Session Notes, Grand Challenges Workshop, 18-19 April 2005, York

Fiona Polack

Caveat

This report is written from notes taken during the Grand Challenges Workshop; it is not a verbatim transcript, and thus may not always be a faithful record of what was actually said.

1 Conference Introduction: Stephen Emmott

Microsoft sponsored the event, not because it wants to build a quantum version of Office next year. Instead, the company is looking to learn something. For example, quantum computing is new to Microsoft and Microsoft Research; there is a very new quantum group in Microsoft in the US.

Microsoft wants to make a scientific contribution, especially in Europe, and to raise the profile of these computational approaches with policy makers, funders and industry. There is an opportunity to create new building blocks for the next century. The new Microsoft Research Science Initiative is lead by Cambridge in Europe, and is starting and seeking long-term partnerships.

2 Session 1: Non-classical substrates

2.1 Invited Speaker: Sam Braunstein, Quantum Computation

There is an imminent and clear challenge in computing to build better, faster computers to counter the classical computing crisis. Moore's Law predicts atomic scale computers in 10 years time; his prediction is 40 years old this month!

Quantum computation would complement classical computer science, not replace it. Information is represented ultimately by physical systems. Moore's Law breaks down when the capacitor is so small that it only accommodates one electron. Technology to allow us to manipulate matter at atomic scale is imminent; can we harness quantum mechanics for information processing? We are not just looking for microscopic computer science; we need to discover how to harness the new technology to produce computation — new paradigms, new hardware, new algorithms; new design approaches and performance metrics; new forms of noise and error correction; new complexity classes from scratch. The biggest challenge is building the machines — many approaches are being explored.

The first serious thoughts on quantum computation are from Richard Feynman in the 1980s; the first breakthrough was not until 1993, with Shor's factoring algorithm. This generated excitement but also scepticism. Over 10 years on, many difficulties have been encountered, but enough has been solved to know that quantum computation will eventually happen.

The power of substituting bits with qubits is illustrated using the physics beam splitter experiment.

The potential of existing algorithms does not only rely on speed-up; the new computation provides new approaches to secret hiding etc. The non-cloning theorem determines that measurement cannot allow all information held to be revealed; it is a general property of quantum information that quantum states cannot be copied, Heisenberg Uncertainty Principle. Generalisations include the absence of a universal NOT gate, since NOT produces a state orthogonal to the input state, and the no-deleting theorem. No universal constructor can be designed or built, no self-replicating automata are possible. These properties are not bugs but features!

Quantum error correction was initially a big block to development of quantum computation. All systems have noise; the conventional approach to noise eradication required copying and measurement. However, a work-around has been developed, using multiple qubits — one qubit of information with error correction requires five qubits of storage. A full fault-tolerant theory is now in place, defining a threshold below which gate infidelity can be corrected, even under random errors. Below the threshold, we could compute indefinitely on arbitrary large numbers of qubits. However, each version of the noise model is system-specific.

Teleportation describes passing quantum info over classical channels such as telephone lines. In general, the no-copying theorem says that this is not possible, because the classical medium could copy the information. This can, however, be worked around with an extra resource, namely shared quantum entangled states. This allows specific principals to be picked out, between which the quantum state can be safely teleported through a classical channel; eavesdroppers are excluded because they can not access the entangled states.

The first successful teleportation experiments were in 1998; a particle was teleported over one metre. Since then, atomic states have been sent over short distances, and light over tens of metres.

Quantum mechanics is easy to illustrate with light. Quantum computation is less easy to implement, and there have been many different attempts [summarised in the talk]. The key problem is to recognise the fundamental tension between the internal tight coupling among qubits required to perform logical, conditional operations, and the weak coupling required between computational qubits and rest of world.

Some implementations are handicapped by scale limitations eg high temperature NMR. Also, semiconductor schemes rely on atomic scale manipulation of matter, the technology for which is not up to speed yet. However, lots of lessons are being learnt about what is necessary and possible; diversity of implementation approach experiments is still important.

2.2 Questions

Andy Tyrrell: *quantum computation presentations all focus on factoring algorithms. Is quantum computation specific to just a few applications?*

Security agencies such as GCHQ and NSA can justify their quantum computation investment on that alone; secrecy and teleportation have provable benefits over classical approaches. However, we do not actually know how hard the factoring problem is; we might be able to make similar improvements in classical sieves, for example.

For Grover's algorithm, the achieved speed-up treats the algorithm as a black box, but if we could open the black box and examine it, we might be able to make further speed-ups; we don't yet know.

Simulation of QC is a second best. We need maturity in understanding to build more sophisticated algorithms. It might be another 10 years before we realise that we can generalise algorithm development approaches.

John A. Clark: *Grover search is fast, but are there other problems expressible in less than 100 bits to which quantum computing could be applied?*

Inevitably, cryptographic problems; brute force search on the square-root of the number of possibilities is a big improvement.

Aaron Sloman: *In the talk, totally secure communication was defined as detecting when an intruder could read a communication; is this adequate?*

The talk did not tell the whole story; omitted issues included quantum privacy amplification, entanglement purification, measurement from the eavesdropper by entangling with one principal. In practice, an eavesdropper on an entanglement introduces more noise; it is possible to take correlations and use the results to amplify the principals' entanglement and minimise anyone else's; this relies on well-understood theory.

Julian Miller: *When will it be possible to teach quantum computing to first-year undergraduates, in the way that classical computing is now taught; would first years be able to write quantum algorithms?*

We are currently teaching quantum computing to third-year computer science students. It is easy to simulate quantum algorithms on classical computers, but the real point is to access the speed-up; for this we need to understand how to harness the power of exponentially-large numbers of simultaneous logical states. At present, there are not many known design tricks; when these exist, it would be possible. The blocks in designing complex algorithms are in turning general calculations into subroutines, taking account of the quantum restrictions that require all subroutines to take the same amount of time (parallelism). We need abstraction paradigms.

Cris Calude: *Randomness was mentioned on the talk slides — Quantum randomness is not understood, and complete randomisation is not yet possible?*

One company selling quantum cryptography (Geneva-based) has built a one-qubit processor that produces random bit streams.

Cris Calude: *They claim that speed of generation is sufficient for security applications, not that quantum randomness is algorithmic or true randomness*

Physicists believe that these quantum bit streams do display true randomness. However, now that it is cheap to acquire the generator, we will be able to actually test the properties of the randomness, for example in Masters projects. It may, for instance, be that quantum mechanics could be shown to be wrong in believing that the Hilbert space vectors are pure probabilities; that would affect quantum randomness, and that would be a big finding for science.

Tom Addis: *a characteristic of quantum algorithms is that multiple states are dealt with almost simultaneously, and an algorithm reveals just one answer; if a process produced a probability distribution of answers, would the probability distribution be as expected if all the states were carried out?*

Yes; this is what happens in the Shor algorithm; it is run multiple times, to learn about the internal lattice space; some observations are found not to be relevant. The algorithm is probabilistic; it can be simulated for a small number of qubits, and shown to be within expected tolerances.

Tom Addis: *if this is so, then a quantum computer would be ideal for game theoretic calculations with enormous states and decisions made on probabilities?*

Even small games, such as a business dilemma game with strategies and strategy algorithms, have a rapid build-up in strategies as rounds cumulate, so this might be a potentially a useful application. Quantum games exist that take variations on game theory, but I don't know any applications to game search spaces, stable spaces.

Aaron Sloman: *evolution has produced more sophisticated varieties of info processing than we can understand; Penrose suggests that evolution is way ahead on quantum algorithms; can we learn from nature?*

Some people are exploring this, but I do not think they'll get anything. Penrose is wrong in his argument; we can do non-algorithmic things but we can get things wrong [comments re older physicists of potentially libellous nature]. The real problem in knowing how to build complex algorithms; this may be impossible to do in the ways that we traditionally deal with complexity, perhaps because complexity might be an emergent property of classicality.

Rob E Smith: *There seems to be a hysteria over the Quantum Fourier Transform that identifies the period in the Shor algorithm, a hysteria based on one problem that everyone thinks is hard. In fact, if a problem has an underlying signal with high complexity, Shor sampling may be so hard that it never produces a solution.*

The number of components is exponential anyway, so if one is trying to learn about more components than can be handled, one wont get there, but in such a case there would also be a problem just writing down the answers. The problem comes down to the difficulty of storing quantum answers and reuse.

Shor is a very narrow application; there's a lot of non-obvious structure within it which makes it very hard to generalise, despite many attempts. All the known variants are very close to the original. It is a showpiece; it is popular; but it is tiny.

2.3 First Panel, chair Andy Adamatsky

Panel: Julian Miller, Jonathan Mills, Klaus-Peter Zauner

The panel was structured as a series of short presentations with questions after each, and, at the end, a more general discussion of the topics covered.

2.3.1 Andy Adamatsky

Emergence is bullshit with no results; quantum computing is fun because it challenges the mind; by contrast, chemical and polymer computing are challenges and real.

A quick introduction to reaction-diffusion computers proceeded to an explanation of chemical-controlled robot navigation, based on the response of a chemical medium to stimuli and collision based logics, based on chemical diffusion wave interaction. Voronoi diagrams were derived, and robot hands controlled by shading BZ; photo-sensors convey the control outcomes as states to robot.

Julian Miller: *what timescales?*

Very slow. The Voronoi diagram takes about 30min; the robots move about 1cm per 2-3 minutes.

Julian Miller: *what are the applications?*

Embedded applications. For example, a chemical gel within a robot doing logical, navigational and decision operations.

Julian Miller: *The illustrated systems seem to use a lot of conventional hardware?*

Yes.

Kester Clegg: *Would smaller systems be faster, because the diffusion effects would be faster relative to scale?*

Up to a point; and speed-up would be limited by wavelengths; semiconductor electron diffusion would be faster.

Martyn Amos: : *What about scale factors? What is the largest possible system*

These systems can go big; room-sized wave generation would be possible.

Martyn Amos: *What are the energy implications?*

A room-sized application would require a lot of energy, but a reaction-diffusion computer could integrate in "gelbot" that hosted photo-electric generation.

Leo Caves: *To what extent does the nature of medium here restrict the time and scale, is BZ oscillation the only possible medium?*

Any excitable medium could be used, including PNP devices, semiconductor sheets, and excitable tissue. It is easier to experiment with chemicals such as BZ.

Susan Stepney: *The experiments described can be summarised as analogue implementation of digital computations; why?*

These experiments are selected because most old-school computer scientists challenge the possibility of chemical computing. By replicating known digital computations, we can show computer scientists that reaction-diffusion systems can do what they recognise; indeed can do anything.

2.3.2 Klaus-Peter Zauner, Computing with macromolecules

Computing started from a need for number tables. These were calculated originally by people, following precise instructions. Turing sought to generalise these manual approaches as if there were infinite paper and time, and thus unlimited computation. It is often assumed that if we have more speed and more space then anything computable is in reach with classical computers. We still have computers that are good to compute tables, but do these computers also represent the right approach for any information processing problem; is it the right paradigm for the substrate?

The characteristics that make classical substrates good include tolerance to parameter variation and aging; resilient to noise; good properties of restoration (high gain); the ability to interconnect. Faster substrates have been less important than these properties; slower components gave bonuses in these other areas.

A computer is anything from which we can interpret the final state; there are lots. I work between inanimate and animate. Synthetic biology can produce simple cells and circuits, and can also build up from the underlying chemistry. Self-assembly molecules can be created, but have poor scaling because of diffusion speed. Protein computation reads conformational changes using enzymes or spectroscopy, and is already used in image processing.

One implementation is a robot with light censors which exchanges chemical signals with a dehydrogenase enzyme; this keeps the robot within light ranges. We are working on miniaturisation (lab-on-chip technology).

Slime mould can be used to provide single cells that grow circuits under light stimuli. The cell oscillations have been used to drive robot legs. There is a lot of setting up to run it, but a chip is being developed that incorporates the slime, interfaced electronically; the slime is confined to hollows in the silicon.

Jonathan Mills: *what is the control?*

Cell walls.

Jan Kim: *You are using the laws of nature to arrive at a solution, but the solution requires interpretation of states. You could claim that anything represents an arbitrary problem and read out any arbitrary solution?*

Anything can compute itself; if you throw an object, it computes its trajectory. There is a trade-off between universal codings and efficient one-off computation. We need to move from overall universality to some degree of specialisation, but must avoid encoding the problem in terms of its solution; the key is to find right balance.

[ed: also, if the interpretation requires even more computation then it is not helpful.]

Richard Overill: *why slime mould?*

Biologists have extensively studied slime mould, and its habit of sporing under stress. Also, the slime mould has a useful facility for co-ordination under chemical signalling, and the researchers could see how to interface with it chemically. It additionally has optical and electrical responses, but for this work we only use plasmoidal single cells.

Andy Tyrrell: *How do you get the slime mould into the initial structure on the chip?*

It grows in to it. The mould likes some surfaces, and will stay within these unless it gets very hungry. The chip can run for 6 days, but eventually the mould will try to run away to find food; no food is provided on the chip.

Martyn Amos: *What happens in the black bits?* [This referred to black areas in a graphic of the slime-mould chip.] They are holes; the mould oscillates by the thickness of its layer; this is detected by a high-frequency bridge across it. An alternative form has light at a wavelength that does not affect the mould shone through, and this reveals the oscillations to a detector. Information processing is all at the molecular level.

Kester Clegg: *how does the mould control robot legs? What about error correcting, smoothing etc*

There are six sensors; each part of the cell connects to one of the legs; if the sensor detects light, it sends it to the cell bridge.

Kester Clegg: *How does light influence the cell phase?*

This uses a regular anti-phase; the chaos phase changes the robot direction. The system is not stable for a long time, but light input influences change of state.

2.3.3 Julian Miller: Evolution *in materio*

Simon Harding and I are using computer controlled evolution to change the properties of a physical substrate, and thinking about methods for programming the materials of computational substrates. This raises questions about how to get non-quantum domain matter to perform computation at the molecular level. The laws of physics do computation for us, but how can we use it for general purpose computation? How can we emulate what natural evolution has achieved?

We define computation in a similar way to the earlier speakers. We can supply data input via physical signals to a piece of material; controlled physical signals can alter properties of material to get outputs that can be used, via molecular interaction. We use computer controlled evolution, details in proceedings.

Liquid crystal is used, because we can manipulate its properties as required, and wipe it clean between runs. In searching for a solution, millions of configurations are tried, so it is essential to be able to clean the substrate easily and impose new configuration; this limits what materials could be used. Liquid crystal is good for experimentation, but is not necessarily a solution to molecular computation; we could use these techniques to manipulate suspensions of nano-particles etc.

So far, we can apply two square wave signals to an electrode on the liquid crystal display and evolve configurations of voltages that output high for one frequency and low for the other; these signals are used to control a simulated robot in real time.

The motivation comes from Adrian Thompson's FPGA evolution. We wanted to see if we could evolve circuits in liquid crystal; in fact, the evolution is easier in liquid crystal (ie it takes fewer evolutionary steps that Thompson's FPGA evolution).

In the robot control simulation, one PC controls the frequencies; another simulates the robot movement according to the responses, It is fairly fast to evolve the controller, and some controllers generalise, controlling the robot navigation through more than one maze.

We can also implement boolean logic, to convince doubters that conventional computation is possible, but it is clear that this is not the best way to tackle digital computation.

A recording of four simulated robot runs was used to show the evolutionary stages of the controller.

Despite some very good solutions, sometimes the controller displays some apparently dumb behaviour; this arises where the context limits the signals that are detected.

How does the liquid crystal remember the strategy being evolved?

A PC provides the voltages to the liquid crystal; these represent the program that controls the robot; we then mutate the voltage configurations — so the the memory is on the external PC not in the liquid crystal.

Martyn Amos: *What properties make liquid crystal better than and FPGA?*

Many, but we don't yet know the key ones for computation. We want to find a way to systematically discover what computation is possible in unusual materials, to short-circuit inspiration-based research.

Richard Overill: *liquid crystal remembers shapes?*

Yes, but we're not using that property; molecules do the same thing under same inputs.

Andy Wuensche: *If you do same simulation twice, does the robot take an alternative path?*

If we keep reapplying evolved voltages, there is a tail-off in the ability to complete the task. However, it is easy to evolve a new solution; this takes less than a minute.

Andy Wuensche: *Is it susceptible to tiny changes in starting conditions?*

No. There's a lot of noise in there already, so solutions are often general; even the absolute voltages vary within parameters.

Chris Haragan: *how is the navigation achieved?*

There is a map that simulates chemical diffusion; this determines how hard it is to reach locations on map. This give a fitness function and the required control.

Tim Clarke: *what happens if you change the crystal?*

Each solution is specific to a particular crystal, but could easily evolve another solution on another crystal. Similarly, having trained a crystal, if you then apply lots of other programs, the original program of voltages no longer works, but the crystal can be retrained quickly.

2.3.4 Jonathan Mills: extended analogue computers (EACs)

In 1993, Roubel came up with the extended analogue computer (then died). There had been doubt that an EAC could be built; in fact, like a Turing Machine, the full generality is not implementable, but we can build a bit of it. The EAC still looks similar to early models (1995): conductive plastic with integrated circuit connections. Computation can alternatively use gelatin, cultured tissue, or other conductive or semi-conductive material. Our EAC can be run over the Internet from Indiana.

We are working on proof of machine logic, a Lukasiewicz logic that allows proof of all the elements (not conventional maths proof).

Our stating point was morphogenesis. Analogue computers don't reproduce accurately, perhaps this is nature's own non-cloning theory? The EAC can put things together, illustrated by student experiments finding letters in butterfly wings and vice versa.

The EAC has a universal sheet. The fastest is a silicon lattice — we don't know how it computes, but we know how to perform input and output. This EAC can solve partial differential equations in sub-nanosecond time. The plastic sheet does it in microseconds, and gelatin in milliseconds. These are two-dimensional partial differential equations; we could do n-dimensional partial differential equations with connected machines. Roubel's original paper foresaw unlimited dimensionality (including non-integer dimensions). In this way we can compute image diffusion. Essentially, we are computing with physics.

We are working on the shape of smell (for US homeland security). The research uses a scent characterisation model from perfumery and aromatherapy, and derives a classifier for each smell. Reusable chemical processors can be used to react to smell and record its shape. We are now working on sensors. This is a cheap technology; we could embed it in walls, giving intelligent walls that can smell.

The EAC on which some of the student work is conducted was illustrated as a bucket containing Jell-o and salt, containing a three-dimensional lattice of 27 connections; this solves three-dimensional partial differential equations. Possible applications include data mining through Jell-o to match three-dimensional shapes.

Every digital computer has an analogue one within it trying to get out, and vice versa.

EACs can be used for image rendering, using multiple sheets with recurrent connection. The functions are configured manually or digitally, using evolve and replace, but configurations are potentially programmable. We can do absorption and reflection, and are now working on diffusion. When finished, this EAC could compute arbitrarily large renderings.

Julian Miller: *How is the EAC programmed to allow solution of partial differential equation?*

A vector of bits is programmed to be active or not. There is a correlation between each problem and the system to solve it; this means that GAs can find solutions quickly. However, the correspondence required by GA approaches breaks down on multiple sheets, due to feedback. Their programming is an open question.

Sam Braunstein: *Is the medium passive, or is it actively changing its properties?*

The medium doesn't change, but the functions and feedback change. Power is micro-amperes for silicon, and milli-amperes for other media.

Aaron Sloman: *The EAC website limits what can be done?*

The website has been configured so that you do something interesting, from our perspective not yours, the visitor. We want people to try anything, and when we detect something of interest, we will work with you to take it further. The website is on Bryce Himebaugh's area, http://www.cs.indiana.edu/cgi-pub/~bhimebau/eac.cgi.

2.4 Panel discussion

Andy Adamatsky: *Could the invited speaker comment on the panel presentations?*

Sam Braunstein: a lot of the schemes do analogue-like computation; this is powerful in some circumstances. Naively, this approach might be good for approximate calculation, but noise accumulation causes problems. It is thus perhaps best for niche applications requiring low precision — you would not use them for completing tax returns.

Paul Marrow: *how do these approaches fare in terms of durability and portability?*

Julian Miller: They give us time. These paradigms are no more outlandish than transputers when they were first introduced, and the initial perception is similar.

Aaron Sloman: for many of these devices, you could grow or evolve something, but not copy and distribute it. However, if the evolution uses a GP, one could copy the program.

Jonathan Mills: you can't clone device when computation embedded in device; can't clone computation itself.

Julian Miller: It is not usually a problem that the devices are not identical; every device does approximately the same thing.

Andy Adamatsky: molecules are very stable over millions of years.

Aaron Sloman: . . . but humans are made in a very precise way

Andy Adamatsky: Life uses non-linear functions that we can't reproduce or mass produce; proteins use non-linear effects on thousands of atoms.

Sam Braunstein: It is clear that these types of schemata wont replace desktop computers for their niche stuff. However, ubiquitous computing will have computation with everything in an infrastructure that could well exploit these for smart objects.

Jonathan Mills: IBM and others are interested in wet stuff because of speed — physics-based simulation and rendering in games is a key potential use of fast partial differential equation solution. It is interesting to ask Microsoft what it would be able to do, for example, if it had a partial differential equation solver like this.

Rob E Smith: Cinematographic animations and special effects, in films such as Troy, employ physicists and finite element analysis etc to try to stop the effects looking unreal. Cinema needs to look right; it needs this sort of imprecision to make it look real. Inherent variation or randomness is definitely a feature not a bug in this domain.

Jerzy Gorecki: *is the randomness of the same form as nature's?*

Jonathan Mills: it is not algorithmic randomness; it passes spectral tests, but this is still an open question. Analysis of the partial differential equation solver predicts the same types of correlations as nature; this would be easy to test. In fact, it would be interesting to see if one gets the same results on all these approaches; statistical analyses would be easy to do.

Geoffrey Canright : *Does the fact that there are no subroutines in quantum computations mean that there can be no modularisation? This might also apply in the other analogue form — modules have precise interfaces, which is important for constructivity.*

Sam Braunstein: In quantum computations, some limited gluing is possible; this may be a big problem for the design of more complex algorithms, but we don't yet know if it's fundamental. We can use other languages for thinking about computation, such as fixed points. The more significant problem for analogue design is noise accumulation over iterations.

Jonathan Mills: with students, the modularity issue constantly arises; subroutines are primarily ordered by time, but students must organise spatial subroutines to find an answer. Computer scientists are thinking about it wrong.

Geoffrey Canright : *classical divide and conquer, analysis testing all break down without modularisation*

Sam Braunstein: we can simulate quantum computation, but that does not give the speed-up; we don't want to just repeat what already been achieved.

Rob E. Smith: engineering divide and conquer is an illusion; all sorts of couplings are hidden. We don't need crisp boundaries.

Jonathan Mills: divide and conquer spatially is applicable to CAs.

Chris Haragan: *Quantum computation focuses on the quantum effects of particles in a system; analogue focuses on behaviour in a not-quite-constrained fashion; both have the random/continuous property, so can we have a focus change between the two and quantum precise results?*

Sam Braunstein: continuous quantum variables have been a success in quantum protocols. All current quantum computation problems are inherently digital at the moment. Alternatively, we could use qubit store and continuous variables to perform communications — lots of hybrids are possible.

Jan Kim: *Regarding non-clonability, is imprecision a way forward, and can it be formalised beyond "it looks right"?*

Jonathan Mills: answers exist related to domain of generation. You can measure using a Lipschitz metric where small perturbations propagate to small changes; this is implicitly related to the physical basis of computation.

John A Clark: There is a sense in which modularity is always an illusion — we believe we see modules when we understand a structure. Quantum computation offers way points, though we don't yet understand these; they could form modules if we understood them, and then we could evolve a suitable understanding of modularity.

Irene d'Amico: *in relation to quantum decoherence, there is a theoretical partial solution under error correction. In analogue computation, is there a theoretical equivalent solution for noise accumulation?*

Jonathan Mills: There are different approaches to noise tolerance; the model "decoherence" can be plotted on a binary tree, with noise tolerance greatest at the root. What we know about analogue systems is that they are robust — you can blow holes in a system and it still computes. Although we don't know the relationship between physical computation and the domain of generation, it still works.

GianLuca Tempesti: *In modern digital circuits, a lot of the redundancy has been removed; in these analogue systems, a lot of redundancy exists, and this is causing problems of scalability; redundancy implies that the systems can't scale down very far, and the absolute limit would be where the analogue unit of computation becomes a single electron.*

Jonathan Mills: Because of the redundancy in small particles, failures are very small; results can be interpolated. EACs do not scale easily, but reaction-diffusion machines can be very large.

Klaus-Peter Zauner: noise at a molecular scale can be used to drive things, but need to predict what happens at any instruction, because programming requires determinable responses. Chemicals have many side effects so prediction is hard; humans try to impose precise logics giving inefficient machines. Reusable structures and reusable evolutions are useful.

Geoffrey Canright: we need a different way to think about modules

Julian Miller: divide and conquer doesn't break down, it just varies.

Klaus-Peter Zauner: evolution does divide and conquer.

3 Session 2: From Classical to Non-classical

3.1 Invited Speaker: Luca Cardelli: Abstract machines of system biology

An introduction to systems biology, focusing on the information processing aspect.

A cell uses energy and materials to produce more cells, copies of itself. Information processing is important to survival, so what are underlying programming paradigms? A cell has a lot of structure, but we need to understand the functional architecture.

A toolkit view of cell components reveals regular composition to complex systems such as DNA, which is constructed from nucleotides; all proteins are built from about 20 building blocks.

We can identify a gene machine, a protein machine, and a membrane machine (there are also sugars; but they are not covered here). For each, biologists have notations to describe their understanding. Each bio-chemical machine has computational power under its own principles, but all the machines interact with each other. We need to understand multiple computational paradigms and interactions.

Protein machine:

Biologists use cartoons to explain the shape-matching of proteins and enzymes. These model the flipping of boolean switches at the surface that allows configurations to evolve. A cartoon summarises a lot of chemical interactions in a very abstract way.

Networks of proteins are very complicated. Proteins are very precise structures built from amino-acid sequences; they form the basic units of the machine.

Kohn introduced molecular Interaction maps, notations to show how proteins interlock and how the switches go on and off. Kohn maps are used to construct circuit diagrams for protein interactions (for instance, the p53 protein). One major diagram summarises hundreds of pages of research, and is a good abstraction of complex system.

The protein machine instruction set comprises on-off switches and attachment points. Changing either the switches or the attachment points changes the operations that are available.

What is the software to run on this instruction set/hardware? Kohn maps are too static for true behaviour models. Various formal or dynamic descriptions are used. Examples include diagram summaries followed by solution of partial differential equations or process calculus; here the formalism just captures the diagram contents. We can simulate the protein machine to see its behaviour. (MAPK), observing, for example, digital switching behaviour emerging from the smooth behaviour of the components — the MAPK cascade happens across molecules; it is a molecular switching process. In effect, we have synchronised bio-interaction.

Gene machine:

The gene machine presents asynchronous bio-interaction.

In this machine, we consider DNA, messenger RNA and proteins. Analogue proteins arise from digital DNA; the known mappings are one way only.

The gene machine instruction set comprises the gene, a long string of DNA; DNA has an output region with protein to bind to other genes; input is achieved by the inverse process. The instructions include stimulators and inhibitors. In scale, *E.coli* has 1Mbit of genetic code; yeast (which has nuclear cells like ours) has 3Mbit. Humans have gigabits.

Biologists' notations show linkages among genes. Again these form circuit diagrams, showing genes as gates; there is a lot of feedback in these models. Biologists use a different notation for the gene machine, and a different programming model based on message sending.

The gene machine programming model is not like conventional computation — there is a fixed output, which is not a true function of the inputs; there is inhibition, so it is not a petri net; it is not communicating sequential processes, because the machine has feedback to avoid self-deadlock; it is not message-passing because the messages have behaviour; it is not data flow, because the loops that emerge would deadlock a data-flow model. A better characterisation is of stochastic broadcast (spam) with stochastic degradation. Could we program in this model?

The biologists again have lots of notations, including gene gates and circuits; simulation captures alternating signal patterns.

Membrane machine:

The membrane machine is a Turing complete model of computation. This is illustrated by an animated model of the sequence of dynamic entry and secretion of proteins.

Again biologists have their notations, using arrows, static representation of a dynamic process (like state charts), fusion and fission of membranes.

The membrane instruction set captures the ways in which membranes change state. These are defined using the terms mate/mita, drip, bud, exo/endo, pina, phaga. Membranes wrap around, divide, merge etc., to transport proteins as required. It is fairly easy to formalise the instruction set in two dimensions, but rather more interesting in three dimensions.

Local molecular machinery in the membrane does the operations.

The algebra is nice; it is compositional. These composites are observed in reality.

For the membrane machine, biologists have fewer notations; one possible approach is Cardelli's bio-ambients.

Membranes can run algorithms. For example, cholesterol exists outside cells; inside the cell there is a vesicle that can degrade cholesterol, but cholesterol is too big to pass through the membrane. Instead, proteins bind to the cholesterol and trigger proteins on the membrane. This triggers the instruction that split off a bubble, for transport into the cell. The receptor protein is detached by merging with a vesicle that splits off receptors; the remaining clean vesicle can merge with the degrader vesicle. This program includes concepts similar to stacks, iterations, loops etc, and specific goal. The operation uses millions of atoms and looks wasteful. Many other operations exist, including viral reproduction, protein production and secretion etc.

So, from an information processing viewpoint, we are looking at many computational abstractions, each with its own models, of different size and speed. The protein machine operations are very fast, gene operations take minutes or hours; membranes compute in minutes or less — for instance, neurons fire through membrane merges.

Stochastic behaviour happens because nature is stochastic. A biological system can be simulated by deterministic or non-deterministic differential equations — stochastic equations cause the system to fluctuate, deterministic ones case it to stabilise. Both have a fixed point, but the deterministic one stays there, whereas the stochastic one may be bounced off the point, when it will follow a trajectory back on to it again.

A lot of experimental data is being generated, but biologists lack ways to describe what they are finding. There are languages that can be adapted, for example, π calculus and process calculi, but perhaps there can be an underlying integrator or integrators to represent all aspects.

3.2 Questions

Wolfgang Banzhaf: *what is the role of transport in abstract machines?*

Membranes transport material; vesicles are cargo transporters; proteins signal to the outside environment or bring things to cell surfaces. A protein is decorated with sugars, iteratively in different configurations — the result is structurally significant. Transport is an assembly line accumulating with proteins.

Jerzy Gorecki: *There are big differences between deterministic and nondeterministic systems, as in, for example, non-linear chemistry. Biological systems are made of small number of types; we need to consider the mass action dynamics of chemical reactions — these need a lot of types of molecules, but only a few types are in each sort of cell; the chemical equations are for average concentrations, so the results obtained for chemistry with a small number of molecules diverge from the standard results..*

Classical computing has hit similar problems. Here, the algorithms are used for stochastic small-scale behaviour, smooth behaviour.

Jerzy Gorecki: *Even for model parameters typical of biology and a few hundred molecules, there may be phase transitions, bistables etc seen for larger numbers?*

Many recent papers in physics and chemistry emphasise the importance of stochastic processes at a molecular scale, sometimes with noise to amplify signals.

Christopher Alexander: *In the models, objects and operations are different in scale. Can you build a simple model with all these things going on as envisaged and run it?*

There is a trade; it depends on abstraction levels. For example, take viral infection of cell with membrane and molecular operations. You could model this atomically, but it would be hopeless. However, at a molecular level, you have a program that could be run. So, you need to pick the abstraction level that is suitable. Differential equations were said to be not useful because we cannot handle the required large numbers of them, but conventional software handles lots more code, so there must be ways of representing systems that would work computationally.

Christopher Alexander: *[a question relating to Shapiro and finding other notations]*

This is illustrated comparing an example of the normal chemical formulae for reactions and one of the new notations — Na + Cl in chemical formulae focuses on reaction. In a dynamic formalism such as petri net, the focus is on transitions. The chemical view explodes as the network of interactions grows; it is not compositional. Instead we want to represent a system in a dynamic style that shows what the chemicals can do under different stimuli etc. This can also show where synchronisation is needed, and identify common transitions, divergences and convergences. The diagrams translate to a text process calculus. All three are the same model, but represented in different ways.

Aaron Sloman: *The impression is that biological structure is crucially important at all levels; by contrast, bio-inspired programs operate on vectors of numbers, not changing structures. Again, biological structures are inherently three-dimensional but the diags shown are two-dimensional, which constrains the operations modellable etc. Are there crucial three-dimensional aspects that the models lose?*

Yes, we do need to capture these things. For example, the nuclear membrane is structured as two spheres connected by holes; it is a highly complex structure that can't be represented in two-dimensions. Every time a cell splits, the division must replicate the complex membrane structure — how do we represent that? We still need new notations.

Julian Miller: *Could you make predictions that biologists can validate?*

We need to validate out models; prediction is one way, but is rare. Biologists use post-diction (predicting the past), but this is also hard. There have been some small successes but no breakthroughs. We need to watch the biologists' ways of recording things then compare these with out formalisations by running through the process; if we then encode what biologists know, we can iteratively validate our models. Predictions will come eventually.

Colin Johnson: *With regard to biological structures, are there tools for doing process calculi? will they answer biological questions?*

There are many useful things — simulation works; program analysis is useful, looking at reachable states, and who can encode whom. Model- checking for hardware and software can be used to ask interesting questions about necessary step points between states.

Colin Johnson: *model-checking questions about temporal properties is likely to be useful for biologists.*

This sort of research is now appearing in conferences; stochastic model checking would be even more useful.

Richard Overill: spaghetti vs modular programs.

Jan Kim: *NaCl comprises independent ions, but H_2O interaction is harder to model and analyse, because the components are not independent.*

We need the process calculi as well as the process automata; we can't represent all the steps graphically in very dynamic behaviour.

Jean-Louis Giavitto: *Does the asymmetry in descriptions cause problems?* It's in the calculus not the chemistry. Proteins are complementary shapes. A model of binary interaction is an amazing accident.

4 Panel, chair Cristian Calude

Panel: Samson Abramsky, John Clark, Peter Welch

4.1 Samson Abramsky: Conversations in NSC

Non-classical computation is starting to work with other scientific disciplines; information flows in both directions, with unpredictable outcomes. For example, the interface with physics is promising things that cause us to rethink basic ideas of computer science such as processes, information, and time-ordering. These have been talked about in concurrency, but now we need to relate them to physically realised systems. Luca's talk was doing the same sorts of things re computer science and biology. We could consider concurrency theory as discrete physics — Petri and Lamport both have papers that are explicit in this area. In the analogue computation discussion, we are still using computation to process output from input, but with the Internet we have a wider idea of what we can use computers for — the Internet is not computing a function; process calculi express more general computations than mapping inputs to outputs.

Information is physical but physics is logical.

In relation to modularity, which is closely related to compositionality, we have compositional descriptions; can we make chemistry (Luca) and physics compositional? Because quantum computing is re-examining quantum mechanics from a computer scientist's viewpoint, we can see how computing tools can be used to make structured and compositional quantum models — entanglement and Bell states — to encode correlations among spatially separated systems. This opens a way to compositional tools of computer science, through, for example, teleportation with classical communication — two classical bits performs communication of much more quantum information. The models exploit correlation by measuring the particle at one end of the system causing the "collapse of the state" of the system in the target; the measurement outcome is then passed to the collapsed end to restore the state. The literature is impressive — matrices and calculations — but it is not clear what is going on. Questions used in computer science would be useful — what are the abstraction principles, the high level languages? How can we build descriptions of protocols? etc.

Looking at bringing in types in an appropriate way to structure descriptions of physical systems can provide compound structures, different possible outcomes, and even parallel worlds! In a consistent way, we talk about the key feature that uses the outcome of a measure to influence a subsequent computation — measurement-based computing to address fault tolerant computation.

Category theory provides a logic of calculus for the quantum work. Some forms of functional abstraction fall out; some notion of modularity or subroutine becomes apparent, but it is not clear if this helps us to build more complex quantum algorithms. Richer notations rather than just circuits may help. No-cloning puts us in a linear or resource limited context.

There is a nice algebra for the quantum information flow from entanglement, and a nice visualisation. This is an algebra of compositionality, which works something like straightening piece of string. The algebra can be thought of logically.

(unidentified questioner): *How are these notations different from those of Louis H. Kauffman?*

They are similar to the diagram notation for tensor categories, but not concerned with knots and overlaps; there is a different purpose but similar structures. Another point of doing the work is to look at the different sorts of structures that emerge, in the same way as Luca Cardelli described. We are discovering natural relationships. This leads us to observe that apparently different phenomena now seen to display similar structures. Compare this to the analogue paradigm — we build up systems by parallel composition and feedback; the same holds here, but we add polarities and reversibility, the extra structure of complex numbers and conjugation and reversal in time. These are characteristic of the complexity of quantum computation, but influence a compositional treatment of analogue too.

Sam Braunstein: *what about multipartite systems?*

The primitives need defining. Systems with many parts and complex wiring can be studied; there may be primitive general behaviour that can't be expressed, but we have not found a good example of what can not be done in the language.

Cris Calude: *You said that the formalism may not lead to new algorithms. By implicit analogy, you must expect understanding from using such an approach. A journey is like a challenge to understand the world of info processing, and not simply the goal of designing new computer; this will be a side effect only.*

Yes.

John A. Clark: protocol analysis in a logic language can be used to analyse, but another form of proof is to write a program in a suitable programming language; the same is true here.

Jean-Louis Giavitto: *In quantum computation, simulation can be done on a classical machine, so code that would run in polynomial time on a quantum computer is no longer polynomial. Where is the fundamental complexity that removes polynomial time characterisation?*

Sam Braunstein: This was Feynman's first focus, and was taken on by David Deutsch. It involves a path integral of Schroedinger equation, a huge amount of calculation. When there is evolution in time, there is exponential growth in calculation over time. Feynman reasoned that there was no reasonable hope of a classical simulation of a really complex quantum systems, but a system based on quantum mechanics could simulated efficiently.

4.2 John A. Clark: trajectories and other side channels.

Cryptography entails trying to find what others know, not finding new things. Smart cards, for instance, perform encryption functions. The maths take on cryptography is as a black box and a key; this has been the traditional view for at least a century [libellous statements about group theory mathematicians]. However, external relations exist; if we compute a function, the computation does work and consumes resources. This led to spectacular attacks in the 1990s from looking at the physics as well as the computation outputs, for example Kocher's timing attacks, Bellcore's differential power analysis, and various forms of physical fault injection.

Ideal implementation don't have faults, but computational dynamics, or the way that you do it, leaks information; the leakage channels are known as side channels.

Search (as used to factorise or crack keys), for example, can be done by simulated annealing. If you treat the system as a black box, then the search is only guided by results; it is a waste of resources. If you open the box, and, for a problem p, try to minimise cost function, you can watch the search trajectories. One run may not help much, but if you look at set of trajectories from repeated searches, then a lot more information is available. So, we need a less half-hearted approach to search heuristics. For an annealing algorithm, you can watch as much as you care to instrument.

Turning to fault injection, evolutionary computing tries to solve a problem. We could perturb the problem; try to solve different problems and then solve mutants. By watching the trajectories we might be able to discover things about the family of solution that relates to the original problem. We can do some big mutations and look at the solution distribution to construct a solution to original problem.

The Perceptron problem is NP-complete. Algorithmic complexity analysis cares only about the worst case, but we consider the current case. Essentially, the problem has a public matrix, and we must find a secret vector that produces a given solution histogram. We run simulated annealing and watch it happening. A timing attack reveals that some bits stick early, and that these are fundamental to solving the problem — the ones that stick early have a reasonable probability of sticking at the correct value. However, some that stick early are wrong, so we can perturb the problem and look at the average properties of proposed solutions. No direct attack has yet been evolved for these crypto systems, but indirect attacks such as this do work. Commonality is probably correct. The technique works best on mutants not on the original version of the problem! The solutions counteract biases in one problem. Note that an attack that solves 60% of the bits makes rest easy to crack.

Lessons: patterns emerge in repeated runs; move from black box analysis to white box observation to get information on the search trajectory; look at the statistics of results; look at related problems. Cryptographers spend thousands of years and countless MIPS factorising a number by mathematical approaches, but are not yet prepared to put any effort into heuristics. Profiling can generate countless examples at will, to learn features of a search process and of the information being sought.

Susan Stepney: *In relation to bio-inspired techniques, real systems use much larger number than we do?*

We use numbers — a billion runs of an annealing simulation might be used to crack a key; vast power should or could be used. A problem engineered against algebraic criteria can be attacked by seeking structure in things that were engineered to reveal none to a different form of attack.

Jan Kim: *molecular level investigation of biological systems also find it hard to conclude what's going on from a few measurements. In real biology, we may have to destroy the system to analyse behaviour. What criteria are needed for side channel attacks, and can bio-systems have side channel attacks?*

We rely on engineered infeasibility. If a system is provably secure, it probably is not secure.

Richard Overill: *How does this relate to the work of Bond et al at Cambridge, looking at hardware engineering?*

This is logical; they do hardware tracking as well as fault injection, hardware timing attacks. Clearly, this is related but they do it in hardware.

Jonathan Mills: *observed bit patterns change dramatically between answers that are close to correct and those that are far from the right answer — have you any observations on intermediate forms?*

That's why these problems are hard. A small change in cypher code may take you far away from the solution. But every cost function achieves something; bizarre cost functions or even arbitrary ones. If you work out a correlation with the secret, it'll reveal something even if you don't know why.

4.3 Peter Welch. Barriers, mobiles, semantics and platelets

Very tiny, very fast, very many. Concurrency helps and is fun. Systems of networks of communicating processes feel right... Luca Cardelli's last question and answer go this way; the example of Na and Cl is nice; there is something there that we ought to get a grip on.

We need simplicity; concurrent systems should simplify things for us. Current practice views concurrency as making things more complex, something that is only used for performance reasons. We need to get out of this mind block. We want rich structures of dynamic communicating processes responding to the environment and their own agendas.

Over ten years, we have developed libraries, toolkits and languages to capture ideas from process algebra — processes, communications, mobility, location awareness and computation that takes place only among local interacting processes. occam comes from the old transputer world; very fast computation but very static. We add π calculus semantics to occam, with the aim of developing new process algebrae amenable to every-day use by engineers. We can give this to first-year undergraduates, who can use the language without training, programming Lego Mindstorms in a very natural way.

We want to build complex systems without complicated programs, things like multi-level simulations at the level that Luca Cardelli is looking at; occam-π offers a possible alternative to his approach.

[To illustrate, a system with occam-π synchronisation on barriers was described.] The barrier equates to a multiway CSP event; we include higher-level patterns such as barrier resignation. Note that barriers are taken from classical approaches, but we introduce mobile processes with parallel registration on barriers; if processes are not parallel, there is a interleaving semantics. Barriers are used to synchronise the phases of a parallel computation.

Using occam-π, we can run millions of parallel processes [on a single PC]. The language preserves the safety rules of classical occam and adds new concepts. The barriers are safe and automatically managed by the way a program must be written; if the programmer writes it wrong, it does not compile.

[The mobile processes were illustrated for busy and lazy platelet models.] In the busy model, a CA models the platelets; barriers are used to synchronise update and display phases. In the lazy model, mobile processes are used, and only cells with platelets undertake computation in any phase. The barrier acts as a control on cells' access to processes, and the barrier can fire new mobiles via a generator.

This is like Luca Cardelli's account of biological animations; bumping, merging, dying are represented simply, with no shared memory race hazards, through the barrier semantics.

Going up to two dimensions, the platelets have neighbourhood awareness. In theory, the matrix topology can be n-dimensional and could be dynamically generated; it could contain wormholes and other features; topology cells may even have their own agenda. Consider just a plane. Each cell has a two-way service channel; the neighbourhood is just seeing client ends of neighbours' channels. Once this topology is connected, it just sits there. Agents are mobiles that attach to arbitrary points in space. Tail channels for mobile communication are registered in the cell that the mobile is linked to, so can have multiple mobile agents on each cell. Mobiles see their neighbours and can grab their channels; visibility is mutual. If mobiles on neighbours are compatible, then the mobiles can link and do business.

This is a design pattern; it may or may not be useful, but it is interesting to look at. The work involves Susan Stepney, Fiona Polack and Heather Turner and others, and is part of the TUNA feasibility study.

Andy Tyrrell: *Twenty years ago, people were trying to move* occam *into hardware — what now?*

That was one side of occam; here, we are looking at generating processes on the fly. This is not so easy in hardware. If we stick to static processes, we can do that in hardware — for example, a group at the University of Surrey is compiling occamon to FPGAs. Handel-C is like original occam for hardware. We are looking at dynamic aspects that allow us to do things that were not in original philosophy. occam is still a good hardware description tool.

4.4 Cristian Calude: randomness

Randomness has been touched on by many presentations. Not all the notions are clear. There are three types:

- pseudo — mimicking randomness in software
- algorithmic — using a notion from algorithmic information theory
- quantum

Imagine a hybrid computer, a PC plus a quantum random processor from idQ in Geneva — what problems are raised? does the computer pass the Turing barrier? Many quantum processes do, but they are not interesting here. What is interesting is whether the hybrid computer can solve a problem that is provably insoluble on a universal Turing machine. The answer is yes. Feynman's 1982 paper demonstrates that we can't reproduce quantum randomness on a Turing machine.

A more difficult question is, given such a computer, which is more powerful than the classical theory, what can you do with it? Little is known about what is possible.

Testing randomness is also part of this sort of conversation between computer science and physics. The Geneva physicists have applied all possible tests to their quantum randomness, and get the expected results, but the tests are finite — what about the infinite situation? We need mathematical tools to examine this. For example, assume a black box that, for every positive integer, returns a string of length n that is algorithmically random. A Monte Carlo simulation gives deterministic result overall, a simple test applied k times — if a composite result arises, the number is composite; similarly there is a k-times probability for prime identification. This is based on randomness encoding.

Computing with algorithmic randomness is powerful; does quantum randomness have similar properties? So far we only know that it can pass the Turing barrier. Now, we need to consider the results for other forms of randomness in a quantum random context.

Do we believe that in ten years we would meet again and be using quantum computers? The field does not evolve to universality in the way we understand for Turing machines. Silicon machines are very fast in executing small computations; quantum computers would be fast in other ways. The PC plus quantum random generator exists and might represent the future — silicon computers with add-ons that enhance and speed up appropriate forms of computation.

Sam Braunstein: *orthodox quantum theory is that quantum randomness approximates algorithmic randomness on average.*

Quantum mechanics does not need the probabilities of alternatives. We believe that there are many hidden properties in other forms of randomness.

John A. Clark: *The physical difficulty of calibrating instruments for randomness raises issues of how to set up instruments, and the precision of calibration.*

Quantum randomness is not new; the important news from Geneva is the engineering achievement — they produce quantum random streams using light, at matchbox scale. Reliable, mathematical tools help precise measurement. The von Neumann procedure can be used to sort out slight biases based on looking at pairs of bits, and filters improve the outcomes of measurement.

4.5 General Discussion

Susan Stepney: *several speakers use pictures — as systems get more complicated, we draw pictures, but we then throw the pictures away and write code. Is there a way to keep the pictures?*

Sampson Abramsky: people are moving into two-dimensional programming with graphs etc.

Peter Welch: We've never gone beyond two-dimensional pictures; there were graphical debuggers for some transputers that had three-dimensional fly-through ability. The maths is important; we must be able to write it down at the end, but the pictures help understanding. We really need a Linux illustrator. [An argument ensued about Powerpoint and Open Office.]

Jerzy Gorecki: In relation to the sorts of random numbers, any finite sequence can be simulated on a Turing machine; infinite sequences cannot, because they do not terminate.

Cris Calude: A Turing machine can't generate an infinite sequence, let alone recognise one; this needs it to recognise infinitely many random strings.

Jerzy Gorecki: A Turing machine makes one finite random string.

John A. Clark: quantum cryptography is on massive space; projections [on to lower dimensional spaces] are important; we have limited technology for extracting projections from large spaces.

[Humorous chat about the ethics of training babies and slime mould.]

5 Session 3: Bio-inspired computation

5.1 Invited Speaker: Przemyslaw Prusinkiewicz: Languages of Morphogenesis. Modelling and development and development computing

A historical perspective on science and nature shows logic with nature's inspiration. New problems generate applications one way and inspirations the other way. For example, Newton's calculus generated applications in mechanics and gravity, drawing inspiration from physics and more.

If biology is the area (cf mechanics), what are the equivalents of Newton's calculus logic and physics problems? Is there a need for a maths of biology? Brenner (1999) says this is the central problem.

Phenomenon	Computing metaphor	note
self-reproduction	CAs	the first application
neural systems	artificial neural networks	also in 1950s
evolution	GAs	
DNA manipulation	splicing systems	Hank and Rosenberg
cell operation	membrane systems	

Turing stated that equations are too complicated for analytical solution without computers.

Key questions:

1. what computational methods are needed to model the growth and development of multi-cellular organisms?
2. Can these techniques inspire new approaches to non-biological problems?

Perhaps we can use traditional methods such as calculus? The standard methods are not directly usable, because there is a new quality of problems; dynamic systems with a dynamic structure (Giavitto and Michel 2001) [illustrated by an animation of bell mustard growth] — the set of equations for growth grows over time; the number of variables increases — we also need automatic interconnection as the model grows.

A key problem is module identification. In a developing system, it is no longer convenient to use co-ordinates to model system components, because components move.

Richards and Riley (1937) devised a deforming co-ordinate system. However, this is not much better for components, because the deformations have to be mapped to real system co-ordinates.

If we use indices, that is indexed cells, what happens when a cell divides? For instance, the index sequence 8-9-10 might become 8-9-42-10, which means that we can't find neighbours, or could become 8-9-10-11, but then the indices of cells above 9 have changed, so we cannot use the indices as identifiers.

Weyl referred to the introduction of numerical co-ordinates as "an act of violence"; here the same applies to indices.

Lindenmayer (1968) proposed a solution: L-Systems are co-ordinate free and index-free. They are strictly limited to branching structures, but the spirit of L-Systems can be extended to volumes etc. Essentially, they form a class of languages in topological spaces for modelling development and other applications.

L-Systems are based on formal language theory. A string of symbols represent filaments then branching structures; at each update, the predecessor is replaced by the successor. For example, a natural language string such as ... *this_is_a_car* might have a rule to replace *is* with *was*. Replacement is by content not locations, so this gives, *thwas_was_a_car*. However, we can improve the grammar by using context, requiring, in the above example, that *is* has a _ either side. This would have resulted in a correct substitution, *this_was_a_car*. Each component is characterised by its content or state and the context of its neighbourhood. There is also a notion of the space in which operations performed.

Parametric L-Systems have parameters that are numerical attributes associated with letters. The grammar takes the form $< leftcontext > \quad < strictpredecessor > \quad < rightcontext > \quad < successor >$ [a sample derivation example was illustrated]

From here, we are able to solve reconfigurable partial differential equations, including a numerical solutions to the decay and diffusion equation [see the accompanying paper]. This allows simulation of diffusion, with relevant boundary conditions at extremes. The expression of the solution is very compact. We have not yet included growth, but we can simulate this with a threshold model (Lindenmayer, 1974, Wolk, 1975). For example, when a cell reaches a threshold for a particular chemical, it divides. This gives a nice model of continuous division, with peaks of high concentration then curves of decreasing concentration. The model can be used to solve partial differential equations in a growing structure, but also models a real filament organism, the cell bacterium, Anabaena. Anabaena has been fully sequenced and much studied, for example, Haselkorn, 1998.

An activator-inhibitor model has also been devised (Wilcox et al 1973, Prusinkiewicz et al 1996). This uses a mechanism with two substances in cell and a mathematical model of two differential equations for the two substances. Simulation shows a growing chain of cells with an activator-inhibitor system in each and communication by diffusion [nice pictures]; as in real biology, the simulation shows that some cells almost activate but then revert. The simulation uses deterministic differential equations, but we could have coupled stochastic differential equations etc to model more complex cell models with communications. Note that there is continuous diffusion but the cells have discretised reactions. The simulation predates the discovery of the gene models in real systems (Anabaena); this model predicted the since-observed biological variations.

This takes you from genes to phenotypes. A gene network and growing organism are in a continuous loop, with developments on the arcs.

L-System extensions include realistic biological plant examples. Transporting and signalling (communications) approaches are important for modelling higher plants, and as we move from cell level to plant module level. In the familiar L-System plant models we can model signal propagation in higher organisms (Lindenmayer) — a segment of branch structure signals on a condition (for instance, if segment x is green, segment $x - 1$ becomes white), and the signal propagates through. This is illustrated by a simulation of a plant that reacts to touch — simulated folding from point of contact.

Many plant signals are hormones. Lindenmayer had a model that showed how flowering zones move down a plant. I have also working on the mechanisms for this in real biological systems, with Ottoline Leyser. In the abstract model, the apex produces laterals; an upward signal propagates to top where it changes the top segment to a segment in the flowering state. Then a second signal propagates down, producing more laterals, where subsequently more flowers appear; as the program iterates, flowering moves down the plant and branching increases. All the mechanisms are from the program, but we can also use this to simulate complex interactions between a plant and its environment. For example, we can create pruning to a shape by bounding the space in which growth is allowed to occur. This forces more lateral growth, which eventually fills the bounded volume. The results are realistic, as shown by comparison with the Levens Hall topiary.

Plants are sensitive to light, within the plant mass as well as without. L-Systems have been used to simulate clover growing under trees. The base is a light-intensity model described across space, with parameter effects. A seed germinates in a light area; the rules prevent permanent spread to shade and cause a change in the form of the plants (long runners, few leaves) in penumbra. Run as a simulation, the clover performs a periodic invasion of less-lit areas; permanent coverage occurs only in the lightest areas. Similar approaches are used to model the response of trees to light in branch-shading, crown shape etc. These models are combined to simulate a forest, and can be used to plan planting in forestry, to get ideal forms for industry.

A model of the propagation of sugars in trees is used to predict how the number of fruit (peaches) affects size.

Bio-mechanics uses non-linear differential equations. L-Systems simulation is illustrated with a model of a hanging-basket plant, where gravity and tropism interact to produce a variable balance of downward and upward growth influences or forces.

L-Systems are a mathematical and computational tool that is still non-standard; they have many unique features and applications. Some of the other problem applications are as follows.

Examples from graphics include shape modelling — the dynamic subdivision of curves, and surfaces. Dynamically-changing structures cannot be computed by traditional methods. In an affine combination of points, a point is specified as a linear combination of the weights of other points, as if end points have attractional masses and the free point moves accordingly. This gives a co-ordinate free point system like that of turtle graphics... Chaikin's algorithm is just one (simple) algorithm for specifying curves from points. It relies on subdivision: the points are joined to a square, and the edges subdivided to form an octagon; iteration results in a smooth curve by cutting off all the corners at each update. Context sensitive L-Systems describe this succinctly, where the attributes are the positions of points relative to others. A complete specification in the language for L-Systems (written in C++) is about 10 lines long; within this, the production rule is represented more or less as normally written. Another such algorithm, the Dyn-Levin-Gregory construction, uses a dimension-two context; the interpolation maintains the original points as well as doing the subdivision. In the L-System language, it is the same length program as Chaikin's algorithm.

For surfaces, L-Systems can be used to model cell division and other problems, including the specification of smooth surface generation.

L-Systems work because they gather information about neighbourhood, producing a succinct summary, they then replace old parts by new, using connections within and between old and new, state and context. The extension to surfaces requires addressing context on a surface. This uses a mesh traversal on a vertex-vertex data structure with unique vertex labels. Rotation graphs (J Edmonds, 1960) are used to specify neighbours ordered around each vertex, so that the definitions of *next* and *previous* are systematic. This allows chains of succession and predecessors to be identified, and allows the extension to surfaces via paths to neighbours — if we insert a vertex, then we specify new neighbours and changes to existing neighbours. In this way, a triangular surface can be subdivided by three insertions and linkage rules. Repeated application smoothes the surfaces, and can be used, for example to shade shapes for development of perspective in three-dimensional graphics.

Spatial L-Systems work can simulate crack propagation by subdivision. A finite elements method is used for distribution of forces and deformations. Normally, fractures are hard to simulate, because the tip actions are critical and require a locally-fine mesh, but to provide a sufficiently fine mesh everywhere is not computationally viable. In the model, dynamic subdivision takes place as a crack spreads. The model assumes two linked surfaces, or layers in tension (under drying), so that cracks appear as one layer contracts relative to the other. Areas of high stress relate to areas of finer mesh, to model in finer detail at the crack ends. This has applications for modelling mud-cracks, bark patterns, and any other surfaces where the superstrate expands at a different rate to its substrate.

The L-System work adds to the table of phenomena and computing metaphors:

Phenomenon	Computing metaphor
development	L-systems and beyond

5.2 Questions

Gianluca Tempesti: *In applications of L-Systems, we sometimes know the final system and want to find the rules that generate it. This can't usually be done by systematic derivation?*

The problem of automatic inference of systems was identified early on; it is a problem of wishful thinking. To create a model, you need good understanding of the hypotheses of the system and a convenient expression. L-Systems allow a convenient expression and tests, but you can't automatically get to best solution. To find any underlying system is a complicated problem in biology — *Nature* journal would publish even partial results in this area — so it is unrealistic to expect any automatic mechanism. L-Systems also give a way to capture the results of biological exploration and to combine the various bits of biological understanding into systems. For example, we are currently looking at spiral organisation in plants, using a new study of molecular basis. We need to know if the theory produces the structures that are predicted, and to fill in any gaps using simulation.

Kester Clegg: *L-Systems are limited to branching systems, but are there any other limitations?*

The open challenges for research is more interesting. Luca Cardelli, yesterday, showed how existing tools were great for bits of biology, but not the whole.

When L-Systems were devised, we did not envisage the computational resources available in the future, but now we see that L-Systems fit in well, and challenges include extensions to new structures, including three-dimensional biological structures, and to larger systems expressing physiology and bio-mechanics. Separate models of each part of such systems exist, but it is not clear yet how to combine them. Yesterday's discussion of the division of systems into modules is relevant. Perhaps an elegant solution is to use aspect programming — instead of dividing by elements, divide by aspects such as bio-mechanics, transport of sugars etc. We need to work out how to program these independently and then put together at the end.

5.3 Panel, chair Andy Tyrrell

Panel: Jon Timmis, Martyn Amos, Wolfgang Banzhaf

The panel was run in the form of four short talks and then questions relating to the whole of the third session.

5.3.1 Andy Tyrrell: Bio-inspired architectures: hardware

[The ability of some natural systems to self-repair was illustrated with an animation of a gecko losing and re-growing a leg.]

It would be nice to have self-repair in hardware. We work on intrinsic evolvable hardware, where the evolution is all run in hardware rather than simulation runs being downloaded; the intrinsic evolution is on FPGAs, allowing for on-the-fly changes. The selection, mutation, and fitness evaluation use about 50% of the FPGA. A PC is involved in initialisation only. We can evolve simple digital circuits, image processing, and robot controllers, but all are pretty simple; what are the possible future effects? In digital computation, we cannot compete — consider Intel chip complexity. But analogue hardware is much less mature, and might be open to new ideas. In general, it is not clear that evolvable hardware is worthwhile, except when things go wrong. For example, in a continuously evolving fitness, if a fault is injected then the fitness plummets, but it then evolves back up to a normal level. This also applies if faults are injected during evolution.

We have developed a Cellular Array model, with Julian Miller. This comprises control and execution units in each element, and a chemical diffuser unit. We evolve the functionality of each unit, such that each array element is the same at the end of an evolution. We then grow it into the structure, and the chemical gradient diffuses to differentiate cells. One application is a two-bit multiplier. In evolution then development with chemicals, we can induce transient faults into states of system; with no external instruction, the results become strange but then recover, motivated by the chemical diffusion.

An EU project worked on evolutionary computation hardware, like an FPGA but with a “molecular” array and router unit. The routers had a distributed hard-coded algorithm for dynamic routing in real time; there was also reconfiguration control. The project developed a chip with an organic subsystem of 144 molecules, one router to 4 molecules.

On basis of this work, we can say that evolutionary hardware probably will not change the future, but it raises interesting questions, especially in analogue area (NASA). Continuous evolution is interesting — keeping evolution running to cope with environmental change and provide fault tolerance. We need a holistic view for real bio-inspired applications such as artificial immune systems.

5.3.2 Jon Timmis, Artificial Immune Systems (AIS)

Immune systems may also not be the solution to the problems. Traditionally, we have seen the immune system naively as the protector; it is this aspect that has been taken up as a basis for security systems. In reality, the immune system is a maintenance process that is continually interacting with systems, preparing, spotting erroneous states. The computational uses represent a typical computer scientist's extraction of bits from biology.

AISs look at immune system ideas and theory (Santa Fe work etc), identifying algorithms to apply to various problems: security, virus detection, data mining, optimisation, robotics — just like the other computer science approaches! However, the discipline development started in theoretical immunology research of the 1980s. This identified an interesting system with computation aspects, and saw the computer scientists and immunologists working together to produce grounded work on intrusion detection, immune nets and self assertion theory. From there, the ideas have trickled into many areas.

AIS is now far removed from immunological reality. This is also true of other bio-inspired paradigms. One way forward is to get out of this reasoning by metaphor and over-simplification.

Security — which here means intrusion detection — has developed techniques that look quite cool, but don't really work because they are based on negative selection, a hypothetical process to prevent T cells from reacting against self. These very abstract models are used as a basis for defining normal behaviour and detecting divergences. It does not work too well, does not scale, and there is a serious lack of theoretical work. We have now done some theoretical work that shows why these systems would not be successful.

Immune networks, clonal selection, clustering and learning algorithms are all plundered, but all the algorithms are very simplistic.

What should we do to develop these things? What is the embodiment of biology in context? The fact that an approach "kind-of" works may or may not be a problem. In reality, the immune system interacts with the hormonal and neural systems, so is there mileage in looking at these contexts? Should we go for true interdiscipliniarity, that is, two-way conversation not just stealing.

We need to think about why the immune system is an appropriate computational metaphor, think about the theory, think about the applications, rather than just benchmarking against genetic algorithm optimisations. Is there a killer application? How close to biology should we go? How much richness should we have? Should we look at cellular computation?

5.4 Martyn Amos: Cellular computing, microbes, molecules, other asynchronous hardware.

The work is done with Alan Gibbons and Dave Hodgson as collaborators.

Computer science piracy on biological notions — John just made the point that we should not try to unnaturally force biology into the Turing machine mentality; we should learn and take inspiration from biology in situ. Original work on bio-inspiration was on molecular systems. Micro-scale storage devices with bio-operations like sorting strands etc. As in Richard Powers' novel, *The Gold Bug Variations*, it was realised, with the biologists, that there are computational components as well as enzymes in biology. DNA is not just a minimal storage medium, as used in DNA computing. DNA carries meaning in its natural environment; we can harness this, as well as just its alternative-to-silicon aspect. Can we use natural systems instead of just modelling them, for human-defined computations?

Early work looked at DNA as the computational substrate, but now we see cells as a better substrate. The work addresses both this GC7 and the *in vivo-in silico* Grand Challenge, harnessing the intricate nano-scale internal machinery and extraordinary sensing, communications, and delivery capabilities of cells.

In nanotechnology, we want devices that are the size of a bacterium, that move and take energy from environment, that sense environment and talk to other devices. We have natural examples of these in bacteria and cells, but we need to harness them.

In cellular computation, one neuron on its own is no use; billions give interesting and useful behaviour. We can treat a cell as a black box — although there is a lot of good work on putting human-defined logics into bacteria DNA by genetic modification, here we just use collective behaviour to do computation.

Cells react to others and to chemical signals. For example, for slime mould, food scarcity causes panic signalling, spawning, and a long shift by a small part of the population. Berg observed salmonella in petri dishes forming regular patterns based on their sensing of other cells; this seems to function to allow colony to search environment efficiently; other patterns give protection by limiting the number of members exposed to toxins. Further patterns arise though small changes in chemicals in the medium. We can view these as emergent results of cellular responses to environmental queues.

Can we simulate the processes that give rise to these patterns, so that we can generate biological patterns? If we know a pattern (eg Olympic rings) can we then generate it, perhaps using simulations with GAs to manipulate parameters in forward runs? We might have *in silico* experiments to find models that work which can be abstracted for optimised bacterial heuristics — perhaps we could derive optimised ant models or whatever. Some research using individual cells has already been done (eg aircraft wing design), but we want the complexity that arises from numbers of bacteria.

How much of complexity comes from elements and how much from interaction with environment?

Can we model, can we use models to optimise biological systems, can we scale up from bacteria to ants to humans etc?

Bacterial models as clumps have been studied, as in Ben-Jacob's multi-agent co-operation systems based on resource availability and chemical communication. A three-dimensional model was proposed, exploring how a substrate structure such as agar affects bacterial pattern formation, including nutrient spectrum and distribution, response to toxins, and genetic components.

The pay-offs are potentially huge in terms of applications and insights — microscale scaffolding and deposition substrates for surfaces etc; possible new paradigm like GAs, ants etc; bio-complexity insights and abstractions.

5.4.1 Wolfgang Banzhaf: getting rid of the program counter.

Work with Chrisian Lasarczyk starts from the recognition that the classical computer is rigidly organised in space and time. Parts have specific locations and the program execution depends on these locations. The program and data are held in the same storage, and the program counter takes instructions in a defined order, so order essential.

Organic paradigms require a more realistic approach to space and time. Molecules and noise are not fixed. Molecules meet and interact by being moved around; it is hard to predict where and when they'll interact, and the sequences of instruction execution are not easy to determine.

Nature's co-ordination is by patterns, key-lock, and recognition leading to coordination.

In our view, we should see Brownian motion as a positive source of energy and creativity. We need to find new ways of doing co-ordination based on nature, to have adaptivity built in at lowest level. We need to accept errors and creativity and to have new ways to find functions. GP uses fitness functions; we could apply these to artificial chemistries, then abstract away nature's features and get arbitrary objects that can interact. These could then be observed, large systems of such artificial molecules interacting.

GP is the tool to produce functions based on desired fitness. To eradicate the program counter, run GP on artificial chemistry — multi-sets of instructions, and reaction as the execution of instructions. Such systems don't determine order, as in real molecular events. This could be implemented in RAM. Simple breeding of programs moves through linear instruction to artificial chemistry's random instruction access; comparison reveals emergent sequences and inheritance of frequencies, without the usual GP bloat. The output of artificial chemistry is related to the frequency of instructions applied (ie chemical concentration analogies); key-locks emerge, and parent programs that have random order crossover.

Evolution of artificial chemistry systems produces similar work to regulatory key systems — elaborate data flow paths evolve, connectivity not order is important.

Discussion points: Is the classical machine paradigm really necessary? In millions of processes, you can't do it. Synchronisation is a higher order phenomenon and should be approached as such. Is parallelism more fundamental than serial processing? Are approximate solutions with scalable resolution better for human systems? In artificial chemistry, concentrations increase results; scalability is easy; is hardware evolution the real goal?

5.5 Panel discussion

Geoffrey Canright: *Amongst all this self doubt, guilt about stealing from biology, work on an EU project stealing from biology, one concept is clearly the search for "nice properties" for engineered systems that they don't traditionally have. When we build steel structures, cracks don't heal, and there is significant expense in detecting and healing them. So, don't steal the mechanisms but do take properties.*

Andy Tyrrell: there was a certain planned negativity in the panel. Borrowing from biology is clearly important, but we do need to keep remembering that the stuff is borrowed, and to keep going back to the biologists.

Geoffrey Canright: Software systems are traditionally more like steel — fragile and unable to fix themselves. Unnecessary, intuitively will become thing of past if we focus on the desirable property of self healing.

Wolfgang Banzhaf: Can we get the features without binding to the mechanisms and materials? This discussion also happens in AI — forget the material implementation and go for intelligence at the level of symbolic representation. This is also not a successful approach, because the materials have important fundamental features. Natural systems are open systems, where as steel is sort-of closed. Cells continuously reconstruct, structures are dynamic. This is what makes self-repair possible, these are the low level things produce these high level properties.

Jan Kim: from a biological background, it also appears the main motivation is that biological systems have nice properties, and we want to transfer the mechanisms in the naive expectation of also importing some of the nice properties by accident. The approach is not guaranteed. The nice properties are not really understood by any group, but one basic notion of their underlying concept is that a system would have a model within itself of what it should be. Could we teach a bridge to know that it should not have a crack and to repair itself? Biological system self-organise in some ways. If we do a meta-shift, as the panelists pointed out, importing aims can cause confusion. This is because things on the formal level can turn out to be very much the same — a neural network is simply a linear classifier. Other fields find it hard to get in to biology simply through metaphors.

Marian Gheorghe: We need two-way inspiration; we need to consider deeply. There is, perhaps, more progress being made in systems biology (via differential equations etc), but now computing is starting to develop. Luca Cardelli's work, L-Systems, biological swarm intelligence experiments. However, if we look at ant maze navigation optimisation we can see the mistakes. The computational model is wrong because it forces "ants" to do turns at unnatural angles [90^o, rather than 60^o]. For ant navigation, we need a reversible algorithms — nest-food; food-nest — so we need to look at the biology to guide sensible algorithms.

Martyn Amos: the pay-backs are not just one way; not pillaging biology. The pay-backs to biology come from work such as that of Grzegorz Rozenberg on applying standard computational models to cilia genome decryption — this treats the task like a extraction of a file from a tarred zip file, and unpacked the working genome. The simple computational model is sufficiently complete to account for all the observed real biological decodings, and represented a positive contribution to the biologists, who could not work out how the process worked. They now have a possible explanatory mechanism. Also, Luca Cardelli's talk shows some of the things that we're contributing to biology.

Ottoline Leyser: from a biologist's perspective, it also feel that it is a one-way steal, but from computer science. However, the present situation seems to be at a point where genuine symbiosis is happening among the disciplines... complicated biological data sets require computational input; computer science is interested in the data sets for other reasons. A problem is the concern of mistaken impression of computer scientists that biological systems should be optimal — they are not and never could be. Biologists want to understand their non-optimality, but I'm not sure if it is sensible to try to import the non-optimality into computing and engineering...

Wolfgang Banzhaf: engineers don't take biology raw.

Rob E Smith: any problem that can be optimised is uninteresting. Stealing is not the problem; the problem is believing our own bullshit. We have to sell proposals; the marketing veneer is currently "bio-inspired". The popular science press [libellous statements about popular science magazines] guides funders. However, we want real models, principles and theories, and real feedback, not dependency on metaphors. I don't hack cool robots with no useful generalisations. This is a funding-inspired problem.

Jon Timmis: yes: there is an AIS EPSRC network with various subgroups; only one is looking at generating good design principles for AIS, rooted in a mathematical framework. We need mathematicians as well in the interdisciplinary pool, and engineers etc...

Colin Johnson: on the space notion from Wolfgang Banzhaf's talk: we have ignored physical space in conventional computing, focusing on logical space and logical organisation of information. We could look at physical models of information organisation, as in cells.

Martyn Amos: yes. There is some work on social memory in two-dimensional space — if you line up bacteria, you can't get anything useful, but if they can talk to each other in two dimensions, then you do get something.

Przemyslaw Prusinkiewicz: as soon as we have a space-related problem, conventional computing breaks down. For example, it is hard to model a piece of paper on a table with differential equations; nature has no problem with this, but simulation is very hard. Computer graphics has made some recent breakthroughs. Note that computing space has properties, but empty space has interesting properties too. Real space properties are hard to reproduce in computing devices, when space is changing as in a growing leaf for instance, or when trying to perform computing in non-equivalent space. These are very interesting things.

Aaron Sloman: this point is lost in AI, and especially in vision work. These areas have lost sight of the fact that vision is about spatial structure and EMPTY space, how we see empty space. Instead of just looking at object recognition and general-purpose recognition, vision should look for structure as well. Affordance — what you see is not just the geometry, but what it is possible or impossible for the viewer to do in the observed space.

Przemyslaw Prusinkiewicz: we can do very different things with space than computers. For example, perspective is easy in a computer but hard for humans (art), but once understood people can do it by hand. However, you don't have to teach people how not to draw things that are hidden, whereas it is hard to get a computer to miss out hidden objects from a representation. mutual problems!

Andy Adamatzky: a huge amount of brain evolution is to do with vision processing.

John A.Clark: observe that we are here discussing slime mould, not the other way around. We might ask what is role of human agency? We need to learn and adapt — the knock-on effect of a tree falling in Switzerland took out power in three countries.

Klaus Peter Zauner: we currently try to make very formal models; we need to add in physics to program these systems, like flight simulators that model flight parameters. We can't get all the complexity in, so we will still need human guidance.

6 Session 4: Engineering the future

[Susan Stepney introduced the invited speaker] Christopher Alexander is an architect with a large influence on software engineering: his *Pattern Languages* book on architecture (of buildings) — which is also deliciously socially subversive — led to the 1995 "Gang of Four" book on *Design Patterns*. The patterns work disproved the idea that the only object-oriented concept that we need is class; we also need higher level structures. However, to date we have only a few pattern vocabularies; we have not got to languages. More recently, Christopher has published four volumes on the *Nature of Order*.

6.1 Invited Speaker: Christopher Alexander, Harmony seeking computations

I have put a lot of thought in to what architects have done wrong, in messing up the earth over the last 50 years — how to do it better, and what I can contribute that's useful. I have spent 25 years experimenting with a form of computation, and my intent is to describe where I've been and what I've been trying to solve.

My computation is different to computation in general. It all takes place in space, in it and transforming it. The computations have a decisive moral component — they can make things well or badly. To strive after making it well is paramount. Even biology is less morally relevant.

Nature of Order describes experiments and underlying theory, practice and results; it has taken 27 years to write!

The key idea is that of *structure preserving or enhancing transformations.* This is coupled with wholeness, as a structural feature of a configuration. Wholeness is vital and very hard to describe, but real. I want to show you that the wholeness is a real feature. All transformations attempt to take one wholeness forward to another wholeness, growing naturally out of one another.

There have been many comments so far of the form, "how can we describe X then get an intelligent swarm to build it". This is ridiculous; we can not arbitrarily pick a goal and then figure out how to get there. [Libellous statements about 20th century architects in general.] We need a non-obnoxious way to introduce goodness into computation; we should always have a positive effect on people and the planet, using computations that are not neutral.

Harvard Centre for Cognition Studies work looks at the wholeness of objective structure. An experiment carried out on students takes 35 strips of black and white areas, laid out on a plain grey background. The students are asked to arrange these by similarity. There are two types of response. 85% did a sort of digital-read classification; the other 15% grouped the strips by some perceived morphological similarity. The second sorting is less obvious and appeared to be more interesting, so I tried to "train" people so that they would be more likely to do the second sort of classification. I tried random creative play, which had no effect, memory and cognitive task experiments etc. The only activity that had an effect was to show one strip, then to closely-pack the scrambled array of all the strips and flash the result on a screen for half a second. Then I offered a nickel anyone who could matched the shown strip. After a period playing this, the students who produced the second form of classification increased to about 50% of the group. This illustrates that the subject of wholeness is removed in some sense from our civilisation — these students are of high intelligence, and good analytical powers. I also did this experiment with mentally-retarded people and with children, both of whom displayed more of the second form of classification.

One reason why cities are a mess is that normal thinking is of a building as an object, whereas in all great and humane architecture, the dominant thing in the space; the architecture creates, harmonises, and makes space beautiful; the building is seen as a servant to space; we need to understand wholeness.

Structure-preserving transformation occur in nature. Images show, by a series of obvious moves over time, a mouse foot forming over three days. Similarly, a willow-tree bench is created, starting with observation of the trunk of tree — the latent structure is in the ring at bole of the tree, which is amplified by adding short stakes around the tree, then weaving to form the seat structure, and finally putting moss on top. The structure is preserved throughout.

The historical growth of Amsterdam over several hundred years, as revealed in old maps, is centred on the Amstel River, with successive walls and canals added in a clear organic growth.

Contrast two images: hayricks in a field in Romania, and wind turbines in a field in Denmark. As two structures in a field, the hayricks have wholeness of structure. They are kind to the view of the land, they enhance it. Their presence is non-destructive but bring a new character to field. The wind turbines are destructive, because they break the patterns in the landscape, and have no relation to the existing structure; they cut across and do not grow out harmoniously.

One experience of wholeness concerns work on acetabularia (a green algae). It has been hard to pin down the pertinent characteristics; Brian Goodwin has done important work on it. As the algae develops, the stalk differentiates, and a whorl forms. Goodwin observed this and recorded it in a series of diagrams; these show a hump moving to an inverted umbrella, but there is one flat-topped phase in the middle. When I saw the diagrams, I questioned whether this could be how it happened. Goodwin was working on differential equations of calcium concentration etc; I noted that one of the (diagram) transformations was not a natural structure-preserving transformation; you can see this with no biological background; and I proposed an alternative based on a structure preserving transformation that I had observed: outside Oxford there is a pre-historic mound with a ring ditch; this is a more natural model for the change from a dome to a whorl than Goodwin's flattening out at the top. Goodwin looked again and agreed that this was what happened. When I was preparing this paper, I decided that I should check the structure inside the whorl, and magnified photographs on the Web reveal that there is indeed the bump preserved in the middle, as inside the ring-ditch — structure-preserving transformations can help, even without academic insight in the domain.

Consider a certain Tokyo apartment building in Y between streets. The initial plan just sat the building on the site; this was revised to take account of the configuration of the streets, giving congruity to an otherwise-uninteresting building.

Students do experiments on structure preservation. We generally find that people agree on what does and does not preserve structure. For instance, if we ask for a sketch of wholeness but give no guidelines, everyone draws something different, but they all then agree on what does and does not preserve that structure of wholeness.

Starting from a line of dots, I get students to draw structure preserving transformations. These can arrive at "beautiful thing". If we follow this process in architecture, we get beautiful towns etc; if ugliness arises, it is not that process. In the paper, I give an account of the construction of Upham House — every detail is structure preserving.

Consider St Mark's Square in Venice through its entire 1100-year history. We can see latency — in the wholeness there is something that the transformation can operate on. Latency not strongly embodied — the Acetabularia has the latent possibility of forming a ring from its dome. In St Mark's Square, because of the arrangement of structures early on, two latent centres emerged. The latent centre was embodied by construction of a building, a church, between them. This changed the latency and started a new cycle on a new latent structure; buildings make the wholeness concrete (not on the latent centre, but focusing it). In third cycle, the Campanile was built, focusing another latency. Existing buildings, entrances and routes influence where the latent centres are. In Venice, the result is a famously lovely place. All the building follows structure-preserving moves. Twentieth-century architecture could not or would not exploit this sort of long-term consistency leading to beauty.

So latent centres are part of wholeness and transformation tries to make the latent structure stronger; coherence is achieved without a master plan. Paying attention to latent centres and wholeness will produce a coherent structure — this is not same as emergence.

Look at natural phenomena to check the relevance of this approach. When people do it, it is a computation; when nature does it it is not an intentional computation as such. I want to have the approach well-defined so that people can "just do it", perhaps using mathematically-assisted intuition. We want a realisable computation, on any sort of computational substrate, so that people can do it better. Perhaps it cannot be embodied on a digital processor, but I think it can be described if one is very smart. I have published enough to share in the task, but we are still at the start. We need to find how to calculate and run structure preserving transformations.

I looked at Reynolds' boids and the characteristic V formation. The flocking literature is ingenious; there are three rules that turn random birds into a coherent flock that stays together, with temporary disturbances. But emergence doesn't fit my mind-set, even though I gave early lecture on the growth of order from small acts... in traditional societies, beautiful structure came from small events, an early view, architect's analogue to emergence. So, where does the V actually come from? Simulations don't create the V; they give the idea but not the whole structure. Behind a bird there are vortices; this means that there is an area with no pull, which is not a comfortable place to fly; so the physics of vortices says that the V arise to stay out of these vortices. Even a single line of birds will be diagonal for this reason. This does a lot to the V production.

The V relates to positive space — an artist looking at the V sees that V birds form beautiful space and straight line ones do not. Can a mathematician see this?

[Andy Wuensche: *is it not a field of vision effect as well; in the V, all birds can see ahead and can see the bird ahead; it is comfortable space?* Yes.]

Trying to make space positive is one of the 15 structure-preserving transformations that I have identified. Take clouds. Fluffy drawings aren't positive space; sky itself almost always has blue stuff as positive space. There are explanations to do with vortices and convection physics.

The Somerville housing scheme, for 200 houses on a 5-acre site (Massachusetts) was a situation where high-density had to be achieved. The site plan plots all the connections, and latent structures emerge. To achieve high density without being obnoxious, we devised a configuration focused on gardens edged by buildings.

More examples showing how the reality is created according to the theory.

6.2 Questions

Julian Miller (and others): please design our next campus...

Seth Bullock: *only one photograph had people in it, though a lot had echoes, such as access points.*

Structures are all achieved with the people who live and work there. This is crucial. The photos of Eishin Campus in Tokyo show how people move within the space. Planning had people moving around in the space.

Andy Wuensche: *The Denmark towers — is there a way to build wind turbines that does preserve wholeness?*

This is a serious problem. I enraged the Schumacher Society over this example. It is a novel experiment that is failing from the perspective of preserving the land. The experiment is important, but there would be a better way to do it. In principle it can be solved.

Leo Caves: *St Mark's Square — you were describing the development of static forms, with the implicit dynamics of people flowing through the space and in and out of the buildings. Could you capture positive space by mapping fluxes of people?*

Yes. To use a drawing as the basis of a building design is nonsense; the space changes at every step, and you have to be in control, or costs escalate etc. Design is a dynamic process, as in other talks at workshop. I'm hoping to help in the project to rebuild Soweto, working with the people who live there.

John A. Clark: *Do you know of any cases where a structure-breaking transformation has been successful?*

Are you a post modernist? There are are lots of ways of doing it, but it's like planting a daffodil and making it do something funny when the shoots appear.

John A. Clark: *Mondrian paintings are constructed on balance; what he tried to do was important. However, some have a different aim, and that hits you in face — it could jar but often does not. Is there architecture that breaks the rules but works?*

No, but takes a long time to achieve balance.

6.3 Panel, chair Susan Stepney

Panel: Tom Addis, Rob E Smith, Andy Wuensche

The panel was run in the form of four short talks and then questions relating to the whole of the third session.

6.3.1 Tom Addis: Socially sensitive computing: a non classical way forward.

Our group's work does not take inspiration from biology or quantum mechanics, despite backgrounds in these areas. Instead, it is motivated by a terminal frustration from a period doing AI when it was still known as cybernetics. I am puzzled that, after 60 years and very fast machines, we still can not address basic functions of the human brain, a thing that looks like porridge and works slowly. And we use the excuse of "if only we had more power".

Because we're not making progress, I return to early ideas. Wittgenstein's *Tractatus* is a theory of language, sought in same way as principles of mechanics, mathematics etc. After he wrote it, he became a gardener, thinking that he had solved it all. Looking at plants, he realised that he'd made a mistake, and the *Philosophical Investigations* were created. I have taken the ideas and applied them to computing.

Denotational semantics is the basis of all formal languages used to define all the machines. It concerns objects with peculiar properties — independence, allowing free combination; atomicity, in all possible worlds; immateriality, indestructibility and self-governance. These are not the generic objects of object-oriented programming; few things really have these characteristics.

Wittgenstein worked on family resemblance, taking a philosophical investigation ("irrational") from the tractatus ("rational") start. The "rational" has finite rules that unambiguously include all members of a set, and exclude all non members. For the "irrational", no such finite set of rules can be constructed.

Everything is not potentially describable unambiguously. Sets depend on dynamic rules of everyday use. Wittgenstein looked at chairs and games etc. Breaking down the chair specification (to be sat on, stands on own, has 4 legs and back, sitter's feet touch floor), what if each element is not met? Is it still a chair? Yes. We could successively take out all the criteria from the specification and still have a chair — it is not solely a rational definition.

Software programs are never finished, because of irrational sets. These have functional properties and accidental properties (such as naming). The world is not detectable unless it can be distinguished, so there's another semantics, that of the world for which the program is written. Feedback is needed from the problem domain; that is socially sensitive.

Dynamic ontology can define sets that are moving. We want a rational machine to have a rational view of a dynamic system, so that we can make inferences by deduction.

If a computer is also uncertain, as in quantum and bio-inspired paradigm, the two informal semantics can survive by walking step by step together. It will be possible to argue with a computer! We want machine-independent programs.

In real life, you can win any argument by using a logic set up to achieve one goal alone. It is not just a question of logic.

6.3.2 Susan Stepney: Engineering emergence.

We can (ultimately) position molecules to make artefacts. We want a universal assembler that can make anything from steak to space ships. Such a universal assembler needs resources and instructions — what are the right assembly instructions? It is not enough to simply be able to build the nanite machines; there are at least two stages, namely a bootstrap phase that will be slow because of the need to grow assemblers exponentially to do something interesting, then self-assembly or construction from raw materials.

The problem is well-known — "grey goo" disassembly by replicating nanites, known scientifically as "global ecophagy". These systems are safety critical.

There is an engineering challenge, because the desired product is the emergent property of the actions of many nanites. We want to reverse-engineer the emergence, from steak to nanites. This is a long way off. What do we need to be able to design these sorts of systems?

We are dealing with open systems; there are constant flows of energy and resources through the system, and structures exist at multiple scales. The systems form stable structures that persist over time — stable not static. These patterns in space and time are the emergent properties.

There are bullshit definitions of emergence, but the real definition captures the difference between the parts of the system and sum of parts. The systems are not simple aggregations of properties of their parts. A key recognition point is when you need a new language or new concepts, as in describing gliders that emerge on CAs — CA cells are static with no movement; the emergent property is the patterns seen over space and time.

In these systems, the phenotype emerges from the structure and dynamics of growth rules etc. Computation emerges from a trajectory through a computational space. There is continual dynamic computation, interacting with the environment, and governed by the structure of the phase space and attractors. The key is to think of computation in terms of these. Phase spaces that are dynamic are particularly interesting, having changing parameters etc. Can we say things about the underlying dynamic phase spaces? Hierarchies of emergence become natural in this paradigm; each level has its own length and time scales, its own phase spaces and trajectories. Perhaps the required emergent property is the name of the trajectory.

We can't expect precision in nanite construction. The hierarchical growth of ever-larger structures would be like growing organisms. Most work only looks at one level; we need lots. We need to predefine emergent phase spaces, which will be hard.

If we look at how we might engineer emergent properties of embodied computational agent systems, we find that correctness proofs and conventional refinement etc not applicable. We need non-classical techniques, involving probabilistic and soft reasoning. Patterns of structure and dynamics will be essential.

6.3.3 Rob E Smith: Quantitative models for this stuff

In evolutionary computation, we note that there is misunderstanding of what quantitative means. Some models are crap but useful. Much of what computer science and mathematics deals with is interactions among clean devices, computer interactions. Things such as gears, wings and headlights also exist, and are assumed to have different interactions; people also get into the systems.

Machines may have built in safety factors. However, subjectivity and irrationality are important when interacting with people. For this need to get into engineering and philosophy.

Consider the bicycle. Bicycles are crap — they are not stable in the standard operating position. However, this instability is the key to their success. This is also true of aeroplanes — the Wright Brothers succeeded because they abandoned idea of a stable aerodynamic structure, putting in a person as an active controller. We need this design principle.

Looking at safety factors in engineered machines, the fundamental engineering theories are nonsensical. For instance, an engineer's model of bending assumes that plane sections remain plane. Some plane sections move a little, some move a lot. This violates conservation of momentum, matter etc; but it is a model that is the foundation for a huge body of work.

The GA schema theorem is true in the engineering sense even though proven false. The proof observes that a finite population can't grow exponentially etc. The schema theorem is the basis for a lot of insight, even though not precisely right; it is a useful source of engineering guidance.

Turning to the Logistic theorem, Crutchfield analysed logistic equation sequences, devising complexity measures such as the entropy of a sequence. Some have a linear relation, some have a bounded relation, but at the boundary, the Feigenbaum number, complexity increases without bound. Kauffman plus Crutchfield consider increasing randomness and measuring inference; there is a monotonic increase except at Kauffman's edge of chaos. If you put more randomness in, you get less information out. These are dimensionless numbers from a wide range of systems; the system independence makes this a useful model.

A journey need not be a random walk. Maps can be useful.

6.3.4 Andy Wuensche: CAs

For an interest in emergence of complex patterns, CAs are a nice system because they are so simple. There is the underlying physics, interactions at another level, and various levels of description that are a nice way to describe complex systems.

In one dimension, you can detect patterns; in two dimensions, there is the Game of Life (GoL), which are of more interest because there is more scope for interacting. It is surprising that GoL is the only really interesting two-dimensional, two-state rule set; to derive general principles, we need a wide variety of examples.

Using three colours (black, white, red), we can search for kinds of CA where the kind of next state depends on the proportions of colours in each neighbourhood, illustrated with a three-dimensional glider gun shooting in four directions. If we add random look-up rules, then we get a mess, as usual.

Two-dimensional CA rules and patterns are easier to see. For instance, with six neighbours on a hexagonal grid, the same rule as the three-dimensional glider and the same initial state also produces glider guns and collisions (dependent on initial state).

Chance is the commonest way to find rules. Systematic discovery depends on an entropy measure, perhaps as in Rob's talk — block entropy might consider how often each rule is used; if the distribution of rule use it is even, then there is high entropy; if the distribution is skewed, then entropy is low. If the variance of entropy is low on either, then the CA stabilises in that entropy level. Complex interacting structures have the signature of varying entropy because when particles emerge, it skews the look-up frequency; when particles collide, we get a period of chaos and higher entropy. We could refine the entropy characterisation into a GA — throw in all the rules and measure the entropy (or its standard deviation). Most rules are located on a high mean; rules with a lower mean are ordered rules that reach an attractor and become boring. The rest are potentially rich in interesting behaviour.

In the two-dimensional hexagonal grid, other rules produce honey-combs, which have been discovered independently. Mutations of the rules using bit-flips also produce interesting patterns usually. For example, Andy Adamatzky found a six-glider gun by chance.

Basins of attraction in discrete dynamical system are large but finite. They are deterministic, so we must find a repeat state on an attractor; there may have several.

A random boolean network is a generalised CA, and can be considered similarly. We can compute precise structures of attractors without running the system to its end. There are lots of visualisations of the same model (from DDL). Basin of attraction can be characterised by the number of nodes etc.

Small changes to the architecture of the network do not make much difference unless the change hits something sensitive in the connectivity or the logic in rules. We observe that mutations to the logic make less difference than moving connections.

The DDL website also covers particle-based computation (with Andy Adamatzky), self organisation etc.

There is a link between the dynamics seen in space-time patterns and attractors, trees focused on the attractor. The more ordered the system, the more bushy the trees, ie the more prior states for each state. In chaotic dynamics, there is sparse branching with many unreachable states.

Consider content-addressable memory. A finite network in discrete synchronised update will have a basin of attraction field. The initial state will fall to one known basin via a tree; every tree root is a sub-category. This is the basis for content addressable memory — given a categorisation that you would like, can you find a subtree for it.

6.4 Panel discussion

Jan Kim: In the context of engineering without conservation of mass, there are always minor deviation from reality. The earth is not a sphere.

Rob E. Smith: The point is that models should either provide insight or feedback. We need more purposeful models.

James Neill: What is the relationship between irrational sets and emergence?

Tom Addis: emergence in cybernetics has a long history; you could put any junk together and get interesting behaviour. We are struggling with determining where one is simply looking for images in the fire and seeing what one wants to see, and where there is something significant being done. Principle-based engineering can be used to dictate design not by algorithmic processes but by conformance to required principles that generate behaviour. For example, bio-systems inspired work such as someone's tortoise; observe the principles and look for these in a system. Others such as Chomsky have looked at principle-based language (in place of Chomsky grammars). Irrational sets were not discovering anything new; the problem is widely known; we simply name a concept that ties together a wide range of related problems. There are entropic elements, but these are not same.

Rob E. Smith: The two axes are the entropy relative to system structure, and inference from a picture of the same measure. Is what we call emergence a psychosocial phenomenon? Is the inference aspect there because emergence and inference are both about perception?

Christopher Alexander: there is a lot of sloppy language about emergence. There are meaningless but interesting observed patterns. In the book, I have something interesting that happens, the simplest emergent form is elements and aggregates. In the interesting cases there is a third level, the bigger thing [environment]. This then helps a yet larger thing that is itself creating a context for the lower levels. Each has different behaviour.

Andy Wuensche: [lecture on origins of life in universal context] we only have one example of life. With different rules, we would not get life. In CAs, we can try any artificial universe; mostly we produce a mess but sometimes we produce interesting structures and interactions. We don't know the underlying physics of that universe but we can make observations and write rules that govern the perceived behaviour. If this supports a high-level description that may unfold without limit then we get interactions among particles to compounds to unbounded complexity. Artificial life is interested in this.

Aaron Sloman: Susan Stepney mentioned three aspects of emergence, which are all needed if it is not just shallow: pattern (dots to columns etc); extension of ontology (can't define result in the concepts of the original, such as a chess game on computer); but the third is key — dynamics. Causal relationships are essential, that are expressible in the extra ontology, that you can't translate into or derive from lowest level physics. Counterfactual condition statements hold — if this structure weren't thus, other things would happen. This defines important emergence. Tom's things would come out of any complex reality that allows new concepts. It is poor philosophy to require all the context to be known first, because as contexts are created, they'll have Tom's sorts of contexts — functional, aesthetic etc.

Susan Stepney: impoverished contexts are used because we can't use the reaction of things with everything out there; instead, for example, we are evolving robots for impoverished mazes. The task is hard (even for us) because there is not enough contextual information. The boundary of ourselves is hard to define; for instance, I now consider my brain to be extended by Google. We need to take in a lot more than we do.

Geoffrey Canright: A lot of nonsense has been said with some has real bones in it. A sceptic would reject the nanite story; it has no facts — "read this and all this will emerge". Can we back this up? Patterns make me nervous; they are either trivially mathematical or based on human psychology. This is wrong and subjective. Patterns are what we notice.

Rob E. Smith: ultimately the endeavour is about creating artefacts

Geoffrey Canright: we call things patterns where we find them, but there are lots of others.

Tom Addis: *purpose* is missing from the argument. A pattern is a pattern because it has use, this allows us to discard arbitrary proposals.

Christopher Alexander: this is close to Mendeleyev who observed a pattern that eventually led to other elements being discovered.

Tom Addis: the purpose was understanding; it is not just a mental thing.

Viv Kendon: what makes you think that it is easier to make a steak than a spaceship? The latter is a more precise functional definition; steak is harder to characterise but is constructed by cows.

Susan Stepney: cows do it by nanite assembly, so we know that it would be possible. For the spaceship, we know that nanites can build people, and that people can build spaceships, but not how to get nanites to build spaceships more directly.

Viv Kendon: the level of definition of an object that we are trying to construct is important.

Susan Stepney: we need to take into account the route as well as the goal.

Jan Kim: taking it further, we have a proof of existence of what's wrong with nanites — the earth. A spaceship going somewhere interesting at will might be like the earth around sun. [This point was not clear to the note taker]. Subjectivity is not a wrong point of departure, but must quickly get to objective and formalised description of the goal.

Susan Stepney: I have a subjective desire for a spaceship.

Rob E. Smith: spacecraft design deals with artefacts without precision; we make it usable through safety factors and over-design. Applications such as Google deal with humans; they are revolutionising thinking because the common man now sees that computers are not just about predictability, accuracy and speed. The computer now works with us; subjectivity is a real practical thing now.

7 Session 5: The way forward, chair, Stephen Emmott

Panel: Seth Bullock, Aaron Sloman, Susan Stepney, Andy Tyrrell, Andy Adamatzky, Cristian Calude

First, the new panelists made statements of their positions.

7.1 Aaron Sloman: Altrical self-organising information processing systems

I am working with Jackie Chappell, the researcher responsible for Betty the hook-making crow, which worked out the properties of wire and how to make a hook. The results have now been repeated for a new crow with absolutely no possible training. Betty does it in many different ways.

In addition to physical growth, bio-organisms grow their own information processing architectures, some of which are virtual machines. How does it happen? How much is genetic, from the environment, individual or cultural?

People pick up simple ideas and run with them as if they were the whole answer, and assume that if design is too hard then emergence will do it.

Structure and structural change pervade biology at all levels. External behaviours are also structured, social structures. Evolutionary change produces structures — virtual, physical, chemical. Most are highly parallel. We need a different way to think about it. Hardware bugs may require dealing with or working round. There are many components, sensing, acting, information processing. Adaptation and discontinuous changes are necessary. As we listen, we build a semantic representation of a new and unique sentence. We need to understand systems that can take in and interpret complex structures. We don't yet know how.

A lot of this is very precise structure. There is also a lot of noise. We need precision. People don't come out with the wrong number of limbs etc (!). There are levels of virtual machines and emergence.

We can observe many physical etc processes, but it is hard to observe virtual machines. We need deep speculative theory construction, tested against hard constraints etc. Many people don't understand or don't notice even the questions to be asked. For example, the vision people summarise image characteristics etc, but not the optical flow calculations needed to miss the walls when walking.

Seeing affordances is about not just seeing what's there, but what isn't or what might not be. Betty is not using modal logic, but how else so we characterise what she does. Can we learn from babies? Can we replicate this in robots? — No!

We need to find out about natural information processing. Observation is that some animals are precocial with rich cognitive abilities from birth, those that stand or feed from birth etc. Others are helpless at birth, for instance crows, eagles and humans. How intelligent and varied the skills of adult life are, how brain grows as skills develop is different in these groups. The precocious animals don't learn so much. There must be hidden skills relating to architecture etc building.

Josh, an 11-month child was filmed and analysed. Here is a baby experimenting with a whole series of yoghurt-related activities.

Humans etc have a complex information processing architecture implemented in all sorts of things, chemistry as well as neural activity. They have many different kinds of virtual machine — reactive, perhaps with some adaptation ability; a deliberative virtual machine; representing what does not exist. Betty had the latter but baby Josh appears not to. Betty can create new representation and use compositional semantics to extend and manipulate representations. A rare virtual machine is the ability to represent processes that are themselves processes with semantic content, and to reflect and capitalise on them. Baby Josh has ideas about experiments to do for this; Betty could already do this as well.

These processes together give a possible cognitive architecture with motor and perceptive elements.

Altricial beings have architectures that grow themselves. This should be relevant to robots, operating systems and HCI etc. The primitives might arrive in 20 years. It is not a valid argument to say that this cannot be run on current hardware.

7.2 Seth Bullock

"Nicheversal" research content suffers from political organisational attitudes in the face of hard problems. Bio-inspired computing's heart has deep problems of complexity etc and these are not new problems; communities have been struggling with it at least since the early cybernetics pioneers. Why should we believe that we can make progress?

Fast computers are not useful. Industry and government want something to happen. There are real problems that governments believe bio-inspired computing can address. We need to take the opportunities.

Communities were disparate, with separate work in sociology, biology, etc. Now the people who are looking at amorphous computing etc are next door. EPSRC funds this work, and interdiscipliniarity is encouraged.

So how can we realise a way forward? Perspective are long. We need to think where the careers of current PhD students will go, and what their students will do. There are exciting subfields, but the progenitors in most of these are no longer impressed with progress. How do we keep addressing the cutting edge, not just, for example, making GAs more efficient?

Computer science is young; it does not find it natural to look at historical precedents, but it must, via PhD students. We must get the thinking right; don't just work from existing research, and take opportunities to collaborate. Go and talk to the biologists. It is our responsibility to train the next generation and make the opportunities to work in an established, integrated community.

We need to help the EPSRC, and especially the BBSRC, which is less receptive to the new ideas. We can use future electoral college elections. Panels need to have reasonable people on them, because we put a lot of work into grant proposals; we need to work on giving them a chance. We also need to work on mechanisms that encourage the BBSRC to support this work too.

7.3 Provocative statements by panel chairs

Stephen Emmott asked the panel chairs for provocative statement, one that can point the way or identify the way points: how do we resource the future to address real problems and make real progress?

Susan Stepney: If you can't say if in maths, then you do not understand it. There have been lots calls for quantification and maths. This is necessary, but do we have the right sort of maths? Perhaps we should co-construct the new mathematical methods with the new models as we build them.

Andy Adamatzky: we need more — more problems, more experimental scientists in computing, more experiments to get real devices, more substrates. Don't be afraid of anything that can compute. More purity is needed. Artificial life is now out of fashion, just a junk yard for second rate papers. We must keep it "cool".

Andy Tyrrell: a lot is good and potentially cool but we need better mathematical models of biology to understand and exploit, and remember that exploitation is not necessarily digital.

Cristian Calude: we're on a journey with no destination. There are dangers in overstating the current tools. We should not be afraid of limitations caused by inconsistency and stability; there is a maths of instability.

7.4 Final panel discussion

Kester Clegg: we want to see a move towards people producing generalisations rather than more tiny problems with no application beyond

Tom Addis: to do that, we need a science structure — hypotheses worked on by groups. Asimov's comment is that "that's funny" is more important than "Eureka" in real science. Engineering and computer science people are not scientifically trained and don't know what an experiment means in a computational paradigm.

Stephen Emmott and Susan Stepney: we need hypothesis-driven research.

Seth Bullock: we must publish more negative results with explanation, not just reported improvements.

Susan Stepney: we should agree that if we are asked to referee a paper that just tweaks an existing model, we should reject it.

John A. Clark: Perhaps the web site could include a position paper on how to write a paper worth reading. Such a paper needs to explain why the work is as it is, and who can make use of what you do. We could have principles or a checklist of what a paper must do. A lot of evolutionary papers are just program-parameter tweaking. We need ties into scientific experiment, statistical experiment, exploratory and confirmatory statistics etc.

Susan Stepney: we are tasking people to put positions forward for a journal paper; it would seem that John just volunteered one.

Mathieu Capcarrere: a page limit of ten pages is not enough to represent experiments in ways that are repeatable. We need a repository of software that is available for the repetition of results etc. In write-ups, the details are often missing, so the experiments cannot be repeated.

Seth Bullock: there is a double-edge sword. We don't replicate results, but we do not just want your software run on someone else's machine.

Aaron Sloman: most people don't package workable software in a usable way: I read your idea and write my model; if I get same results I've replicated yours. You can always email the author.

Jan Kim: complex systems often have seemingly innocent details that make a lot of difference. If we don't have a full theory in these areas, we have to give full details in case we've overlooked something important.

Christopher Alexander: little has been said on statements that might turn out to be false. You have to make statements that could be false, and you should indicate how they could be proved false; if the statement is not shown to be false, then you have made progress. Everyone could write one paragraph making an assertion that could be proven false and saying how it could be done.

Leo Caves: students in science, especially physics, are not taught to formulate and test hypotheses. It is not just writing, but training. Students need to know how to get ideas over properly. In the previous session, discussion was moving towards a philosophy of causality, formation of relationships etc. This is a spanner in much scientific method. A lot of re-creation hoovers-up but does not solve, and then recedes. We can all chose something to work on and survive in good funding times, but are we doing the right things at a deeper level or higher level?

Aaron Sloman: what is wrong with complaining about the subjectivity of what we are say, is that if something could be proved wrong it is not subjective. NB Newton's particles versus Young's waves — they prove that Newton was wrong, but then the particle theory was rehabilitated. Popper would lead us to the situation where we can decide only between those that are progressive and those that are degenerative. The former change things, the latter just add details.

Tom Addis: research on non-progressive topics shows that it gets ever more complicated as it fills in gaps.

Stephen Emmott: we can make progress on a way forward by teaching the history and philosophy of science.

Tom Addis and Aaron Sloman: philosophy is being squeezed out in all subjects.

Jeff Johnson: in complexity and design, there is an EU ONS initiative. There is an open network, which is new. We can benefit from it and contribute to it. At the moment, it needs computer science input. European support is driven by a belief in the importance of these areas to Europe. I am running part of it. There is an educational element, and an element to do with writing and communicating. They are setting up a European PhD programme in complex systems science. There are lots of EU-funded projects. There is also a new free Complex Systems Society with a conference in Paris in November. We can also influence design. Someone has a road map.

Christopher Alexander: design is the science of systems not as they are but as they ought to be. There is an initiative in twenty-first century design by the AHRB, embracing complexity in design, with biologically-inspired design clusters. The biological people don't know a lot about design. A way forward is to engage and connect with complex systems and design communities.

Jon Timmis: How does the panel see the community? What is it, what are its links? There is huge diversity here. What will we do that is concrete from this? Subgroups, clusters or whatever?

Susan Stepney: it is not yet at community; it is an aggregation. The purpose is to make new connections. We are also interested to know if meetings like this are worthwhile — should we do more of this?

Martyn Amos: We need the UK version of Santa Fe; it's been mooted many times over the last five years by the EPSRC and others. Can it be taken forward? Is there a desire for this?

Susan Stepney: York is putting together YCCSA [York Centre for Complex Systems Analysis], as a virtual organisation; it will be in a new building on the new campus. There are already lots of people involved from different departments, a mini-community. We are trying to get people together on a more formal basis, and we want visitors. Other places are doing similar things. A distributed network can emerge.

Stephen Emmott sought a show of hands on support for an SFI in UK. The dissent was only in terms of widening the question to an SFI in Europe.

Jeff Johnson: all researchers want the new SFI at their institute or on their patch. We need to learn how to make the links within Europe and a network beyond.

Tim Clarke: —I'm a control engineer working on space systems, autonomous ones. The project has been running for two years, with several PhDs and other students. We're definitely on the right track given what's been said here. We have a real problem, "space, the final frontier", for which our approach is the only way forward for distant exploration. We want to collaborate on aspects such as reactive and deliberative systems, uncertainty, levels of information representation. There's a lot of work and it is all exciting; it is a good vehicle for such a grand challenge.

Geoffrey Canright: a network is not enough. SFI gives more; it gives physical co-location, and that is worth the cost of getting there. We want something the same in Europe. We need to sort it out; a network needed, but more is needed as well. We would all love it.

Cristian Calude: this year saw the Valparaiso (Chile) Institute Of Complex Systems inaugurated. It's lead by someone famous and is following in the steps of SFI. This is a model to look at and an opportunity.

John A. Clark: it would be nice to identify specific problems for PhD students to work on, to get people in. They need to be worthwhile things. We might consider providing PhD topics for other people to supervise; a good PhD programme would need this.

Stephen Emmott: in this discussion, problems in the past are dominating, which is a shame. Some future ideas are coming through.