are of the same form, illustrating that in general equations of this type may have broad utility for learning. The strong similarity of these two systems reinforces Holland's original notion that natural systems provide a rich source of ideas and insight for the development of effective schemes for parallel computation.

We have observed that both the immune system and the classifier system are strongly nonlinear dynamical systems having the feature that the equations of motion and the state space of the system both change with time. Thus, the equations of motion are complicated and cannot be completely specified *a priori*. Simplicity is nonetheless found in the simple rules for the construction and modification of the equations. The equations of motion are dissipative, and include both driving and damping to enable selection of useful sets of variables and to achieve effective computation.

Equations having the form of eq. (6) appear to have sufficient richness to accomplish nontrivial computation, learning, and response to external demands. The manner in which these computations are done is distinctly different than that of a standard serial Turing machine (though of course a Turing machine could simulate any such system as long as it remained finite). Systems of the type discussed here have the great advantage that they are automatically parallel, with no problems as the scale of the system is increased. Furthermore, they incorporate in a very natural manner a means of allowing many different computational strategies to be present simultaneously, with a natural means of regulating competition for control of the system's responses.

One of the central features of both these systems is that the equations of motion change in time, in response to changes in the state of the system. When variables are deemed useless, they are removed from the system and replaced by others. In both cases, the replacement is done by genetic algorithms, which mimic the biological reproduction process. Thus creativity is achieved in much the same manner as it is in biological evolution. The end result is that with the passage of time the system arrives at a state where it performs a given task more efficiently than it did with the set of solutions originally at hand. As demonstrated by Holland [24], this process is vastly more efficient than random replacement. Systems of this type have received very little attention in the theory of dynamical systems, and pose several very interesting fundamental questions. For example, can the notion of an attractor be generalized to apply to such systems? Or, are there other more useful notions that can be used to classify their behavior?

Ultimately it may be profitable to investigate the process of adaptation on a higher level than that of an individual organism or an individual classifier system. We could ask how ensembles of organisms whose immune systems obey different rules evolve when they are in competition with each other. Or alternatively, classifier systems whose parameters for bidding, payoff, breeding rate, etc. are set to different values can be allowed to compete with each other. An approach in this direction has been taken by Smith [23].

Although our primary goal in simulating the immune system has been to learn more about the internal operation of the immune system in real organisms, the close correspondence between the immune system and the classifier system leads us to believe that generalized versions of our model may be capable of performing artificial intelligence tasks. For example, by changing the structure of epitopes and paratopes from one-dimensional strings to two-dimensional matrices, more conventional pattern recognition tasks may become possible, e.g. recognizing a letter on a noisy background. The immune system has evolved over millions of years to efficiently and accurately recognize patterns. By abstracting from the immune system the essence of the methods it employs, new powerful pattern recognition algorithms may result.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy. We would like to thank John Holland and Stephen Pope for valuable conversations, and Geoffrey Hoffmann for critical comments.

References

[1] N.K. Jerne, "The immune system", Sci. Am. 229 (1973) 52–60.
[2] N.K. Jerne, "Towards a network theory of the immune system," Ann. Immunol. (Inst. Pasteur) 125 C (1974) 373–389.
[3] N.K. Jerne, "The immune system: A web of V-domains," Harvey Lect. 70 (1976) 93–110.
[4] J.H. Holland, "Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-base systems," in: Machine Learning 2, R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds. (Morgan Kauffman, Los Altos, in press), chap. 20.

[5] O.E. Rössler, "A system theoretic model of biogenesis," Z. Naturforsch. 26b (1971) 741–746.

[6] O.E. Rössler, "Chemical automata in homogeneous and reaction-diffusion kinetics," Springer Lect. Notes in Biomath. 4 (1974) 399–418.

[7] A.S. Perelson and G.F. Oster, "Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self-non-self discrimination," J. Theoret. Biol. 81 (1979) 645–670.

[8] G.W. Hoffmann, "Incorporation of a nonspecific T cell dependent helper factor into a network theory of the regulation of the immune response," in: Theoretical Immunology, G.I. Bell, A.S. Perelson, and G.H. Pimbley Jr., eds. (Marcel Dekker, New York, 1978), pp. 571–602.

[9] J. Hiernaux and C. Bona, "Network regulatory mechanisms of the immune response", in: Regulation of Function of Lymphocytes by Antibody, C. Bona and P.A. Cazenave, eds (Wiley, New York, 1979).
C. Bona and J. Hiernaux, "Immune response: Idiotype anti-idiotype network," CRC Crit. Rev. Immunol. 1 (1981) 33–81.

[10] G.W. Siskind, T. Hayama, G.M. Shepherd, A.F. Schrater, M.E. Wekler, G.J. Thorbecke and E.I. Goidl, "Autoanti-idiotype antibody production following antigen injection and immune regulation," Annals N.Y. Acad. Sci. 392 (1982) 345–348.

[11] S. Tonegawa, "Somatic generation of antibody diversity," Nature 302 (1983) 575–581.

[12] M. Wahl, P.D. Burrows, A. von Gabain and C. Steinberg, "Hypermutation at the immunoglobulin heavy chain locus in a pre-B-cell line," Proc. Natl. Acad. Sci. USA 82 (1985) 479–482.

[13] J.D. Farmer, N. Packard and A.S Perelson, in preparation.

[14] G.W. Hoffmann, "On network theory and H-2 restriction," in: Contemporary Topics in Immunobiology, vol. 11, N.L. Warner, ed. (Plenum, New York, 1980), pp. 185–226.

[15] P.H. Richter, "A network theory of the immune system," Eur. J. Immunol. 5 (1975) 350–354.

[16] P.H. Richter, The network idea and the immune response, in: Theoretical Immunology, G.I. Bell, A.S. Perelson and G.H. Pimbley Jr., eds. (Marcel Dekker, New York, 1978), pp. 539–569.

[17] G.W. Hoffmann, "A theory of regulation and self-nonself discrimination in an immune network," Eur. J. Immunol. 5 (1975) 638–647.

[18] J.Hiernaux, "Some remarks on the stability of the idiotypic network," Immunochem. 14 (1977) 733–739.

[19] M. Seghers, "A qualitative study of an idiotype cyclic network," J. Theoret. Biol. 80 (1979) 553–576.

[20] S.A. Kauffman, "Autocatalytic sets of proteins," J. Theoret. Biol., in press.

[21] M. Eigen and P. Schuster, The Hypercycle: A Principle of Natural Self-Organization (Springer, New York, 1979).

[22] S. Goldberg, "Computer Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning," Dissertation, University of Michigan (1983).

[23] S. Smith, "A Learning System Based on Genetic Adaptive Algorithms," Dissertation, University of Pittsburgh, (1980).

[24] J.H. Holland, "Genetic Algorithms and Adaptation", Technical Report #34, Univ. Michigan, Cognitive Science Department (1981).

[25] N.J. Jerne, "Structural analogies between the immune system and the nervous system," in: Stability and Origin of Biological Information, I.R. Miller, ed. (Wiley, New York, 1975), pp. 201–204.

[26] A.J. Cunningham, "Some similarities in the way the immune system and nervous system work," in: The Immune System, C.M. Steinberg and I. Lefkovits, eds. (Karger, Basel 1981), pp. 43–50.

[27] G.W. Hoffman, "A neural network model based on the analogy with the immune system," J. Theoret. Biol., submitted.