Computational models of cells and tissues: Machines, agents and fungal infection

Mike Holcombe

Date received (in revised form): 8th June 2001

Abstract
Computational models have been of interest in biology for many years and have represented a particular approach to trying to understand biological processes and phenomena from a systems point of view. Much of the early work was rather abstract and high level and probably seemed to many to be of more philosophical than practical value. There have, however, been some advances in the development of more realistic models and the current state of computer science research provides us with new opportunities through both the emergence of models that can model seriously complex systems and also the support that modern software can give to the modelling process. This paper describes a few of the early simple models and then goes on to look at some new ideas in the area with a particular application drawn from the world of mycology. Some general principles relating to how new and emerging computational techniques can help to represent and understand extremely complex models conclude the paper.

INTRODUCTION: COMPUTATIONAL MODELS

Computational models are models of systems inspired by the model of an information processing system. In its most common manifestation a model is a digital computer but need not be as restrictive as that in practice. The key issues are:

- systems interacting with their environment (Figure 1);
- information processing models;
- models exhibiting communication and concurrency;
- models that might be amenable to automated analysis as well as simulation.

The most basic discrete model is the finite state machine (Figure 2):  
- a set of internal states;
- a set of external inputs — events;
- a set of system outputs — actions (this is an optional feature — in some machines there are no explicit outputs);

Figure 1: A simple system interacting with its environment

Figure 2: A simple state machine
a transition structure to link it all together.

The inputs are a, b, c and the outputs are: x, y. How does it work?

- if in state 1 and input (event) a occurs then it changes to state 2 and outputs x;
- if in state 1 and input b occurs then it changes to state 3 and output y occurs.

The system then waits for the next input to occur before the next state change. This continues as long as the system continues to function.

These models have been used to analyse metabolic pathway models and we can illustrate this for the simple Krebs (tricarboxylic acid) cycle (Figure 3). If we regard the inputs as being C1, C2 and C3 which are specific coenzymes that drive the cycle and the intermediate substrates as being the states of the system then it behaves like a state machine. Enzymes that are required for each reaction are assumed to be present in sufficient concentration.

This is a simple model of organisation which ignores many factors, such as reaction kinetics and enzyme production. Several quite complex metabolic pathways have been modelled in this way.

The algebraic theory of these machines can be used to construct decompositions into simple components (generated by finite simple groups – a theory that is now well understood in mathematics). What the theory is saying is that any system of this type can be broken up naturally into a collection of subsystems, manufactured from simple mathematical objects, which are joined together as systems in two main ways – parallel and serial connections. For a full description of the decomposition theory and its application to the Krebs cycle see Holcombe, which is an extension of the holonomy decomposition theory of Eilenberg. In the case of the Krebs cycle the algebraic structure of the machine can be decomposed to form an equivalent machine which is built up from cyclic groups of order 2 and 3 connected together as wreath products which are also combined with some simple aperiodic semigroups of order 2 and 3.

What does this decomposition mean, biochemically? It does not seem immediately clear and for this reason research in this area seems to have faded away. However, the impact of a greater understanding of the genomic basis for the production of the enzymes to drive the system might throw new light on the problem.

Many systems can be modelled in this way but:

- it is unsuitable for complex systems because of state space explosion;
- the functions represented by these machines are too simple for many situations;
- continuous behaviour is not modelled;
- concurrent systems are not modelled well without large problems;
- communication is hard to model this way.

Cellular automata, however, are built from these machines and have proved useful in some cases, for example:

![Diagram of the Krebs cycle](image-url)
models of simple development;
models of simple ecologies.

X-machines

A more sophisticated and powerful model can be introduced if we simply add an internal memory and adjust the operation of the machine to match (Figure 4). The system is in some state, an input $a$ is received, the initial contents of the memory are $m$ and, depending on both $a$ and $m$, the system changes state and produces an output $x$ and updates the memory to $m'$. This model is the stream X-machine and is very general. It can model almost all computations and has been much studied in the past five years. The simple device of allowing an arbitrary internal memory has opened up its capabilities in a remarkable way. It is possible, using this, to model very complex systems that would, otherwise, require state spaces with millions of states. The stream X-machine model allows us to abstract these complexities and to wrap them up in hierarchical structures that can be analysed much more easily.

To obtain full Turing computability it is necessary to adapt it slightly. To model systems that operate concurrently and that communicate with each other in a more efficient way we introduce a new approach.

Communicating X-machines

Consider a number of separate stream X-machines (Figure 5) which have the following properties:

- there are certain communication channels between some of the machines;
- some of the states in these machines are solely used for communicating messages to other machines;
- the other states are ordinary processing states.

Each machine works separately and concurrently in an asynchronous manner until it reaches a communication state. It then waits either to send a message to another machine or to receive a message from another machine which may not be ready to send it. The receiving machine must then wait for the message. Once a machine has been involved in a communication event successfully it can proceed with further internal processing until it reaches the next communication state. There are a number of slight variants on this model. All, however, try to model a type of asynchronous communication in a reasonably simple and intuitive way.

The message passing between individual machines is controlled by a matrix which is illustrated in Figure 6.

HYBRID MACHINES

All of the machines discussed previously are discrete, so only instantaneous...
processing can be modelled and only finite discrete data are processed. Continuous functions and real valued data cannot be incorporated into traditional finite state machine models. The hybrid X-machine7 overcomes this.

A hybrid machine (Figure 7) has states and transitions as usual and responds to discrete events and performs discrete actions which are observable. The internal memory consists of a set of discrete variable and a set of continuous variables. When it is in a given state there are sets of equations that apply to the system’s continuous variables and all the time it is in that state, with time progressing, these variables change according to these equations. When either an appropriate external event occurs or a leaving condition is met (eg a set point) the system moves to its next state where a different set of equations takes over (Figure 8).

Some simple examples that can be modelled this way include ion flow through voltage gated channels and antigen–antibody interaction.7 The continuous variables can exhibit complex behaviour. The equations are often composed of relatively simple functions compared with the equations that try to describe the complete functions over all states.

**AGENTS**

Agents are autonomous systems that interact with their environment and other agents, using a set of rules to describe their behaviour. They can be simple reactive systems or they can be very sophisticated with complex strategies and learning capabilities. The important aspect to them is that they are low-level phenomena whose behaviour in their environment emerges according to the rules. In a different environment the behaviour may change. Agents can cooperate or compete with one another.

Many biological processes seem to behave like agents, an interesting example being the system of ants and their behaviour as they form trails between the nest and a food source (Figure 9). Suppose that these ants are simple agents that obey rules relating to the concentration of pheromone and communication events with other ants. Rules in order of priority might be:

1. if detect obstacle then change direction;
2. if detect pheromone then follow trail;
3. if meet ant with food then continue;
4. if find food pick up, turn round and lay trail;
5 if meet ant with no food then turn round;

6 if true then move randomly, etc.

This agent can be described in full detail using an X-machine model, where each function is defined by the rules and the associated inputs and memory. Memory will be the position; whether with or without food; the direction the agent is moving and other relevant parameters. See Figure 10 for a state diagram of such an agent. The details of the function definitions are left out in this review due to space limitations. This will be a hybrid machine. While in the following trail state the motion could be determined by standard equations of motion. A refinement is to introduce a probabilistic dimension to the possible state changes.

Other examples of agents can be found in the immune system, eg T cells, B cells; different molecular species could also be interpreted in this way.

The next issue is how to model communities of agents. Suppose that we had a collection of agents, in this case represented as individual hybrid X-machines. We now have to try to identify the communication channels and how these might work. Suppose that there are \( N \) agents and each is potentially able to communicate with any other. We thus have an \( N \times N \) communication matrix.

Suppose that an agent can only communicate with other agents within a specific distance of it. There is a global memory that maintains the current position of each agent (an \( N \)-vector of coordinates). At any communication state an agent can interrogate this memory to ascertain which other agents are within communication distance. A number of strategies can be used to determine which and how many attempts at a communication can be made. The agent then puts data into the appropriate slots of the communication matrix and continues processing, moving or whatever.

Some of these data items may time out if the agent an item is intended for fails to retrieve it. Similarly, when an agent is within reach of another and wishes to do so it can retrieve data from the specific slot. Agents can die, in which case both the row and column in the matrix become empty (void). Agents can be created, in which case the matrix is expanded to an extra row and column and the memory extended as appropriate.

Some simple ground rules must cover situations where two agents try to occupy the same coordinates. We should be able to simulate communities of simple agents in this way – when \( N \) becomes very large this may be a problem. It should be possible for emergent properties to be identified, analytically or numerically. This needs further investigation.
CASE STUDY OF A HYBRID MACHINE: *Magnaporthe griseum* (rice blast fungus)

Rice is the world’s most important food crop. *Magnaporthe griseum* destroys 40 per cent of crops. The way that this fungus infests rice plants is a major area of research which has made significant advances in understanding the genetic basis of this process. This is a partial model of the infection stage. It is a hybrid machine and uses some of the most recent information about the genetic basis of the behaviour of the fungus.

The spore or *conidium* is a three-celled structure, which is present in the atmosphere in affected areas. Spores alight on the surface of rice plants and attach themselves to the surface, which is possible despite the fact that the leaf surface is highly hydrophobic: the conidium releases from its tip a powerful adhesive stimulated by wetting. The infection process takes place in high humidity conditions during which dew drops form on the leaf surfaces (Figure 11).

The spore germinates and produces a germ tube from one of the terminal cells of the conidium which then hooks itself into the leaf. An *appressorium* of a roughly hemispherical shape then develops at this point of contact.

The penetration of the rice leaf surface is carried out by the build-up of pressure within the appressorium until a penetration peg – part of the appressorium where it adjoins the leaf surface – is forced under this enormous pressure to penetrate the leaf surface. The pressures generated are as high as 8 MPa (or 40 times the pressure in a car tyre).

Once penetration has been achieved the fungus forms tentacles (hyphae) within the host, which then proceed to cause massive damage to the plant through the production of toxic substances and lead to the manufacture of further conidia which become the mechanism for propagation of the fungus to other plants.

The internal variables (or memory) provide information about the status of various important internal aspects of the fungus. For example these will include: the concentration of glycogen and of glycerol; the internal pressure of the appressorium.

Each state either has a leaving condition, some condition that has to be satisfied by some internal parameter in order for a state change to occur, or there is some external event that triggers the state change (Figure 12). For example, the event that occurs when a dormant spore alights on a receptive surface will trigger the initial state transition which then causes initial adherence to the receptor surface. This is triggered by way of some dormancy-release signalling mechanism, thus activating genes needed to initiate the first process of spore tip mucilage release and adhesion.

A state transition involving a leaving condition might be state 9 to state 10 where the leaving condition is the internal appressorium pressure reaching a set point sufficient to cause the puncturing of the host leaf surface. While in state 9 this pressure is undergoing change, mediated by some suitable set of equations determined by the inflow of water through the appressorium surface. If this surface is regarded as a hemisphere and the porosity of the surface is constant throughout, then the rate of increase in mass, the pressure, is given by the linear equation:

\[ p(t) = \rho t/r + \text{initial pressure} \]

where \( p \) is pressure, \( t \) is time, \( r \) is the radius of the appressorium and \( \rho \) is constant.

![Figure 11: *Magnaporthe* infection process (after Talbot)](http://bib.oxfordjournals.org/)

---

Fungus model

Magnaporthe hybrid machine

---

**Figure 11**: *Magnaporthe* infection process (after Talbot)
During state 9, then, the pressure is increasing linearly, assuming that there is sufficient water surrounding the appressorium, until the set point is reached at which time the transition to state 10 occurs, thus triggering further gene activation for the next phase of development.

There are a number of gaps in this model which will be filled as further research into the genetic and molecular basis of the disease is carried out. It is essentially a high-level model which needs to be developed in a hierarchical way so that the individual transitions actually involve complex hybrid submachines and these, themselves, will also break down into lower-level structures until we reach an appropriate representational level.

CONCLUSIONS AND FURTHER WORK

One fundamental problem in modelling such complex systems as biological systems will be trying to understand the complex interactions between many subsystems and the vastly complicated molecular and genetic activity that exists. We might be able to build these models but will we be able to understand and analyse them? It is likely that we will only be able to do this if we simplify them greatly. As an alternative approach Hybrid Projection Temporal Logic (HPTL) has been developed specifically for hybrid...
machines. This logic allows us to define such a machine in a precise formal logic which is the first step towards using automated reasoning techniques. The basic process involves trying to establish properties about the model, now represented as a logical formula in HPTL. There are two, related, ways of doing this. First, we could try to prove theorems about the system by using theorem proving engines, which is probably impracticable since the success of automated theorem provers in dealing with extremely complex systems is limited. An alternative approach is the use of model checking techniques, either alone or in combination with theorem proving. This is potentially feasible and would allow us to ask ‘what if’ questions and query whether the system could ever get into a state with a given property holding, etc. This is more feasible since model checking technology can handle models with very large state spaces. However, the technology needs to be substantially extended to cope with hybrid machines of this type. It does, however, offer a potentially rewarding direction for research.

We could, in the mean time, use simulation, in virtuo, to run these models and derive some useful information about the system from it.

References