

Traditionally, modeling in science has been *mathematical modeling*. Even today, the gold standard for respectability in science is still the use of formulas, symbols and integrals to communicate concepts, ideas and arguments. There is a good reason for this. The language of mathematics is concise and precise and allows the initiated to say with a few Greek letters what would otherwise require many pages of text. Mathematical analysis is important in science, but it would be wrong to make its use an absolute criterion for good science.

Many of the phenomena modern science studies are complicated, indeed so complicated that the brightest mathematicians have no hope of successfully applying their craft to describe these phenomena. Most of reality is so complex that even formulating the mathematical model is close to impossible. Nevertheless, many of these phenomena are worth our attention and meaningful knowledge can be obtained from studying them.

This is where computer models become useful tools in a scientist's hands. Simulation models can be used to describe and study phenomena even when traditional mathematical approaches fail. In this chapter we will introduce the reader to a specific class of computer models that is very useful for exploring a multitude of phenomena in a wide range of sciences: the class of *agent-based models*.

Computers are still a relatively recent addition to the methodological armory of science. Initially their main use was to extend the range of tractability of mathematical models. In the pre-computer era, any mathematical model had to be solved by laborious manual manipulations of equations. This is, of course, time intensive and error prone. Computers made it possible to out source the tedious hand-manipulations and, more importantly, to generate numerical solutions to mathematical models. What takes hours by hand can be done within an instant by a computer. In that way, computers have pushed the boundaries of what could be calculated.

2.1 Mathematical and Computational Modeling

In what follows, we are not going to be interested in computer-aided mathematical modeling, i.e., methods to generate numerical solutions. Instead, we will be looking at a class of computer models that is *formal* by virtue of being specified in a programming language with precise and unambiguous semantics. Yet the models we are interested in are also non-mathematical, in the sense that they represent and model phenomena that go well beyond what can even be formulated mathematically, let alone be solved.

Computer models are formal models even when they are not mathematical models. They are written in a precise language that is designed not to leave any room for ambiguity in its implementation. Every detail of the model must be expressed in this language and no aspect can be left out. All the computer does, once the model is formulated, is to mercilessly draw conclusions from the model's specification. Formal analysis of this kind is a useful tool for generating a deep understanding of the systems that are studied, and is far superior to mere verbal reasoning. It is therefore worth spending time and effort to develop computational representations of systems even when (or precisely when) a mathematical analysis seems hopeless.

How do we know that a system is not amenable to mathematical analysis but can be modeled computationally? Maybe, it is easier first to understand what it is that makes a system suitable for mathematical analysis. Possibly the most successful and influential mathematical model in science is that formulated by Newton's laws of motion. These laws can be applied to a wide variety of phenomena, ranging from the trajectory of a stone thrown into a lake, to the motion of planets around their stars. One feature that Newton's laws share with many models in physics is that they are *deterministic*. The defining feature of a deterministic system is that, once its initial conditions are fixed, its entire future can be calculated and predicted. The word "initial" implies somehow that we are seeking to identify the conditions that apply at the "beginning" of a system. In fact, initial conditions are often associated with the condition at the time $t = 0$; in truth, this is only a convenient label for the time at which we have a complete specification of the system and does not, of course, denote the origin of time. All that matters is that we have, for a single time point, a complete specification of the system in terms of all the positions and velocities of its components. Given those, we can compute the positions and velocities of the system for all future (and, indeed, past) times, if the system is deterministic.

A good example of such deterministic behavior is that of the planets within our solar system. Since we know the laws of motion of the planets, we can predict their positions for all time, if we only know their positions at one specific time (that we would arbitrarily label as time $t = 0$). The positions of the planets are relatively easy to measure, so determining the initial conditions is not a problem. Equipped with this knowledge, astronomers can make very accurate descriptions of phenomena such as eclipses or the reappearance of comets and meteors (that might or might not collide with the Earth). Applying the same laws further, we can predict when a certain beach will reach high tide or how long it will take before a tsunami hits the nearest coast.

Determinism is also the essential requirement for our ability to engineer electrical and electronic circuits. Their behaviors do not follow from Newton's laws, but they are also deterministic, as are many of the phenomena described by classical physics.

2.1.1 Limits to Modeling

2.1.1.1 Randomness

In the intellectual history of science, determinism was dominant for a long time but was eventually replaced by a statistical view of the world—at least in physics. Firstly thermodynamics and then quantum mechanics led to the realization that there is inherent randomness in the world. The course of the world is, after all, not determined once and forever by its initial conditions. This insight was conceptually absorbed into the scientific *weltanschauung* and, to some extent, into the body of physical theory. Despite the conceptual shift from a deterministic world-view to a statistical one, mathematical modeling, even in the most statistical branches of science, continues *not* to represent the randomness in nature, at least not directly. Take as evidence that the most basic equation in quantum mechanics (the so-called Schrödinger equation) is a deterministic equation, even though it represents fundamentally stochastic, indeterminate, physical phenomena. Randomness in quantum mechanics only enters the picture through the interpretation of the deterministic equation, but the very random behavior itself is not modeled.

To be fair, in physics and physical chemistry there are models that attempt to capture randomness, for example in the theory of diffusion. However, the models themselves only describe some deterministic features of the random system, but not the randomness itself. Mathematical models tell us things such as the expected (or mean) behavior of a system, the probability of a specific event taking place at a specific time, or the average deviation of the actual behavior from the mean behavior. All these quantities are interesting, but they are also deterministic. They can be formulated in equations and, once their initial conditions are fixed, we can calculate them for all times. Mathematics allows us to extract deterministic features from random events in nature. True randomness is rarely seen in mathematical models.

An example might illustrate how inherently stochastic systems can be described in a deterministic way. Think of a small but macroscopic particle (for example, a pollen grain) suspended in a liquid. Observing this particle through a microscope will show that it receives random hits from time to time, resulting in its moving about in the liquid in a seemingly random fashion. This so-called *Brownian motion* cannot be described in detail by mathematical modeling precisely because it is random. Nevertheless, very sophisticated models can give an idea of how far the particle will travel on average in a given time (mean), by how much this average distance will be different from the typical distance (standard deviation), how the behavior changes when force fields are introduced, and so on. Note that the results of this mathematical modeling, as important as they are to characterize the motion of the particle, are themselves deterministic. This does not make them irrelevant; quite the

opposite. The point to take from this discussion is that the description of the random particle is indeed a description of the deterministic aspects of the random motion, and not of the randomness itself.

One could argue that there is not much more we could possibly want to know about the pollen grain beside the deterministic regularities of the system. What good is it to know about the idiosyncracies of the random drift of a particular particle? And yes, perhaps in this case we really only want to know about the statistical regularities of the system, which are deterministic. In that respect, the randomness of Brownian motion is quite *reducible* and we are well served with our deterministic models of the random phenomenon.

Another example of systems where stochasticity is reducible are gases in statistical mechanics. A gas (even an ideal one) consists of an extremely large number of particles, each of which is characterized, at any particular time, by its position and momentum. Keeping track of all these individual particles and their time evolution is hopeless. Collisions between the particles lead to constant re-assignments of velocities and directions. In principle, one could calculate the entire time-evolution of the system if given the initial conditions—ideal gases are deterministic systems. In reality, of course, this would be an intractable problem. Moreover, the initial conditions of the system are unknown and unmeasurable.

As it turns out, however, this impossibility of describing the underlying motion of particles in gases is not a real limitation. All we really need to care about are some stochastic regularities emerging from the aggregate behavior of the colliding molecules. Macroscopically, the behaviors of gases are quite insensitive to the details of the underlying properties of the individual particles and can be described in sufficient detail by a few variables. It is well known that an ideal gas in thermal equilibrium can be described simply in terms of its pressure, volume and temperature:

$$PV \propto T$$

There is no need to worry about all the billions of individual molecules and their interactions. We do not need to know where every molecule is at any given time. All we care about is how fast they are on average, the probability distribution of energies over all molecules, and the expected local densities of the gas. Once we have those details we have reduced its random features to deterministic equations.

This approach of reducing randomness really only works if the individual random behavior of a specific particle in our system is unimportant. In physics, this will often be the case, but there are systems that are *irreducibly random*; systems where the random path of one of its components *does* matter and a deterministic description of the system is not enough. One example is natural evolution in biological systems. Random and unpredictable mutations at the level of genes cause changes at the level of the phenotype. Over evolutionary time scales, there will be many such mutations, striking randomly, and often leading to the demise of their bearer. Sometimes a mutation will have no noticeable effect at all but, on rare occasions, a mutation will be beneficial and lead to an increase in fitness.

Imagine now that we wish to attempt to model this process. If we tried to come up with a deterministic model of mutations and their effects, after much effort and

calculation we would perhaps know how beneficial mutations are distributed, for how long we need to wait before we observe one, and so on. This might be what we want to know, but maybe we want to know more. Imagine that we would like to model the actual evolution of a species; that is, we are interested in creating a model that allows us to study how and when traits evolve and what these traits are. Imagine that we want to model the evolutionary arms race between a predator and its prey. Imagine that we want to have a model that allows us to actually see evolution in action. In this case, we do not care about all the mutations that led nowhere. All we are interested in is those few, statistically insignificant events that resulted in a qualitative change in our system: a new trait or defense mechanism, for instance. What we would be interested in are *the particulars* of the system, not its general features.

Mathematical models are normally not very good at representing particulars of random systems; different approaches are required if we are interested in those. We can say, therefore, that if our system is irreducibly random then mathematical approaches will be limited in their usefulness as models.

2.1.1.2 Heterogeneity

Another aspect of natural systems that limits mathematical tractability is system *heterogeneity*. A system is heterogeneous if: (i) it consists of different parts, and (ii) these parts do not necessarily behave according to the same rules/laws when they are in different states. As a simple example, one can think of structured populations of animals, say lions. Normally lions live in packs, and each pack has an intrinsic order that determines how individuals act in the context of the entire group. This group structure has evolved over time and is arguably significant for the survival of lions. It is also difficult to model mathematically.

One could try to circumvent this and ignore the detail. Often such an approach will be successful. If one wanted to model the population dynamics of lions in the Serengeti, it may be sufficient to look at the number of prey, the efficiency with which prey is converted into offspring by lions, and the competition (in the form of leopards, cheetahs, etc.). With this information, we might then formulate a reasonable model of how the lion population will develop over time in response to various environmental changes, despite the model having ignored much of the structure of lion populations.

The interactions between the individual animals would be difficult to model in a mathematical model. Lions behave differently depending on their age, their rank within the group and their gender. Keeping track of this in a set of equations would quickly test the patience and skill of the modeler. Moreover, lions reproduce and die.

Equations don't seem to be able to offer a good approach in this case. At the same time, it is not unthinkable that one may want to model the behavior of packs of lions formally. One might, for example, be interested in the life strategies of lions and complement empirical observations with computer models. The heterogeneity of the pack is irreducible in such a model. It would make no sense to assume an "average"

lion with some “mean” behavior. The dynamics of the packs and how the behavior patterns contribute to the evolutionary adaptability of lions rest crucially on the lions being a structured population. We are not aware of attempts to model life histories of lions, but in the context of social insects and fish there have been many attempts to use computer models to understand group dynamics (see [1,2]); but never have these studies used purely equation-based approaches.

While we can clearly see that most systems are heterogeneous, often they are *reducibly* so. The difference between the component parts can often be ignored and be reduced to a mean behavior while still generating good and consistent results. Whether or not a system is reducibly or irreducibly heterogeneous depends on the particular goals and interests of the modeler and is not a property of the system *per se*.

In physics, the heterogeneity of most problems is reducible, which has to do with the type of questions physicists tend to ask. In biology things are different. Many of the phenomena bioscientists are interested in are essentially about heterogeneity. Reducing this heterogeneity is often meaningless. This is one of the reasons why it has been so difficult to base biology on a mathematical and formal theoretical basis, whereas it has been so successful in physics. Irreducible heterogeneity makes mathematical modeling very difficult and life consequently harder.

2.1.1.3 Interactions

Finally, a third complicating property of systems is component interaction. Unlike heterogeneity and randomness, interactions have been acknowledged as a problem in physics for a long time. In its most famous incarnation this is known as, “the *n*-body problem”. Theoretical physics has the tools to find general solutions for the trajectories of 2 gravitating bodies. This could be two stars that are close enough for their respective gravitational fields to influence each others’ motions. What about three bodies? As it turns out, there is no nice (or even ugly) formula to describe this problem; and the same is true for more than three bodies.

Interaction between bodies poses a problem for mathematical modeling. How do physicists deal with it? The answer is that they do not! In the case of complex multi-body interactions it is necessary to solve such problems numerically. Furthermore, there are cases of systems that are so large that one can only approximate the interaction between parts with a “mean field”, which essentially removes any individual interactions and makes the system solvable. There are many cases of such *reducible interactions* in physics, where the mean-field approximation still yields reasonable results. Before the advent of computers, irreducible interactions were simply ignored because they are not tractable using clean mathematical approaches.

In biology, there are few systems of interest where interactions are reducible. Nearly everything in the biological world, at all scales of magnification, is interaction: be it the interactions between proteins at a molecular level, inter-cell communications at a cellular level, the web of interactions between organisms at the scale of ecology and, of course, the interaction of the biosphere with the inanimate part of our world. If we want to understand how the motions of swarms of fish are generated through

the behavior of individual fish, or how cells interact to form an embryo from an unstructured mass of cell, then this is irreducibly about interactions between parts.

The beauty of mathematical modeling is that it enables the modeler to derive very general relationships between variables. Often, these relationships elucidate the behavior of the system over the entire parameter space. This beauty comes at a price, however: the price of simplicity. If we want to use mathematics then we need to limit our inquiry to the simplest systems—or at least to those systems that can be reduced to the very simple. If this fails then we need to resort to other methods, for example computer models. While computational models tend not to satisfy our craving for the pure and general truths that mathematical formulas offer, they do give us access to representations of reality that we can manipulate to our pleasure.

In a sense, computational models are a half-way house between the pure mathematical models as they are predominantly used in theoretical physics, and the world of laboratory experimentation. Computer models are formal systems, and thus rigorous, with all the assumptions and conditions going into the models being perfectly controllable. Experimentation with real systems often does not provide this luxury. On the other hand, in a strict sense, computer models can only give outcomes for a particular set of parameters, and make no statement about the parameter space as a whole. In this sense, they are inferior to mathematical models that can provide very general insights into the behavior of the system across the full parameter space. In practice, the lack of generality is often a problem, and it makes computer models a second-best choice—for when a mathematical analysis would be intractable.

2.2 Agent-Based Models

The remainder of this chapter focuses on a particular computer modeling technique—*agent-based modeling* (ABM)—which can be very useful in modeling systems that are irreducibly heterogeneous, irreducibly random and contain irreducible interactions. The principle of an ABM is to represent *explicitly* the heterogeneous parts of a system in the computer model, rather than attempting to “coarse grain” it. In essence, this is achieved by building a virtual copy of the real system [3]—the model represents components of the real system explicitly and keeps track of the behavior of individuals over time. So, in a sense, each individual lion of the pack would have a virtual counterpart in the computer model. As such, ABMs are quite unlike mathematical models, which represent components by the values of variables rather than by behaviors. In an ABM, the different components (the “agents”) represent entities in the real world system to be modeled. As well as the individual entities, an ABM also represents the environment inhabited by these entities. Each of these modeled entities has a *state* and exhibits an explicit *behavior*. An agent can interact with its environment and with other entities.

The behavior of agents is often “rule-based.” This means that the instructions are formulated as *if-then* statements, rather than as mathematical formulas. For example, in a hypothetical agent-based model of a lion, one rule might be:

If in hunting mode and there is a prey animal closer than 10 meters, then attack.

A second rule would likely determine whether the lion should enter hunting mode:

If not in hunting mode, and T is the time since the last meal, then switch into hunting mode with probability $(T_{\max} - T)/T_{\max}$.

This second rule clearly contains a mathematical formula, which illustrates that the distinction between rules and mathematical formulas is somewhat fuzzy. Rules are not always purely verbal statements but often contain mathematical expressions. However, what is central to the idea of rule-based approaches is that ABMs never contain a mathematical expression for the behavior of the system as a whole. Mathematical expressions are a convenient means to determine the behavior of individual constituent parts and their interactions. The behavior of the system as a whole is, therefore, *emergent* on the interaction of the individual parts.

The underlying principle of an ABM is that the behavior of agents is traced over time and observed. The approach is thus very different from equation-based mathematical models that try to capture directly the higher-level behavior of systems. ABMs can be thought of as *in silico* mock-ups of the real system. This approach solves the representational problems of standard mathematical models with respect to irreducible interaction, randomness and heterogeneity. An explicit representation of random behavior is unproblematic in computer models: (pseudo) random number generators can be used to implement particular instances of random behaviors and to create so-called sample trajectories of stochastic systems, without the need to reduce the stochasticity to its deterministic aspects. Similarly, while it is often difficult to represent irreducible heterogeneity in mathematical models, in ABMs this aspect tends to arise naturally from the behavioral rules with which the agents are described—distinct individuals of the same agent type naturally end up in distinct states at the same time. There is a similar effect with respect to interactions.

The major drawback of ABMs is their computational cost. Depending on the intricacy of the model, ABMs can often take a very long time to run. Even when run on fast computers, large models might take days or weeks to complete. What exacerbates this feature is that the result of an individual model run is often shaped by stochastic effects. This means that the outcome of two simulation experiments, even if they use the same configuration parameters, may be quite different. It is necessary, therefore, to repeat experiments multiple times in order to gain (statistical) confidence in the significance of the results obtained.

Altogether, ABMs are a mixed blessing. They can deal with much detail in the system, and they can represent heterogeneity, randomness and interactions with ease. On the other hand, they are computationally costly and do not provide the general insights that mathematical models often do.

2.2.1 The Structure of ABMs

Let us now get to the business of ABM modeling in detail. ABMs are best thought of in terms of the three main ingredients that are the core of every such model:

- the agents;
- the environment inhabited by the agents;
- the rules defining how agents interact with one another and with their environment.

2.2.1.1 Agents

Agents are the *raison-d'être* of any ABM, representing the entities that act in the world being modeled. These agents are the central units of the model and their aggregate behavior will determine its outcome. In an ABM of a pack of lions, we would most likely choose the individual lions as agents. Models will often have more than one type of agent with, typically, many instantiations of each particular type of agent. For example our model may have the agent types: lion, springbok, oryx, etc., and there will exist many instances of each in the running model. A particular agent type is characterized by the set of internal states it can take, the ways it can impact its environment, and the way it interacts with other agents of all types, including its own. A lion's state, for example, might include its gender, age, hunger level and whether it is currently hunting; its interactions would likely include the fact that it can eat oryxes and springboks. On the other hand, a springbok's interactions would not involve it preying on either of the other two species. In a model of a biochemical system the agents of interest might be proteins, whose internal state values could represent different conformations. Depending on the model, the number of internal states for a particular type of agent could be very large. While all agents of a specific type share the same possible behaviors and internal states, at any particular time agents in a population may differ in their actual behaviors. So, when hunted by a lion, one oryx may get eaten while another escapes; one lion is older than another, and so on.

In the context of ABMs in biology, it is often useful to limit the life-time of agents. This requires some rules specifying the conditions under which agents die. In order to avoid the population of agents shrinking to zero, death processes need to be counterbalanced by birth or reproduction processes. Normally the reproduction of agents is tied to some criterion, typically the collection of sufficient amounts of "food" or some other source of energy. Models with birth and death (more generally: creation and destruction) processes allow a particularly interesting kind of effect: If the agents can have hereditary variations of their behavior (and possibly their internal states) then this could make it possible to model evolutionary processes. In order for the evolutionary process to be efficient, one would need one more feature—competition for resources. Resources need not necessarily be nutrient, but could be space, or even computational time. An example of the latter is the celebrated simulation program *Tierra* [4], where evolving and self-reproducing computer programs compete against one another. Each program is assigned a certain amount of CPU-time and needs

to reproduce as often as possible within the allotted time. The faster a program reproduces, the more offspring it has. This, together with mutations—i.e., random changes of the program code—leads to very efficient reproducers over time.

There have been many quite successful attempts to model evolutionary processes mathematically. These models usually make some predictions about how genes spread in a well defined population. Such mathematical predictions of the behavior of certain variables in an evolutionary process are very different from the type of evolutionary models that are possible in ABMs. In a concrete sense, evolution always depends on competition between variants. The variants themselves are the result of random events, i.e., mutations. The creation of variants in evolutionary models requires an explicit representation of randomness (rather than a summary of its statistical properties); and the bookkeeping of the actual differences between the variants requires the model to represent heterogeneity. Agent-based modeling is the ideal tool for this. If we think again of lions, we could imagine a model that allows each simulated lion to have its own set of hunting strategies. Some lions will have more effective strategies than others, an effect which could drive evolution if lions that are more successful have more offspring.

2.2.1.2 Environment

Agents must be embedded in some type of environment, that is, a space in which they exist. The choice of the environment can have important effects on the results of the simulation runs, but also on the computational requirements of the model. How to represent the environment and how much detail to include will always be a case-specific issue that requires a lot of pragmatism. In the simplest case, the environment would be simply an empty, featureless container with no inherent geometry. Such featureless environments are often useful in models that assume so-called “perfect mixing” of agents. More on that later.

A simple, featureless space can be enhanced by introducing a measure of distance between agents. The distance measure could be discrete, which essentially means that there are compartments in the space. In ABMs it is quite common to use 2-dimensional grid layouts. Agents within the same compartment would be considered to be in the same real-world location. A further progression would be to introduce a continuous space, which defines a real-valued distance between any pair of agents. For computational reasons, continuous spaces used in practice are often 2-dimensional, but there is nothing preventing a modeler from using a 3-dimensional continuous space if required, and computational resources permit.

In ABMs, spatially-structured models are usually more interesting than completely featureless environments. The behavior of many real systems allows interactions only between agents that are (in some sense) close to one another. This then introduces the notion of the *neighborhood* of an agent. In general an agent *A* tends to interact with only a subset of all agents in the system at any particular time. This subset is the set of its neighboring agents. In ABMs “neighborhood” does not

necessarily refer simply to physical proximity; it can mean some form of relational connectedness, for instance. An agent's neighborhood need not be fixed but could (and normally will) change over time. In many ABMs, the only function of the environment is to provide a proximity metric for agents, in which case it is little more than a containing space rather than an active component. There are environments that are more complex and, in addition to defining the agent-topology, also engage in interactions with agents. A common, albeit simple example, is an environment that provides nutrients. Sometimes, very detailed and complicated environments will be necessary. For, example, if one wanted to model how people evacuate a building in an emergency, then it would be necessary to represent the floor-plans, stairs, doors and obstructions. There have been attempts to model entire city road systems in ABMs, in order to be able to study realistic traffic-flow patterns. A description of such large modeling attempts would go beyond the purpose of this book, and the interested reader is encouraged to read Casti's "Would-be Worlds" [3].

2.2.1.3 Interactions

Finally, the third ingredient of an ABM is the rules of action and interactions of its agents. In each model, agents would normally show some type of activity. Conceptually, there are two types of possible activities of agents: (i) taking actions independent of other agents, or (ii) interacting with other agents. In the latter case, agents are generally restricted to interacting with only their neighbors. The behavior rules of agents are often rather simple and minimalistic. This is as much a tradition in the field as it is a virtue of good modeling. Complicated interactions between agents make it hard to understand and analyze the model, and require many configuration parameters to be set. In most applications it is a good idea to try to keep interactions to the simplest possible case—certainly within the earliest stages of model development.

2.2.2 Algorithms

Once the three crucial elements of an ABM (the agents, the environment, and the interactions) have been determined, it still needs to be specified how and under which conditions, the possible actions of the agents are invoked. There are two related issues that we consider when modeling real systems: the passage of time and concurrency.¹

In real-world systems, different parts usually work concurrently. This means that changes to the environment or the agents happen in parallel. For example, in a swarm of fish, every individual will constantly be adjusting its swimming speed and direction to avoid collisions with other fish in the shoal; lions hunting as a pack attack together; in a cell some molecules will collide, and possibly engage in a chemical reaction,

¹Concurrency denotes a situation whereby two or more processes are taking place at the same time. It is commonly used as a technical term in computer science for independent, quasi-simultaneous executions of instructions within a single program.

while, simultaneously, others disintegrate or simply move randomly within the space of the cell.

Representing concurrency of this sort in a computer simulation is not trivial. Computer programs written in most commonly-used programming languages can only execute one action at a time. For example, if we have 100 agents to be updated (or to perform an action), this would have to be done sequentially as follows:

1. Update agent number 1.
2. Update agent number 2.
3. ...
100. Update agent number 100.

Clearly, if the action of one agent depends on the current state of one or several other agents, then sequential implementation of what should be a concurrent update step may mean that the outcome at the end of the step might be affected by the order in which the agents are updated. This should not be the case. In order to simulate concurrent update it is therefore necessary that the state of agents at the beginning of an update cycle is remembered, and all state-dependent update rules should refer to this. Once all agents have been updated the saved state can be safely discarded.

For managing the passage of time, there are essentially two choices: time-driven and event-driven. The simplest is time-driven, which assumes that time progresses in discrete and fixed-length time steps, or “ticks”, in an effort to approximate the passage of continuous time. On every tick, each agent is considered for update according to the changes likely to have happened since the previous time step. For instance, if the time step is one day in the lion model, then the hunger-level of each lion might be increased by one unit, possibly leading to some lions switching into hunting mode. The real-world length of the time step is obviously highly model dependent; it could be nanoseconds for a biochemical reaction system or thousands of years for an astrophysics model. One of the skills of the modeler will be choosing an appropriate time-step size that is small enough to capture all significant effects of interests, but not so small that the model’s runtime is prohibitively long.

The event-driven approach also models a continuous flow of time but the elapsed time between consecutive events is variable. For instance, one event might take place at time $t = 1$, the next at $t = 1.274826$, and the next at $t = 1.278913$ —or at any time in-between. *Event-driven algorithms* are appropriate in models of scenarios where events occur naturally from time to time rather than periodically. This is the case, for example, in chemical-reaction systems. Individual reactions occur at system-dependent points in continuous time, and between two reactions nothing of interest is considered to take place.

Both time-driven and event-driven approaches naturally lead to their own distinctive implementations in computer models and we illustrate both, along with further detail, in the following sections.

2.2.3 Time-Driven Algorithms

Time-driven algorithms (Algorithm 1) are also often referred to as *synchronous* update models, in contrast to event-driven algorithms being referred to as *asynchronous*. A synchronous updating algorithm approximates continuous time by a succession of discrete time steps in which all agents are considered for update. Synchronous updating is inexact and can only approximate real-world time. Depending on the specific application, this could make a significant difference to the result, although in many cases the approximation is very good. It is also often the case that the modeler does not care about quantitative correctness and synchronous updating becomes a choice of convenience.

Algorithm 1 Update scheme for synchronous/time-driven ABM.

```

Set initial time.
Set initial conditions for all agents and the environment.
loop
  for All agents in the model do
    Invoke update rule.
  end for
  Increment time to next time step.
end loop

```

In those cases where quantitative accuracy is important, the size of the chosen time step determines the temporal granularity of the simulation, and hence the accuracy of the simulation. In essence, synchronous models chop time up into a sequence of discrete chunks; the finer the chunks the better the resolution of events. In the limiting case, where each time step corresponds to an infinitesimal amount of time, synchronous algorithms would be precise. In each time step the updating rule would then simulate what happens in the real system during an infinitely short moment of time. As one can easily see, in this limiting case it would take an infinite number of time steps to simulate even the shortest moment of real time. In practice, this limit is therefore unattainable. We cannot reach it, but we can choose to make our time intervals shorter or longer depending on the desired accuracy of the simulation. The trade-off is usually between numerical accuracy of the model and the speed of simulation. The shorter the time interval, the longer it takes to simulate a given unit of real time.

Take as an example a fish swarm. If we simulate a shoal of fish using synchronous update, then each time step would correspond to a given amount of physical time (i.e., fish time). During each of these intervals the fish can swim a certain distance. Given that we know how fast real fish swim, the distance we allow them to travel per time step defines the length of an update-step in the model. The shorter the distance, the shorter the real-world equivalent of a single update step. In practice, the choice of the length of an update step can materially impact on the behavior of the model.

Real fish in a shoal continually assess their distance to their neighbors and match speed and direction to avoid collisions and to avoid letting the distance to the neighboring fish become too large. If we simulate a swarm of fish using a synchronous updating algorithm, then each fish will assess its relationship to its neighbor at discrete times only. If we now choose our temporal granularity such that a fish swims about a meter between update-steps, then this will result in very inaccurate models. A meter is very long compared to typical distance between fish. It is therefore quite likely that after an update step two animals may overlap in space. Between two time steps the nearest neighbors may change frequently. On the whole, the behavior of the simulated swarm is likely to be very different from the behavior of a real school of fish and the model a bad indicator of real behavior. The temporal resolution is too crude.

The accuracy of the overall movements will be increased if we decrease the “size” of each update step. If we halve the time step, say, then the forward distance will then be only 0.5 m per time step. The cost would be a corresponding doubling of the number of update steps, in the sense that for every meter of swarm movement we now need to update two cycles rather than just one. One could, of course, make the model even more accurate with respect to the real system if each fish only swam 1 cm per time step at the cost of a corresponding increase in update steps and run time. The “right” level of granularity ultimately depends upon the level of accuracy required and the computing resources available.

It will not be the case with all models that every agent is updated similarly on each time step. The heterogeneous nature of some systems means that differential updating is quite likely. Where this is the case, care should also be taken with the time step. It should not be so large that a lot of agents are typically updated at each time step, because that risks missing subtleties of agent changes that might otherwise cause significant effects in the model with smaller steps. A corollary, therefore, is that we should expect many agents to remain unchanged at each individual time step, but that changes gradually accumulate within the population over many time steps.

There is a balance to be struck between the algorithmic cost of considering every agent for possible update at each time step, and the number actually needing updating—which might turn out to be none on some cycles. Where it is possible to identify efficiently only those agents to be updated on a particular cycle then those costs may be mitigated, but the event-driven approach should also be considered in such cases.

2.2.4 Event-Driven Models

The basic principle behind event-driven algorithms is that they are controlled by a *schedule*. A schedule is, in effect, a list of pending future events, typically maintained in sorted order of when each event is due to occur. Such events include the updating of agents, the creation of new agents, and also changes to the environment. All the program does then is to follow the list of updates as specified in the schedule (Algorithm 2). In addition, there needs to be a way for future events to be added to

the schedule as, typically, the occurrence of an event leads to the spawning of one or more future events.

Algorithm 2 Update scheme for asynchronous/event-driven ABM.

Set the model time: $t = 0$

Set the initial conditions for all agents and the environment.

loop

Update the model time to that of the next event.

Determine which agent(s) will be updated next and which update rule will be applied.

Apply the selected rule to the selected agent(s).

Update the schedule.

end loop

Let us illustrate this process via a simple example where the focus will primarily be on the association of events and their time rather than the individual agents. Consider a system model consisting of at least two types of molecule, and that these molecules can engage in reactions with one another. Let us assume that reactions between molecules of the two types happen with a rate of 1. This means that, on average, there will be one reaction per time unit. We will assume that the system we are modeling operates in continuous time. However, continuous time is difficult to implement in computers so we use an approximation: We define one update step of the model as representing the passing of one time unit. We can then naively simulate our system as follows:

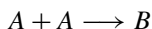
1. Randomly choose one molecule of each type.
2. Simulate the reaction (this could, for example, be the production of a third type of molecule).
3. Update the time of the model by one.

The system exhibits precisely the behavior we expect, namely that it has one reaction per time unit. We could even, again rather naively, extend this scheme to a reaction rate of 2. In this case we would then execute two reactions per update step.

For some applications, this algorithm would be acceptable, but it will never be an accurate model of the real system. If the reaction rate is 1 per time unit *on average* then this does *not* mean that in any given time unit there will be exactly one reaction. Instead, during a particular time unit of observation there might be two reactions, whereas during the next three time units there might be no reactions. From a simulation point of view, the problem is how to work out how many events actually happen during any given time unit. While it is possible to do this but, in practice, it turns out that there is a better approach.

Instead of calculating how many events take place in any given time interval, there are event-driven algorithms to calculate, for each reaction, at what time it next takes place. In the case of chemical reaction systems, this time can be determined by

drawing a random number from an exponential distribution with the mean equal to the reaction rate. Making the example concrete: assume a very simple chemical system consisting initially of $N_A = n_0$ molecules of type A and $N_B = 0$ molecules of type B . Assume, further, that these molecules are embedded in a featureless environment—there is no sense of distance between them—and that the molecules engage in a single chemical reaction:



This means that two molecules of A are used up to produce one molecule of B . Over time, we would expect that the system approaches a state where there are only B molecules.

Assume that we start at time $t = 0$. The question we have to ask now is: When does the next reaction occur? For this we need to draw a random number, in this case from an exponential distribution.² The mean of the distribution will be chosen based on the current value of N_A and the reaction rate. Let us assume that the random number we draw determines $\Delta t = 1.0037$. We now set the new time to $t = t + \Delta t = 1.0037$ and use the update rule which, in this case, is simply to destroy a pair of A molecules and create a new molecule of type B . We would need to choose which pair of A -agents is used up in the reaction and, for the sake of simplicity, let us assume that we simply draw a random pair of A -agents. That was the first step.

The second step is identical. We need to determine a new time, by drawing a random number from an exponential distribution. There is one complication here, namely that the new N_A is $n_0 - 2$ because two of the original A molecules have been used up to form the new B . In real applications, this effect will be important, but for the moment we will ignore it. A detailed treatment of this will be given in Chap. 7. For the moment we will simply note that we need to draw a new random number from an exponential distribution with an appropriately updated mean. This generates a new Δt , say $\Delta t = 0.873$. We set the new time to $t = t + \Delta t = 1.0037 + 0.873 = 1.8767$ and update the relevant agents again. We continue in this fashion until we hit a stopping condition, which will normally be a fixed end time of the model (or the end of the patience of the modeler).

Notice that, here, because there is only one rule and one type of event, we do not even need to maintain a schedule of future events. In general, the situation will be more complicated, because typically there will be more than just two molecular species and more than one possible reaction.

This particular example is somewhat artificial in that ABMs are not the best way to simulate such systems. There is no irreducible heterogeneity in the system, in the sense that all molecules of type A are indistinguishable in their state and identity. It is therefore not necessary to simulate each of the molecules individually. Chapter 7 will provide a detailed discussion of how to simulate chemical systems where there is no heterogeneity between the individual molecules.

²There are specialized algorithms available to do this, and there will be no need to re-implement them. For programmers of C/C++ the Gnu Scientific Library [5] is one place to look.

In those practical cases where event-driven ABMs are necessary, the details of how the schedule is created and maintained will very much depend on the particular demands and properties of the system to be modelled. It is then essential to think carefully about how to choose the next events and to correctly calculate how much time has passed since the most recent event. If these rules are chosen properly, then event-driven algorithms can simulate continuous time.

2.3 Game of Life

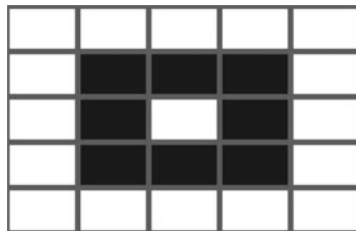
ABMs have been found to be useful over a wide range of application areas. They have been used to study the movement of indigenous tribes in the Americas [6], how sand piles collapse [7], how customers move through supermarkets [8] and even to model the entire traffic system of Albuquerque in the US state of New Mexico [3]. So far, perhaps the greatest following of ABMs is among economists. Agent-based Economics [9] is a field that uses ABMs to study economic systems in a new way.

In this book, we will not re-trace the historical roots of ABMs, although we will look at one of the examples of an ABM that was important in generating initial interest in this modeling technique in the scientific community: *cellular automata* (CA). We will examine a particularly interesting example of a CA that is well known and, to this day, continues to keep researchers busy exploring its properties: the Game of Life. However, we will not present a complete exposition of CAs in all their shapes and varieties, and the reader who would like to explore further is referred to the excellent book by Wolfram [10].

Despite its suggestive name, the Game of Life is not really a model of anything in particular. Rather, it is a playground for computer scientists and mathematicians who keep discovering the many hidden interesting features this model system has to offer. Unlike mathematicians and computer scientists, natural scientists are normally interested in models because they can help them understand or predict something about real systems. The Game of Life fails in this regard, but it is still interesting as an illustration of the power of ABMs, or more generally, how interactions can lead to interesting and complex behaviors. A Java program implementing the game is provided with the materials associated with this book at <http://www.cs.kent.ac.uk/imb/>.

The Game of Life is played in an environment that is a two-dimensional grid (possibly infinitely large). Agents can be in one of only two possible states, namely ‘1’ or ‘0’ (in the context of the Game of Life these states are often labeled “alive” and “dead”, but ultimately it does not matter what we call them). This particular CA is very simple and the range of actions and interactions of agents is very limited: agents have a fixed position in the grid, they do not move, and they have an infinite life-span (despite the use of “dead” as a state name). They have no internal energy level or age, for instance, and time plays no significant role in the model. The most complex aspect of an agent is its interaction rules with other agents. These rules are identical for all agents (as we have only one type of agent) and depend solely on

Fig. 2.1 A configuration in the Game of Life. The *black cells* are the Moore neighborhood of the central *white cell*



an agent's own state and that of its neighbors. In the Game of Life, neighbors are defined as those agents in the immediately adjacent areas of the grid. Each agent has exactly 8 neighbors corresponding to the so-called *Moore neighborhood* of the grid (Fig. 2.1). The complete set of rules governing the state update of an agent is as follows:

- Count the number of agents in the neighborhood that are in state 1.
- If the number of neighbors in state 1 is less than two or greater than three then the agent's new state will be 0.
- If the agent is in state 1 and it has two or three neighbors in state 1 then it will remain in state 1.
- If the agent is in state 0 and it has two neighbors in state 1 then it will remain in state 0.
- If the agent is in state 0 and it has three neighbors in state 1 then it will switch to state 1.

The global rule is that the state of all agents is updated synchronously.

Note that there is no random element to this behavior—the Game of Life is completely deterministic. Once it is determined which agents on the grid are in state 1 and which are in state 0, then the behavior of the simulation is implicitly determined thereafter. Many, or perhaps most, of the initial conditions will lead to trivial behavior. For example, if we start with all agents being in state 1 then, according to the rules above, at the second time step all agents (in an infinite grid) will have switched to state '0', and they will all remain in that state thereafter. Most initial conditions will eventually end up in a similarly trivial state. But then again, given the simple rules of the system, what more do we expect?

It is intriguing, therefore, that there are some configurations whose time evolution is far from trivial. There are a number of initial conditions that lead to very interesting and diverse long term behaviors, with new ones continuously being added to the list. Let us look at a couple of the best known ones. We will here concentrate on local configurations only; that is, we will assume that at time $t = 0$ all agents are in state 0, except for a few that make up the local configuration.

One of the most famous initial conditions of interest is the so-called “blinker.” This is a row (or column) of three agents in state 1. Say, we start with a row configuration:

```

0 0 0
1 1 1
0 0 0

```

(remember here we assume that all other agents are in state 0). The middle agents in the top and bottom row are both in state 0 and both have exactly three neighbors in state 1. According to the rules of the Game of Life, these agents now switch to state 1. Similarly, in the middle row, the first and the third agent only have one neighbor in state 1. Following the rules, they will switch to state 0; the central agent in the middle row will not change state.

Synchronous update means that the states of agents are only changed once the new state for each agent has been calculated and the new configuration will be:

```

0 1 0
0 1 0
0 1 0

```

The reader can easily convince herself that updating this configuration will lead back to the original horizontal configuration. The blinker will endlessly move back and forth between these two configurations in the absence of intervention from any other surrounding patterns.

The Game of Life illustrates very well the principle of synchronous updating of agents. At every time step the agents should be updated according to the neighborhood they had at the beginning of the update step. Yet the computer program needs to update one agent after the other. The risk is that, as each agent is updated, the environment for its neighbors changes as well, with the result that the outcome of the update step depends on the order in which the agents are updated. This problem has been touched upon in more general terms above. In order to avoid this dependence on the order of update, the full state of the grid must be saved at the beginning of a time step; any application of update rules must be made with respect to this saved copy.

The importance of this can be exemplified using the blinker. Instead of the synchronous update algorithm, let us assume an asynchronous one, that updates agents in order from the top left of the grid to the bottom right. That is, we go from left to right until we hit the end of the grid and then we start again on the left of the next line. The original blinker configuration would lead to the following end state in the next time step:

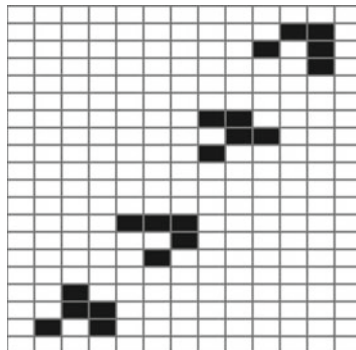
```

0 1 1
1 0 1
0 0 0

```

2.1 Check on paper that failing to keep the old states and new states distinct during the update step leads to the result we have shown.

Fig. 2.2 A partial glider sequence in the Game of Life. The four distinct configurations show elements of the sequence a glider would undergo while moving diagonally north-eastwards (intermediate stages omitted)



In the standard Game of Life, the blinker configuration oscillating forever is perhaps not the epitome of an interesting configuration. However, there is more in store. Another important configuration is the “glider,” a structure with this initial condition³:

```

1 1 1
1 0 0
0 1 0

```

The glider represents a local pattern of states of agents that propagates through the environment diagonally (Fig. 2.2). Note that it is not actually the agents that are moving but the configuration pattern of states. Apart from being fun to watch, gliders can be thought of as transmitting information from one part of the grid to another—information that is relayed from one group of agents to another. Gliders are often used as essential parts in many more complex configurations in the Game of Life. There have even been successful attempts to build entire computing machines purely from initial configurations in the Game of Life. In some versions of these configurations gliders are used as channels to transmit basic units of information between the parts of the “computer”.

2.2 Start with a glider pattern and use the rules to update it by hand for a few time steps.

2.3 Check, by hand, the effect of different asynchronous update rules on the glider.

2.4 What happens if two gliders collide?

2.5 If Java is available on your system, try using the implementation provided with the materials associated with this book to experiment with different starting patterns, including the blinker and glider.

³Alternative symmetrical configurations would obviously be equivalent.

As far as we are concerned here, the Game of Life is an example of an ABM that allows for only limited actions and interactions between agents, but can still produce complex and interesting behaviors. Unfortunately, it is not really useful beyond being a playground for mathematician. Nevertheless, it does illustrate the principles of agent-based modeling—agents with state and rule-based behavior interacting in a structured environment—and shows that even simple interactions can lead to complex behavior.

The Game of Life also illustrates how ABMs can deal with irreducible heterogeneity. All agents in the system are the same, and only differentiated from one another by their state. Modeling this mathematically would be nearly impossible, but it is easy to implement as a computer model.

2.4 Malaria

The Game of Life is one of the simplest possible non-trivial ABMs; it is an interesting world to explore, but it does not have any immediate use as a model of *something real*. ABMs as models of some aspects of the real world need to be more advanced than the Game of Life. In this section we will discuss a somewhat more advanced (and perhaps practically more useful) example of an ABM, modeling the transmission of malaria. Modeling diseases and their spread is an important thing to be able to do. Models of disease transmission could, at least if they are accurate enough, be used in a variety of contexts, such as in optimizing responses to emerging epidemics, or for efficient planning of immunization programs, particularly when resources are limited.

Particularly in sub-Saharan Africa, Malaria is the single biggest killer, ahead even of HIV/AIDS. The disease is *vector borne*; that is, it is transferred from person to person by a secondary agent, the male *Anopheles* mosquito (the female is not a vector of malaria). An infection is transferred both from human to mosquito and *vice versa* when *Anopheles* bites a human.

The dynamics of infection depend on the local interactions between individual agents, namely the vectors (mosquitoes) and the humans. Both humans and mosquitoes have a limited range of movement and can only interact when in each other's neighborhood. The exact details of how agents move across their available space—in particular the mosquitoes—could have a major influence on how fast and how far malaria spreads; in this sense there are irreducible interactions. The system is also irreducibly heterogeneous in that some agents are infected while some are susceptible (that is uninfected). There is a random element to the model in that the contact between the agent-types is probabilistic. As will become clear below, this randomness and heterogeneity is, under some conditions, reducible, but irreducible under others. In order to deal with the latter case, an ABM is appropriate.

Let us now leave the preliminaries behind and put into practice what we have discussed in theory. Our aim is to find the simplest non-trivial model of malaria that captures the essence of reality. For the reasons we outlined in Chap. 1, in every

modeling enterprise one should always try to find the simplest model first, and only introduce complexities once this simplest model is thoroughly understood. There are many possible choices one can take when modeling the spread of malaria. The shape and form of the final model will depend on the particular goals that are to be achieved. Depending on whether they are predictive, explanatory or exploratory, different requirements will be placed on the model. Here we do not have any particular purpose in mind, other than demonstrating how to design ABMs, and we will therefore design the simplest non-trivial model of the spread of malaria. Generally, it is a good idea to start in this way, even when one has a particular purpose in mind. It is a common mistake for modelers to start designing models that are full of details but are also opaque. Particularly with ABMs, this danger is ever-present.

As a minimal requirement, an ABM of the transmission dynamics of malaria will have to consider at least two types of agent, namely mosquitoes and humans. In a first model, it is best to avoid the complication of distinguishing between female and male mosquitoes, although for some applications this could become relevant. Similarly, humans could be differentiated according to age, as different age-groups will likely have different susceptibilities for the disease and different survival rates. However, for a first attempt it is best to ignore these complicating effects as well. Instead, to keep matters simple for an initial model, we reduce the possible states of the agents to two: either an agent is infected or not. We have to assume that both types of agent could be either infected or not, and these would be the only two states of the agents. The difference between human-agents and mosquito-agents is in the details of the rules that determine how they carry an infection, and for how long they stay infected. To keep the complexity of the model to a minimum, one would initially also ignore that agents die (of malaria or otherwise). This means that there is a fixed number of agents in the environment. Each agent can be either infected or susceptible. Naturally, we have to assume at least one interaction between mosquitoes and humans, namely cross-infection; the model needs to have some mechanism whereby mosquitos can infect humans, and vice-versa. Cross-infection can only happen between neighboring agents (where neighboring refers to some spatial proximity) and between agents of different type.

A feature that we cannot ignore, even in the initial simple model, is the spatial structure of the population. Agents need to be embedded in an environment. In the present case the simplest way to represent an environment is to assume a 2-dimensional space, say a square of size $L \times L$. We will allow the agents to move in their environment. Static agents seem simpler at first, however they are not as we shall see below. Moreover, at least the mosquitoes have to be allowed to move. If all agents were static then no non-trivial disease transmission dynamics could emerge.

Abstracting away all but the most essential details, we end up with a model that has a fixed number of ageless and immortal agents that move about in their environment. We summarize here the essentials of our model:

- There are two types of agent: humans and mosquitoes.
- Agents live in a continuous rectangular environment of size $L \times L$.
- Agents can be in either of two states: infected or susceptible.

- If a mosquito is infected then all humans within a radius b of the mosquito will become infected.
- If a mosquito is susceptible, and there is an infected human within a radius b of the mosquito, then the mosquito will become infected.
- Once infected, humans remain infected for R_h time steps, mosquitoes will remain infected for R_m time steps. The actual numbers chosen will be arbitrary, but should reflect the fact that the life span of a mosquito is very short compared to the likely time a human remains infected.
- All agents move in their world by making a step no larger than s_h (humans) or s_m (mosquitoes) in a random direction; essentially they take a random position within a radius of size s_h or s_m from their present position. This rule introduces randomness into the model; the randomness which will turn out to be irreducible under some conditions. Henceforth we will always keep $s_m = 1$; this is a rather arbitrary choice but it simplifies the investigation of the parameter space by reducing its dimensionality. At the same time, we expect that it will not make a material difference to the behavior of the model as we observe it.

These choices are of course not unique and have a certain degree of arbitrariness to them. Following the same principles of simplicity, one could have come up with a slightly different model. But, for the rest of this section, we accept this choice.

Note that this description includes provision for a number of *parameters* or configuration variables, such as L , b , s_h , etc. Identifying appropriate parameter values is always an important and unavoidable part of the development of an ABM. The problem with parameters is that they are often not known, but their precise value could (and typically does) make a huge difference to the behavior of the model. Finding the correct parameters is, at least within biological modeling, one of the greatest challenges of any modeling project. In practice one often finds that a model has more parameters than one anticipated. (We like to refer to this phenomenon as the *law of mushrooming parameters*.) One strategy to reduce the number of parameters is to leave out interactions, phenomena, and/or details from the model that (i) require additional parameters and (ii) are perhaps not essential to the particular purposes of the investigation. Keeping the mushrooming under control is not something that can be done by following the rules given in a book, but requires the sweat and tears of the modeler who has to go through every aspect of their model and ask, “Do I really need to include this effect right now?” In general it is a good rule to commit model details to the executioner, unless their necessity is proved beyond a reasonable doubt. No mercy should be shown.

The reader will perhaps agree with us that our model of the spread of malaria is a bare-bones representation of the real system. Not much can be stripped away from it without it becoming completely trivial, yet even this simple model has many parameters that need to be determined. Before we start considering these in more detail, we are going to make a seeming digression and explore the way in which the mathematics that we have abandoned in favor of computer modeling still has something left to offer us.

2.4.1 A Digression

One of the problems of computer simulation models in general, and ABMs in particular, is their lack of transparency. Even when a model is relatively simple, it is often hard to understand exactly how one would expect the model to behave. This is particularly true during the early stages of the modeling cycle. Even more so, one can never be quite sure that the model is actually behaving the way it is meant to. One source of problems is simply programming bugs. Subtle errors in the code of the program are often hard to detect, but could make a big difference to the simulation results. In the case of the Game of Life, testing the model is not such a problem. Once one has tried out a few configurations with known behavior, one can be reasonably confident about the correctness of the code. The source of this confidence is that there is a known reference behavior. The user knows how the model should behave and can compare the expected behavior with the actually observed one. Unfortunately, in most realistic modeling ventures, this will not be true, at least not across the whole parameter space. However, it is often possible to find parts of the parameter space where the dynamics of the model simplifies sufficiently to make it amenable to mathematical modeling. The mathematical model will then provide the reference behavior against which the correctness of the ABM can be checked.

Apart from the correctness aspect, there is another advantage in making a mathematical model: A mathematical formulation will often provide some fundamental insights into how changes of parameter values impact the overall behavior of the model. These insights might not carry over precisely to more general situations in the parameter space, but they often still provide basic insights that might help shape the intuition of the modeler.

Let us consider a few key parameters of our Malaria model—those affecting the ranges of movement of both the humans and the mosquitoes. The simplest case to consider is that humans move by taking a random position in their world; this corresponds to $s_h = L$. This particular choice of parameter is perhaps not very realistic with respect to the real system, but it does have the significant advantage of making the model amenable to a mathematical analysis. When all humans choose a random position in their world then the spatial configurations of the agents become unimportant (we keep $s_m = 1$). This limiting case then corresponds to what we call a *perfectly mixed* model. In what follows, we will walk through the process of comparing the model with a mathematical prediction of its behavior.

In the Malaria model with perfect mixing, a mathematical model can be easily formulated (see [11] for details). If M is the total number of mosquitoes, m the number of infected vectors, H and h the corresponding variables for the numbers of humans, R_m and R_h the times for vectors and humans to recover from an infection, b the area over which a vector can infect a human during a single time step and W the total area of the world; then the proportion of infected vectors and humans can be written as follows:

$$\begin{aligned}\frac{m}{M} &= 1 - (1 - Q(h))^{R_m} \\ \frac{h}{H} &= 1 - (1 - Q(m))^{R_h}\end{aligned}\tag{2.1}$$

Here the function $Q(x)$ is given by:

$$Q(x) = 1 - \exp\left(-\frac{bx}{W}\right)$$

Note that these equations depend only on the density of the agents (the number of agents over the size of the environment) and not on the absolute numbers.

Considering these equations, the astute reader will immediately object that the variables m and h are not dependent on time. This appears to be counter-intuitive. There surely are parameters that allow overall very high infection levels? Given such parameters, if we introduced a single infected agent into a healthy population, it will take some time before high infection levels are reached. It is not clear how the mathematical model could describe the transient infection levels, given that it has no representation of time.

This objection is correct, of course. The model does not describe initial conditions at this level of detail and the model does not contain time. Instead, it describes the long-term or steady state behavior of the model. The steady state of a system, if it exists at all, is the behavior that is reached when the transient dynamics of the model has died out. Often, but not always, the steady state behavior is independent of the initial state. Certainly in the case of the Malaria model, in the long run the behavior will be the same, whether we start with a single agent infected or all of them infected. It is this steady state behavior that the mathematical model (2.1) describes.

There is one complication: What if we start with no infection at all? According to the rules of the ABM, independent of the values of b and W an infection can never establish itself unless there is at least one infected agent in the system. This seems to indicate that, for the special case of $m = 0$ and $h = 0$, initial conditions are important after all. A second glance at (2.1) reveals that the mathematical model can indeed represent this case as well. The reader can easily convince herself that the case $h = 0$ and $m = 0$ always solves the set of equations, independent of the values of b and W . This tells us that, for some parameter sets, the model has two solutions:

- The case where a certain fraction of the mosquitoes and the humans are infected. This solution, if it exists, will always be observed in the model, unless we start with no infected agents at all.
- The other case is possible over the entire space of parameters, and describes the situation where there is no infected agent in the model.

Whether or not there are two or just one solution to the model depends on the parameters. If there are two solutions then the solution corresponding to no infection at all (the “trivial solution”) is, in a technical sense, unstable. In practice this means that the introduction of a single infected agent (human or mosquito) is sufficient for the system to approach the non-trivial solution. In this case, the disease would establish itself once introduced.

In the part of the parameter space that admits only one solution, the long-term behavior will always be the trivial behavior. No matter how the initial state is chosen—whether there are many infected agents, or no infected agents—malaria will never be sustained in the system. For the particular mathematical model in (2.1) it is possible to derive the range of parameters for which there is only one solution. We will not derive the formula here, but be content to state the result. The reader interested in exploring the details of the model is referred to the primary literature [11]. The model in (2.1) has two solutions only if the following condition is fulfilled:

$$\frac{b}{W} > \frac{1}{\sqrt{MHR_m R_h}} \quad (2.2)$$

Essentially, this means that the area b over which the mosquitoes can infect humans within a single time step needs to be sufficiently large in relation to the overall size of the system (W) so that an infection can establish itself. What counts as “sufficiently large” depends on the total numbers of mosquitoes and humans and for how long they remain infected once they are bitten by a mosquito.

2.4.2 Stochastic Systems

There is one additional qualification to the validity of (2.2) that is of more general importance in modeling. Strictly speaking, the condition is only valid in the case of a *deterministic system*. The concept of a deterministic system is perhaps best understood by explaining what is *not* a deterministic system.

Assume in our Malaria model a set of parameters that allows a non-trivial solution. Assume further that we choose our initial conditions such that all human agents are susceptible, and only a single mosquito is initially infected. According to the preceding discussion we would now expect the infection to take hold in the system and, after some time, we should observe the non-trivial steady state infection levels as predicted by the model (2.1).

In the model, this is not necessarily what we will observe. Remember that, in the case of perfect mixing, both mosquitoes and humans take up random positions at every time step. It is not impossible that, by some fluke event, at the beginning of the simulation the single infected agent is in an area where there are no other human agents at all. This probability can be easily calculated from the Poisson distribution:

$$P(\text{no human}) = \exp\left(-\frac{Hb}{W}\right) \quad (2.3)$$

Here Hb/W is the average number of human agents one would expect within the bite area of the mosquito. Remember now that the infection of a single agent lasts for R_m or R_h time steps. In the case we are considering here, the probability that an infection is introduced by a single mosquito is not passed on is then given by P^{R_m} . Depending on the values of the parameters, this probability could be very small, but it is never vanishing. Simple stochastic fluctuations can wipe out an infection, particularly when the number of infected agents is low. Hence, the behavior predicted

by the mathematical models (2.1) and (2.2) might not always be correct, because of *stochastic fluctuations* in the system.

There is another—probably more important—sense in which our Malaria model is non-deterministic. When simulating the model, one will typically find that the actually observed number of infected agents is not exactly equal to the predicted number, but will be a bit higher or lower than the predicted number. The reason for this is that the mathematical model only tells us about the *mean* number of agents in the long run. Stochastic fluctuations ensure that actual infection levels will vary somewhat over time, and either be a bit higher or a bit lower than the mean. The source of these fluctuations is the randomness in the rules of the system, particularly the randomness in contact between humans and mosquitos. A good comparison is a coin flipping experiment. If one flips a coin 10 times then, on average, one would expect to see a “head” 5 times. In an actual experiment this could happen, but one will often observe that the number of heads deviates from this “mean.” In essence, this is the same type of stochastic fluctuation that leads to the deviation from the calculated mean behavior in the Malaria model.

Figure 2.3 shows an example of a simulation of the Malaria model. The particular graph plots the proportion of infected agents at every time step. The simulation shown in the figure starts with no humans being infected. In order to avoid the trivial solution, we start with half of the mosquito population being infected. In the perfect-mixing

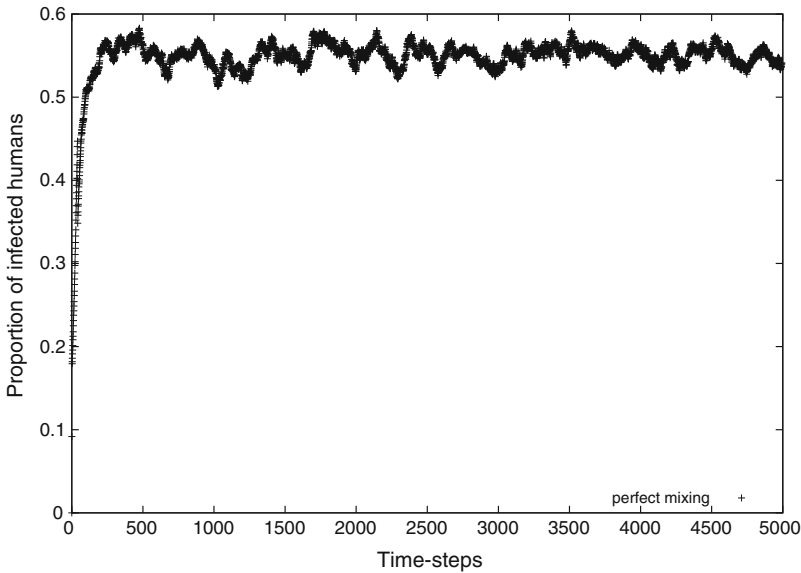


Fig. 2.3 An example of the time-evolution of the proportion of infected humans in the Malaria model when perfect mixing is assumed. We start with half of the mosquitos infected. After a rapid initial increase in the infection levels, the system finds its steady-state and the infection level fluctuates around some long-term mean value

case, the population of infected agents rises initially, until it reaches its expected steady-state behavior. After this transient period, the infection level of the humans stays close to the theoretical steady state value, as calculated from the model (2.1). The level of infection never quite settles on the expected behavior, which is another manifestation of the stochastic fluctuations.

Stochastic fluctuations are often referred to as *noise*. Just how much noise there is in this system depends on the size of the population. In practical modeling applications it is often necessary to reduce the noise. In these situations it will be useful to *scale* the model. By this we mean, a change of the parameters that leaves the fundamental properties of the model unchanged but, in this particular case, increases the number of agents in the system. What counts as a “fundamental property” is problem dependent. In our case here, the fundamental variable we are interested in is infection level, so the fraction of agents that carry the infection.

If we attempted to reduce the noise by simply changing the number of humans in the model, we might reduce the noise, but we would also change the expected levels of infection by changing the density of humans in their “world” and the number of humans in relation to mosquitoes; this would not be scaling. In order for the properties of the model to remain the same, we would also need to change the size of the world and the number of mosquitoes, such that the scaled up model has the same density of mosquitoes and agents as the original model.

The expected behavior, as calculated from (2.1), will be the same at all scales, but the relative fluctuations around the expected behavior will be smaller the greater the number of agents; note that the mathematical model itself is formulated in terms of agent densities only. Only in the limit of an infinitely large population will the expected steady state behavior equal the observed one. The effect of scaling on noise is illustrated in Fig. 2.4. Particularly for smaller systems, with few agents, the mean or expected behaviors of a system can be very poor predictors of the actual behavior.

In general, stochastic fluctuations are mathematically very difficult to describe and model, unless the system is extremely simple. At the same time, they are important to consider in many contexts. ABMs can be very powerful tools to explore the effects of stochastic effects in biological systems. In addition, there are other approaches that will be discussed in both Chaps. 6 and 7.

Let us summarize before we continue. If we set our parameters such that at each time step the agents take random points in their environment, then we can predict the long term (or steady state) infection levels of the system using a relatively simple mathematical model. However, the accuracy is limited by the level of noise in the model that, in turn, depends on the number of agents. Noise can be systematically reduced by scaling the model up. The steady state behavior predicted by the mathematical model will be correct only in the case of an infinite number of agents in the model. Also, the mathematical model tells us nothing about the behavior of the model when there is no perfect mixing. For those cases, we will have to rely on simulation.

As we reduce the scope for agent motion in the model, the spatial structure of the population becomes more important and the perfect-mixing approximation, as expressed by the mathematical model (2.1), becomes more inaccurate.

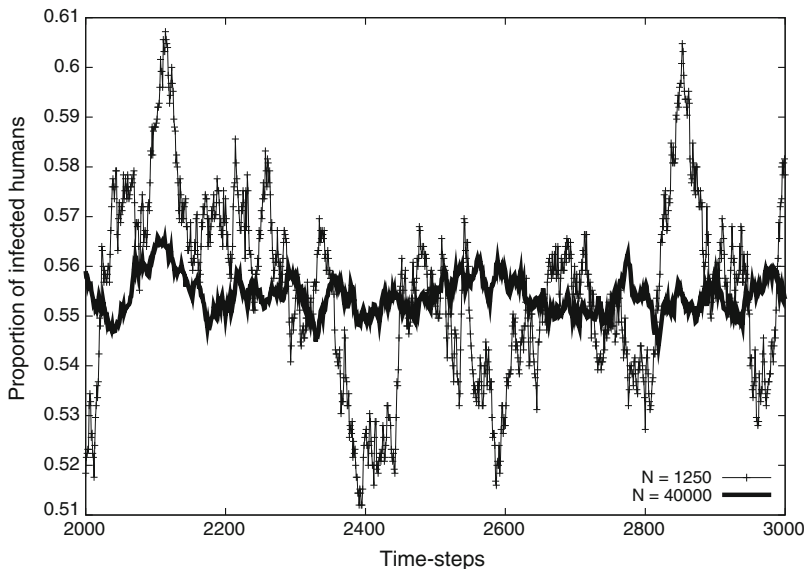


Fig. 2.4 Noise in agent-based simulations depends on the population size. These are simulation runs for perfectly-mixed populations using the same parameters as in Fig. 2.3, with two different population levels. The stochastic fluctuations are clearly larger for the smaller population size

2.4.3 Immobile Agents

Let us now consider the opposite extreme of perfect mixing, namely that humans do not move at all⁴ ($s_h = 0$). Humans are initially dropped randomly into the world, and remain in their place throughout the simulation. Remember that humans cannot infect each other directly, but any transfer of the disease must be mediated by a mosquito.

Mosquitoes lose their infectivity after a relatively short time—2 time steps, say. Since we set their maximum step size $s_m = 1$, it will not be possible to transmit an infection over distances greater than 2. This means, in effect, that an infection can only be carried from one agent to another if these agents are within a distance of 2 of each other.

To help in the following discussion, let us define *clusters* of human-agents that allow cross-infection: Two agents are within the same cluster if the distance between them is 2 or smaller. Depending on the size of the system and the number of humans in it, there could be only a single cluster encompassing all agents, or there could be as many clusters as there are agents. In the latter cases we would have a very sparsely populated environment. The idea of the definition of a cluster is that an

⁴We assume that the mosquitoes continue to move; if all agents were immobile, this would be a meaningless model.

agent A can only transmit (either directly or indirectly via other agents) an infection to another agent B if A and B are in the same cluster. Mosquitoes can still move between clusters; yet moving from one cluster to the next will take more than 2 time steps and the mosquito would lose its infection on the way. A corollary of this is that, once a cluster is free from infection, there is no possible mechanism to re-infect the agents in it—the infection has died out in that particular cluster. In this sense, clusters partition the agents into a number of sub-populations that are independent of one another.

For such clusters, there are now a number of interesting questions to be asked (and answered). For example, we may want to know whether particular clusters can actually lose their infection altogether; how does this depend on the size of the cluster and the length of an infection? We will not present formulas to calculate these probabilities, but a qualitative understanding of what influences this can be reached easily using simulations.

To address this, we will approximate each cluster as a system in its own right, with a given number of agents and a size. To understand the dynamics within clusters, it is worth remembering that they may be very small. As discussed above, stochastic fluctuations are relatively much more important in the case of small systems. It is therefore conceivable that, just by some statistical fluke, in a small cluster infection dies out altogether, sooner or later.

Figure 2.5 addresses this by simulation. It shows three simulations of the Malaria model, each with very few agents (and a correspondingly small world size). To make things a bit simpler, we assume perfect mixing in the clusters. In these simulations we are interested in understanding how long it takes for small populations to lose infection levels altogether by chance. To do this we assume a certain infection level as an initial condition and let the model run until there are no more infected agents in it. We then record the time it took for this to happen. From the figure it is clear that the population dies out relatively quickly for the particular parameters chosen.

Clearly, in this world infections are not sustainable. Note, however, that this result of the computer model contradicts the predictions of the mathematical model. For the parameters used in Fig. 2.5 the stability condition (2.2) predicts stability of the non-trivial solution. The mismatch between the mathematics and the simulation is purely due to the stochastic fluctuations around the mean, that are not accounted for in the mathematical model. Even though the simulations in Fig. 2.5 assume perfect mixing, they still do not behave as the mathematical formula predicts, because there are so few agents in the system and the stochastic effects become large.

2.6 Confirm that the mathematical model predicts stability for the parameters in Fig. 2.5.

Returning now to the full models with immobile humans. The individual clusters do not quite behave like perfectly mixed sub-populations, although they are perhaps not far away. Nevertheless, the clusters of immobile agents are subject to the same random fluctuations that led to a loss of infection in the small models with perfect mixing. Hence, the simulations in Fig. 2.5 explain why the infections in the small

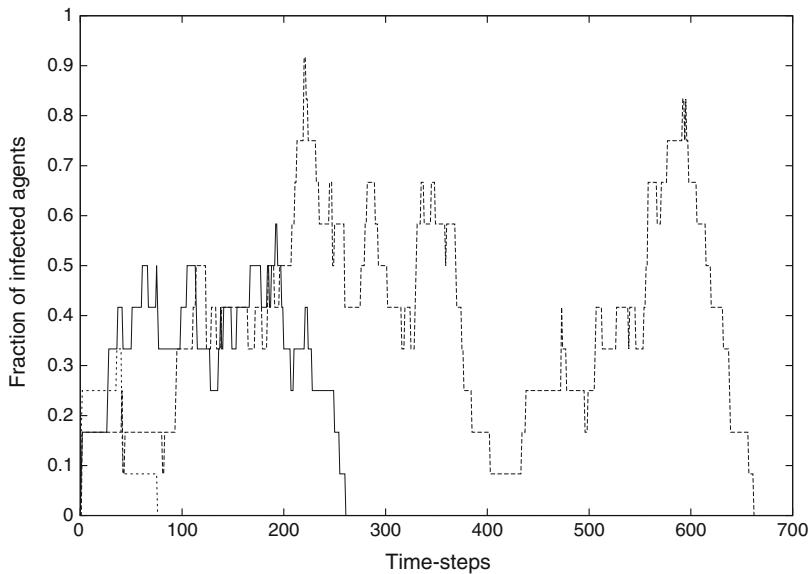


Fig. 2.5 Three runs of a system consisting of 25 agents in a world of size 2×2 . The infection levels fluctuate wildly. The times required for the infection to disappear vary between 100 and nearly 700 time steps. For systems of this small size, the mean steady-state infection levels are no longer a good description of the system

clusters are lost over time. Overall we can predict, therefore, that in the case of immobile human-agents it takes much higher population densities to sustain an infection than predicted by the steady state solution for perfect mixing cases (as formulated by the condition in (2.2)). We still do not know exactly how dense the population must be to sustain the infection, but at least we can make a qualitative statement about the behavior of the system: Infection levels in the case of immobile agents are lower than in the case of perfect mixing.

What now for intermediate values of s_h ? After all, we would expect that realistic cases are somewhere in between perfect mixing and immobile humans; we would also expect that long-term infection levels are somewhere in between the two extreme cases, but we really need to simulate the system to find out. Figure 2.6 shows a comparison of the Malaria model with two different levels of agent mobility. The parameters of these example simulations are the same as in Fig. 2.4; the difference between the simulations is only the level of movement of the humans, s_h . The perfect-mixing case settles around a steady state of between 0.5 and 0.6. For $s_h = 1$, the infection level in the population is markedly lower and only about 1/3 of the population is infected. In simulation runs where s_h is set to zero, the infection dies out quickly (not shown).

This dependence of the model result on the mobility of agents is a recurrent theme in ABMs. Often it is even the main purpose of a particular ABM to explore the dependence of system behavior on the range of allowed motion of agents. It is certainly one of the strengths of ABMs to be able to address this kind of question.

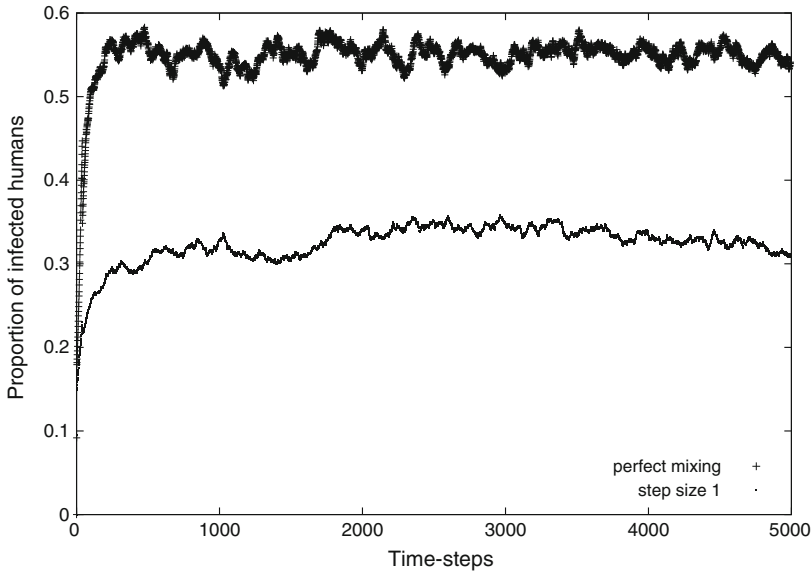


Fig. 2.6 The perfect mixing case is compared with a step size of 1. If agents do not move at all, the infection will die out. Moderate levels of movement lead to relatively low infection levels compared to perfect mixing

2.5 General Considerations When Analyzing a Model

While this particular, simplified model of malaria is unlikely to be useful in any real-world scenario, it is a very useful model for understanding how various assumptions about the nature of agents impact on the results one can expect. The Game of Life and Malaria models highlighted a number of features that suggest a modeling approach based on ABMs, which we summarize as follows.

- **Simplicity and complexity:** Systems consisting of very simple components can often display complex behavior at the aggregate level. The Game of Life was an example of such a system. Agent-based models are useful in modeling how interaction between many types of agent leads to interesting behavior.
- **Stochasticity:** The Game of Life is an example of a system where agents are strictly deterministic. This is rather unusual because the behavior of agents in most models will have stochastic components. When this is the case, the aggregate behavior will often (but not always) show stochastic fluctuations. In general, the fewer agents there are in the system, the larger the relative fluctuations. For some systems, these fluctuations can be so large that they become an important feature of the system in which case the randomness of the system is irreducible. The Malaria model with static human agents is a case in point. More generally, in many biological systems noise is increasingly becoming accepted as an important effect. As discussed

above, sometimes stochastic behaviors can be described deterministically, often they are irreducible.

- **Interactions:** The transmission of diseases over geographical areas is a result of many interactions of agents, as are the patterns produced in the Game of Life. When these interactions become irreducible to a mean-field approximation then ABMs are the method of choice. A case in point is the Game of Life where local interactions are essential for the large scale emergent patterns.
- **Heterogeneity:** Systems consisting of agents that assume different internal states over time are often not reducible to their average dynamics, and a mathematical representation of the system becomes hard or impossible. In the Malaria model, an example of heterogeneity is the infection state of the agents and their position and movement properties. More generally, two of the most important sources of heterogeneity in systems are the spatial distribution patterns of agents, and the internal states which distinguish one agent from another one. In the case of the Malaria model, perfect mixing renders the heterogeneity reducible, which makes mathematical approaches feasible. If one relaxes the assumption of perfect mixing then the heterogeneity of agents becomes irreducible and an analytic treatment very difficult.
- **Steady state:** ABMs often, but not always, approach a steady state behavior after a certain time. In a sense, this is good from a modeling perspective because it means that the initial conditions of the model are not important. Yet, natural systems do not always reach a steady state and sometimes the transient state is crucial to the understanding of the system. Mathematically, transient behavior is much more difficult to model, whereas this is not difficult in ABMs.

2.5.1 Testing ABMs

Sooner or later, every modeler using ABMs faces the difficulty of establishing whether or not her models are correct. By “correct” we do not mean that the model reproduces reality sufficiently well. That is a concern too, indeed philosophically a very deep one. Philosophically less difficult, yet practically of at least the same importance, is a different sense of correctness: Does the model actually show the behavior the modeler intended?

Larger ABMs can be quite complex pieces of software. Even the most experienced programmers will find that their programs contain programming errors. Some of these bugs lead to quite obviously wrong behavior but the majority, by far, do not and can be very hard to detect. In principle, one can never be sure that one’s model behaves as intended. There are some strategies, however, to reduce the probability of falling prey to such bugs.

As a general rule, the modeler should always keep a healthy scepticism about the correctness of her software. Both experience and logic tell us that exhaustive testing of even very simple software is impossible, hence it is never possible to prove the absence of bugs [12]. Therefore, the basic assumption must be that the model is incorrect rather than correct. While such a standpoint might appear very depressing at first, there are a few strategies that can be used to detect errors in a model:

1. Mathematical modeling;
2. Extreme parameter tests;
3. Hypothesis-test cycles;
4. Reproducibility.

Mathematical modeling is the best method to create confidence in agent-based models. Unfortunately, the cases where agent-based models can be treated mathematically over the entire parameter space are rare. Of course, if there were an efficient way to use mathematics to describe the system then there would be no need for the ABM in the first place! Mathematical models provide much more general and powerful results than computer models. When it is possible to approximate the behavior of the ABM mathematically for some parameters (as in the case of the Malaria model) then this can provide the reference behavior against which to compare the computer model. This will not always be possible and, even if it is, one should never take this partial consensus between model and mathematics as a confirmation of the correctness of the model. The bug may well only strike in the area of parameter space that is not accessible to equations.

Another method to test the model is to perform extreme parameter tests. The behavior of models of interacting agents is often hard to predict in general, but sometimes easier to understand when some of the parameters are extreme. For example, in a model of agents that take up food, decreasing the supply of food to zero should lead to the entire population dying within a time that is defined by the life-time of the agent. Running the simulation with this extreme parameter will immediately test whether or not the part of the program relating to the death of agents works as intended. This is just an example. Which extreme parameter tests are useful and which ones are not very much depends on the model. In practice, these test are crude but they are a time-efficient way to catch some basic mistakes early on in the development cycle.

A more refined way to find errors in simulation models is to use hypothesis/test cycles. This is a variation on the extreme parameter methods. It is usually very difficult to understand how a given model behaves for a specific set of parameters, but one can normally predict how a *change* of parameters affects the behavior of the model qualitatively. A simple example would be that the population size tends to increase with the food supply, or that an increase in the length of a malaria incubation leads to a higher proportion of sick agents, and so on. During testing of a model, the modeler should form as many hypotheses of this sort as possible and systematically test them. Sometimes the prediction will not be confirmed by the model. If that is the case then it is necessary to understand why exactly the behavior differs from the expectation. The iterative process of forming hypotheses and trying to predict the effect of this change does not only sharpen the intuition of the modeler but, most importantly, can uncover internal inconsistencies in the model. So it is an important part in the process of the modeler gaining confidence about the model and eradicating errors.

Finally, the results of a particular simulation run should be reproducible. Because ABMs are usually stochastic, two simulations will not normally reproduce exactly

the same data (if they do then this usually indicates a problem with the random number generator). However, when repeating many runs, the qualitative behavior should be the same in most of them.

In the case of evolutionary simulations (as they will be discussed in the next section), it might be that a specific behavior only evolves with a certain probability, rather than with certainty. In those cases, the model should still be stochastically reproducible in the sense that this probability can be reliably assessed by repeated experimentation.

2.6 Case Study: The Evolution of Fimbriation

The popular account of biological evolution is phrased in terms of individual selection, prominently featuring the “survival of the fittest.” The basic idea of the individual selection narrative is that individual organisms are not equal in their ability to collect food and produce offspring. Those that are slightly more efficient will tend to have more offspring. If the more efficient organisms have some traits that the less efficient do not have then, over time, these traits will spread. To illustrate this, imagine a population of animals of species *X*. Suppose that, over a long period of time, those members of *X* that had bigger ears had, on average, more offspring. Over evolutionary time periods one would then likely observe a gradual increase in the mean size of ears among all members of *X*.

Of course, this very brief account of evolution is only a cartoon of the reality of evolutionary theory, and it would go far beyond the scope of this book to explore this topic in any depth. However, the simplified account of the evolution of the hypothetical species *X* reflects the basic idea of evolution by natural selection. Phenotypical changes are driven by heritable genotypical variations. Typically, one thinks of evolution as acting at the level of the individual (which in the example above would be the owner of the pair of ears).

2.6.1 Group Selection

It is not necessary to dig very deep into biological examples before one finds that evolutionary change is not always driven by variations between individuals, but could also be thought of as acting on entire groups of individuals. To illustrate this, imagine a group of birds sitting on a tree doing whatever birds do. Assume now that a predator approaches. One of the birds of the flock spots the danger and produces warning noises. This alerts the entire group, enabling its members to take-off and flee to safety. In this example, the action of an individual warning-bird has saved multiple members of the flock from potential death through predation. At the same time, however, the warning-bird itself has perhaps directed the attention of the predator to it, thus increasing its chances of ending its days as a dinner. As long as one assumes

that evolutionary change is driven strictly by differences between individuals (e.g., size of ears), it is difficult to explain how the “warning bird gene” ever evolved.⁵

The warning-bird example seems to suggest a picture of evolution that acts on the entire group; a narrative that is different from individual-selection scenarios. A higher fitness of the group as a whole need not mean that each of its members has a higher fitness, too. In fact, as the bird-example shows, some of its members could be worse off, even if the fitness of the group as a whole increases.

Whether or not evolution does indeed act at a level of the group, as well as on differences between individual organisms, is still the subject of a heated debate between evolutionary theorists. Much work has been done on “group-selection” theory (for example, [13–17]) in an attempt to understand how it could evolve. The topic also receives significant attention from evolutionary biologists (see, for example, [18, 19]). Debates and models in group selection are often centered on the idea of cooperation in social groups. In this context, “social group” does not necessarily mean a group of higher animals that have a group structure of some complexity; instead, it simply means that there exists a population with some form of interaction between its members.

As an example, one might think of *E. coli* bacteria. Under starvation conditions, some strains of these produce *siderophores*. These are a class of chemicals that make iron soluble, which enables the bacteria to scavenge the metal from the environment in which they live—normally the gut of a host. Siderophores are simply excreted into the host environment by the bacteria. The concerted action of the group facilitates the release of significant amounts of iron to the benefit of all group members. The simultaneous release of siderophores by all bacteria can be seen as a kind of cooperation. Each cell participates in constructing a small amount of the chemical, but only the collective effort leads to a reasonable and useful amount of dissolved iron.

So far so good. The problem starts when one considers that a cooperation of this sort comes at a cost. In order to synthesize the siderophores, the bacteria have to use energy. They do this, of course, in the “expectation” that their “investment” will reap an adequate return. The energy and resources the bacteria exert on producing siderophores could alternatively have been invested in growth and production of offspring. However, if an individual cell stops producing the siderophores it will still be able to benefit from those released into the shared environment by other cells and continue capturing iron. At the same time, unburdened by the cost of producing siderophores it would grow faster and produce more offspring. Evolutionarily, it would be better off.

The situation here is similar to that of a shared coffee-making facility as is common in some university departments. Assume that the purchase of new coffee beans is funded through the contributions of coffee-drinkers. Each time somebody takes a cup of coffee she is expected to contribute a small amount of money to the communal

⁵We do not really assume that a single warning bird gene exists, and this expression should therefore be understood as a label for a number of genetic modifications that impact on the said behavior.

coffee fund. Normally, such an arrangement is unenforced, in the sense that it is fully possible to drink the coffee without paying for it. A coffee scheme run in this way relies on the honesty of the coffee drinkers for its continued functioning. If a large number of drinkers suddenly failed to pay their expected fees, there would be no money left to purchase new coffee once supplies run out. In this case, anybody who wanted a coffee would have to go to the campus café, where a coffee costs considerably more (furthermore, for busy academics the time investment would be even more punishing). In this sense, the communal coffee scheme relies on the coffee drinkers cooperating; all drinkers are better off as long as everybody contributes their share.

The crucial point is that, as long as most members are honest, an individual is better off not paying. Using such a “cheating” strategy means the get to drink coffee for zero cost.

In the realm of bacteria and siderophores, the situation is similar but worse. Unlike dishonest coffee-drinkers, bacteria that fail to contribute their share of siderophores will tend to reproduce faster than their honest conspecifics. Assuming that the dishonesty in bacteria is an hereditary trait, non-cooperators will tend to increase over time, ultimately resulting in a breakdown of the cooperation between the cells. That is the naive expectation, anyway.

We will not go any deeper into the details of siderophores and communal coffee facilities. Suffice to say that the problems of cooperation in those realms of life seem to be relevant for many other aspects of social life on earth. Cooperation between individuals (be it coffee-drinkers, bacteria, or birds in a flock) seems to exist in real systems. Classical “survival of the fittest” accounts struggle to explain how such cooperation can be sustained unless there is some type of selection at the level of the group. Group-level selection would have to function somehow so as to restrain the individual organisms from following their selfish path to defection (which would ultimately leave everybody worse off).

The problem with group selection is that it lacks a convincing mechanism that could explain it. While the core narrative of the survival of the fittest individual is intuitive and convincing, there is no corresponding story telling us how evolutionary forces could work at higher levels. Groups seem to lack the features of classical Darwinian individuals, in the sense that they are often not clearly defined; they do not reproduce/leave offspring in any defined sense, and it is not immediately clear in what sense they have fixed traits that can be reliably inherited by their offspring and lead to fitness differentials between groups. For this reason, group-selection theories have been eyed with suspicion by many evolutionary theorists.

Computer models are ideal tools for studying this. One can use ABMs to implement evolutionary processes and test explicitly under which conditions group-selection works. The advantage of computer models over observation and interpretation of real systems is that ABMs do provide a fully specified system. Every underlying assumption is, by necessity, spelled out in the code of the implementation. In the remainder of this chapter we will discuss in detail a case study that uses ABMs in this context. This will illustrate how these computer models can be used to model evolution. However, as the primary aim of the following discussion is to demonstrate

in detail how a modeling idea is translated into a design for an ABM, the case study should also be of interest for readers who do not specialize in evolutionary modeling.

2.6.2 A Model of Martian Mice

2.6.2.1 Group Selection Without Dilemmas

Group-selection is often seen as an issue only when there is the potential for cheating—when there is a social dilemma. A group of individuals and organisms contribute to and take from a common pool. This tempts some into defection, or cheating, meaning that they draw from the pool without contribution—very much like the coffee drinkers who do not pay. Most of the research work on group selection focuses on these kinds of scenarios.

Lesser known are scenarios where there is no social dilemma of this kind, but there still seems to be selection at the level of the group and it is unclear how this group-level selection could evolve and under which conditions. This is best illustrated using an example. Imagine a population of entities, say Martian mice. These live in caves and exist in two main forms: those having a green tail and those with a red tail. They are otherwise identical. Throughout the life of a mouse, the color of its tail may change from red to green, or vice versa. These color-changing events are essentially random, but the rate of change is somewhat influenced by environmental conditions. Since there is not much vegetation on Mars, the mice depend on a rather peculiar mechanism to acquire food: at the back of each cave, Martian cheese mushrooms grow. These mushrooms are an important source of food for the mice, although they do not entirely rely on this food source. What makes these cheese mushrooms and their interaction with the mice scientifically interesting is the fact that the growth rate of the mushroom depends on the color of the tail of the mice (by a mechanism that, to this day, puzzles Martian—and indeed all other—scientists). Up to a certain point the growth rate of the mushrooms increases with the number of mice with red tails in the cave. However, once there are more than 20 mice with red tails, the mushrooms turn toxic (because of an excessively high growth rate) and kill any mice that ingest them.

Interestingly, the probability of a color change depends on the amount of mushrooms the mice eat. The more mushrooms they eat, the more likely it is that red tailed mice change into green tailed mice. The switch from green to red, however, has been found to be independent of the mushroom consumption (or any other environmental condition). Experimental work by Martian biologists has led to a rather detailed understanding of the mechanisms that lead to the color change. As it turns out, the switching rate from one color to another is genetically controlled, whereas the individual color changing events are purely stochastic. This means that the tail color of an individual mouse cannot be predicted with certainty based on knowledge of the environmental conditions alone. It is, however, possible to predict the probability of a particular mouse being red- or green-tailed given the amount of mushrooms

available in a particular cave. Observations have also led to the discovery that there are usually between 10 and 15 red-tailed mice in a cave.

Martian mice and their tail color are a typical example of a group selection effect without a social dilemma. Maintaining either a red or a green tail is energetically equivalent, that is having either color is not an advantage or burden for the mice in an evolutionary sense. Both red-tailed and green-tailed ones will have the same amount of offspring, on average, if they have the same amount of food. For an individual mouse, having a red tail or a green tail is energetically equivalent. On the other hand, at the level of the group the number of red-tailed mice is very important; it determines the growth rate of the mushrooms in a cave. Since the mushrooms are shared equally between all mice in a cave, the growth rate in turn determines the size of the mouse population in the cave. This tells us that the genes that determine the rate of the color changing events are significant for the group, but do not provide differential fitness advantages to the individuals.

There is, quite apparently, an optimum state in which a cave has exactly 20 red-tailed mice. This would be the best state for the population as a whole, because at this point mushroom growth is maximal, but mushrooms have not yet turned toxic. It is, however, not entirely clear how the system could evolve to that state. We have to assume that, initially (or at least at some time in the past), the gene networks that implement the change of tail color were not optimally adapted. Either there were too many red-tailed mice around, leading to toxic mushrooms, or there were too few, and the population did not grow as fast as it could have. The question is now: By what mechanisms can evolution steer the group as a whole to the optimum?

The standard individual-selection process that relies on the fitness differential between individuals in a population would not work here. To see this, consider the following: Assume the parameters of the genetic networks regulating the color switching rate are such that the population is not getting the best mushroom yield because the switching rate from a green tail to a red tail is too low. In this case, the mice have to wait for a mutation to improve their lot. Note that such a mutation will happen at the level of an individual mouse, so will change the genetic make-up of one individual, not the entire group. What is needed is a mutation that either reduces the switching rate from red to green or increases the opposite. As long as the mice are patient enough, such a mutation will eventually happen. In fact, it is not that unlikely. Any mutation affecting the switching probability can lead to either an increase or a decrease of the probability of the affected mouse being red-tailed. Hence, the chances of a beneficial mutation are quite good.

Let us assume now that, at some point, such a beneficial mutation has occurred, leaving one of the mice more likely to be red-tailed. At the level of the group, this will have the effect that on (time-)average there will be more red-tailed mice than before the mutation took place. Naturally, the effect will only be very small because the mutation affected a single mouse only. This is not a problem *per se*. The question is, rather, whether there is a mechanism that allows this small improvement to be amplified: Will the mutation spread?

This question is not straightforward to answer. For one, in finite populations simple stochastic fluctuations can prevent the spread of even the most beneficial mutation,

even in classical individual-based systems. The effect is very similar to the stochastic extinctions discussed in the case of the Malaria model above. The existence and importance of such effects have been well studied in theoretical evolutionary biology. Stochastic fluctuations of this sort are certainly a relevant effect in the case of the Martian mice because the number of mice in a cave is very low—small population numbers tend to re-enforce the importance of stochastic fluctuations. However, in the present case this is not the main problem and we will ignore this complication for the moment.

More related to the question of group selection is another complication; assuming that the new mutant mouse has only a slightly higher probability of being red- than green-tailed then, depending on the population size, the effect on the population could be miniscule. The increase of the (time-)average number of red-tailed mice might be tiny and, given the stochastic nature of the switching between the tail colors, the effect might completely disappear in the normal background noise of stochastic switching. A small mutation in one cave could, therefore, result in no significant change in the growth rate of the mushrooms. Admittedly, whether or not this is true depends on the population size and the specifics of the relation between the growth rate of the mushrooms and the number of red-tailed mice.

The larger the mutational change in a particular mouse, the larger the potential impact on the population as a whole. A mutation in a mouse could result in a dramatically increased probability of being red-tailed—say it is always red-tailed. In this case, the effect on the (time-)average number of red-tailed mice in a cave may be sufficiently strong to make a material difference to mushroom growth.

Let us now consider what happens if, by some mechanism that we still do not understand, this initially beneficial mutation spreads in the population; that is, all the mice become red-tailed. By the rules of the mushroom, this would be disastrous for the mouse population. If all mice were red-tailed all the time, then all mushrooms would turn poisonous, spelling the end of the mice in this cave. They would all die. These *gedankenexperiments* highlight an inherent tension in the evolutionary dynamics of the red-green switch in Martian mice. Small mutations will not have any effect and too large mutations will lead to the extinction of the colony, at least if they spread.

So far we have only considered mutations that are (at least in the short term) beneficial in the sense that they lead to an increase of the mushroom yield in a cave. What we have ignored are all those mutations that have the opposite effect—those leading to a short term decrease of the mushroom yield. Since mutations are random events, “good” and “bad” mutations happen at roughly the same frequency and will therefore tend to cancel each other out. The net effect will be zero. On average, the population moves nowhere. Note, however, that this does not mean that there cannot be significant biases in particular populations. As discussed above, stochastic fluctuations can take systems very far away from the expected mean behavior.

Note that in the case of a classical individual-based Darwinian evolution the balance between beneficial and detrimental mutations would be no problem. The bearers of the beneficial mutations will tend to have more offspring, whereas the sufferers of detrimental mutations will leave few, if any, offspring. In the case of the

Martian mice, classical Darwinian arguments cannot be invoked to explain the spread of “good” mutations because the mushrooms are shared equally among all members of the cave. There is no fitness-feedback to bearers of individual mutations, always only to the group as a whole. If, by some stroke of luck, there are two beneficial mutations then everybody in the group will equally enjoy the increased mushroom yield, and the population as a whole will expand. Even those who do not have the beneficial mutation will have more offspring (on average). Similarly, there might be times when, by some statistical flukes, there are mutations leading to fewer red-tailed mice on (time)-average. Again, the population will contract—both the bearers of the detrimental mutations and all others. Over evolutionary time, the average number of mice will perform a random walk. One unfortunate consequence of this is that, sooner or later, the number of red-tailed mice in a cave will go beyond the crucial threshold of 20, the mushrooms will go sour, and the population become extinct. This means that, as far as the evolution of tail-color-switching in Martian mice is concerned, we need to explain two things: (i) how can an optimal switching frequency evolve and (ii) how can random extinctions be prevented?

As it turns out, there is a model that allows the spread of beneficial mutations. A *sine qua non* for this is that individual caves are not isolated. In other words, there needs to be more than one cave and it must be possible for mice to migrate between the caves. There is no mechanism by which beneficial adaptations can spread within a single cave (except chance). In fact, in every particular cave the mouse population will eventually die out, as we have just seen. However, if there is occasional migration between caves, then empty caves can be re-colonized by migrants and new colonies can be established, and a total population crash can be avoided.

To simplify the detailed explanation of the evolutionary mechanisms, let us introduce some shorthand notation. When we say that a mouse is *fitter* than another, then we mean that its switching rates from red to green and vice versa are such that, if an entire population had these switching probabilities then this population would be closer to the *achievable optimum*. Rather than referring to the individual, the notion of fitness here implicitly always refers to a group. By “achievable optimum” we mean the following: The more red-tailed mice there are (up to a certain point), the more mushrooms grow and the larger the population of mice will grow, since the growth rate of mushrooms depends on the *absolute* number of red-tailed mice. Food is shared between the mice, and more red-tailed mice means a higher population growth of all mice. Higher growth means that the population increases, which means that there will be more red-tailed mice, which further increases growth until there are too many red-tailed mice and the mushrooms become toxic. The optimal point, or the point of highest fitness, is just before mushrooms turn toxic.

We can now explain how migration between caves can serve as a mechanism for adaptation of the switching between red and green tails in a population of Martian mice. As before, we assume that the system starts in an unadapted state, i.e., an average number of red-tailed mice that is too low—we do not need to consider the case of too high a number of red-tailed mice, because such populations would immediately go extinct. Let us further assume that there are many caves with mice. Each sub-population is genetically diverse at first, in the sense that the switching

rates between red and green within a cave may be very different. Starting from such an initial state, after some time has passed sub-populations will start to become extinct—then statistical flukes will lead to this. The key point is that it is extremely unlikely that all caves become extinct simultaneously, to the extent that we can ignore the possibility. As long as some caves are populated then a total collapse of the population can be avoided by occasional migration of mice from one cave to another. Empty caves will be discovered by migrating mice who then establish new populations. If we assume that migration is relatively rare, then we would expect a strong *founder-effect*. As a result, the newly established cave-populations will be genetically rather homogeneous. All offspring of the founder will have the same, or at least similar, color switching probabilities as the original ancestor. Some variation enters through occasional mutations that adjust the probability of some mice to be red-tailed. Another source of variation is influx from relocating mice, i.e., mice that come from other caves. While there will always be some diversity within the caves, we can assume for the moment that this diversity is relatively low.

Once the population in each cave has become extinct at least once, we would expect a high heterogeneity between different caves, but relative genetic homogeneity within each cave. The differences between the sub-populations will manifest themselves in different population sizes in the caves and different expected times before the inevitable extinction event happens. This difference is a key element for the adaptation mechanism. The longer it takes before a population becomes extinct, and the larger a population in a specific cave, the more mice will, on average, leave it by migration. This translates into an increased rate of colonization of empty caves. Hence, bigger and longer-lasting sub-populations are more likely to be the source for new founder mice in new caves. Given the homogeneity within each cave, we can also assume that a newly established population of a cave, will normally be very similar to the parent population (although this will not always be true).

This provides a direct mechanism for competition between caves. A cave is fitter if it is less likely to become extinct and if it has a larger population. Re-colonization is the group-level equivalent of reproduction. Group-level mutations are achieved by a constant influx of outside mice and mutations within a cave. Any newly established cave will be similar to its “parent-cave,” but not necessarily equal.

Altogether, this suggests a possible mechanism for selection at the group level. Unfortunately, verbal reasoning is limited in its deductive powers. While this scenario for how group selection could work seems plausible, one needs more precise reasoning to be sure, and to be able to determine under which conditions it can work.

As a modeling problem, this scenario has all the ingredients that make it suitable for an ABM and hard for many other techniques (specifically mathematical models). There is irreducible randomness in the system. What we are interested in in evolutionary systems is to see how the action of rare events (positive mutations) can shape the fate of a population. A statistical description of the system is in this context not satisfactory. With the help of a random number generator, ABMs can generate instances of random events and replay evolution. Also crucial to the model is the heterogeneity of the system. All the mice are different, or at least potentially different to one another in the sense that they differ in their switching rates. This difference cannot

be reduced to a description of the “mean” switching rate, or some other macroscopic variables, whence the system is irreducible heterogeneous. Moreover, crucial to the system is the interaction between the agents (mice) and their environment (cave). Again, this interaction is irreducible. We conclude that an ABM is the ideal method to model this system.

Before we commit to the expense of setting up a model and simulating it, we should convince ourselves that the problem is indeed worthy of our attention. Martian mice are perhaps too remote to earthly concerns to justify the investment in time and effort that a model requires. However, there is a system here on earth that behaves in a similar way.

2.6.2.2 Fimbriation in *E. coli*

Fimbriae are hair-like structures that grow on the surface of some bacteria, such as *E. coli*. They are so-called *adhesins*, which essentially means that they are used to attach to host cells. Fimbriae are what biologists call a *virulence factor*. This means that they directly cause an immune reaction and make the host ill. *E. coli* comes in many genetic variants and not all of them are virulent. In fact, some *commensal* (that is non-disease causing) strains of *E. coli* permanently colonize human gastro-intestinal tracts. One thing that makes them interesting from a health perspective is that even these commensal strains still express some of the virulence factors, such as fimbriae, but only do so at a low level. In the current context, “low level” means that, at any one time, only a small proportion of the population actually expresses them. Whether or not a specific cell expresses fimbriae is a random decision that is genetically coded—by the so-called *fim* operon. Within the *fim* operon is an invertible element called *fimS*. Two proteins, FimB and FimE can (by themselves) catalyse an inversion of *fimS*. They do so by binding on each side of the invertible element, then literally cutting out the element and re-inserting it in the opposite orientation. The expression level of fimbriae depends on the orientation of *fimS*: If it is inserted in the “on-orientation” then it is expressed; if it is in the “off-orientation” it is not expressed (Fig. 2.7).

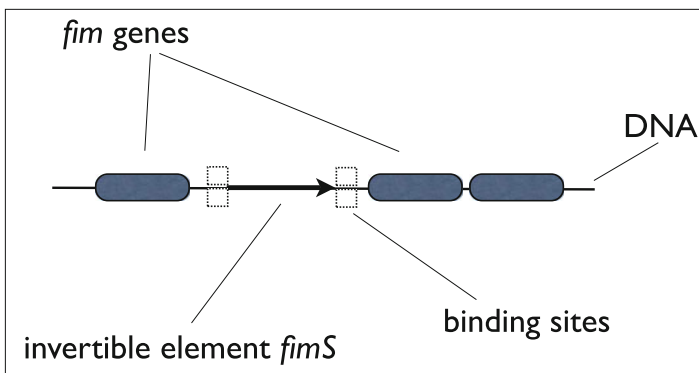


Fig. 2.7 The organization of the *fim* operon and the *fim* switch

There is a subtle difference in the way FimB and FimE function. FimB switches at about the same rate in both directions. Its overall switching rate, however, is very low. FimE, on the other hand only switches efficiently from on to off and it does so with a much higher efficiency than FimB. Yet FimE is not expressed when the *fim* switch is in the “off” position. In practice, this architecture results in a division of tasks between the two proteins: FimB essentially only switches fimbriae on. Once FimB is expressed it does not play much of a role compared to the much more efficient FimE that switches fimbriation off.

There are a number of factors controlling FimB expression, including temperature and growth rate. The dominant determinant for the FimB concentration is the concentration of sialic acid in the host cell. Sialic acid is a potential nutrient for *E. coli* and is released by host cells in response to low levels of fimbriate bacteria. Up to a certain point, an increase in the number of the bacterial fimbriation levels leads to a further increase in the amount of sialic acid released by the cell. The situation is analogous to the case of the Martian mice, where an increase of the number of red-tailed animals in a cave leads to a higher growth rate of the mushrooms. *E. coli* faces the same fate as the Martian mice once the fimbriation levels become too high. Instead of stimulating further sialic acid release, fimbriation levels above a certain threshold trigger an immune response by the host cells. Such an immune response normally leads to extinction of the bacterial colony, and makes the host sick. In this sense, sialic acid is for earthly *E. coli* what cheese mushrooms are for Martian mice, and fimbriae correspond to red tails.

The Martian population is partitioned into only weakly interacting sub-populations by the caves. In *E. coli* this role is played by different hosts. Ingestion of these bacteria is unavoidable. Hence, even if an immune response leads to the complete elimination of previous populations, the host will be re-colonized by new bacteria. This re-colonization would be the equivalent of migrating Martian mice. Crucially, also equivalent to the Martian mice is that, within each sub-population, we can assume that all bacteria share the benefits of increased release of nutrients. Sialic acid is released by the host into the environment of the bacteria, all of which are free to take it up.

2.6.2.3 Designing a Model of Fimbriation

The question remains: How can the switching rates between red and green tails, and fimbriate and afimbriate states, evolve so as to optimize the benefit for all, while avoiding toxic mushrooms and deadly immune reactions by the host? Above, we presented some hand-waving arguments to suggest a mechanism. Let us now design a model in order to subject our hypothesis to rigorous and merciless test by a computer. ABMs are precisely the tools that allow us to implement the conditions that we have described above. Pure verbal reasoning glosses over many details of the problem. A computer model, on the other hand, forces us to proceed with rigour and to specify

and spell out every single assumption we are making. In what follows, we will discuss in detail how the ideas we have presented on group selection can be converted into an ABM. In particular, this discussion will place emphasis on highlighting how design choices are made. These choices are not definitive, however, in the sense that one could come up with a different model that does the same thing. Nevertheless, the principles are general ones. The overriding aim is to obtain a model that is as simple as possible, while still showing the desired behavior. As always in modeling, before choosing to implement a particular feature we will carefully deliberate whether or not it really is necessary for the particular purposes of the model.

At the beginning of a design process for ABMs we need to ask ourselves three questions:

1. What are the agents?
2. What is the agent's environment?
3. What are the interaction rules: between agents and agents, and between agents and the environment?

2.6.2.4 Agents

Starting with the agents, the entity in the system that is most subject to change is the bacteria, and it is changes in them that are the focus of interest within the model. An important agent type is therefore the *E. coli* bacterium. Since we are (at least for the moment) only interested in a single species of parasite, a single type of agent will suffice in our evolutionary model.

There are a number of minimal requirements that this agent has to fulfill in order to be useful. Firstly, since we are interested in evolutionary change, there should be the ability for agents to die and reproduce. A standard design choice in ABMs is to couple reproduction to the accumulation of nutrients. One way to implement this is to convert nutrients upon uptake into “energy points.” Over time, agents accumulate these energy points—the more food they gather, the faster they accumulate energy. In many models one finds that individual agents are required to pay a “maintenance-tax,” that is, they lose some part of their accumulated energy at each time step simply in order to stay alive; this simulates the costs of maintaining cell processes. Real bacteria certainly do have ongoing maintenance costs and it would make sense to include such a tax in our model. On the other hand, this introduces an additional parameter to the model. We would have to decide how much an agent pays for maintenance. The additional complication is not very great, but there is no obvious benefit from having it either, at least not in the present case. We are interested in group selection here, and an individual maintenance tax does not seem to be very relevant to our model. It seems best, therefore, to simply set the maintenance tax to zero in the present case.

Contrary to popular belief, real bacteria do not live indefinitely. Cell division is asymmetric and one part of the dividing cell inherits the “age” of the parent cell, whereas the other part can be considered the “newly born” cell. We could directly reflect this feature in the model. Yet, again, in order to simplify the problem (without making much of an error) we will assign an age to every agent instead of keeping

track of every division event. The age is updated by 1 at every time step. Whenever an agent has reached a certain threshold age it will be eliminated from the model with a given probability per time step. In a simplistic way, this reflects the finite life times of cells. Both the threshold age and the probability to be eliminated need to be set as parameters. One might think that it is simplest to set the probability of death to 1 once the crucial age is reached, as this avoids setting an extra parameter. In practice, this has the undesired side-effect that subsets of the population then tend to act in “lock-step”; for instance, for periods of simulation time no agents die, but at certain intervals a large proportion of the population dies. This effect leads to artificial swings in the population size and complicates the analysis of the model. Making death a probabilistic event (depending on age) de-synchronizes the life-cycles of the agents, albeit at the cost of a new parameter. This cost is not too high, however, because the particular parameter value chosen will not materially influence the outcome of the models and can be kept fixed over all simulations.

Similar arguments apply to agent-reproduction: An agent should only produce offspring if it is successful enough, that is if it has collected sufficient nutrient. Just how much nutrient is “enough” is to some degree an arbitrary decision, as long as we do not insist on quantitative accuracy of the model (and this would be impossible to achieve anyway). So, again, the modeler is free to choose a value that is reasonable. Agents should not be able to collect all nutrients necessary to reproduce in one step because this would mask subtle differences in performance between agents. If finding food once were sufficient, then there are essentially only two categories of agents, those that reproduce and those that do not. Since reproduction is intimately linked with fitness, this would result in a very crude “evaluation” of fitness dominated by noise. On the other hand, if agents have to collect nutrients over many update steps then more computing cycles are spent on evaluating the fitness of agents, which increases the computational cost of the model. A good parameter value for the reproduction threshold strikes a balance between these two opposing demands. Independent of the chosen value of the parameter, reproduction should also be stochastic (like death) in order to avoid simultaneous reproduction of all agents. That is, once the agent has collected enough energy, it will reproduce with a certain probability per time step.

A crucial element of the model is the design of the evolutionary search space the bacteria can explore. This is the feature on which both the credibility and the feasibility of the model depend. One could try and design the artificial cells such that they have a vast space of possible behaviors to explore. This would, perhaps, lead to credible models, but certainly also to infeasible ones. The bigger the search space for evolution the harder it is to program the model and, most of all, the more likely the model contains bugs that are hard to find. Models that are too complex are bad from a practical point of view.

When designing a model, it is paramount to keep focused on the particular research question that motivates the model. Our particular task here is to test a hypothesis about group selection. This hypothesis might ultimately be motivated by an interest in fimbriation or red-tailed mice. Yet not all details of these systems are necessarily relevant for the particular evolutionary scenario we are interested in. Clearly, Martian

mice and *E. coli* are very different. Yet if we have understood the evolution of the switch from red to green tails, we have also understood how the control of fimbriation evolves. When modeling the evolution of either system, we can therefore ignore the features that are not shared by them. Considering the ultimate goal and motivation of a model is usually helpful to make key-simplifications. In modeling, stripping away detail often results in better models rather than worse ones.

The genetic network that controls the *fim* switch is reasonably well understood and could be explicitly implemented in biochemical detail. However, in practice this would not work very well. Firstly, biochemical interactions are slow to simulate. This means that we would spend much of the simulation time calculating the interactions of molecules, which is not the focus of the model. Secondly, the time scale of biochemical interactions is very much shorter than the time scale over which evolutionary change happens. Hence, taking into account biochemical interactions would transform our model into a multi-scale simulation problem, which requires significantly higher technical skills than single-scale models. Thirdly, and most importantly, there is no need to implement detailed biochemical models. The detailed genetic processes in Martian mice and *E. coli* are certainly different, which should tell us that these differences are irrelevant for our particular research question. In the context of our model, we are interested in the *strategy* cells use. The specifics of the *fim* switch are only of parochial, earth-centered importance, and should be left out. What we are interested in is the rate with which bacteria switch from fimbriate to afimbriate, which constitutes their evolutionary strategy. Any such strategy could be implemented in a number of ways biochemically, and any conclusions reached about the strategy are likely to be of inter-planetary importance, rather than limited only to a particular earth species.

Instead of worrying about the underlying molecular mechanisms that implement the *fim* switch in reality, we simply model it by two numbers, representing the rate of on-to-off and off-to-on switching. In the particular case of *fim*, the on-to-off can be thought of as essentially fixed, albeit with an unknown value. The switching rate is a core issue in the evolution of the system and we leave it therefore to adaptive forces to determine this value; the additional benefit is that this relieves us of the burden of choosing a value for it. The off-to-on switch is somewhat more complicated because there is a coupling between the amount of sialic acid that bacteria find in their environment, and their probability of switching from off-to-on.

In order to find a plausible representation of the relationship between nutrient/food and the tail/fimbriae switching rates, we choose to be inspired by the *fim* genetic networks of *E. coli*. FimB needs to bind to four binding sites to effect the switch (see Fig. 2.7 on p. 57). Furthermore, there could be cooperativity between the binding sites, that is the binding is stronger when there are two molecules binding than when only a single site is occupied (see also Sect. 6.3.3 on p. 235). Mathematically, cooperativity is normally modelled using a so-called *Hill function*, which is a step-like function with a parameter h . (For an illustration of the Hill function see Fig. 2.10;

also Sect. 4.5.1 on p. 159.) The higher h , the more the Hill function resembles a true step function. Another parameter of the Hill function, K , determines at which point the function is halfway between the minimum and the maximum value. For our purposes we can write a Hill function as follows:

$$h(x) = C \frac{x^h}{x^h + K^h} \quad (2.4)$$

The variable of the Hill function would here represent a measure of the availability of nutrient, for example the number of mushrooms in the cave or the concentration of sialic acid. The constant parameter, C , determines the maximum and minimum values of the system. The function $h(x)$ reaches its maximum value for an infinitely high x . This is easy to see. When x is very large then K^h is small compared to x^h and the term containing the fraction will tend towards 1, giving $h(\infty) = C$. On the other hand, for a vanishing x the function approaches zero, i.e., $h(0) = 0$. Given that we are interested in probabilities per time step, and that Hill functions also (crudely) model how molecules bind to the DNA, $h(x)$ seems a good *ansatz* for the switching function we are seeking. It should be stressed here that the main feature that makes Hill functions a good choice is their mathematical flexibility; the fact that they can also be interpreted as a description of how FimB and FimE bind to the DNA is, and should be, secondary.

In bacteria we observe that a higher sialic acid concentration reduces the rate of switching. If we take $h(x)$ as this rate and x as the amount of sialic acid, then we would like $h(x)$ to reach its minimum value for high values of x . One way to achieve this is to use $1 - h(x)$ in the switching function rather than $h(x)$ itself. This works, as long as we constrain C to be between 0 and 1. This leaves us now with the switching function S^+ :

$$S^+(x) = 1 - C \frac{x^h}{x^h + K^h} \quad (2.5)$$

Having determined the functional form of this equation, the next question is, which values to assign to the parameters? The answer is very simple: We do not worry about them, but let evolution do the job. The expression for the switching probability in this equation is a general expression that determines the broad shape of how the probability of switching from off to on depends on the sialic acid concentration x ; by changing the parameters (i.e., K , h , C) the function can take a variety of different responses. In this model, we are precisely interested in finding out how evolution shapes these parameters to find a suitable response function that enables stable and well adapted populations of *E. coli* (or Martian mice). The only thing we have to do, as a modeler, is to set an initial value for these parameters, to give evolution some starting point. The presumption is that this initial value does not matter, and the simplest thing is therefore to assign random numbers to them. After that we will leave it to evolutionary change to find more suitable values.

2.6.2.5 Environment

A potential second category of agents is the hosts (i.e., the equivalent of the caves on Mars). It might be a good choice to introduce hosts as a type of agent if we plan on expanding the model. At the moment it is simpler not to have host agents. We assume that there are several hosts, but the hosts themselves are all the same. We also assume them to be static. Instead of being agents, hosts are modelled as compartments in the environment. These compartments each contain a sub-population of agents. There is limited migration between them. To simplify the model further we assume that the agents spend only a negligible amount of time in migration between two hosts compared with the typical residence time within a host. This means that it does not take any time to move from one host to another. In reality this is, of course, not completely correct; yet the error we make by this assumption is very small and the benefit of a simpler model easily outweighs the costs. Most likely, including the time of migration between hosts does not result in a better model at all because: (i) it is unknown how long it takes to migrate from one host to the next; and (ii) we are not primarily interested in numerically-accurate models of Martian mice or the *fim* switch, but in testing an evolutionary hypothesis. This hypothesis can be tested even if we do not know many of the particulars of the system. The reader is again reminded of the mantra: Leaving unnecessary elements out results in better models, not in worse ones.

There is one feature of hosts that does need to be included in the model, namely the release of nutrients. Since we chose to represent hosts as a part of the agent's environment, we need to design our compartments such that they release nutrients in response to the state of the agents within them.

In summary, we choose the environment to be partitioned into a number of compartments, each of which represents a single host. Each compartment contains a number of agents (bacteria) and releases nutrients. Following what we know about the biology of the hosts, we need to couple the nutrient release by the host to the fimbriation levels of the bacterial cells colonizing this particular host (or the growth rate of the mushrooms to the number of red-tailed mice). The precise functional shape of the host response is not known, but some qualitative features of it are: for low levels of fimbriation, only small amounts of nutrient are released; once a threshold level of fimbriation is reached then a full immune response is triggered and the bacterial colony is killed. Since we do not have any quantitative information about the response function, we simply assume it to be of the form of a Hill function again. Other than in the case of agents, we now need to specify the values of the parameters, because the host is assumed to be static. The choice is to some degree arbitrary; we are not interested in quantitative models, only in checking the feasibility of a hypothesis. Let us tentatively assume a response function such as:

$$R(n_f/N) = F \frac{(n_f/N)^h}{(n_f/N)^h + D^h} \quad (2.6)$$

Here we assume that n_f is the number of fimbriate agents in the host, N is the total number of agents, and F is the maximum amount of released nutrient. Essentially, this response function is the same as the preliminary switching function in (2.4); the functional argument here is now the fraction of fimbriate agents in the population, rather than the sialic acid concentration in the environment. There is no real justification for using this function, other than convenience—by varying the parameters h and D , a number of different shapes can be achieved, which in turn allows testing different scenarios.

The response function (2.6) has three independent parameters: the constant D that specifies the point at which the host releases half of its maximum amount of nutrient; a scaling factor F ; and the Hill parameter h . The first two parameters, F and D , are arbitrary in the sense that they simply scale the model. A higher F can be counterbalanced by increasing the energy required for bacteria to reproduce. Similarly, D scales the possible population size that is achieved during simulation runs. Even though the values of the individual parameters are arbitrary to some extent, the set as a whole determines the achievable population sizes. This must be chosen with care. Too large a population may slow the simulation down to a level where it becomes infeasible. A very small population, on the other hand, poses the danger that the individual sub-populations in the compartments are dominated by stochastic fluctuations to the extent that a meaningful adaptation becomes impossible. It is crucial for the success of the modeling exercise to find good values for these parameters to balance these conflicting requirements.

The argument to the function in (2.6) is the fraction of fimbriate cells in the given sub-population, n_f/N . This looks reasonable at first, but a comparison with the real world reveals that it is unlikely to be correct. If the response only depends on the *fraction* of cells, then this would allow *E. coli* to grow to any population size without triggering a host response. In the real cell, however, what leads to the host response is the direct interaction between the fimbriate cells and the host. What counts is the *absolute number* of contacts a host-cell makes with the bacterial virulence factors, i.e., the fimbriae. The relevant variable is therefore the absolute number of fimbriate cells and not their proportion. In addition to the sialic acid released in response to fimbriation, we assume that there are also some other food sources in the environment, that are not further specified. We denote this additional user-defined parameter by G ; it does nothing to the evolutionary behavior of the model, but it is essential to set $G > 0$ to avoid premature population crashes. The correct response function is therefore:

$$R(n_f) = G + F \frac{n_f^h}{n_f^h + D^h} \quad (2.7)$$

This new host response function also introduces some additional requirements on the parameters. Very small population sizes might have the undesired side-effect that they could never trigger an inflammatory host response. If the population size is $P < D^h$ then, trivially, n_f can never reach the inflammatory threshold D^h . This would make the entire model pointless.

In general, the easiest way to determine the correct parameters in situations where there are no reliable empirical measurements is through trial and error. Doing this is somewhat time-consuming, of course, but has the added advantage that it allows the modeler to obtain a feeling for the run-time requirements of the model under various parameter settings, and choose the parameters accordingly.

2.7 Why do we expect population crashes in the absence of G ?

2.6.2.6 Interactions

The final item to be determined is the interaction rules of the model. It is useful to distinguish between interactions between individual agents and the interaction between an agent and its environment. In the current model there is no direct interaction between agents but they do affect each other indirectly via their interactions with the shared environment. At each time step, each host releases an amount of nutrient into the environment. Agents can take up this nutrient. Each agent will obtain approximately the same amount of cheese mushrooms/sialic acid. In the simulation we model this in a simplified way. At every time step we divide the amount of nutrient released by the number of bacteria in the model to determine how much nutrient each bacterium is allocated. This ensures that the entire amount of nutrient is used up at every time step, and shared equally among all agents. This may not accurately reflect real life, but it reflects a key-part of our hypothesis, namely that resources are shared between agents. If one wishes, one could later refine the model and introduce more sophisticated rules of resource sharing. For the moment, the simplest possible solution provides most insight.

The indirect interaction of the agents is mediated through the host response, which is the reaction of the environment to the number of fimbriate agents in the system. At every time step, we assume that the environment releases a certain amount of nutrient that depends on the number of fimbriate agents, according to the corrected response function in (2.7). An inflammatory host response is triggered if the number of fimbriate agents in the host is greater than or equal to D^h , that is, if $n_f \geq D^h$. In this case, the entire sub-population within this host is killed.

2.6.2.7 The Simulation Algorithm

Having clarified the basic structure of the model, we now need to determine the details of the simulation algorithm to use. Evolution clearly takes place in continuous time, but we are not interested in quantitative predictions; there is no need for the additional complexity of an event-driven structure, and we will therefore use the simpler time-driven algorithm that updates the entire population in discrete time steps. The structure of the model's implementation is outlined in Algorithms 3 and 4. At each time step, every agent in every host is updated (that is we use a simultaneous update algorithm).

Algorithm 3 The main loop of the agent-based model for the evolution of fimbriation.

```

Time = 1
loop
  for All hosts do
    Determine the number,  $n_f$ , of fimbriate agents in host.
    if  $n_f \geq D^h$  then
      Delete the entire population of the host and skip to next host.
    end if
    {Release nutrient according to the response function (2.7).}
     $R \leftarrow G + F \frac{n_f^h}{n_f^h + D^h}$ 
    {Determine the number of cells  $N$  in this host.}
     $N \leftarrow \text{CountAgentsInHost}(\text{thisHost})$ 
     $f \leftarrow R/N$ 
    for All agents in this host do
      {Update the internal energy state.}
       $e \leftarrow e + f$ 
       $\text{age} \leftarrow \text{age} + 1$ 
      if  $\text{age} > \text{thresh}_{\text{age}}$  then
        With probability  $p_1$  place agent in the reaper queue and skip to the next agent.
      end if
      {Reproduction places offspring in birth queue.}
      if  $\text{agent} > \text{thresh}_e$  then
        Reproduce cell with probability  $p_2$ .
      end if
      if agentsFimbriate then
        Switch off fimbriation with probability  $p_{af}$ .
      else
        {This uses (2.5).}
        Switch on fimbriation with probability  $p_f = 1 - C \frac{f^h}{f^h + K^h}$ .
      end if
    end for
    {Movement between hosts.}
    With probability  $p_m$  move a randomly chosen agent to a randomly chosen host.
    Delete agents from the reaper queue.
    Place agents from birth queue into same host as parents.
    Clear reaper queue and birth queue.
  end for
  Time = Time + 1
end loop

```

When designing an update algorithm for an ABM, there are a number of choices regarding the parameters. Intuitively, one would think that, in synchronous models, the order of update would not be important (because it is synchronous). Indeed, it *should not* matter. In synchronous algorithms a potential pitfall arises in connection with the order in which rules are applied. Algorithm 3 specifies that agents first collect energy and then check their age and die if their age is above a certain threshold; reproduction happens only after checking for death. Alternatively, one could check for death only at the end of the update step. This would allow agents one additional

Algorithm 4 Reproduction of an agent

```

Agent to be reproduced is A.
{Set the energy to 0 for parent agent.}
 $e \leftarrow 0$ 
Create new agent  $A'$ .
for All parameters  $p_{af}, C, K, h$  do
    Copy parameter from A to  $A'$ .
end for
{Mutate.}
With probability  $m$  change a randomly chosen parameter of  $A'$  by a small amount.
Place  $A'$  into the birth queue.

```

time step to reproduce. The difference between these two alternatives is not always great, but could sometimes make a material change to small sub-populations. The amount of energy released by the host depends on the number of agents in each host. At the same time, the switching probability of agents depends on the amount of food released by the host. During the update procedure, the number of agents will typically change, through death and birth events, which affects the amount of energy available to each agent. This means that the number of living agents at the next time step is not known before all agents have been updated. For this reason, reproduction and birth events need to be updated in a separate update loop from the assignment of energy and the updating of switching events.

One solution is to assign to agents the amount of energy that has been calculated at the previous time step. This means that the nutrients are consumed and sensed at the beginning of the time step; alternatively this could be done at the end of the time step, when agent numbers have been updated. The difference between these two possibilities is small and it does not matter which one is chosen. The important thing to remember is that the details of the update order need some careful thought in order to keep the model consistent.

A further complication, arising from the separation of agent birth from placement, is that an agent giving birth might move host in between those two steps. Should the offspring be placed in the old host or the new one? An implied assumption is that the implementation makes it possible to relate a new agent to its parent, or to its parent's new or previous host.

2.8 Confirm that the verbal description of the algorithm matches the pseudo-code in algorithm 3.

2.6.2.8 Testing the Model

Let us now analyze the behavior of the model. In the early stages of a modeling project, it is often not clear which variables will be the most revealing of the behavior of the model, or which aspects of the model one should focus on to reach an understanding of how it behaves. Once a model is programmed, the modeler will

need to spend significant time exploring it, in order to understand how it behaves, to figure out which variables are most informative, and to find any surprising features that potentially suggest semantic errors. This phase can feel like time-wasting, because it does not actually contribute to a better understanding of the underlying system. However, this is not so. The detailed exploration of the model sharpens a modeler's intuition and is, in any case, an inevitable part of every modeling project.

In the present case, the total size of the population turned out to be the most important indicator. Incidentally, this is true for many models of evolutionary systems and was not entirely unexpected. One would expect adaptation to lead to a better exploitation of environmental resources, which results in a higher sustainable population number. In the present case, we would expect that evolution optimizes the number of agents that are in the fimbriate state, so as to maximize nutrient release and minimize the probability of being wiped out by a host immune response.

Indeed, Fig. 2.8 (left) shows the size of the total population (in all hosts) over time in an example run of the model. During the first 15000 or so time steps, not much seems to be happening. Then suddenly the population increases to a value between 15000 and 18000. Stochastic fluctuations mean that there are swings in the numbers of agents, but all within a well defined region. At around time step 140000, there seems to be another transition to about half a million agents in the population.

The, apparently discontinuous, jumps of the population are very much what one would expect to see in an evolutionary system. They correspond to the emergence of innovations and are commonly observed in evolutionary systems. In this sense, the results are encouraging. Yet, this is only the starting point; more work is needed to understand what is going on in the model and to confirm that the jumps are indeed due to evolution. The observed time evolution of the model could reflect some bias in the system, some delays in birth events, some stochastic fluke of a population that essentially moves randomly, or it could be simply down to a programming error. Ultimately, in large computer programs one can never be sure that the model does what it is supposed to do.

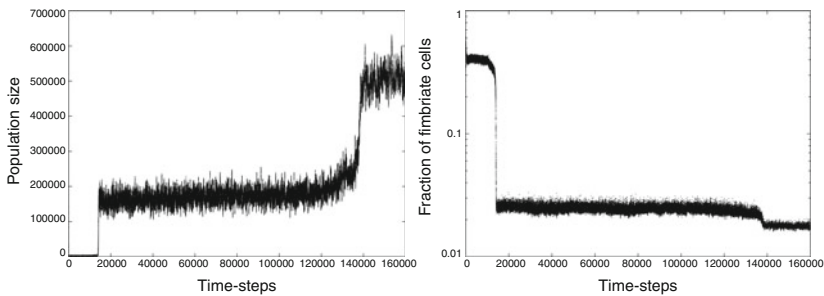


Fig. 2.8 Example of a simulation run of the fimbriation model. The *left graph* shows the population size as a function of time. The *graph on the right* shows the proportion of fimbriate agents (log scale). The following parameters were used: Size of the system 625 hosts, mutation rate 0.1, basic energy per cell 5, reproduction energy 0.2, reproduction probability 0.2, agent life time 40 time steps, death probability 0.2, maximum nutrient release before immune reaction sets in 10

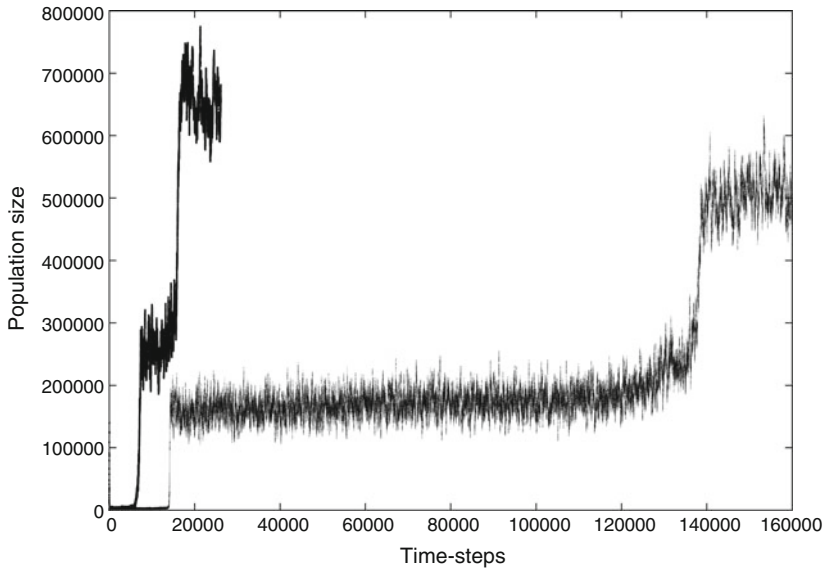


Fig. 2.9 A repeat run of the model shown in Fig. 2.8 for 30000 time steps with exactly the same parameters is shown in *bold*. The transition happens much faster which is entirely a stochastic effect. The other curve is identical to Fig. 2.8 (left) for comparison

Figure 2.9 shows a repetition of the simulation with exactly the same parameters but a different random seed. Qualitatively, the dynamics are the same but the quantitative details of the behavior are very different. Once again, there are two transitions, but these happen much earlier than in the first run. In addition, the population levels reached are higher. These differences are mostly due to the different random sequence resulting from the random number generator of the model, highlighting that a single simulation run only serves to indicate the general properties of the model. It is always necessary to repeat a simulation many times before any conclusions can be drawn from the results as a whole.

So far we have seen that there are more or less discontinuous jumps of the population in the model. The question that we need to address now is, what causes these jumps? The problem one often encounters with ABMs is that even simple systems can be rather difficult to analyze and understand. In the present case, however, we have a clear expectation of what should happen. The crucial parameter that agents need to adjust in order to be successful in their environment is the fimbriation level. Initially, all agents are just in a random state; the average fimbriation probability over the entire population should be around 0.5. The expectation we have is to see some change in that value, synchronized with increases in the population. The right-hand graph in Fig. 2.8 shows the average proportion of fimbriate agents over time corresponding to the simulation in the left-hand side. In order to make it more readable, it is presented using log scale for the fimbriation rate. The main feature of the graph is two clear drops of the fimbriation levels. The first drop is rather steep, whereas the

second is more modest. Comparing this with the graph showing the population size, it becomes clear that the drops in the fimbriation levels coincide with an increase of the population size. Given that the proportion of fimbriate agents is a rough measure for the virulence of the agents, this graph suggests that the agents are able to adjust their virulence in an evolutionary process.

So far the data is encouraging, but by no means conclusive. The observed effect could be due to some other (as yet, unidentified) mechanism, or a process other than evolution. A simple, but not conclusive, way to test whether the observed effects are due to evolution is to repeat the simulations with mutation turned off. In this case, we would expect, at most, a very small growth of the population, and only at very early stages of the simulation. In such a random model without mutations one must expect that the initially heterogeneous population becomes increasingly homogeneous. This effect is simply due to the fact that the genetic diversity of the population becomes impoverished as agents die out. Eventually there will be only a single genotype left in the simulation model. We do not show the graph here, but experiments have confirmed that, indeed, there is no increase in the population size when mutations are turned off. In any evolutionary model this is a key test to generate a baseline against which the creative potential of evolution can be assessed.

A similar test of whether mutation is responsible for the adjustment of population size is to allow mutations, but to start with a completely homogeneous population (or a single agent only). In this case, the diversity is lowest at the start of the simulation, but increases over time driven by mutations. This setup tests the power of mutation to explore the space of possible behaviors. Simulation experiments show that starting with an homogeneous population restores the dynamics observed in Figs. 2.8 and 2.9, although it tends to take a bit longer before the transition from a low to a high population happens. (Again, we do not show the graphs here.)

In Sect. 2.6.2.1 we hypothesized that evolution can only work when the population of agents is partitioned into subpopulations that have limited contact with each other. Using the agent-based model we can now test this hypothesis. The hypothesis is that, if movement between populations is prevented then there should be no transition of the population size from low to high. Indeed, we would expect that the population would die out relatively quickly. Again, we tested this and, not showing the data here, we confirmed this prediction. Partitioning the population into weakly-interacting subpopulations is essential for both evolution and, indeed, survival of the population. If migration between the hosts is prevented then the total population will die out within a relatively short time.

2.6.2.9 Exploring the Behavior of the Model

All this suggests that the model truly shows an evolutionary effect, although one can never be absolutely certain, even in systems as simple as the present model. Our model seems to behave as expected and shows results that we can comfortably explain from our understanding of the system, but this is no foolproof confirmation.

Let us now take the leap of faith and accept that the model does indeed show evolution of fimbriae. Once this point is reached, the next question to be asked is, how does the model's behavior depend on the parameters? In this model of fimbriation

there are so many parameters that it is essentially impossible to reasonably cover the entire parameter space with simulation. Luckily, this is not necessary either.

Many of the parameters of the model are arbitrary in that they simply scale the model. For example, the parameters F and C in (2.7) only determine how much nutrient is released and at which point the limit is reached. These parameters should be set so as to ensure that there is a sensible number of agents in each host (on average) while still maintaining acceptable run-times. Similarly, the lifetime of the agents and the amount of nutrient they require before reproducing are, to a large extent, arbitrary, as long as they do not lead to overly large or small populations. In order to be efficient in exploring the properties of the model these parameters should be kept fixed once practical values have been determined.

A parameter that recurs in most evolutionary models is the mutation rate. In general, the experience with evolutionary systems of this kind is that the precise value of the mutation rate does not matter too much, as long as it is not too small or too large. A reasonable value is best found by experimentation.⁶

While those parameters can be kept fixed once chosen, others need to be explored in more detail. As it turns out, the most crucial parameter of the model is the Hill coefficient in the response function (2.7). Figure 2.10 shows a few examples of a Hill function of the form $f(x) = x^h/(x^h + K^h)$. In the present case we chose the parameter K to be 0.5, just for illustration. From the point of view of our model, the interesting feature is that the inflection point of the function is at K and occurs for the same input value of x independently of h .

For this reason, the inflection point seems a good choice for the point at which the host response is triggered. For one, at the inflection point the slope is maximal; furthermore, this choice of trigger point of the host response also means that the optimum fimbriation level is always at the same value of K in the model and always gives the same amount of nutrient independent of the parameters C and h . The invariance of the best fimbriation levels makes it easy to compare different parameter values directly.

While the optimal fimbriation point does not change for different values of h , what changes considerably is the slope with which this optimal value is approached. The higher the Hill coefficient, the steeper the approach to the inflection point. Biologically this means that, close to the inflection point, a small change of the number of fimbriate agents could have a large effect on the host response. Such a change may well be due to a stochastic fluctuation alone.

Seen from an evolutionary point of view, the problem of *E. coli*/Martian mice is to adjust their switching functions such that they come close to the inflection point where the nutrient release is maximal, yet without crossing it. This task is more difficult than it seems at a first glance. One problem is the stochastic nature of the population. The number of fimbriate cells cannot be directly controlled by the cells, but only the (time-)average number of fimbriate cells. Particularly in small populations, at any given time there will possibly be quite significant deviations from the average.

⁶In the simulations here we used a value of 0.1 per reproduction event.

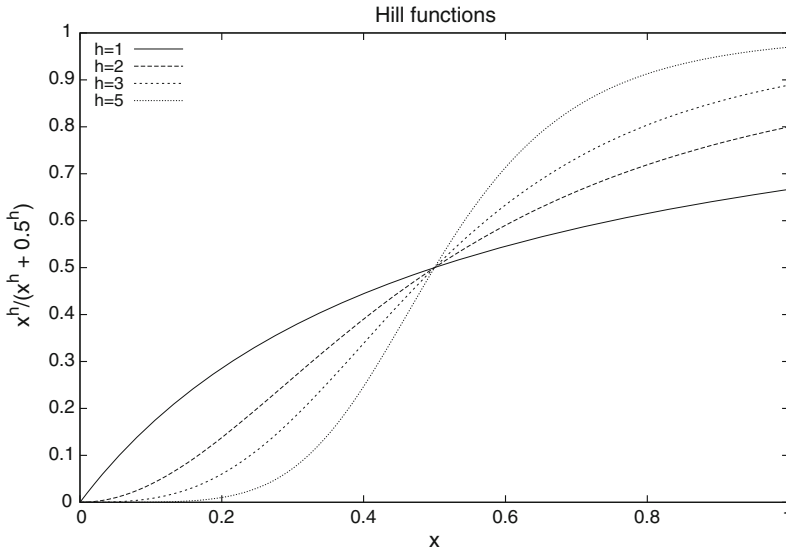


Fig. 2.10 An example illustration of a Hill function for various coefficients h . As h increases the function more and more resembles a step function

At any one time, the actual fimbriation levels can be significantly higher or lower than predicted by the mean. Therefore, the theoretical optimum point, close to the maximum nutrient release, is not feasible in practice. Stochastic fluctuations beyond the mean value would trigger a host response. Therefore, the population needs to keep some distance away from the optimum in order to avoid crossing the inflection point.

Assume now, for the sake of argument, that there is a mechanism for a sub-population to adapt towards a fitness gradient. (We have already indicated what this mechanism could be. For the moment, let us ignore the complexities and simply assume that sub-populations can evolve very much like individual organisms in a classical Darwinian individual selection scenario.) We can now ask: What does this fitness gradient look like? The main factor determining the fitness of a sub-population is the fimbriation rate. Up to a certain point, a higher fimbriation rate means that more nutrient is released, which in turn implies a higher population (we simply equate higher fitness with higher population numbers). Considering this, we see that below the optimum fimbriation point there is a positive fitness gradient for increasing fimbriation probabilities. We can assume that Darwinian actors will move along positive fitness gradients; that is, they will tend to increase their fimbriation levels. The problem starts once the population reaches the point where the host triggers a fully-fledged immune response. At this point, the fitness changes discontinuously from maximal to minimal, i.e., a small change of fimbriation takes us from the point of highest fitness to the point of worst fitness (where the sub-population becomes extinct).

This illustrates that the adaptive problem these populations face is rather difficult. The evolving populations do not “know” where the optimal fitness is, let alone that disaster strikes once they move even one step beyond that. Evolving sub-populations are thus driven up the fitness gradient, just to find themselves becoming extinct once they surpassed the point of optimal fimbriation. Luckily, for the reasons outlined above, sub-populations do not evolve that efficiently. Rather than being driven by adaptive changes within a group, the evolution of the system is mainly determined by migration between groups. This is a relatively inefficient mechanism and, once the population is globally relatively homogeneous, the rate of change will tend to be low.

The analysis of the fitness gradient shows that there is no barrier preventing the population from going “over the cliff” of optimal fimbriation. One would therefore expect that sub-populations would be prone to crossing the virulence threshold and triggering an immune response from time to time. Can we see this in the model?

Figure 2.11 shows the aggregate number of extinction events in the simulation shown in Fig. 2.9. The key aspect of this figure is the sudden decrease of the slope of the graph coinciding with sudden jumps in the population size in Fig. 2.11. This suggests that the population adapts, predominantly by decreasing the fimbriation

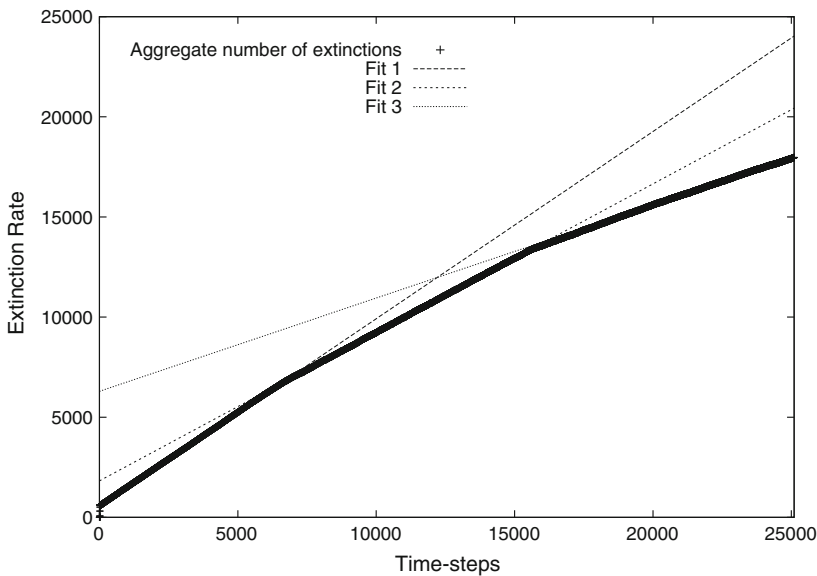


Fig. 2.11 The aggregate number of extinction events in the simulation run shown in Fig. 2.9. The *slope* indicates the rate of extinctions. The *fitted curves* in the figure are merely a guide to the eye. Clearly, the evolutionary transitions in Fig. 2.9 coincide with a reduction of the extinction rates in this model

levels. Yet, Fig. 2.11 indicates that the extinction rate never actually reaches zero.⁷ There are occasional extinction events even in well adapted sub-populations. This means that becoming extinct is a part of life for our simulated agents. Evolution drives the population constantly over the cliff. Seen from the point of view of the host, this means that even commensal strains are virulent sometimes. This is simply an unavoidable result of the shape of the fitness landscape.

Next we want to know how similar individual runs are to one another, that is, what kind of variation we should expect for different random seeds? As the example simulations in Figs. 2.8 and 2.9 demonstrate, two simulation runs with the same parameters can have different outcomes. This, in itself, is an indication that the evolutionary process in this context does not necessarily lead to the optimum outcome. The question we can ask now is, how far away from the optimum will the population typically end up? Furthermore, how large is the variation between different runs of the model compared to the possible range of behaviors?

In order to address this question, we need to understand the possible behaviors of the system. One way to explore this is to take sets of hand-picked values for the parameters that are normally subject to change by evolution, and use them in simulations without evolution (i.e., with a mutation rate of 0). One should choose the parameters such that they cover the possible behaviors well. In the present case, this would mean that one has parameter sets that lead to very high fimbriation levels, as well as very low ones, and everything in between. If we also initialize simulations with a single agent only, then we can be sure that the entire population will be homogeneous with respect to its fimbriation probability.

Figure 2.12 is a graph summarizing the results of such runs. The figure shows the population size as a function of the proportion of agents that are fimbriate. Each individual point in this graph represents the time average of the value pair taken from one simulation run with a homogeneous, non-evolving population. Note that both the proportion of agents and the fimbriation rate are measured quantities that have been obtained from simulations, and are not fixed parameters. The graph suggests that there is a single fimbriation level at which the population becomes maximal. The coverage of the parameter space is not dense enough to determine exactly where this point is; the figure indicates that a fimbriation level of slightly less than 0.1 seems to be ideal, for the parameters used in these simulations. The slope near the maximum appears to be very steep. What is not apparent from this graph, but has been determined by independent examination of the parameters, is that the maximum point seems to coincide with the onset of extinction events in the following sense: For fimbriation levels below the maximum point, there is no extinction whatsoever (data not shown).

The useful feature of Fig. 2.12 is that it shows the possible behavior of the model for homogeneous populations of agents. Evolving, heterogeneous populations will do at most as well (in terms of population size) as the homogeneous ones. Due to

⁷Strictly, it only shows this for this particular run, but we have found this qualitative feature confirmed over all the simulation runs we performed.

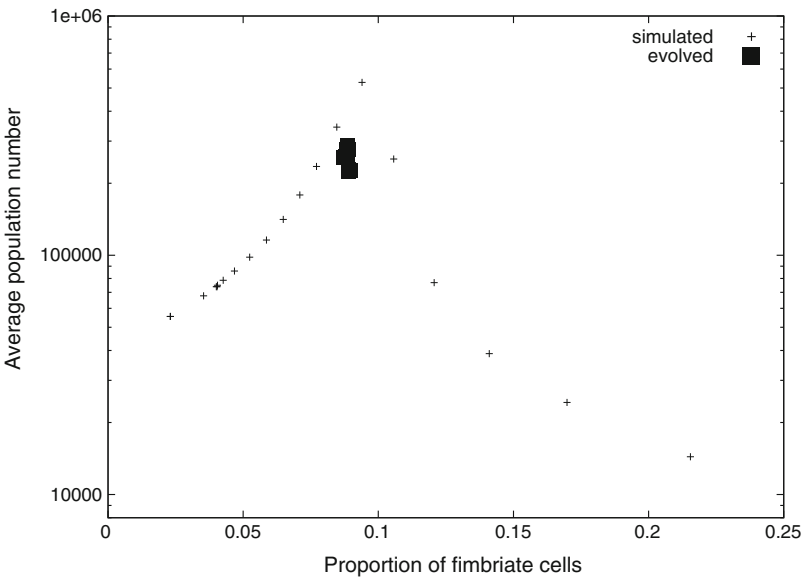


Fig.2.12 Simulations with homogeneous populations and no evolution. This gives an indication of the achievable population sizes. Evolving populations remain well below the achievable population sizes

the peculiar dynamics of the evolving system, one would expect sub-populations to be driven over the cliff of extinction more or less continuously when the mutation rate is greater than zero. This will tend to reduce the population size of evolving populations when compared to non-evolving populations.

Figure 2.12 also shows evolving populations of agents. Two observations can be made: Firstly, compared to the possible behaviors of the model, i.e., the range of possible fimbriation levels and populations sizes, the results of the evolved populations cluster together rather tightly. Secondly, the evolving populations are sub-optimal in two senses: (i) they are below the optimality curve formed by the homogeneous non-evolving populations; and (ii) they are below the optimal fimbriation level.

From this, one is forced to draw the conclusion that evolution leads to sub-optimal outcomes, in the present case. Given the specific “fitness landscape” of the problem, this should come as no surprise. The peculiar fitness gradient that is abruptly truncated by the “cliff of extinction” leads to constant and apparently unavoidable extinction events; hence the sub-optimality (i). Stochastic fluctuations make it necessary for the average population to have a much lower average fimbriation probability. Populations that are too close to the optimum point may be pushed over the cliff; hence the sub-optimality (ii).

2.9 Explain how it is possible that the non-evolving populations show no extinction events?

2.10 Why are homogeneous populations optimal?

An aspect of fimbriation that we have neglected is the issue of regulation. In *E. coli* the probability of being fimbriate depends on the amount of nutrient released by the host. Fimbriation is dynamically regulated in response to released nutrient via the switching function, (2.5). So far, it has not played a major role because we have assumed a static host that always has the same response function, (2.7) (although not the same response). Dynamic regulation of fimbriation, presumably only becomes relevant once we abandon this simplifying assumption and allow variable response functions.

Figure 2.13 shows simulations with hosts that alternate the amount of nutrient they release from high to low every 500 time steps. The figure compares two simulation runs where, in one, the parasites have a static fimbriation function that does not depend on the amount of nutrient and, in the other, the population's fimbriation

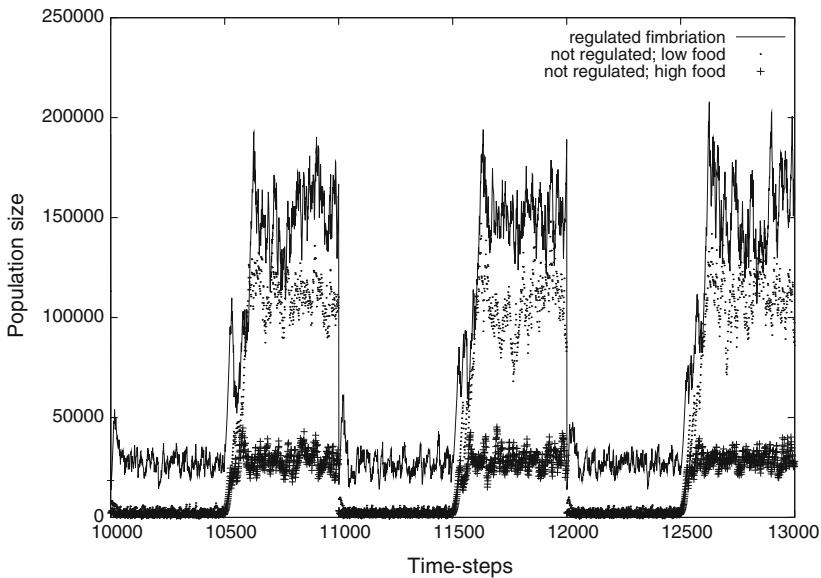


Fig. 2.13 Evolving the population under variable conditions. In these simulations, the host response changed every 500 time steps from low to high to low. The curve labeled “regulated” shows the time-course of a model that has been evolved under these conditions and is allowed a variable fimbriation rate. In contrast, the other two graphs show results from models that had a fixed fimbriation rate independent of the amount of nutrient released by the host. The labels “high” and “low” refer to the amount of nutrient to which the fixed switching rate is a response. This corresponds to the high and low nutrient rates in the “regulated” graph

function does. In practice, the three populations have been obtained from a single run with evolution enabled. We waited until this evolutionary run had reached what appeared to be a stationary behavior. We chose a time point and recorded the average parameter values for the agents that had evolved. We used this to initialize a single agent in a new run with evolution turned off. This is the curve labeled “regulated” in Fig. 2.13. The other two figures, labeled “high” and “low”, were obtained as follows:

- Record the average amount of nutrient sensed by the regulated curve during the phase of high and low nutrient release respectively. From this calculate two switching rates corresponding to the two conditions.
- Start two new simulations with evolution turned off.
- Seed these simulations with a single agent only, with fixed switching rates corresponding to the high and low nutrient release conditions respectively.
- Finally, start a third simulation, also seeded with one agent corresponding to the measured parameters, but with a variable switching rate.

Note that the on-to-off switch is always unaffected by the nutrient contents, and hence is identical in all runs.

Figure 2.13 shows, somewhat surprisingly, that the regulated population is larger than the unregulated curves over the entire cycle. This is quite remarkable, given that the reaction of the adaptable population to the nutrient released at the high point is exactly the same as the one of the fixed response populations; similarly, at the low point of the cycle, one would expect the regulated population to be more or less equivalent to the population that has a fixed response to low levels of nutrients.

2.6.2.10 Summary

Even in the absence of knowledge of the values of parameters, we can still create a good understanding of the evolutionary processes of natural systems—such as Martian mice or fimbriation in *E. coli*. Of course, the computer model presented here is a crude approximation to reality. It does, however, provide some insights into the evolutionary dynamics of this kind of system. Real systems will have many additional layers of complication.

This particular example, worked through from the description of the problem to the analysis of the result, has demonstrated how to distill a computational model from the understanding of a system. We stressed that the guiding principle of this process should be simplicity. We also showed how to choose the agents, the agent-types, the interaction rules and how to represent the environment. Once the model is programmed it needs to be tested. This is a laborious process, but it can be made more efficient when a few rules are followed. We discussed how this can be done.

Even if the model developed in this section is perhaps not sufficiently accurate to reflect the “true” evolutionary history of real bacteria, at the very least it encourages us to ask further questions. An immediate question to ask is: What would happen if

the host response was also subject to evolutionary change, rather than being user-determined? In particular, what types of response curves would be expected to evolve. However, it would go beyond the scope of this book to pursue this question any further.

References

1. Huse, G., Giske, J.: Ecology in Mare Pentium: an individual based spatio-temporal model for fish with adapted behaviour. *Fish. Res.* **37**, 163–178 (1998)
2. Bonabeau, E., Theraulaz, G., Dorigo, M.: Self-organization in social insects. Santa Fe Institute Working Paper 97-04-032 (1997)
3. Casti, J.: *Would-Be Worlds*. Wiley, New York (1997)
4. Ray, T.: *An Approach to the Syntheses of Life*. Oxford Readings in Philosophy, pp. 111–145. Oxford University Press, Oxford (1996)
5. Foundation, F.S.: GSL—GNU scientific library. <https://www.gnu.org/software/gsl/>. Accessed 22 June 2015
6. Kohler, T., Gummerman, G.: *Dynamics of Human and Primate Societies*. Oxford University Press, Oxford (1999)
7. Bak, P.: *How Nature Works*. Oxford University Press, Oxford (1997)
8. Venables, M., Bilge, U.: *Complex Adaptive Modelling at Sainsbury*. Business Processes Resource Centre (1998)
9. Tesfatsion, L.: Agent-based computational economics: growing economies from the bottom up. *Artif. Life* **8**(1), 55–82 (2002)
10. Wolfram, S.: *Cellular Automata and Complexity*. Addison-Wesley, Reading (1994)
11. Chu, D., Rowe, J.: Spread of vector borne diseases in a population with spatial structure. In: *Proceedings of PPSN VIII—Eight International Conference on Parallel Problem Solving from Nature*. Lecture Notes in Computer Science, vol. 3242, pp. 222–232. Springer, Birmingham (2004)
12. Dijkstra, E.W.: *Chapter I: Notes on Structured Programming*. Academic Press Ltd., London (1972)
13. Nowak, M.: *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, Cambridge (2006)
14. Traulsen, A., Nowak, M.: Evolution of cooperation by multilevel selection. *Proc. Natl Acad. Sci. USA* **103**(29), 10952–10955 (2006). doi:[10.1073/pnas.0602530103](https://doi.org/10.1073/pnas.0602530103)
15. Wilson, D.S.: A theory of group selection. *Proc. Natl. Acad. Sci. USA* **72**(1), 143–146 (1975)
16. Wilson, D.: Evolutionary biology: struggling to escape exclusively individual selection. *Q. Rev. Biol.* **76**(2), 199–205 (2001)
17. Sober, E., Wilson, D.: *Unto Others, the Evolution and Psychology of Unselfish Behaviour*. Harvard University Press, Cambridge (1998)
18. Gould, S.: *The Structure of Evolutionary Theory*. Belknap Press, Cambridge (2002)
19. Dawkins, R.: *The Selfish Gene*. University Press, Oxford (1989)

<http://www.springer.com/978-1-4471-6761-7>

Guide to Simulation and Modeling for Biosciences

Barnes, D.J.; Chu, D.

2015, XII, 339 p. 80 illus., Hardcover

ISBN: 978-1-4471-6761-7