Proceedings of the 2009 Workshop on
Complex Systems Modelling and Simulation

CoSMoS 2009

Susan Stepney, Peter H. Welch,
Paul S. Andrews, Jon Timmis,
Editors

CoSMoS 2009

Luniver Press
2009

Published by Luniver Press
Frome BA11 6TT United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

CoSMoS 2009

ISBN-10: 1-905986-22-X
ISBN-13: 978-1-905986-22-4

Preface

Building on the success of the first CoSMoS workshop, we are pleased to be running the second CoSMoS workshop in association with the 8th International Conference on Artificial Immune Systems (ICARIS), in York, UK. The immune system exemplifies a complex system — immune regulation and protection against harmful micro-organisms emerges from the interaction of large populations of different immune cells. Artificial immune systems seek to understand and exploit the properties of the real immune system. As such, the modelling and simulation of complex systems fits well within the scope of the ICARIS series of conferences.

The genesis of the CoSMoS workshop is the similarly-named CoSMoS research project[1], a four year EPSRC funded research project at the Universities of York and Kent. The project aims are stated as:

> The project will build capacity in generic modelling tools and simulation techniques for complex systems, to support the modelling, analysis and prediction of complex systems, and to help design and validate complex systems. Drawing on our state-of-the-art expertise in many aspects of computer systems engineering, we will develop CoSMoS, a modelling and simulation process and infrastructure specifically designed to allow complex systems to be explored, analysed, and designed within a uniform framework.

As part of the project, we are running annual workshops, to disseminate best practice in Complex Systems modelling and simulation. To allow authors the space to describe their systems in depth we put no stringent page limit on the submissions.

We are delighted this year to welcome the world renowned immunologist Irun Cohen from the Weizmann Institute of Science, Israel as our keynote speaker. In recent years Cohen has worked closely with David Harel, a computer scientist also at the Weizmann Institute of Science. Together they have led the way in developing predictive models and simulations of real complex systems, notably the immune system. In the first paper presented here, Cohen and Harel reflect on their experiences of coming from very difference disciplines to work together in an truly inter-disciplinary setting.

Continuing the immune system theme, Read, Timmis, Andrews and Kumar present a model of the autoimmune disease Experimental Autoimmune Encephalomyelitis in mice. The model is the first step in producing a predictive simulation of the disease to provide insight into the

[1] The CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1, http://www.cosmos-reseach.org

real system. Tools from the unified modelling language are used to express the model, which provides insight into UML's expressive capabilities when applied to complex system modelling.

Hone examines the issue of using numerical integration methods to analyse mathematical models formulated in terms of differential equations, focussing on a number of non-standard discretisation methods that have the potential to be extremely useful in modelling biological systems.

Hoverd and Stepney consider the design of complex systems and the role played by the environment in the interactions of complex systems agents. They present an abstract software architecture for environment-oriented complex system simulations, providing examples of how this could be implemented.

Nash and Kalvala focus on the use of process calculi and the well studied aggregation behaviour of the *Dictyostelium discoideum* amoeba, showing how the π-Calculus can be used to model various aspects of the aggregation behaviour such as cell locality and intra-cellular signal transduction.

Finally, Ghetiu, Alexander, Andrews, Polack and Bown examine the issue of applying argumentation techniques used for safety critical systems to complex system. The work presented here shows how these techniques can be used to argue that two different implementations of a complex system simulation are adequately equivalent.

Our thanks go to Irun Cohen for presenting his keynote and to all the contributors for their hard work in getting these papers prepared and revised. We thank the programme committee for their prompt, extensive and in-depth reviews of all the papers submitted. We would also like to thank Bob French and Mandy Kenyon from the Research Support Office in the Department of Computer Science, University of York for their invaluable assistance behind the scenes. We hope that readers will enjoy this set of papers, and come away with insight on the state of the art, and some understanding of current progress in Complex Systems Modelling and Simulation.

Programme Committee

Table of Contents

Two Views of a Biology-Computer Science Alliance [⋆]

Irun Cohen[1] and David Harel[2]

[1] Department of Immunology
The Weizmann Institute of Science, Israel
irun.cohen@weizmann.ac.il
[2] Department of Computer Science and Applied Mathematics
The Weizmann Institute of Science, Israel
dharel@weizmann.ac.il

Abstract. The editors of these *CoSMoS* Workshop Proceedings have invited us to describe the two sides of our joint approach to computer simulation of biological systems: the biological side and the computer science side. Irun Cohen, an immunologist, will voice the biological side; David Harel, a computer scientist, will voice the computer science side.

1 Cohen

I was led to David Harel and to the tools and thinking of computer science by a combination of factors emerging from my research into autoimmune diseases and the regulation of the immune system. Some of these factors are common to biology research generally and some are linked specifically to immunology and to my particular way of thinking. I shall list the factors that appear to have been the most influential for me, beginning with the general and proceeding to the particular.

1.1 The mass of experimental information staggers the imagination.

A biologist who zooms out for a comprehensive view of his or her field of study is frustrated; the mass of data defies memory and intuitive understanding. It appeared clear to me that only a computer could remember all the details and might help the biologist sort them out.

[⋆] Part of this work was supported by The John von Neumann Minerva Center for the Development of Reactive Systems at the Weizmann Institute of Science and by an Advanced Research Grant from the European Research Council (ERC) under the European Community's 7th Framework Programme.

1.2 Pleiotropism and redundancy thwart understanding.

Human understanding is most comfortable with one-to-one, linear causality; for example, intuition early on posited that each DNA gene sequence is expressed through one messenger RNA and translated into one protein with a single function. Now we have learned about DNA methylation and other epigenetic control mechanisms; alternative splicing; post-transcriptional and post-translational modifications; single proteins with five or ten different functions. In my own field, we see that a single cytokine can make some cells die, some cells live and grow, some cells differentiate – depending on the type and states of the cells. Redundancy and pleiotropism – multi-functionality – are the norm [5]. Most human minds find it difficult to keep track of pleiotropic and redundant systems; our thinking favors simple engineering diagrams. We need the help of systems and design experts to handle all the redundant and multi-functional details of living organisms.

1.3 Experimental analysis is static; living systems are dynamic.

Living systems are dynamic at every level – molecules, cells, organisms, populations; continuous interaction is the essence of biology. Yet, the experimental method almost always involves a controlled but static analysis of the defining dynamics. If we want to see how living systems work, we must go beyond analysis and study running simulations that can integrate the piecemeal experimental data to recreate a working, dynamic whole.

1.4 We will not be able to reduce living systems to fundamental laws of nature; biology is not like physics.

The paradigm established by physics asserts that truly scientific understanding is founded on simple underlying laws, preferably quantitative, that account for the manifest complexity of the real world. Science, so viewed, is the quest for fundamental laws of nature. The laws of nature endure; such laws define being. The transient, mortal details of life are trivial accidents. Physics is interested in being; physics would see no fundamental difference between C. elegans and H. sapiens; in essence, they both operate according to the same chemical reactions and cellular networks, albeit to different degrees of complexity. Biology, in contrast to physics, strives to understand the impermanent differences between C. elegans and H. sapiens; the differences are based on cursory details; the differences are not being, but becoming. Biology cannot reduce the messy details of the living organism to explanatory, fundamental laws of nature; biology is defined by attending to the fleeting details. Biology is interested in the reactive designs of life. Biologists, like me, need the help of computer scientists and their computers to record, characterize, and catalog the complex details of living systems, and to render them quantitative and dynamic. Computer scientists know about design principles. Indeed, the only principle of biology that approaches the status of a law is the notion of evolution through genetic variation and survival of the fittest – evolution is a design for becoming, not a law of being.

1.5 Mathematical analysis has been incomplete.

Biologists have collaborated with mathematicians to clarify biologic complexity by reducing it to relatively simple abstract relationships that can be analyzed by differential equations and other mathematical tools. My formative experience with mathematical theoreticians has been fruitful [1, 7, 11, 21–24] and has prepared me for later work with Harel and computer science. But these forays into theory could not take into account the rich details of the actual biology behind the phenomena that were modeled mathematically. The math misses the ontogeny.

1.6 The immune system is cognitive and computes the state of the body.

The prevailing view has been that the immune system has only one particular bias: it must not respond to self-molecules; the functional immune repertoire is selected by the foreign antigens that happen to enter the body. In contrast, my experimental investigations of autoimmunity and autoimmune diseases led me to conclude the immune system is cognitive – it has a built-in representation of particular self-molecules and molecular contexts that guides its response decisions both to self and to foreign; the immune system, as it where, knows what it is looking for [2, 3, 8].

The immune system, moreover, is engaged in an ongoing computation of the state of the body: Its receptors gather information about the body (injury, infection, the need to grow blood vessels or destroy them, the need to stimulate cells or kill them, the need to heal wounds, etc) and tranduces this input information into an output of regulated inflammation that feeds back to modify the body and the immune system itself [4–6]. The details are beyond the scope of this chapter. The point here is that my appreciation of the internal structure, the cognitive behavior and computational function of the immune system led me to see what David Harel and computer science might have to offer.

2 Harel

2.1 Origins

I had been working since the early 1980's on the problem of specifying the behavior of complex *reactive systems* [18]. This started via a one-day-per-week consultation job at the Israel Aircraft Industries (IAI), working with the avionics team of a fighter aircraft project. During that period, I proposed a *visual formalism* [13] — a rigorous diagrammatic language — for specifying reactive behavior, which I called *statecharts* [12]. This language has been around now for exactly 25 years and is considered a useful way of specifying behavior of such complicated systems [17]. It is in broad use in many industries, from cellular telephones to aircraft, automobiles, interactive software systems, and the like.

During that period the two main issues I was obsessed with were specifying the system's behavior in ways that would be intuitive and natural, but, at the very same time, formal and executable. We want a medium with which it is easy to specify and model and which is easy to understand, but, on the other hand, it must also have rigorous mathematical underpinnings, detailed and precise enough so that the system's full behavior can be executed (simulated) and analyzed by computer. The language of statecharts is a modest attempt to approach these two goals, in that it is highly diagrammatic, and seems to fit the way people think about behavior, yet it is as rigorous as any mathematical equation or computer program.

2.2 Biology and computer science

In the late 1980's it occurred to me that such problems of specifying behavior are not only common and problematic in human-made systems, such as the ones mentioned above, but might also be problematic in modeling Nature — in particular, biological systems. It seemed that the notion of a reactive system and the great difficulties that lie in designing them, are common to biological systems too. Perhaps the methods and languages used to design human-made ones could be used to help understand biology by "reverse engineering" it. However, having almost no experience in biology I wasn't able to take this much further on my own. And then, as Irun Cohen wrote in his portion of this paper, we happened upon each other, which led us to carry out an initial modest project with Na'aman Kam, a joint student who indeed started to use statecharts to specify biological phenomena [19].

The current paper will not be discussing the details of the work that we have done together in the last decade or so (see, e.g., [9, 10, 25, 26], but, in retrospect, I would submit that modeling, specifying and analyzing biology is one of the most exciting imaginable uses of computer science. This includes a lot of work by systems biologists, by bioinformatics researchers, but also by people like us who are doing something that is not exactly bioinformatics and is not exactly systems biology either. Rather, it is an attempt to mimic and imitate biology using means taken from software engineering and computer science, in an attempt to understand an entire system in a broad way [15, 16]. Many people, including myself, believe that computer science — as opposed to computer software or computation per se — will play a role in the science of the 21st century (which will no doubt be dominated by the life sciences) similar to the role played in the science of the 20th century (which was dominated by the physical sciences).

2.3 On aircraft and elephants

If we were to take as a typical example of a complex human-made system an F-16 or F-35, we might want to take as an example a complex biological system a multi-cellular organism, such as an elephant, or a lizard, or a fly, or a worm. The former has to be specified, designed, built, tested, debugged, maintained, and so on, and the latter has to be understood, which is by no means a small feat! In fact, the work that Irun and I have been doing with a cadre of extremely talented students raises the possibility of carrying out a *whole organism project* (WOP), the main goal of which would be to model an entire multi-cellular organism [14]. The organism would be modeled up to a certain pre-agreed-upon level of detail, and we want a model that would be fully and interactively executable. We would be reverse engineering an elephant, so to speak, rather than engineering an F-35...

The benefits of a successful WOP are almost unimaginable. Imagine us being able to actually simulate an elephant, in its entirety, have it develop from a fetus, have it walk and eat and play, and produce offspring.Moreover, and perhaps most importantly, during all of that the user of the model can intervene, by anything from switching off a gene, to treading on the elephant's foot and seeing what happens. If we could do that, especially if the model would reach down into the cellular and sub-cellular levels, there's almost no limit to the kinds of things we might be able to discover.

My claim is that although such a WOP is an almost unimaginably complicated task, even for a small organism, and one would need something like 15-20 years to approach it, it is possible. An elephant is obviously not a particularly good idea, but something smaller and more manageable is, such as the *C. elegans* nematode, which has a little over

1000 cells, all easily traceable in the lab. This is a multi-cellular organism of sufficient complexity to teach us an enormous amount about life, sickness and death, and the processes leading to them, but it is also of sufficiently modest proportions so as to be imaginable in a long, but reasonable, time frame [14, 16].

2.4 Visuality and formality

It is worth re-emphasizing that the recommendation is that modeling biology would have to be done using approaches that are intuitive enough so that biologists, who are not trained as mathematicians or as computer scientists, would be able to do much of the modeling themselves. My feeling is that the education that biologists will be getting in the next 10-15 years will bring them closer to the ability to master techniques for the modeling and analysis of complex systems, just as the education physicists got many tens of years ago, became a lot more mathematical, enabling them master and use mathematical techniques to model and analyze physical phenomena. Of course, as mentioned earlier, the languages and approaches used should be intuitive, but formal, so they their attractiveness to the biological community notwithstanding, they would be rigorous enough to be fully executed.

These days, biologists use many kinds of diagrams to describe pathways and networks, and the interdependency of genes and other intracellular entities. Many of these diagrams are informative and clear, but for the most part they are not sufficiently formal. Most of the approaches that advocate such diagrams do not give rise to full execution. First of all, this is due to lack of sufficiently detailed biological data and parameters; but more importantly, it is because the semantics of the diagrams themselves does not lend itself to full executability; see [20]. In contrast, the approaches we have been using (for example, statecharts-based modeling) are such that once you have your model intact, it can be run, or executed just like any other computer program.

2.5 One more word on the collaboration

From my personal point of view, the collaboration with Irun, which started in the late 1990's, has been one of the most productive and exciting periods of my scientific life. I hope that we will continue to work for many years and that our work will be at least as exciting as it has already been. Perhaps we will indeed be able to contribute a little to the feeling that the marriage between computer science and the life sciences is already, but will become even more so, one of the most fruitful marriages in modern science.

3 Two Views but One Product

Our two views of this biology-computer science alliance, as you can see here, reflect two different vantage points. Each of us entered the collaboration motivated by different personal and professional histories, arising from our different trainings, experiences and mindsets. Our colleagues, nevertheless, need see only one product of interest to science — the work itself, which is the unity of our joint thinking and our joint efforts. The meaning of this unified opus will, in turn, diverge into the different ideas and experiments that we hope our work will trigger in our fellow scientists. Such is the evolution of science: diverse views generate a new viewpoint that generates new diverse views.

References

[1] H. Atlan and I.R. Cohen. Immune information, self-organization and meaning. *Int. Immunol.*, 10(6):711–717, 1998.

[2] I.R. Cohen. The cognitive paradigm and the immunological homunculus. *Immunol, Today*, 13(12):490–494, 1992.

[3] I.R. Cohen. The cognitive principle challenges clonal selection. *Immunol. Today*, 13(11):441–444, 1992.

[4] I.R. Cohen. Discrimination and dialogue in the immune system. *Semin. Immunol.*, 12(3):215–9; 321–323, 2000.

[5] I.R. Cohen. *Tending Adam's Garden: Evolving the Cognitive Immune Self.* Academic Press, London, UK, 2000.

[6] I.R. Cohen. Immune system computation and the immunological homunculus. In *MoDELS 2006*, pages 499–512. Springer-Verlag, Berlin, 2006.

[7] I.R. Cohen, U. Hershberg, and S. Solomon. Antigen-receptor degeneracy and immunological paradigms. *Mol. Immunol.*, 40(14-15):993–996, 2004.

[8] I.R. Cohen and D.B. Young. Autoimmunity, microbial immunity and the immunological homunculus. *Immunol. Today*, 12(4):105–110, 1991.

[9] S. Efroni, D. Harel, and I.R. Cohen. Towards rigorous comprehension of biological complexity: Modeling, execution and visualization of thymic t cell maturation. *Genome Research*, 13:2485–2497, 2003.

[10] S. Efroni, D. Harel, and I.R. Cohen. Emergent dynamics of thymocyte development and lineage determination. *PLOS Computational Biology*, 3(1):127–136, 2007.

[11] Z. Grossman and I.R. Cohen. A theoretical analysis of the phenotypic expression of immune response genes. *Eur. J. Immunol*, 10(8):633–640, 1980.

[12] D. Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Programming*, 8:231–274, 1987.

[13] D. Harel. On visual formalisms. *Comm. Assoc. Comput. Mach.*, 31(5):231–274, 1988.

[14] D. Harel. A grand challenge for computing: Full reactive modeling of a multi-cellular animal. *Bulletin of the EATCS (European Association for Theoretical Computer Science)*, 81:226–235, 2003.

[15] D. Harel. On comprehensive and realistic modeling: Some ruminations on the what, the how and the why. *Clinical and Investigative Medicine*, 28(6):334–337, 2005.

[16] D. Harel. A turing-like test for biological modeling. *Nature Biotechnology*, 25:495–496, 2005.

[17] D. Harel. Statecharts in the making: A personal account. *Comm. Assoc. Comput. Mach.*, 52(3):67–75, 2009.

[18] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems (NATO ASI Series, Vol. F-13)*, pages 477–498. Springer-Verlag, 1985.

[19] N. Kam, I.R. Cohen, and D. Harel. The immune system as a reactive system: Modeling t cell activation with statecharts. In *Proc. Visual Languages and Formal Methods (VLFM'01)*, pages 15–22, 2001.

[20] H. Kugler, A. Larjo, and D. Harel. Biocharts: A visual formalism for complex biological systems. *to appear.*

[21] Y. Louzoun, H. Atlan, and I.R. Cohen. Modeling the influence of th1- and th2-type cells in autoimmune diseases. *J. Autoimmun.*, 17(4):311–321, 2001.

[22] Y. Louzoun, S. Solomon, H. Atlan, and I.R. Cohen. Modeling complexity in biology. *Physica A*, 297:242–252, 2001.

[23] Y. Louzoun, S. Solomon, H. Atlan, and I.R. Cohen. Proliferation and competition in discrete biological systems. *Bull. Math. Biol.*, 65(3):375–396, 2003.

[24] L.A. Segel, E. Jager, D. Elias, and I.R. Cohen. A quantitative model of autoimmune disease and t-cell vaccination: does more mean less? *Immunol. Today*, 16(2):80–84, 1995.

[25] Y. Setty, I. R. Cohen, Y. Dor, and D. Harel. Four-dimensional realistic modeling of pancreatic organogenesis. *Proc. Natl. Acad. Sci.*, 105(51), 2008.

[26] N. Swerdlin, I. R. Cohen, and D. Harel. The lymph node b cell immune response: Dynamic analysis in-silico. *Proceedings of the IEEE (special issue on Computational System Biology)*, 96(8):1421–1443, 2008.

A Domain Model of Experimental Autoimmune Encephalomyelitis

Mark Read[1], Jon Timmis[1,2], Paul S. Andrews[1], and Vipin Kumar[3]

[1] Department of Computer Science, University of York, UK.
{markread,jtimmis,psa}@cs.york.ac.uk
[2] Department of Electronics, University of York, UK.
[3] Laboratory of Autoimmunity, Torrey Pines Institute for Molecular Studies, CA, USA.

Abstract. Experimentation with simulations of complex systems can be used to gain insights into those systems' nature and operation. Such *in silico* experimentation represents a valuable tool that can complement conventional *in vivo* experimentation. Validation of a simulation's representation of the real world system remains an open question in complex systems research. As the engineer of a complex system simulation, demonstrating one's understanding of the complex system through the creation of models which can be validated by a domain expert affords some degree of confidence that the results obtained through *in silico* experimentation are representative of the real world system. As a precursor to the creation of simulations of experimental autoimmune encephalomyelitis, a complex autoimmune disease in mice, we present here a model of the disease. The models are expressed using UML, and their construction has afforded insight into UML's expressive capabilities when applied to complex system modelling.

1 Introduction

Experimental Autoimmune Encephalomyelitis (EAE) [14, 15] is an autoimmune disease in mice that serves as a model for multiple sclerosis in humans. The disease, and its subsequent spontaneous recovery, is complex. A large number of immune system cells interact with one another across several bodily compartments to mediate both EAE autoimmunity and its recovery. As is the case with many complex systems EAE is difficult to understand through a reductionist scientific approach alone. The construction of models and simulations of the disease can afford insights into the disease's behaviour and can guide wet-lab experimentation to points of interest. Experiments that would be difficult to engineer *in vivo* can be engineered into a simulation with relative ease. Simulations permit the investigation of hypotheses concerning the disease's operation within the context of known biological data.

A major concern for *in silico* experimentation using a simulation is the simulation's validity. It is important that one can demonstrate that the results obtained through experimentation with a simulation are representative of the *in vivo* system. How to demonstrate the validity of a simulation remains an open question in complex systems research. The CoSMoS project[4] [1] purposes to develop general principles for the creation and validation of models and simulations of complex systems [2]. The project is developing "the CoSMoS process", an approach to the engineering of complex systems that proposes the construction of models of the complex system as a prerequisite to the construction of any simulations of it.

It is critical that the developers of a complex system simulation possess a decent[5] understanding of how the system works. By developing models of the complex system this understanding can be demonstrated, and can be validated by a domain expert. The construction of such models frequently necessitates examination of the complex system from angles that might not otherwise be considered, and can raise further questions of its operation. The models themselves can form a specification for the construction of a complex system simulation. Validation of a simulation's specification (models) by a domain expert can go some way towards instilling confidence that the simulation is representative of the real world system.

The unified modelling language (UML) is a collection of diagrammatic tools that are were designed for the purpose of specifying software systems. It has been suggested that UML holds potential for modelling biological systems [6]. In this paper we present a model of EAE expressed using UML. By employing UML in this fashion we have identified several strengths and weaknesses of the language when expressing complex biological systems, some of which are outlined in [19]. We find that although UML incorporates several mechanisms that are useful in expressing EAE, such as activity diagrams and state machine diagrams, there are aspects of the biological system that UML cannot satisfactorily communicate. For example, feedback mechanisms that manifest through populations

[4] The CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1, http://www.cosmos-reseach.org.

[5] A "complete" or "detailed" understanding is not always possible, since attempting to resolve uncertainty concerning the system's nature is one motivation for developing the simulation in the first place.

of cells that amplify and counteract each other's operation. Through constructing the present domain model of EAE we have identified some general principles and approaches for applying UML to modelling complex systems. The models we present here serve as an example of how to create a domain model of a complex biological system.

Section 2 details the CoSMoS approach to developing models and simulations of complex systems. Section 3 provides a detailed account of EAE and its recovery. Section 4 introduces UML. In Section 5 we present the model of EAE, highlighting the assumptions we have made in its creation, and how UML has been employed in the generation of the models. Section 6 concludes the paper.

2 CoSMoS Process and the Domain Model

The CoSMoS project is concerned with the development of a modelling and simulation infrastructure that facilitates the design and analysis of complex systems [3]. Ongoing work within CoSMoS seeks to develop a "minimal process" for the development of models and simulations of complex systems. The creation of accurate models and simulations is non-trivial, and demonstrating that they are representative of the real complex systems that they attempt to capture is critical to their use in research. The minimal process represents a first step towards building *validated* models of complex systems.

Figure 1 shows the minimal process as it currently stands. The domain represents the real-world system of interest, in this case EAE. The models and simulations constructed attempt to capture behaviours and properties exhibited by this real world system.

The domain model details the current understanding of the biological domain as held by the modeller. It captures the behaviours present in the biological domain that the modelling and simulation process hopes to investigate. A domain model may span multiple levels of abstraction, from the high level depiction of perceived emergent behaviours exhibited at a system-wide level, to the low-level entities of the system and how they interact with one another. The model should be free from any implementation-specific bias. Validation of the domain model is important, and is carried out by a domain expert. If the domain model is invalid, then the understanding that the modeller has of the system is most likely incorrect, and any simulation built upon that understanding is unlikely to be representative of the real biological domain.

The software model is constructed from the concepts captured in the domain model. It is tailored toward the design and implementation of the

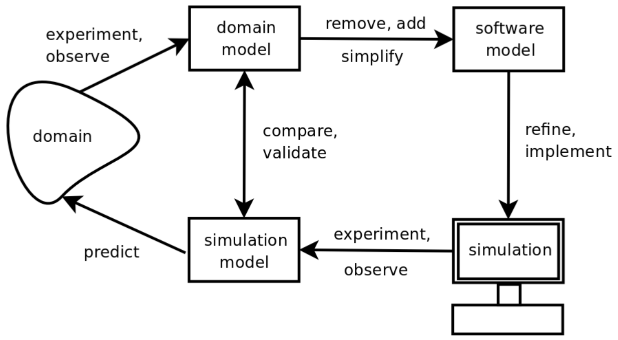

Fig. 1. The CoSMoS minimal process for the development of complex systems simulations [4].

simulator itself; explicit notions of emergent properties and behaviours are removed, and implementation specific concepts may be introduced.

The simulation model is derived through observations of and experimentation with the simulation. The simulation model is to the simulation what the domain model is the domain. Validation can be performed between the domain model and the simulation model; if the simulation correctly captures the desired behaviours present in the domain, then the simulation model should closely resemble the domain model.

The nature of complex systems research dictates that there will exist unknown aspects of the system that no domain expert can be sure of. One of the motivations in creating models and simulations of the system is to bring these areas to light, but where no certain answer currently exists an assumption must be made. Likewise, it is infeasible to model and hence reason about every aspect of a complex system system; abstracting away from complexity believed not to be integral to the system's behavioural dynamics is essential. Whenever an abstraction is made to simplify the system there entails an implicit assumption that the abstraction will not compromise the simulation/model's capture of the behaviours present in the real-world system. These assumptions should be recorded and validated by the domain expert as being appropriate and sensible. If the simulation fails to properly capture the behaviours of interest, then it is likely that an assumption was inappropriate, and it should be readdressed. Thus, the minimal process is iterative.

It is important to recognise and document the questions and issues that one hopes to address through modelling and simulating a complex system. The nature of the assumptions and abstractions that are made in constructing models and simulations of a complex system are motivated by what those models and simulations are to be used for. The assumptions and abstractions must be appropriate for the problem at hand.

In the event that a simulation fails to capture a complex system's emergent properties, yet all abstractions and transitions between the minimal process's models were deemed just and appropriate by a domain expert, then it may hold that some higher level hypothesis upon which the simulation was built is incorrect. It is essential that the simulation's world be properly delineated. There exists a huge variety of interacting elements in a biological system, and accurately simulating all of them is impractical. At its highest level the domain model assumes a hypothesis over which elements in the complex system are responsible for the manifestation of some target abstract behaviour, and thus which elements it will represent. It is plausible that this hypothesis itself will prove to be incorrect, hence the importance of documenting it.

3 Experimental Autoimmune Encephalomyelitis

In this section we provide a detailed description of experimental autoimmune encephalomyelitis and its subsequent recovery. The information provided here provides the basis for the domain model that follows.

3.1 Autoimmunity

Experimental autoimmune encephalomyelitis (EAE) is an autoimmune disease in mice that serves as a model for multiple sclerosis in humans [14, 15]. The disease constitutes the body's immune system attacking myelin, an insulator material that covers the neurons of the central nervous system (CNS) and is essential to their function. Damage to the CNS through demyelination can lead to paralysis and death [16].

Figure 2 presents an informal depiction of how EAE is induced through immunisation with MBP, a myelin derivative. The immunisation is accompanied by complete Freund's adjuvant (CFA) and pertussis toxin, both immunopotentiators which stimulate the immune system. The immunisation occurs subcutaneously. The phagocytosis of MBP by dendritic cells (DCs) resident in the periphery leads to its presentation as MHC-I-MBP and MHC-II-MBP molecules on the DCs. The CFA and pertussis toxin stimulate the DCs, and they up-regulate co-stimulatory molecule expression and migrate to the secondary lymphoid organs. There populations of naive autoimmune MBP-reactive CD4Th1 and CD4Th2 cells bind with MHC-I-MBP as expressed on the immigrant DCs and derive signal 1. The high level of co-stimulatory molecule

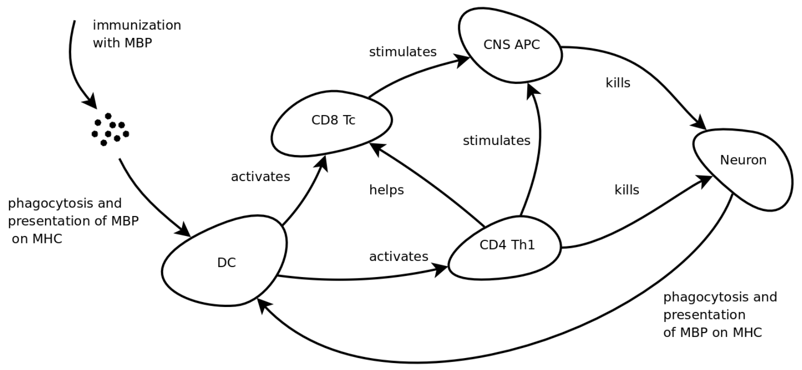

Fig. 2. An informal depiction of how EAE is induced.

expression by these DCs delivers signal 2 to the CD4Th1 and CD4Th2 cells resulting in their activation. MHC-I-MBP molecules, as expressed by these same DCs, are bound by MBP reactive CD8 cytotoxic T (Tc) cells. With help of the MBP-reactive CD4Th1 cells these Tc cells become fully activated.

The now activated CD4Th1, CD4Th2, and CD8Tc cells migrate to the CNS compartment. The CD4Th1 and CD8Tc cells secrete pro-inflammatory type 1 cytokines such as IL-2, INF-γ, and TNF-β [13]. These cytokines represent an inflammatory context to resident antigen presenting cells (APCs) such as macrophages and microglia which become stimulated. When stimulated these CNS APCs secrete TNF-α, reactive oxygen species (ROS), and nitric oxide (NO), all of which are toxic to neurons in high doses [11, 18, 22]. Neurons contain MBP, and those that are killed in this manner are subsequently phagocytosed by CNS APCs, which then express MHC-I-MBP and MHC-II-MBP. The inflammatory conditions in the CNS prompt these APCs to upregulate co-stimulatory molecules, and hence induce the full activation of naive CD4Th1, CD4Th2, and CD8Tc cells that result from proliferation in the CNS.

The CD4Th2 cells secrete type 2 cytokines such as IL-4, IL-5, and IL-10. Type 1 cytokines suppress Th2 cell activity, and type 2 cytokines suppress that of Th1 cells, reducing the cells' proliferative and differentiation capabilities [13]. During the course of EAE autoimmunity the Th1 cell population is dominant, they have a higher affinity for MHC-I-MBP (bindings are stronger and last longer, resulting in less failings to receive signal 1 before the bindings are broken) and proliferate more quickly.

3.2 Regulation-mediated recovery

A network of immune cell interactions can mediate recovery from EAE, and is depicted (at an abstract level) in Figure 3. The physiological turnover of CD4Th1 cells results in their apoptotic death, and subsequent phagocytosis by APCs (such as the dendritic cell) in the CNS draining lymph nodes. Two regions of the T cell receptor (TCR) of MBP-reactive CD4Th1 cells form peptides that are presented on MHC molecules by the APC to prime two populations of regulatory T cell (Treg). These two regions are complementarity determining region 1/2 (CDR1/2) and Fr3, which are presented on non-classical MHC-I (Qa-1) and MHC-II respectively. Binding of MHC-II-Fr3 by Fr3-reactive CD4Treg cells leads to their receipt of signal 1. Molecules generated by the inflammation in the CNS drain into the draining lymph nodes and stimulate the APCs that reside there to upregulate their expression of co-stimulatory molecules. This upregulated expression of co-stimulatory molecules delivers signal 2 to the CD4Treg cells. When activated, and upon binding with MHC-II-Fr3, CD4Treg cells secrete INF-γ, which is required for the processing and presentation of CDR1/2 on non-classical MHC-I (Qa-1) molecules by the APC [23]. This phenomenon is called "licensing" of the APC by the CD4Treg.

CD8Treg cells bind with MHC-I-CDR1/2 as expressed on APCs resident in the CNS's draining lymph nodes and derive signal 1. The high level of co-stimulatory molecule expression on these APCs allows the CD8Treg cells to derive signal 2, becoming fully activated. For a short period of time, around eight hours, following their initial activation CD4Th1 cells express MHC-I-CDR1/2. If this is bound by a fully activated CD8Treg cell the Treg cell can induce the apoptotic death of the CD4Th1 cell.

On a population-wide scale this rise in CD8Treg cell population number leads to a reduction of CD4Th1 number. This occurs in the circulatory system and lymphoid organs such as the spleen; however, Treg cells have not been identified in the CNS, hence this regulation is not assumed to occur there. The transient expression of MHC-I-CDR1/2 by CD4Th1 cells renders them susceptible to regulation for only a short period of time into their full activation. Once this period of time has passed these cells are still susceptible to death through the Fas-FasL pathway. Once activated T cells begin to upregulate their expression of both Fas and FasL on their cell membranes. Sufficient bindings between these two types of molecule can induce a cell's death. Apoptotic death though the Fas-FasL pathway is called activation induced cell death (AICD). The decline in CD4Th1 population number through regulation results in a global reduction of type 1 cytokines being produced. This reduction al-

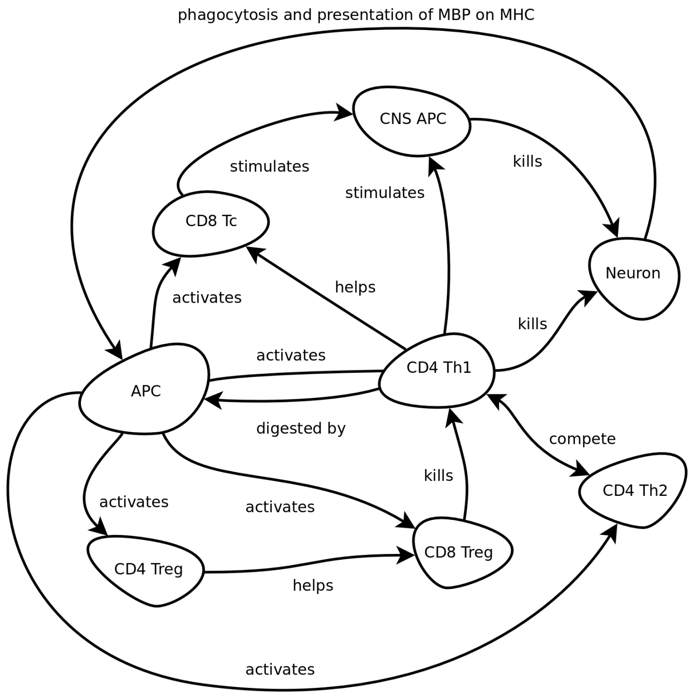

Fig. 3. An informal depiction of all the cells involved in EAE and its regulation-mediated recovery, and their relations to one another.

leviates the suppression of the CD4Th2 population, which then expands and assumes dominant status. The activity of MBP-reactive CD4Th2 cells is not toxic to neurons, and their population expansion does not result in EAE.

4 The Unified Modelling Language

The unified modelling language (UML) [17] is a collection of diagrammatic modelling tools designed to aid the specification and construction of software systems. The diagrammatic tools incorporated within UML provide a wide range of specification scopes, from the relationships held across an entire software system, to full low-level expression of a single system entity. UML diagrams can represent both static and dynamic views of a system. Static views depict the relationships that system entities may hold with one another, whilst dynamic views express the collaborations between system entities and the changes to their internal states (which influence their external behaviours).

This multi-viewed approach to specifying systems has made UML a popular modelling tool, and it finds application outside of the software domain within which it was conceived. There have been numerous attempts to model biological systems using UML, for example, [2, 6, 9, 20] [7, 12][6].

In total UML describes 13 different modelling notations [8]. In the domain model presented here we make use of class diagrams, activity diagrams, and state machine diagrams. An assessment of the use of various UML diagram notations in creating the present domain model is to appear in ICARIS 2009 [19].

Class digrams depict the static relationships between entities in the system. Relationships can be assigned a role name, and cardinalities at both ends of the relationship indicate how many instances of each entity may partake in a relationship at any one point in time.

Activity diagrams represent a dynamic view of a system, and indicate an ordering of events between instances of system entities that occur within a particular scenario. The events (called activities) depicted in an activity diagram may be any abstract concept.

State machine diagrams are a dynamic view of individual entity types in the system. All instances of a particular entity follow the dynamics defined for their type. State machine diagrams describe the states that an entity may exist in. An entity's state determines which events and interactions it is capable of partaking in. States can be mutually exclusive, orthogonal, and hierarchical.

5 Domain Model of EAE

This section presents the domain model of EAE and its regulation. As detailed in section 2, the abstractions and assumptions that are made in arriving at a simulation of a complex system are heavily dependent on the intended purposes of the simulation, and so we document these purposes here.

The models and simulations we are constructing are for the purposes of conducting *in silico* experimentation. Through construction of a simulation of EAE that intergrates known biological data about the disease we hope to extrapolate the potential values of otherwise unknown biological parameters. This is elaborated upon in section 5.2. Once constructed,

[6] These works used state charts [10] as their modelling medium. State charts are very similar to the state machine diagrams of UML.

it is trivial to remove or alter the nature of entities in the simulation. By observing the altered dynamic of the system we hope to ascertain the importance of those entities to the system's behaviour. As an example, we can experiment with the length of time that a recently activated CD4Th1 cell expresses MHC-I-CDR1/2 for, and observe how the system's behavioural dynamics are affected by this change. Such experimentation is extremely challenging to engineer into an *in vivo* system, yet is relatively trivial to perform through simulation. By performing *in silico* experimentation we hope to highlight areas of significance within the system that can then be used to direct wet-lab experimentation to points of interest.

5.1 Delineating the system

As detailed in section 2, it is essential to delineate the system of interest; modelling and simulating an entire biological system is intractable. Figure 4 denotes the observable phenomena of the real-world domain. Argument over these phenomena is deemed to be outside the scope of this modelling work. "Autoimmunity" is an overloaded term which immunologists may disagree over the exact origins of; there is more than one form of autoimmunity. This diagram delineates the system we intend to model in exact terms, both the physical entities within it and the behaviours we expect them to manifest, omitting overloaded definitions such as "autoimmunity". Note that Figure 4 does not conform to any UML notation.

The diagram explicitly depicts several levels of hypothesis that the model and simulation will incorporate. The transitions across the dotted line depict our hypotheses concerning those abstract behaviours/events that we believe to be responsible for the observable phenomenon. These transitions delineate the outer bounds of our investigations; we will not attempt to investigate whether anything other than our "expected behaviours" are responsible for the observable phenomenon. Our investigations are scoped within the context of these expected behaviours, as indicated on the diagram. It is these expected behaviours that we are attempting to capture in our models and simulations. The transitions over the dotted line indicate how the work carried out with our simulations fits into the wider context of study on EAE and autoimmunity.

Further hypotheses are detailed in the links between the expected behaviours and the real physical entities of the system. These links indicate which entities in the real-world system we believe to be responsible for manifesting the expected behaviours, and will thus find explicit representation within our system. The "expected behaviours" are so named because we expect these system-wide behaviours to manifest from the lower level entities and their interactions, as depicted on the diagram (albeit at an abstract level).

In our case the observed phenomena are that mice experience paralysis from EAE, and that the immune system is regarded as being responsible for carrying out damage to the central nervous system. Mice can recover from EAE spontaneously. Following recovery mice are typically insusceptible to further attempts to induce EAE autoimmunity in them.

We hypothesise that "autoimmunity against the CNS" is caused by the behaviour of immune cells harming central nervous system (CNS) cells. This behaviour manifests through the actions of several immune cells. Dendritic cells activate auto-reactive CD4 Th1 cells, which in turn facilitate the activation of CD8 Tc cells, which together stimulate CNS Macrophages into secreting molecules that are toxic to neurons (CNS Cells). These actions are all quite abstract, and are expanded upon in other diagrams, as discussed below.

Of note is that one of the observable phenomena is not linked to an expected behaviour. We are unsure as to what is responsible for "protection against subsequent attempts to induce autoimmunity against CNS". Two possibilities include the establishment of an equilibrium between the rise of CD4Th1 cells and their apoptotic death through regulation, or the action of memory Treg cells that efficiently subvert the onset of autoimmunity before significant damage is caused.

Figure 4 is an alternative to another technique of expressing the expected behaviours or emergent properties of a system; Garnett *et al.* [9] have represented the emergent property that their simulations attempted to capture as a first class entity on a class diagram. We have found this technique unsuitable for EAE; a class labelled "autoimmunity" or "regulation" cannot be instantiated in the same manner that class labelled "dendritic cell" can be, yet their representation as such would imply same semantic behaviour. Instead, through Figure 4, we have captured the system-wide behaviours of autoimmunity and regulation as unique abstract entities and linked them to the physical components in the system responsible for their manifestation.

5.2 Validating models and simulations

A central issue for validation of a simulation of a complex system, and results obtained thereof, is identifying how well it captures the behaviours exhibited by the real world system. In the case of EAE there is no available metric to measure how well "autoimmunity" has been captured. *In vivo* experimentation defines a scale based upon the degree of paralysis experienced by a subject. Since modelling and simulating the entire mouse is unmanageable, a parallel of this metric for the simulation is not possible. It is unknown how much "damage" to a central nervous system in terms of neuron death corresponds to a particular degree of paralysis, so although neurons can find explicit representation within a simulation the extent to which they are attacked by the immune system cannot be used as a metric either.

Through interaction with a domain expert a timeline of significant events that can be observed within the *in vivo* system can be devised. In the case of EAE these events are depicted in Table 1. They correspond with observations made at the cell population level. This timeline can potentially be used to validate a simulation's capture of EAE; if the population dynamics within the simulation match those of the timeline, and if the the behaviours of entities represented within the simulation are validated by the domain expert, then some level of confidence that the simulation is representative of EAE can be obtained.

The nature of current immunological research dictates that not all biological parameters are known, and some will be subject to controversy within the field. This is the case with EAE, and presents a problem for any simulation that attempts to capture it. The present domain model details which biological parameters are and are not known; see Table 2. Those that are known can be incorporated into a simulation, whilst those that are not will be subject to experimentation. Given the timeline of *in vivo* EAE, and that the behavioural dynamics of the cells that mediate it are validated by a domain expert, correct values for the unknown biological parameters should recreate the timeline within the simulation. This hinders on the assumptions that have been made in arriving at the simulation being appropriate and valid, as indicated by a domain expert. Documenting these assumptions is critical for determining the validation of models and simulations. Appendix A captures the assumptions made in the present domain model.

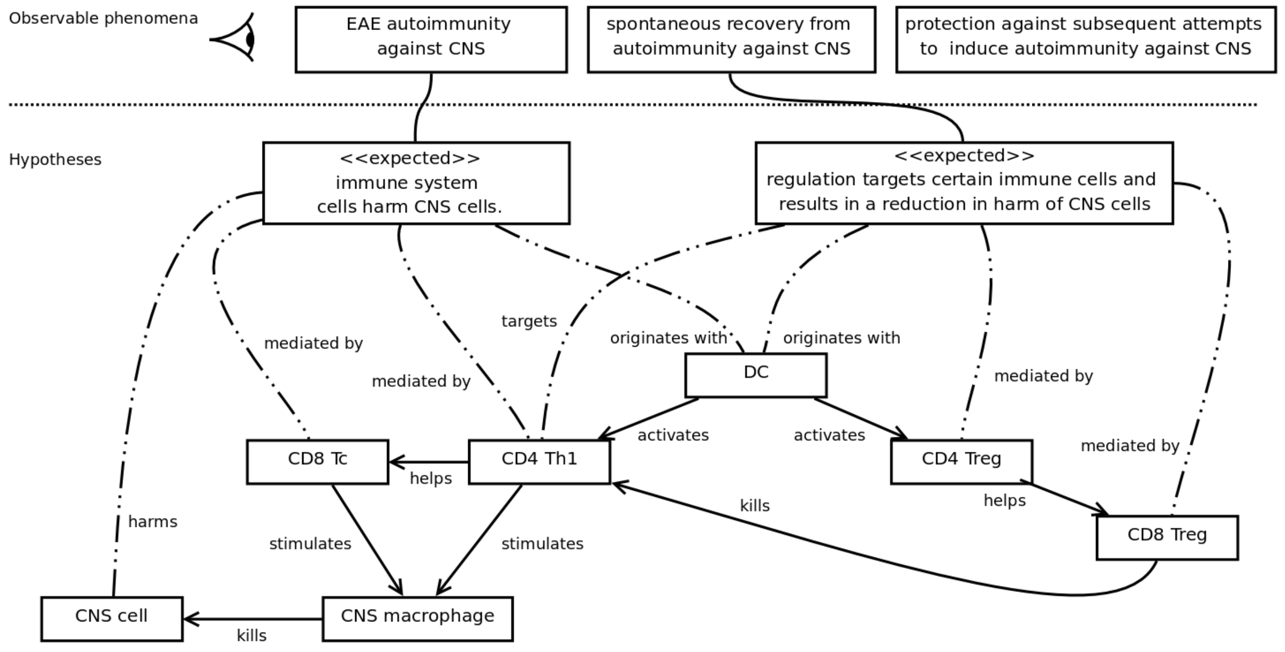
Observable phenomena
EAE autoimmunity against CNS
spontaneous recovery from autoimmunity against CNS
protection against subsequent attempts to induce autoimmunity against CNS
Hypotheses
<<expected>> immune system cells harm CNS cells.
<<expected>> regulation targets certain immune cells and results in a reduction in harm of CNS cells
mediated by
mediated by
targets
originates with
originates with
mediated by
mediated by
DC
activates
activates
CD8 Tc
helps
CD4 Th1
CD4 Treg
helps
kills
CD8 Treg
harms
stimulates
stimulates
CNS cell
kills
CNS macrophage

Fig. 4. This diagram details: the observable phenomenon of the biological domain; the behaviours that we hypothesise to be responsible for those phenomenon; and, at an abstract level, which physical entities of the real-world biological domain we believe to be responsible for manifesting those behaviours. Note that there are not formal semantics attached to this diagram.

Time	Event
0 days	Immunisation with MBP, CFA, and pertussis toxin in the periphery
3-5 days	Detectable proliferation of CD4Th1, CD4Th2, and CD8Tc cells in the secondary lymphoid organs
5-7 days	Detectable proliferation of CD4Th1, CD4Th2, and CD8Tc cells in the CNS
10-15 days	Visible paralysis of mouse
10 days	Detectable proliferation of CD4Treg and CD8Treg cells in secondary lymphoid organs
30-40	Recovery from EAE.

Table 1. The key population level events in EAE and its regulation-mediated recovery.

Event	Time
Delay in phagocytosis of substance to its appearance on MHC	1-2days
Delay in upregulation of co-stimulatory molecules following stimulation of APC	~2 hours
Persistence of Qa-1-CDR1/2 on CD4Th1 cell following activation	~8 hours
CD4Th2 cell dies from AICD after initial activation after	~8 days
Stimulated, activated CD4Th2 cell proliferates every	~3 days
Other T cells die from AICD after initial activation after	~5 days
Other activated T cells, given sufficient stimulation, proliferate every	~2 days
Persistence of MHC-peptide on APC membrane	unknown
Lifetime of DC	unknown
Half life of cytokine (this is the case for all cytokines in the domain model)	unknown

Table 2. The time taken for key events within EAE to occur. Some of these biological parameters are not known.

5.3 Modelling expected behaviours

The expected behaviours of autoimmunity ("Immune system cells harm CNS cells") and regulation, as depicted on Figure 4, represent system-wide behaviours that manifest from low-level interactions between cells. Activity diagrams represent a powerful medium in which to express how these scenarios occur. The activity diagrams in Figures 5 and 6 depict the order in which events at the individual cell level occur for autoimmunity and regulation to manifest. Figure 7 shows how the single cell events in regulation (Figure 6) translate to deviation from autoimmunity at the system-wide level.

The events depicted as activities in these activity diagrams are abstract concepts. They do not themselves specify the behavioural dynamics of individual cells given a range of scenarios; that is accomplished through use of state machine diagrams, discussed below. The activity diagrams are very effective at showing how the individual cell-level dynamics expressed in state machine diagrams integrate to constitute a system-wide dynamic. EAE contains many cascades of events, and the top level behaviours manifest from the interactions of population dynamics which themselves manifest through the concurrent actions of many individual cells. Of all the diagrams defined within UML, activity diagrams are the most expressive in terms of depicting a break down of system-wide dynamics.

From the activity diagrams that depict scenarios within the system, class diagrams that represent a static perspective of the scenario can be created. Figures 8 and 9 represent class diagrams of EAE and regulation respectively. Class diagrams are concerned with expressing the relationships that entities in the system hold with one another, and the number of entities that take part in those relationships at any particular point in time. These diagrams are somewhat informative for EAE and its recovery; however, reasoning about the system in a static manner is not as informative to its operation as is examination from a dynamic viewpoint. Ordinarily there is little to constrain the number of biological entities that attempt (either successfully or not) to partake in a particular interaction at a time, and this can potentially manifest in "0..*" cardinalities on class diagrams (note that in diagrams 8 and 9 this is not the case, due to assumption 1). "0..*" cardinalities appearing all over a diagram can perhaps convey that the system is complicated, but they do not highlight how the system operates. Furthermore, in biology there is relatively little to stop anything from attempting to interact with anything else, and many such interactions produce effects. This can lead to highly connected class diagrams that are difficult to interpret in a meaningful manner. The approach taken in Figures 8 and 9 has been to represent *partial* class diagrams that depict a subset of the entire system; in this case delineated by how low level entities manifest different high level system-wide behaviours.

5.4 Capturing low-level dynamics of system entities

State machine diagrams are used to depict low-level behavioural dynamics, and are constructed for all entities that either actively change the state of the system, or those that play important roles in mediating system dynamics.

Correctly capturing the dynamics of a complex biological entity, such as a cell, on a two dimensional diagram can prove challenging. The dynamics of a cell exhibit high dimensionality, and the dimensions are not necessarily completely independent. As an example, Figure 10 shows the state machine diagram for a CD4Th1 cell. The locations in which the CD4Th1 cell may reside are depicted as a mutually exclusive set of states that are orthogonal to the rest of the cell's behaviour; however this is not really the case. The state transitions that depend on binding with MHC-II-MBP complexes can only occur when the cell is in the SLO or CNS compartments, since these are the only locations in the model where APCs reside. Depicting this diagrammatically would make the diagram very cluttered. The use of guards for relationships such as this would add to the complexity of the diagram. This particular example is covered by the case that the state machine diagrams of the dendritic cell and CNS macrophage (Figures 15 and 16) indicate where they can reside. However, the purpose of these models is to be informative and transparent; excess complexity should be avoided.

There are behavioural aspects that are impossible to represent through conventional use of state machine diagrams. For example, Figure 15 presents the state machine diagram for a dendritic cell. The levels of MHC-I-peptide presentation are dependent on: being licensed by a CD4 T cell; the quantity of peptide available within the cell for presentation; the level of stimulation within the cell, which is itself dependent on the perception of CFA and type 1 cytokines. Exactly how these variables interact with one another to dictate MHC-I-peptide presentation is not completely clear; resolution of these unknowns will require experimentation with the simulation and interaction with the domain expert. However, even if this were not the case, state machine diagrams incorporate no way for us to represent these complex relationships and variable values without the use of equations or significant quantities of text.

Several of the state machine diagrams presented here incorporate a single state with no transitions to alternative states that is orthogonal to the others, for example "express MHC-II-peptides" on Figure 16. It

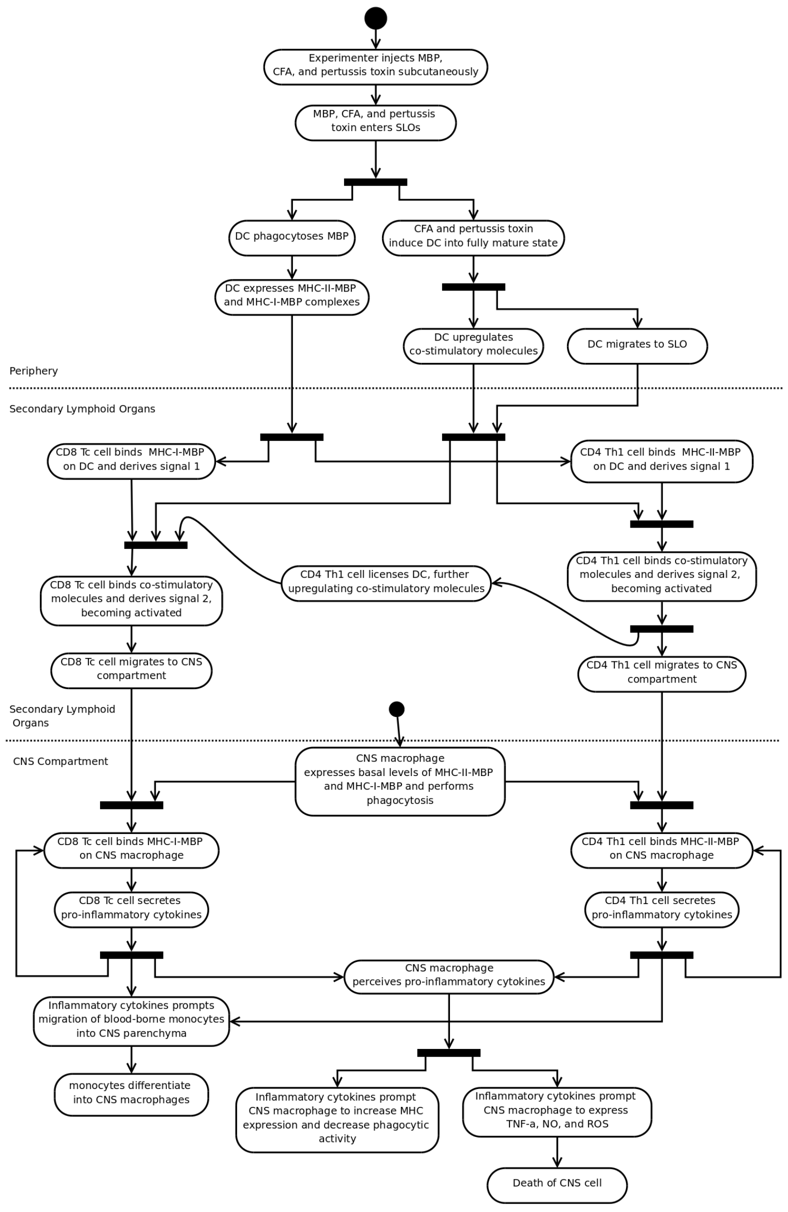
Experimenter injects MBP, CFA, and pertussis toxin subcutaneously
MBP, CFA, and pertussis toxin enters SLOs
DC phagocytoses MBP
CFA and pertussis toxin induce DC into fully mature state
DC expresses MHC-II-MBP and MHC-I-MBP complexes
DC upregulates co-stimulatory molecules
DC migrates to SLO
Periphery
Secondary Lymphoid Organs
CD8 Tc cell binds MHC-I-MBP on DC and derives signal 1
CD4 Th1 cell binds MHC-II-MBP on DC and derives signal 1
CD4 Th1 cell binds co-stimulatory molecules and derives signal 2, becoming activated
CD8 Tc cell binds co-stimulatory molecules and derives signal 2, becoming activated
CD4 Th1 cell licenses DC, further upregulating co-stimulatory molecules
CD8 Tc cell migrates to CNS compartment
CD4 Th1 cell migrates to CNS compartment
Secondary Lymphoid Organs
CNS Compartment
CNS macrophage expresses basal levels of MHC-II-MBP and MHC-I-MBP and performs phagocytosis
CD8 Tc cell binds MHC-I-MBP on CNS macrophage
CD4 Th1 cell binds MHC-II-MBP on CNS macrophage
CD8 Tc cell secretes pro-inflammatory cytokines
CD4 Th1 cell secretes pro-inflammatory cytokines
CNS macrophage perceives pro-inflammatory cytokines
Inflammatory cytokines prompts migration of blood-borne monocytes into CNS parenchyma
monocytes differentiate into CNS macrophages
Inflammatory cytokines prompt CNS macrophage to increase MHC expression and decrease phagocytic activity
Inflammatory cytokines prompt CNS macrophage to express TNF-a, NO, and ROS
Death of CNS cell

Fig. 5. Activity diagram representing the order of low-level inter-cellular events that lead to EAE.

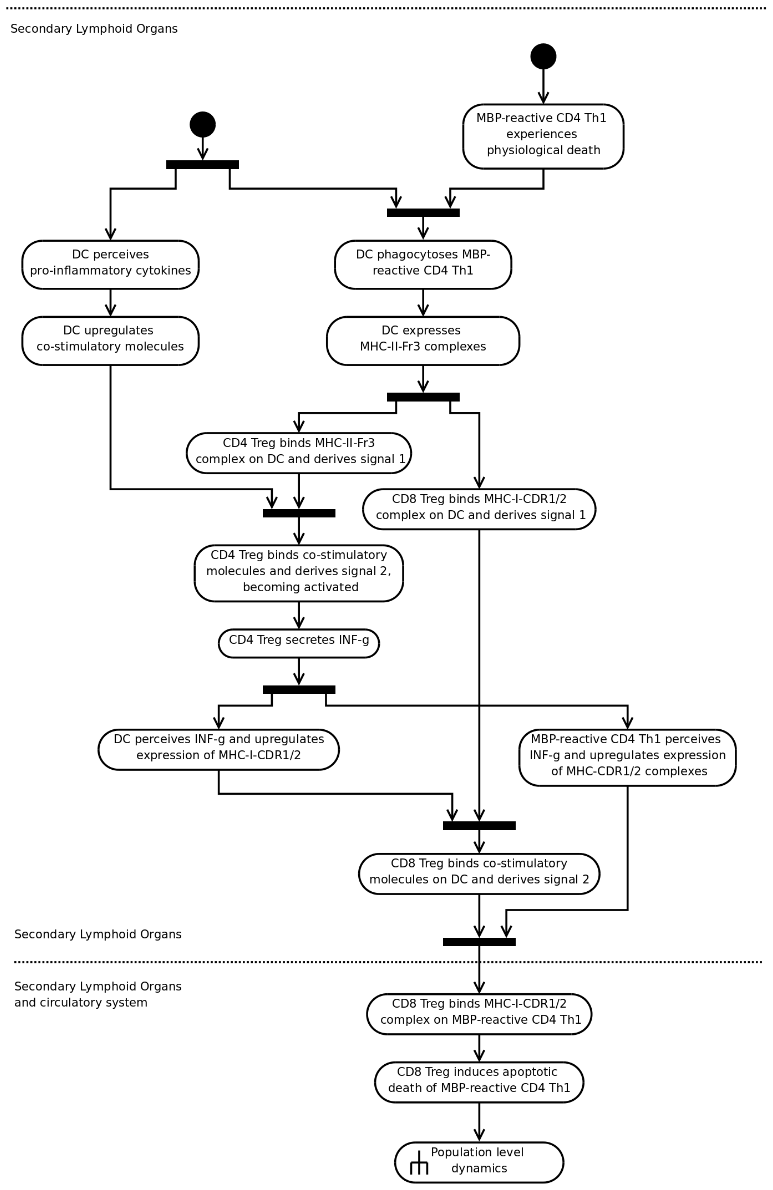
Secondary Lymphoid Organs
MBP-reactive CD4 Th1 experiences physiological death
DC perceives pro-inflammatory cytokines
DC phagocytoses MBP-reactive CD4 Th1
DC upregulates co-stimulatory molecules
DC expresses MHC-II-Fr3 complexes
CD4 Treg binds MHC-II-Fr3 complex on DC and derives signal 1
CD8 Treg binds MHC-I-CDR1/2 complex on DC and derives signal 1
CD4 Treg binds co-stimulatory molecules and derives signal 2, becoming activated
CD4 Treg secretes INF-g
DC perceives INF-g and upregulates expression of MHC-I-CDR1/2
MBP-reactive CD4 Th1 perceives INF-g and upregulates expression of MHC-CDR1/2 complexes
CD8 Treg binds co-stimulatory molecules on DC and derives signal 2
Secondary Lymphoid Organs
Secondary Lymphoid Organs and circulatory system
CD8 Treg binds MHC-I-CDR1/2 complex on MBP-reactive CD4 Th1
CD8 Treg induces apoptotic death of MBP-reactive CD4 Th1
Population level dynamics

Fig. 6. Activity diagram representing the order of low-level inter-cellular events that lead to regulation mediated recovery from EAE. This diagram leads into Figure 7

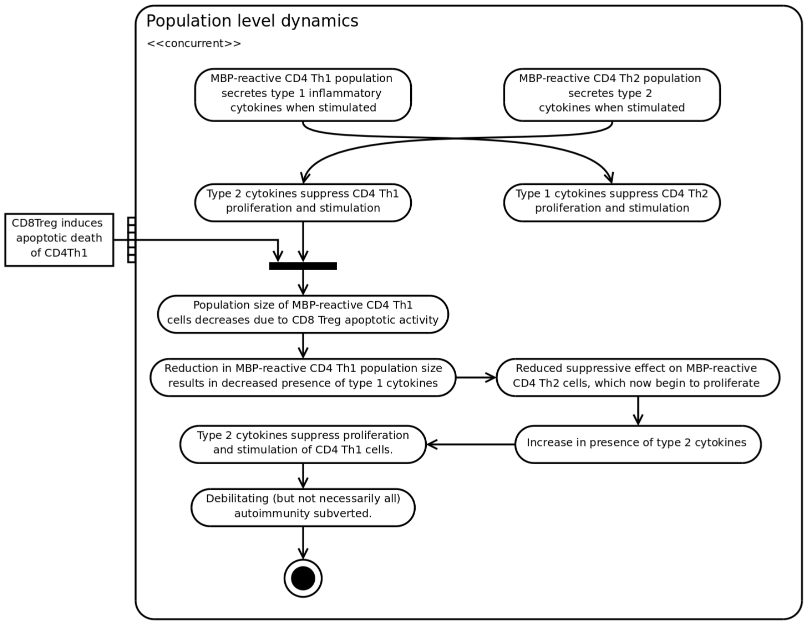
Population level dynamics
<<concurrent>>
MBP-reactive CD4 Th1 population secretes type 1 inflammatory cytokines when stimulated
MBP-reactive CD4 Th2 population secretes type 2 cytokines when stimulated
Type 2 cytokines suppress CD4 Th1 proliferation and stimulation
Type 1 cytokines suppress CD4 Th2 proliferation and stimulation
CD8Treg induces apoptotic death of CD4Th1
Population size of MBP-reactive CD4 Th1 cells decreases due to CD8 Treg apoptotic activity
Reduction in MBP-reactive CD4 Th1 population size results in decreased presence of type 1 cytokines
Reduced suppressive effect on MBP-reactive CD4 Th2 cells, which now begin to proliferate
Increase in presence of type 2 cytokines
Type 2 cytokines suppress proliferation and stimulation of CD4 Th1 cells.
Debilitating (but not necessarily all) autoimmunity subverted.

Fig. 7. Activity diagram representing regulation mediated recovery from EAE. This diagram follows from Figure 6.

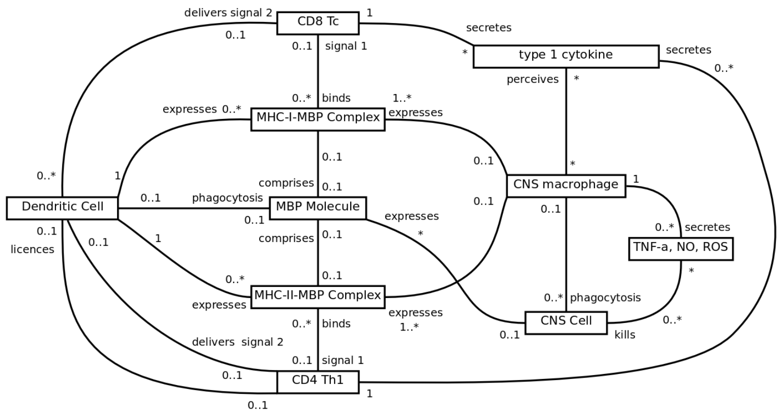

Fig. 8. Class diagram depicting the entities responsible for EAE.

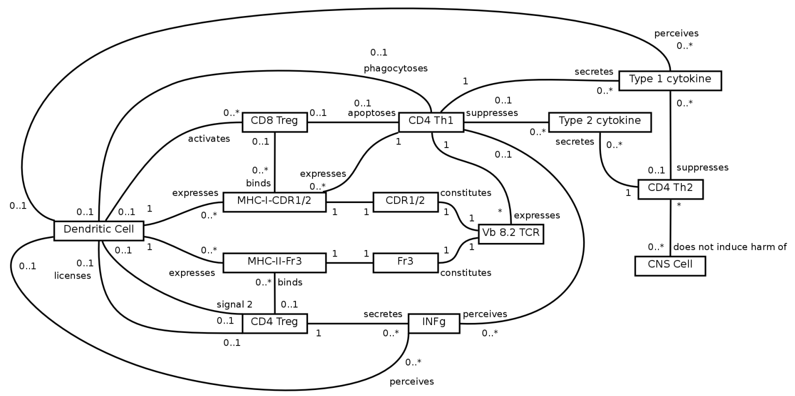

Fig. 9. Class diagram depicting the entities responsible for regulation mediated recovery from EAE.

is unconventional to express only one state in this manner, but the approach has been useful in depicting certain activities of cells.

It has proven useful to construct state machine diagrams of entities that do not necessarily carry state. For example, Figures 19 and 18 respectively show the locations in which MBP molecules may reside, and the effects that INF-γ has on cells the perceive them. These aspects do not necessarily comprise internal states of the molecules portrayed in the state machine diagrams. However, since these system elements mediate the actions of other elements that do carry state, then it can be informative to depict the system from their perspective.

Several of the state machine diagrams in this model depict the physical locations in the model where a cell may reside. It can be argued that a cell's physical location is not part of its internal state, but depicting it in this manner is informative. A similar approach has been used by [9].

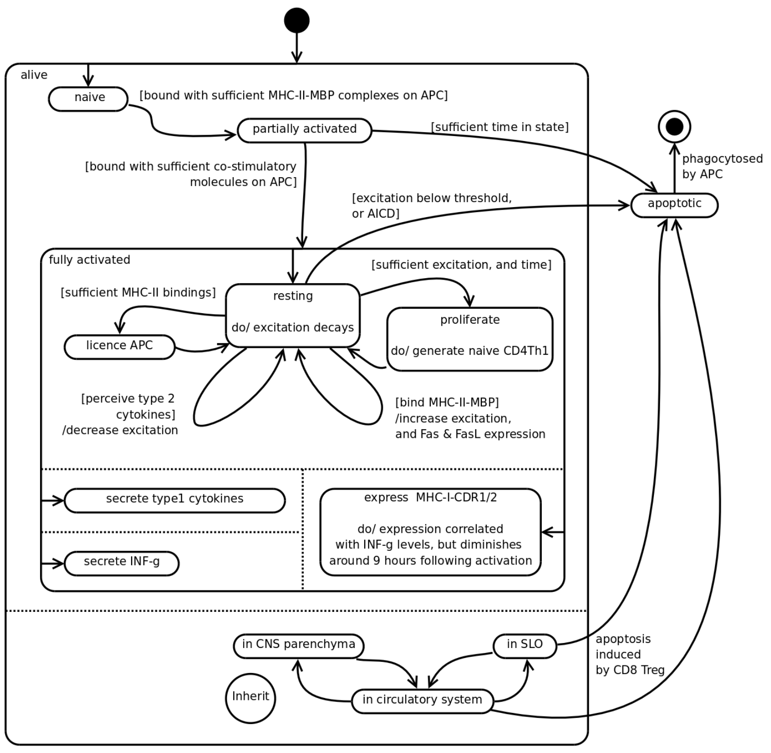
alive
naive
[bound with sufficient MHC-II-MBP complexes on APC]
partially activated
[sufficient time in state]
[bound with sufficient co-stimulatory molecules on APC]
[excitation below threshold, or AICD]
phagocytosed by APC
apoptotic
fully activated
[sufficient excitation, and time]
[sufficient MHC-II bindings]
resting
do/ excitation decays
proliferate
do/ generate naive CD4Th1
licence APC
[perceive type 2 cytokines] /decrease excitation
[bind MHC-II-MBP] /increase excitation, and Fas & FasL expression
secrete type1 cytokines
express MHC-I-CDR1/2
do/ expression correlated with INF-g levels, but diminishes around 9 hours following activation
secrete INF-g
in CNS parenchyma
in SLO
apoptosis induced by CD8 Treg
Inherit
in circulatory system

Fig. 10. State machine diagram of a CD4 Th1 cell.

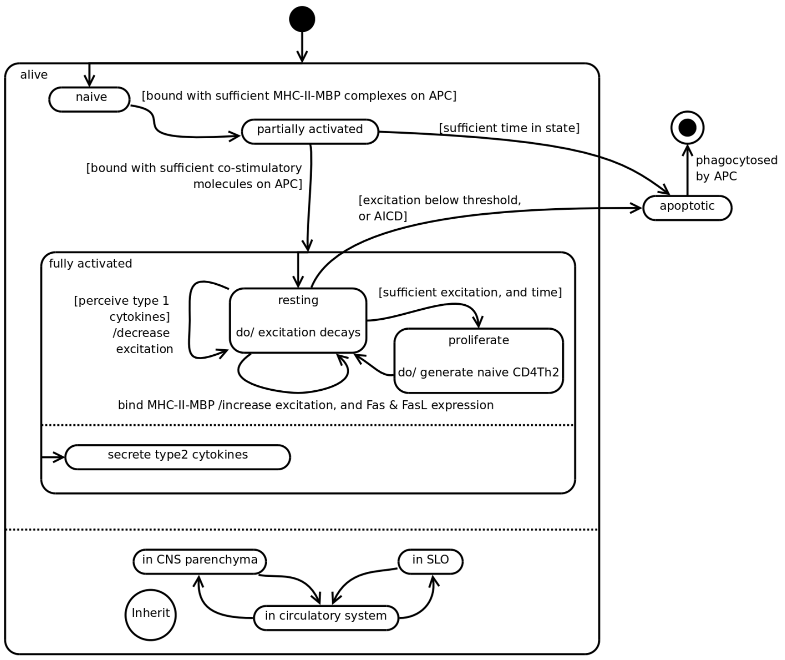
alive
naive
[bound with sufficient MHC-II-MBP complexes on APC]
partially activated
[sufficient time in state]
phagocytosed by APC
[bound with sufficient co-stimulatory molecules on APC]
[excitation below threshold, or AICD]
apoptotic
fully activated
[perceive type 1 cytokines] /decrease excitation
resting
do/ excitation decays
[sufficient excitation, and time]
proliferate
do/ generate naive CD4Th2
bind MHC-II-MBP /increase excitation, and Fas & FasL expression
secrete type2 cytokines
in CNS parenchyma
in SLO
Inherit
in circulatory system

Fig. 11. State machine diagram of a CD4 Th2 cell.

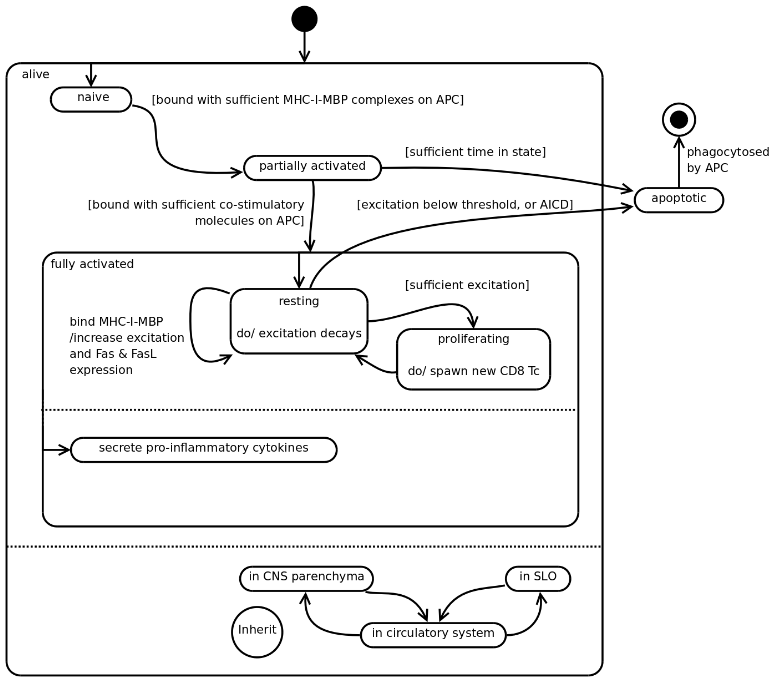

Fig. 12. State machine diagram of a CD8 Tc cell.

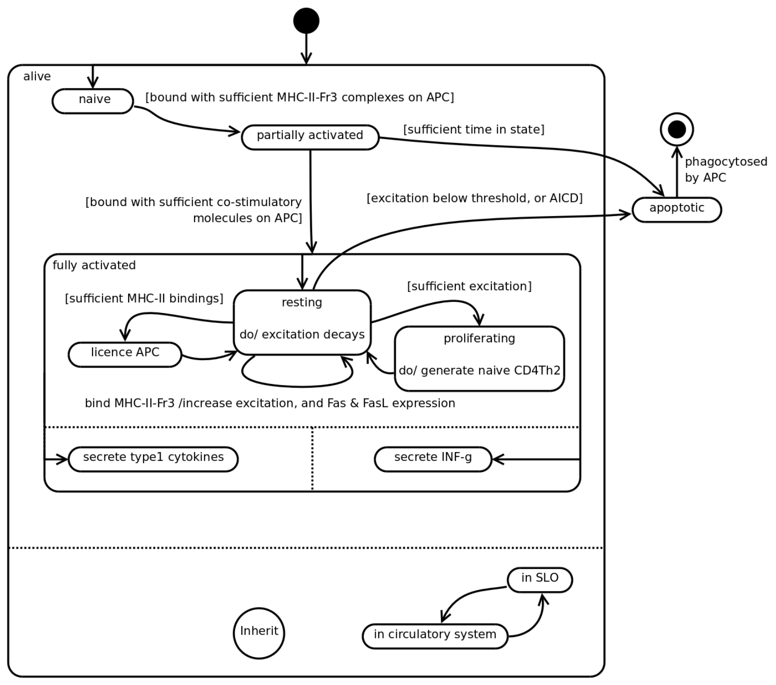
alive
naive
[bound with sufficient MHC-II-Fr3 complexes on APC]
partially activated
[sufficient time in state]
phagocytosed by APC
apoptotic
[bound with sufficient co-stimulatory molecules on APC]
[excitation below threshold, or AICD]
fully activated
[sufficient MHC-II bindings]
resting
do/ excitation decays
[sufficient excitation]
proliferating
do/ generate naive CD4Th2
licence APC
bind MHC-II-Fr3 /increase excitation, and Fas & FasL expression
secrete type1 cytokines
secrete INF-g
in SLO
Inherit
in circulatory system

Fig. 13. State machine diagram of a CD4 Treg cell.

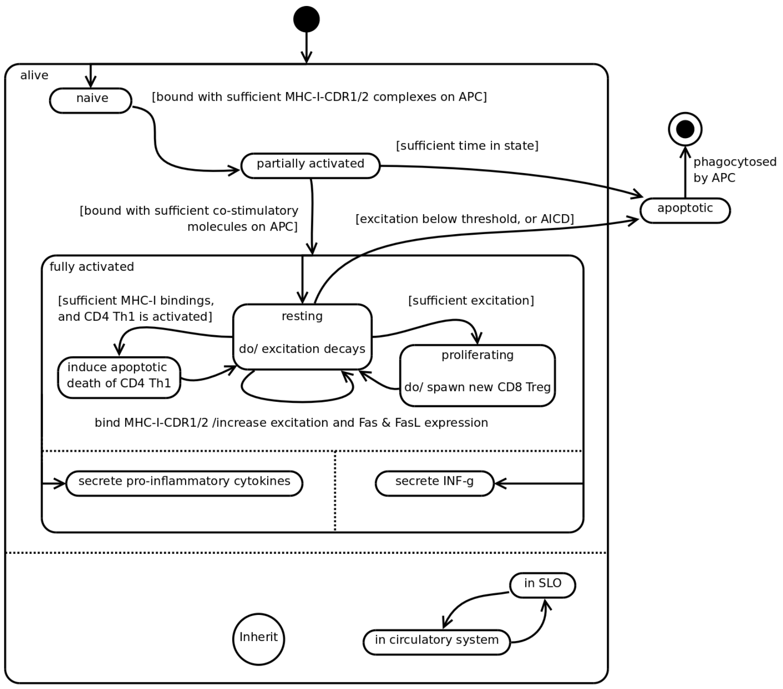
alive
naive
[bound with sufficient MHC-I-CDR1/2 complexes on APC]
partially activated
[sufficient time in state]
phagocytosed by APC
apoptotic
[bound with sufficient co-stimulatory molecules on APC]
[excitation below threshold, or AICD]
fully activated
[sufficient MHC-I bindings, and CD4 Th1 is activated]
resting
do/ excitation decays
[sufficient excitation]
proliferating
do/ spawn new CD8 Treg
induce apoptotic death of CD4 Th1
bind MHC-I-CDR1/2 /increase excitation and Fas & FasL expression
secrete pro-inflammatory cytokines
secrete INF-g
in SLO
Inherit
in circulatory system

Fig. 14. State machine diagram of a CD8 Treg cell.

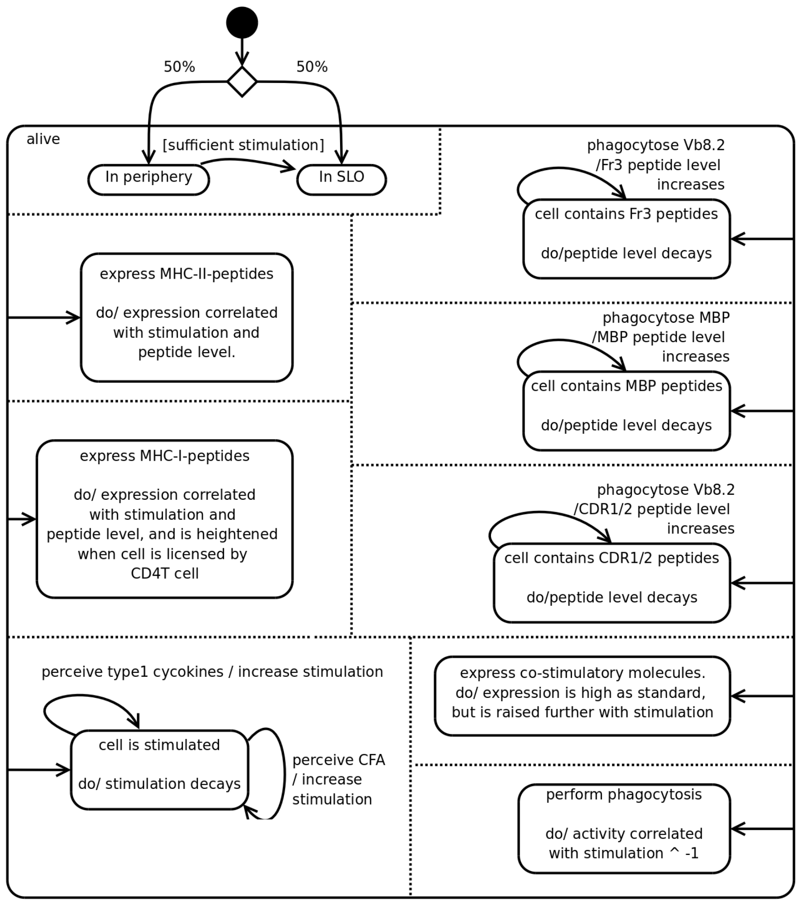
50%
50%
alive
[sufficient stimulation]
In periphery
In SLO
phagocytose Vb8.2 /Fr3 peptide level increases
cell contains Fr3 peptides
do/peptide level decays
express MHC-II-peptides
do/ expression correlated with stimulation and peptide level.
phagocytose MBP /MBP peptide level increases
cell contains MBP peptides
do/peptide level decays
express MHC-I-peptides
do/ expression correlated with stimulation and peptide level, and is heightened when cell is licensed by CD4T cell
phagocytose Vb8.2 /CDR1/2 peptide level increases
cell contains CDR1/2 peptides
do/peptide level decays
perceive type1 cycokines / increase stimulation
express co-stimulatory molecules.
do/ expression is high as standard, but is raised further with stimulation
cell is stimulated
do/ stimulation decays
perceive CFA / increase stimulation
perform phagocytosis
do/ activity correlated with stimulation ^ -1

Fig. 15. State machine diagram of a dendritic cell.

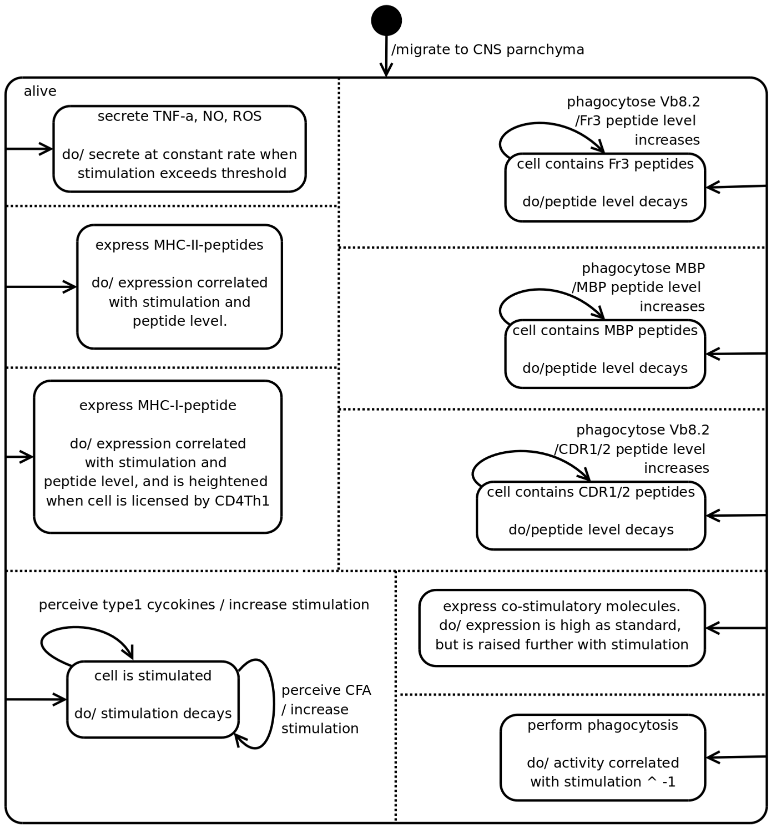
/migrate to CNS parnchyma
alive
secrete TNF-a, NO, ROS
do/ secrete at constant rate when stimulation exceeds threshold
express MHC-II-peptides
do/ expression correlated with stimulation and peptide level.
express MHC-I-peptide
do/ expression correlated with stimulation and peptide level, and is heightened when cell is licensed by CD4Th1
phagocytose Vb8.2 /Fr3 peptide level increases
cell contains Fr3 peptides
do/peptide level decays
phagocytose MBP /MBP peptide level increases
cell contains MBP peptides
do/peptide level decays
phagocytose Vb8.2 /CDR1/2 peptide level increases
cell contains CDR1/2 peptides
do/peptide level decays
perceive type1 cycokines / increase stimulation
cell is stimulated
do/ stimulation decays
perceive CFA / increase stimulation
express co-stimulatory molecules.
do/ expression is high as standard, but is raised further with stimulation
perform phagocytosis
do/ activity correlated with stimulation ^ -1

Fig. 16. State machine diagram of a CNS macrophage.

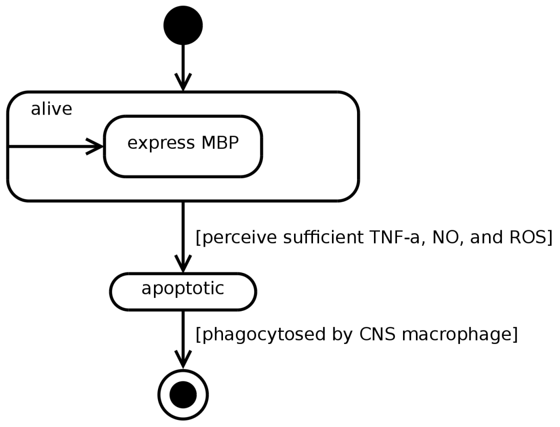
alive
express MBP
[perceive sufficient TNF-a, NO, and ROS]
apoptotic
[phagocytosed by CNS macrophage]

Fig. 17. State machine diagram of a CNS cell.

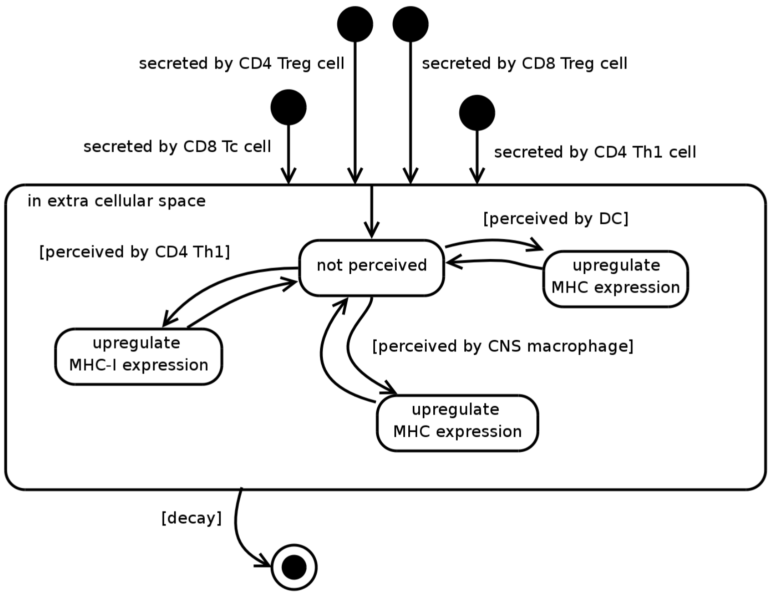
secreted by CD4 Treg cell
secreted by CD8 Treg cell
secreted by CD8 Tc cell
secreted by CD4 Th1 cell
in extra cellular space
[perceived by DC]
[perceived by CD4 Th1]
not perceived
upregulate MHC expression
upregulate MHC-I expression
[perceived by CNS macrophage]
upregulate MHC expression
[decay]

Fig. 18. State machine diagram of INF-γ, a cytokine.

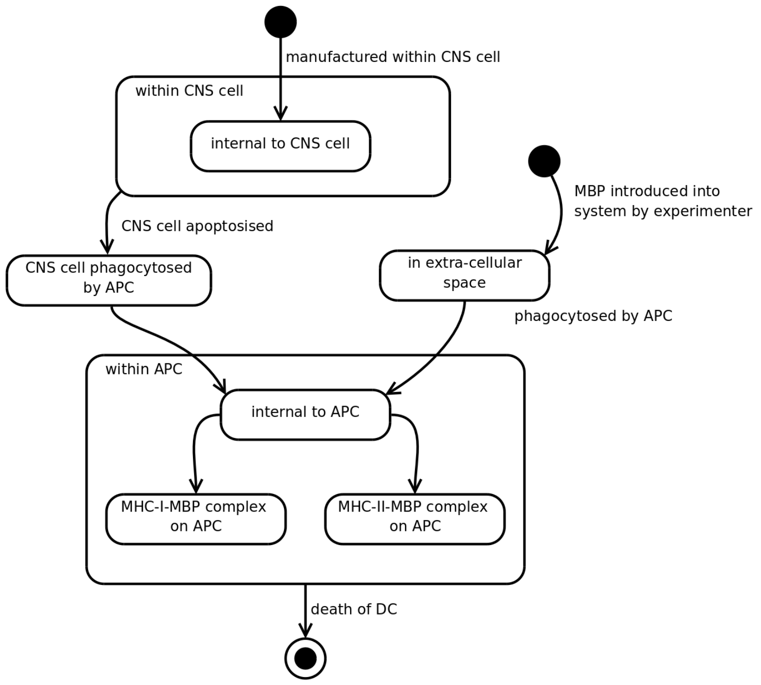

Fig. 19. State machine diagram of MBP.

5.5 Representing feedback

Activity diagrams can demonstrate the order in which critical interactions and events must take place for a high level behaviour to manifest. However they (incorrectly) imply that one activity stops and another starts. In reality the entity responsible for a preceding activity does not hand off control to that which follows, it continues and can potentially perform the same activity again. This concurrency amongst system elements can manifest in feedback, where an increasing number of elements engage in some activity.

To give two examples in the context of Figure 5, a fully activated CD4Th1 will not stimulate a single CNS macrophage and then stop, it will repeat this process. Likewise, the death of a CNS cell (through the secretion of TNF-α, ROS, and NO by a CNS macrophage) will lead to its phagocytosis by a CNS Macrophage, which then presents MBP to additional naive CD4Th1 cells, facilitating their activation and accelerating the progress of EAE. This latter feedback can further amplify the effect of the former. Relative population dynamics play a significant role in EAE (for example, consider the interplay between CD4Th1 and CD4Th2 cells) and it is important to communicate this information in the domain model. Depicting these feedbacks, and others like them, on the activity diagrams will significantly clutter it; as yet we have found no mechanism within UML to satisfactorily express these feedbacks and the interplay between them.

6 Conclusion

We believe that the modelling of a complex system is a necessary precursor to the implementation of a simulation of that complex system. It is necessary to demonstrate a detailed understanding of the complex system of interest and have this validated by a domain expert. If one's understanding of the system is incorrect then the simulation will not be representative of the real world system; uncovering such errors can be difficult and time consuming. A model of a complex system can serve as a specification for a simulation, and its validation by a domain expert can deliver some measure of confidence in the simulation's own validity.

We have presented here a domain model of experimental autoimmune encephalomyelitis (EAE), a complex autoimmune disease in mice, and its regulatory T cell mediated recovery. The models are expressed using UML, and the creation of the present domain model has afforded insights into UML's expressive capabilities when applied to complex systems.

Complex systems tend to exhibit many interactions between entities within the system. Attempting to capture all this interaction in one diagram that fully describes the system renders the diagram cluttered and illegible. Since a primary purpose of creating a domain model is to communicate one's understanding of a complex system a balance must be struck between fully specifying the system where ever possible and maintaining informative diagrams. We have found it useful to identify the scenarios in EAE, being autoimmunity and recovery, and depict these separately using activity diagrams. Class diagrams of the entities that partake in a scenario have been constructed, but we have found their contribution to the model to be minor; diagrams that depict system dynamics and the order in which events that comprise a scenario take place have been more relevant than a static depiction of all the interactions that are possible. UML state machine diagrams of system entities that don't themselves carry state or instigate interactions, but do mediate interactions between other entities have been informative.

The dynamics of EAE are heavily dependent on the interplay between cell populations and feedback mechanisms. We have found no satisfactory method to use UML in expressing these aspects of the system. Biological cells incorporate many features that are subject to continuous domains, such as variable levels of stimulation or molecule expression. These aspects cannot be expressed through state transitions alone, and require either textual explanation or use of equations to specify. UML encompasses several mechanisms that have proven useful in modelling EAE; however, in its current form, UML alone is insufficient to fully specify the system.

The CoSMoS minimal process, the principled approach to complex system simulation development that we are following, is iterative. As we explore EAE through simulation and investigate alternative hypotheses concerning its operation our domain model may require amendment. Should we wish to investigate the effect that a cell previously not represented in the simulation has on EAE our domain model will be modified to reflect its incorporation into the system. Validation of the domain model by a domain expert is intended to provide some measure of confidence in the results of experimentation with a simulation. The model must be maintained to reflect the experiments we conduct, and changes must be validated.

The next stage in our work is to refactor the domain model into an implementation specific simulation model. The simulation model will form the specification for the construction of a simulation, and will be used to conduct *in silico* experimentation with the intention of gaining insights into EAE's nature and operation.

7 Acknowledgements

The work of Paul Andrews is supported by the CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1.

References

[1] Complex Systems Modelling and Simulation Infrastructure (CoSMoS) project homepage. *http://www.cosmos-research.org/*.

[2] Paul S. Andrews, Fiona Polack, Adam T. Sampson, Jon Timmis, Lisa Scott, and Mark Coles. Simulating biology: towards understanding what the simulation shows. In Stepney et al. [21], pages 93–124.

[3] Paul S. Andrews, Adam T. Sampson, John Markus Bjrndalen, Susan Stepney, Jon Timmis, Douglas N. Warren, and Peter H. Welch. Investigating patterns for the process-oriented modelling and simulation of space in complex systems. In *ALife XI, Winchester, UK, September 2008*, pages 17–24. MIT Press, 2008.

[4] Paul S. Andrews, Adam T. Sampson, Fiona Polack, Susan Stepney, and Jon Timmis. Cosmos development lifecycle, version 0 (in preparation). Technical report, The University of York, 2008.

[5] Etty N. Benveniste. Role of macrophages/microglia in multiple sclerosis and experimental allergic encephalomyelitis. *Journal of Molecular Medicine*, 75(3):165–173, March 1997.

[6] Hugues Bersini. Immune system modeling: The OO way. In *ICARIS*, pages 150–163, 2006.

[7] Sol Efroni, David Harel, and Irun R. Cohen. Toward rigorous comprehension of biological complexity: Modeling, execution, and visualization of thymic T-cell maturation. *Genome Research*, 13:2485–2497, 2003.

[8] Martin Fowler. *UML Distilled*. Addison-Wesley, 3rd edition, 2004.

[9] Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an Executable Model of Auxin Transport Canalisation. [21], pages 63–91.

[10] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[11] Jerome J. A. Hendriks, Charlotte E. Teunissen, Helga E. de Vries, and Christine D. Dijkstra. Macrophages and neurodegeneration. *Brain Research Reviews*, 48(2):185–195, April 2005.

[12] Na'aman Kam, Irun R. Cohen, and David Harel. The Immune System as a Reactive System: Modeling T Cell Activation with Statecharts. In *Proceedings of Visual Languages and Formal Methods (VLFM'01), part of IEEE Symposium on Human-Centric Computing*, pages 15–22, 2001.

[13] Thomas J. Kindt, Richard A. Goldsby, and Barbara A. Osbourne. *Kuby Immunology*. W. H. Freeman and Company, 6th edition, 2007.

[14] Vipin Kumar. Homeostatic control of immunity by TCR peptide-specific Tregs. *The Journal of Clinical Investigation*, 114(9):1222–1226, November 2004.

[15] Vipin Kumar and Eli Sercarz. An integrative model of regulation centered on recognition of TCR peptide/MHC complexes. *Immunological Reviews*, 182:113–121, 2001.

[16] Loui Thomas Madakamutil, Igor Maricic, Eli E. Sercarz, and Vipin Kumar. Immunodominance in the TCR repertoire of a TCR peptide-specific CD4+ Treg population that controls experimental autoimmune encephalomyelitis. *The Journal of Immunology*, 180(1):4577–4585, April 2008.

[17] Object Management Group. Maintainer of the unified modelling language standards. *http://www.uml.org.*

[18] Gennadij Raivich and Richard Banati. Brain microglia and blood-derived macrophages: molecular profiles and functional roles in multiple sclerosis and animal models of autoimmune demyelinating disease. *Brain Research Reviews*, 48(3):261–281, November 2004.

[19] Mark Read, Paul S. Andrews, and Jon Timmis. Using UML to Model EAE and its Regulatory Network. To appear in ICARIS '09, 2009.

[20] Avital Sadot, Jasmin Fisher, Dan Barak, Yishai Admanit, Michael J. Stern, E. Jan Albert Hubbard, and David Harel. Toward Verified Biological Models. *IEEE/ACM transactions on Computational Biology and Bioinformatics*, 5(2):223–234, April-June 2008.

[21] Susan Stepney, Fiona Polack, and Peter Welch, editors. *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation, York, UK, September 2008.* Luniver Press, 2008.

[22] Bart R. Tambuyzer, Peter Ponsaerts, and Etienne J. Nouwen. Microglia: gatekeepers of central nervous system immunology. Journal of Leukocyte Biology, published online. doi:jlb.0608385, November 2008.

[23] Xiaolei Tang, Igor Maricic, Nikunj Purohit, Berge Bakamjian, Lisa M Read-Loisel, Tara Beeston, Peter Jensen, and Vipin Kumar. Regulation of immunity by a novel population of Qa-1-restricted CD8$\alpha\alpha^+$tcr$\alpha\beta^+$ T cells. *The Journal of Immunology*, 117:7645–7655, 2006.

A Domain Model Assumptions

This appendix details the assumptions that have been made in creating the present domain model of EAE. Assumptions are labelled with the cell or phenomenon that they correspond to, and are numbered.

CD4Th1-1. All CD4Th1 cells considered in this domain model are specific for MHC-II-MBP complexes only, though their individual affinities for the complex may vary. An implication of this assumption is that the spatial/binding events brought about by Th cells of other specificities are absent.

CD4Th2-1. All CD4Th2 cells considered in this domain model are specific for MHC-II-MBP complexes only, though their individual affinities for the complex may vary.

CD4Th2-2. CD4Th2 cells do not license APCs in this model, and they do not provide "help" to any cell (the model contains no B cells).

CD4Treg-1. All CD4Treg cells considered in this domain model are specific for MHC-II-Fr3 complexes only, though their individual affinities for the complex may vary.

CD8Tc-1. All CD8 Tc cells considered in this domain model are specific for MHC-I-MBP complexes only, though their individual affinities for the complex may very.

CD8Tc-2. A single CD8 Tc cell can induce apoptosis in at most one CNS cell at any specific point in time.

CD8Treg-1. All CD8 Treg cells considered in this domain model are specific for MHC-I-CDR1/2 complexes only, though their individual affinities for the complex may vary.

CD8Treg-2. A CD8 Treg can induce the apoptotic death of at most one CD4Th1 cell at any point in time.

CD8Treg-3. CD8Treg cells as represented in this model cannot induce apoptosis in APCs that express MHC-I-CDR1/2, although this is potentially possible *in vivo*.

TCell-1. Cytokine secretion by a T cell is assumed to have only two rates of secretion: none at all, or a steady rate of secretion. There is no notion of variable secretion based on a cell's stimulation, or any other effect.

TCell-2. Activated T cells have associated with them an "excitation" level. This is an abstraction of the cell's internal metabolic activity.

TCell-3. Signal 1 and 2, delivered through bindings with MHC and co-stimula-tory molecules, can only be derived through *simultaneously* binding sufficient such molecules; there is no notion of how many molecules were "recently" bound. *In vivo* this may not necessarily be the case.

TCell-4. Where applicable, a CD4 T cell cannot simultaneously license an APC and proliferate.

Cell-1. A cell can interact with at most one other cell at any point in time. For example, a T cell can bind with molecules expressed on only one APC at a time. *In vivo* these aspects are dictated by the physical space surrounding a cell, and what occupies that space.

CNS-1. The "CNS Cell" of this domain model is an abstract represention of the various MBP-expressing cells of the *in vivo* central nervous system.

CNS-2. An apoptotic CNS cell can be phagocytosed by only a single dendritic cell.

CNS-3. CNS cells do not reproduce/divide.

CNS-4. CNS cells do not incur natural death.

CNS-5. Upon phagocytosis by an APC only MBP is received by that APC. No other molecules that might stimulate the APC are derived from phagocytosis of a CNS cell.

CNSMacrophage-1. CNS Macrophage is an abstraction of microglia and macrophages that reside within the CNS during EAE. A study of the literature has revealed that there is currently no consensus from which the functions of macrophages and microglia can be distinguished within the context of EAE [5, 11, 18, 22]

CNSMacrophage-2. Secretion of TNF-α, ROS, and NO by CNS macrophages is at a constant rate, and occurs only when the cell is heavily stimulated.

CNSMacrophage-3. CNS macrophages in this model do not secrete any cytokines, other than TNF-γ.

Cytokine-1. This domain model represents all type 1 and pro-inflammatory cytokines as one cytokine abstraction, called "type1". Where a specific cytokine (for example INF-γ) exhibits some function that is not well represented by this abstraction, that specific cytokine is explicitly represented, but only to serve the concerned function.

Cytokine-2. The model represents all type 2 cytokines as one cytokine abstraction, called "type2".

Cytokine-3. Despite being a pro-inflammatory type 1 cytokine, INF-γ is not depicted in this model to suppress CD4Th2 cell activity, since that is already handled by assumption 1.

DC-1. A dendritic cell can provide signal 2 to only a single T cell at a time.

DC-2. A dendritic cell in this domain model will never die, though its expression of MHC-peptide levels is variable.

DC-3. Dendritic cells in this model do not secrete any cytokines.

MHC-1. The only MHC-peptide complexes considered in this domain model are: MHC-I-MBP; MHC-II-MBP; MHC-I-CDR1/2; MHC-II-Fr3. No other MHC-peptide complexes are considered integral to EAE or its recovery.

Co-stimulatory-1. CD4Th1, CD4Th2, and CD8Tc cells all require equal quantities of co-stimulatory molecule bindings to derive signal 2.

Apoptosis-1. We have omitted any notion of an “anergy” state, since anergic cells can be rescued though receipt of signal 2. T cells that spend sufficient time in a “partially activated” state will become apoptotic.

Apoptosis-2. *In vivo*, interaction between an anergic T cell and an APC can have a regulatory effect on the APC. This is not represented in this domain model.

On Non-Standard Numerical Integration Methods for Biological Oscillators

Andrew Hone⋆

Isaac Newton Institute for Mathematical Sciences,
20 Clarkson Road, Cambridge CB3 0EH, UK.
anwh@kent.ac.uk

Abstract. Mathematical models of biological processes, whether they be the dynamics of populations of individuals, cell populations within an organism, or reactions between different chemical species within a cell, are traditionally formulated in terms of differential equations. For a given model, specified by a system of differential equations, in the generic situation one is unable to solve the equations explicitly, and one must resort to numerical integration methods. There are many standard numerical techniques, both in the literature and in readily available software packages. However, these techniques do not always reproduce the required qualitative features of the solutions, particularly if the system shows regular oscillations and the integration is performed over long time periods. This article highlights some non-standard discretization methods appearing in the work of Kahan, Mickens and others, which have the potential to be extremely useful in modelling biological systems.

1 Introduction

The purpose of this article is to draw attention to some interesting developments in numerical analysis, and more specifically in numerical integration methods for ordinary differential equations, which have taken place over the last twenty years or so, but have only started to filter into the wider scientific community more recently. There is a vast literature on numerical integration, and the standard methods for integrating ordinary differential equations, namely Euler's method, Runge-Kutta methods and their adaptive versions (Runge-Kutta-Fehlberg), and multi-step

⋆ On leave from Institute of Mathematics, Statistics & Actuarial Science, University of Kent, Canterbury CT2 7NF, UK.

methods (Adams-Bashforth-Moulton) are described in many introductory texts on differential equations (e.g. [32]) and have been implemented in numerous software packages that are both highly sophisticated and readily available. While the aforementioned techniques are rather successful at dealing with generic differential equations, they can often come unstuck for some particular problems of interest that arise in applications, particularly when such problems exhibit special properties such as symmetries or conservation laws, or when there are solutions with some special structure. Indeed, standard numerical integrators can completely fail to capture the correct qualitative nature of the solutions of such problems, and can be extremely inefficient at providing accurate results. In contrast, for numerical integration of differential equations that preserve a measure or symplectic structure [31], and those that have conserved quantities [29] or are even completely integrable [30], there are a variety of *ad hoc* methods that give more accurate and qualitatively correct solutions with greater efficiency, because they retain the same features as the original differential equations. A general approach to numerical integration based on incorporating the symmetry structure of a problem into a numerical scheme is known as geometric integration [4]. For some applications, such as molecular dynamics, or the N-body problem in celestial mechanics, which has the inverse square force law, it can be very important to conserve energy, but this poses serious technical difficulties for numerical integration (see [16] and references).

While symmetries and conservation laws are common in physics, and especially in mechanics, most differential equation models of biological systems do not exhibit any particular symmetry. Nevertheless, these biological models can have particular special solutions (which, most importantly, are often attractors in the space of all solutions) whose features are missed by conventional numerical integrators. For such systems, some relevant techniques for constructing non-standard or unconventional integration methods has been developed in various works by Mickens, Kahan and other researchers. It is the latter set of techniques that are outlined here.

This paper is a contribution to the CoSMoS workshop, which is primarily concerned with agent-based simulations of real-life processes. When considering the interactions of chemical or biological species with a small number of molecules or agents, the use of continuous variables is no longer relevant, as finite-size and stochastic effects become important. In the latter context, ordinary differential equations cannot provide successful models. Indeed, the differential equations considered here are very trivial from this point of view. The Gillespie algorithm is the prototype for an agent-based stochastic model of reaction kinetics, and this stochastic model converges to the solution of the corresponding ordinary differential equation model in the limit when the number of molecules goes to infinity [8]. In all such models, whether they be deterministic or stochastic, continuous or discrete, there are a number of parameters. Some parameters, such as reaction rates (or the corresponding reaction probabilities), have to be determined experimentally for the case of well diluted mixtures of reagents with high concentrations (i.e. in the experimental situation where the continuous differential equations *do* provide a good model of the system). However, there can be other parameters for which it is very difficult to provide an experimental measurement, so that their values can only be inferred or fitted indirectly based on simulations. In the latter case, a continuous model in terms of differential equations can provide a useful independent method to validate or approximate the range of parameters to be used in an agent-based stochastic model. All simulations, whether deterministic or stochastic, are only attempts to model reality, and which type of model is more realistic depends on the context. However, this paper deals exclusively with the problem of accurately determining the behaviour of solutions to differential equations by numerical integration. The question of whether all these solutions correspond to aspects of the real chemical or biological system being modelled is not addressed here.

To understand some of the problems that can arise with numerical integration schemes, it is instructive to consider a very simple example, namely the ordinary differential equation describing the logistic growth of a population:

$$\dot{r} = \rho r(1 - r/K). \tag{1}$$

In the above, $r = r(t)$ denotes the size of a population with linear growth rate $\rho > 0$, $\dot{r} = dr/dt$ is the rate of change of the population, and $K > 0$ is the carrying capacity, which is the stable limiting population size as $t \to \infty$. If one performs a numerical integration of this system using the simplest technique available, namely the forward Euler method, then one obtains the logistic difference equation

$$\frac{r_{t+h} - r_t}{h} = \rho r_t(1 - r_t/K), \tag{2}$$

where $r_t \approx r(t)$ denotes the approximation to the solution of the ordinary differential equation at time t. The important thing to notice about (2) is that, while it gives reasonable approximations to the solutions of (1) for small enough h, it also displays behaviour not present in the original model: as h increases the steady state K becomes unstable and goes through an infinite series of period-doubling bifurcations leading to chaos [7]. Although the Euler method is the crudest possible technique for numerical integration, this example illustrates the point that discretization can introduce properties of the solutions that are not present in the original continuous system.

In the next section, to set the scene, Kahan's unconventional discretization [26] of the classic Lotka-Volterra predator-prey model is presented, and the solutions of this discrete model are compared with the results of solving the corresponding ordinary differential equation using a standard numerical package, namely Maple, implementing the Fehlberg fourth-fifth order Runge-Kutta (RKF45) method. Kahan's general technique for discretizing quadratic vector fields is described, and another non-standard discretization of the same system, due to Mickens, is also mentioned for comparison. The third section is concerned with a model of a trimolecular reaction that exhibits a Hopf bifurcation and a limit cycle (for a suitable range of parameters). By following the guidelines for discretization due to Mickens, a discrete version of this trimolecular reaction model is obtained, and the properties of its solutions are compared with the continuous case. The final section contains some conclusions.

2 Discretization of predator-prey models

The classic Lotka-Volterra model for the interaction between a predator population $P(t)$ and its prey $N(t)$, both specified at time t, has the form

$$\dot{N} = N(a - bP), \qquad \dot{P} = P(cN - d) \tag{3}$$

for positive parameters a, b, c, d. Upon rescaling N, P and t it can be reduced to the dimensionless form

$$\dot{x} = x(1 - y), \qquad \dot{y} = y(x - \alpha), \tag{4}$$

which depends on the single parameter $\alpha > 0$. For the interpretation as a predator-prey model the variables x, y are considered only in the non-negative quadrant $x \geq 0$, $y \geq 0$. There are two steady states of the system: the fixed point (steady state) at the origin $(0, 0)$ is an unstable saddle point, while linearization around the steady state at $(\alpha, 1)$ gives imaginary eigenvalues and hence a neutrally stable centre. (The reader who is unfamiliar with linear stability analysis should consult the first volume of [20], or [12].)

A complex conjugate pair of imaginary eigenvalues of the Jacobian, at a fixed point in the plane, corresponds to periodic orbits around this point in the linearized system, but then the linear analysis is insufficient to determine the nature of the orbits for the original nonlinear system. However, in this particular case it turns out that all the non-trivial orbits for the nonlinear system (4) in the interior of the positive quadrant are actually periodic, lying on closed curves $H = H(x, y) =$ constant encircling the point $(\alpha, 1)$, where

$$H = \log(x^{\alpha} y) - x - y. \tag{5}$$

The shape of twenty of these orbits can be seen in figure 1, in the case $\alpha = 1$. The quantity H is conserved along trajectories, and in fact the system can be written in the non-canonical Hamiltonian form

$$\dot{x} = xy\frac{\partial H}{\partial y}, \qquad \dot{y} = -xy\frac{\partial H}{\partial x}, \tag{6}$$

so that (4) is analogous to a conservative system in classical mechanics, with H corresponding to the "energy" (which is neither created nor destroyed). Moreover, if we introduce new coordinates $X = \log x$, $Y = \log y$ in the positive quadrant $x > 0$, $y > 0$, then one can verify from the equations for $\dot{x}$ and $\dot{y}$ above that the infinitesimal area element $A = \mathrm{d}X\,\mathrm{d}Y$ is preserved by the flow in time t, where $\mathrm{d}X$ and $\mathrm{d}Y$ are the infinitesimal line elements in the X and Y directions respectively. With a slightly more sophisticated approach, one can use the fact that $\mathrm{d}X = \mathrm{d}\,\log x = \frac{1}{x}\mathrm{d}x$ (and similarly for $\mathrm{d}Y$) and show that both the infinitesimal area and the orientation in the plane are preserved by the flow. Mathematically this is encoded into the statement that the oriented area element, or symplectic form, given by

$$\omega = \frac{1}{xy}\mathrm{d}x \wedge \mathrm{d}y, \tag{7}$$

is independent of t; the expression $\mathrm{d}x \wedge \mathrm{d}y$ has a clockwise orientation, and $\mathrm{d}y \wedge \mathrm{d}x = -\mathrm{d}x \wedge \mathrm{d}y$ has an anticlockwise one.

Standard numerical integration schemes reproduce the shape of the orbits of (4) for relatively short times, but a well known problem is that such schemes do not preserve energy, so that over long times the orbits produced numerically either spiral inwards or outwards. To see an example of this, here version 10 of Maple was used to integrate the system for $\alpha = 1$ with the command `dsolve`, which uses the RFK45 method (an adaptive version of the Runge-Kutta method) and the default initial stepsize $h = 10^{-7}$. In order to integrate up to $t = 5000$ with this initial timestep, the RFK45 method will typically generate of the order of 5×10^{10} points, and then the Maple package can extract approximations to the solution at a given sequence of t values. From a plot of part of a single orbit, as in figure 2, one can see that the curve is somewhat thickened compared with the curves obtained in figure 1 (to see this clearly, a zoom in to the same part one of these curves is given in figure 2 for comparison). The reason for this thickening is that the energy H actually

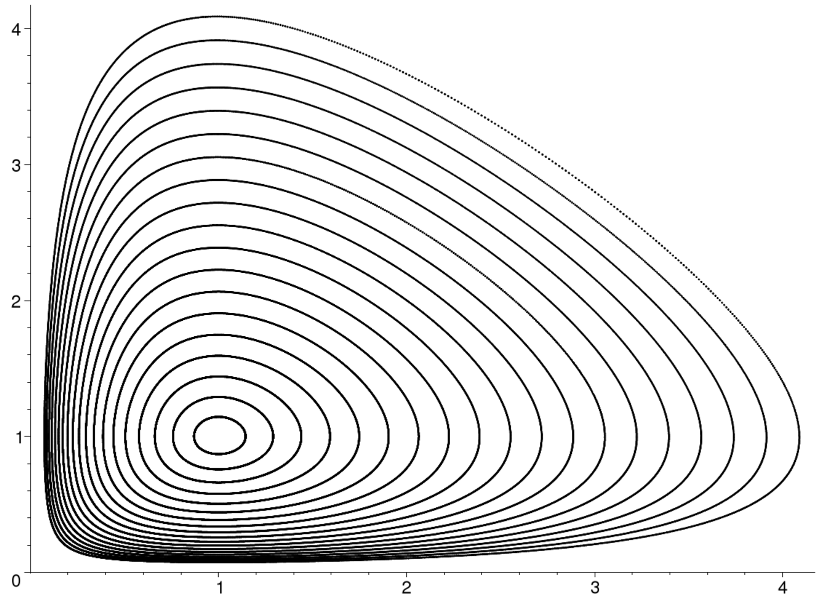

Fig. 1. Twenty orbits of the predator-prey system (4) with $\alpha = 1$, generated with Kahan's discretization (8).

increases monotonically with t when the RFK45 method is applied to the system (4). With $\alpha = 1$ and the initial condition $(x(0), y(0)) = (0.9, 0.8)$, Maple was used to generate 50000 points $\{(x_{0.1t}, y_{0.1t})\}_{t=0}^{50000}$ of the numerical solution, which produced figure 2. Using these values the energy $H_t = H(x_t, y_t)$ was calculated and found to increase linearly with t, as in figure 3, which shows the energy difference $H_t - H_0$, with the initial energy being $H_0 \approx 2.0285041$. (20-digit floating point numbers have been used to perform all numerical calculations with Maple, set with the `Digits` command.) For longer and longer integration times, the numerically calculated orbit spirals outwards, so that the points (x_t, y_t) lie in an increasingly thick band around the actual solution curve $H = H_0$.

To produce the qualitatively more accurate plot of closed orbits shown in figure 1, a discretization of the Lotka-Volterra system due to Kahan was used. This discretization method, which was apparently first presented in some unpublished lectures by Kahan in 1993, is directly applicable to systems of first order ordinary differential equations whose right hand sides are at most quadratic (degree two) in all dependent

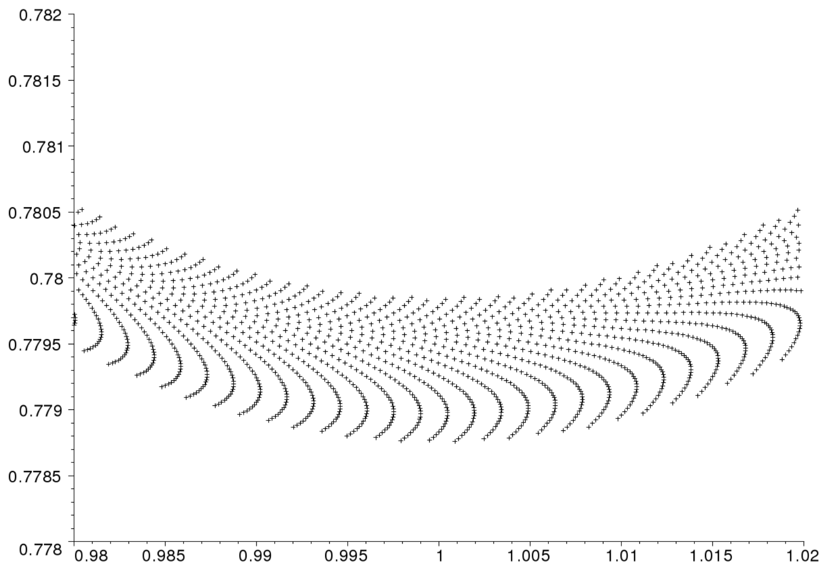

Fig. 2. Part of the orbit from numerical integration of the predator-prey system (4) with $\alpha = 1$ and initial point $(0.9, 0.8)$, using the RKF45 method up to $t = 5000$.

variables. Each derivative term on the left hand side should be replaced by the standard forward difference (as in the Euler method), so that e.g. $\dot{x}$ becomes $(x_{t+h} - x_t)/h \equiv (\tilde{x} - x)/h$, where the tilde is used to denote the forward time shift $t \to t + h$. Each quadratic term is replaced by an average involving products of variables at time t and those shifted forwards, so e.g. xy on both right hand sides of (4) becomes $\frac{1}{2}(\tilde{x}y + x\tilde{y})$; linear terms are also replaced by an average, so x (on the right hand side of the equation for $\dot{x}$) becomes $\frac{1}{2}(\tilde{x} + x)$ in the discrete version. (There are no purely constant terms in (4), but where such terms appear these should be left as they are.) Applying these rules of discretization to (4) with $\alpha = 1$ produces the system of two difference equations given by

$$\frac{\tilde{x} - x}{h} = \frac{1}{2}\Big(\tilde{x} + x - (\tilde{x}y + x\tilde{y})\Big), \quad \frac{\tilde{y} - y}{h} = \frac{1}{2}\Big(\tilde{x}y + x\tilde{y} - (\tilde{y} + y)\Big). \quad (8)$$

This discretizaton can be said to be nonlocal in time, in the sense that the forward difference of each variable (on the left) is given by a function of both the local variables $(x, y) = (x_t, y_t)$ at time t and the forward shifted variables $(\tilde{x}, \tilde{y}) = (x_{t+h}, y_{t+h})$ (on the right).

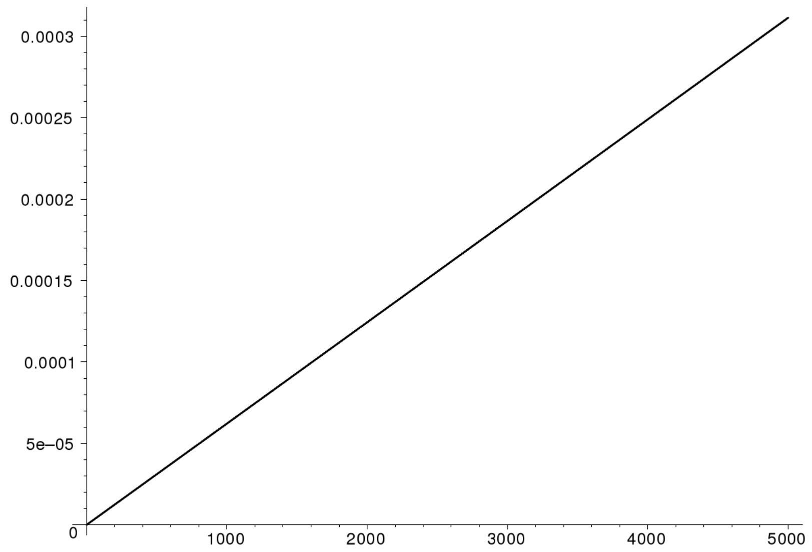

Fig. 3. The energy difference $H_t - H_0$ plotted against t for numerical integration using RKF45 as in figure 2.

The above system clearly reproduces the differential equations (4) in the continuum limit $h \to 0$, and has certain positive features which are immediately obvious, perhaps the first being its inherent simplicity (especially compared with the complexity of Runge-Kutta schemes, for instance). A second important aspect is that the above discretization scheme always has the same fixed points as the original differential equation (these being $(0,0)$ and $(1,1)$ in the case at hand) and does not introduce any extra ones. Thirdly, the two equations (8) are both polynomial in $x, y, \tilde{x}, \tilde{y}$, and the system is linear in both pairs of variables $\tilde{x}, \tilde{y}$ and x, y, which means that it can be solved to give the updated $\tilde{x}, \tilde{y}$ as explicit rational functions of x, y, and (vice-versa) the inverse transformation is also given by explicit rational functions, so it constitutes a birational map of the (x, y) plane (also known as a Cremona transformation; see e.g. [3] and references). Upon solving (8) for $\tilde{x}, \tilde{y}$ the following

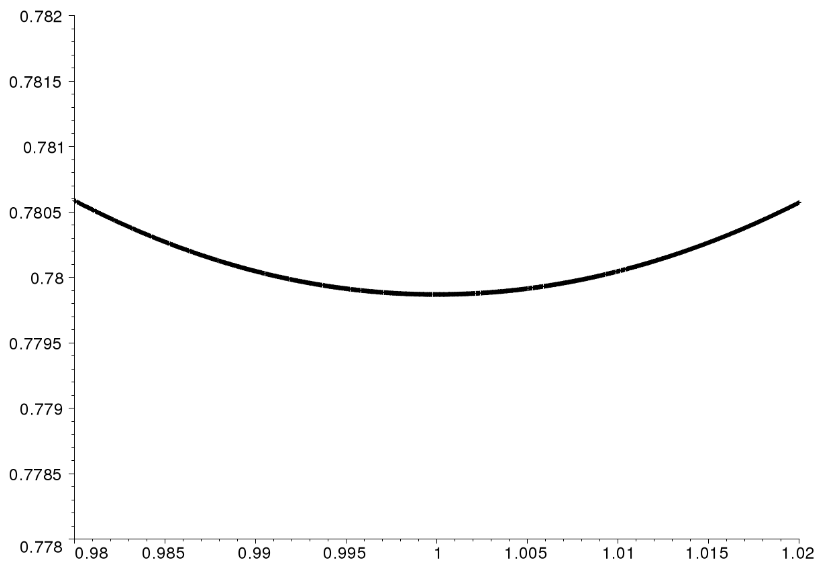

Fig. 4. The same part of the orbit as in figure 2, but obtained from numerical integration of the predator-prey system (4) with $\alpha = 1$ and initial point $(0.9, 0.8)$, using Kahan's method (8) up to $t = 5000$.

expressions are obtained:

$$\begin{aligned} \tilde{x} &= \frac{x\left(4+(4-2x-2y)h+(1-x+y)h^2\right)}{\left(4+(2y-2x)h+(x+y-1)h^2\right)}, \\ \tilde{y} &= \frac{y\left(4+(2x+2y-4)h+(1+x-y)h^2\right)}{\left(4+(2y-2x)h+(x+y-1)h^2\right)}. \end{aligned} \tag{9}$$

Each of the orbits shown in figure 1 was produced from 10000 iterations of the transformation (9) with timestep $h = 0.1$, by starting with twenty different initial conditions $(x, y) = (x_0, y_0)$. Each orbit appears to lie on a closed curve encircling the fixed point $(1, 1)$, as is required for an accurate representation of the solution of (4).

In order to probe the properties of Kahan's discretization more closely the same initial condition $(x_0, y_0) = (0.9, 0.8)$ was taken as for the RKF45 method, and then 50000 iterations of (9) were applied with timestep $h = 0.1$ (to give a total time of length 5000 as before). The energy of the iterates for Kahan's scheme was calculated and the difference $H_t - H_0$ was plotted against time as in figure 5; the results are

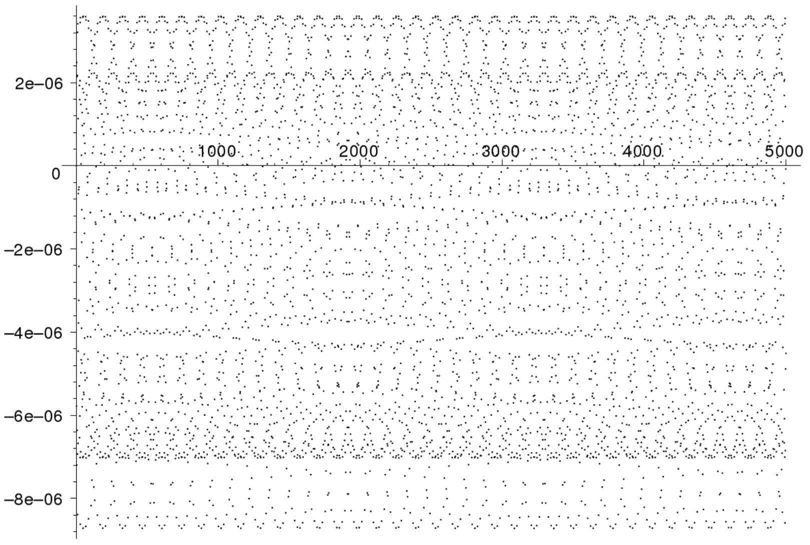

Fig. 5. The energy difference $H_t - H_0$ plotted against t for numerical integration of (4) using Kahan's discretization (8).

seen to be quite different. While for the RKF45 method the value of the energy has increased by more than 3×10^{-4} after time 5000 (and carries on increasing linearly), Kahan's method in contrast gives an oscillating energy value that never increases or decreases by more than 10^{-5} within the same length of time. This would be a consequence of the system (9) having closed orbits lying on curves that are close to the original orbits of (4), since in that case the oscillations in energy would be bounded above and below. In [26], Sanz-Serna observed that Kahan's discrete Lotka-Volterra system is also symplectic, in that it preserves the same area form as the original differential equation i.e. $\frac{1}{\tilde{x}\tilde{y}}\mathrm{d}\tilde{x} \wedge \mathrm{d}\tilde{y} = \frac{1}{xy}\mathrm{d}x \wedge \mathrm{d}y$ holds. Since the differential equation is Hamiltonian and has one degree of freedom, it is completely integrable, so that (as also mentioned in [1]) one can apply KAM theory in the neighbourhood of the elliptic fixed point at $(1, 1)$. Nevertheless, the stability of Kahan's discretization of the Lotka-Volterra model as a numerical scheme seems quite remarkable.

Kahan's method for discretizing quadratic vector fields is mentioned in some of his published work with Li (see [14], and also [13], where it is described how to turn it into a higher order method). More recently, it has been applied to Lotka-Volterra models with three species [25], and it has been shown that also in that case it correctly reproduces the same qualitative features of the dynamics as are present in the corresponding continuous systems. In fact, Kahan's method for systems with quadratic right hand sides was later rediscovered by Hirota and Kimura in the context of integrable systems in classical mechanics, when they found a new completely integrable discretization of the Euler equations for a rigid body rotating about a fixed point [10]. A variety of new discrete integrable mechanical systems have been discovered very recently via this approach [6, 11, 22], and the precise properties preserved by Kahan's scheme are currently the subject of theoretical investigation. However, it is fair to say that Kahan's method fits in with a rather broad set of principles for discretizing differential equations that was developed somewhat earlier by Mickens, and is formulated in his book [18]. Mickens has used his approach to study a wide variety of ordinary (and partial) differential equations, and his non-standard methods have been implemented by a number of researchers in different areas of science and engineering; an extensive review can be found in [21].

Mickens has also applied his general guiding principles for discretization to the same predator-prey model (4) in the case $\alpha = 1$ [19]. Rather than just replacing the linear term x by the average $(\tilde{x} + x)/2$, one can consider more general (asymmetric) replacements $x \to A\tilde{x} + (1 - A)x$, and similarly for the quadratic terms. Moreover, in the forward difference Mickens takes a general denominator function $\phi(h) = h + O(h^2)$; while this has little effect for small h, it becomes important as h increases. For certain systems there are explicit exact discretizations for which ϕ must be specified precisely; one can also ensure that ϕ is bounded for all h, e.g. by setting $\phi(h) = \sin h$. The particular discrete predator-prey system given in [19] has the form

$$\frac{\tilde{x} - x}{\phi} = 2x - \tilde{x} - \tilde{x}y, \qquad \frac{\tilde{y} - y}{\phi} = -\tilde{y} + 2\tilde{x}y - \tilde{x}\tilde{y}. \tag{10}$$

Although the overall system is not linear in $\tilde{x}, \tilde{y}$, the first equation can be solved for $\tilde{x}$ and this can be put back into the second equation to find $\tilde{y}$, so that this gives another explicit birational map of the plane. Numerical results show that the discrete system (10) also has closed orbits in a large neighbourhood of $(1, 1)$ for small values of h. It is straightforward to verify that the map of the plane defined by the above equations for $\tilde{x}, \tilde{y}$ preserves the same symplectic form $\omega = \frac{1}{xy}\mathrm{d}x \wedge \mathrm{d}y$ as before, which is an indication of why its stability properties appear to be the same as for (9). The application of Mickens' methodology to other Lotka-Volterra systems in the plane is treated in [1] and [5]. For an implicit discretization of the Lotka-Volterra predator-prey model, that conserves energy, see [29].

As already mentioned, models in terms of conservative systems are rather rare in ecology or biochemistry (although enzymatic reactions do involve conservation of total enzyme [28]). The Lotka-Volterra model for predator-prey interaction is somewhat unrealistic, because it predicts that any non-zero pair of populations will go through periodic cycles, and thus always return to their initial values after a fixed time. A more realistic scenario is that there is a single attracting periodic cycle which all solutions approach asymptotically, namely a limit cycle. In the next section, limit cycles in reaction kinetics are considered.

3 Trimolecular reaction model

Biological systems that are close to equilibrium can be modelled extremely effectively by linear differential equations [2]. The disadvantage of using linear models is that a linear differential equation of the form $\dot{\mathbf{x}} = M\mathbf{x}$, with a state vector $\mathbf{x}$ and matrix M, has only a single fixed point at $\mathbf{x} = 0$ (at least generically, when M has no non-trivial kernel). Thus in order to allow the possibility of multiple equilibria, one should use nonlinear systems. For nonlinear systems that are close to an equilibrium of node, spiral or saddle point type, the linearized system provides correct qualitative information about solutions in the neighbourhood of the fixed point [12]. The Euler method is exact for homogeneous autonomous linear systems, so standard numerical integration methods are usually completely adequate for modelling systems close to equilibrium. Therefore in order to see novel phenomena that are inherently nonlinear, and thus more difficult to analyze numerically, one should consider nonlinear models away from equilibrium. In this section we consider the application of nonstandard integration schemes to nonlinear systems with limit cycles.

It is worth emphasizing that quadratically nonlinear systems of differential equations arise immediately when one considers reaction kinetics in chemistry, or biochemistry, where the fundamental processes are all dimolecular, involving reactions of the form $A + B \to C$, $A \to B + C$, or $A + B \to C + D$, where A, B, C, D represent different molecular species. Upon applying the Law of Mass Action to such reaction schemes, the equation for the rate of change of concentration of each reactant is given by a sum of linear and quadratric terms in the concentrations. Thus Kahan's discretization method, making symmetric replacements of variables, that is $x \to (\tilde{x} + x)/2$ for linear terms and $xy \to (\tilde{x}y + x\tilde{y})/2$ for quadratric terms, is ideally suited to numerical integration of reaction kinetics models (where the variables x, y etc. would correspond to concentrations). For the situation where N, the number of variables, is

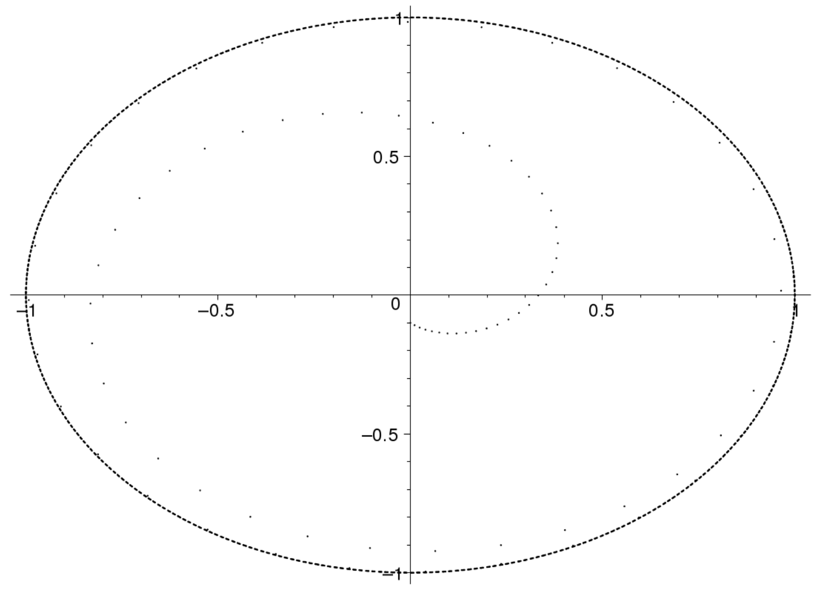

Fig. 6. Numerical integration of (11) using the discretization (13) with stepsize $h = 0.1$ for 2000 iterations starting from $r = 0.1$, $\theta = -\pi/2$ (and plotting $(x, y) = (r\cos\theta, r\sin\theta)$).

large, it is no longer practical to solve the system for the upshifted variables explicitly, as was done to obtain (9) from the original form (8) of the discrete system, because this requires inversion of an $N \times N$ matrix whose entries are polynomial in the variables; this becomes an increasingly difficult problem in symbolic algebra as N increases. However, the system for the upshifted variables $\tilde{x}, \tilde{y}, \ldots$ is always linear, so with numerical values of the iterates there are efficient algorithms for solving such systems, which will be stable when h is small, because in that case the $N \times N$ matrix to be inverted (with numerical entries) is a small perturbation of the identity.

In order to illustrate a simple example of a reaction scheme that includes a limit cycle, i.e. an isolated periodic solution corresponding to a closed curve in the phase space, it is not sufficient to consider dimolecular reactions with two reactants, because limit cycles cannot occur. Indeed, one can write down a general two-species reaction model with variables x, y and (by carefully studying all possible quadratic, linear, and even constant terms) check that limit cycles never appear; in [20] (see p.234) the first derivation of this result is attributed to Hanusse [9]. Thus in

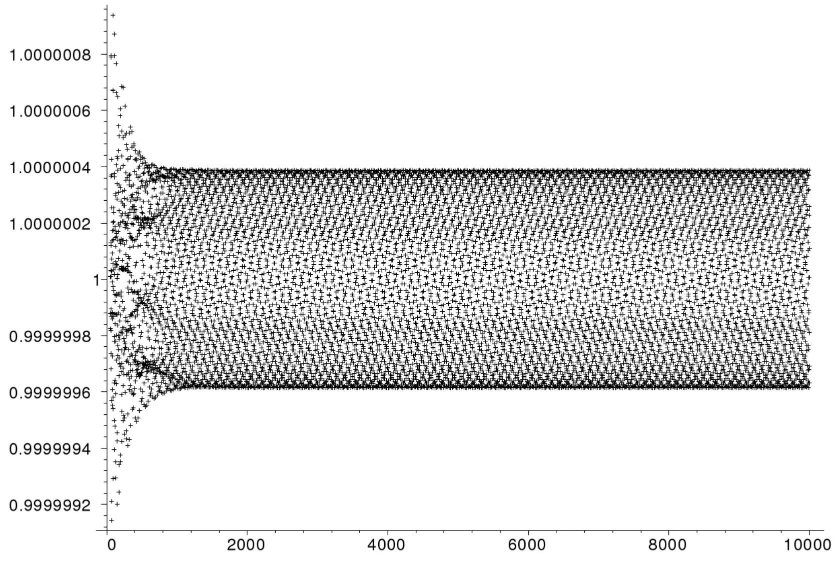

Fig. 7. Plot of r_t against t for $50 \leq t \leq 10000$ from numerical integration of (11) using the RFK45 method.

order to see periodic oscillations (which, unlike the predator-prey system in the previous section, are not orbits in a conservative system) one should either consider three or more species, or allow trimolecular reactions. Here the latter option is taken, but first it is instructive to use a toy model to illustrate how standard numerical integration methods fare with limit cycle solutions.

A simple toy model of a limit cycle in the plane is provided by the pair of differential equations

$$\dot{r} = r(1-r), \qquad \dot{\theta} = 1 + r^2, \tag{11}$$

which is written in terms of polar coordinates r, θ to represent the evolution of the point $(x, y) = (r\cos\theta, r\sin\theta)$ in the plane. The first equation is just the continuous logistic growth model, and has the explicit general solution

$$r(t) = 1/(1 + ke^{-t}) \to 1 \qquad \text{as} \qquad t \to \infty, \tag{12}$$

with arbitrary constant k. Thus the point $(x(t), y(t))$ approaches the unit circle $r = 1$ as $t \to \infty$, which is the limit cycle given by the exact periodic solution $(x, y) = (\cos(2t), \sin(2t))$. Moreover, trajectories that start inside the cycle remain inside (see figure 6), so $r(t) < 1$ for all t, while those that begin outside approach it with $r(t) > 1$ always. However, upon performing a numerical integration of the system (11) with the RKF45 method in Maple version 10, it is observed that, after an initial period of rapid change (starting from $r_0 = r(0) = 0.8$ in the example illustrated in figure 7), the value of the approximation $r_t \approx r(t)$ oscillates indefinitely between several different values, both above and below $r = 1$. For an iterative integration method with fixed floating point precision, one is dealing with a map in a finite state space, so all iterations must either reach a fixed point or be periodic in the long run [24].

Alternatively, upon applying Kahan's unconventional discretization method to (11) one obtains the discrete system

$$\frac{\tilde{r} - r}{h} = \frac{\tilde{r} + r}{2} - \tilde{r}r, \qquad \frac{\tilde{\theta} - \theta}{h} = 1 + \tilde{r}r, \tag{13}$$

from which the equation for r decouples as

$$\tilde{r} = \frac{r(2+h)}{2 + (2r-1)h}. \tag{14}$$

The latter equation is of discrete Riccati type (it is a Möbius transformation), and has the exact solution

$$r = r_t = \left(1 + k\left(\frac{2-h}{2+h}\right)^t\right)^{-1} \to 1 \quad \text{as} \quad t \to \infty \quad \text{whenever} \quad h > 0. \tag{15}$$

Thus the solutions of the latter discretization have precisely the same qualitative behaviour as those of the original system of differential equations (11). The discrete equation for r has the same fixed points (an unstable one at $r = 0$ and a stable one at $r = 1$) as for $\dot{r} = r(1-r)$, but from a numerical implementation of (13) with timestep $h = 0.1$ and 20-digit precision in Maple 10, using the initial value $r_0 = r(0) = 0.8$ (as for the RKF45 method), one finds that $r_t < 1$ holds for all t but the radius converges prematurely (in just over 400 steps) to the false equilibrium $r_\infty = .99999999999999999952 < 1$. This false value is due to rounding errors, and can clearly be improved by performing the same calculations using floating point numbers with successively more digits.

Having considered a toy model of a limit cycle, we now focus on the following simple reaction scheme, including trimolecular reactions, that was presented by Schnakenberg [27]:

$$X \underset{k_2}{\overset{k_1}{\rightleftarrows}} A, \quad B \overset{k_3}{\rightarrow} Y, \quad 2X + Y \overset{k_4}{\rightarrow} 3X. \tag{16}$$

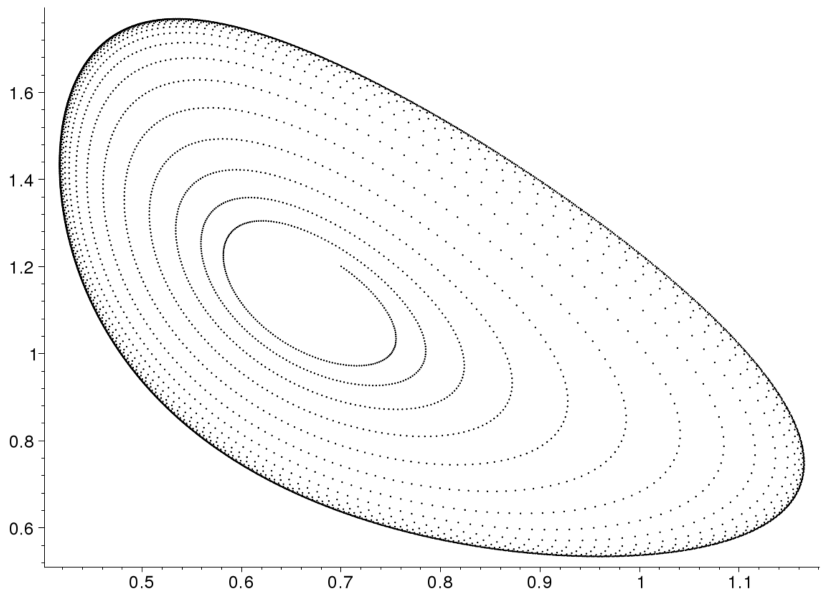

Fig. 8. Numerical integration of (17) with $a = 1/6$, $b = 1/2$ using the birational map φ_h defined by (19), taking the first 5000 points on the orbit of $(0.7, 1.2)$ with $h = 0.05$.

In the above there are four molecular species A, B, X, Y and four rate constants k_j for $j = 1, 2, 3, 4$. If it is assumed that the concentrations of A and B are both held constant, then upon applying the Law of Mass Action and rescaling all the variables one obtains the dimensionless differential equations

$$\dot{x} = a - x + x^2 y, \qquad \dot{y} = b - x^2 y \tag{17}$$

for the scaled concentrations of X, Y (denoted x, y respectively) and the constants $a, b > 0$ (corresponding to the fixed concentrations of A, B). The system (17) has limit cycle solutions for the parameter range $b - a > (b + a)^3$ with $0 < b < 1$, giving a closed periodic cycle encircling the fixed point at $(a + b, b/(a + b)^3)$ (see chapter 7 in [20] for details, and see figure 8 for numerical integration of an orbit with parameters in this range).

Due to the presence of the cubic (degree 3) terms $x^2 y$ on the right hand sides in (17), it is not possible to apply Kahan's scheme for quadratic vector fields. However, we can still obtain a non-standard discretization in the spirit of Mickens' approach. To do so, derivatives are replaced by forward differences, the linear term x in the first equation of (17) is

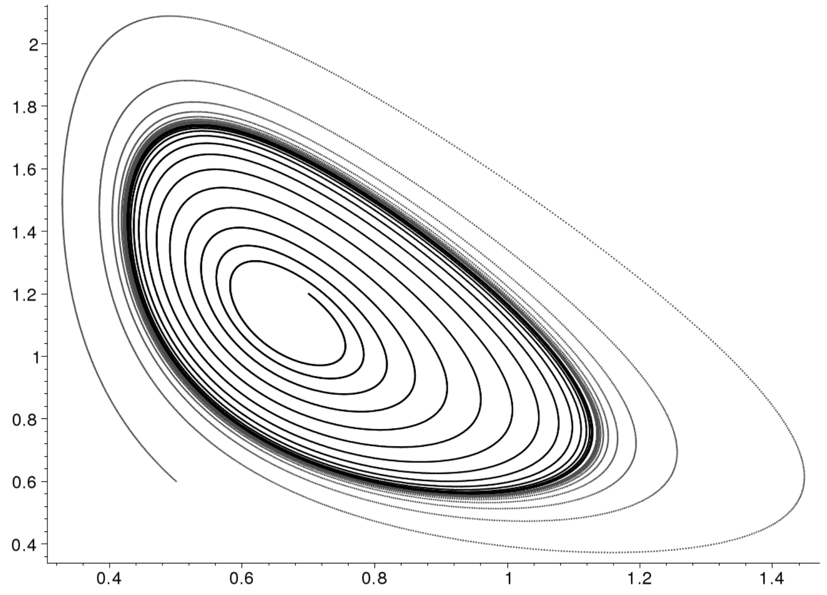

Fig. 9. Two solutions of (17) for $a = 1/6$, $b = 1/2$ integrated numerically using an adaptive modification of the scheme defined by (19), taking variable stepsize $h_{n+1} = h_n d_n / d_{n+1}$ with $h_0 = h_1 = 0.01$ (where d_n is the distance between successive points).

replaced by the symmetric nonlocal expression $(\tilde{x} + x)/2$, and for each cubic term we make a replacement of the general form

$$x^2 y \to (cy + d\tilde{y})(ex^2 + fx\tilde{x} + g\tilde{x}^2), \quad c + d = 1, \quad e + f + g = 1, \tag{18}$$

where the constants $c, d = 1 - c, e, f, g = 1 - e - f$ are allowed to be different in each of the two equations. Without any further restrictions on the latter constants, in general one has a multivalued map for $\tilde{x}, \tilde{y}$: in order to find these upshifted variables in terms of x, y one must solve a pair of polynomial equations, which have multiple roots. However, if it is required that the two equations have an unique solution for $\tilde{x}, \tilde{y}$, and furthermore that one can also solve uniquely for x, y, then there are only two possible discretizations, namely

$$\varphi_h : \quad \frac{\tilde{x} - x}{h} = a - \frac{1}{2}(\tilde{x} + x) + x\tilde{x}\tilde{y}, \quad \frac{\tilde{y} - y}{h} = b - x^2\tilde{y}, \tag{19}$$

and

$$\psi_h : \quad \frac{\tilde{x} - x}{h} = a - \frac{1}{2}(\tilde{x} + x) + x\tilde{x}y, \quad \frac{\tilde{y} - y}{h} = b - \tilde{x}^2 y. \tag{20}$$

The proof that these are the only possible uniquely invertible discretizations with the replacements (18) is somewhat cumbersome, and will be presented elsewhere. In each case the pair of equations defines a birational map of the plane, $\varphi_h : (x, y) \mapsto (\tilde{x}, \tilde{y})$, given by

$$\tilde{x} = \frac{2x + h(2a - x + 2x^3) + h^2(2ax^2 - x^3)}{2 + h(1 - 2xy + 2x^2) + h^2(x^2 - 2bx)}, \quad \tilde{y} = \frac{y + hb}{1 + hx^2}, \tag{21}$$

and similarly for ψ_h. Furthermore, it is not too hard to see from (19) and (20) that these two transformations define each other's inverses, in the sense that $\varphi_h^{-1} = \psi_{-h}$, so henceforth we can concentrate on φ_h.

One can perform exact analysis of the Jacobian of the map φ_h at the fixed point $(a + b, b/(a + b)^3)$; this is along the same lines as the analysis for discretizations of Lotka-Volterra models done by Roeger [25]. Such analysis shows that, just as the differential equation (17) undergoes a Hopf bifurcation along the curve $b - a = (b + a)^3$ in the (a, b) parameter space, which produces a limit cycle lying on an invariant curve in the (x, y) plane, for small h the map φ_h similarly undergoes a Neimark-Sacker bifurcation to produce an invariant curve which is at least $O(h)$ close to the limit cycle of the differential equation. (For technical details of Hopf and Neimark-Sacker bifurcations, see [15].) The correct qualitative nature of the orbits of this discretization can be seen from the numerical results. The first plot for (19) is the orbit of the point $(0.7, 1.2)$, shown in figure 8, for 5000 steps with stepsize $h = 0.05$, which shows convergence towards the attracting invariant curve from the inside. In figure 9, results are shown of applying a simple modification to the method to produce a crude adaptive scheme, by varying the stepsize by a factor proportional to d_n/d_{n+1} at each step, where $d_n = |\mathbf{x}_n - \mathbf{x}_{n-1}|$ is the distance between successive iterates. The latter figure shows the orbit of $(0.5, 0.6)$ spiralling onto the invariant curve from the outside, together with the orbit of $(0.7, 1.2)$, taking 20000 iterations with an initial stepsize $h = 0.01$. The adaptive method shows better resolution than the non-adaptive one in places where the solution of the ordinary differential equation is varying more rapidly. We reserve the detailed properties of the discretization (19), as well as a proper comparison with the RFK45 method, for a forthcoming article.

4 Discussion

For the modelling of biological phenomena in terms of systems of differential equations that exhibit truly nonlinear behaviour, standard techniques for numerical integration can produce approximate solutions of the equations which are qualitatively incorrect. However, the non-standard methods developed by Mickens and Kahan (among others) provide simple discretization schemes that manage to incorporate the correct qualitative features of the original continuous system. Thus the unconventional methods outlined above are worthy of consideration by anyone who is interested in modelling with differential equations. Of course, a considerable amount of theoretical work remains to be done in order to understand precisely when and why such methods outperform conventional ones. In particular, it is important to understand the complexity of non-standard methods compared with Runge-Kutta schemes, e.g. in terms of the number of arithmetic operations involved.

When implementing a numerical scheme, one wishes to avoid introducing new properties into the discrete system which were not present in the original continuous one. We have asserted that Kahan's discrete predator-prey system (8) reproduces the closed orbit structure of the differential equations (4), but to be more precise one should qualify this by saying that this structure holds only in a certain neighbourhood of the elliptic fixed point, with this region being large when h is small. An undesirable feature of the scheme is that sufficiently large positive values of x and y, beyond a boundary line, go outside the positive quadrant after iteration, but in fact the Kahan discretization also displays the characteristics of a chaotic map in a fringe region before this boundary is reached, with the size of this "chaotic fringe" increasing as h increases [23] (and vanishing when $h \to 0$). Of course, for modelling predator-prey systems [17] or interacting populations in other biological scenarios, one can avoid the problem of numerical integration altogether by working directly with discrete models in terms of difference equations or cellular automata [24]. Moreover, one might dispense with deterministic models and work with purely stochastic systems. However, in setting up such models one is often guided by intuition based on continuous models in terms of differential equations.

In this paper, we have illustrated how non-standard discretization methods can be used to study the simplest type of nonlinear oscillation that is not exhibited by a linear system, namely a limit cycle. As far as we are aware, the discretization (19) for the simple trimolecular reaction equations (17) is new. Further analysis of this discrete system will be presented elsewhere.

Finally, we should mention that although (due to the Poincaré-Bendixson theorem [12]) only fixed points and limit cycles can arise as attractors in autonomous two-dimensional systems of differential equations, in three or more dimensions there is the possibility of strange attractors, such as the one appearing in the famous Lorenz model. The Lorenz model is defined by a quadratic vector field in three dimensions, and as such can be integrated using Kahan's unconventional scheme. Indeed, the appropriate discretization has been presented as an example in [13], and our own numerical experiments suggest that, for different values of the stepsize h, this discrete system in 3D contains a strange attractor which is qualitatively like the Lorenz attractor, and converges to it as $h \to 0$.

Acknowledgments. The author is grateful to the Isaac Newton Institute, Cambridge for providing him with a Visiting Fellowship during the completion of this work. He also thanks Matteo Petrera, Yuri Suris and Claude Viallet for helpful correspondence and conversations on related matters.

References

[1] H. Al-Kahby, F. Dannan, and S. Elaydi. Non-standard discretization methods for some biological models. In R.E. Mickens, editor, *Applications of nonstandard finite difference schemes*, pages 155–180. World Scientific, 2000.

[2] U. Alon. *An Introduction to Systems Biology.* Chapman and Hall/CRC, 2007.

[3] J. Blanc. Finite abelian subgroups of the cremona group of the plane. *C. R. Acad. Sci. Paris Ser. I Math.*, 344:21–26, 2007.

[4] C. J. Budd and A. Iserles. Geometric integration: numerical solution of differential equations on manifolds. *Phil. Trans. R. Soc. Lond. A*, 357:943–1133, 1999.

[5] D. Dimitrov and H. V. Kojouharov. Nonstandard finite-difference schemes for general two-dimensional autonomous dynamical systems. *Appl. Math. Lett.*, 18:769–775, 2005.

[6] V. Dragovic and C. Gajic. Hirota-kimura type discretization of the classical nonholonomic suslov problem. *Regular and Chaotic Dynamics*, 13:250–256, 2008.

[7] S. N. Elaydi. *Discrete Chaos.* Chapman and Hall/CRC, 2000.

[8] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.

[9] P. Hanusse. De l'éxistence d'un cycle limite dans l'évolution des systèmes chimiques ouverts. *C. R. Acad. Sci. Paris C*, 274:1245–1247, 1972.

[10] R. Hirota and K. Kimura. Discretization of the euler top. *Jour. Phys. Soc. Jap.*, 69:627–630, 2000.

[11] A. N. W. Hone and M. Petrera. Three-dimensional discrete systems of hirota-kimura type and deformed lie-poisson algebras. *J. Geom. Mech.*, 1:55–85, 2009.

[12] D.W. Jordan and P. Smith. *Nonlinear ordinary differential equations.* OUP, 3rd edition, 1999.

[13] W. Kahan and R. C. Li. Composition constants for raising the order of unconventional schemes for ordinary differential equations. *Math. Comp.*, 88:1089–1099, 1997.

[14] W. Kahan and R. C. Li. Unconventional schemes for a class of ordinary differential equations – with applications to the korteweg-de vries equation. *J. Comp. Phys.*, 134:316–331, 1997.

[15] Y. Kuznetsov. *Elements of Applied Bifurcation Theory.* Springer, 1998.

[16] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics.* CUP, 2004.

[17] R. M. May, M. P. Hassell, R. M. Anderson, and D. W. Tonkyn. Density dependence in host-parasitoid models. *J. Animal Ecol.*, 50:855–865, 1981.

[18] R. E. Mickens. *Nonstandard finite difference models of differential equations.* World Scientific, 1994.

[19] R. E. Mickens. A nonstandard finite-difference scheme for the lotka-volterra system. *Appl. Num. Math.*, 45:309–314, 2003.

[20] J. D. Murray. *Mathematical Biology*, volume I & II. Springer-Verlag, 3rd edition, 2002.

[21] K. C. Patidar. On the use of nonstandard finite difference methods. *J. Difference Eq. Appl.*, 11:735–758, 2005.

[22] M. Petrera, A. Pfadler, and Y. B. Suris. On integrability of hirota-kimura type discretizations. experimental study of the discrete clebsch system. *Exp. Math. to appear*, 2009.

[23] M. Petrera and Y. B. Suris. Hirota-type discretization of 2d lotka-volterra system. *preprint*, 2008.

[24] F. Robert. *Les Systèmes Dynamiques Discrets.* Springer, 2000.

[25] L. W. Roeger. A nonstandard discretization method for lotka-volterra models that preserves periodic solutions. *J. Diff. Eq. Appl.*, 11:721–733, 2005.

[26] J. M. Sanz-Serna. An unconventional symplectic integrator of w. kahan. *Appl. Num. Math.*, 16:245–250, 1994.

[27] J. Schnakenberg. Simple chemical reaction systems with limit cycle behaviour. *J. Theor. Biol.*, 81:389–400, 1979.

[28] L. A. Segel. *Modeling dynamic phenomena in molecular and cell biology.* CUP, 1984.

[29] B. A. Shadwick, J. C. Bowman, and P. J. Morrison. Exactly conservative integrators. *SIAM J. Appl. Math.*, 59:1112–1133, 1998.

[30] Y. B. Suris. *The problem of integrable discretization: Hamiltonian approach*, volume 219 of *Progress in Mathematics.* Birkhäuser Verlag, Basel, 2003.

[31] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, 150:262–268, 1990.

[32] D. G. Zill and M. R. Cullen. *Differential equations with boundary value problems.* Brooks/Cole - Thomson Learning, 2005.

Environment Orientation: An Architecture for Simulating Complex Systems

Tim Hoverd and Susan Stepney

Department of Computer Science, University of York, UK, YO10 5DD
{tim.hoverd,susan}@cs.york.ac.uk

Abstract. A naïve implementation of a complex system simulation with its plethora of interacting agents would be to represent those interactions as direct communications between the agents themselves. Considerations of the real world that a complex system inhabits shows that agent interactions are actually mediated by the environment within which they are embedded and which embodies facilities used by the agents. This suggests an "environment oriented" simulation architecture.

Here we motivate and describe an abstract software architecture for an environment oriented approach to complex systems simulation, and sketch the implementation of this architecture in a number of different ways.

1 Introduction

Complex systems comprise of a number of agents that interact in some particular environment. The behaviour of any individual agent is relatively simple and local. A complex global behaviour emerges as a consequence the interaction of a large number of such agents in a particular environment.

A complex system can be simulated using computational devices to provide an executable model of the real world situation. Like all such models it should be constructed in manner that can feasibly be implemented, and may well avoid many real world details. However, such a model must encapsulate the key interactions between agents from which emerges the global behaviour.

2 Motivation

Complex systems get their emergent behaviour from interactions between the agents that comprise the system. Naïve implementation models therefore describe direct interactions between those agents.

Such an approach, however, leads to many implementation difficulties. Firstly, scaling the number of agents in a simulation to something representative of the modelled world is infeasible, because the number of communication channels required rapidly exceeds the capabilities of the simulation. Secondly, and of particular importance here, if such a model were to be implemented without detailed attention paid to concurrency issues, then it would doubtless deadlock very quickly because of the loops apparent in the agent/channel graph. Consequently, such naïve implementations are never seen.

These deadlock issues are resolved in simulations by the introduction of techniques, such as the “client server” pattern for concurrent systems [1, 12] and barrier synchronisation [2], which impose a processing pattern onto communications between the various components of the simulation. These patterns seek to prevent the appearance of deadlocks.

In the case of the client server pattern components of the simulation are coded so as to operate in a manner reminiscent of “client-server” enterprise systems [21] or, more generally, multi-tier architectures [24]. In such systems the clients and the servers are layers in an architecture where the servers provide a pre-defined set of services to the clients. Each client is able to operate in a manner largely independent of others because the implementation of the system constrains the overall patterns of behaviour, for example by transactional access to an underlying repository [26], in such a manner as to guarantee various overall system properties.

Use of this convention introduces a pattern into the simulation that does not at first sight appear to exist in the real world being modelled. For example, the birds that flock above a city-centre park are not apparently working to some standard global pattern lest they deadlock and fall out of the sky. It appears that each bird is observing other birds and then doing what it wants, when it wants, and in whatever order it wants doing so in the sure and certain knowledge that its world really is deadlock free. That is, it appears as if the real world of these birds is rather different from a set of agents communicating with each other in a simulation of, for example, bird flocking behaviour.

The rest of this paper looks in more detail at what is really going on in a such a complex system, leading to some alternatives for the software architecture of complex systems implementations.

3 Real world agents and their environment

3.1 Action at a distance *versus* mediating fields

Let us think about how the real world agents actually interact. Although at first sight it is convenient to think about flocking birds interacting directly some thought shows that in fact this is a simplification of what is really going on.

A bird flying along reflects the ambient light into the space around it; as it sings it pressurises the air about it. Another bird, assuming that it is awake, is sensitive to the propagated light and air pressure, and in this way can both see and hear the first bird. That is, these two birds are not directly communicating with each other. The first is placing information into its environment, which information can be detected by the second bird when it observes its own environment, if it is interested in that sort of information.

Such a view, which is essentially an alternative model of interaction between the birds, relies upon a very detailed environment in which the agents, in this case the birds, are embedded. One bird can always come along and look in the environment and see what another bird is frequently placing in the environment. A different bird might update the environment only seldomly, if it is just sitting quietly on some perch.

The real world is such a detailed environment; one where fields interact, photons pass each other and the rest of physics is implemented with ease. In this view the agents are *embodied* in the environment [19], and it provides services to those agents. Each agent just does what it wants without regard to direct interactions with other agents. That is, even in the real world, the agents in a complex system are interacting in a manner reminiscent of a client server architecture. The environment provides services to the agents, in a manner analogous to a *server*. The agents are *clients* of those services.

The naïve model of a complex system, with agents directly interacting with each other, is essentially "action at a distance". One agent must know directly what other agents exist that are interested in it and must directly interact with those agents. As this is happening those distant agents are also potentially interacting in the reverse direction.

Reflecting our observation of the real world, we instead take an "environment oriented" approach. Here agents do not interact directly, but communicate through some mediating fields that exist in their environment. In this approach there is no direct interaction at all; the lives of individual agents just affect each other by existing within the same set of fields; within the same physics.

As a different example of this "environment orientation" consider an adaptive immune system. Here the agents are the various molecules and cells that form the active components. The molecules are not directly signalling to each other about the feasibility of particular interactions. In this case an environment oriented model would represent these molecules by their concentrations in the environment which would affect the probability of interactions occurring as a consequence of the stochastic processes mediated by the environment.

3.2 State

The notion of "environment orientation" reflects the real world in a useful manner. However, what is it that agents communicate with the environment?

In something like a collection of birds flocking in the real world each bird has a large and complex internal state: it knows whether it is flying or not, how hungry it is, whether it needs to drink or defecate. But, from the point of view of flocking, other birds are interested only in the distances between the birds and what the perceived relative velocities of the other birds are.

That is, each agent has an "internal state" that represents everything it needs to know to behave appropriately. Further, each agent exposes an "external state" to the environment, which is available to other agents in the same environment. This external state could be simply a subset of the agent's internal state. For example, in the case of the bird it could just be that part of the internal state that represents the position and velocity of the bird. However, there are cases where the agent could deliberately mislead other agents with its external state. For example, when one insect species mimics another it is deliberately creating external state to mislead observers about its internal state.

Complex system agents are essentially egocentric. That is, the emergent behaviour appears as a consequence of each agent just doing what it wants to do in its own environment. A flocking bird, then, does not know precisely where it is, just merely where other birds are relative to it. In a complex system simulation, something does need to know where the agents are, because those positions are the overall context of execution of the complex system. This context is the environment. That is, the environment must know where each agent, is and therefore the environment will know what other agents are in the vicinity of each agent. That is, the environment knows things about the agents that are not actually part of the agent's internal state. For example, a bird just thinks that it is flying in the direction of an interesting looking food source, but the environment knows that it is actually flying north-by-northwest.

A refinement to this notion is the observation that an agent generates some external state just by virtue of the physics of its environment. For example, photons just bounce off a bird, so other birds can see it, and are also able to infer position and velocity from those photons. This "involuntary" external state is contrasted with other state placed into the environment by an agent in a "voluntary" manner. Voluntary state could be, in the example of birds, a song that is sung in response to hearing the song of another bird of the same species (which it hears through the mediating environment), sung maybe for territorial enforcement or finding a mate.

3.3 Querying

In this environment oriented approach, each agent interacts with the environment to access information about the other agents' (external) states. Simplistically, each agent "asks" the environment for information about other relevant agents' state (the agents it can see, or hear, for example); this state information can then be used by the querying agent to update its internal state appropriately.

The reply to such a query is a set of values in some topology [7], which not only represents the set of all possible values but also describes how the values might change.

For example, in a bird flocking example, one of the items in a query result could represent a bird that is close to the querier. As such, the environment can accurately describe the (relative) position of the nearby bird and its velocity in terms of a three-dimensional Cartesian space. Furthermore, the topology of the particular space used might show that the nearby bird could move freely in the two horizontal dimensions but it was constrained to move only upwards in the vertical dimension because it is, at the moment, standing on the ground. That is, the reply to a query about the position of the bird gives a precise position in a space, but that space is further described by its extent and its shape.

If the bird being described is distant then the position of the bird may not be accurately described; for example, it might be clear in what direction the bird lies but its distance from the querier could be only poorly known. Similarly, the velocity of the bird might be only poorly described, if at all, as the velocity of a distant agent which appears merely as a distant speck might be very hard to determine. In this case the reply is again a position in a space. However, in this case that space is two-dimensional being the surface of a portion of a sphere centred on the querying agent. Because the distance to the observed bird cannot be determined it cannot be moved inside or outside of that sphere.

Here the simulated environment is acting as the *embodiment* of sophisticated functions performed in the real world by both the agent itself and the environment. The agent itself detects the photons impinging on its retinas from a distant bird and attempts to calculate size, distance and velocity of the bird from those photons and, probably, experience in these sorts of situations. The real world, that is the environment, affects many aspects of the passage of those photons; it understands the albedo of the distant bird and can calculate how photons from the Sun are reflected by the bird, and how effectively those photons are transferred to the observing bird.

The environment oriented approach provides a way to separate concerns between the agents and their environment. In a particular simulation, the choice of what computation is performed by the environment, and what by agents, is a modelling decision. Certain functions may be embodied in the environment itself, and those calculations performed by the environment. Alternatively, responsibility for those those functions may be assigned to certain agents (either existing ones, or new ones designed to support those functions).

The notion of the results of the query being embedded in a topology allows the interaction between agents to follow a number of different patterns simultaneously. The example given above is a purely spatial one, the notion of space clearly being of significance in complex systems implementation as in [1]. However, the exact same query/response model could be used for any interaction between agents in a complex system. One extension of the simple spatial model is to note that a human agent is physically "near" to a collection of other human agents but may nonetheless communicate simply with other human agents whose telephone numbers are in the first agent's address book. That is, there are two sorts of "nearness" here: one is physical nearness, the other is "communicable" nearness. For some aspects of complex systems behaviour only the first sort of nearness would be relevant, for others both sets of "near" agents might be important. (This example is inspired by Milner's *bigraphical model* designed to model both a spatial and a connectivity configuration simultaneously [13].)

3.4 Environment orientation

In summary, the environment oriented approach to complex systems simulation eschews all representations of direct interactions between agents. Rather, all agent behaviour is seen as mediated through the environment within which all the agents are embedded; the essential rationale for this being that this is the way that the real world is structured.

Although the notion of the role of the environment is based on observations of the real physical world, the particular agents and behaviours that exist in a simulation is a modelling decision. Each simulation should be constructed with the explicit knowledge of which aspects are to be embodied in the environment.

Regardless of its particular role, each agent has an internal state, representing what the agent knows of itself. It publicises some aspects of its state, its "external state" to the environment within which it is embedded. The agent may decide when to publicise its external state. Agent behaviour is provided for by allowing the agent to retrieve, from its environment, information about the external state of the agents with which it is interacting. Consequently, the environment must be aware of the agents with which each other agent can interact.

4 Software architectural styles

The "environment orientation" approach to complex systems must be readily implementable to be of use as an implementation platform for complex systems simulations. That is, we must define an abstract architecture that defines this sort of systems implementation.

The model as described is essentially a client server one. As has been described, real world complex systems are inherently "client server" in that the agents function essentially as clients of the environment.

A client server architecture is an appealing approach, since there is considerable experience with this approach that forms the basis of most high performance commercial computing. There are also several standardised abstract client server architectures, such as the REST architecture [4] that is the core of the Internet and the services it supports. These show the value of defining services in this manner.

The server in an "environment orientation" complex systems implementation must provide services that:

1. retain the external state of agents
2. provide that external state to other agents as and when required

The second of these services must reflect what aspects of each agents' external state is visible to a requesting agent. That is, the environment must know which other agents are in the "neighbourhood" of a requesting agent and must also know the topology of the result space in which to embed responses to requests.

In addition to providing such services to its clients, that is the agents, the environment may embody many aspects of the world that is being

```
while (true)
{
  Neighbourhood n = env.query(queryText,
                   <parameters drawn from internal state>)
  internalState.update(n)
  env.update(generateExternalState(internalState))
}
```

Fig. 1. Pseudo-code for agents using *query oriented* server

simulated or modelled. For example, if the complex system were modelling ant communication via stigmergy [3] then the environment itself could modify the external state of ant trails so that they decayed at the appropriate rate. This approach is a particular modelling decision. Alternatively, the ant trail might be modelled an agent; then it, and not the environment, would implement the process of pheromone decay.

Some aspects of this sort of architecture are seen in [1] where the implementation of various approaches to the representation of *space* in a complex system are investigated. The related "boids" simulation (based on [17]) uses a notion of "location" that is similar to the environment oriented server discussed here.

The first of these services listed above is susceptible to many different implementations, although the precise form of the delivered state is not defined here.

For the second service, there are two strategies, relating to a possible inversion of control. One approach would be for an agent that wishes to see the external state of a set of other agents, to make a *query* of the underlying "environment orientation" server. The query would provide the server will all the information it needed, along with its knowledge of the agents, to select the information required and provide it to the agents. This strategy, referred to here as *query oriented*, is summarised by the pseudo code in figure 1.

A complementary approach would be for agents to inform the server of the sort of information they were interested in, and to have that information delivered as and when it was available. In the meantime the agent would carry on with its normal behaviour. This strategy, referred to here as *subscription oriented*, is summarised by the pseudo code in figure 2.

These two approaches have different characteristics. The query oriented is appropriate for systems, perhaps like bird flocking simulation, where an individual agent can always be sure that its environment will change rapidly and apparently continuously. The subscription oriented

```
...
env.registerInterest(topic, callback)
...

void callback(Neighbourhood n)
{
  internalState.update(n)
  env.update(generateExternalState(internalState))
}
```

Fig. 2. Pseudo-code for agents using *subscription oriented* server

approach would be useful for systems where some information was available only occasionally and unpredictably, or where it was needed to "interrupt" an agent from its normal activities. That is, in situations where the particular environment was not changing apparently continuously.

In this abstract architecture, the server is the entire locus of inter-agent concurrency. That is, the agents execute without consideration for each other, simply relying on the server to provide pertinent information. This is the approach used in the world's largest commercial systems.

There are, though, at least two other issues that must be addressed here.

The first concerns that of fairness. If an environment server is being queried by a, potentially, very large number of clients then it must be the case that requests from those clients are handled in a fair manner. This is already an issue in multi-tier commercial systems and will not be further addressed here as it seems likely that existing approaches will satisfy the demands of a complex system simulation.

The second issue is that of time. Commercial systems are all "real time" systems, in the sense that the clients are usually aware of what the real world time is because that time is often pertinent to the processing that is being carried out. For a complex systems simulation there are further considerations. The simulation may run, as a whole, faster or slower than real time. In particular, individual agents can run at different rates from other agents, depending on how much processing they have to do (an active flying flocking bird will require more processing time to simulate unit time of its life than will an inactive perching sleeping bird). That is, the simulation as a whole, and the components of the simulation, are running in simulated time. As such, the "simulated time" is properly part of the environment within which the simulation's agents are embedded. Hence, an environment server should also provide a time service, that defines the current simulated time for each of the agents in the simulation. These agents can then, when necessary, consult the current time and use that to influence their activities.

5 Implementations

The architecture discussion so far has been devoid of implementation choices. The principal implementation choice is that of an environment server that can

- support the agents' external state where each item item of such state is in essence a tuple that contains whatever information is necessary for the particular application
- provide a means of accessing and distributing that state
- provide a mechanism for tracking the progress of simulated time

For example, in a bird flocking simulation each tuple retrieved by, or presented to, an agent would include another agent's relative position and perceived velocity. Additional entries in the tuple would allow the topology of the result space to be determined. For example, if the agent in question was distant then the perceived velocity might well be represented in a single-dimensional space with very restricted possible changes instead of the three-dimensional space that would be appropriate for the velocities of nearby agents.

Regardless of these decisions, the data provided to a requesting agent takes the form of tuples. There are several possible implementation choices for how a server could provide the supply of tuples, described below.

5.1 Tuple spaces

The Linda programming language was first proposed in the mid 1980s [6] as a new way of handling concurrency and coordination. A running Linda system provides a "tuple space" which is populated, and examined, by a potentially large collection of concurrently executing agents. Linda provides primitives allowing the connected agents both to query the tuplespace for tuples that match some expression and to block waiting for an appropriate tuple to appear. As such the model supports both types of server architecture discussed in a straightforward manner.

The Linda concepts have been implemented in a number of modern programming languages. For example, JavaSpaces [5, 11] provides Linda-like facilities in the Java programming language as part of the Jini infrastructure. Rinda [18] provides tuplespaces for Ruby. TSpaces [8] is a simple implementation of the Linda ideas within Java from IBM.

A refinement of tuple spaces which is also relevant to this subject is that of tuple centres [16]. Tuple centres are essentially the notion of tuple spaces which have some behaviour. As such, a tuple centre could be seen as the implementation of a particular environment server.

5.2 Publish/Subscribe systems

The publish/subscribe pattern [25] is frequently supported by enterprise middleware, in particular by message oriented middleware [23]. For example, the Java Message Server [15] provides publish/subscribe facilities for users of the Java 2 Enterprise Edition. The publish/subscribe pattern provides for a server to distribute information on a number of *topics* to a number of connected clients. The pattern is often used, for example, in trading systems where some clients might require to be informed of changes in the prices of particular financial instruments when they occur. This is a very similar situation to that described as here as subscription oriented. A topic here could be, for example in the context of a bird flocking system, "the state of agents in the vicinity". Whenever one of those agents does indeed move the agent that registered the topic could be informed of a set of new tuples of information.

Publish/subscribe systems are used commercially in situations where there is a very high data rate, such as the instrument/price situation described above. As such they are also suitable for distributing information in a complex system simulation.

5.3 RDBMS

The use of a relational database management system (RDBMS) is a further possible implementation mechanism. Relational databases are essentially large containers for tuples. Each table in the RDBMS is a set of tuples with the same layout. Furthermore RDBMSs provide a highly expressive declarative query language (SQL [9, 10]) and are commonly used in situations where very high performance is required. As such they provide an attractive mechanism for the query oriented approach to the abstract architecture.

It is less clear how an RDBMS could be used for the subscription oriented architectural pattern. RDBMSs do support mechanisms that are capable of use in this manner (typically, triggers). However, they are clumsy in use and probably not suitable for the very flexible scenarios of complex systems.

5.4 Process oriented programming languages

Process oriented programming is at the heart of the CoSMoS[1] project (of which this work is part). The environment oriented architecture could be implemented using a process oriented language such as occam-π [20]. This is the language used for the models of space described in [1]. Using occam-π to implement simulations with the environment oriented architecture would ideally require the definition of a set of standard libraries that would hide many of the internal details, and allow the programmer to operate at a higher level of abstraction, purely in terms of things like tuples and queries.

6 Prototypes

We have implemented prototype complex systems simulations following the environment oriented architectural style. These prototypes have explored only the query oriented approach to the server. In particular, two prototype systems have been implemented, each of which is an implementation, in Java, of Reynolds' Boids [17], a very simple set of rules to simulate flocking.

As yet neither of the prototypes has been subjected to significant performance analysis and testing. In this first instance, we are simply establishing the capabilities of the abstract architecture.

6.1 Tuplespace prototype

The first prototype is an implementation using TSpaces (chosen due to the simplicity of configuring the server as compared with JavaSpaces).

The design of this system uses a single TSpaces server, running as a separate heavyweight process (a process running under control of the operating system and isolated from other such processes). A single boids heavyweight process implements each boid with a separate thread (a lightweight process not isolated from other such threads by operating system mechanisms). Each such thread executes an instance of the pseudocode shown in figure 3, which is a simple variant of that shown in figure 1.

So each boid gets the tuples about boids in its neighbourhood, delegating the notion of what "its neighbourhood" means to the environment itself. As far as the boid is concerned it is querying for "all the boids". The environment knows where each boid is in the entire world and answers a *relative* neighbourhood of the querier: the positions on the boids in the returned neighbourhood are expressed relative to that

[1] http://www.cosmos-research.org

```
while (true)
{
  Neighbourhood n = env.query("allBoids")
  Vector acceleration = n.centreOfMassRule() +
                        n.matchVelocityRule() +
                        n.repelBoidsRule()
  velocity = velocity + acceleration
  env.updateTuple(boidId, velocity)
  wait(short_delay)
}
```

Fig. 3. Pseudo-code for Reynolds' boids using query oriented server

of the querier. Furthermore, only the boids that are in what the environment deems to be "the neighbourhood" of the querier are supplied in the neighbourhood.

The boid uses the returned neighbourhood information to implement the three rules of Reynolds' algorithm, using the relative positions provided in the neighbourhood, to calculate its acceleration, which is applied to its internal state, here just the boid's velocity. The environment is then updated with its external state which in this simple example is the same velocity. There is no "position" in this state, because the boid is just where the boid is. It is up to the environment to know where the boid actually is in world, which it can calculate from the boid's velocity.

Nowhere in this pseudo-code, or in the Java code actually written, is there anything about directly coordinating the activities of separate boids. All of these details are delegated to the environment, which embodies both a knowledge of the world as a whole, for example it knows that it is a toroidal space, and of the perception of the boids, that is it knows how far away a boid has to be to be deemed "not in the neighbourhood". In this simple example, it is not necessary for the environment to support a time server, as each boid agent performs the same amount of processing to update its state.

A consequence of this lack of interaction between boids is that other versions of the same code, ones where multiple boid agents are supported by each thread, can been written. Each thread sequentially executes the same code for each of the boids in its control. The behaviour of this variant is essentially identical to the thread per boid version, although requiring fewer threads.

The implementation of the environment is carried out by using a façade object [22], in the boids process, that provides a layer above the TSpaces server itself. This façade, in the TSpaces code, retrieves all of the boids from the server and filters them for locality before presentation to the querier as the querier's neighbourhood. It must be done this way because the TSpaces query mechanisms are limited to essentially pattern matching between a template tuple and the tuples in the server's tuplespace.

There are TSpaces mechanisms that could be used to implement an "interrupt oriented" server but these have, as yet, not been investigated.

6.2 RDBMS prototype

A second prototype has also been constructed that uses an RDBMS, specifically MySQL [14]. This prototype also functions well.

The code executed by the RDBMS version is much the same as for the TSpaces variant. The difference, though, is in the environment façade. The RDBMS version can be much simpler, as the process of filtering for local boids may be done using SQL in the database query itself.

7 Future work

The architecture as described is the essential core of the environment oriented approach to complex systems simulation. Future work will concentrate on two main issues.

The first issue is that of the appropriateness, or otherwise, of the two architectural patterns, query orientation and subscription orientation. This will be investigated by producing further prototype implementations that use each style, and combinations of the two.

The other issue is of more theoretical interest. When an agent makes a query (which is logically the same as describing a topic on which it will receive tuples in the subscription oriented architecture) then, as has been described, the response essentially carries with it the topology of the space in which the response is embedded. Realistic complex systems are likely to either:

1. make multiple queries each of which generates a response in a different space or
2. receive responses to a single query with varying topologies (such as near and distant birds in a bird flocking example)

Future work will look at the issues relating to how the responses in different topologies are combined, if that is feasible, and what that implies for more complicated complex systems which more closely represent the details of the real world.

8 Conclusions

The agents in real world complex systems do not directly interact via some "action at a distance"; they interact through the mechanisms mediated by a complicated environment in which they are all embedded. Producing complex systems simulations in an environment oriented manner uses environment implementations in which many complicated functions of the agents are embedded. The use of the environment oriented approach to complex systems simulation promises to raise the level of abstraction in simulation development. This approach allows design to avoid many details related to deadlock and communication. Other issues become apparent, such as how to handle the varying resolution and accuracy inherent in a typical complex real world situation. These issues can potentially be represented as a set of topologies in which real work values are embedded.

8.1 Acknowledgements

The work described here is part of the CoSMoS[2] project, funded by EPSRC grant EP/E053505/1 and a Microsoft Research Europe PhD studentship.

References

[1] P. Andrews, A. Sampson, J. Bjørndalen, S. Stepney, J. Timmis, D. Warren, and P. Welch. Investigating patterns for the process-oriented modelling and simulation of space in complex systems. In *Artificial Life XI*, pages 17–24. MIT Press, 2008.

[2] Fred R. M. Barnes, Peter H. Welch, and Adam T. Sampson. Barrier synchronisation for occam-pi. In Hamid R. Arabnia, editor, *PDPTA*, pages 173–179. CSREA Press, 2005.

[3] J. L. Deneubourg and S. Goss. Collective patterns and decision-making. *Ethology, Ecology & Evolution*, 1:295–311, 1989.

[4] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Inter. Tech.*, 2(2):115–150, May 2002.

[5] Eric Freeman, Susanne Hupfer, and Ken Arnold. *JavaSpaces Principles, Patterns and Practice.* Addison-Wesley, 1999.

[6] David Gelernter. Generative communication in Linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, January 1985.

[7] Jean-Louis Giavitto and Olivier Michel. Data structure as topological spaces. In *Proceedings of the 3rd International Conference on Unconventional Models of Computation*, pages 137–150, 2002.

[2] http://www.cosmos-research.org

[8] IBM. The TSpaces vision. `http://www.almaden.ibm.com/cs/TSpaces/html/Vision.html`, accessed on 6th May, 2009.

[9] ISO. *ISO/IEC 9075-1:1999: Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*. 1999.

[10] ISO. *ISO/IEC 9075-2:1999: Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*. 1999.

[11] Jini. The community resource for Jini technology. `http://www.jini.org`, accessed on 6th May, 2009.

[12] J. M. R. Martin and P. H. Welch. A design strategy for deadlock-free concurrent systems. *Transputer Communications*, 3(4), 1997.

[13] Robin Milner. *The Space and Motion of Communicating Agents.* CUP, 2009.

[14] MySQL. Open source database. `http://www.mysql.com`, accessed on 6th May, 2009.

[15] Sun Developer Network. Java Message Service (JMS). `http://java.sun.com/products/jms/`, accessed on 6th May, 2009.

[16] Andrea Omicini and Enrico Denti. From tuple spaces to tuple centres. *Sci. Comput. Program.*, 41(3):277–294, 2001.

[17] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[18] Masatoshi Seki. dRuby and Rinda: Implementation and Application of Distributed Ruby and its Parallel Coordination Mechanism. *International Journal of Parallel Programming*, 37(1):37–57, 2009.

[19] Susan Stepney. Embodiment. In Darren Flower and Jon Timmis, editors, *In Silico Immunology*, chapter 12, pages 265–288. Springer, 2007.

[20] Peter H. Welch and Fred R. M. Barnes. Communicating mobile processes. In Ali E. Abdallah, Cliff B. Jones, and Jeff W. Sanders, editors, *25 Years Communicating Sequential Processes*, volume 3525 of *LNCS*, pages 175–210. Springer, 2004.

[21] Wikipedia. Client server architecture. `http://en.wikipedia.org/wiki/Client-server`, accessed on 18th June, 2009.

[22] Wikipedia. Facade pattern. `http://en.wikipedia.org/wiki/Facadepattern`, accessed on 6th May, 2009.

[23] Wikipedia. Message oriented middleware. `http://en.wikipedia.org/wiki/MessageOrientedMiddleware`, accessed on 6th May, 2009.

[24] Wikipedia. Multitier architecture. `http://en.wikipedia.org/wiki/Multitierarchitecture`, accessed on 18th June, 2009.

[25] Wikipedia. Publish/subscribe. `http://en.wikipedia.org/wiki/Publish/subscribe`, accessed on 6th May, 2009.

[26] Wikipedia. Transaction processing. `http://en.wikipedia.org/wiki/Transactionprocessing`, accessed on 18th June, 2009.

A Framework Proposition for Cellular Locality of *Dictyostelium* Modelled in π-Calculus

Anthony Nash and Sara Kalvala

Department of Computer Science, The University of Warwick, Coventry, UK,
Anthony.Nash@warwick.ac.uk

Abstract. The aim of this paper is to review the use of process calculi as a means of representing various biochemical networks and processes. Recent literature, ideas and formulated systems are explored in conjunction with their respective biological examples. We then show how π-Calculus can be used to model various aspects of cell locality in a cellular automaton, in addition to the signal transduction responsible for *Dictyostelium discoideum* aggregation.

1 Introduction

Biochemical networks are built from a collection of biochemical signals, which in turn are formed from organised groups of cells. Within each cell commands carried out are the product of molecular interactions such as protein kinase, gene transcription and translation and the release of calcium. Such a system is so complex, scientists are finding it very hard to replicate these processes without resorting to a significant level of abstraction. Most cases of biological simulation require the extrapolation of ordinary differential equations. Yet, the idea that differential equations only provide a very rigid set of results [14] allows room to explore biological systems from the perspective of formal language theory. By expressing every component of a biological network as a process we find an interesting set of parallels between biological networks and process calculi.

We use the soil-dwelling amoeba *D. discoideum* as our biological subject of interest, and from it propose a framework in which to model the chemotactic aggregation of the cells. The framework is two-fold; firstly, by representing *D. discoideum* as a population within a discrete two-dimensional cellular automaton it will be possible to experiment with cell locality by making modifications to the π-Calculus representing the underlying grid; secondly, the cells are driven by π-Calculus structured chemical activation and chemical transport. Cell behaviour can be altered by making modifications to the underlying π-Calculus: for example, modifying required levels of internal cAMP (Cyclic adenosine monophosphate) to activate PKA. We must note that a CA (Cellular Automaton) system's behaviour is not dictated by its rules but rather by the amalgamation of cell locality and cell rules across the system as a whole. Such behaviour can be seen throughout biology; for example, the global effects of cAMP waves through *D. discoideum.*

This paper's biological representative, *D. discoideum*, is a thoroughly studied amoeba with studies starting as early as the 1940s and mathematical models making their way into publications by the 1970s. We propose the use of π-Calculus in an attempt to structure transmembrane signals and cAMP waves across a two-dimensional cellular automaton space, leading to the expected wave-like aggregations. π-Calculus will lead to a simplified framework, where biochemical values can be modified to reveal emergent behaviour of aggregating cells.

The paper will follow with an in-depth description on *D. discoideum.* This is immediately followed by a short investigation into some of the ODE (ordinary differential equation) models. We then explore a select sample of how process calculi have made their way into biology. Finally, we include a proposal and discussion on modelling given aspects of *D. discoideum* using π-Calculus and how this formal language can be used to implement such a model in a discrete cellular automaton environment.

2 *Dictyostelium discoideum*

D. discoideum is a predatory soil-dwelling amoeba, which collectively gathers to form a slime mold. Each amoeba feeds on decaying matter and a variety of micro-organisms; for example, *E. coli.* The amoeba is typically $10\mu m$ in diameter. Due to the number of observable biological processes throughout its complete life cycle, the amoeba has acquired a history of extensive study [2].

The structure of its cellular makeup suggests features common to both plants and animals. It contains cellulose, the most prolific organic compound found on the planet, and develops spores to further its survivability.

As the structure of the amoeba is similar to animal cells, movement is achieved through a biological process known as morphogenesis. Upon receiving a signal, whether it is a hormone, a toxic chemical, or by mechanical stresses, a protrusion is extended and through contraction the body of the cell moves forward. Specifically, *D. discoideum* uses the detection of the chemical cAMP to cause a cytoplasmic release of calcium, causing the extension of a pseudopod in the positive direction of the chemical gradient.

There are three stages to the life cycle of the amoeba. The first stage is a vegetative cycle where the amoeba lives a solitary existence feeding on bacteria. Whilst there is an abundant quantity of food *D. discoideum* behaves on an individual basis, performing unicellular reproduction. Feeding occurs through phagocytosis, a process by which the organism engulfs the food source with a membrane.

The second stage of the life cycle begins as soon as the cells begin to starve. This is commonly known as the chemotaxis and aggregation stage. Starvation leads to a number of cells forming into *pacemaker cells*, periodically releasing cAMP [8]. Neighbouring cells detect the release of cAMP and move towards the source [9] whilst excreting cAMP themselves. This causes the cells to form into aggregation waves. Aggregation can involve up to 100,000 cells with a combined circumference of 20mm. The cAMP signal does not diffuse far from its originating source; it is destroyed within $57\mu m$ (approximately equal to six cell diameters) by arcasinase (a phosphodiesterase enzyme) also produced by the starving amoeba. The global effect of cAMP synthesis and detection causes the amoebas to congregate into a multicellular tipped aggregate.

Figure 1 (taken from [11]) illustrates part of the stage of cAMP transduction, creation and oscillatory control. The cAMP secreted by pacemaker cells acts as a ligand for *D. discoideum* CAR1 extracellular membrane receptors. CAR1 is a cell receptor located on the outer-surface of the cell membrane (extracellular). A cell receptor is responsible for accepting ligands such as amino acids and messenger molecules. Once the receptor has found a matching ligand it starts the *signal downstream* process by activating ERK2, a protein kinase intracellular signalling molecule.

ERK2 has two primary roles. The first, to continue suppressing the signal pathway component REG A until ERK2 itself is suppressed, and the second is to relay the incoming signal to the component ACA. ACA secretes cAMP back into the environment along with PDE (phosphodiesterase); a chemical compound used to degenerate cAMP. It also increases levels of internal cAMP to trigger activation of PKA (protein kinase A) after a given threshold is reached. PKA activates three biochemical processes; firstly the execution of gene expression leading to the release of calcium and ultimately the movement of the cell, secondly the disabling of ERK2, which in itself leads to the activation of REGA and

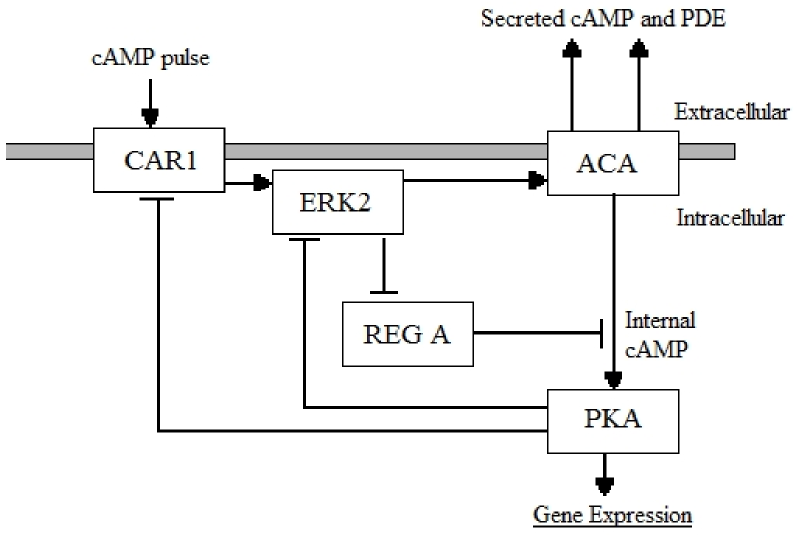

Fig. 1. The oscillating feedback loop of external and internal cAMP production and degeneration as presented in [11].

as such internal cAMP is hydrolyzed to lower internal levels, thirdly, the membrane receptor CAR1's ability to accept cAMP ligands is reduced. As [11] makes clear, the cAMP transduction sequence creates an oscillating feedback loop.

Gradually the cellular mass forms a slug, where the tip is constructed from prestalk cells and the body from prespore cells, ready to begin forming a standing slug aggregate. The slug is typically made from up to 100,000 cells (one in five are prespore, the others prestalk) and behaves as a single organism capable of both phototaxis and thermotaxis. The thermotaxis is the slug's ability to detect and move along a temperature gradient and phototaxis is the movement along a light gradient. It is enclosed in a sheath of muco-polysaccharide and cellulose and its tip is considered the control centre for specialised global development. In the third and final stage, as the slug settles into place, the prestalk and prespore cells switch place, with the prespore cells pushing past the prestalk to form a bulbous tip and the prestalk retreating back to form the stalk of the fruiting body allowing spore cells to disperse and become a new amoeba.

2.1 A brief summary of *D. discoideum* modelled using ODE

Although the proposed approach to modelling *D. discoideum* will be based on π-Calculus we will briefly outline some of the work involving modelling the amoeba using ordinary differential equations.

The mathematical models summarised by the publications of Dallon, Othmer, and Tang [7, 15, 20, 21] express the behaviour of chemotaxis via non-linear ODEs. The majority of publications on *D. discoideum* have used these equations to model cellular behaviour. By relying on their accuracy, biologists have been able to concentrate on various other areas of the *D. discoideum* life cycle such as cell differentiation [24]. In [24] we see that the transformation from a hemispherical mound into an elongated slug results from cellular movement in response to cAMP.

ODE models have been successful in capturing a significant set of cellular and extracellular behaviour in a population of *D. discoideum.* Not only should a model be concerned with biochemical reactions, it also needs take into account the environmental physics [15]. These physical properties include cell-cell and cell-substrate adhesion, interaction of cells through locomotive forces and resistance to cellular deformation.

Using a previously defined set of ODEs [7], the publication [15] looks at representing individual cells as deformable viscoelastic ellipsoids so as to implement a system that generates active forces, interacts via surface molecules, and can detect and respond to chemotactic signals. In the literature referenced in this paper the researchers spend significant time experimenting with models of cell locomotive forces to reveal the effects of movement speed on the formation of aggregate streams. The effects of using previously defined mathematical models in addition to physical forces can be seen in figure 2.

cAMP cellular threshold dynamics have been expressed with success using the FitzHugh-Nagum model [24]. The implementation provides a means of regulating the required cAMP threshold in the cell to activate chemotaxis before the variables oscillate back to a resting state [10]. The model helps identify the shape and dynamics of the *D. discoideum* mound through a process of cell sorting and differentiation.

3 Process-algebraic modelling of *D. discoideum*

Process calculi in biology can be seen as a shift in paradigm from the traditional numerically-intensive (and usually non-linear) ODE models to a more structured ontology. Process calculi's ability to model vast networks of parallel units lends itself very well to studying mass parallelism found in nature. Other discrete parallel distributed models exhibit the

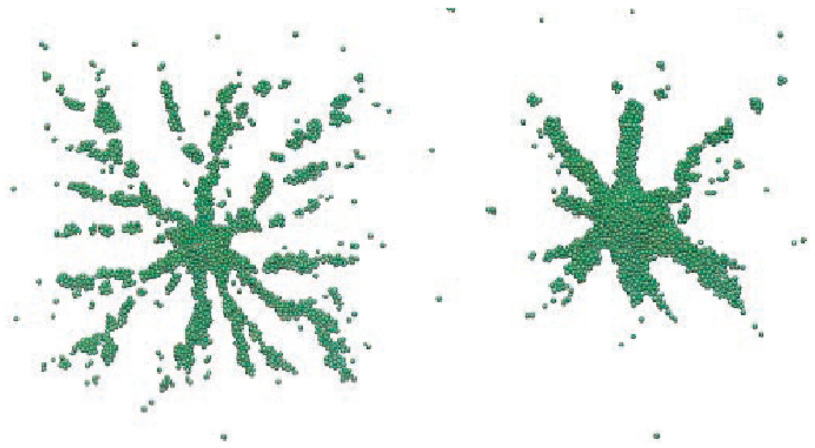

Fig. 2. Samples taken from [15], demonstrating the aggregation of cells into streams. As the system evolves over time, cells begin to clump together whilst those outside of relaying cAMP signals remain isolated.

same benefits; for example, Wolfram's work on cellular automata [25] documents this very well.

Many biological systems involve cyclic message-based operations; for example, oscillation of cAMP in *D. discoideum* [5]. For this reason, a variation on stochastic π-Calculus using graphical representation of biological cycles has been successfully implemented to model the behaviour of a MAPK signalling cascade [17].

Not all biological systems react in a deterministic manner, for example, parts of an immune system behave on conditional probabilities via the activation rate of lymphocyte according to the balance of cytokine to antigens in the blood stream [14]. This has been expressed using the probabilistic process calculi WSCCS [23].

A system of three genes uses negative feedback to mutually repress each other [18] has been successfully modelled in stochastic π-Calculus and implemented on SPiM (Stochastic Pi-Machine) [16]. SPiM uses a variation on the Gillespie algorithm to select reactions proportional to chemical reaction rates. In addition to SPiM, similar process calculus implementations such as BioSPI [19] have been used to model biological systems. A further modelling tool based on a process calculi is PEPA (Performance Evaluation Process Algebra), it is a stochastic formal language which allows the modelling of distributed systems. Finally, work in [6] has shown great success in running simulations of ERK signalling pathways.

Moving away from *D. discoideum* we see how stochastic process calculi is being used to create formal model representations of neurological processes [3, 4]. The formal models enjoy the freedom of direct implementation from mathematical notation straight into SPiM. The publications focus on particular neuron areas and synaptic processes; for example, [3] explores a process calculi representation of a presynaptic terminal along with a discussion on facilitation and depression.

It is evident from the literature reviewed on process calculi along with the biological examples how the modularity and dynamic capabilities of process calculus would warrant further study by biologists and computer scientists.

4 A proposed model framework

The following two sections give a brief example of π-Calculus modelling cAMP regulation via oscillating signal transduction biochemical reactions along with basic CA neighbourhood interactions. Please refer to appendix A for a short description of the syntax involved.

We begin our description of both intra- and inter-cellular parts of the model by first defining the names used throughout the formal model by the set notation in equation 1. Note, within the model a cell refers to a location in an environment/phase space. A phase space cell can either contain a chemical, a biological cell (referred to as *D. discoideum*) or can be empty.

$$\begin{aligned} N = \{ & a, c, absorb, release, activateACA, suppressERK, \\ & releasePDE, releaseCAMP, camplevel, campthreshold, cellTransition\} \end{aligned} \tag{1}$$

Table 1 gives a description to each element in the set N from equation 1.

4.1 Intra-cellular communication

The following section describes a few of the basic π-Calculus processes to modularise a single *D. discoideum* internal signal network occupying a single phase space cell. Our treatment broadly follows the other formalisations described in section 3.

Set member	Description
a	A channel used to bind together communication between the process CAR1 and the process CAMP. The two processes are then able to move to two different states simultaneously.
c	A channel used to bind together the processes ACA and PKA. By activating this channel, $\overline{suppressERK}$ channel is executed, consequently moving to a state containing the REGA process.
absorb	A channel which allows the CAMP process to behave as ERKPathway and the CAR1 process to behave as itself, i.e., CAR1 $\xrightarrow{\tau}$ CAR1.
release	A channel which upon execution allows the CAMP ligand to disengage from the CAR1 receptor.
$activateACA$	This channel facilities the ERK2 process binding with the ACA process.
$suppressERK$	Allows the transition to a REGA process state by the ACA process binding with ERK2.
$releasePDE$	A dummy process used to illustrate a possible state transition facilitating interaction with the surrounding environmental cells via the release of the chemical PDE.
$releaseCAMP$	A dummy process used to illustrate a possible state transition facilitating interaction with the surrounding environmental cells via the release of the chemical cAMP.
camplevel	An arbitrary π-Calculus name representing the internal level of cAMP.
campthreshold	An arbitrary π-Calculus name representing the quantity of cAMP required before PKA becomes excited.
campthreshold	An additional mechanism used to illustrate the action of transferring a chemical, signal or biological cell across two neighbouring phase cells.

Table 1. A list of π-Calculus names comprising of arbitrary names; e.g., *camplevel* and *campthreshold*, and actions; e.g., *absorb* and *release*. Actions *a* and *c* are used as channels to pass a set of channels between communicating processes. This is a strong feature of π-Calculus

The *D. discoideum* signal transduction is made up from a number of processes each of which is described below (a description on the individual biological components can be found under section 2). The composition of processes can be extended to include greater detail on the operations behind a *D. discoideum* amoeba (see section 5 for further work).

$$\text{LigandBond} \overset{def}{=} \{\nu a, \nu\, absorb, \nu\, release\}\, (\text{CAMP}|\text{CAR1}) \tag{2}$$

$$\text{CAMP} \overset{def}{=} a\,(absorb, release)\,.\,\left(\overline{absorb}.\text{ERKPathway} + \overline{release}.\text{CAMP}\right) \tag{3}$$

$$\text{CAR1} \overset{def}{=} \overline{a}\,(absorb, release)\,.\,(absorb.\text{CAR1} + release.\text{CAR1}) \tag{4}$$

The arbitrary parallel composition of the two processes CAR1 and CAMP in equation 2, represents the joining of a ligand to a cell receptor. The binding of the two processes in equation 2 can be expressed through the mutual channel a, which facilitates execution by giving each process a choice of being in either one of two states. The first choice, *absorb* will cause the state to transform from LigandBond to a parallel composition of ERKPathway and CAR1. The labelled state transition notation for this modification of state is LigandBond $\xrightarrow{\tau}$ CAR1|ERKPathway. The second choice, *release* causes the process to return to its original state.

$$\begin{aligned}\text{ERKPathway} \overset{def}{=} &\{\nu\, activateACA, \nu\, suppressERK, \nu\, lockpathway,\\ &\nu\, unlockpathway, \nu\, c\}\\ &(\text{ERK2}|\text{Pathway}|\text{PKA}|\text{ACA})\end{aligned} \tag{5}$$

The ERKPathway process can be decomposed into four components, each of which have mechanisms in the form of private channels to bind allowing internal communication between processes. The Pathway process in equation 6 allows a semaphore lock on the complete ERK2 pathway, i.e., from activation of ERK2 up to the suppressing of the CAR1 and ERK2 components. It works by preventing an existing ERK2 process from reactivating the ACA process. This is only visible after a complete run of the ERKPathway process along with another binding of CAMP.

$$\text{Pathway} \overset{def}{=} lockpathway.unlockpathway.0 \tag{6}$$

The ERK2 process immediately locks the process flow by using the *lockpathway* from equation 6. Having bound with the ACA process via the new operation *activateACA*, ERK2 is able to activate the ACA process. On the other hand, towards the end of a cycle the ERK2 process can be suppressed via the *suppressERK* channel, which in turn moves to state REGA.

$$\mathrm{ERK2} \overset{def}{=} \overline{lockpathway}.\left(\overline{activateACA}.0 | suppressERK.\mathrm{REGA}\right) \tag{7}$$

ACA as seen in equation 8 has the ability to move across a number of states. The arbitrary channels *releasePDE* and *releaseCAMP* are responsible for binding with the external cellular environment. This process has yet to be implemented and will be featured in future work (please see section 5).

One of the possible states which the parallel composition can transform to is the binding of the ACA process over a channel *c*. This is only possible if the *internalcamp* reaches the *thresholdcamp* value. Internal quantities of cAMP can be maintained via a basic counting mechanism.

$$\begin{aligned} \mathrm{ACA} \overset{def}{=} & \{\nu releasePDE, \nu releaseCAMP\} \\ & activateACA.\left(\overline{releasePDE}.0 | \overline{releaseCAMP}.0 \right. \\ & \left. | \text{if } internalcamp = campthreshold \text{ then } \overline{c}.\mathrm{ACA}\right) \end{aligned} \tag{8}$$

Once a significant quantity of cAMP has built up within the cell, PKA suppresses the ERK pathway by communicating with the ERK2 process to move to a REGA state via the *suppressERK* channel. The removal of cAMP from the inside of the *D. discoideum* cell would require an additional mechanism, along with the degeneration of cAMP in the external cellular environment from the collision of the PDE chemical.

$$\mathrm{PKA} \overset{def}{=} \{\nu geneExpression\}\ c.\left(\overline{suppressERK}.\mathrm{PKA} | geneExpression.0\right) \tag{9}$$

geneExpression suggests an arbitrary channel allowing further decomposition of *D. discoideum* cell to $\overline{\text{facilitate the}}$ intake of CA^{2+} causing cellular movement. The process $\overline{geneExpression}.0$ evolves to 0, which suggests that only one gene expression transition per activation of PKA is possible until the next successful build up of cAMP.

$$\mathrm{REGA} \overset{def}{=} \text{if } camplevel \neq campthreshold \text{ then } unlockpathway.0 \tag{10}$$

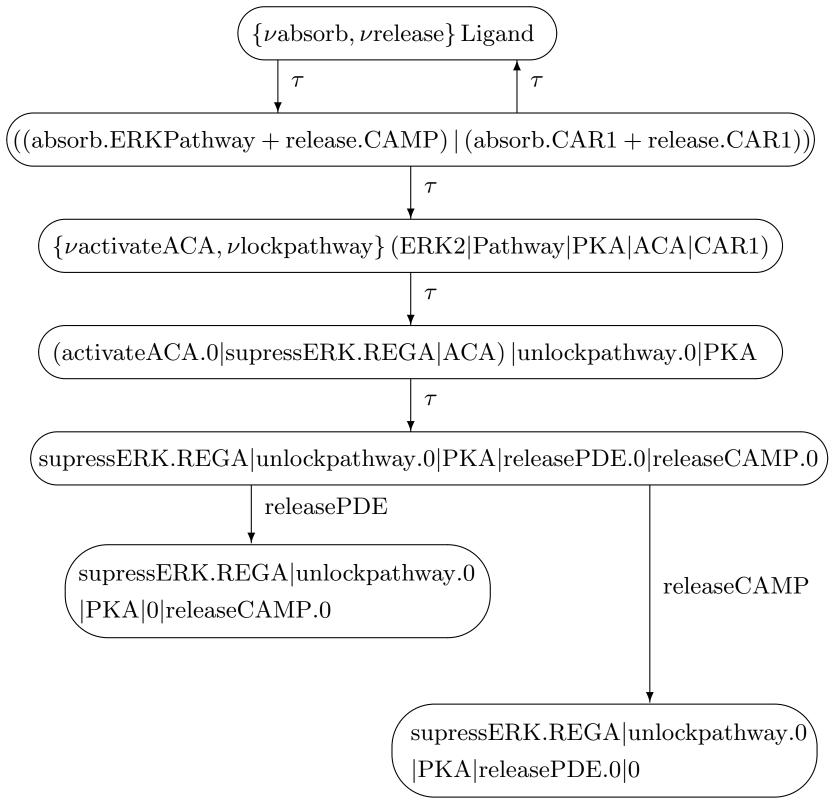
{νabsorb, νrelease} Ligand
τ
τ
((absorb.ERKPathway + release.CAMP) | (absorb.CAR1 + release.CAR1))
τ
{νactivateACA, νlockpathway} (ERK2|Pathway|PKA|ACA|CAR1)
τ
(activateACA.0|supressERK.REGA|ACA) |unlockpathway.0|PKA
τ
supressERK.REGA|unlockpathway.0|PKA|releasePDE.0|releaseCAMP.0
releasePDE
releaseCAMP
supressERK.REGA|unlockpathway.0 |PKA|0|releaseCAMP.0
supressERK.REGA|unlockpathway.0 |PKA|releasePDE.0|0

Fig. 3. A brief example of a transition state diagram illustrating some of the few initial steps involved in the *D. discoideum* signal transduction cycle. Between states is a labeled transition, a τ indicates an action silent only to those processes directly involved.

4.2 Inter-cellular communication and CA neighbourhood

The π-Calculus model is used to represent a discrete cellular automaton environment. Cells (as mentioned in section 4, unless explicitly defined, refers to a single space in the environment, not a biological entity) interact with their neighbours according to defined rules and a given neighbourhood structure. Cell-to-cell signals are biochemical, where each signal occupies a single cell and travels across the CA space in a motion similar to in-vivo chemical signals (please refer to section 5 for additional work).

$$\mathit{\Gamma} = \left\{ \begin{array}{lll} \text{(x-1,y-1)}, & \text{(x,y-1)}, & \text{(x+1,y-1)}, \\ \text{(x-1,y)}, & & \text{(x+1,y)}, \\ \text{(x-1,y+1)}, & \text{(x,y+1)}, & \text{(x+1,y+1)} \end{array} \right\} \tag{11}$$

The neighbourhood $\mathit{\Gamma}$ (illustrated in figure 4) required to detect the presence of cAMP is equivalent to the Moore neighbourhood. This reflects the notation that the *D. discoideum* amoeba is unable to sense a cAMP wave until the cAMP ligand binds to the CAR1 transmembrane receptor. The degeneration of cAMP by a phosphodiesterase uses the same neighbourhood, only reacting when the chemicals cAMP and PDE meet.

A single neighbouring cell is defined as

$$\text{NCell}_{(x,y)} \overset{def}{=} \overline{\mathit{cellTransition}}\ (\mathit{state})\,.\text{NCell}_{(x,y)} \tag{12}$$

The $\overline{\mathit{cellTransition}}$ is an arbitrary channel which allows communication between a given cell and its neighbours. Any number of additional channels can be added to cater for multiple signals from the same cell. Alternatively, it would be possible to send multiple names down a single channel with the polyadic π-Calculus [12]. The ordered pair (x, y) refers to the physical location within the neighbourhood. Neighbourhood coordinates of the current cell are defined by the set $\mathit{\Gamma}$. A pictorial representation can be seen in figure 4.

The centre cell surrounded by its neighbours will be referred to as the current cell; this can be seen in figure 4. Execution of a cell's neighbourhood occurs in parallel; this is defined as

$$\mathit{Neighbours} \overset{def}{=} \prod_{l \in \mathit{\Gamma}}^{max-n} \overline{\mathit{cellTransition}_l}\ (\mathit{state})\,.\text{NCell}_l \tag{13}$$

where max-n refers to the last of the cell's neighbours.

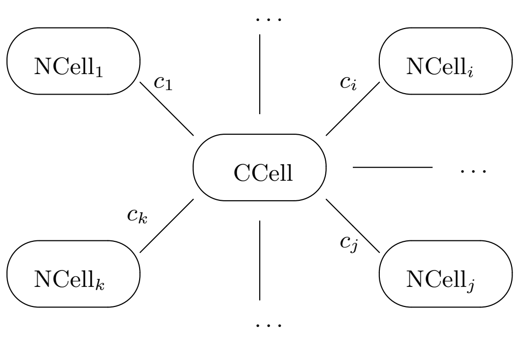

Fig. 4. A brief example of the cellular automaton environment. The centre environment cell can contain a single *D. discoideum* cell. Note how the layout of the environment can take the form of a neighbourhood from lattice gas or traditional cellular automata. This implies that the neighbourhood set Γ can also change according to the neighbourhood structure.

5 Conclusion

D. discoideum has been the subject of a lot of research by computer scientists over the years, but much of it has concentrated on *either* the formation of aggregates *or* the signalling pathways but rarely has there been a systematic attempt to combine both facets. In this paper we present our attempt at modelling the intracellular cAMP oscillation in π-Calculus and apply this formalisation to explain how the quorum sensing between cells can be achieved, thus capturing the complex, multi-scale characteristics of the system. At the moment each of the two aspects are represented in a simple way, but we hope to develop more detailed models, all the time mapping intra- and inter-cellular phenomena.

The core of the representation is the use of a cellular automaton grid to anchor the population of cells and capture the neighbourhood over which amoeba influence each other and, more importantly, the association of π-Calculus formulae with each location. We do not yet capture space in a very realistic way, or the movement of amoeba from one location to another. We plan to eventually move to a more sophisticated representation, such as lattice-gas cellular automata. The π-Calculus abstract representation of biological processes creates an open model allowing for further details in the mechanics behind *D. discoideum.* For example; the implementation of gene expression, which ultimately leads to the intake of calcium ions to encourage the amoeba to move (see section 2).

Once a sophisticated π-Calculus model has been established we intend to implement the framework on a high-throughput Condor machine [22]. Unlike [3] the authors of this paper intend to building their own software process calculi interpreter rather than using existing systems such as SPiM.

References

[1] J. Bergstra, A. A, Ponse, and Scott A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science Inc., New York, NY, USA, 2001.

[2] John Tyler Bonner. *Lives of a Biologist: Adventures in a Century of Extraordinary Science*. Harvard University Press, 2002.

[3] Andrea Bracciali, Marcello Brunelli, Enrico Cataldo, and Pierpaolo Degano. Expressive models for synaptic plasticity. *Computational Methods in Systems Biology*, 4695:152–167, 2007.

[4] Andrea Bracciali, Marcello Brunelli, Enrico Cataldo, and Pierpaolo Degano. Stochastic models for the in silico simulation of synaptic processes. *BMC Bioinformatics*, 9, 2008.

[5] Joseph A. Brzostowski and Alan R. Kimmel. Nonadaptive regulation of ERK2 in Dictyostelium: Implications for mechanisms of cAMP relay. *Molecular Biology of the Cell*, 17:4220–4227, October 2006.

[6] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Lecture Notes in Computer Science*, 4230:1–23, 2006.

[7] J. C. Dallon and H. G. Othmer. A Discrete Cell Model with Adaptive Signalling for Aggregation of Dictyostelium discoideum. *Royal Society of London Philosophical Transactions Series B*, 352:391–417, March 1997.

[8] P. N. Devreotes and S. H Zigmond. Chemotaxis in eukaryotic cells: A focus on leukocytes and Dictyostelium. *Cell Biology*, 4:649–686, 1988.

[9] Merkl R. Fisher, P. R. and Gerisch G. Quantitative analysis of cell motility and chemotaxis in Dictyostelium discoideum by using an image processing system and a novel chemotaxis chamber providing stationary chemical gradients. *Journal of Cell Biology*, 108:973–984, 1989.

[10] Peter Grindrod. *The theory and applications of reaction-diffusion equations : patterns and waves*. Oxford University Press, 1996.

[11] Michael T. Laub and William F Loomis. A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium. *Molecular Biology of the Cell*, 9:3521–3532, December 1998.

[12] R. Milner. *The polyadic pi-calculus: a tutorial*, pages 203–246. Springer-Verlag, 1993.

[13] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part i. *I and II. Information and Computation*, 100, 1989.

[14] Ral Monroy. A process algebra model of the immune system. *Knowledge-Based Intelligent Information and Engineering Systems*, pages 526–533, 2004.

[15] E. Palsson and H. G. Othmer. A model for individual and collective cell movement in Dictyostelium discoideum. *Proceedings of the National Academy of Science*, 97:10448–10453, September 2000.

[16] A Phillips. The stochastic pi-machine, 2006.

[17] Andrew Phillips and Luca Cardelli. A graphical representation for the stochastic pi-calculus. *Bioconcur'05*, August 2005.

[18] Andrew Phillips and Luca Cardelli. Efficient, correct simulation of biological processes in the stochastic pi-calculus. *Computational Methods in Systems Biology*, pages 184–199, 2007.

[19] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pac Symp Biocomput*, pages 459–470, 2001.

[20] Y. Tang and H. G. Othmer. A G protein-based model of adaptation in Dictyostelium discoideum. *Math. Biosci*, 120:25–76, 1994.

[21] Y. Tang, P. Schaap, and H. G. Othmer. A model for pattern formation in Dictyostelium discoideum. *Differentiation*, 61:141–141, 1996.

[22] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(2-4):323–356, 2005.

[23] Chris Tofts. Processes with probabilities, priority and time. *Formal Aspects of Computing*, 6:536–564.

[24] B Vasiev and C J Weijer. Modeling chemotactic cell sorting during Dictyostelium discoideum mound formation. *Biophysical Journal*, 76:595–605, February 1999.

[25] Stephen Wolfram. *A New Kind of Science.* Wolfram Media, January 2002.

A Appendix - π-Calculus

The following syntax is a short extract from a complete list of π-Calculus syntax [13]. A companion set of examples are also available [13]. We would also like to bring to the reader's attention [1] as a good source on π-Calculus and as an introduction to a number of π-Calculus adaptations.

A.1 Process Definition

The set of processes can be defined as:

$$\begin{aligned} P ::=& 0 \\ & |\bar{x}y.P \\ & |x(y).P \\ & |\tau.P \\ & |(x)P \\ & |\,[x = y]\,P \\ & |P|Q \\ & |P + Q \\ & |A\,(y_1, ..., y_n) \end{aligned} \tag{14}$$

A.2 π-Calculus actions

The following gives a short incomplete list of π-Calculus actions. Please note that we have used a slightly different syntax to help define an output channel. Rather than $\bar{a}x$ to denote a name x to be sent over channel a, we have surround x in brackets as such $\bar{a}\,(x)$. This helps keep the syntax clear due to the number of characters in some of the biological channel/process names.

- τ silent prefix - the silent prefix of a process P written as $\tau.P$ executes P with no visible action.
- $\bar{x}y$ free output - the free output of name y across channel x written $\bar{x}y.P$, outputs y before behaving as process P.
- $x(z)$ bound input - this process receives a name which substitutes over the place holder z across the channel x to then behave as $P\left\{\frac{w}{x}\right\}$. Please see reference [13] for a detailed description on binding of names and name substitution.
- $P|Q$ composition - the parallel execution of processes P and Q if each has an executable action prefix. Processes may act independently or share a channel in which to communicate across. Reference [13] contains a list of very clear examples of parallel communicate.
- $\sum_{i \in I} P_i$ (finite index set I) process summation - behaves like one of P_i. The binary equivalent is written as $P_1 + P_2$ where either P_1 or P_2 executes.
- $[x = y]P$ match - if an incoming name x is identical to the name y then process P executes. On a similar note the mismatch operator $[x \neq y]P$ executes P so long as the names x and y are not equal. In combination sophisticated conditional statements can be constructed.

A.3 Rules of actions along with examples

The following list of equations state the transformation rules (rules of inference) used throughout the framework proposal. A complete list of rules along with further examples can be seen in [13].

Tau action

$$\frac{}{\tau.P \xrightarrow{\tau} P} \tag{15}$$

Output action

$$\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P} \tag{16}$$

Input action

$$\frac{}{x(z).P \xrightarrow{x(w)} P\left\{\frac{w}{z}\right\}} \tag{17}$$

Sum

$$\frac{P \xrightarrow{\alpha} \acute{P}}{P + Q \xrightarrow{\alpha} \acute{P}} \tag{18}$$

Match

$$\frac{P \xrightarrow{\alpha} \acute{P}}{[x = y]\, P \xrightarrow{\alpha} \acute{P}} \tag{19}$$

Parallel

$$\frac{P \xrightarrow{\alpha} \acute{P}}{P|Q \xrightarrow{\alpha} \acute{P}|Q} bn\,(\alpha) \cap fn\,(Q) = 0 \tag{20}$$

Communication

$$\frac{P \xrightarrow{\bar{x}y} \acute{P}\; Q \xrightarrow{x(z)} \acute{Q}}{P|Q \xrightarrow{\tau} \acute{P}|\acute{Q}\left\{\frac{y}{z}\right\}} \tag{21}$$

Given a process P $\overset{def}{=}$ a.P, executing action a causes the process to remain constant. In terms of a labelled state transition diagram P $\xrightarrow{a}$ P, i.e., a process onto itself.

The next example demonstrates bound channels over parallel composition. Given the two processes P $\overset{def}{=}$ $\overline{a}$.Ṕ and Q $\overset{def}{=}$ a.Q́, along with the parallel composition $\{\nu a\}$ (P|Q), the restricted name a implies that a can only execute between the two processes P and Q. This would evolve as $\{\nu a\}\,(\mathrm{P}|\mathrm{Q}) \xrightarrow{\tau} \{\nu a\}\left(\acute{\mathrm{P}}|\acute{\mathrm{Q}}\right)$. The τ is a silent operator, only visible to the processes directly involved.

Where as the parallel composition allows a given state trajectory to maintain every state involved, each with a likelihood of evolving, summation evolves on a single state, and thus all other states involved in the operation are removed. For example, given $\mathrm{P} \overset{def}{=} a.\acute{\mathrm{P}}$ and $\mathrm{Q} \overset{def}{=} b.\acute{\mathrm{Q}}$ summing them together to form $\mathrm{P} + \mathrm{Q}$ provides two directions of state evolution. The first, $\mathrm{P} + \mathrm{Q} \xrightarrow{a} \acute{\mathrm{P}}$ and the second, $\mathrm{P} + \mathrm{Q} \xrightarrow{a} \acute{\mathrm{Q}}$. Notice how only the activated process continues on that given trajectory.

Equivalence Arguments for Complex Systems Simulations – A Case-Study

Teodor Ghetiu[1], Robert D. Alexander[1], Paul S. Andrews[1], Fiona A. C. Polack[1], and James Bown[2]

[1] Dept of Computer Science, University of York, York, YO10 5DD, UK
{teodorg,rda,psa,fiona}@cs.york.ac.uk
[2] University of Abertay, Dundee, UK
j.bown@abertay.ac.uk

Abstract. Complex systems are often simulated to provide a basis for research or analysis. However, complex systems simulation often fails to properly demonstrate that the constructed simulation is an adequate tool to support investigation of the system under study. To address this issue we adopt and adapt argumentation techniques traditionally used for safety critical systems (SCS). Here we present part of an on-going case-study in which these techniques are used to demonstrate that two different implementations of a complex system simulation are adequately equivalent. This is a first step in producing further simulations of the system under study, which will be shown to be valid models on which to explore particular ecological phenomena.

1 Introduction

This paper presents part of a case study that is using a principled approach to computer simulation of a complex system. The work is part of the CoSMoS project[3], which is developing a general framework for the simulation of complex systems using agent-based approaches. One of our long-term goals is to argue the validity of complex systems simulations against domain models that capture an explicit expression of scientific understanding. More generally, we want to present properly-evidenced arguments that one model is an adequate representation of another model, or a particular perspective on reality. Such arguments form the basis for discussion between the simulators and the domain experts, and capture the rationale for the simulation, in terms of both

[3] http://www.cosmos-research.org

the domain understanding of the science, and the engineering of the simulation – see [2, 19] for further discussion. We believe the ability to argue properties of a complex system simulation (such as equivalence or validity) is an important element of CoSMoS and any other similar approach.

In this case study, a first step is to re-engineer a simulation of intra-specific plant variation [6]. The existing object-oriented simulation, in C++, needs to be scaled-up to support the scientific research. Elsewhere, we discuss the use of occam-π for efficient, process-oriented parallel agent-based simulation [17]. A number of recent complex systems simulations [5, 23, 24, 29] have successfully used this programming language. Here, we will ultimately exploit parallelisation to distribute the occam-π simulations over a cluster of machines [22]. This will allow us to simulate larger numbers and variations of plant, and a greater range of environmental influences than is possible in a purely sequential implementation.

The re-engineering is supported by construction of an argument that the occam-π simulation is *adequately equivalent* to the original C++ simulation. We call this an equivalence argument. The description of this argument forms the main subject of this paper. The original C++ simulation was used in ecological research that has some credibility within its research community, and through the equivalence argument we can support a claim that our re-implementation should share that credibility. Thus, our definition of adequate equivalence must make a case that the simulations capture equivalent aspects of the scientific domain, rather than simply presenting evidence of the technical equivalence of the two programs. Our argument is acceptable if the scientists – here represented by the person who has overseen the C++ simulation effort, James Bown (referred to subsequently as *the scientist*) – accept the argument that we have captured equivalent aspects of the scientific domain.

The paper continues with a brief introduction to argumentation techniques and their relation to simulation validity, section 2. Section 3 then describes the plant ecology case-study. Section 4 considers what adequately equivalent means and shows how we can build an explicit, structured argument of adequate equivalence for the two simulations. Section 5 gives examples of the required evidence from the simulations, to support the argument presented in section 4. The paper concludes with a discussion, section 6, conclusions and proposals for future work.

2 Argumentation and Complex Systems Validity

Elsewhere, we summarise current scepticism about the ability of computer simulations to adequately support scientific research (see [18], and cited work, [7, 8, 15, 16, 30]). In [2, 19], we report on an immunological case study undertaken in conjunction with immunologists, in which we found that a systematic collection and exposure of assumptions, made by the immunologists in relation to the scientific domain and by us as modellers and simulation-engineers, helped the immunologists to understand the value and limitations of our simulations. This understanding meant that the immunologists could use even basic agent-based models to test their understanding and guide their laboratory experiments; the documented assumptions gave rise to new avenues of scientific research. Here, we follow a suggestion in [19], and turn to conventional techniques from critical systems engineering, to start the process of systematising the use of *arguments* to capture and analyse evidence and assumptions.

In critical systems engineering, arguments are used to demonstrate a *case* to regulators that a system has certain properties, most commonly properties related to safety. In critical systems, it is impossible to absolutely demonstrate properties such as safety; instead evidence is collected based on criteria such as use of accepted development practices, software, system and sub-system testing, mechanical analysis, past experience or cumulative usage outcomes, and field trials. The evidence is used to support an argument that the risk associated with the system is *As Low As Reasonably Practicable* (ALARP), within the operational environment for which the system is designed. A general approach to constructing and documenting safety cases can be found in Kelly [10], whose other published research includes a range of studies and applications of critical systems argumentation. For an example of safety case creation for a – hypothetical – complex system, see [1].

2.1 Argumentation in Safety Critical Systems

Early safety-critical systems were unregulated, and were potentially grossly unsafe [12]. Consequent deaths and damage costs from, for instance, industrial and vehicle accidents, led in time to regulation, part of which is usually *certification*. Potentially-dangerous systems are allowed if there is sufficient evidence that they would be safe to operate. For a long time, *evidence* was based on process – "I have followed good engineering practice, so my system is safe". This approach is unsatisfactory in many ways, not least of which is its limiting of engineers to use only approved processes, thus inhibiting innovation.

A significant improvement in safety management came with *product-based certification*. Independent regulators are appointed, who set the safety criteria that a system must meet, in terms of specific evidence requirements. Developers collect evidence, and tie it together by means of a structured argument known as a *safety case*. It is still possible to cite an approved process as evidence, but this evidence is relegated to an appropriately-subordinate role. A safety case is accepted or rejected based on independent review of its arguments and evidence. Acceptability is not an absolute, and can change over time, in the light of experience or new evidence. This presents an important parallel to scientific investigation, particularly in biological domains, where the understanding of complex natural systems is a developing area, with much debate and many competing theories.

2.2 Summarising the Structure of an Argument: GSN

Safety cases were conventionally presented as free text, which is easy to create and immediately readable, but hard to systematically review. As Kelly [10] notes, not all safety engineers are gifted writers, and free text safety cases are often ambiguous. Construction and review of cases is improved if the structure of the argument and evidence can be summarised, for example using the Goal Structuring Notation (GSN) [10, 31]. Existing examples and patterns for GSN are predominantly concerned with safety cases.

GSN is a graphical way to express argument structures. A GSN diagram shows a hierarchy from the top-level claim – a typical safety case might seek to establish that *The system is safe* – down through subclaims that support that claim (e.g. *The hazard 'loss of temperature control' will not occur*) and eventually to the evidence supporting those claims (e.g. *Software test results for component X show no faults*). Anybody using GSN is guided by the rules of the notation, which helps to avoid gross errors of logic.

It is important to understand that GSN as a notation is of limited value – it is the argumentation culture and the safety-case literature that gives it its power in the safety field. Similarly, it would be a culture of argued validation that would be most important in addressing the criticisms (noted above) of complex systems simulation for scientific research.

2.3 Adapting Argumentation for Scientific Simulation Validity

When a computer simulation is used in a scientific study, the user of a simulation needs to demonstrate the extent to which the computer simulation matches reality (and other models). Traditionally, these arguments have been, at best, informal discussions in papers and reports. This causes many problems. Evidence or detail is omitted, making it difficult to assess the validity of simulation results. In an attempt at clarity, many arguments are reduced to vacuous or partial claims. There is a need to improve the quality and presentation of validation arguments; GSN is an obvious candidate for constructing argument structures and recording the evidence that supports (or could support) the argument.

Whilst there is a range of work on the validity in simulation, for example [25, 26], we are not aware of any existing work on structured arguments of computer simulation validity.

In safety analysis, the safety properties and the case for safety are normally created and rehearsed by the developers before the argument is constructed and represented in GSN. There are few specific argument construction methods, and experience shows that, whilst a top-down approach is impractical because it requires oversight of the body of evidence before the top-down structure can be identified, a bottom-up approach risks losing sight of the point of the argument.

The argumentation that we require for simulations is somewhat different to safety case argumentation, in that we are constructing arguments in parallel to simulation development, and can use the top-down construction of the argument to guide development. Similarly, we do not have a regulator dictating what is and is not acceptable evidence, but instead we have a scientific collaborator who must be able to understand and review our argument. In this paper, the goal is to demonstrate that two simulations are adequately equivalent. Our argument proceeds by analysing and recording what we will accept as evidence of adequately equivalent. We then establish this evidence by systematic analysis, recording the result as a GSN argument structure. First, we briefly introduce the intra-specific plant variation domain and the existing C++ simulation.

3 The Example: Intra-specific Plant Variation Simulations

The work presented in this paper is the first phase of a case study to provide computer simulations to support extensions to the ecological research of Bown et al [6], based on their novel model of plant physiology and interactions, based on physiological traits.

In [6], computer simulation is used to demonstrate that defining plants in terms of a suitable set of traits yields results that are acceptable to the ecological community, for example, the model produces species-area and species-abundance distributions that have typical characteristic statistical signatures (curves) [20]. However, the existing simulations are limited in the number and complexity of components that can be modelled, even if the implementation and platform were fully optimised, because of the difficulty of distributing a C++ program.

3.1 Ecological Modelling and Plant Trait Models

Begon et al define ecology as the "scientific study of the distribution and abundance of organisms and the interactions that determine distribution and abundance" [4]. The "holy grail" of ecology [11] is to find general rules that relate environmental conditions, species characteristics and community composition.

To complement field experiments, ecologists attempt to capture observational patterns and behaviours in models. At one extreme, equation-based models (EBMs) focuses on characteristics of the plant population as a whole, while at the other extreme individual-based models (IBMs) that allow for some of the individual variations within and between species. IBM is the more appropriate technique for study of intra-specific variation, and has the advantage that IBM individuals can map directly to and from real plants, so biological understanding can be mechanistically reflected in computer models. However, a computer model cannot hope to express all the characteristics of a real plant. A popular ecological technique is to summarise the characteristics of a plant in terms of numerical traits, with much ecological research to establish the most appropriate traits and value-ranges. Traits typically characterise visible, phenotypical properties such as shoot height, as well as ongoing biological processes such as water uptake capacity. A good model has rich informational content built using traits whose validity is supported by the direct mapping to biological data.

Ecological research has shown that *trait trade-off* is important in explaining the distribution and abundance of ecological communities [28]. Computer-based IBMs that model plants in terms of traits play a key role in this research. However, the models do not always map well to research goals, and it has been shown that the identification and representation of traits has a significant influence on the simulation results [13, 21].

Trait	Description
Essential uptake	Amount of resources that a plant needs for normal development without reproduction
Requested uptake	Amount of resources that a plant will request to support development and reproduction.
Spatial distribution of uptake	Uptake capacity of a plant with respect to the distance
Compartment partition	Resource allocation ratio of structural compartment to structural store
Structural store release proportion	Proportion of structural store that can be released
Surplus store release proportion	Proportion of surplus store resource that can be released
Time dependent reproduction	Time needed before initiating reproduction.
Development dependent reproduction	Resource level needed to initiate reproduction
Storage/fecundity relation	Ratio of the resource available for reproduction to the resources necessary for creating a seed
Seed dispersal pattern	Radius of the area of local seed dispersal
Survival threshold	Minimal resource level for plant survival
Survival assessment period	Number of consecutive timesteps over which the resources level can be below the survival threshold before the plant dies

Table 1. Bown et al's twelve plant traits [6]

3.2 The Computer Simulation of Bown et al

The intra-specific plant variation models of Bown et al [6] uses an IBM based on a resource-centric physiological scheme [27]. The model allows the study of the relationship between trait trade-off and the distribution and abundance of species.

Firstly, Bown et al [6] establish twelve traits (table 1) that adequately describe plant physiology. The plant species is described by a set of twelve distributions, one for each of these traits. The distributions determine the probability of each trait value across the set of plants, with individual trait values assigned to achieve the species distribution. This approach gives appropriate intra-specific variation.

In the model of Bown et al [6], a plant individual is modelled as a *phenotype* and a *genotype*, figure 1. The phenotype consists of appropriate representations of the resource storage and usage of a plant: the structural compartment represents resources corresponding to the

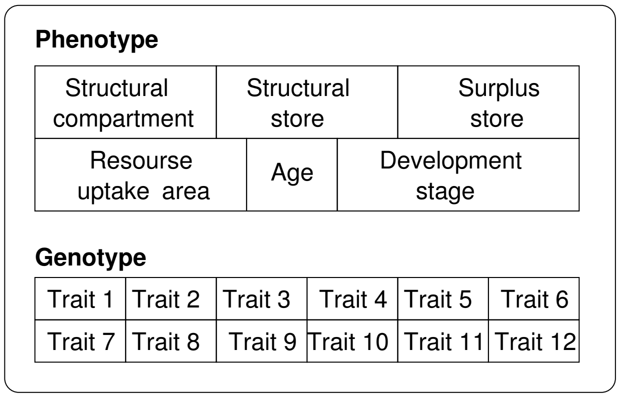

Fig. 1. Bown et al's model of an individual plant [6]

plant's fixed structure; the structural store holds resources that are used for reproduction; and the surplus store represents any excess of resource-uptake over the level essential to maintain the plant. In addition, the phenotype records age and development stage. In the genotype, a value is assigned to each of the twelve species traits, using a random sampling of the trait distribution to introduce intra-specific variation. Trait value distributions were obtained from field observations of the *Rumex acetosa* plant species [3].

Four biological processes drive the generic life-cycle of a plant: resource uptake, resource allocation, reproduction and development, as shown in figure 2. In the model, each plant takes up resources from the environment and allocate it to the three resource components of the phenotype. As resource is accumulated, the plant develops, which is denoted by incrementing the Development stage in the phenotype. Four of the trait values are related to a plant's development stage: spatial distribution of uptake, development dependent reproduction, and the two uptake traits.

There is an initial population of plants. When a plant reproduces the distribution of seeds is controlled by the seed dispersal pattern trait. A seed is only viable if it lands at a valid location that does not contain a plant. In [6], reproduction is clonal, so a seed has the same trait values as its (single) parent plant. The Reproduction process may be triggered according to the trait value for time dependent reproduction or for development dependent reproduction.

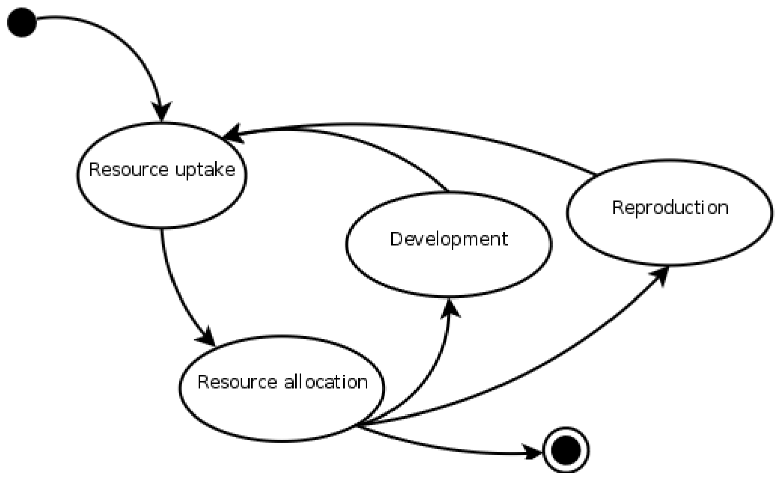

Fig. 2. State machine model of the biological processes of Bown et al [6]: ellipses represent the states of the plant associated with each biological process, and arrows represent possible transitions between theses states; the plant is created in the Resource uptake state and must be in the Resource allocation state when its death is determined

The environment is represented by a single type of resource, which is distributed evenly across its surface. The resource level has an upper limit defined by a saturation level. The flow or resource to plants is constrained by release and replenishment rates, which specify the maximum quantity of resource that can be released or added to the environment at any time. In the computer simulation, the environment is modelled on a two-dimensional grid. Bown et al [6] note that a cell represents an area of approximately 100cm^2, which, in the model, can be occupied by at most one plant. The number of plants that take resource from a cell is determined by the location of each plant and its root area, as represented by the spatial distribution of uptake trait. Grid cells contain a resource substrate, which is parametrised by the saturation level and the release and replenishment rates.

A timestep in the simulation represents one day in the real-world. Accordingly, the values that are used for parametrisation of the model reflect the resource flow through a plant during one day [6].

In order to compare the trait-model intra-specific results to inter-species distribution results, Bown et al [6] introduce 75 individual plants, which are treated as representing 75 different species. Because the model uses clonal reproduction, these 75 species either persist and increase in numbers, or die out. A simulation run lasted for 50 000 timesteps, which corresponded to around 1250 generations of plants. The simulation was run over different environment sizes (grids of 10×10 up to 50×50 cells) to collect statistics on the relationship of the size of the environment to the number of species co-existing (the species-area curve), and to the abundance of each species (the species-abundance curve).

3.3 The C++ and occam-π Simulations

Bown et al [6] use a mechanistic model of plants through which community level processes can be studied. We have re-implemented the simulation in occam-π, but in order to use our simulation to scale up the original experiments, we need to show that the new simulation is still based on the same underlying biological model.

The C++ simulation code is sequential, running on a single thread of execution. The model uses two passes per timestep to reduce sequential bias. For example, for resource uptake, all plant demands are made in the first pass, then, in a second pass, each plant receives a normalised percentage of the quantity it requested – where the total demand on a grid cell is more then the cell can release then each demand is reduced accordingly. The limitation of running on a single thread constrains the size of the environment and population that can be used in this simulator, which cannot handle the real-world scale of several hectares containing millions of plants.

A traditional re-engineering approach would create an abstract model of the data and processing implemented in the C++ simulation, and then re-develop this model in occam-π. This would, in theory at least, allow formal refinement relations to be established between each implementation and the abstract model, and a formal proof of equivalence. In practice, whilst model-driven engineering provides semi-formal transformation approaches to move between object-oriented models at different levels of abstraction, the potential for formal refinement between abstract models and object-oriented code is limited. Furthermore, having extracted an abstract model from the object-oriented code, there is no established way to refine this model into the process-oriented occam-π language – occam-π is formally underpinned, but by CSP [9], an event-driven formal language.

If a formal approach were to be found, it could establish a measure of equivalence between the implementation codes of the two simulations, but would not allow the re-engineered version to take full advantage of the strengths of occam-π. Most significantly, here, the re-engineered occam-π simulation can represent plants and locations as individual

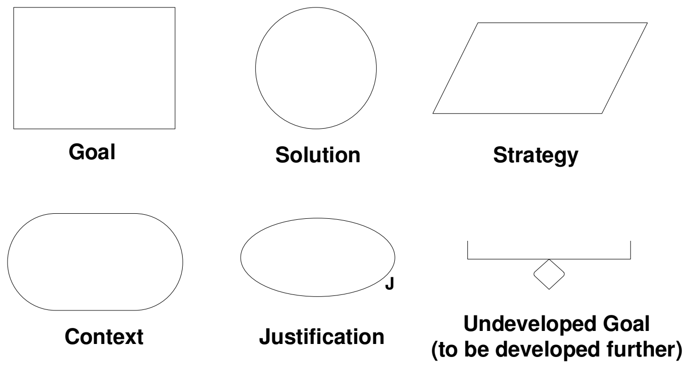

Fig. 3. GSN notations used in the equivalence argument: explanations are given in the text description of the arguments that follow

occam-π *processes*, each having its own thread of control. The occam-π processes communicate through *channels* through which data can be passed. This gives a closer mapping between the implementation and the biological reality than was evident in the C++ simulation.

4 A Structured Argument for Adequate Equivalence

This section works through the argument of adequate equivalence constructed for the C++ and occam-π implementations. For simplicity, we will refer to the C++ implementation as C, and the occam-π implementation as O. The argument is presented in GSN, using the standard notations, given in figure 3. The meaning of these symbols in our work is elucidated in the description of the argument that follows.

Note that the equivalence argument does not attempt to address the rationale or engineering of the C++ simulation – this is an established system that we cannot change. We do not compare the performance of the two implementations, as the motive for the re-engineering is not any immediate performance gain, but the distribution potential of the occam-π simulation across computer grids [24], with the efficient management of processes and events [32].

4.1 The Top Goal

A GSN argument starts with a *top goal*. In figure 5, this is shown as the rectangle labelled OCEquiv – O simulation is adequately equivalent to C simulation. This is the claim that we want to make, and the whole argument below is devoted to making that claim. In the diagram, lines with solid arrowheads connect each goal to lower-level components that together meet the goal.

A goal exists in a context. In figure 5, DefAdEq labels a context node, here promising that a definition of *adequately equivalent* is given elsewhere – in fact, the definition is given and explained in this section of the paper.

It is hard to definitively define equivalence. Structures in different languages may be syntactically different but semantically equivalent, or *vice versa*; we may have behavioural bi-similarity from different structures, or, since we are modelling complex systems, we may observe different results from similar initial conditions even within the same implementation. Despite this we need a definition of what we mean by *adequate equivalence* in order to argue convincingly about it.

We therefore propose that:

> *the two simulations are adequately equivalent if they produce the same results over the whole range of concern.*

In common with most analyses of complex systems, *same results* can be defined by statistical analysis – we run each simulation many times, and collect the results. This gives a distribution for each result. We then use an accepted statistical test (usually a non-parametric test that medians and inter-quartile ranges represent the same distribution at some confidence level) to determine whether the results can be considered equivalent.

The *range of concern* is defined by ranges for parameters over which the equivalence should hold. In the plant simulations, this relates to the range of environment sizes and initial plant numbers. Note that, because we cannot execute the C++ simulation on very large populations, we can only consider equivalence within the range of this simulation. Instead, we present direct comparisons of results within the range of the C++, and theoretical arguments for the rest of the range. The comparison of the results gives us high confidence within part of the range, while the theoretical arguments give us some confidence, but at a lower level, beyond that. This is represented figuratively in figure 4.

Note that the crucial factor in determining whether the definition of *adequately equivalent* is sufficient is a discussion with the scientists.

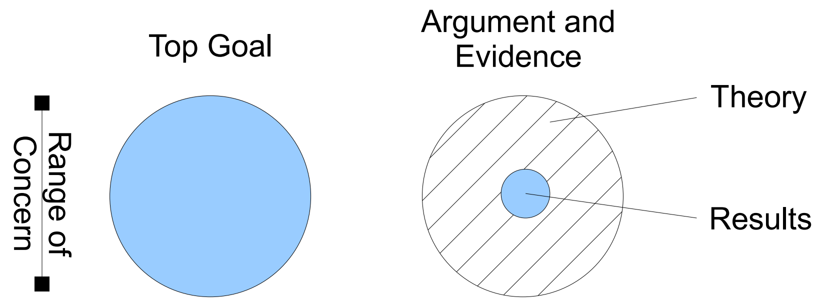

Fig. 4. Range of concern for arguing *adequate equivalence*: we wish to be convinced over the whole range of both simulations (the Top Goal), but we can only produce results evidence for part of the range; in the rest of the range we rely on other forms of evidence

Thus, in our case study, we consult the scientist directly; since he considers that our definition is sufficient, we can proceed. It is, of course, possible that this initial acceptance may be reversed when the evidence is complete and the whole argument presented – perhaps the scientist can demonstrate that our non-parametric tests of statistical equivalence are inappropriate, or our theoretical arguments are flawed, or perhaps we find that there are bugs in one of the simulations that affect the comparability of the results in other ways. The dialogue to establish the definition and associated argument is essential in the establishment of trust and understanding between simulators and scientists [2, 19].

4.2 Decomposing the Top Goal

Having agreed a top goal and the definition of the key terms that it uses, we need to provide an argument that the goal is satisfied. In figure 5, the top goal OCEquiv is met by following the ArgSciImplRes *strategy*. A strategy in an argument explains the connection between a goal and its sub-goals. Here, ArgSciImplRes states that we argue over three distinct areas – the underlying science, the details of how the simulations are implemented, and the actual results that they produce. The relationship here is complementary – each child goal gives us some confidence that the parent goal holds, and together, they give us *adequate* confidence that the goal is met.

Note that the three-goal sub-argument in figure 5 is not an alternative definition of what it means for two simulations to be adequately

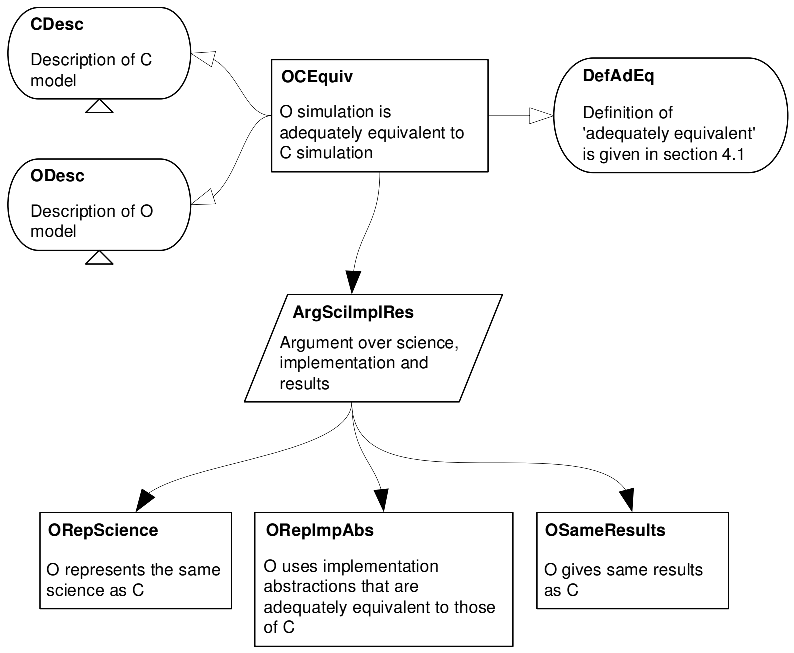
CDesc
Description of C model
ODesc
Description of O model
OCEquiv
O simulation is adequately equivalent to C simulation
DefAdEq
Definition of 'adequately equivalent' is given in section 4.1
ArgSciImplRes
Argument over science, implementation and results
ORepScience
O represents the same science as C
ORepImpAbs
O uses implementation abstractions that are adequately equivalent to those of C
OSameResults
O gives same results as C

Fig. 5. Top level of the argument that the C++ (C) and occam-π (O) simulations are adequately equivalent

equivalent. Rather, it is an approach to substantiating such a claim. We are using the three-legged argument to support a claim that the results will be the same across the whole range of concern.

The text in the GSN goal boxes is necessarily terse, and refers to concepts that need to be defined, as in the above discussion of *adequately equivalent*. It is hard to provide compelling contexts and definitive definitions. This is seen as a benefit, not a cost, of making structured arguments – you get to see where your definitions are vague or unsatisfying. (It is also much easier to see when another person's arguments are weak.)

In GSN, an upward triangle beneath a context box means that it has yet to be instantiated – it is a placeholder for concrete content that is not yet available. In figure 5, the CDesc and ODesc context boxes could be instantiated by a reference to the code of the simulations, to common abstractions such as figure 2, above, or to summary text such as the descriptions in section 3. The argument is not complete until this instantiation is performed.

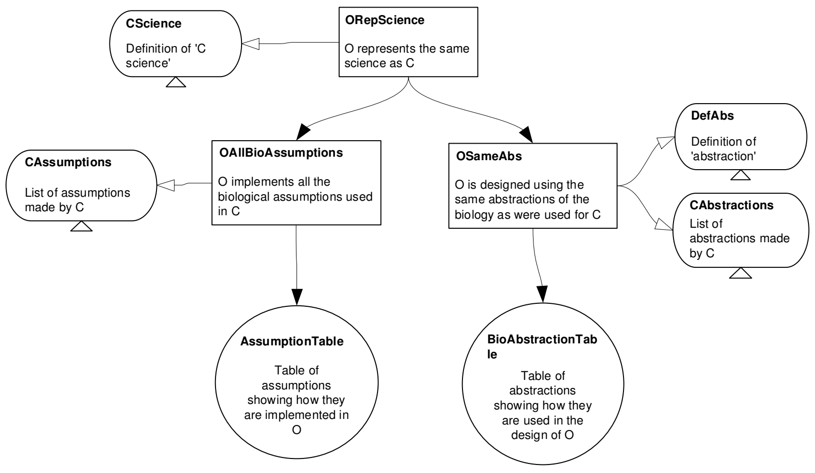
CScience
Definition of 'C science'
ORepScience
O represents the same science as C
DefAbs
Definition of 'abstraction'
CAssumptions
List of assumptions made by C
OAllBioAssumptions
O implements all the biological assumptions used in C
OSameAbs
O is designed using the same abstractions of the biology as were used for C
CAbstractions
List of abstractions made by C
AssumptionTable
Table of assumptions showing how they are implemented in O
BioAbstractionTable
Table of abstractions showing how they are used in the design of O

Fig. 6. Elaboration of the sub-goal to show that the simulations represent the same science, from figure 5

Again, the point of GSN arguments is not to demonstrate with absolute certainty that the top goal holds, but to demonstrate why the author of the argument believes that it is holds. The reviewer can disagree with the assumptions, strategy, and eventual evidence, and can challenge the author to find a better argument. Here, for instance, our scientist may dispute the strategy of arguing over three distinct areas, or may dispute the totality of these complementary areas, and challenge the author to make better justifications for its argument.

The three lowest-level goals shown in figure 5 are expanded in figures 6 to 8. Each of these argument fragments terminates in a circular *solution* node. Solutions refer to the *evidence* that supports a claim. In very simple arguments, evidence might directly support the top goal, but in practice, such intermediate sub-goals and strategies are needed to create a compelling argument. The following sections consider each of the three sub-goals in turn.

4.3 The Science Goal

In figure 6, the goal, ORepScience, is shown as being solved by two further goals. OAllBioAssumptions presents an argument that the occam-π version is based on the same assumptions about the actual biology as the C++ version. OSameAbs argues that the occam-π simulation abstracts

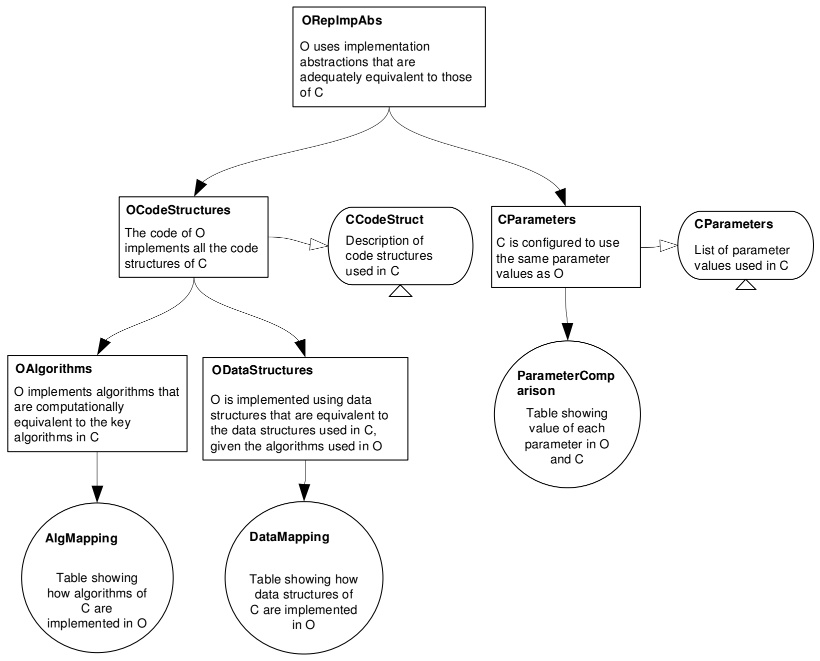
ORepImpAbs
O uses implementation abstractions that are adequately equivalent to those of C
OCodeStructures
The code of O implements all the code structures of C
CCodeStruct
Description of code structures used in C
CParameters
C is configured to use the same parameter values as O
CParameters
List of parameter values used in C
OAlgorithms
O implements algorithms that are computationally equivalent to the key algorithms in C
ODataStructures
O is implemented using data structures that are equivalent to the data structures used in C, given the algorithms used in O
ParameterComparison
Table showing value of each parameter in O and C
AlgMapping
Table showing how algorithms of C are implemented in O
DataMapping
Table showing how data structures of C are implemented in O

Fig. 7. Elaboration of the goal to show that the simulations represent the same implementation abstractions, from figure 5

from the details of the biology in the same way as the C++ version. Again, we expand the goals by providing context. From these goals, we directly reach the evidence required, with solutions pointing to tabular comparisons.

We could expand the argument, and the GSN, further to argue over each compared assumption or abstraction, providing a specific argument that each pair is adequately equivalent. This might be necessary if scientist found the comparison tables unacceptable without further evidence.

4.4 The Implementation Goal

The second child goal in figure 5, ORepImpAbs, is expanded in figure 7, with new sub-goal relating to the adequate equivalence of the code structure (OCodeStructures) and parameters (OParameters) in the two simulations. The reasoning here is that the simulation implementations are adequately equivalent if they run equivalent algorithms on equivalent data structures and use the same parameter settings.

OParameters is solved directly by a table comparing parameters in the two simulations, whilst OCodeStructures is further decomposed into a claim about algorithms and a claim about data structures. Each of these is, again, solved by a table that compares key elements of the two implementations.

Again, despite the appearance of precision provided by the GSN notation, much of the argument here is still implicit and left to the reader to infer. For example, it is implicit that equivalence of code structures and equivalence of parameters is sufficient to argue equivalence of implementation. Similarly, the argument that two algorithms in the table are equivalent is not made explicit. A software expert could verify or refute our claims, by whatever means they chose, but the non-expert must take our assertions on trust or ask for a further level of argument.

Also note that although the top-level goal talks about equivalence in terms of a black box that produces results, the argument here is white-box – we talk about how the simulation works internally. We are using white-box methods to support a claim expressed in black-box terms. This is similar to software testing, where it is common to combine white-box and black-box methods.

4.5 The Results Goal

The third child goal in figure 5, OSameResults, is decomposed, in figure 8, into claims relating to the testing and experimentation on the two simulations.

OCBoundaryCases claims that the two simulations provide the same results for boundary and extreme cases within the valid range. This is based on a common testing strategy, to establish that unusual situations are properly managed. We have not developed this goal yet, as is shown by the diamond beneath the goal box. To develop it, we need to consider what cases to test, in terms of the parameter and value settings that characterise each case – for instance, we may test both simulations on the case where all plants are the same, in order to check that clonal reproduction is implemented similarly; we might then check the behaviours that result with very small and very large initial numbers of plants (starting with the same plant populations), then look at the effects of extreme environments. Unless equivalence were obvious – in a very poor environment, we might be able to see that all plants died as soon as the minimum time (trait survival assessment period) had elapsed – in all cases, we would be using statistical analysis to determine acceptable equivalence of the results, as described above.

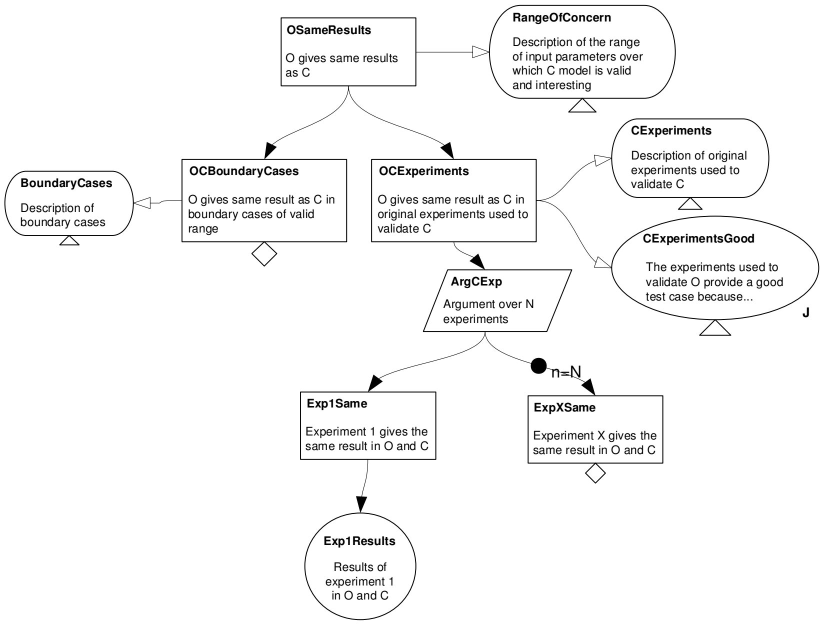
OSameResults
O gives same results as C
RangeOfConcern
Description of the range of input parameters over which C model is valid and interesting
CExperiments
Description of original experiments used to validate C
BoundaryCases
Description of boundary cases
OCBoundaryCases
O gives same result as C in boundary cases of valid range
OCExperiments
O gives same result as C in original experiments used to validate C
CExperimentsGood
The experiments used to validate O provide a good test case because...
J
ArgCExp
Argument over N experiments
n=N
Exp1Same
Experiment 1 gives the same result in O and C
ExpXSame
Experiment X gives the same result in O and C
Exp1Results
Results of experiment 1 in O and C

Fig. 8. Elaboration of the goal to show that the simulations produce equivalent results, from figure 5

OCExperiments states that, when the simulations are set up to replicate the same experiments (e.g. same environment and plant population, same trait and resource distributions), the results are the same – again using statistical analysis to determine equivalence.

As OCExperiments is critical to our argument, we expand the goal further to argue under the strategy of result similarity from n experiments (ArgCExp) – we could add a context here, that n represents the specific experiments conducted on the C++ simulation, as reported in the literature. Below ArgCExp, experiments are enumerated – here using the GSN version of ellipsis for brevity. We are showing that each entry in some list has been considered, and evidence produced.

The whole argument fragment in figure 8 is in the context of RangeOfConcern. This returns to the point made in defining *adequately equivalent* for the top goal, that there is a range over which we can produce equivalent results, and that, in this case, we can only claim that the two simulations are equivalent when performing the type and scale of experiments for which the C++ simulation was originally designed.

In figure 8, CExperimentsGood is a *justification* node – shown by a **J** next to the node. When expanded, it identifies a justification of why we can assume that OCExperiments supports OSameResults.

5 Solution Data

The previous section summarises the argument of adequate equivalence which we are making, and which we present to the scientist for review and external scrutiny. We now consider some of the evidence, or solutions, that support the argument.

Most of our argument of adequate equivalence points to tabular comparisons. We briefly cover two of the biological aspects, but then focus on structural comparison from the implementation argument structure, which raises most of the interesting issues of equivalence. The generation of evidence for the argument of *adequately equivalent science* is, in general, more interesting, and the establishment of this argument will be essential when we extend the simulation to support further experiments on the intra-specific plant variation. However, for the argument of simulation equivalence, the science has already been captured by J. Bown in constructing the original C++ simulation (and reviewed by the scientists with whom he was working). We have essentially one source, Bown et al [6], and, throughout, we refer to an interpretation of it that is directly expressed in the C++ implementation.

5.1 Biological Assumptions

Biological assumptions were not explicitly identified in the body of work represented by Bown et al [6]. However, we have had to identify some assumptions in order to complete the re-engineering, and can use these to strengthen the argument of equivalence. Table 2 lists some of the assumptions that form the context CAssumptions in figure 6. These have been confirmed by the scientist, giving us confidence that the occam-π simulation captures the assumptions on which the C++ simulation was based.

5.2 Biological Abstractions

Between biological facts and assumptions and the construction of computer simulations, we make various abstractions to map the real world into the computational one. The abstractions are influenced by the platform on which the computer simulation is built, as well as subjective factors. To expand the CAbstractions context in figure 6, we collect the

	Environment Assumptions
1	The soil properties do not change radically in time.
2	The environment can be seen as a plain. Various three-dimensional landscapes will not affect the outcome.
	Plant Assumptions
3	The uptake area of a plant can be considered conic.
4	The tap root is generally more developed than the fine roots.
5	The ratio between resource allocation towards growth and towards reproduction varies slowly in time.
6	Germination takes place in no longer than one day.
7	Plants develop unhindered, if having necessary resources.
8	Plants release their resources back into their environment, when they die.
9	Plants may die of starvation or due to unpredictable events.
10	Seed dispersal happens over a short period (a matter of days).
11	Each seed requires a similar amount of resource.
12	Seeds that fall in populated areas, most often do not germinate.

Table 2. Expanding CAssumptions – some of the assumptions made in the C++ model [6], and mirrored in the occam-π simulation

abstractions made by Bown et al [6], some of which are listed in table 3). We then checked that the occam-π simulation respects each of these abstractions.

5.3 Algorithm Mapping

To compare the algorithms of the two simulations, a sub-goal of OCodeStructures in figure 7, we present pseudo-code summaries and check subjectively for similarity. Figure 9 gives a pseudo-code overview of the two simulations, whilst figure 10 focuses on the algorithm for resource uptake. Note that the pseudo-code for the occam-π implementation is written to facilitate comparison with the C++, rather than in a way that native occam-π programmers would use.

The sequential C++ implementation has a centralised architecture. This requires loop-iteration over, for example, all instances of `location` and all `plant` individuals. Because occam-π is a parallel language, all the occam-π processes (plants, locations) could execute in parallel, shown in figure 9 as `each individual` and `each location`.

In the C++ model, a double-pass approach is used to reduce positional biases – resource uptake and usage are separated into two phases, otherwise subsequent behaviours such as seed dispersal would take place in the order in which plants are iterated. In the occam-π simulation,

	Environment Abstractions
1	Resource release and replenishment rates are constant.
2	The environment is 2D and each grid cell can hold only one plant.
3	The maximal level of resource is homogeneous across the environment.
	Plant Abstractions
4	Requested uptake is homogeneous with respect to the distance from the plant.
5	The uptake area has a regular shape and is not affected by neighbouring competitors roots.
6	The ratio between resource allocation towards growth and towards reproduction, does not vary in time.
7	Germination is instantaneous (takes only one time step).
8	When they die, plants release all of their resources into the environment.
9	Plants die of random events and starvation.
10	Reproduction is instantaneous (takes only one time step).
11	Each seed requires an identical amount of resource.
11	Seeds die if cells are occupied, otherwise they become plants.
12	Reproduction is clonal.

Table 3. Expanding CAbstractions – some of the abstractions made in the C++ model [6], and mirrored in the occam-π simulation

C++ simulation

```
instantiate locations
instantiate individuals

 foreach timestep
  /* resource uptake */
  foreach location
    assess resource demand
    release resources
    replenish substrate

  /* resource usage */
  foreach individual
    allocate uptake
    assess death
    if not dead
      assess development
      assess reproduction
```

occam-π simulation

```
instantiate locations and servers
instantiate individuals

 while simulation_running
  /* resource uptake */
  each individual
    place resource demand
    SYNCHRONISE
  each location
    process resource demands
    replenish substrate

  /* resource usage */
  each individual
    allocate uptake
    assess death
    if not dead
      assess development
      assess reproduction
```

Fig. 9. Comparing the C++ and occam-π simulations

Resource uptake (C++)
Sequential algorithm

```
Main control loop:
foreach location
 select uptake_area of location
 create empty demand_list
 foreach loc in uptake_area
   if loc occupied
     select occupying plant
     calculate plant demand on
                      location
     add demand to demand_list
 normalise demand_list
 foreach demand in demand_list
   select demanding plant
   add resources to plant uptake
 replenish location's substrate
```

Resource uptake (occam-π)
Parallel algorithm

```
Plants:
foreach location in uptake_area
  send resource request
SYNCHRONISE
foreach location in uptake_area
  if resources released
    uptake resource

Locations:
create empty demand_list
while running
  if resource request received
    store request
  if all requests received
    normalise demand vector
    foreach d in demand_list
      select demanding plant
      send resources to plant
    reset demand_list
    replenish substrate
```

Fig. 10. Comparing resource uptake algorithms for the C++ and occam-π simulations

synchronisation means that the plant processes will be blocked until all have finished sending their resource requests, when all processes will be released to proceed to resource uptake.

In reviewing the complete comparison of the high-level algorithm, we found that, in terms of semantics and results, the two implementations can be considered equivalent. The resource flow is identical; only the architecture through which it is carried out differs.

The second pseudo-code comparison, figure 10, refers to the process of resource uptake. In the C++ implementation, resource uptake is location-centric – the neighbourhood of each location is scanned for plants and the demand of each plant is calculated and stored. A normalisation process is necessary to divide the resource fairly among the plants. In the occam-π implementation, however, the process relates more closely to the biology, as each plant interacts directly with its location. The computational abstraction is, in this case, that of plants and locations interacting through a *client-server protocol* [14].

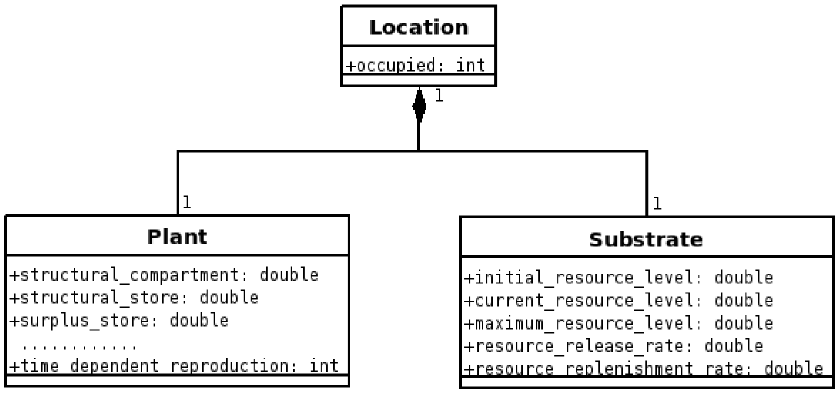
Location
+occupied: int
1
1
1
Plant
+structural_compartment: double
+structural_store: double
+surplus_store: double
....
+time_dependent_reproduction: int
Substrate
+initial_resource_level: double
+current_resource_level: double
+maximum_resource_level: double
+resource_release_rate: double
+resource_replenishment_rate: double

Fig. 11. Class diagram (UML notation) of classes in the C++ simulation

In this case, the algorithm comparison shows that, although the input and output of the algorithms is equivalent, the detail is different. To make a strong equivalence argument, we would need also to look at evidence of resource uptake behaviours through experimentation and testing.

5.4 Data Mapping

The second sub-goal of OCodeStructures in figure 7 concerns the argument of adequate equivalence of data representations. We can explore this similarity starting from a class diagram of the C++ implementation, figure 11, and a similar diagram of the occam-π processes and channels, figure 12.

UML provides an object-oriented modelling notation which is well-adapted to expressing the class structure of C++, but the notation of figure 12 is just an *ad hoc* representation of occam-π processes and channels. However, informally, we can compare data types between the two diagrams. As in earlier argument fragments, we present the evidence at this level for review; we only need to elaborate the comparison if the scientist is not prepared to accept it.

The C++ implementation of Bown et al [6] uses the class Location to represent represents cells of the environment. Each location contains a resource substrate, of class Substrate, and a plant individual, of class Plant. Because plants do not move, a Location instance is represented as being composed of one Plant instance and one Substrate instance. The Location attribute, `occupied` takes the value `1` if there is a plant growing at a location, and `0` when a location is empty – in the C++ an

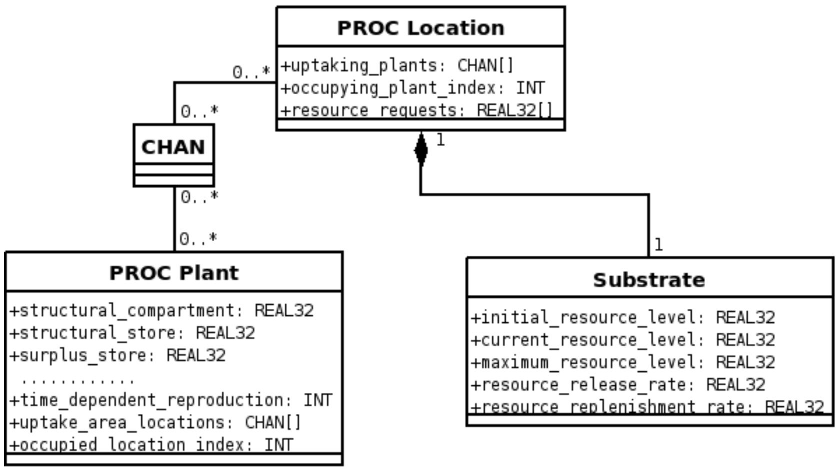
PROC Location
+uptaking_plants: CHAN[]
+occupying_plant_index: INT
+resource_requests: REAL32[]
0..*
0..*
CHAN
0..*
0..*
1
1
PROC Plant
+structural_compartment: REAL32
+structural_store: REAL32
+surplus_store: REAL32
....
+time_dependent_reproduction: INT
+uptake_area_locations: CHAN[]
+occupied_location_index: INT
Substrate
+initial_resource_level: REAL32
+current_resource_level: REAL32
+maximum_resource_level: REAL32
+resource_release_rate: REAL32
+resource_replenishment_rate: REAL32

Fig. 12. Processes and channels in the occam-π implementation (notation undefined)

unoccupied location is associated to an "empty" plant instance, rather than to no plant instance. The diagram does not show the relationship between a location and the plants that are taking up resource from it, as the C++ implementation calculates this from the plant traits and location at run-time.

In the occam-π implementation, a similar form is used for the location substrate, but occam-π supports more flexible data structuring for locations and plants. These are dynamic processes (the occam-π PROC structure), which interact through channels (the occam-π CHAN structure). The relation between plants and locations is implemented through explicit channel communication. The channel ends held by each plant process can be connected to the corresponding channel ends in any location process.

Comparing the two data structure implementations, we can observe differences in terms of attributes and their data types, the nature of plants, locations and their relationship. By reference to the biological model that these represent, we could declare ourselves adequately confident that these implementations represent implementations of the same abstract model. However, there are some subtleties that may present problems, such as the subtle quantitative effects of internal data formats: the C++ implementation uses the type double while the occam-π one uses REAL32. The two differ in terms of precision, double being represented on 64 bits, while REAL32 on 32 bits. Again, we need to check the effect of this difference through appropriate experimentation and testing – at this stage, we do not believe that the difference in precision qualitatively affects the simulations results, but we may need more evidence to convince the scientist.

6 Discussion

In this paper, we present a summary of an argument of adequate equivalence between an existing C++ simulation and a re-engineered version in occam-π. If we can assume that the original simulation is valid, then establishing the equivalence of the occam-π version would imply its validity in the same context and for the same purposes as the original simulation (see [26]). We used GSN to visualise the argument structure: those faced with evaluating our argument can immediately see the basis of the belief that the two models are adequately equivalent, and can challenge areas that they do not consider to be sufficiently supported by evidence.

This work is part of the CoSMoS project[4], which is developing a general framework for the simulation of complex systems. Part of this framework concerns the routine collection of assumptions – about the domain, the design, and the implementation. In the CoSMoS context, just the exposure of assumptions has led to scientific acceptance of some of our experimental simulations [2]. The work presented in this paper is a first step towards producing guidance and techniques for systematising argumentation relating to simulation development. However, turning assumptions into evidence for arguments that a simulation is a *valid imitation of the real world*, for a given scientific purpose, is a non-trivial activity, which is the subject of ongoing research.

Computer scientists who have spent a career in the deterministic world of the digital computer are often sceptical about the value of arguments of validity, safety etc. However, in simulating complex systems for scientific study, we are not seeking to model or implement traditional deterministic computer systems. A simulation that reduces the interacting complex systems of the real world to a deterministic system is unlikely to be adequate for the areas of scientific research that we seek to support.

Our work, here and in the CoSMoS project, also signals a departure from the common form of computer applications, in that our simulations are designed to support specific domain models – a particular expert's view of a particular scientific context. The aim of the simulation is to

[4] http://www.cosmos-research.org

support those areas of scientific experimentation that rely on that specific scientific context. If the scientist wishes to extend or adjust the context, then the simulation models must be extended or adjusted, and the adequate equivalence re-established. Later revisions can be facilitated through the careful recording of the argument of equivalence or validity for each simulation; if a preceding simulation was already acceptable, scientifically, and a new simulation corresponds to that simulation for part of its range, then we need concentrate only on what has changed.

This brings us to the use of occam-π in the re-engineered simulation presented here. To support the need to extend or adjust simulations in line with scientists' requirements, we need flexible implementations. In CoSMoS, we have used a range of implementation languages, and, although occam-π does not have the mature support of languages like C++ and Java, we have found that applications written in occam-π are easy to adapt and re-use. The CoSMoS project is assembling concrete evidence of this assertion, as well as seeking to improve the maturity of the occam-π programming environment.

7 Future Work

In relation to the specific example presented here, we need to complete the argument of adequate equivalence, and expose it all to the critical review of our scientist and his colleagues. Our next step is then to use the occam-π implementation to scale up the original experiments, which will improve the quality of the scientific evidence we can provide. We will then produce a series of modified simulations to support other experiments on intra-species and inter-species plant ecology, in collaboration with Bown's group.

In relation to the CoSMoS project, the work is contributing to the body of evidence on use and suitability of occam-π for developing flexible, validated simulations to support scientific work. The argumentation processes will form part of the CoSMoS framework for complex systems modelling and simulation – we continue to review SCS work for guidance in analysis, evidence collection and management, argument construction and validation. We plan to provide specific guidance on producing GSN type arguments in the context of complex systems simulation. In addition, we are applying the activities outlined here in a range of other case studies including various scientific studies of the immune system and work on swarm robotics.

8 Acknowledgements

The work of Ghetiu, Andrews and Polack is supported by the CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1.

References

[1] R. Alexander, R. Alexander-Bown, and T. Kelly. Engineering safety-critical complex systems. In *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation, York, UK, September 2008*, pages 33–62. Luniver Press, 2008.

[2] P. Andrews, F. Polack, A. Sampson, Timmis J, L. Scott, and M. Coles. Simulating biology: towards understanding what the simulation shows. In *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation, York, UK, September 2008*, pages 93–123. Luniver Press, 2008.

[3] U. Bausenwein, P. Millard, B. Thornton, and J. A. Raven. Seasonal nitrogen storage and remobilization in the forb rumex acetosa. *Functional Ecology*, 15(3):370–377, 2001.

[4] M. Begon, C. R. Townsend, and J. L. Harper. *Ecology: From individuals to ecosystems.* Blackwell Publishing, fourth edition, 2006.

[5] Eric Bonnici and Peter H. Welch. Mobile processes, mobile channels and dynamic systems. In *2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 232–239. IEEE Press, 2009.

[6] James L. Bown, Elizaveta Pachepsky, Alistair Eberst, Ursula Bausenwein, Peter Millard, Geoff R. Squire, and John W. Crawford. Consequences of intraspecific variation for the structure and function of ecological communities: Part 1. model development and predicted patterns of diversity. *Ecological Modelling*, 207(2-4):264–276, October 2007.

[7] J. Bryden and J. Noble. Computational modelling, explicit mathematical treatments, and scientific explanation. In *Artificial Life X*, pages 520–526. MIT Press, 2006.

[8] J. M. Epstein. Agent-based computational models and generative social science. *Complexity*, 4(5):41–60, 1999.

[9] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[10] T. P. Kelly. *Arguing safety – a systematic approach to managing safety cases.* PhD thesis, Department of Computer Science, University of York, 1999. YCST 99/05.

[11] S. Lavorel and E. Garnier. Predicting changes in community composition and ecosystem functioning from plant traits: revisiting the holy grail. *Functional Ecology*, 16(5):545–556, 2002.

[12] Nancy Leveson. High-pressure steam engines and computer software. *IEEE Computer*, 27(10):65–73, 1994.

[13] C. Marks and M. Lechowicz. A holistic tree seedling model for the investigation of functional trait diversity. *Ecological Modelling*, 193(3-4):141–181, March 2006.

[14] J. M. R. Martin and P. H. Welch. A Design Strategy for Deadlock-Free Concurrent Systems. *Transputer Communications*, 3(4):215–232, 1997.

[15] G. F. Miller. Artificial life as theoretical biology: How to do real science with computer simulation. Technical Report Cognitive Science Research Paper 378, School of Cognitive and Computing Sciences, University of Sussex, 1995.

[16] E. Di Paolo, J. Noble, and S. Bullock. Simulation models as opaque thought experiments. In *Artificial Life VII*, pages 497–506. MIT Press, 2000.

[17] F. Polack, S. Stepney, H. Turner, P. Welch, and F. Barnes. An architecture for modelling emergence in CA-like systems. In *ECAL*, volume 3630 of *LNAI*, pages 433–442. Springer, 2005.

[18] F. A. C. Polack, T. Hoverd, A. T. Sampson, S. Stepney, and J. Timmis. Complex systems models: Engineering simulations. In *ALife XI*, pages 482–489. MIT Press, 2008.

[19] Fiona A. C. Polack, Paul S. Andrews, and Adam T. Sampson. The engineering of concurrent simulations of complex systems. In *CEC 2009*, pages 217–224, 2009.

[20] F. W. Preston. The canonical distribution of commonness and rarity. *Ecology*, 43(3):185–215, 410–432, 1962.

[21] B. Reineking, M. Veste, C. Wissel, and A. Huth. Environmental variability and allocation trade-offs maintain species diversity in a process-based model of succulent plant communities. *Ecological Modelling*, 199(4):486–504, December 2006.

[22] Carl G. Ritson, Adam T. Sampson, and Frederick R. M. Barnes. Multicore Scheduling for Lightweight Communicating Processes. In John Field and Vasco T. Vasconcelos, editors, *Coordination Models and Languages, 11th International Conference, COORDINATION 2009, Lisboa, Portugal, June 9-12, 2009. Proceedings*, volume 5521 of *Lecture Notes in Computer Science*, pages 163–183. Springer, 2009.

[23] Carl G. Ritson and Peter H. Welch. A process-oriented architecture for complex system modelling. In Alistair A. McEwan, Steve Schneider, Wilson Ifill, and Peter Welch, editors, *Communicating Process Architectures 2007*, volume 65 of *Concurrent Systems Engineering Series*, pages 249–266, Amsterdam, The Netherlands, 2007. IOS Press.

[24] Adam T. Sampson, John M. Bjorndalen, and Paul S. Andrews. Birds on the wall: Distributing a process-oriented simulation. In *2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 225–231. IEEE Press, 2009.

[25] R. G. Sargent. The use of graphical models in model validation. In *18th Winter Simulation Conference*, pages 237–241. ACM, 1986.

[26] R. G. Sargent. Verification and validation of simulation models. In *37th Winter Simulation Conference*, pages 130–143. ACM, 2005.

[27] G. R. Squire. *The Physiology of Tropical Crop Production.* Oxon (UK). C.A.B. International, 1990.

[28] David Tilman. Causes, consequences and ethics of biodiversity. *Nature*, 405(6783):208–211, May 2000.

[29] Heather Turner, Susan Stepney, and Fiona Polack. Rule migration: Exploring a design framework for emergence. *International Journal of Unconventional Computing*, 3(1):49–66, 2007.

[30] M. Wheeler, S. Bullock, E. Di Paolo, J. Noble, M. Bedau, P. Husbands, S. Kirby, and A. Seth. The view from elsewhere: Perspectives on alife modelling. *Artificial Life*, 8(1):87–100, 2002.

[31] S. P. Wilson, J. A. McDermid, C. H. Pygott, and D. J. Tombs. Assessing complex computer based systems using the goal structuring notation. In *2nd Int. Conf. Engineering of Complex Computer Systems*, pages 498–505, 1996.

[32] David C. Wood and Peter H. Welch. The Kent retargetable occam compiler. In *WoTUG '96: Proceedings of the 19th world occam and transputer user group technical meeting on Parallel processing developments*, pages 143–166. IOS Press, 1996.