Stephanie Forrest
Catherine Beauchemin

Computer immunology

Authors' address
Stephanie Forrest, Catherine Beauchemin
Department of Computer Science, University of New Mexico, Albuquerque, NM, USA

Correspondence to:
Stephanie Forrest
Department of Computer Science
University of New Mexico
MSC01 1130
Albuquerque, NM 87131, USA
Tel.: 505 277 7104
Fax: 505 277 6927
E-mail: forrest@cs.unm.edu

Acknowledgements
The authors thank G. Bezerra, D. Chao, R. De Boer, F. Koster, J. Mata, M. Moses, and A. Somayaji for their help with this manuscript. For many years the Adaptive Computation Lab at UNM has attracted an unusually talented and creative group of students, all of whom have contributed to the ideas described in this paper. Alan Perelson introduced S. F. to the complexities of immunology and contributed many of the important insights describe in this paper. S. F. gratefully acknowledges the support of the National Science Foundation (grants CCR-0331580, CCR-0311686 and CCF 0621900) and the Santa Fe Institute. C. B. acknowledges the UNM/LANL Joint Science and Technology Laboratory for her support. This publication was made possible by NIH Grant Number RR-1P20RR18754 from the Institutional Development Award (IDeA) Program of the National Center for Research Resources. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NIH or NSF.

Immunological Reviews 2007
Vol. 216: 176–197

Immunological Reviews
0105-2896

Summary: This review describes a body of work on computational immune systems that behave analogously to the natural immune system. These artificial immune systems (AIS) simulate the behavior of the natural immune system and in some cases have been used to solve practical engineering problems such as computer security. AIS have several strengths that can complement wet lab immunology. It is easier to conduct simulation experiments and to vary experimental conditions, for example, to rule out hypotheses; it is easier to isolate a single mechanism to test hypotheses about how it functions; agent-based models of the immune system can integrate data from several different experiments into a single *in silico* experimental system.

Keywords: *artificial immune system, agent-based models, in silico modeling, computational immunology*

Introduction

This article reviews a body of work that takes a synthetic or constructive approach to immunology, engineering artificial immune systems (AIS) in computational settings. In AIS, both the components of the immune system and their environments are defined as computations. In some cases, the AIS simulate immune system function in digital environments (as in all computer simulations), and in others they are practical solutions to real problems such as computer security. A common thread, however, is that hypotheses about components and mechanisms are expressed mechanistically as computer programs. When the programs are executed, their behavior is observed and, in the case of models, compared with the behavior of the real system. In practical applications, system behavior is evaluated by how well it solves the specified problem. The motivation is to engineer a system that can operate successfully in an environment with constraints similar to those faced by the natural immune system, thereby learning the functional significance of different components and analyzing how they interact with one another. The engineering process, like natural selection, leads to designs that are adapted to the constraints of their environments. Engineered systems can be studied and analyzed more easily than their biological

counterparts, in some cases revealing phenomena that would be difficult to discover experimentally.

In contrast to other theoretical models of immunology, AIS are usually constructed as agent-based models (ABM) (1). In ABM, entities in the model are represented explicitly. For example, each individual cell might be represented rather than each different cell type, as is common in other approaches such as differential equations. An essential feature of ABM is the ability to observe how behavior at different spatial and temporal scales arises from local mechanisms. This requires studying interactions among large numbers of components, and thus ABM exclude much biological detail by design. The trick is to define the model components at a proper level of abstraction, neither including irrelevant or incorrect detail nor leaving out essential features. The behavior and interactions of the entities in the model are encoded as computer programs. Consequently, experimental findings and hypotheses can be incorporated directly, even when they are not easily characterized as mathematical equations. The low-level components and interactions of an ABM are specified as programs, the simulation is run, and high-level behaviors are observed. For example, in a biological simulation, the high-level behavior might show how cell populations change over time. This is known as a bottom-up approach to modeling. In ABM, the simulation can be run repeatedly, with slightly different initial conditions, showing a distribution of outcomes rather than a single average behavior. For some immunological phenomena, this is relevant to understanding why some individuals become ill and others do not.

This review first gives an overview of ABM techniques as they are typically applied to immunology. It then describes representative examples of how ABM have been used to develop models of immunological phenomena. Next, it describes how similar methods have been used in engineering applications, where the abstractions and techniques that succeeded in modeling biology also work for computer security and other applications. Then, returning to natural immunology, it discusses examples of AIS applied to problems of biomedical significance. Finally, we speculate about the future prospects and the usefulness of engineered immune systems.

Agent-based modeling for immunology

In the past two decades, many methods have been used to model the immune system. Differential equation models are perhaps the most common, typically simulating how concentrations (cells, antibodies, cytokines, etc.) change over time and identifying critical parameters of an immune response (2, 3). Neural networks were used to model Jerne's immune network theory (4–6), and genetic algorithms were used to model the evolution of diversity (7, 8). The concept of 'shape space' (9–11), proposed as an abstraction of receptor/ligand binding, provided a convenient formalism for many subsequent models, and many immune system models still use some form of shape-space abstraction. More comprehensive and general immune system simulators use ABM techniques, incorporating significant amounts of immune detail (12–17).

In ABM the approach to immunology, each entity, or agent, represents a single cell or pathogen, and a computer program encodes its behavior and rules for interacting with other agents. Some common behaviors include cell death (usually by deleting the cell from the simulation), division (by making a copy of the dividing cell), or changing an internal state variable (for example, to model cell activation or differentiation). An agent in an ABM is a designated region of computer memory, similar to a variable, that contains details about the particular cell. This information can include its size, location, age, what receptors it has on its surface, and so forth. The agents can move through space, interacting locally with other agents at nearby locations, following a set of predefined rules. Thus, the behavior of the low-level agents is prespecified, and the simulation is run to observe global behaviors, such as determining an epidemic threshold. ABM specify local interactions in terms of simple mechanisms, which give rise to the large-scale complex dynamics of interest.

Why is ABM an appropriate method for studying immunology? First, the agent behaviors can directly incorporate biological knowledge or hypotheses about low-level components, even if they cannot be expressed mathematically. Second, data from multiple experiments can be combined into a single simulation, to test for consistency across experiments or to identify gaps in our knowledge. In the future, integrative methods such as ABM will probably be essential tools for comprehending and aggregating vast amounts of experimental data. Third, the immune system is a complex biological system with many different interacting mechanisms, and many biologically relevant values cannot be measured directly. For example, there are too many different protein/protein and virus/protein interactions to expect that we can isolate all of them experimentally. In an ABM, it is relatively easy to disable mechanisms altogether, adjust their relative contributions, and perform sensitivity testing of parameters. Through its functional specifications of cell behavior, ABM can thus help to bridge the current gap between intracellular descriptions and multicellular infection dynamics. Variation among individuals, each making different amounts of innate proteins that almost certainly impact the trajectory of an infection, is an important

complication. These effects can be studied using ABM by incorporating a distribution of parameter values in the agent population. Finally, there are important spatial and temporal interactions easily studied in ABM, for example, paracellular signaling between infected and uninfected cells.

Most ABM models of immunology represent receptors and ligands as character strings (*Fig. 1*) and use a string matching rule to model affinity. This clever idea was introduced by Farmer *et al.* (18) as a way to perform calculations for determining molecular complementarity and predicting the optimal size of an epitope. An overview of the calculation is given in Perelson and Weisbuch (19). The strings use an alphabet of m characters where each character corresponds, for example, to a given amino acid. In the most common case, however, $m = 2$, although larger alphabets have also been studied. In immunology, binding is a threshold effect consisting of two components: the affinity of a single receptor and ligand, and the total binding, or avidity of multiple binding pairs. Most models focus on affinity by simply counting the number of positions in the string where the symbols are identical. Many variations of this basic scheme have been proposed, including different-sized alphabets, different numbers of symbols in the string, and many string matching rules. Smith *et al.* (20) reviewed some of the variations. The strings, together with some internal state information (e.g. the age of the cell), are the 'agents' in an immune system ABM. *Fig. 1* illustrates this modeling strategy.

This fundamental modeling abstraction ignores nearly all of the physical details that determine receptor/ligand interactions. Careful modeling of a single interaction, say using a molecular dynamics simulation, is expensive computationally. By adopting character strings, many binding events can be simulated quickly, making it feasible to study large-scale properties of the immune system. Although character strings are unphysical, they can produce surprisingly accurate models when benchmarked to experiment (21), suggesting that the abstraction captures important features of receptor/ligand binding.

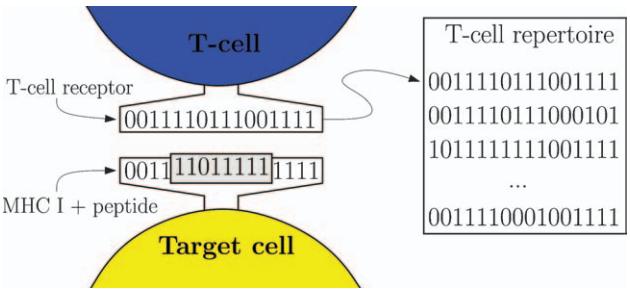

***Fig. 1.* Receptors and ligands modeled as strings**. Illustrated on the left is an example of a T-cell receptor binding to a MHC/peptide complex. Binding is modeled by a string matching rule, for example, by counting the number of positions in the string at which the symbols are complementary (known as Hamming distance). Repertoires are represented in the model as sets of strings, shown on the right.

Interactions between agents and between agents and their digital environment determine the dynamics of an ABM. In immune modeling, most interactions are mediated by receptor/ligand binding. So, when strings bind above the threshold value, the simulated cells may be stimulated to proliferate, increase their mutation rates, migrate to a new location, die, or secrete simulated molecules. Antigen can be added to the system in various locations, in varying doses, and at different times. The model is then 'run', and the dynamics of the infection are observed.

As computers became more powerful and less expensive, ABM became a practical method for studying complex systems such as the immune system. The following sections review a representative sample of conceptually important systems. The models were highly simplified and abstract in the beginning, but over time they gained sophistication as more was learned about the immune system, and advances in computation made it feasible to construct more complex models. Thus, the original models were one dimensional, and since then, there has been a progression to two- and now three-dimensional simulations.

However, more detail is not always desirable as it can be difficult to interpret results from an overly complex model. A simple model that isolates a few relevant phenomena so that they can be studied in detail is often more illuminating than an overly complicated one with many extraneous features. Thus, there is a tension between incorporating everything that is known and abstracting away from the physical details to capture general principles. The most compelling and influential ABM have been those where just enough detail was included to show a phenomenon of interest.

Immune system modeling with ABM

Early host–pathogen immune models consisted of simple one-dimensional binary networks of automata (22, 23). Each network contained a set of nodes, where each node represented a cell population (e.g. B cells or T cells) that could be in one of the two states: 0 = population is absent or 1 = present at high levels. The nodes contained rules specifying how connected populations would interact. For example, the presence of antigen might trigger a high level of B cells in neighboring populations. The networks were initialized by setting certain populations (nodes) to 0 or 1, and then simulating the network to find, for example, attractor states that could be interpreted in terms of an immune response. Perelson and Weisbuch (19) provide more details about these early models.

Early ABM of the immune system focused on immune network theory and were often implemented as cellular automata (CA) (24–26). Each site of the CA grid represented an idiotype or clone, and the state of the site represented the concentration of that particular clone. The dimensionality of the grid (e.g. one or two dimensional) represented the variable characteristics of the clone (e.g. geometric shape and electric charge), and the size of the grid represented the number of different possible values (e.g. the number of different shapes that were possible). The interaction rules specified that a clone situated at $\vec{x} = (x_1, x_2 \ldots x_N)$, where N is the dimensionality of the grid, could stimulate the proliferation of clones that were nearby, thus simulating the phenomenon of cross-reactivity. When these models were simulated, they produced complex patterns resembling immune activity, with stable patterns corresponding to memory, and changing patterns attributed to perturbations caused by new antigens. A variation of this approach represented the network of idiotypic interactions using a classifier system, in which each classifier rule specified a particular interaction, and its strength represented the concentration of the idiotype (18). As interest in Jerne's network theory waned, so did the use of such models.

The next generation of computational immune models was more ambitious, incorporating significantly more immunological detail (13, 27, 28). The CA grid was used to represent physical space, rather than abstract properties of clones. The simulators incorporated enough detail that one model could be used to study several aspects of immune dynamics or disease.

In the following, we discuss two examples of CA systems, IMMSIM and ma_immune. We then describe two three-dimensional models that incorporate molecular modeling as well as cellular modeling, Simmune and CyCells. Finally, we describe more recent work that emphasizes graphical visualization of immunological processes. IMMSIM is a canonical example of the ABM approach applied to immunology, so we describe it in more detail than the other systems. These general models are useful on their own as a means of organizing specific hypothesized mechanisms and studying how they interconnect – the value of this synthesis is shown by the many extensions developed for specific purposes.

IMMSIM

An early CA model of the immune system introduced in 1992 is IMMSIM (13, 27). The original version modeled the humoral response (13) and contained bit string representations of T cells, B cells, and other antigen-presenting cells (APC), as well as antigen and antibody molecules. Later versions added cytotoxic and helper T cells, epithelial cells, and cellular responses (29, 30).

The original version of IMMSIM was written in APL2 with the IBM APL2 runtime environment. Bernaschi and Castiglione later developed a parallel version called ParImm and later CIMMSIM (15). Finally, a C^{++} tutorial version of IMMSIM based on CIMMSIM was developed by Kleinstein and Seiden (14).

Most simulations were run on small grids, typically a 15×15 hexagonal grid, and the size of the bit string, N, was typically eight, yielding $2^8 = 256$ different possible clones. The time scale of a simulation is not always specified, but Kleinstein and Seiden (14) suggest that one time step in the simulation corresponded roughly to the time for a single B-cell division.

Representing receptor-specific interaction

Fig. 2 illustrates the schematics of cells and molecules in an IMMSIM model using eight-bit strings. As in most AIS, immune components are characterized by their receptors, and each receptor is represented as a character string. Typically (and this differs among the various implementations of IMMSIM), a B cell is composed of a single receptor and a single major histocompatibility complex class II (MHC II) molecule, each represented by a binary string. Among the B-cell population, for example, the simulations typically consider only one or two different MHC II types. Antibody molecules are represented as a single receptor. Finally, antigen molecules consist of segments of two different types: B-cell epitopes and presentable peptide strings.

Interaction rules

Interactions between agents are specified by a set of interaction rules. For example, a B-cell receptor interacts with the 'bare' part of an antigen. A T-cell receptor interacts with the pair made

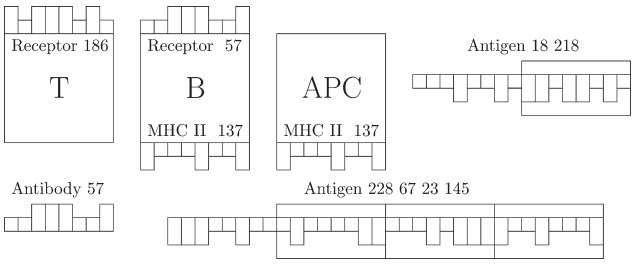

***Fig.* 2. Schematic representation of IMMSIM**. The figure depicts T cells, B cells and other APCs, antibodies, and antigen molecules. On antigen, epitopes are shown exposed and the presentable peptides are boxed. Receptors, MHCs, epitopes, and peptides are numbered according to the decimal value of their eight-bit string. For example, the B cell's receptor 57 is represented as (00111001), with zeroes and ones depicted as short and long blocks, respectively. Adapted from Seiden and Celada (13).

up of an antigen peptide and the MHC II of a B cell. To determine binding, the receptor strings are compared symbol by symbol, looking for mismatches, using a variant of Hamming distance. Two bits match if they are complementary. Thus, 0 matches 1 and *vice versa*. *Fig.* 2 illustrates this approach, where there is a two-bit mismatch between the B-cell receptor 57 (00111001) and the epitope (bare part) of antigen 228 (11100100). If the number of matches is above the binding threshold, which is a parameter of the simulation, then the agents interact.

On any given step of the simulation, the set of potential interactions for each agent (a cell or antigen) is determined, and out of this set, one action is chosen for each agent probabilistically. Then, each agent's state (naive or activated) is updated synchronously. Possible actions include cell death, cell division, and antibody production. If, for example, an antigen–antibody interaction is successful, they are considered to have formed a complex, and both are removed from the simulation. Finally, the entities diffuse to a randomly chosen neighboring grid site and that concludes one time step of the simulation.

IMMSIM studies

The IMMSIM models were used to investigate immunological phenomena such as affinity maturation and hypermutation in the humoral response (12), the rheumatoid factor paradox (31), transitions between immune and disease states and the relative contributions from the different branches of the immune system (29, 30), vaccine efficiency (32), and the dynamics of human immunodeficiency virus (HIV) infection (33). IMMSIM was a conceptually important advance, because it developed a general modeling framework that could be used for multiple studies. It incorporated enough immunological detail to support studies involving real immunological problems. IMMSIM also illustrates the use of bit string representations of receptors and ligands.

ma_immune

A more recent example of CA used for modeling immunology is `ma_immune` (34). `ma_immune` is implemented on a two-dimensional grid, representing a tissue that is patrolled by generic immune cells. `ma_immune` was designed as a simulation platform for localized tissue infection, where the cells affected by pathogen are immobile, tightly packed, and the infection spreads to immediate neighbors. The simulation considers two cell types: tissue cells that are immobile, and generic immune cells that move randomly to neighboring locations. The simulation platform, called `ma_immune`, together with the supporting visualization software MASYV, is documented and freely available (35).

The model was used to study how the spatial distribution of agents affects the dynamics of an infection (36), something that is difficult to assess in a differential equation model. Differential equation models normally assume that populations (target cells, infected cells, virions, etc.) are uniformly distributed in space. Consequently, the rate at which target cells become infected, for example, is proportional to the total abundance of target cells and virions, without regard for the spatial localization of the target cells and virions. Beauchemin (36) showed that grouping the initially infected cells into patches rather than distributing them uniformly on the grid reduced the infection rate, because only cells on the perimeter of the patch have healthy neighbors to infect. This approach yielded a better fit to experimental influenza A infection data than the equivalent non-spatial model. `ma_immune` is conceptually important, because it isolates the effect of spatial localization and provides an elegant explanation of how spatial localization can change infection dynamics.

Multipurpose modeling frameworks

Two recent modeling frameworks, Simmune and CyCells, are significantly more general than earlier systems and represent a conceptual advance in immune system modeling. Simmune is a two-level immune system simulator (17, 37, 38). At the lower level, molecules such as cytokines are defined as continuous quantities, and their dynamics are modeled using differential equations. At the higher level, cells are modeled as discrete computational agents. Thus, Simmune is a hybrid of continuous and ABM techniques. The basic framework is defined generally enough that it could in principle model almost any kind of cell population.

Different types of cells can be defined by the user (e.g. T cells and B cells), and the user specifies rules for how cells move between locations on the grid. Because molecules are represented as continuous quantities, they move using diffusion rules, whose parameters are also specified by the user. Simmune is run on a three-dimensional grid. The user defines different compartments (e.g. lymph nodes and thymus) and specifies properties for each compartment within the simulation such as its dimension, diffusion rates for each molecular type, which types of cells are in each compartment, and their initial concentrations. The exchange of agents between the different compartments can be regulated, for example, which kinds of agents are allowed to pass from one compartment to another and at what rate. In Simmune, a cell's action depends on the

stimuli it senses from its environment, known as a cellular stimulus response mechanism.

CyCells is a similar modeling framework, but its functions and usage are better documented (39) and its source code is available online (40). Similar to Simmune, it represents molecular concentrations continuously and cells discretely. CyCells is implemented on a three-dimensional square grid. In CyCells, models are defined by specifying initial numbers of cells, cell types (e.g. B cells and macrophages), and molecular signals (e.g. cytokines). Each cell of each cell type is represented explicitly, and the molecular signals are represented as real-valued concentrations at each site. For each type of molecular signal, the modeler supplies a decay and/or diffusion rate. CyCells uses the 'sense-process-act' abstraction for cell behavior. For each cell type, the modeler specifies its attributes and associates with it sensing, processing, and/or action procedures. An attribute might contain the cell's diameter, a sensing procedure could respond to a particular molecular concentration, the processing function might specify that the cell is activated when it senses the molecular concentration above a threshold, and the action could be death, division, migration, or the secretion of a molecular substance. Hence, the framework is highly flexible, allowing both simple and complex models to be implemented in a single framework.

Warrender *et al.* (16) used CyCells to investigate two hypotheses about the maintenance of peripheral macrophage population sizes in the lung. Under the first hypothesis, macrophage proliferation was local and caused by the division of resident macrophages. Under the second hypothesis, proliferation was the result of influx of circulating blood monocytes. Although either scenario was plausible, the model showed that the influx-driven system is inherently more stable and that a proliferation-driven system requires lower cell death and efflux rates than an influx-driven model. CyCells was also used to model early infection dynamics of *Mycobacterium tuberculosis* (Mtb) bacteria (41).

These models were conceptually important, because they introduced more realistic treatment of cytokines and the other molecular players in the immune response. By combining continuous models of molecular diffusion on a grid with ABM of cells, these hybrid models represented an important step in ABM approaches to immune modeling. Other hybrid models combine continuous and ABM methods in different ways, for example, as in Chao *et al.* (42), where a cytotoxic T-cell life cycle is divided into stages, and all individuals in a given stage are assumed to be identical. A single integer represents the individuals in a given stage, rather than one data structure per individual required in a pure ABM approach.

Representing large repertoires in immune system models

Running an ABM that explicitly represents a realistic number of cells can be computationally expensive. An estimated 10^7 unique clones comprise an individual's B-cell (43) and T-cell (44) repertoires. Simulations containing this many clones have become feasible with recent increases in computing power (15, 45), although running simulations of this size is still time-consuming. Thus, most immunology simulations use artificially small immune cell repertoires that contain tens or hundreds of clones. This number is sufficient for studying some immunological phenomena but not sufficient to address issues such as cross-reactivity (46, 47) and alloreactivity (48) quantitatively. Because most cells in the repertoire are not involved in any given immune response, only a small fraction of cells needs to be updated on any given time step. This observation led to the use of 'lazy evaluation', a computational technique in which only the computations that need to be performed are actually carried out (49). Rather than create, for example, 10^7 explicit B-cell clones, the majority of which would not respond to a given infection, the lazy evaluation version of an ABM would not create any clones until the simulated infection began, then it would produce only the 10^2-10^3 that have sufficient affinity to participate in the given immune response. Although using lazy evaluation complicates the software implementation, it can reduce the memory and running time by orders of magnitude without affecting simulation results. This strategy has been applied to B-cell (49) and T-cell (42) models.

Statecharts and visualization

Efroni *et al.* (50) developed an immune system programming framework based on the visual language of Statecharts and describe the structure of a statechart model for the thymus. Statecharts, introduced by Harel in 1987 (51), are a method for representing complex computational processes in terms of states and the events that cause transitions between states. It also considers substates and orthogonal states. For example, a cell's orthogonal states could be its expression of different receptors, and the substates could be the conditions or substates under which the cell expresses a particular receptor. Although the terminology and motivation are different from that of agent-based or CA models, they share many features, and we discuss them briefly.

Agents in the model are the moving thymocytes and the stationary epithelial cells of the thymus. A thymocyte's motion depends on the gradient of the various cytokines and the

thymocyte's expression of the markers that detect these gradients. The simulation consists of four types of cytokines and seven thymocyte markers, five of which have binary values (expressed or unexpressed) and two of which have three values (expressed, unexpressed, and an intermediate low level of expression). Then, for example, based on which of the $2^5 \times 3^2 = 288$ different possible marker states the thymocyte is in, it is sensitive to a particular set of cytokines, and its movement is determined by the cytokine set.

The diagrammatic representation of statecharts is intended to be easier to understand for people from various disciplines, thus facilitating collaboration between modelers and experimentalists. In addition, the model's interactive graphical user interface allows the user to see the agents move and interact, navigate the simulation by clicking on particular agents, and either retrieve or set information about the agent's state and decision process (*Fig.* 3). Because immunological knowledge is often incomplete, the model lets the user define different hypotheses for the outcomes of interactions and choose which instance of the available hypotheses will be executed on a given run, thus supporting exploration of different hypotheses.

An even more sophisticated visualization tool is PathSim (Pathogen Simulation), a simulator developed for displaying three-dimensional-anatomical models of host–pathogen interactions (52). PathSim's programming framework is described briefly in Polys *et al.* (53). As more detailed information becomes available about how individual immune cells move through tissue (54–56), visualization packages such as the Statechart system and PathSim will probably play a more central role in immunological modeling.

Engineering an immune system

This section examines a body of work that translated the mechanisms and organizational principles of the immune system into algorithms for solving computational problems (57, 58). It is surprising that the abstractions and concepts discovered through immune system modeling are general enough to form the basis of non-biological engineered systems. However, AIS methods have been applied to a wide range of problems, including control engineering, robotics scheduling, fault tolerance, and bioinformatics. The most prevalent example

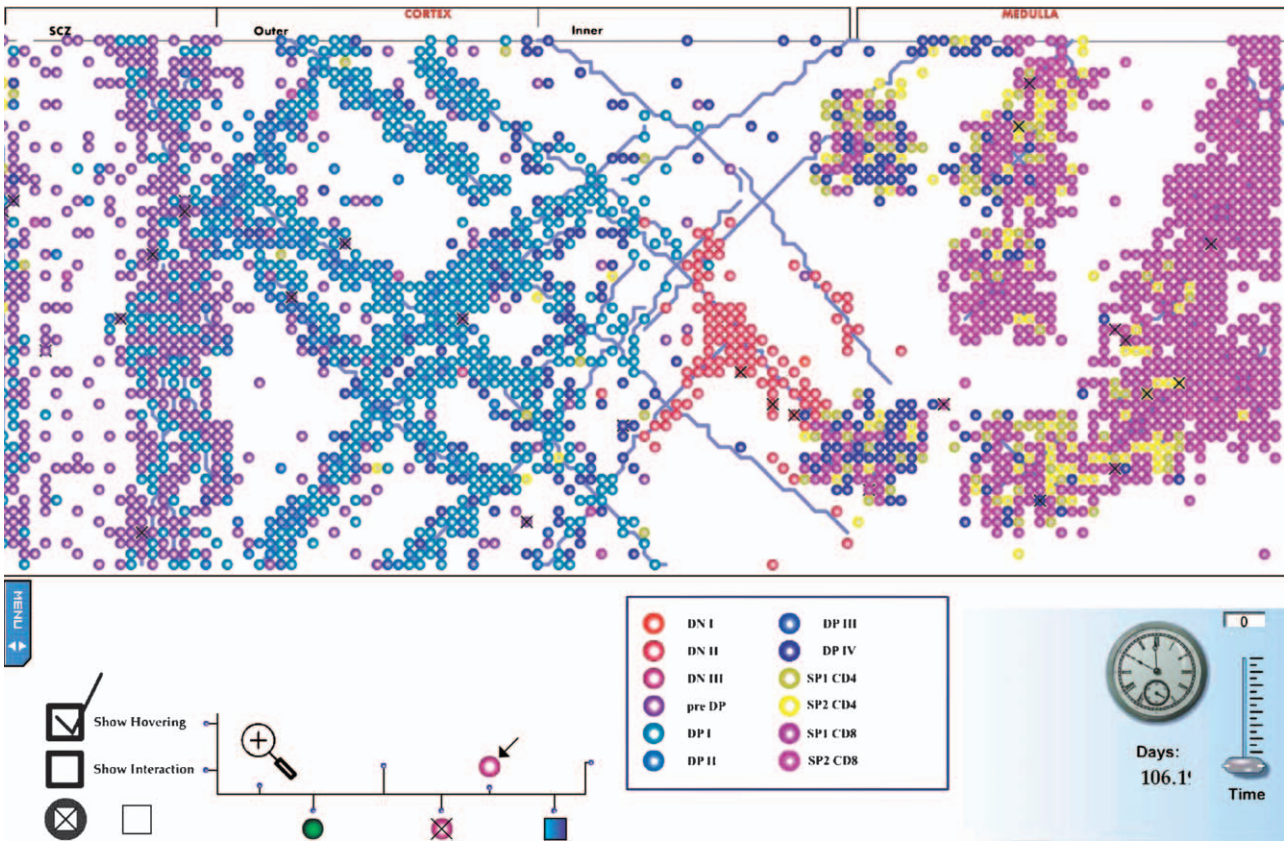

Fig. 3. A screenshot of the Statecharts graphical interface during execution. Figure reproduced with permission from Efroni *et al.* Genome Research, copyright 2003, Cold Spring Harbor Laboratory Press (50).

has been in computer security, and we focus on that example in the following subsections.

A computer security system should protect a computer or network of computers from unauthorized intruders, which is analogous in functionality to the immune system protecting the body from invasion by foreign pathogen. Further, a computer security system should protect against insider attacks, malfunctioning software (analogous to misbehaving cells), and other internal errors, maintaining the computer within normal operating tolerances. The similarities between the computer security problem and the problem of protecting a body against damage from internally and externally generated threats are compelling, and they were recognized as early as 1987, when the term computer virus was introduced by Adelman (59). Later, Spafford (60) argued that computer viruses are a form of artificial life, and several authors (61–64) investigated the analogy between epidemiology and the spread of computer viruses across networks. The connection to immunology then was made explicitly (65, 66), and since that time the ideas have been extended to incorporate significant amounts of immunology and to tackle ambitious computer security problems, including computer virus detection (65, 66), spam filtering (67), and computer forensics (68).

Many different aspects of the immune system have been used as inspiration for engineering applications. In addition to the examples given below, there has been work on danger theory (69), idiotypic networks (70–72), affinity maturation through somatic mutation (73, 74), V-region libraries (75, 76), and the innate immune system (77, 78). Nearly all of the applications exploit analogies with the pattern matching and learning mechanisms of the immune system to perform desired computations. The studies illustrate how immunological processes of interest can be defined computationally and studied in detail to understand their functional significance. As discussed in earlier sections, most computational realizations of immunology focus on the adaptive immune response, and that is true in the computer security domain as well, although some recent work emphasizes the innate response.

Elements of a computer immune system

If we set out to engineer a computer immune system to solve problems in computer security, what functional components would we need? Similar to the natural immune system, there are computer defenses that correspond to non-specific and specific responses. Firewalls evolved to prevent unwanted communication between computer networks; access controls (e.g. logins and passwords) were developed to control how much access users have to computers and data. These generic defenses correspond to the immune system's non-specific response. Specific responses, known as intrusion-detection systems (IDS), recognize active intrusions, including those that may not have been seen before. An IDS continuously monitors the dynamic behavior of a computer system to determine if a security violation or denial-of-service attack has occurred. Such violations include injected foreign code (as in the case of viruses) or exploitation of vulnerabilities in existing code by illegitimate users. There are two broad classes of IDS, loosely corresponding to primary and secondary responses in natural immunology. Similar to the primary response, 'anomaly' IDS can detect novel forms of attack, while signature detection systems respond only to known attacks, corresponding to a secondary response. Finally, some systems have an automated response component that corresponds to the effector side of the immune system, such as inflammatory processes or cytotoxic T cells.

In earlier sections, we saw how immune systems are modeled using computational agents, with receptors and ligands represented as short strings of symbols, death implemented by deleting agents from the simulation, and proliferation and cell division implemented by copying agents. These abstractions are used in the computer security domain as well, but additional decisions must be made. These decisions include identifying what data will be observed to detect infection (e.g. what corresponds to normal peptides of the body), devising a scheme for generating a diverse repertoire of detectors, specifying the details of the adaptive response (how will new infections be noticed and remembered), and determining what actions the immune system will take to control infections once they have been identified. In the following, we review each of these elements briefly.

Defining self

Protecting computers involves activities such as detecting unauthorized use of computer accounts, maintaining the integrity of data files, mitigating denial-of-service attacks, and detecting and eliminating computer viruses and spyware. These activities can be viewed as instances of the more general problem of distinguishing self (legitimate users, uncorrupted data, etc.) from dangerous non-self (unauthorized users, viruses, and other malicious agents). Just as the natural immune system evolved to monitor certain observables in the body, notably peptides, so must AIS be designed to monitor particular aspects of a computer.

Thus, the first step in designing a computer immune system is deciding what data or activity patterns will play the role of self

and what entities will correspond to pathogens. Despite debate in the immunological literature about the role of self, we accept the notion that proteins and peptides are a fundamental unit of recognition for the immune system. To build a computer immune system, a computational analog to the protein must be found. Researchers typically make this choice with regard to a particular class of threats in which they are interested. As the threats of interest have evolved, so have the computer immune systems that protect against them.

When computer immune systems were introduced, the primary threats were computer viruses that infected user files (file infector viruses) or the code used to boot up the operating system (boot sector viruses). The viruses consisted of short sections of computer code attached to another program. When the host program ran (was executed), control passed to the virus code, which searched for other uninfected files and copied itself into them. Consequently, early systems tried to protect the integrity of programs stored on disk, either by preprogrammed databases of virus signatures (as in popular anti-viral software) or by change-detection programs (65, 66). Change-detection programs are analogous to a biological system that checks for genetic damage, noticing changes to the genetic codes that control biological processes. This approach is limited, because most of the data and code stored on computer disks are never used. Because computer viruses do not cause damage until they are executed, it is more efficient to protect executing programs or data being copied across a network.

The second generation of computer immune systems explored the possibility of protecting executing programs (79, 80). Here, the unit of protection (organism) is a single computer, and each executing process is roughly analogous to a cell. Discrimination between normal and abnormal behavior is based on what functions (or subroutines) are normally invoked by the running program, in particular, the requests issued to the operating system from the running program, known as system calls (79, 81). As a single program executes, it might make several million system calls in a short period of time, and this signature of normal behavior is sufficient to distinguish between normal behavior and many attacks. A record is kept of the system-call history, and the list is split up into shorter 'peptides' (typically 6–10 system calls long), which define the normal behavior of the program. Most attacks trick the victim program into executing infrequently used code paths, which in turn leads to anomalous patterns of system calls. This approach defends against intrusions that target a single computer, the most prevalent example being the buffer overflow attack (82). This more dynamic approach resembles the kind of 'run-time' checking performed by RNA interference in cells that are actively translating genetic information into proteins.

A large number of researchers adopted the system-call approach, some seeking to improve on the original methods (83–87), some applying its method to other problems (88–90), and some attempting to defeat the system (91). Sana Security developed a product known as 'Primary Response' based on this technology and is actively marketing it to protect servers. At this writing, the system-call method is the most mature application of the immunology analogy to computer security.

As the Internet expanded and information exchange became routine between computers around the world, protecting against widespread network attacks such as e-mail viruses and worms became more important. We use the term 'virus' to refer to malicious software that requires help from computer users to spread to other computers. E-mail viruses, for instance, require someone to read an e-mail message or open an attached file in order to spread. We use the term 'worm' for infections that spread without user intervention. Because they spread unaided, worms can often spread much faster than viruses.

Immunological mechanisms have been employed to protect computer networks (92). The equivalent of an organism is a local area network (LAN) of computers. Transmission control protocol over internet protocol (TCP/IP) is the most common communication protocol used to connect computers, and the behavior of the protected system can be characterized by its normally occurring TCP/IP connections (93). The connection is represented by the source IP address, the destination IP address, and the program used to make the connection (represented as an eight-bit number). This information specifies a network connection. The analog of a peptide is a binary string representing the connection (by concatenating the source, the destination, and program type) (*Fig.* 4). All normally observed and acceptable connections, both those within the LAN and those connecting the outside world to the LAN, form the set of self-patterns, and all others (potentially an enormous number) form the set of non-self patterns.

To summarize, there has been an evolution of threats, which has forced a progression from methods that protect the integrity of a computer program, to methods that detect when an executing program behaves abnormally, to more recent methods that protect networks of computers. All of these levels of protection are important, and the progression has led to dynamical definitions of self that are quite different from those taken by traditional anti-virus software, which looks for specific patterns in files stored on hard disks. The distinction is roughly analogous to that between gene products and genes themselves.

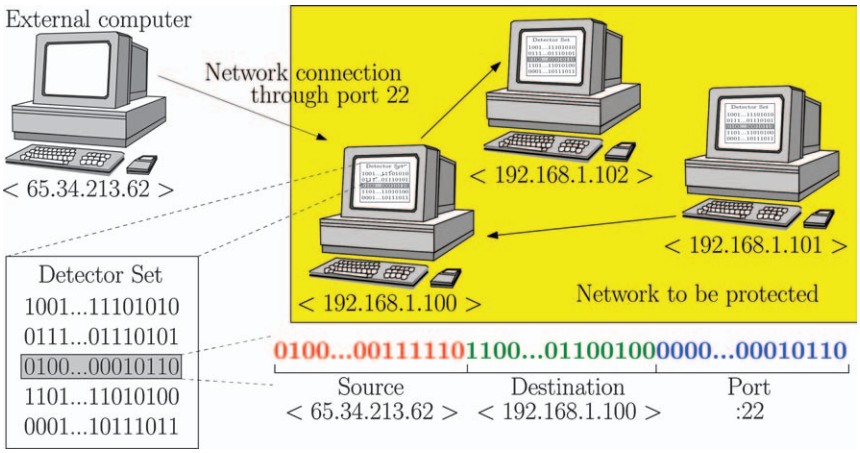

Fig. 4. Architecture of the LISYS intrusion-detection system (92). The shaded area shows the LAN of computers to be protected, although the network may have connections with external computers. Normally occurring connections between computers are indicated by directed arrows. Each computer in the LAN contains its own detector set (collection of immune cells). Each detector represents a single connection consisting of the source computer for the connection, the destination computer, and the port number for the program that initiates the connection. Binding between detectors and new connections is measured using the r-contiguous bits matching rule (10) (not shown).

If the immune system had evolved to inspect directly the genomes of all cells for irregularities, we would have a system more closely analogous to anti-viral software. Instead, the immune system monitors gene products. Anti-virus software has recently adopted several features analogous to the innate immune system (detecting general patterns that are harmful), but most commercial products do not yet have the adaptive immune system's ability to address novel threats.

Negative and positive selection

With any of the above schemes, a strategy is needed for generating the immune cells (called detectors) that detect abnormalities. Similar to the natural immune system, this can be achieved with either positive or negative representations (analogous to positive and negative selection of T cells) (94). The negative selection algorithm (65, 95) is based on one aspect of the multistage maturational process of T cells in the thymus, where they are censored against the body's normally occurring peptide patterns. T cells that react too strongly with self are deleted before they can become active and cause autoimmunity. Although mature T cells have survived at least two other censoring operations, genetic rearrangements and positive selection, we focus on the negative selection aspect here.

The translation of this process into an algorithm for computers is straightforward. First, we assume that the anomaly detection problem is posed as a set *RS* (real-self) of strings *s*, all of fixed-length *l* of which we can access only a sample *S* at any given time. The universe of all *l*-length strings is referred to as *U*, and the set of anomalous patterns to be detected is the set *U–RS*. Candidate detectors (strings) are generated randomly and censored against *S*; those that fail to match the strings in *S* (analogous to expressed peptides in the thymus) are retained as active detectors. Such detectors are known as 'negative detectors', and if *S* is a good sample of *RS*, each negative detector will cover (match) a subset of non-self without matching self. By generating sufficient numbers of independent detectors, good coverage of the non-self set is obtained. *Fig.* 5 shows the relationship of these sets.

Since its introduction in Forrest *et al.* (65), interest in negative detection has continued, especially for applications where noticing anomalous patterns is important, including computer virus detection (96, 97), intrusion detection (92, 98–100), and industrial milling operations (101). Recently, other categories of applications have been proposed, including color image classification (102), collaborative filtering and evolutionary design (103), and privacy (104, 105).

Analysis of the computational properties of the negative selection algorithm and its descendents showed several advantages (95, 106). One advantage is that negative representations require virtually no communication among the individual detectors. Negative selection is thus an important mechanism, because it allows detection to be distributed rather than

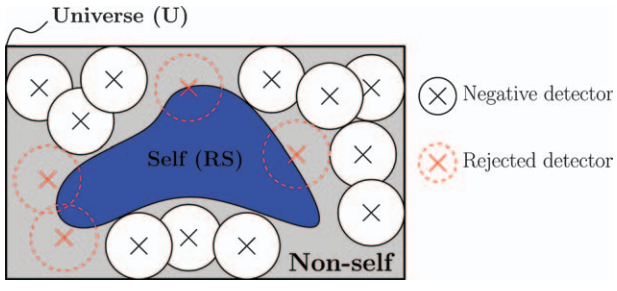

Fig. 5. Self–non-self discrimination. A universe U of data points inside the black border is partitioned into two sets: self (shown in blue) and non-self (everything else). The points in the space correspond to features of the problem domain, for example, fragments of computer code or individual network connections. Negative detectors analogous to T cells are generated randomly. Those that overlap with self (shown in red) are deleted by the negative selection algorithm. This leaves a set of detectors (shown in black) that collectively cover most of non-self.

centrally controlled. Once censored by the negative-selection process, each detector can function independently of other detectors. This independence is because each detector covers part of non-self. Thus, a set of detectors can be split up over multiple sites, reducing the coverage at any given site but providing good system-wide coverage. To achieve similar coverage using detectors that match self would be computationally inefficient, because each positive detector would have to check with all other positive detectors to confirm a mismatch. This property allows several forms of distributed processing: checking small sections of a large object independently, several independent detector sets (*Fig.* 4), or independent evaluation of each detector in a single detector set. Indeed, the natural immune system uses negative detection in a massively distributed environment – the body.

A second advantage of negative representations is the ability to conceal information about the system it is protecting (104, 107). Thus, in some negative representations, it is provably difficult to infer the original (positive) set from the negative information alone. This is analogous to the problem of trying to infer all of the normal self-proteins in the body simply by inspecting all of the mature lymphocytes. Although not necessarily important for the natural immune system, it may have application in computing for enhancing the privacy of sensitive information.

Affinity maturation

The immune system uses affinity maturation to evolve B cells that respond to foreign antigen, and so do some computer immune systems. 'Clonal selection' algorithms automatically construct detectors tailored to observed patterns (108). This is a more focused learning process than that achieved through negative selection alone. This approach has been applied to intrusion detection (109), where the clonal selection algorithm mimics the processes of somatic mutation and proliferation, evolving detectors toward non-self network patterns. In this work, detectors were defined as IF/THEN rules, which classified new patterns either as normal or abnormal. The antecedent (IF part) of each rule specified the conditions to be tested, and the consequent (THEN) described the class label (normal or abnormal) assigned to the rule.

The algorithm proceeds as follows. First, a set of detector rules is generated randomly, and each rule's 'fitness' is initialized to 0. A sample of rules is randomly selected from the initial set. Each detector (rule) in the sample is tested against an existing corpus of non-self patterns, and the detector in the sample with the highest score has its fitness increased (110). This process is repeated (with new random samples) until each detector in the population is evaluated in several contexts. Then the genetic algorithm (111) operators of mutation and crossover are applied to the more fit individuals to produce new candidate detectors. Finally, the new candidate detectors undergo negative selection against a corpus of known self-patterns to prevent autoimmunity (false positives). Detectors that pass this test then enter the population, replacing an existing detector. The algorithm departs from known biology in two ways: first, by using recombination in an algorithm mimicking somatic mutation and second, by applying negative selection after somatic mutation. However, it is effective at evolving good sets of detectors, and it has been applied to several different problems.

Controlling autoimmunity

Early experiments with simple AIS for computer network security showed high false-positive rates (112), corresponding to autoimmunity. False positives have been a persistent problem for anomaly IDS, preventing their widespread adoption for computer security. Within the computer immunology framework, several mechanisms have been introduced for controlling autoimmunity that are similar in spirit to immunological processes. Two examples are avidity and second signaling, described below.

A single anomalous connection in the network does not necessarily signal an attack, so a parameter known as the 'activation threshold' was introduced (92). A detector is required to accumulate enough matches to exceed the activation threshold to become active. This simple modification reduced autoimmunity significantly, and of the various methods that were tested for controlling autoimmunity, it had the most dramatic effect. Activation thresholds are analogous to avidity in the natural immune system, where multiple receptors on the lymphocyte must be bound simultaneously in order for it to become activated. Here, the integration of signals takes place over time instead of in space.

Hofmeyr and Forrest (92) introduced a mechanism similar to the costimulation that a B cell must receive from a T-helper cell. Originally, the second costimulatory signal was provided by a human observer, although Begnum and Burgess (113) provide an interesting extension. In the original system, a detector that survived negative selection became mature and was matched against all new connections in the network. If it matched enough connections to exceed the activation threshold, it was activated. However, if an activated detector did not receive a costimulatory signal within a given period (typically 24 h), it died and was

deleted. If it received costimulation, it entered a competition to become a memory detector. Once a memory detector, it lived indefinitely, requiring only a single match for activation, thus capturing the idea of a secondary response.

Costimulation automatically eliminates detectors responsible for false positives, and a human must intervene to confirm true positives. This process allows adaptation to incomplete or evolving definitions of self, both of which are common in the intrusion-detection domain. It also allows for shorter maturation periods in the negative selection phase, by providing a backup in case the maturing detector does not encounter all possible self-patterns. Costimulation in the natural immune system presumably has similar benefits, as well as protecting against inappropriate somatic mutations. The costimulation process is inefficient computationally, and the ideal system from an engineering point of view is one where most tolerization is centralized, and peripheral tolerization addresses issues of perpetual novelty, incomplete descriptions of self, and somatic mutation. There have been debates in immunology about the relative roles of tolerization in the thymus and peripheral tolerization; our experiences suggest that both are essential, as they play complementary roles.

MHC and diversity

Generalization is an important tool in resource-limited environments. If each detector can match multiple non-self patterns, fewer detectors are needed. However, generalization introduces potential discrimination errors, especially those caused by overgeneralization, in which foreign patterns that resemble self are categorized as normal. These are known as false negatives. As the generality of detectors increases (and specificity decreases), the potential for overgeneralization also increases.

Using a diverse set of representations for detectors proved to be effective at reducing the overall number of false negatives (92). Representation diversity was achieved by permuting the bits in a detector, thus moving some co-located bits away from one another and placing others together. Because detector matching was based on adjacent matches in the string (10), the different permutations effectively changed the structure of the self-set for each detector. Consequently, where one detector failed to detect a pattern, a permuted version of it might succeed. This strategy was particularly effective at reducing the number of false negatives when the non-self patterns were similar to self-patterns.

The immune system also has limited resources, and it appears to use both generalization and diversity. Generalization is a consequence of the fact that a monoclonal lymphocyte can bind to a set of structurally similar peptides. It is not unreasonable to assume that generalized detection also leads to coverage gaps (false negatives), and if so, pathogens would evolve away from detection toward the gaps. We speculate that each different MHC allele can be regarded as a different way of presenting a protein (depending on which peptides it presents). Hence, varying the MHC varies which coverage gaps exist. This idea is illustrated by the existence of diseases, such as leprosy, that are strongly affected by MHC types.

Effectors

The previous subsections outlined a mature and growing body of work defining computational abstractions that correspond to the immune system's ability to detect and remember foreign pathogens. Much less effort has been devoted to the effector arm of the immune response. Early work on immune-based approaches to control and robotics (71, 114) incorporated effectors, but most engineering applications emphasize detection. This lacuna was emphasized by several publications (115–117), which hypothesized a complex feedback control system controlled by cytokine signaling. This is an exciting and important direction for future work in computer immunology.

In computer security applications to date, most approaches emphasize low-level generic actions (e.g. homeostasis), rather than targeted killing (e.g. cytotoxic T lymphocytes) or repair. In the first such example, a system called 'process Homeostasis' (pH) was developed for a single computer (80). In pH, the computer autonomously monitors its own activities at the system-call level (described above), making small corrections to maintain itself in a normal state. In particular, when an executing program is determined to be behaving abnormally (by the detection apparatus), pH slows down the program by delaying the execution of system calls. The more anomalies detected, the more aggressively pH slows the program down. This graduated response has the advantage that small delays (possibly corresponding to false positives) are imperceptible to users, while long delays trigger timeout mechanisms at network and application levels, effectively killing the misbehaving process.

This basic approach of 'throttling' misbehaving systems was extended to computer networks and was used for controlling the spread of e-mail viruses and worms (118, 119). In virus throttling, a hard limit is placed on the rate at which a single computer can initiate new connections to other computers. When the limit is exceeded, new connection requests are simply dropped. This simple effector mechanism was integrated with immune-like detectors that could discriminate between different classes of connections (e.g. web requests and e-mail

messages), adaptively setting the appropriate threshold for each detector (120).

Putting it all together

The previous subsections reviewed computational analogs of several immunological processes and described how they contribute to computer security applications. The computational mechanisms are crude compared with the details of natural biology, and in many cases it is only the highest level concepts that have been borrowed. Negative selection allows the system to operate without central coordination, affinity maturation provides a directed learning process, avidity and costimulation help control autoimmunity, representation diversity (analogous to allelic diversity in MHC) compensates for gaps in coverage (false negatives), and homeostatic mechanisms help the system to achieve a graduated response to perturbations. To date, however, no system incorporates all of these elements into a single functioning system. Some of the more comprehensive implementations include LISYS (92), CDIS (97, 121), pH (80), and more recently, a system developed for mobile *ad hoc* networks (100), although this latter system was tested only in simulation. *Fig.* 6 illustrates how the components were combined in the LISYS system using the setup shown in *Fig.* 4.

Applications to biomedicine

ABM methods for modeling immunology were described above. This section describes how these methods have been used to study immune responses to specific infections: HIV, tuberculosis, influenza, and the primary immune response in lymph nodes.

Models of HIV

HIV has been modeled extensively using mathematical methods (3), and there have been several ABM as well, including the two CA models of HIV in lymph nodes described below.

Modeling multiple time scales

Zorzenon dos Santos and Coutinho (122) introduced a CA model that reproduces the two time scales of an HIV infection: the short time scale (weeks) associated with the primary response and the long one (years) associated with the clinical latency period and the onset of acquired immunodeficiency syndrome (AIDS) (122). The studies suggest that mean-field ordinary differential equation (ODE) models fail to reproduce the two-scale dynamics of HIV, because the initial immune response in the lymphoid organs is localized. Experimental results illustrating the two-scale dynamics of HIV along with results from CA simulations are reproduced in *Fig.* 7.

In the model, each site in the two-dimensional square grid represents a target cell for HIV, namely a $CD4^+$ T cell or a monocyte. A cell's neighbors consist of the eight adjacent cells. Each target cell can be in one of four states: healthy, infected A1, infected A2, or dead. An infected-A1 cell corresponds to an infected cell that is capable of spreading the infection. An infected-A2 cell corresponds to an infected cell in its final stage before apoptosis. Infected-A2 cells can infect healthy cells only when other infected-A2 cells are present in sufficient concen-

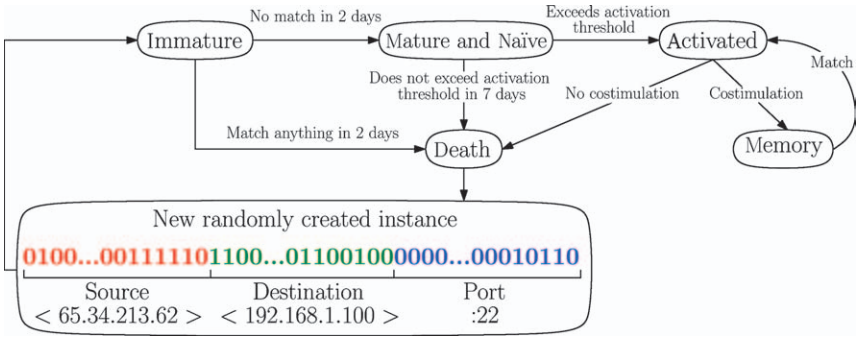

***Fig.* 6. Detector life cycle in a computer immune system.** Detectors corresponding to network connections are generated with random bit patterns. Each detector is immature for two days while it undergoes negative selection (analogous to T cells in the thymus). During this time, it is matched against all new network connections, and if it matches even a single connection, it is deleted and replaced by a new randomly created detector. After 2 days, the detector is labeled 'mature'. For the next 7 days it is matched against all new network connections. If during this time, the activation threshold is exceeded, the detector is activated, and otherwise it dies (analogous to B cells). Once activated, the detector must receive costimulation from a human operator within 24 h (analogous to T-cell help), otherwise it dies. There is no effector arm in this system. Activated detectors that receive costimulation signal an alarm rather than taking autonomous action. After costimulation, the activated detectors enter the memory pool, living indefinitely. Figure adapted with permission from Hofmeyr and Forrest (92). Copyright © 2000 by the Massachusetts Institute of Technology.

trations. The rules for the CA simulation are the following: a healthy cell becomes infected A1 in the next time step if any of its eight neighbors are infected A1 or if at least $2 < R < 8$ of its neighbors are infected A2; an infected-A1 cell becomes infected-A2 cell after τ time steps; an infected-A2 cell dies in the next time step; a dead cell is replaced by a healthy cell with probability p_{repl} and otherwise remains dead in the next time step; and new cells are created as infected-A1 cells with probability p_{infec}, such that the rate at which dead cells are replaced by infected-A1 cells is $p_{newinfec} = p_{repl} \times p_{infec}$. The last rule simulates the arrival of infected cells, either from other compartments or from the activation of latent infected cells.

In the model, infection is permanent, because new infected cells are continually added. These infection seeds lead to the formation of predictable square waves of infection, as seen in *Fig. 8B,C*. In turn, the square waves interact with each other, ultimately forming a more complex square wave pattern, illustrated in *Fig. 8D*.

Spatial effects

Strain *et al.* (123) described a three-dimensional model of HIV, which incorporates additional biophysical properties. The model's goal was to study the role of spatial effects in viral propagation. Assuming that virus is released as a single burst by an infectious cell, the diffusion coefficient was determined, and an expression derived for the probability $P_b(i)$ that a cell i sites away from the burst becomes infected. From this determination, the basic reproduction ratio (R_0) was calculated for HIV, taking into consideration the localized spatial nature of viral bursts. The calculation predicts that viral propagation is limited by viral stability at low target cell density, and by geometry (target cell's radius) at high cell density.

Each site of the three-dimensional grid can be in one of three states: empty (E), infected (I), or target (T). The following rules determine the simulation dynamics: a target cell at site i becomes infected at the next time step with a probability $\sum_{j \neq i} P_b\left(i-j\right)$; a site containing an infected cell becomes empty in the next time step (death); and an empty site acquires a target cell in the next time step with probability $\sigma_T + \frac{1}{n}\sum_{nn}\delta_T$, where nn indicates nearest-neighbor lattice sites, and n is the total number of nearest neighbors [six for the simple cubic lattice used in Strain *et al.* (123)]. The terms δ_T and σ_T are the rate of repopulation of empty lattice site, as a result of the division of immediate neighbors, and influx of cells from peripheral blood or from the thymus, respectively.

The spatiotemporal dynamics of the model are determined by propagation efficiency and recovery rate. If the propagation efficiency causes the reproduction ratio to be less than one, the infection does not propagate. The long-term dynamics of the model are determined by the rate of recovery of target cells. Infection propagates as radial wave fronts, leaving a wake of empty cells. If cells recover quickly, virus can diffuse from producer cells in the wave front back across the wake. After the initial wave propagates to the edge of the grid, the system can settle into a chaotic attractor in which infected, target, and empty sites coexist (123). If recovery is slow, the infection propagates transiently as a unidirectional wave. In this case, infection can be sustained only if the influx of target cells to random empty sites of the grid is non-zero ($\sigma_T \neq 0$).

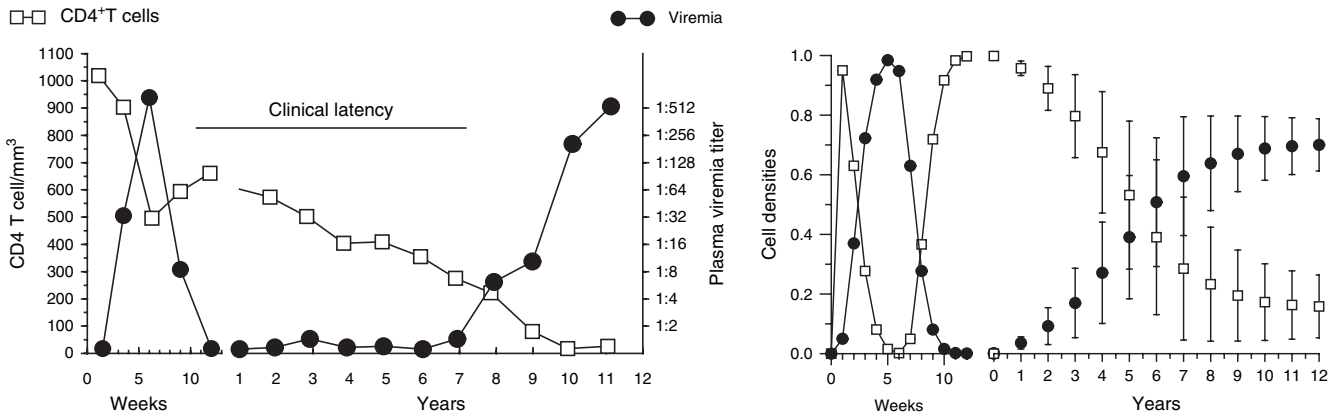

***Fig. 7.* Experimental HIV data (A) and simulation results from the CA model (B).** (A) The density of $CD4^+$ T cells (open squares) and virus concentration (full circles) are shown. (B) The density of healthy (open squares) and infected (full circles) target cells are shown. *Fig. 7A* is reproduced and adapted by permission from Pantaleo G, Grazioi C, Fauci AS. The immunopathogenesis of human immunodeficiency virus infection. N Engl J Med 1993;**328:**327–335. Copyright © 1993 Massachusetts Medical Society. All rights reserved. *Fig. 7B* is reprinted with permission from Zorzenon dos Santos and Coutinho (122). Copyright 2001 by the American Physical Society.

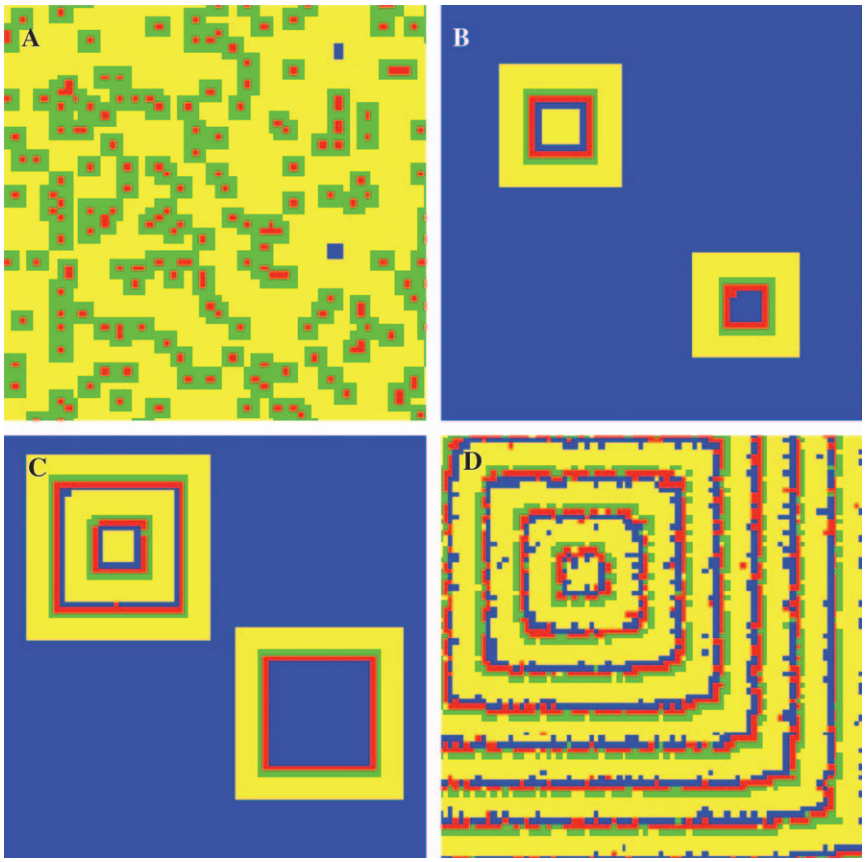

Fig. 8. Screenshots from the Zorzenon dos Santos and Coutinho CA simulation of HIV infection at (A) 5, (B) 18, (C) 25, and (D) 200 weeks. Colors mark healthy (blue), infected A1 (yellow), infected A2 (green), and dead (red) $CD4^+$ T cells or monocytes, the targets of HIV. Figure is reprinted with permission from Zorzenon dos Santos and Coutinho (122). Copyright 2001 by the American Physical Society.

Summary and comparison

Neither model accounts for target cell motility. This assumption is more serious in the case of the first model, because the model simulates HIV dynamics on the scale of several weeks to even years. Allowing the cells to move, for example, would prevent the formation of the perfect square waves, and it could potentially prevent the emergence of the more complicated square patterns that were interpreted as the onset of AIDS.

The second model is interesting because it highlights how ABM can be used to study the contribution of spatial effects on viral propagation and how this contribution can change the conclusions arrived at using mean-field approaches. Although the results need to be validated in more realistic models, the finding that ODE models, such as those proposed in Perelson *et al.* (124), might overestimate viral infectiousness by more than an order of magnitude, is of interest to modelers and experimentalists alike.

Models of Mtb

We discuss two recent models of Mtb, one special purpose simulation known as the Segovia-Juarez *et al.* model (125) and one using the CyCells simulator (41). Although the models are similar, there are three key differences. First, the Segovia-Juarez *et al.* model simulates a two-dimensional slice of tissue, while the CyCells model is in three dimensions. The models also treat cell movement differently. The high density of cells in granulomas makes accurate treatment of collisions between cells and cell pressure in overcrowded regions crucial. In the Segovia-Juarez *et al.* model, cells move discretely from grid site to grid site. In the CyCells model, cell positions are defined by continuous-valued coordinates, and cells adjust their position in response to pressure from other cells. Cells can overlap to account for the fact that real cells can deform to pack tightly, but they are subject to a repulsive force that increases with increasing overlap (41). Finally, the two models differ in how they represent extracellular bacteria. In the CyCells model, extracellular bacteria are modeled explicitly, like T cells and macrophages, rather than as a continuous-valued concentration at each site, as in the Segovia *et al.* model.

Two-dimensional model

The Segovia-Juarez *et al.* model (125) is defined on a two-dimensional square grid with toroidal boundary conditions, representing a cross-section of alveolar lung tissue. The size of one lattice site matches the diameter of the largest cells in the model: macrophages, with a diameter of 20 μm. The discrete

agents are macrophages and T cells, both of which move randomly on the grid biased by cytokine concentration. The macrophages can be in any of four states: resting, infected, chronically infected, or activated. Resting macrophages ingest extracellular bacteria and either kill them and remain in the resting state, or become infected. Once phagocytosed, the intracellular bacteria grow according to the logistic equation. An infected macrophage with only a small number of Mtb ($<N_c$) is activated in the presence of T cells. Otherwise, it becomes chronically infected once the number of intracellular Mtb exceeds a set threshold. Chronically infected macrophages die either by bursting (releasing all their Mtb) or from contact with cytotoxic T cells (releasing a smaller number of Mtb). An activated macrophage clears all of its intracellular Mtb and kills most of the extracellular Mtb in the site it occupies.

Extracellular Mtb and cytokines are represented as real-valued variables that diffuse across the grid. In contrast with the CyCells model, there is one generic cytokine type, which biases the random movement of T cells and macrophages. Cytokines are released by infected, chronically infected, and activated macrophages, and they are cleared at a fixed rate. In a real infection, the immune response can lead to multicellular structures known as granulomas. In the model, granulomas arise from the attractive effect of cytokines and because the motility of infected macrophages is reduced compared with the resting ones.

The model is distinctive, because its designers conducted an extensive sensitivity analysis of 12 of the 27 major parameters using the number of extracellular bacterium and granuloma size as outcome variables. For example, intracellular growth rate of Mtb is positively correlated with the number of extracellular Mtb at early times of the infection and at much later times post-infection, but it is negatively correlated at intermediate times. This finding suggests that at early times, a large growth rate is necessary for the infection to establish itself, while at later times, a smaller growth rate is more favorable to ensure a large and lasting infection.

CyCells model of tuberculosis

The second model, built at about the same time using CyCells, is three dimensional, represents a slightly richer set of cytokines, and uses more realistic rules for cell movement (41). The simulation volume is discretized as a three-dimensional cubic lattice with toroidal boundary conditions. As in the earlier model, two cell types are modeled: macrophages and T cells. The CyCells model includes tumor necrosis factor (TNF), interleukin-10 (IL-10), and interferon-γ (IFN-γ), although the three cytokines act as surrogates for an even greater array of cytokines. Cytokines diffuse through the grid with fixed diffusion and clearance rates. IFN-γ has the effect of downregulating IL-10, and it causes newly infected macrophages to activate and clear intracellular bacteria in the presence of T cells. IL-10 can shut off TNF production by newly infected macrophages, and TNF increases the chance of newly infected macrophages to become activated and clear intracellular bacterium.

Macrophages in the model can be in one of three states: uninfected, newly infected, and chronically infected. Uninfected macrophages become newly infected by phagocytosing extracellular Mtb. Newly infected macrophages can produce cytokines and/or kill their intracellular Mtb, provided that they encounter T cells and are exposed to the appropriate local cytokine environment. Over time, infected macrophages lose the ability to acquire these functions (to activate), and they become chronically infected.

The cells in the model can reside at any point in three-dimensional space. Cell movement follows a persistent random walk, where each cell type has a speed at which it moves for a fixed time interval, after which it randomly picks a new direction. Cells sense the local cytokine concentration at their grid site, but in some cases they interact with cells from neighboring sites as well as their own site.

Simulations with this model replicated qualitative outcomes from several different experiments in murine models of tuberculosis. These outcomes included the unrestricted growth of Mtb in the presence of macrophages alone, and restricted replication following the influx of immune T cells. The model also qualitatively reproduced experimental results on the effects of IFN-γ, where reduced signaling because of IFN-γ leads to enhanced mycobacterial growth. These results were surprising because the model is highly simplified. Yet, it captured some of the contradictory effects of IL-10, which both inhibits inflammation (reducing macrophage influx) and inhibits macrophage activation.

Models of influenza

Until recently, there were relatively few mathematical or computational models of influenza. The first one, dating back to 1976, investigated influenza infection dynamics in mice (126). It was a simple model, consisting of seven variables (or compartments) and five rate parameters. The second model in 1994 studied the dynamics of influenza infection in humans (2). It is more complex, uses ODEs with delays, and consists of 13 variables and 60 parameters.

ABM approaches to influenza date back at least to Smith *et al.*'s (46, 127) study of the effect of repeated annual influenza vaccination. This model focused on cross-reactive memory by observing that immune memory resembles associative

memories used in computing, in particular, Kanerva's Sparse Distributed Memory (128). In the model, B cells (naive cells, plasma cells, and memory cells) and antibodies were represented as memory elements, and antigens (both vaccines and infectious strains) were interpreted as memory probes. All of these elements were represented as strings of 20 symbols over a four-letter alphabet, where each symbol could take on one of four different values. The model simulated a realistic-sized B-cell repertoire and was used to investigate vaccine design for influenza. The results suggested the antigenic distance hypothesis, that variation in repeat vaccine efficacy is because of differences in antigenic distances among vaccine strains and between vaccine strains and the epidemic strain in particular outbreaks. Since the original publications, the model has been extended and refined (129, 130), which led to the concept of 'antigenic cartography' (21, 131, 132). Antigenic cartography is now a core component of the human influenza vaccine strain selection process.

A more recent ABM influenza model, the above-mentioned `ma_immune`, consists of a two-dimensional square lattice, where each site corresponds to a ciliated lung epithelial cell that can be in one of five states: healthy, containing, expressing, infectious, or dead. Additionally, a population of generic immune cells patrols the simulated tissue (the grid), moving randomly from site to site. The simulation is initiated with a certain fraction of cells containing virions. After 4 h, these cells start expressing viral peptides, which means that they can be recognized and killed by the patrolling immune cells. Then, 2 h later, the expressing epithelial cells also start to infect their immediate neighbors. Finally, after a cell has been infected for 24 h, it dies. Dead epithelial cells are replaced by healthy cells at a fixed rate. When a generic immune cell encounters an epithelial cell expressing viral peptide, it kills that cell, and recruitment takes place by probabilistically adding a new immune cell to random locations on the simulation grid.

In Beauchemin *et al.* (34), the model was calibrated to the dynamics of an influenza A viral infection, reproducing the general shape of a response to an uncomplicated viral infection and giving quantitatively reasonable results when parameterized for a particular viral infection. When the parameter values were set to biologically plausible values or ranges, only 5 of the 12 parameters could be fit using available experimental data. Even with these restrictions, the model was able to reproduce accurately the available dynamical features of influenza A infection. The model's agreement with available data also compares well to the Bocharov and Romanyukha differential equation model (2). The results obtained with the `ma_immune` model suggest that by adding additional details to the model, such as specific immune cell types, explicit representation of virions and the appropriate cytokines, `ma_immune` could become a useful model of influenza A infection.

Model of a primary immune response in a lymph node

Catron *et al.* (133) describe an ABM of a primary immune response induced by antigen in a skin-draining lymph node. The ABM consists of a two-dimensional plane. The plane corresponds to a 10 μm slice (approximately one cell diameter) through a hypothetical spherical lymph node 2 mm in diameter (the side dimension of the simulation plane). The model includes T cells, B cells, and dendritic cells (DCs). Random motion paths for each cell in the simulation were designed individually and scaled to their known approximate speed, based on observations from two-photon microscopy (54, 55). Additional constraints were placed on the paths such that B cells would be confined to the follicles (except 5 h after exposure to antigen when their movement is restricted to the outer edge of the follicles near the T-cell area), DCs to the T-cell area, and T cells to the T-cell area and outer edges of the follicles for 90 and 10% of their paths, respectively. When a collision occurs between two cells, the motion of the cells along their respective paths is halted temporarily to mimic intercellular interaction. This model is conceptually important, because it was constructed by an experimentalist, showing that ABM techniques are starting to bridge the gap between theory and experiment.

Summary and conclusions

The previous sections have reviewed a body of work that seeks to construct computational immune systems that behave analogously to the natural immune system. Some of the examples (*Immune system modeling with ABM*: and *Applications to biomedicine*) were developed as models and others (*Engineering an immune system*) as practical solutions to engineering problems. In both cases, computer immunology proceeds by hypothesizing a sufficient set of mechanisms needed to produce a desired behavior and implementing them as computer programs. This constructive approach to understanding immunology differs from experimental methods that selectively remove functionality such as experiments with knockout mice. Although the computational mechanisms are crude compared with their biological analogs, the resulting computer immune systems can show surprisingly realistic behaviors and sometimes be calibrated closely with experimental data (*Applications to biomedicine*). In the context of engineering problems, it is often possible to analyze the functional behavior of a given mechanism more rigorously than what might be achieved experimentally, in some cases providing insight into the natural immune system.

The ABM approach to immune modeling (*Agent-based modeling for immunology*) led to the comprehensive abstract models described in *Immune system modeling with ABM*, which in turn were used to create specialized models of particular immunological phenomena (*Applications to biomedicine*). As computational power increased, the geometry of the models expanded from one-dimensional to two-dimensional and now three-dimensional simulations, as well as incorporating cell movement. In addition, visualization methods became more sophisticated, simplifying the task of understanding and specifying the models. This facilitated the use of models by experimentalists, culminating in the Catron model (*Model of a primary immune response in a lymph node*) in which the model itself was developed by an experimentalist. In the future, we can expect these trends to continue, with even more ambitious and detailed models, more sophisticated visualizations that run in real-time, and more direct involvement by experimentalists in the model building process.

There are strengths and limitations to this approach. It can be difficult to identify the proper level of abstraction, decide what aspects of the immune response are important and what their proper role or 'purpose' is, and how they should be translated into computation. In spite of these limitations, computational abstractions and concepts have proved powerful enough to provide important insights into immunological processes (*Applications to biomedicine*) and to solve challenging engineering problems (*Engineering an immune system*). By abstracting away from physical realism, AIS can enhance our understanding of the large-scale patterns of interaction that occur among the millions of individual components that comprise a natural immune system. Efforts to build an immune system tailored for computer networks have highlighted the crucial roles played by certain immune system mechanisms.

The synthetic approach to modeling immune system behavior has generated interest, but there is a question about what an ABM model contributes to understanding the natural immune system. There are several ways that the models described in this paper complement experimentation in 'wet labs'. First, if a synthetic computer model captures the relevant phenomena, it is much easier to perform experiments on the model than on the natural system. In particular, it is much easier to isolate mechanisms and test hypotheses about how they function and what their significance is to the overall system. For example, the Smith model (46) showed that the antigenic distance hypothesis provided a parsimonious explanation of complex results on vaccine effectiveness. Second, in an era when an overwhelming volume of experimental results have become available, it is no longer humanly possible to comprehend all of the data that might be relevant to a problem of interest. Synthetic models, such as CyCells, can be used to integrate specialized models for different phenomena into one system to see how they interact (e.g. to test if the assumptions of one specialized model contradict those of another). Models that can incorporate data and hypotheses from many different experiments will likely be necessary in the future to integrate knowledge so that it can be used productively. Although many of the models described here do not lend themselves to rigorous mathematical analysis, they encode assumptions and hypotheses in a precise, mechanical way. Running models allows researchers to identify gaps and inconsistencies in their knowledge by making assumptions explicit, allowing them to make predictions, generate new hypotheses, and suggest new experiments. By better understanding the functional significance of different components of the immune system, it may be possible to better predict the effects of therapeutic interventions. In the future, models similar to those described in this review may be used to predict efficacy of new treatments and vaccines, thus avoiding some costly experiments.

The modeling process itself has value. Although we have focused in this paper on the artifacts that modeling produces, modeling is not only about building a model. At its best, it involves an iterative process of model construction, model analysis, followed by the creation of new models determined by the results of the analysis. An important feature of the models described here is their flexibility, allowing researchers to try out variations within the same framework and to add complexity to the model incrementally. This greatly simplifies the work of testing alternative hypotheses, designing experiments, and discovering both necessary and sufficient mechanisms to explain observed behavior.

References

1. Bonabeau E. Agent-based modeling: Methods and techniques for simulating human systems. Proc Natl Acad Sci USA 2002;**99**:7280–7287.
2. Bocharov GA, Romanyukha AA. Mathematical model of antiviral immune response III. Influenza A virus infection. J Theor Biol 1994;**167**:323–360.
3. Perelson AS, Neumann A, Markowitz M, Leonard J, Ho D. HIV-1 dynamics in vivo: virion clearance rate, infected cell life-span, and viral generation time. Science 1996;**271**:1582–1586.
4. Jerne NK. Towards a network theory of the immune system. Ann Inst Pasteur Imm 1974;**125C**:373–389.
5. Varela F, Coutinho A, Dupire B, Vaz NM. Cognitive networks: immune, neural, and otherwise. In: Perelson AS, ed. Theoretical Immunology, Part Two. Redwood City, CA: Addison-Wesley, 1988:359–375.
6. Vertosick FT, Kelly RH. Immune network theory: a role for parallel distributed processing. Immunology 1989;**66**:1–7.

7. Oprea M, Forrest S. How the immune system generates diversity: pathogen space coverage with random and evolved antibody libraries. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). San Francisco, CA: Morgan-Kaufmann, 1999:1651–1656.
8. Hightower RR, Forrest S, Perelson AS. The evolution of emergent organization in immune system gene libraries. In: Eshelman LJ, ed. Proceedings of the 6th International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kaufmann, 1995:344–350.
9. Perelson A, Oster G. Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-non-self discrimination. J Theor Biol 1979;**81**:645–670.
10. Percus JK, Percus O, Perelson AS. Predicting the size of the antibody combining region from consideration of efficient self/non-self discrimination. Proc Natl Acad Sci USA 1993;**90**:1691–1695.
11. Weinand R. Somatic mutation and exploration of the antibody landscape: a shape-space computer model. In: Perelson AS, Kauffman SA, eds. Molecular Evolution on Rugged Landscapes: Proteins, RNA and the Immune System. Redwood City, CA: Addison-Wesley, 1991.
12. Celada F, Seiden PE. Affinity maturation and hypermutation in a simulation of the humoral response. Eur J Immunol 1996;**26**:1350–1358.
13. Seiden PE, Celada F. A model for simulating cognate recognition and response in the immune system. J Theor Biol 1992;**158**:329–357.
14. Kleinstein SH, Seiden PE. Simulating the immune system. Comput Sci Eng 2000;**2**:69–77.
15. Bernaschi M, Castiglione F. Design and implementation of an immune system simulator. Comput Biol Med 2001;**31**:303–331.
16. Warrender C, Forrest S, Segel L. Homeostasis of peripheral immune effectors. Bull Math Biol 2004;**66**:1493–1514.
17. Meier-Schellersheim M, Xu X, Angermann B, Kunkel EJ, Jin T, Germain RN. Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method. PLoS Comput Biol 2006;**2**:710–724.
18. Farmer JD, Packard NH, Perelson AS. The immune system, adaptation, and machine learning. Physica D 1986;**22**:187–204.
19. Perelson AS, Weisbuch G. Immunology for physicists. Rev Mod Phys 1997;**69**: 1219–1267.
20. Smith DJ, Forrest S, Hightower R, Perelson AS. Deriving shape space parameters from immunological data for a model of cross-reactive memory. J Theor Biol 1997;**189**:141–150.
21. Smith DJ, et al. Mapping the antigenic and genetic evolution of influenza virus. Science 2004;**305**:371–376.
22. Kaufman M, Urbain J, Thomas R. Towards a logical analysis of the immune response. J Theor Biol 1985;**114**:527–561.
23. Weisbuch G, Atlan H. Control of the immune response. J Phys A Math Gen 1988;**21**:L189–L192.
24. Dasgupta S. Monte Carlo simulation of the shape space model of immunology. Physica A 1992;**189**:403–410.
25. Stauffer D, Weisbuch G. High-dimensional simulation of the shape-space model for the immune system. Physica A 1992;**180**:42–52.
26. De Boer R, van der Laan JD, Hogeweg P. Randomness and pattern scale in the immune network: a cellular automata approach. In: Stein WD, Varela FJ, eds. Thinking About Biology: An Introductory Essay. Reading, MA: Addison-Wesley, 1993:231–252.
27. Celada F, Seiden PE. A computer model of cellular interactions in the immune system. Immunol Today February 1992;**13**:56–62.
28. Kesmir C, Boer RJD. A spatial model of germinal center reactions: cellular adhesion based sorting of b cells results in efficient affinity maturation. J Theor Biol 2003;**222**:9–22.
29. Bezzi M, Celada F, Ruffo S, Seiden PE. The transition between immune and disease states in a cellular automaton model of clonal immune response. Physica A 1997;**245**:145–163.
30. Celada F, Seiden PE. Modeling immune cognition. In: Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA, 11–14 October 1998: 3787–3792 (IEEE cat. no. 98CH36218).
31. Stewart JJ, et al. A solution to the rheumatoid factor paradox: pathologic rheumatoid factors can be tolerized by competition with natural rheumatoid factors. J Immunol 1997;**159**:1728–1738.
32. Kohler B, Puzone R, Seiden PE, Celada F. A systematic approach to vaccine complexity using an automaton model of the cellular and humoral immune system. i. Viral characteristics and polarized responses. Vaccine 2000;**19**:862–876.
33. Bernaschi M, Castiglione F. Selection of escape mutants from immune recognition during HIV. Immunol Cell Biol 2002;**80**:307–313.
34. Beauchemin C, Samuel J, Tuszynski J. A simple cellular automaton model for influenza A viral infections. J Theor Biol 2005;**232**:223–234.
35. Beauchemin C MASyV: A Multi-Agent System Visualization Package 2003. Open source software available online on SourceForge at: http://masyv.sourceforge.net. Last accessed 20 February 2005.
36. Beauchemin C. Probing the effects of the well-mixed assumption on viral infection dynamics. J Theor Biol 2006;**242**:464–477.
37. Meier-Schellersheim M, Mack G. SIMMUNE, A Tool for Simulating and Analyzing Immune System Behavior (arXiv:cs.MA/ 9903017), 1999. Available at http://arxiv.org/abs/cs/9903017
38. Meier-Schellersheim M. The Immune System as a Complex System: Description and Simulation of the Interactions of its Constituents. PhD thesis, Physics Department, University of Hamburg, Hamburg, 2001.
39. Warrender CE. Modeling intercellular interactions in the peripheral immune system. PhD thesis, Computer Science Department, University of New Mexico, Albuquerque, NM, 2004.
40. Warrender CE. CyCells Simulator 2005. Open source GNU GPL software available online on SourceForge at: http://sourceforge.net/projects/cycells. Last accessed 22 February 2005.
41. Warrender C, Forrest S, Koster F. Modeling intercellular interactions in early mycobaterium infection. Bull Math Biol 2006;**68**:2233–2261.
42. Chao DL, Davenport MP, Forrest S, Perelson AS. A stochastic model of cytotoxic T cell responses. J Theor Biol 2004;**228**:227–240.
43. Kohler G. Frequency of precursor cells against the enzyme beta-galactosidase: an estimate of the BALB/c strain antibody repertoire. Eur J Immunol 1976;**6**:340–347.
44. Arstila TP, Casrouge A, Baron V, Even J, Kanellopoulos J, Kourilsky P. A direct estimate of the human $\alpha\beta$ T cell receptor diversity. Science 1999;**286**:958–961.
45. Detours V, Mehr R, Perelson AS. A quantitative theory of affinity-driven cell repertoire selection. J Theor Biol 1999;**200**:389–403.
46. Smith DJ, Forrest S, Ackley DH, Perelson AS. Variable efficacy of repeated annual influenza vaccination. Proc Natl Acad Sci USA 1999;**96**:14001–14006.
47. Chao DL, Davenport MP, Forrest S, Perelson AS. The effects of thymic selection on the range of t cell cross-reactivity. Eur J Immunol 2005;**35**:3452–3459.
48. Detours V, Perelson AS. Explaining high alloreactivity as a quantitative consequence of affinity-driven thymocyte selection. Proc Natl Acad Sci USA 1999;**96**:5153–5158.
49. Smith DJ, Forrest S, Ackley DH, Perelson AS. Using lazy evaluation to simulate realistic-size repertoires in models of the immune system. Bull Math Biol 1998;**60**:647–658.
50. Efroni S, Harel D, Cohen IR. Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. Genome Res 2003;**13**:2485–2497.

51. Harel D. Statecharts: a visual formalism for complex systems. Sci Comput Program 1987;**8**:231–274.
52. Harris S. Scientists model interaction of viruses and immune system. Virginia Tech Research 2004. Virginia Tech, Blacksburg, VA: Office of the Vice President for Research.
53. Polys NF, Bowman DA, North C, Laubenbacher RC, Duca K. PathSim visualizer: an Information-Rich Virtual Environment framework for systems biology. In: Brutzman DP, Chittaro L, Puk R, eds. Proceeding of the Ninth International Conference on 3D Web Technology, Web3D 2004, Monterey, California, USA, 5–8 April 2004. New York: ACM Press, 2004; 7–14.
54. Miller MJ, Wei SH, Parker I, Cahalan MD. Two-photon imaging of lymphocyte motility and antigen response in intact lymph node. Science 2002;**296**: 1869–1873.
55. Miller MJ, Hejazi AS, Wei SH, Cahalan MD, Parker I. T cell repertoire scanning is promoted by dynamic dendritic cell behavior and random T cell motility in the lymph node. Proc Natl Acad Sci USA 2004;**101**:998–1003.
56. Mempel TR, Henrickson SE, von Adrian UH. T-cell priming by dendritic cells in lymph nodes occurs in three distinct phases. Nature 2004;**427**:154–159.
57. Dasgupta D, ed. Artificial Immune Systems and Their Applications. Heidelberg: Springer, 1999.
58. de Castro LN, Timmis J. Artificial Immune Systems: A New Computational Intelligence Approach. London: Springer, 2002.
59. Cohen F. Computer viruses. Comput Secur 1987;**6**:22–35.
60. Spafford EH, Computer viruses – a form of artificial life? In: Langton CG, Taylor C, Farmer JD, Rasmussen S, eds. Artificial Life II. Redwood City, CA: Addison-Wesley, 1992:727–745.
61. Kephart JO, White SR, Chess DM. Computers and epidemiology. IEEE Spectrum 1993;**30**:20–26.
62. Murray WH. The application of epidemiology to computer viruses. Comput Secur 1988;**7**:139–150.
63. Pastor-Satorras R, Vespignani A. Epidemic spreading in scale-free networks. Phys Rev Lett 2001;**86**:3200–3203.
64. Newman M, Forrest S, Balthrop J. Email networks and the spread of computer viruses. Phys Rev E 2002;**66**.
65. Forrest S, Perelson AS, Allen L, Cherukuri R. Self-nonself discrimination in a computer. In: Rushby J, Meadows C, eds. Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy. Los Alamitos, CA: IEEE Computer Society Press, 1994:035101-1. .035101-4.
66. Kephart JO, Sorkin GB, Arnold WC, Chess DM, Tesauro GJ, White SR. Biologically inspired defenses against computer viruses. In Keebling L, safiotti A, eds. *IJCAI* '95, International Joint Conference on Artificial Intelligence, Denver: Professional Book Center 1995:958–996.
67. Oda T, White T. Immunity from spam: an analysis of an artificial immune system for junk email detection. In: Jacob C, Pilat M, Bentley P, Timmis J, eds. Artificial Immune Systems, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005:276–289.
68. Li T, Ding J, Liu X, Yang P. A new model of immune-based network surveillance and dynamic computer forensics. In: Wang L, Chen KC, Ong Y, eds. Proceedings of the First International Conference on Natural Computation, IEEE. Berlin, Heidelberg: Springer-Verlag, 2005:804–813.
69. Cayzer S, Aickelin U. The danger theory and its application to artificial immune systems. In: Bentley P, Timmis J, eds. Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS-2002), University of Kent at Canterbury, 2002:141–148.
70. Bersini H. Hints for adaptive problem solving gleaned from immune networks. Technical report, IRIDIA – Universite Libre de Bruxelles, Bruxelles, Belgium 1990.
71. Bersini H. Immune network and adaptive control. In: Valera F, Bourgine P, eds. Proceedings of the First European Conference on Artificial Life, Paris, France. Cambridge, MA: MIT Press, 1992:217–226.
72. de Castro LN, Zuben FJV. ainet: an artificial immune network for data analysis. In: Abbass HA, Sarker RA, Newton CS, eds. Data Mining: A Heuristic Approach. Hershey, PA: Idea Group Publishing, 2001:231–259.
73. Forrest S, Javornik B, Smith R, Perelson A. Using genetic algorithms to explore pattern recognition in the immune system. Evol Comput 1993;**1**:191–211.
74. de Castro L, Zuben FV. The clonal selection algorithm with engineering applications. . In: Whitley LD et al., eds. Proceedings of the Genetic and Evolutionary Computation Conference. California: Morgan Kaufman, 2000:36–37.
75. Hart E, Ross P. An immune system approach to scheduling in changing environments. In: Banzhaf W, et al. eds. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). California: Morgan Kaufmann, 1999:1559–1566.
76. Cayzer S, Smith J, Marshall J, Kovacs T. What have gene libraries done for AIS? In: Jacom C, Pilat M, Bentley P, Timmis J, eds. Artificial Immune Systems. Vol 3627 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005:86–99.
77. Greensmith J, Aicklin U, Cayzer S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In: Jacom C, Pilat M, Bentley P, Timmis J, eds. Artificial Immune Systems. Vol 3627 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005: 153–167.
78. Trapnell B., A peer-to-peer blacklisting strategy inspired by leukocyte-endothelium interaction. In: Jacom C, Pilat M, Bentley P, Timmis J, eds. Artificial Immune Systems. Vol 3627 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2005:339–352.
79. Forrest S, Hofmeyr S, Somayaji A, Longstaff T. A sense of self for Unix processes. In: McHugh J, Dinolt G, eds. Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy. California: IEEE Press, 1996.
80. Somayaji A, Forrest S, Automated response using system-call delays. In: Bellovin S, Rose G., eds. Usenix Security Symposium. Denver, Usenix, 2000.
81. Hofmeyr S, Somayaji A, Forrest S. Intrusion detection using sequences of system calls. J Comput Secur 1998;**6**:151–180.
82. Cowan C, Wagle P, Pu C, Beattie S, Walpole J. Buffer overflows: attacks and defenses for the vulnerability of the decade. In: DARPA Information Survivability Conference and Exposition (DISCEX 2000), 2000:119–129.
83. Marceau C. Characterizing the behavior of a program using multiple-length n-grams. In: McHugh J, Blakley B., eds. Proceedings of the New Security Paradigms Workshop 2000. Cork, Ireland: Association for Computing Machinery, 2000:111–118.
84. Naiman DQ. Statistical anomaly detection via httpd data analysis. Comput Stat Data An 2004;**45**:51–67.
85. Lee W, Stolfo S. Data mining approaches for intrusion detection. In: Rubin A, ed. 7th USENIX Security Symposium, 1998. San Antonio, TX: Usenix, 1998: 79–94.
86. Sekar R, Bendre M, Bollineni P, Dhurjati D. A fast automaton-based method for detecting anomalous program behaviors. In: IEEE Symposium on Security and Privacy. Las Alamitos, CA: IEEE Computer Society Press, 2001:144–155.
87. Jones A, Li S. Temporal signatures for intrusion detection. In: Epstien J, ed. Seventeenth Annual Computer Security Applications Conference, 10–14 December 2001, New Orleans, LA, USA. Los Alamitos, CA: IEEE Computer Society, 2001:252–261.
88. Stillerman M, Marceau C, Stillman M. Intrusion detection for distributed applications. Commun ACM 1999;**42**:62–69.

89. Jones A, Lin Y. Application intrusion detection using language library calls. In: Epstien J, ed. Proceedings of the 17th Annual Computer Security Applications Conference. New Orleans, LA, 2001. Las Alamitos, CA: IEEE Computer Society Press, 2001: 442–449.
90. Michael C, Ghosh A. Two state-based approaches to program-based anomaly detection. In: Epstien J, ed. Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00). New Orleans, LA, 2000. Las Alamitos, CA: IEEE Computer Society Press, 2000: 21–31.
91. Wagner D, Dean D. Intrusion detection via static analysis. In: IEEE Symposium on Security and Privacy, 2001. Las Alamitos, CA: IEEE Computer Society Press, 2001: 156–159.
92. Hofmeyr SA, Forrest S. Architecture for an artificial immune system. Evol Comput J 2000;**8**:443–473.
93. Mukherjee B, Heberlein LT, Levitt KN. Network intrusion detection. IEEE Netw 1994;**8**:26–41.
94. Kappler JW, Roehm N, Marrack P. T cell tolerance by clonal elimination in the thymus. Cell 1987;**49**:273–280.
95. D'haeseleer P, Forrest S, Helman P. An immunological approach to change detection: algorithms, analysis and implications. In: McHugh J, Dinolt G, eds. Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy. Las Alamitos, CA: IEEE Press, 1996:110–119.
96. Lamont GB, Marmelstein RE, Veldhuizen DAV. A distributed architecture for a self-adaptive computer virus immune system. In: Corne D, Dorigo M, Glover F, eds. New Ideas in Optimization, Advanced Topics in Computer Science Series. London: McGraw-Hill, 1999:167–183.
97. Dasgupta D, ed. An agent based architecture for a computer virus immune system. GECCO 2000, Workshop on Artificial Immune Systems, July 2000.
98. Kim J, Bentley PJ. An evaluation of negative selection in an artificial immune system for network intrusion detection. In: Beyer H, ed. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). San Francisco, CA: Morgan-Kauffman, 2001:1330–1337.
99. Williams PD, Anchor KP, Bebo JL, Gunsch GH, Lamont GD. Cdis: towards a computer immune system for detecting network intrusions. In: Lee W, Me L, Wespi A, eds. Fourth International Symposium, Recent Advances in Intrusion Detection. Berlin: Springer, 2001:117–133.
100. Sarafijanovic S, Boudec JYL. An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal, and memory detectors. In: Nicosia G, Cutello V, Bentley P, Timmis J, eds. Artificial Immune Systems, volume LNCS 3239 of Lecture Notes in Computer Science. Heidelberg, Berlin: Springer-Verlag, 2004:342–355.
101. Dasgupta D, Forrest S. Novelty detection in time series data using ideas from immunology. In: Harris FC Jr, ed. Proceedings of The International Conference on Intelligent Systems, 1996.
102. Sathyanath S, Sahin F. Artificial immune systems approach to a real time color image classification problem. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2001. doi: 10.1109/ICSMC.2001.972897.
103. Chao DL, Forrest S. Information immune systems. Genet Program Evolvable Mach 2003;**4**:311–331.
104. Esponda F, Ackley ES, Forrest S, Helman P. On-line negative databases. J Unconventional Comput 2005;**1**:201–220.
105. de Mare M, Wright R. Secure set membership using 3SAT. In: Ning P, Qing S, Li N, eds. Eighth International Conference on Information and Communications Security (ICICS '06), Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2006:452–468.
106. Esponda F, Forrest S, Helman P. A formal framework for positive and negative detection schemes. IEEE Trans Syst Man Cybern 2004;**34**:357–373.
107. Esponda F, Jia H, Forrest S, Helman P. Protecting data privacy through hard-to-reverse negative databases. Int J of Inf Security (in press).
108. de Castro LN, Zuben FJV. Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 2002;**6**: 239–251.
109. Kim J, Bentley PJ. Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In: Proceedings of the 2001 Congress on Evolutionary Computation. Seoul: IEE Press, 2001:1244–1252.
110. Smith R, Forrest S, Perelson AS. Searching for diverse, cooperative populations with genetic algorithms. Evol Comput 1993;**1**:127–149.
111. Holland JH. Adaptation in Natural and Artificial Systems, 2nd edn. Cambridge, MA: MIT Press, 1992.
112. Glickman M, Balthrop J, Forrest S. A machine learning evaluation of an artificial immune system. Evol Comput 2005;**13**:179–212.
113. Begnum K, Burgess M. A scaled, immunological approach to anomaly countermeasures: Combining pH with Cfengine. In: Goldszmidt GS, Schonwalder J, eds. Integrated Network Management, IFIP Conference Proceedings. Kluwer, 2003. Vol 246: 31–42.
114. Ishiguro A, Kuboshiki S, Ichikawa S, Uchikawa Y. Gait control of hexapod walking robots using mutual-coupled immune networks. Adv Robot 1996;**10**: 179–195.
115. Segel L, Bar-Or RL. Immunology viewed as the study of an autonomous decentralized system. In: Dasgupta D, ed. Artificial Immune Systems and their Applications. Berlin: Springer-Verlag, 1998:65–88.
116. Segel L, Bar-Or RL. On the role of feedback in promoting conflicting goals of the adaptive immune system. J Immunol 1999;**163**:1342–1349.
117. Bersini H. Self-assertion vs. self-recognition: a tribute to Francisco Varela. In: Timmis J, Bentley P, eds. Proceedings of the First International Conference on Artificial Immune Systems (ICARIS). Canterbury: University of Kent at Canterbury,2002: 107–112.
118. Williamson MM. Throttling viruses: restricting propagation to defeat malicous mobile code. In: Proceedings of ACSAC Security Conference, Las Vegas, Nevada, 2002. http:// citeseer.ist.psu.edu/williamson02throttling.html.
119. Balthrop J, Forrest S, Newman M, Williamson M. Technological networks and the spread of computer viruses. Science 2004;**304**:527–529.
120. Balthrop J. RIOT: a responsive system for mitigating computer network epidemics and attacks. Master's thesis, The University of New Mexico, Albuquerque, NM, 2005.
121. Harmer PK, Williams PD, Gunsch GH, Lamont GB. Artificial immune system architecture for computer security applications. IEEE Trans Evol Comput 2002;**6**: 252–280.
122. Zorzenon dos Santos RM, Coutinho S. Dynamics of HIV infection: a cellular automata approach. Phys Rev Lett 2001;**87**:168102.
123. Strain MC, Richman DD, Wong JK, Levine H. Spatiotemporal dynamics of HIV propagation. J Theor Biol 2002;**218**:85–96.
124. Perelson AS, et al. Decay characteristics of HIV-1 infected compartments during combination therapy. Nature 1997;**387**: 188–191.
125. Segovia-Juarez JL, Ganguli S, Kirschner D. Identifying control mechanisms of granuloma formation during *M. tuberculosis* infection using an agent-based model. J Theor Biol 2004;**231**:357–376.

126. Larson EW, Dominik JW, Rowberg AH, Higbee GA. Influenza virus population dynamics in the respiratory tract of experimentally infected mice. Infect Immun 1976;**13**:438–447.
127. Smith DJ, et al. Modeling the effects of updating the influenza vaccine on the efficacy of repeated vaccination. In: Osterhaus ADME, Cox NJ, Hampson AW, eds. Options for the Control of Influenza Virus IV. Vol 1219 of International Congress Series. Amsterdam: Excerpta Medica (Elsevier), 2001: 655–660. Proceedings of the World Congress on Options for the Control of Influenza Virus IV, Crete, Greece, 23–28 September 2000.
128. Kanerva P. Sparse Distributed Memory. Cambridge, MA: MIT Press, 1988.
129. Rimmelzwaan G, Berkhoff E, Nieuwkoop N, Smith D, Fouchier R, Osterhaus A. Multiple clustered computations are required for full functional compensation of a detrimental mutation in an influenza virus CTL epitope. J Gen Virol (in press).
130. Voordouw A, et al. Annual revaccination against influenza effectively reduces mortality risk in community dwelling elderly. JAMA 2004;**292**:2089–2095.
131. Boon ACM, et al. Recognition of homo- and heterosubtypic variants of influenza a viruses by human CD8+ T lymphocytes. J Immunol 2004;**172**:2453–2460.
132. Smith D. Predictability and preparedness in influenza control. Science 2006;**312**: 392–394.
133. Catron DM, Itano AA, Pape KA, Mueller DL, Jenkins MK. Visualizing the first 50 hr of the primary immune response to a soluble antigen. Immunity 2004;**21**:341–347.