

Novel visualisation and analysis of natural and complex systems using systemic computation

Erwan Le Martelot^{a,*} and Peter J. Bentley^b

^aElectrical and Engineering Department, University College London, Torrington Place, London WC1E 7JE, UK.

E-mail: e.le_martelot@ucl.ac.uk

^bComputer Science Department, University College London, Gower Street, London WC1E 6BT, UK.

E-mail: p.bentley@cs.ucl.ac.uk

*Corresponding author.

Abstract The study, analysis and understanding of natural processes are difficult tasks considering the complex nature of such processes. In this respect, the visual analysis of such systems can be of great help in the understanding of their behaviour. The increasing power of modern computers enables novel possible uses of computer graphics for such tasks. Previous work introduced systemic computation, a new model of computation and corresponding computer architecture aiming at enabling a clear formalism of natural and complex systems and providing tools for their analysis. Here, we present an online visualisation of dynamic systems based on this novel paradigm. The observation is done at a high level of abstraction, focussing on information flow, interactions and emergent behaviour, and enabling the identification of similarities and differences between models of complex systems. This visualisation framework is then applied to two biological networks: a bistable gene network and a MAPK signalling cascade.

Information Visualization (2011) **10**, 1–31. doi:10.1057/ivs.2010.8; published online 16 September 2010

Keywords: systemic computation; natural computation; biological systems; complex systems; dynamic visualisation

Introduction

Following and understanding the phases and states that natural and complex systems or processes go through over time can be a difficult task. For instance, the analysis of time course in gene regulation requires tools to identify patterns of interest in DNA sequences.¹ In addition, emerging patterns or behaviours² can be observed in biological or bio-inspired networks such as gene and protein regulatory networks,³ immune systems,⁴ molluscs shells⁵ or cellular automata⁶ and neural networks,² but the steps and building blocks⁷ leading to this emergence remain hidden in the final result. One way to help us understand the information flow, interactions and emergent behaviour in such models is by using computer graphics and visualising it using a non-ambiguous mapping between a computational system and a graphical output.

Previous work introduced *systemic computation* (SC),⁸ a new model of computation and corresponding computer architecture based on a systemics world-view and supplemented by the incorporation of natural characteristics. This article bio-inspired paradigm aims at enabling a clear formalism of natural systems and such formalism can provide the grounding for a non-ambiguous graphical representation. This article introduces a novel use of computer graphics for the understanding of natural and complex systems as an online visualisation of dynamic systems based upon SC. The underlying formalism of SC enables a visualisation focussing on interactions and structure, thus guaranteeing a comparison at a high

Received: 28 January 2010

Revised: 24 June 2010

Accepted: 28 June 2010

level of abstraction between models resembling their original natural system rather than a computer program. A single model might progress towards various possible distinct states, all with a similar memory usage, yet their meaning can be literally different, which only a high level of abstraction analysis could reveal (for example, two similar neural networks trained to recognise two different objects process the information differently even though using memory in the same way as their structure is the same). Similarly, two models can seem to be very different and so could be their memory usage, yet their behaviour could present some strong similarities that again can only be observed at a high abstraction level analysis (for example, a neural network and an immune system could perform a similar data processing even though their structure and thus memory usage is different). Biological systems may resemble each other but behave differently whereas some other systems might look different but behave in a similar way. Do certain chemical interactions resemble each other in the way they interact? Do certain neural structures resemble each other in the way their structures change over time?

This work describes and illustrates how a systemic computation model can be visualised and analysed in terms of information flow, interaction, organisation and emergent behaviour in order to achieve such analysis and help answer such questions in the future. The following section locates the place of SC visualisation by providing a background to the area of visualisation in general and within bioinformatics. The section after provides an overview of systemic computation. We then present the visualisation framework, followed by concrete case studies of biological networks showing how gene and protein regulation can be simulated, observed and visually analysed over time. It is then followed by other interesting visualisation examples. The final section discusses and concludes on the potential of this novel visualisation for the study of natural and complex systems.

Background

Many visualisation techniques have been developed over the past 20 years to help understand systems or organise large amounts of data.

The field of bioinformatics for instance is commonly facing complex biological problems with large quantities of data such as genome analysis. To help users browsing through the data, visualisation software involving hierarchical clustering have been developed to aid in the analysis of genomic data.⁹ Frameworks such as GVis¹⁰ permits to interactively analyse genomic data of organisms in order to study phylogeny hierarchies. Pathway tools¹¹ such as, *KEGG* pathways,¹² *ExpASy*¹³ can be found and aim at representing knowledge that can then be analysed and potentially visualised. In addition, popular bioinformatics tools such as *Cytoscape*¹⁴ provide a platform for

the visualisation of molecular interaction networks and biological pathways. The software supports various data formats enabling the visualisation of networks available from external sources. Visualised data can be positioned, labelled, decorated and commented. The software also allows users to enrich its features by the addition of plugins. While such tools enable a representation and an analysis of the complex structure linking the various entities, along with the knowledge associated to them, it does not provide a simulation of how such models actually behave. It represents knowledge and provides tools and methods to exploit it. The aim of the systemic computing visualisation is to provide an online visualisation of models that represents the simulation of these models over time. It thus enables an analysis of the interactions between entities and their evolution through time, hence of the models' actual dynamic behaviour. Such simulations can indeed inform users about the model and results may then be included within models such as *Cytoscape*'s. Another aim of the SC visualisation is to remain mainly software controlled rather than user controlled in order to enable a generic representation of models and interactions that can lead to comparisons between models of various natures (hence allowing the comparison of models that can be thought as being significantly different but that can be revealed more similar than expected).

Beyond bioinformatics, valuable visualisation techniques have been explored to deal with complex systems or large data sets. State transitions in graphs were visualised to help users understand computer systems represented as transition systems. Tools such as *FSMView*,¹⁵ *StateVis*¹⁶ or *Bar tree*¹⁷ have been developed to study the structure and transitions within large transition graph systems. Attractor states in complex systems were visualised in Gröller¹⁸ using *Advanced Visual Systems*, a general-purpose visualisation system based on a data-flow model. Löffelmann *et al*¹⁹ later investigated two different methods of visualisation of dynamical systems near critical points such as linear dynamical systems or Lorenz system. These two papers investigated critical states of mathematical systems, providing insights about their flow dynamics. The modelling of a two-user system with shared resource and visualisation of the ongoing process for each user in order to understand the way the system works was performed in Viste and Skartveit.²⁰ This resource-sharing problem, seemingly simple, is a non-linear complex system that may be difficult to control. This work discusses how visualising this system can help users to understand its working principle. Software structures were visualised in Hendley and Drew²¹ within a 3D world using repulsive or attractive forces between objects, providing an example of program structure representation in a 3D space using force-based data layout. The world wide web (WWW) was visualised in Wood *et al*²² also using force-based layout in order to provide the user with information about the structure, the organisation and the content of the space being explored. Benford *et al*²³ also presented visualisations of the WWW

structure, browsing history, searches, presence and activities of multiple users in 3D. These two papers highlight the potential of 3D spaces and force-based algorithms for data structure layout and representation with a large amount of data. The use of motion within visualisation was investigated in Bartram.²⁴ This work focussed on the exploration of new display dimensions to support the user in information visualisation and explains that motion holds promise as a perceptually rich and efficient display dimension. Tree structures were visualised in Robertson *et al*,²⁵ highlighting the potential of 3D spaces and interactive animation for the human perceptual system.

These methods all address particular problems and highlight the potential of approaches such as force-based layout engines in space or coloured animation. They provide insights regarding the visualisation approaches to consider for the development of a more generic approach to complex systems visualisation aiming at unifying all the complex systems within a single formalism. In this respect Bosch *et al*²⁶ introduced *Rivet*, a visualisation system for the study of modern computer systems. This framework provides various visualisation tools to visualise data, computer programs memory usage, observe code execution on multiprocessors computer and has a flexible architecture allowing users to define their visualisations. However, this approach relies on a conventional view of computation, as opposed to natural computation. Memory and computer resource observation might tell us everything that happens at the hardware level, yet the information explaining the states complex systems like bio-inspired systems go through is of a higher level of abstraction, involving interactions between parts of a system leading to some emerging behaviour.

Higher-level implementations of bio-inspired systems along with a graphical representation was presented in Phillips *et al*²⁷ using the stochastic π -calculus. As discussed in Le Martelot and Bentley,²⁸ the mathematical nature of π -calculus makes the model implementation non-intuitive, unnecessarily complicated, and therefore difficult to approach for a non-specialist. In this respect, a graphical notation for the stochastic π -calculus was introduced to allow a clearer presentation of the models. Yet this visualisation technique suffers the same issues in expressing the interactions and transformations of entities, resulting in an unclear model.

To provide a clear formalism for the modelling of natural processes, systemic computation (SC) was introduced in Bentley⁸ and unifies notions of natural computation and electronic computation. Previous work introduced a computer platform for SC²⁹ and explored various bio-inspired models implemented using SC and their respective properties.^{28,30–32} Systemic computing visualisation exploits some of the aforementioned ideas and aims at providing a unified dynamic representation of interactions and structural changes for natural and complex processes. This article follows the introduction

of SC visualisation in Le Martelot and Bentley³³ and presents in detail three visualisation methods for SC using two models of biological systems. The following section provides an overview of SC.

Systemic computation

This section provides the major definitions and tools that will enable a global understanding of systemic computation, necessary for the understanding of the work presented in this article. First, systemic computation is defined. Then calculus and graphical notations are introduced to help describing the models presented here. Finally, the systemic analysis that explains how to *think in SC* is presented.

Definition

Looking at a biological brain, an ant colony, an immune system, the growth of a plant or a crystal, nature is clearly performing some kind of computation. We can state that natural computation is stochastic, asynchronous, parallel, homeostatic, continuous, robust, fault tolerant, autonomous, open-ended, distributed, approximate, embodied, has circular causality, and is complex.⁸ The traditional von Neumann architecture however is deterministic, synchronous, serial, heterostatic, batch, brittle, fault intolerant, human-reliant, limited, centralised, precise, isolated, uses linear causality and is simple. The incompatibilities seem clear.

To address these issues, Bentley⁸ introduced Systemic Computation, a new model of computation and corresponding computer architecture based on a systemics world-view and supplemented by the incorporation of natural characteristics (listed above), as opposed to conventional computation paradigms (for example, procedural, object-oriented). SC stresses the importance of structure and interaction, supplementing traditional reductionist analysis with the recognition that circular causality, embodiment in environments and emergence of hierarchical organisations all play vital roles in natural systems. Systemic computation makes the following assertions:

- Everything is a system (building block of the systemic world).
- Systems can be transformed but never destroyed or created from nothing.
- Systems may comprise or share other nested systems.
- Systems interact, and interaction between systems may cause transformation of those systems, where the nature of that transformation is determined by a contextual system.
- All systems can potentially act as context and affect the interactions of other systems, and all systems can potentially interact in some context.

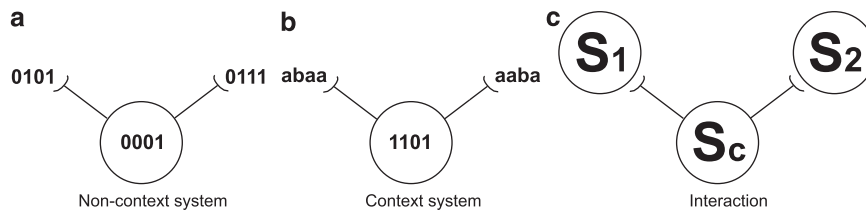


Figure 1: (reproduced from Le Martelot *et al.*²⁹) (a) A system used primarily for data storage (showing its kernel and schemata values). The kernel (in the circle) and the two schemata (at the end of the two arms) hold data. (b) A system acting as a context (showing its kernel and schemata values). Its kernel defines the result of the interaction while its schemata define allowable interacting systems. (c) Representation of an interacting context. The contextual system S_c matches two appropriate systems S_1 and S_2 with its schemata and specifies the transformation resulting from their interaction as defined in its kernel.

- The transformation of systems is constrained by the scope of systems, and systems may have partial membership within the scope of a system.
- Computation is transformation.

Computation has always meant transformation in the past, whether it is the transformation of position of beads on an abacus, or of electrons in a CPU. But this simple definition also allows us to call the sorting of pebbles on a beach, or the transcription of protein, or the growth of dendrites in the brain, valid forms of computation. Such a definition is important, for it provides a common language for biology and computer science, enabling both to be understood in terms of computation.

Therefore, in systemic computation, everything is a system (that is, systems are the building blocks of any SC model), and computations arise from interactions between systems. Two systems can interact in the context of a third system. All systems can potentially act as contexts to determine the effect of interacting systems. Systems are defined and identifiable by a *shape*. Context systems make use of these shapes to identify the systems allowable for interaction. The shape of a context system also describes the nature of the interaction it defines.⁸

In a digital environment, one convenient way to represent and define a system (that is, its shape) is as a binary string. Each system (that is, here string) is divided into three parts: two schemata and one kernel. These three parts can be used to hold anything (data, typing, and so on) in binary as shown in Figure 1(a). The primary purpose of the kernel is to define an interaction result and also optionally to hold data. (Data is held as information coded in binary similarly to variables in conventional programming languages like C or Java.) The two schemata define which subject systems may interact in this context as shown in Figure 1(b) and (c). The schemata thus act as shape templates, looking for systems matching their shape. The resultant transformation of two interacting systems is dependent on the context in which that interaction takes place. A same pair of systems interacting in a different context would produce a different

transformation. How templates and matching is done precisely is explained in Le Martelot *et al.*²⁹

Thus, each system comprises three elements: two schemata that define the allowable interacting systems in the context of the current system, and a kernel that defines the nature of the transformation two interacting systems can undergo. This behaviour enables more realistic modelling of natural processes, where all behaviour emerges through the interaction and transformation of components in a given context. It incorporates the idea of circular causality (for example, A may affect B and simultaneously B may affect A) instead of the linear causality inherent in traditional computation.³⁴ Such ideas are vital for accurate computational models of biology and yet currently are largely ignored.

Finally, systemic computation also exploits the concept of scope. In all interacting systems in the natural world, interactions have a limited range or scope, beyond which two systems can no longer interact (for example, binding forces of atoms, chemical gradients of proteins, physical distance between physically interacting individuals). In cellular automata this is defined by a fixed number of neighbours for each cell. Here, the idea is made more flexible and realistic by enabling the scope of interactions to be defined and altered by another system. Thus a system can also contain or be contained by other systems. Interactions can only occur between systems within the same scope. Therefore any interaction between two systems in the context of a third implies that all three are contained within at least one common super-system, where that super-system may be one of the two interacting systems. For full details see Bentley⁸ and Le Martelot *et al.*²⁹

With its origins grounded into natural computation, SC provides a method to model nature-inspired systems in a way that resembles their true nature. SC has been used to model genetic algorithms, neural networks, artificial immune systems and has demonstrated properties of flexibility, fault tolerance, self-repair and self-organisation.^{29–32} The next section explains how SC can be visualised dynamically to represent and follow the flow of information in SC models. This is then followed by a study of two biological networks.

Table 1: Systemic calculus notation

Expression	Signification
system	A system called <i>system</i> .
system $[x_1, \dots, x_n]$	<i>system</i> contains variables x_1 to x_n .
system (sub ₁ ... sub _n)	<i>system</i> contains sub systems <i>sub</i> ₁ to <i>sub</i> _n .
(system ₁ ... system _n)	Systems <i>system</i> ₁ to <i>system</i> _n share a same unnamed scope.
(system ₁ ()system ₂)	Systems <i>system</i> ₁ and <i>system</i> ₂ are in the scope of one another (ie <i>system</i> ₁ contains <i>system</i> ₂ and <i>system</i> ₂ contains <i>system</i> ₁).
system ₁ }- context -{ system ₂	Systems <i>system</i> ₁ and <i>system</i> ₂ interact in the context system <i>context</i> .
system ₁ (system ₂) }- context	Systems <i>system</i> ₁ and <i>system</i> ₂ , with <i>system</i> ₂ being also contained within <i>system</i> ₁ , interact in the context system <i>context</i> .
(system ₁ ()system ₂) }- context	Overlapping systems <i>system</i> ₁ and <i>system</i> ₂ interact in the context system <i>context</i> located out of both <i>system</i> ₁ and <i>system</i> ₂ .
interaction → result	<i>interaction</i> is a triplet comprising two interacting systems and a context of interaction. <i>result</i> is the interaction result showing the organisation of the systems. In <i>result</i> the context of interaction can be discarded assuming it remained unchanged. Both <i>interaction</i> and <i>result</i> expressions can include variable notations, as well as generic notations. <i>result</i> can be several stochastic outcomes separated by . eg s_1 }-inject- {s ₂ → s ₁ (s ₂) s ₂ (s ₁)

Calculus notation

A calculus notation for SC was developed^{28,35} and the relevant definitions to this work are given in Table 1. Systems are represented by labels (that is, names), as well as the attribute variables (where data can be held). The schemata are represented using the textual notation }- or -{. The scope notation uses parenthesis and the transformation from one expression to another is denoted using the symbol →. Note that for any interaction to be valid, the two interacting systems and the context system must share a common scope, not necessarily shown in a calculus expression when unnamed.

Graphical notation

To help understand and represent better SC models, this article introduces a graphical notation for SC (inspired from the notation initially given in Bentley⁸). The graphical notation represents systems, hierarchies, interactions and their resulting scope changes and transformations. Figure 2 provides the graphical notations along with the corresponding calculus notations. A context system is displayed with its schemata. A non-context system is displayed without schemata. (Note that any system can potentially act as a context, and therefore a system not acting as a context in an interaction may act as a context in another one.) An interaction between two systems within a context where the system are transformed can be indicated using dashed arrows going from the original system to the resulting system and passing by the context to indicate that the transformation results from an interaction in that context. The insertion and ejection of a system into and from a system is shown in a similar way. Figure 3 provides graphical representations for the most common scope situations that can be encountered, along with the corresponding calculus notations.

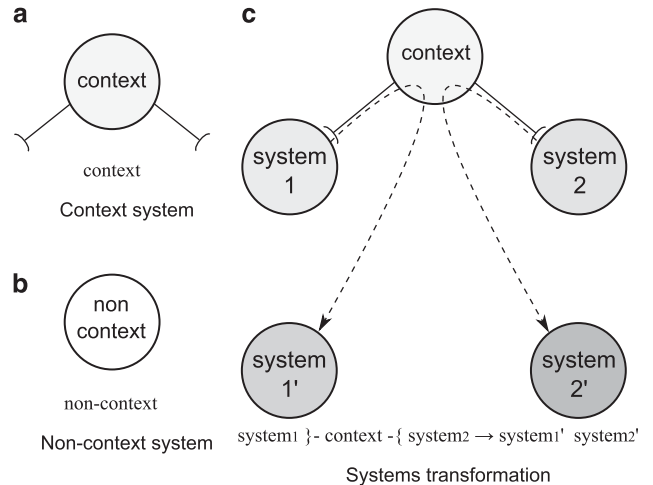


Figure 2: Graph notations for interactions and corresponding calculus notations: (a) shows a context system where schemata are drawn; (b) shows a non-context system, without schemata; (c) shows an interaction and its result where *System*₁ and *System*₂ got transformed into *System*₁' and *System*₂'.

The graph notation does not need to show the variables or attributes systems may hold (that is, data). This is done using the calculus notation presented above. The concept of the graphical notation is to represent with colour and/or labels the various kinds of systems. Any major data that might affect the role of a given system should rather be represented using two similar colours or labels (for example, assuming Figure 2(c) changes an attribute in *System*₁, it then becomes *System*₁' instead of having textual data aside). Note also that the arrows notation is given to enable the drawing of both an interaction and its result within one drawing, but successive graph

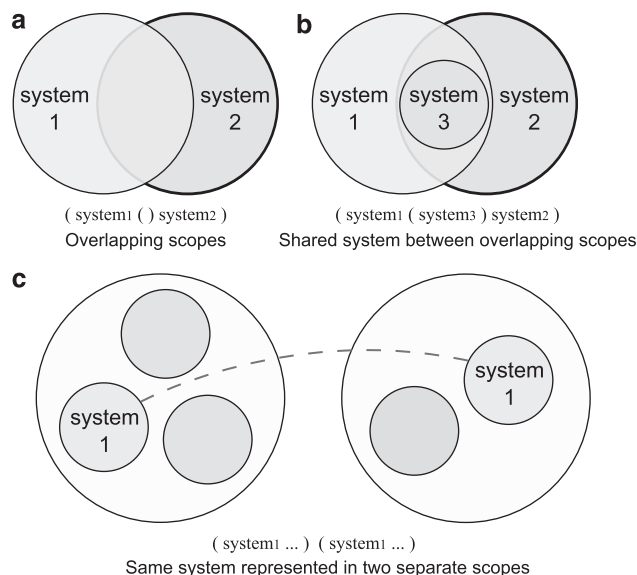


Figure 3: Graph notations for locality and corresponding calculus notations: (a) represents two overlapping scopes $System_1$ and $System_2$; (b) represents two overlapping scopes $System_1$ and $System_2$ sharing the system $System_3$; (c) represents $System_1$ twice at various locations (for clarity or to represent systems placed in more than two non-overlapping scopes), the two graphical instances being linked by a dashed curve (with long dashes, in red in the coloured version).

representations can also be given to illustrate the states of transformation.

Note that the sharing of a system between scope does not necessarily imply that the scopes are overlapping. However if considering neighbourhoods (for example, in the real world as well as in a cellular automata) sharing systems, it is sensible to also represent the scopes as overlapping (even if the model implementation is not computationally impacted by such additional information and does not need it coded).

Systemic analysis

SC is an alternative model of computation that was designed to improve fidelity and clarity in the modelling of natural processes. It was therefore necessary to develop a method of analysis that helps users to develop nature-inspired models that resemble their original natural process enough in order to display the features of interest. To address this, the *systemic analysis*,^{29,35} was developed.

Before any new program can be written, it is necessary to perform this systemic analysis in order to identify and interpret appropriate systems and their organisation. The systemic analysis provides a method for analysing and expressing a given problem or natural system more formally in SC. When performed carefully, such analysis

can itself be revealing about the nature of a problem being tackled and the corresponding solution. A systemic analysis is thus the method by which any natural or artificial process is expressed in the language of systemic computation. The steps are in order:

1. *Identify the systems*: determine the level of abstraction to be used, by identifying which entities will be explicitly modelled as individual systems.
2. *Analysis of interactions*: determine which system interacts with which other system in which context system.
3. *Analysis of structure*: determine the order and structure (scopes) of the emergent program (which systems are inside which other systems) and the values stored within systems.

These steps will be illustrated below with each model developed in this article.

Visualising SC models

Systemic computation provides a distributed and parallel approach where components interact and can be intricately entwined with the other components. SC models contrast significantly with traditional models as the outcome of an SC model results from an emergent process rather than a deterministic predefined algorithm. For full details about programming and modelling in SC (see Le Martelot *et al.*²⁹ and Le Martelot³⁵). Thus, visualising a line of code is here irrelevant as computation is transformation, and transformation is the result of interaction. The SC visualisation must thus reveal the interplay between components and their environment, rather than a low-level memory usage observation.

This section introduces the visualisations developed for SC models and illustrates them using a sample model before studying more complex natural systems. This toy model offers a simplified fire chemistry applied to fire spreading. This simple network of chemical reactions enables easy-to-understand illustration cases the reader can come back to when seeking a visual explanation for what a visualisation tool offers.

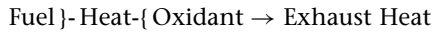
The visualiser was built on top of the SC platform presented in Le Martelot *et al.*²⁹ A model is therefore visualised as it runs (online visualisation); the user can record all interactions and go back in time or replay a previously recorded execution. A frame rate can be applied to control the speed of the execution if going too fast for the user's needs.

Sample model: A simple fire-spreading model

To help introduce and illustrate the visualisation tools presented here, a toy model of a familiar chemical reaction is used: fire. This model was chosen for it allows to

illustrate simply and clearly the notions of scope, interaction and transformation within the visualisation framework.

When modelling with SC it is necessary to perform a systemic analysis, as defined above, in order to identify and interpret appropriate systems and their organisation. The first stage is to identify the low-level systems (that is, determine the level of abstraction to be used). Fire is a combustion, hence some fuel is required. Combustion is a type of oxidation reaction, it therefore requires oxygen. Oxidation is an exothermic reaction, which means it releases heat energy. In addition, for fire to exist the temperature must be high enough to cause combustion. A simple model can therefore take fuel (for example, wood, petrol) and oxidant (for example, oxygen) as reactants, combusting in a heat context, and producing heat and exhaust as a result of the combustion. This interaction is shown below using the SC calculus notation:



Some fuel and oxidant interact in a heat context transforming the fuel and oxidant into exhaust and heat. The heat context is not cited in the right term of the reaction as in SC the context remains unchanged during an interaction.

The distribution of the fuel and oxidants in various geographical places can be done using neighbourhood scopes, sharing some fuel and oxidant, therefore making fire propagation from one area to another possible. Finally, the fire has to be initiated in an area by an initial heat system (for example, match, lightning). The model should thus show the fire spreading across the various areas in contact with one another starting from the fire ignition. The whole fire model is part of a global environment (for example, atmosphere, ecosystem, universe) commonly represented in SC by a universe system. The structure and principle of the model are summarised in Figure 4 using the SC graph notation. The dashed arrows indicating systems transformation are merging to emphasise that the interaction of fuel and oxidant results in heat and exhaust as a product of both reactants.

The complete states sequence is given below using the calculus notation:

- Initial** Neighbourhood₁ (Heat Fuel Oxidant Neighbourhood₂)
 Neighbourhood₂ (Neighbourhood₁ Fuel Oxidant Fuel Oxidant Neighbourhood₃)
 Neighbourhood₃ (Neighbourhood₂ Fuel Oxidant Fuel Oxidant)
- Step 1** Neighbourhood₁ (Heat **Heat Exhaust** Neighbourhood₂)
 Neighbourhood₂ (Neighbourhood₁ **Heat Exhaust** Fuel Oxidant Neighbourhood₃)
 Neighbourhood₃ (Neighbourhood₂ Fuel Oxidant Fuel Oxidant)
- Step 2** Neighbourhood₁ (Heat Heat Exhaust Neighbourhood₂)
 Neighbourhood₂ (Neighbourhood₁ Heat Exhaust **Heat Exhaust** Neighbourhood₃)
 Neighbourhood₃ (Neighbourhood₂ **Heat Exhaust** Fuel Oxidant)
- Step 3** Neighbourhood₁ (Heat Heat Exhaust Neighbourhood₂)
 Neighbourhood₂ (Neighbourhood₁ Heat Exhaust Heat Exhaust Neighbourhood₃)
 Neighbourhood₃ (Neighbourhood₂ Heat Exhaust **Heat Exhaust**)

Graphic representation of models

An SC model is a set of systems in which some can act as context of interaction between other systems and some can act as scope for other systems. Scope expresses the notion of hierarchy: a system S_1 within another system S_2 is in the scope of system S_2 . The graphical representation must therefore show systems, the hierarchy linking them (that is which systems are contained within other systems), the interactions between them and the changes that occur.

The visualisation framework provides three representations of a model: a *2D graph*, a *3D explorer* and a *3D informational structure*. A global colour scheme is controlled by the user to give each system type (template of schemata and kernel) a different colour. Figure 4 shows the colour per system used in the fire-spreading model. Also at any time, the internal binary state of a system (schemata and kernel as held in memory) can be visualised in order for instance to check a counter (for example, age, amount) or a flag (for example, initialised/non-initialised) variable. However this system data access is not part of the visualisation process but part of the systemic computer as any significant change in internal representation (system's data) should be reflected in the visualisation as a change in system's type (therefore colour).

2D graph

This first view of the visualiser is a standard 2D graph representing the hierarchy of systems and their types only. The graph is updated over time to display systems in their current state or hierarchical changes, both of which may change due to interactions. Figure 5 shows the fire-spreading model undergoing interactions and transformations. The fire progression through neighbourhoods can be observed between the two states.

3D explorer

The second view is a 3D graph working like a systems explorer: the user can explore the graph in depth by zooming onto a particular area of a map to see more details. Each view is planar and the use of 3D enables going deeper in the structure to see more details (similar to a

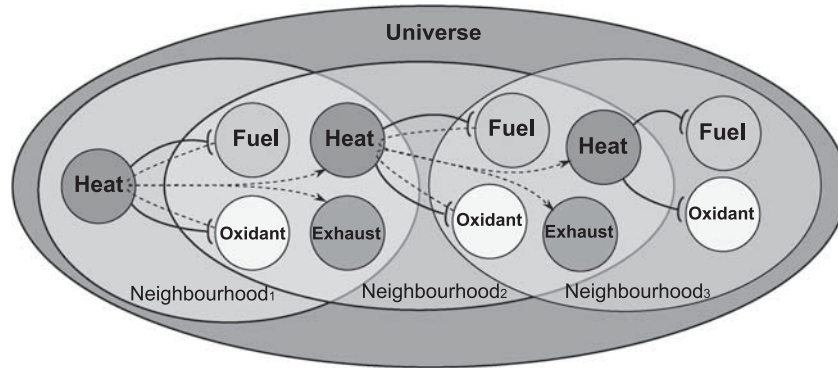


Figure 4: Structure and behaviour of a simple fire-spreading model. The whole model is part of a universe encompassing neighbourhoods (here three was chosen for clarity), sharing fuel and oxidants. The left neighbourhood has an initial heat. The schemata show the interactions happening and the dashed arrows represent the transformation of systems through interactions.

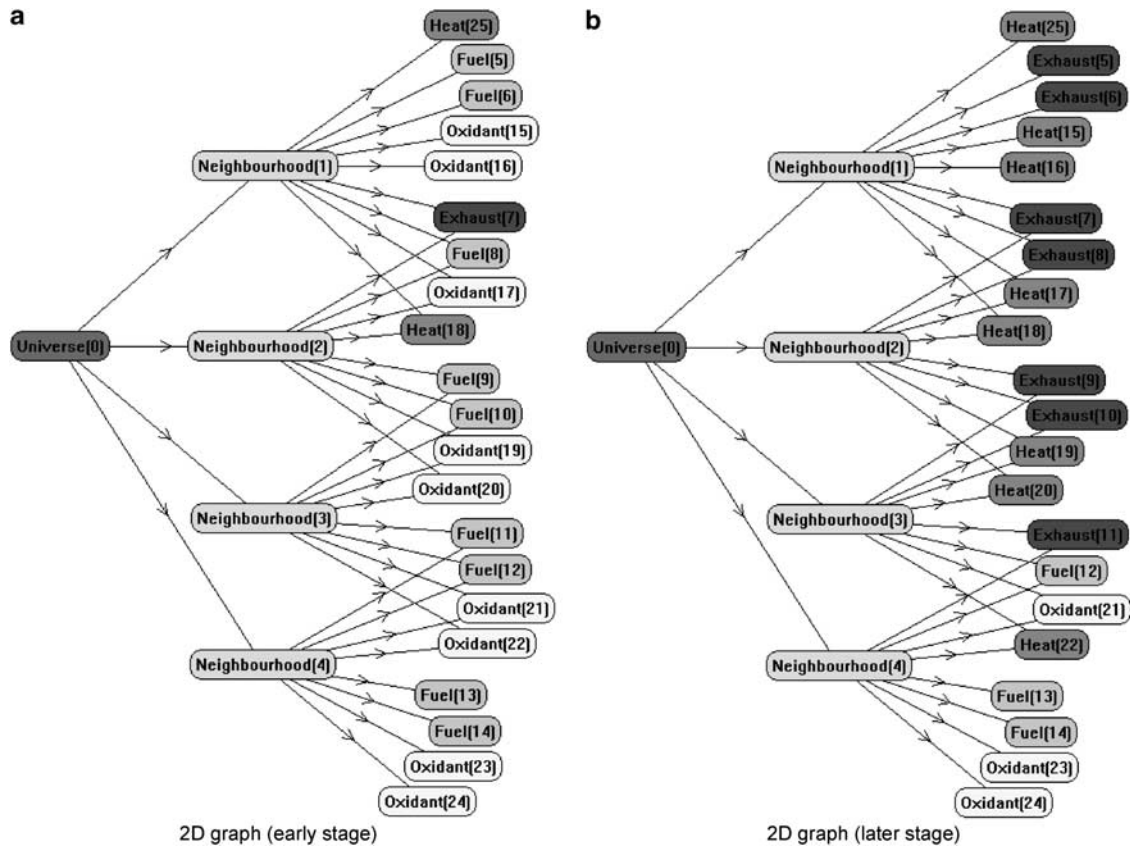


Figure 5: 2D graph view of the fire-spreading model (a) at an early stage and (b) at a later stage. Each coloured box is a system. The arrows between them indicate the scopes, going from the parent node to the contained node.

fractal explorer). Each system is represented as a sphere. Each sphere can contain the subsystems' spheres and so on as shown in Figure 6. Each deeper level can be explored by zooming into a subsystem. This visualisation method thus provides a local view per scope rather than a global view over the whole model (even though the whole model

can be seen from the universe view). Note that the top half of the spheres containing subsystems is hidden to allow the camera to see inside.

This view no longer uses arrows to represent hierarchies as in the 2D graph but represents the subsystems physically within their parent systems (that is, within

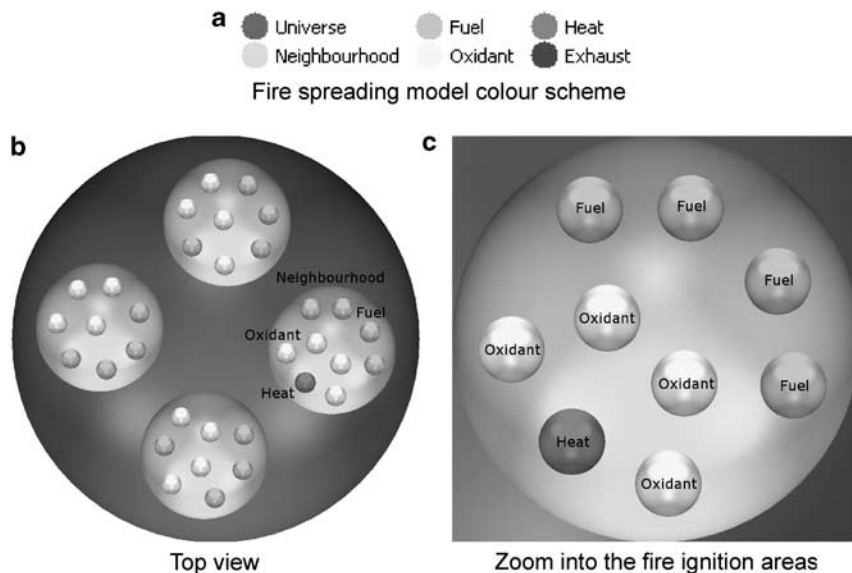


Figure 6: Explorer view of the fire-spreading model showing (b) the top view (universe) and (c) zooming from the universe into the neighbourhood containing the fire ignition (in dark grey, or red in the coloured version).

the sphere representing their parent system). It allows the visualisation to focus on the interactions happening within each scope. This concept of physical space however leads to a limitation: systems can have several parent systems. In the fire-spreading model, in order to spread the fire across consecutive neighbourhoods some fuel and oxidant was shared between neighbourhoods, making these systems belong to several parent systems. Figure 4 used overlapping space to represent this sharing of systems; however, when systems belong to many parent systems, themselves having constraints of location due to other systems, and so on, it becomes at some point impossible in 2D or 3D view to solve the physical location constraints imposed by the hierarchy. In an abstract space, adding dimensions could solve the problem but in visualisation the amount of dimensions is limited. Another physical space limitation that no dimensional space can solve is the recursivity in hierarchy (for example, a system S_1 inside a system S_2 itself inside S_1). (Recursivity cases will be discussed further below.)

The chosen solution was to consider each scope as a dimension and each view is dedicated to a scope dimension. Therefore each scope has its own physical systems representing the corresponding abstract systems within this scope only. Each system is thus represented with as many graphical instances (spheres) as it has scopes. The full representation in the 3D explorer of a computational system is thus the combination of all its scope-wise instances.

The graphical layout of the systems is handled by a form of force-based layout algorithm.³⁶ The principle here is, within each scope, to push each system away from each other up to a certain distance (margin) while maintaining the systems physically contained within their

scopes (hence constrained in spatial locations). The size of the systems is determined depending on the amount of systems in each scope. Both margin and systems size can be adjusted by the user by applying a global scaling factor to all systems. Interactions between systems are represented by temporary forces that attract the interacting systems towards their context of interaction. Note that the SC behaviour does not rely on spacial location but only on scopes. Systems are placed in this way to improve clarity of visualisation.

The most global view is the view from the universe, directly or indirectly containing the whole model. In the fire-spreading model, within the universe are the four neighbourhoods, themselves containing the combustible or combusted material as shown in Figure 7. In this model, some systems are represented several times, once in each scope they belong to. Some neighbourhoods indeed share fuel and oxidant, which thus have several graphical instances. In Figure 7(b) the right neighbourhood has the initial heat, or match (red system). Then in Figure 7(c) one pair of fuel–oxidant (green and yellow) has been burnt (red and brown) but a burnt pair (red and brown) also appears in the top neighbourhood. These two pairs are the same but they are represented in their two respective scopes. Figures 7(d) and (e) show progression of the fire until everything has burnt (that is, no more fuel and oxidant).

Informational 3D structure

The previous visualisation enabled an online representation per scope that enables an exploration of the hierarchy. However, the per scope view ruled out the uniqueness of systems representation (systems being

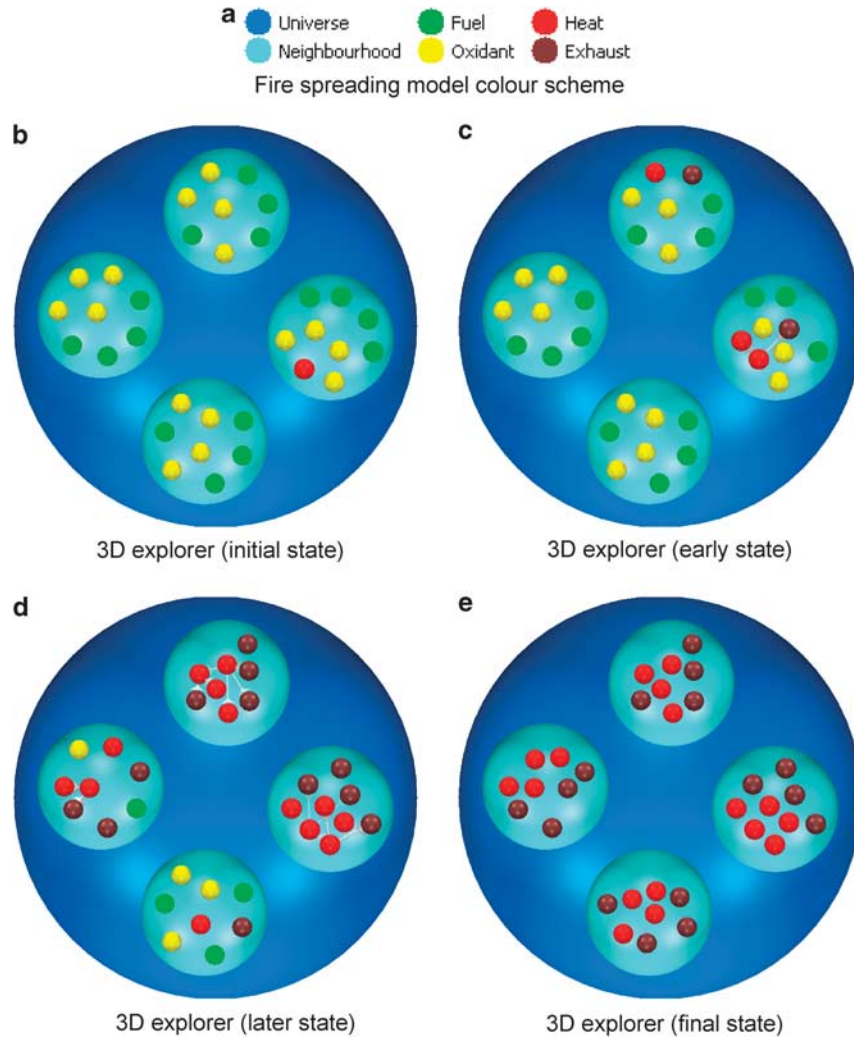


Figure 7: 3D explorer view of the fire-spreading model: (a) provides the colour scheme; (b) shows the model in its initial state, (c) at an early state, (d) at a later state and (e) in its final state (that is, no more to burn).

represented once per scope). The aim of this new visualisation is therefore to provide a representation where the systemic structure of models is maintained (that is, one system for all its scopes). This final visualisation is a global representation of the model in its current state through an abstract structure floating in a 3D space.

The layout is also handled by a force-based layout algorithm³⁶ that constantly pushes systems away from each other using Coulomb's law.³⁷ Subsystems are connected by spring-like links to their super systems, keeping them within distance of the super systems using Hooke's law.³⁸ As a result, systems sharing a same scope tend to remain close to each other, as can be seen on Figures 8(b). Again, the spatial position is used for visualisation alone—it has no affect on the SC behaviour and absolute spatial location does not represent any information (relative location does represent information, however).

An anchor, usually a root system (for example, universe), is grounded at the centre of the space making the whole model centred and keeps, by its structure, systems within distance of the space centre (without that, systems would drift away indefinitely).

Interactions between systems create a temporary spring-like connection going from each interacting system to the context, as can be seen on Figures 8(c)–(e). As long as these links remain they attract (also using Hooke's law³⁸) linked systems towards each other. The lifetime of the links can be changed. Links get more and more transparent as they age until they disappear. Once they disappear, there is no longer an attraction force between the systems until a new link is created by a new interaction.

The changes of systems over time can be recorded and presented similarly to a tree ring view. The colour at the centre is the current state, and the rings around are the successive past states of the system. Figures 8(d)

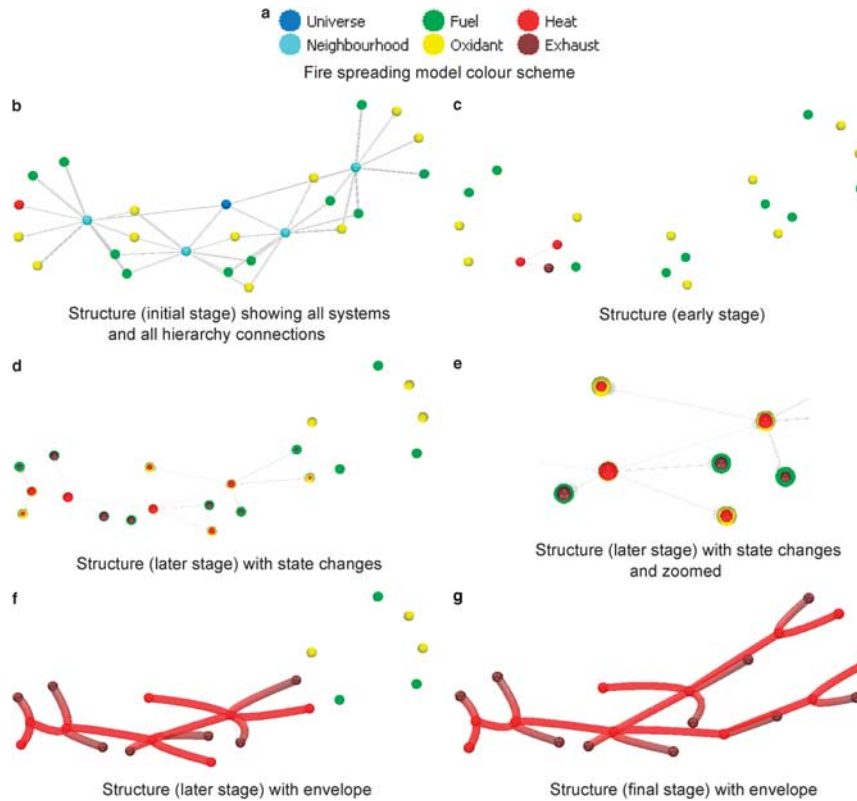


Figure 8: Structure view of the fire-spreading model at various stages: (b) shows all systems at initial stage. All other figures hide the universe, the neighbourhoods and their hierarchy links. Early stage is shown in (c); (d) displays a later stage with the past states of the systems shown using a tree ring view; (e) provides a zoom of the central part of the previous figure; (f) and (g) show the envelope during computation and at the final stage (no more computation can happen as everything combusted).

and (e) display the changes of types in systems, showing the initial states of fuel and oxidant as well as their new state of heat and exhaust. Within each system the larger a coloured ring the longer the system remained in the corresponding state. The successive rings also indicate the order in which changes occurred. If the initial state ring of a system is larger than another system's then it means that the former remained in its initial state longer and thus its transformation occurred later.

In addition, an average *abstract shape* of the model is provided as an *envelope* of this model. The envelope shows all the interactions that happened over time (during a given time window or the whole run). Each possible interaction is represented as a *3D bendy pipe* going from one interacting system to the other and passing in its centre by the context. The width of a pipe (linking two systems through the context) depends on the amount of times an interaction occurred within the recording window. This envelope therefore gives an overview, an average, of all the interactions and transformations a model underwent. Figures 8(f) and (g) show the envelope of the fire-spreading model at two stages in time.

Finally, the time since the last involvement in any interaction can be recorded for each system so that systems not involved in any recent computation can be hidden (time occlusion), revealing only the relevant systems. Systems that recently interacted are opaque and they become increasingly transparent over time if they do not interact. This selection of systems changes over time as systems that have not interacted for a long time might interact again at some other time in the execution. This is illustrated in Figure 9 showing a fire (of a larger size than previous examples to highlight the effect of time) after total combustion with in 9(a) everything displayed and in 9(b) progressively hiding systems and their hierarchies if they do not interact any more over time. (Note that although it might be a bit hard to see the exact difference in time locations between systems on Figure 9(b), the visualiser enables to zoom in, to change the time window the user wants to look at, and provides a significantly higher screen resolution.)

To navigate in the structure's space, the camera can be moved and rotated in any direction to enable a user to find the best view.

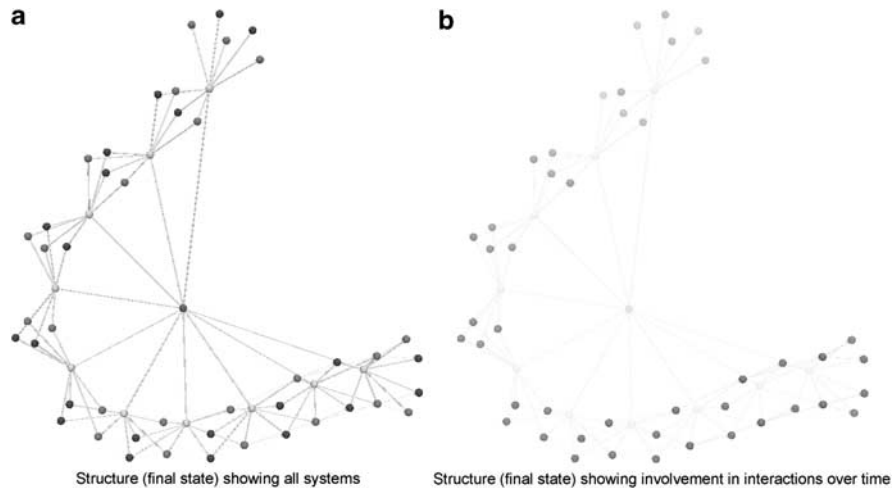


Figure 9: Large-scale fire (10 neighbourhoods) after total combustion using 3D structure views: (a) shows the structure of the whole model with all systems and hierarchy; (b) shows the same state but with systems involvement in interactions over time displayed with transparency. It can be observed that the fire ignition is almost transparent whereas the last neighbourhoods are opaque.

First case study: A bistable gene network

In order to demonstrate the utility of the visualisation framework introduced here this section uses a form of *gene regulatory network* called a *bistable gene network* that was first visualised in Le Martelot and Bentley.³³ The network used here has also been used in Phillips *et al*²⁷ in an approach for visualising stochastic π -calculus models. The aim is then to show that visualising such model using SC provides all the material for the analysis of the model's behaviour over time, whether step by step or overall.

First the model is presented. Then the systemic analysis is performed to create the SC model. The following section analyses the model's behaviour and compares it with the stochastic π -calculus model's. Finally, the SC visualisation and its analysis are presented.

Model presentation

A bistable gene network is a form of gene regulatory network (GRN). A GRN is a collection of DNA segments (genes) in a cell interacting with each other indirectly through the proteins they produce. These regulations govern the rates at which genes create proteins. A bistable gene network is a GRN with two distinct possible states. The bistable gene network model used here was presented in³⁹ and is summarised in Figure 10.

In this case study, the network has two genes a and b that can, respectively, produce proteins A and B at a given rate. In physics a rate would be an amount of proteins produced per second, in this computer model the rate is the probability of production per systemic interaction. In the network, gene b can be repressed (or inhibited) by protein A and then produces B at a much slower rate than

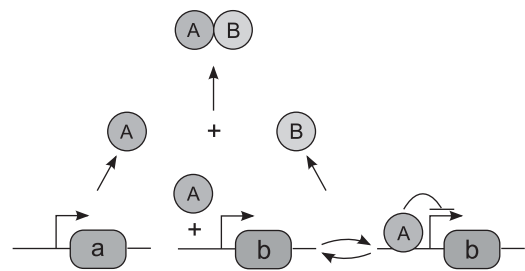


Figure 10: Bistable gene network obtained by an evolutionary procedure *in silico* in François and Hakim.³⁹ Genes a and b can, respectively, produce proteins A and B at a given rate. Gene b can be repressed by protein A and then produces proteins B at a much slower rate. Proteins A and B can irreversibly bind to form a complex that cannot inhibit any gene and eventually degrades. Two scenarios are thus possible: either proteins A will initially inhibit gene b and proteins A will be abundant, constantly repressing gene b ; or gene b generates a large amount of B , systematically binding to proteins A , bringing proteins A to a low level and preventing a constant repression of gene b by proteins A .

if not inhibited. In addition, proteins A and B can irreversibly bind to form a complex that cannot inhibit any gene. This complex then eventually degrades. From these rules, two scenarios are possible: either proteins A initially inhibit gene b and proteins A are abundant, constantly repressing gene b ; or gene b generates a large amount of proteins B , systematically binding to proteins A , bringing proteins A to a low level and preventing a constant repression of gene b by proteins A .

The rates that were used are the ones used in Phillips *et al*.²⁷ They are the same as in François and Hakim³⁹

Table 2: Bistable gene network's reactions rates used in François and Hakim³⁹ except for the second and fourth that were modified in Phillips *et al.*²⁷ The symbol ':' means bound; $A:B$ thus means a protein A is bound to a protein B .

Reactions	Rates
$a \rightarrow a + A$	0.20
$A \rightarrow \emptyset$	0.002
$b \rightarrow b + B$	0.37
$B \rightarrow \emptyset$	0.002
$A + B \rightarrow A : B$	0.72
$A : B \rightarrow \emptyset$	0.53
$b + A \rightarrow b : A$	0.19
$b : A \rightarrow b + A$	0.42
$b : A \rightarrow b : A + B$	0.027

except for the degrading rates of unbound proteins A and B . Table 2 provides all the reaction rates.

Systemic analysis

As mentioned earlier, a systemic analysis is necessary to identify and interpret the appropriate systems and their organisation. The first stage is to identify the low-level systems (that is, determine the level of abstraction to be used). This model involves genes and proteins so the level of abstraction should be the one of proteins and genes.

A possible approach is to take a system for each gene a and b . For the proteins it is less clear as this model involves amounts of proteins. In such case, two ways of modelling can be considered: modelling each protein with a dedicated system, or modelling all proteins with a single system that holds a protein quantity variable. In this model, what is to be observed are the two possible states of the network: A is abundant or B is abundant. These states are because of the inhibited or not state of gene b , and to the creation of protein complexes making proteins A unable to bind to gene b . To record the history of proteins A and B interactions, one protein system with a quantity variable is well suited: its amount of interaction with a gene or other proteins system would reflect the quantity held (no interaction if no protein) and would reveal the network's behaviour. If there are many interactions between proteins systems A and B then proteins A and B are present in large amounts, and thus gene b is not inhibited. Alternatively, if interaction between gene b and proteins B is weak, then gene b is inhibited, and interaction between gene a and proteins A should be significantly more important than between gene b and proteins B . In addition, having many protein systems would add some complexity to the analysis (many more systems to observe), which is here not necessary.

Genes regulation turns the information on genes into gene products (here proteins). DNA chunks (here genes) can be transcribed into a messenger RNA chunk carrying a message for the protein-synthesising machinery of the cell. This process can be approximated by using an

RNA context that makes interact a gene system and its corresponding proteins system to potentially increase the amount of proteins. The RNA chunk involved in the production of proteins A and B being different, the contexts of interactions between gene a and proteins A , and gene b and proteins B can therefore, respectively, be RNA A and RNA B .

Proteins degrade (proteolysis), therefore proteases (enzymes that conducts proteolysis) have to be part of the model. A proteases system modelling all proteases present in the cell is appropriate (as opposed to many proteases systems) as proteases quantity is not involved in this model. Proteins degrade as a result of chemical interactions within a physical space, therefore an approximation can consider the local energy involved in the process as being context of interaction between proteins and proteases. Local energy is then also involved in the binding of proteins A and B . Finally, corepressor and inducer can be the contexts respectively binding and unbinding proteins A to gene b .

The interactions are summarised in Table 3 and illustrated in Figure 11.

Model behaviour

To allow comparison and ensure the model behaves like a bistable switch similarly to the stochastic π -calculus model by Phillips *et al.*²⁷ the model was run starting with no protein and recorded the evolution of proteins over time. Thirty runs were performed and showed that the network always falls in one or the other of the two states. Figure 12 shows the evolution of protein quantities along two representative runs in each simulation (SC and the Stochastic Pi-Machine $SPiM^1$) for the two possible states the network can fall in. These results are similar to the ones presented in Phillips *et al.*²⁷ and show that the network presents the same stable states.

Visualisations

The abstract structure is designed to display the various interactions and transformations that occurred over time in a model. This case study focusses on the envelope, expected to be particularly relevant as the amount of interactions between systems will appear on it, thus revealing the state and behaviour of the model. Also the universe, not playing a computational role in the model, is hidden from visualisations to only display the relevant systems. Figure 13 provides in (a) the colour scheme and then explains the model at various stages of progression, first over early steps and highlighting in the end the two possible behaviours.

Figure 13(b) shows the network in its initial state. No interaction happened yet and systems are located around

¹ <http://research.microsoft.com/en-us/projects/spim>.

Table 3: Bistable gene network interactions. q is the quantity of proteins. q_{b_A} and q_{b_B} are, respectively, the bound proteins of type A and B . δ represents a quantity variation during an interaction. The notation $(S_1)S_2$ indicates that a system S_1 is in the scope of another system S_2 and reciprocally. *inhibit* is a flag indicating the state of inhibition of gene b .

Interactions				Results
Gene_a	}-	RNA_A	-{	Proteins_A[q] → Gene_a Proteins_A[q + δq]
Proteins_A[q, q_b]	}-	Energy_A	-{	Proteases → Proteins_A[q - δq , $q_b - \delta q_b$]Proteases
Gene_b	}-	RNA_B	-{	Proteins_B[q] → Gene_b Proteins_B[q + δq]
Proteins_B[q, q_b]	}-	Energy_B	-{	Proteases → Proteins_B[q - δq , $q_b - \delta q_b$] Proteases
Proteins_A[q, q_{b_A}]	}-	Energy_AB	-{	Proteins_B[q, q_{b_B}] → (Proteins_A[q - δq , $q_{b_A} + \delta q$])(Proteins_B[q - δq , $q_{b_B} + \delta q$]), if $q_{b_A} > 0$ and $q_{b_B} > 0$
(Proteins_A[q_{b_A}])(Proteins_B[q_{b_B}])	}}-	Energy_AB	→	Proteins_A Proteins_B, if $q_{b_A} = 0$ or $q_{b_B} = 0$
Proteins_A[q]	}-	Corepressor	-{	Gene_b[!inhibit] → (Proteins_A[q-1] () Gene_b[!inhibit])
(Proteins_A[q] () Gene_b[!inhibit])	}}-	Inducer	→	Proteins_A[q+1] Gene_b[!inhibit]

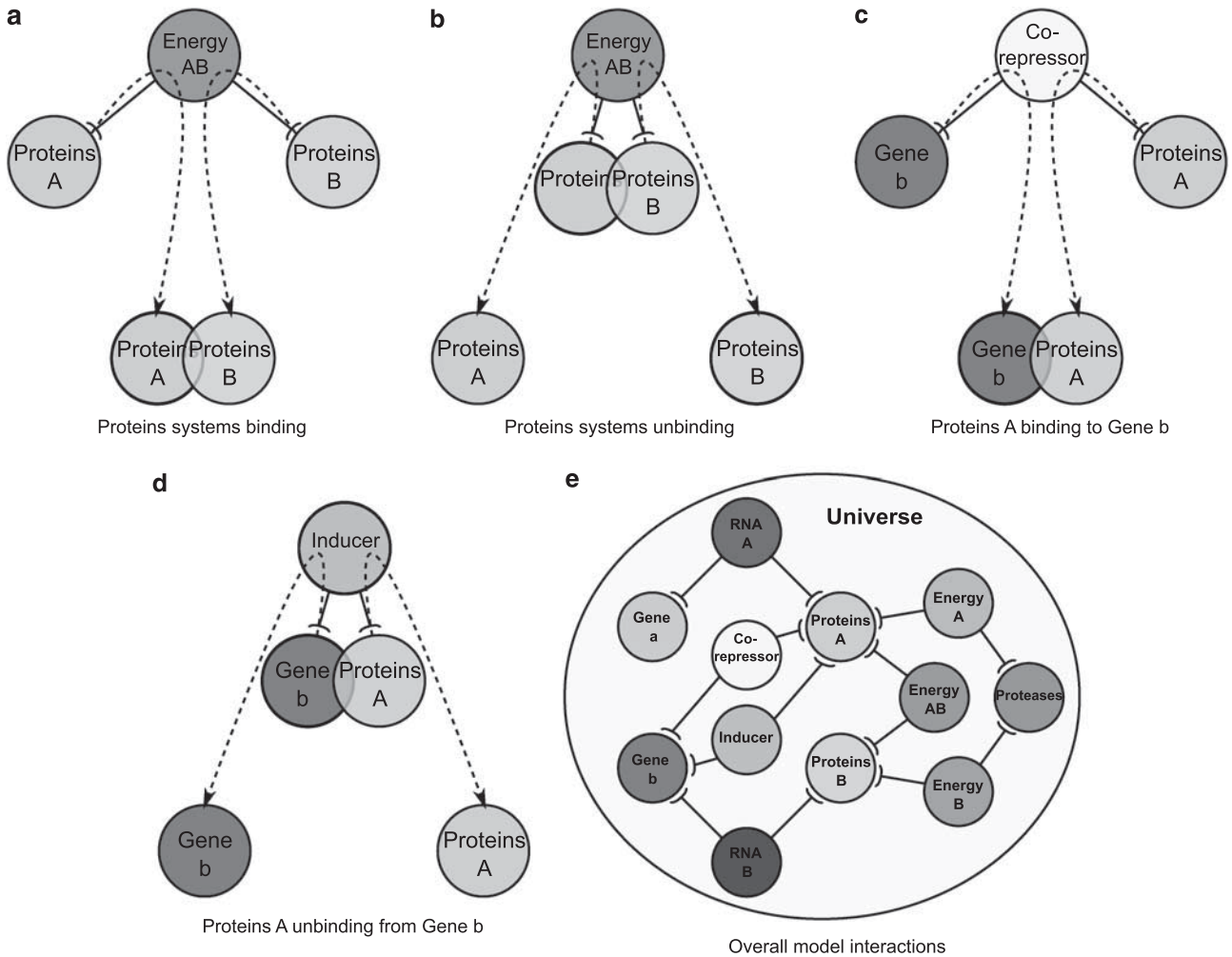


Figure 11: Interactions and structure of a bistable gene network model. (a)–(d) show the potential structural changes. (e), reproduced from Le Martelot and Bentley,³³ shows the overall interaction patterns: proteins are created from genes in the context of RNA, proteins can bind to form a complex using local energy, they can also degrade using local energy. Proteins A can inhibit gene b using a corepressor and an inducer can unbind proteins A from gene b .

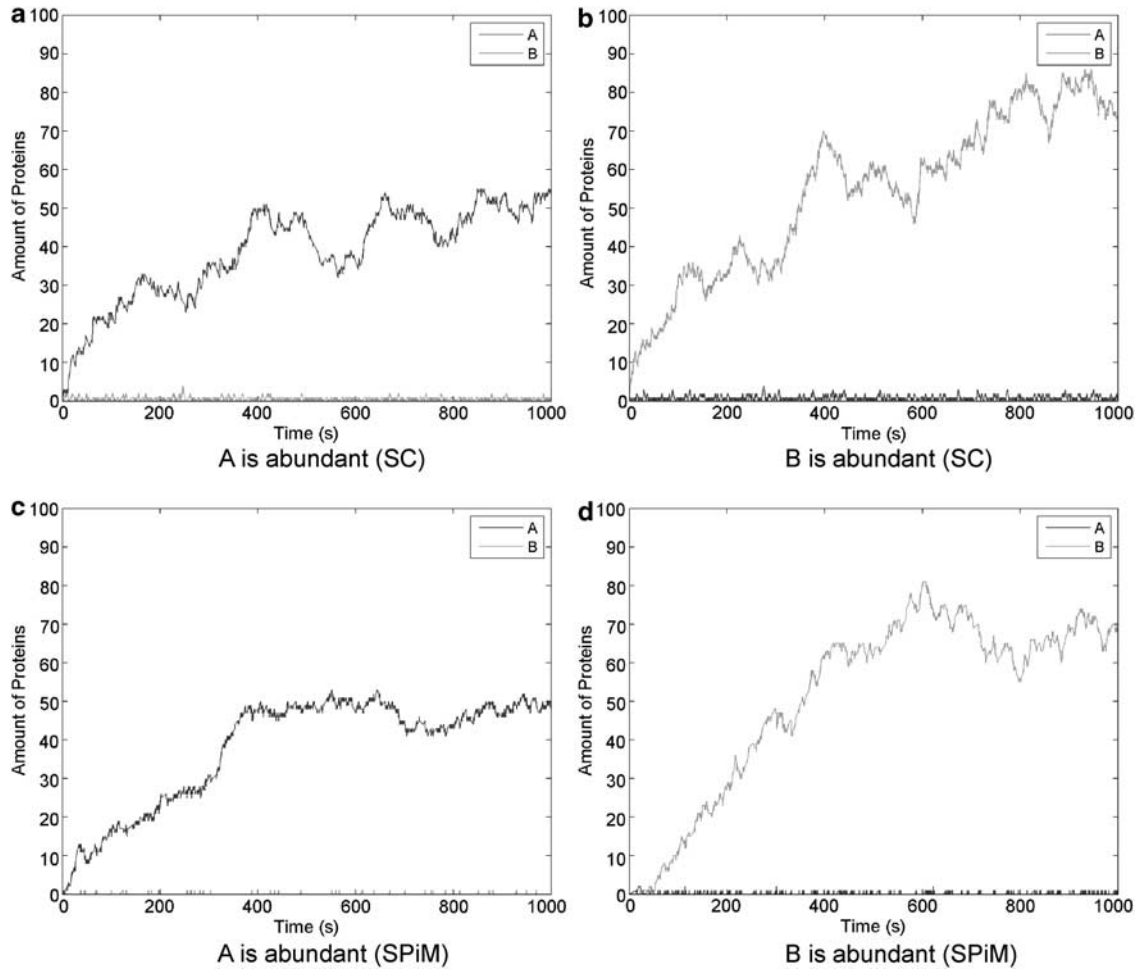


Figure 12: Evolution of the amount of proteins over time in the two possible states of the switch: (a) and (b), and (c) and (d) respectively show the evolution if proteins A and B are highly transcribed for the SC model and for the stochastic π -calculus model.

the universe (not shown here but would be located at the centre).

Figure 13(c) shows the envelope after a couple of interactions. It reveals that proteins A and B were produced (genes *a* and *b*, respectively, produced proteins A and B in the respective contexts *RNA_A* and *RNA_B*). Proteins B appear to have been produced more (more interactions between gene *b* and proteins B in the context of *RNA_B*) as the visualisation shows more interaction between gene *b* (sea blue) and proteins B (light blue) in the context of *RNA_B* (dark blue) than between gene *a* (green) and proteins A (light green) in the context of *RNA_A* (dark green) (the pipe between the gene and the proteins systems passing through the RNA system, representing the amount of interaction, is larger in the former than in the latter).

Figure 13(d) shows that some proteins A and B are bound together. An interaction between both proteins in the context of energy (fuchsia) occurred and a hierarchy link between proteins A and B systems appeared (proteins

A and B systems are in the scope of each other). There is also a similar production state of proteins A and B as the visualisation reveals a similar amount of interaction between genes and proteins of both type (pipes equally large).

Figure 13(e) shows the first degradation of proteins for proteins B as an interaction between proteins B and proteases (red) appeared.

Figure 13(f) shows an equal amount of proteins A and B degraded, with similar amount of interactions between proteases and proteins of both type. A higher amount of protein B has been produced and gene *b* is still not inhibited. This state already shows signs of advantage proteins B are taking over proteins A as the production rate of proteins B is higher than the production rate of proteins A (see Table 2) and proteins A can be produced faster than proteins B only if gene *b* is repressed.

Figure 13(g) shows that gene *b* got inhibited and then released as interactions occurred between gene *b* and proteins A in the context of corepressor (yellow) and

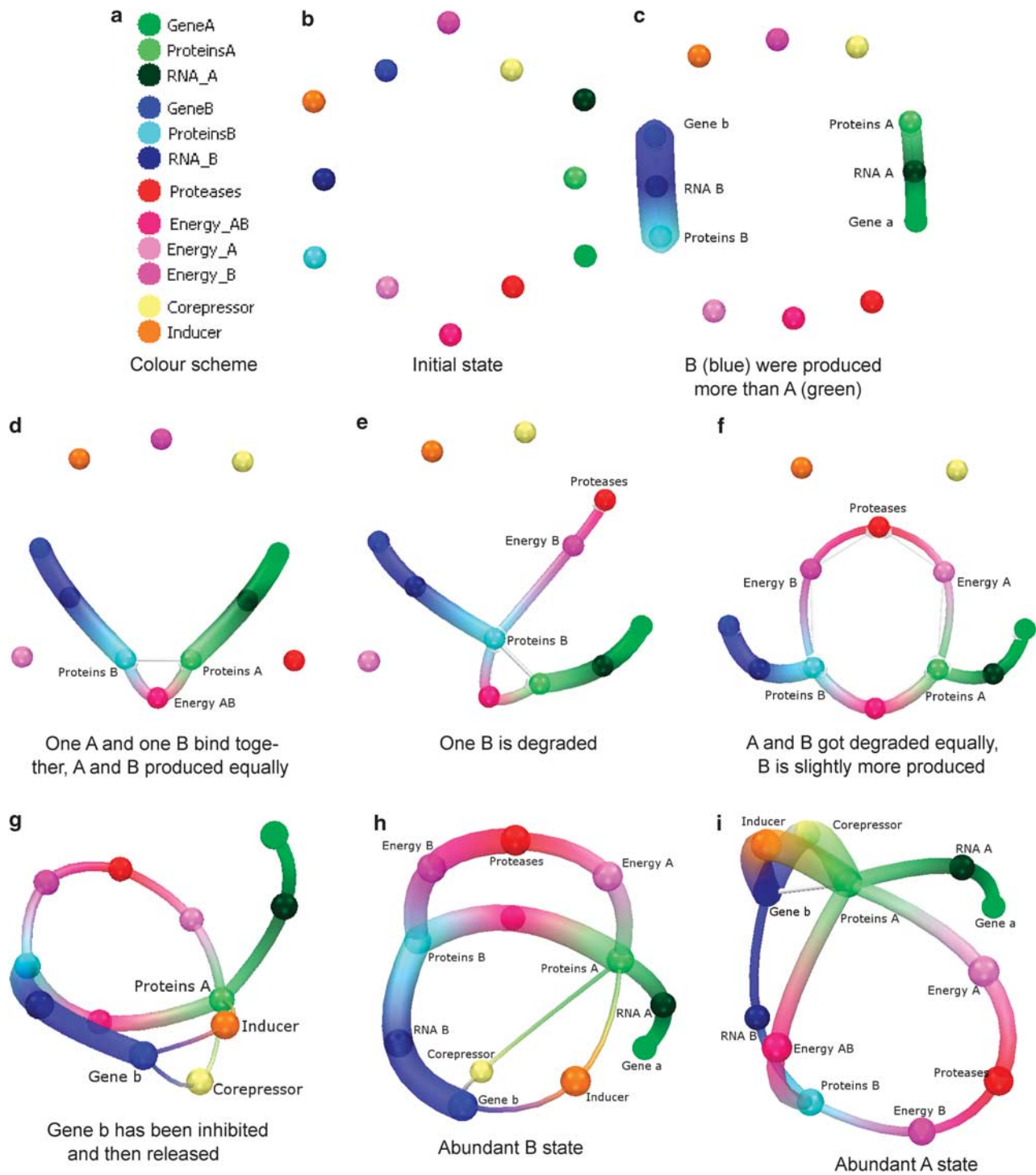


Figure 13: (reproduced from Le Martelot and Bentley³³) Bistable gene network visualisation: (a) colour scheme; (b) network in its initial state; (c) envelope after a few interactions, proteins A and B were produced with protein B produced more; (d) some proteins A and B bound together; (e) some proteins B degraded; (f) equal amount of proteins A and B degraded, higher amount of proteins B produced; (g) gene b got inhibited and then released; (h) protein B abundant state; (i) proteins A abundant state.

inducer (orange). Proteins B appear still advantaged over proteins A (higher production of proteins B): the switch is most likely going to carry on that way and proteins B would be abundant in the final stage.

Figure 13(h) shows the protein B abundant state, as expected, and looking very similar to Figure 13(g). Both proteins degraded, with slightly more proteins B that degraded compared to proteins A . Gene b got little inhibited (very few interactions between gene b and proteins A in the context of corepressor and inducer as shown by the thin pipes around the corepressor and inducer systems compared to others). Proteins A and B bound a lot (many interactions between them in an *Energy_AB* context) and then degraded, but proteins B are more numerous, and therefore some proteins B remained whereas proteins A all disappeared.

Figure 13(i) shows the alternative final state where proteins A are abundant. This state looks different from Figure 13(h) where proteins B are abundant. The main difference lies in the amount of interactions involving the corepressor and the inducer systems. Interactions between gene b and proteins A through, respectively, the corepressor and the inducer occurred the most (bigger pipes), revealing a constant repression and release of gene b . In this respect, a hierarchy link between proteins A and gene b can be spotted, showing that at the moment of the snapshot gene b and proteins A are in the scope of each other, which means here that they are bound to each other. This repression of gene b led to a lower production of proteins B (fewer interactions between gene b and proteins B than between gene a and proteins A) and a total degradation of proteins B by proteases (amount of

interaction between gene b and proteins B as important as between proteins B and proteases).

Second Case Study: A MAPK Signalling Cascade

The bistable gene network was an example of a non-predictable system with a behaviour nevertheless reasonably simple. The second case study is a significantly bigger and more complicated system: a *mitogen-activated protein kinase* (MAPK) *cascade*. This network has also been used in Phillips *et al*²⁷ and is reused for the same reasons as mentioned in the previous case study.

First the model is presented. Then the systemic analysis is performed to create the SC model. The following section analyses the model behaviour and compares it with the stochastic π -calculus model's. Finally, the SC visualisation and its analysis are presented.

Model presentation

The model presented here is an MAPK cascade, as presented in Huang and Ferrell⁴⁰ and summarised in Figure 14.

A protein kinase is a kinase enzyme that modifies other proteins by chemically adding phosphate groups to them (phosphorylation). MAPKs are serine/threonine-specific protein kinases that respond to extracellular stimuli (mitogens) and regulate various cellular activities, such as gene expression, mitosis, differentiation and cell survival/apoptosis.⁴¹ Here, extracellular stimuli lead to the activation of an MAPK via a signalling cascade composed of

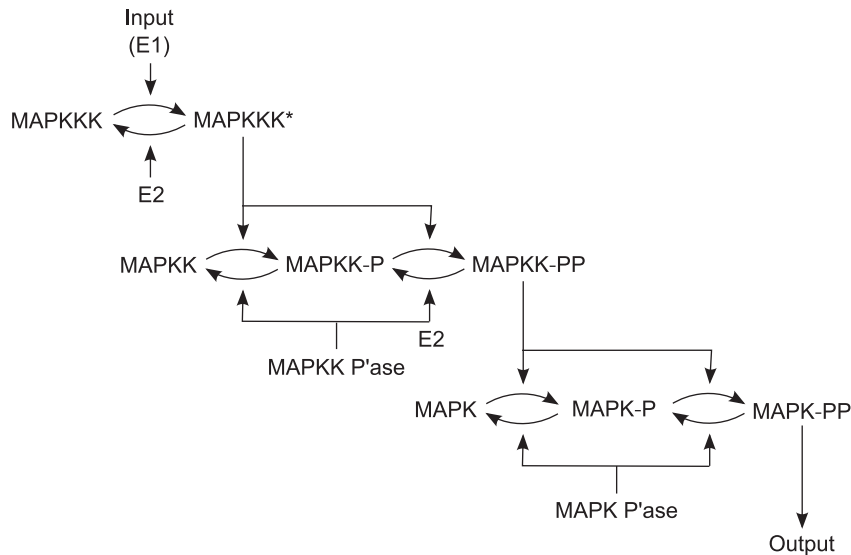
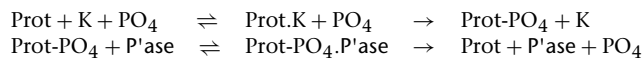
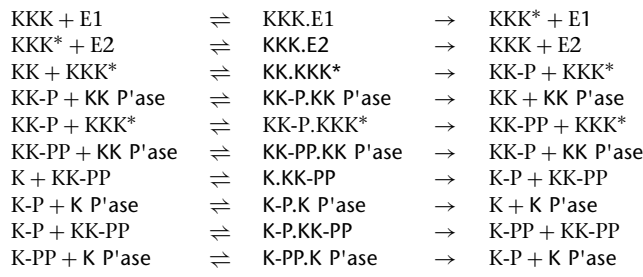


Figure 14: MAPK cascade: the activation of both $MAPK$ and $MAPKK$ requires the phosphorylation of two sites, $MAPKKK$ is activated by extracellular stimuli named here $E1$. $MAPK-P$, $MAPK-PP$ and $MAPKK-P$, $MAPKK-PP$, respectively, denote singly and doubly phosphorylated $MAPK$ and $MAPKK$. $MAPKKK^*$ denotes activated $MAPKKK$. $E2$ denotes the enzyme deactivating $MAPKKK^*$. $P'ase$ denotes *phosphatase*.

Table 4: Principle of phosphorylation and dephosphorylation in the MAPK cascade. *Prot*, *K*, *P'ase* and PO_4 , respectively, stand for *protein*, *kinase*, *phosphatase* and *phosphate*.**Table 5:** List of reactions involved in the MAPK cascade. Phosphates are only shown here when bound to a protein (ie KKK^* , $KK-P$, $KK-PP$, $K-P$ and $K-PP$).

MAPK, MAPKK (mitogen-activated protein kinase kinase) and MAPKKK (mitogen-activated protein kinase kinase kinase). An MAPKKK is a kinase enzyme that phosphorylates an MAPKK, which itself phosphorylates an MAPK. Reciprocally, phosphatase enzymes can remove a phosphate group from its substrate (dephosphorylation).

In this case study, each kinase (respectively phosphatase) first binds to a protein and then can either add to (remove from) it a phosphate group or unbind, letting the protein as it is. Table 4 summarises this principle with generic reaction equations.

All possible reactions in this case study are listed in Table 5. To simplify the notation, *MAPKKK*, *MAPKK* and *MAPK* are from now on, respectively, referred to as *KKK*, *KK* and *K*.

Each action can be performed at a given rate. As in the previous case study, rates are here transcribed into probabilities. In physics a rate would be an amount of enzymes binding, phosphorylation or dephosphorylation per second; in this computer model the rate is the probability for a reaction to happen per systemic interaction. The rates that were used here are the one from the code example in Phillips *et al.*,²⁷ in which all transitions have a rate of *1.0*.

Systemic analysis

As mentioned earlier, a systemic analysis is necessary to identify and interpret the appropriate systems and their organisation. This model involves enzymes (kinase and phosphatase) as well as phosphate groups, and therefore the level of abstraction should be the one of enzymes and phosphate groups.

A possible and straightforward approach is to use one system per protein (kinase or phosphatase) or phosphate

group. The model should then contain as many phosphate groups as necessary to enable the reactions from Table 5 to occur without shortage of phosphate groups (this potential situation is not part of this study). Considering that kinase can bind to one (for *KKK*) or two (for *KK* and *K*) phosphates, then the amount of phosphate systems necessary is given by equation (1) in which $|X|$ is the cardinal number of the given set X and *Phosphates*, *KKKs*, *KKs* and *Ks* are, respectively, the set of systems representing *phosphate groups*, *KKK*, *KK* and *K*.

$$|\text{Phosphates}| = |\text{KKKs}| + 2 \times (|\text{KKs}| + |\text{Ks}|) \quad (1)$$

The interactions in phosphorylation and dephosphorylation are happening between a kinase (phosphorylated or not depending on the reaction) and a phosphate group provided that the right activated kinase is present (a kinase is activated once it is phosphorylated enough to be a reactant, as determined by the rules from Table 5). Activated kinase systems can thus appropriately be considered as contexts of interaction between a kinase system and a phosphate system. During phosphorylation, a phosphate group is bound to a non-activated kinase; therefore, phosphate and kinase systems bound to each other should have a relationship that reflects this connection. One way to achieve that is to set the kinase and the phosphate systems within the scope of each other. Eventually, the kinase might get activated (simply phosphorylated *KKK* or doubly phosphorylated *KK*). Reciprocally, dephosphorylation unbinds phosphate systems from kinase systems and deactivates activated kinase systems.

However, looking at Tables 4 and 5 it can be observed that the activated kinase (or the phosphatase) first approaches a kinase (or phosphorylate (or dephosphorylate) but may eventually not create any reaction. The interactions between kinase or phosphatase systems and phosphate systems must therefore take this into account. As we chose all reaction rates to be equal to *1.0*, each interaction thus has a probability of *0.5* to create a change in the phosphorylation state of the interacting kinase (for example probability of *0.5* that a phosphate is bound to or unbound from a kinase and probability of *0.5* that no change occurs). The SC interactions are summarised in Table 6.

To ensure that the context systems select the appropriate kinase and phosphates (the ones bound to each other in the case of dephosphorylation), in the same manner as physical location would indicate which system is bound to which other system, the notion of scope can be used to encapsulate in an abstract space phosphate and kinase systems that can interact together. Figure 15 illustrates the area scope distribution and Figure 16 summarises the whole model organisation.

A visualisation of a small MAPK cascade is provided for illustration in Figure 17. It shows an MAPK model involving two *KKKs* and five *KKs* and *Ks* in its initial state using the 3D explorer and the 3D abstract structure.

Table 6: MAPK cascade interactions. The notation $(S_1)(S_2)$ indicates that a system S_1 is in the scope of another system S_2 and reciprocally. The notation $|$ in the results indicates several possible outcomes (here two outcomes with a probability of 0.5 each). The states of activation of kinases are indicated with flags (using the same as in Table 5) in addition to the scope relationship they potentially share with phosphate systems.

Interactions	Results
$KKK \} - E1 \} - \{ \text{Phosphate}$	$(KKK[*] () \text{Phosphate}) KKK \text{Phosphate}$
$(KKK[*] () \text{Phosphate}) \} - E2$	$KKK \text{Phosphate} (KKK[*] () \text{Phosphate})$
$KK \} - KKK[*] \} - \{ \text{Phosphate}$	$(KK[P] () \text{Phosphate}) KK \text{Phosphate}$
$(KK[P] () \text{Phosphate}) \} - KK \text{P'ase}$	$KK \text{Phosphate} (KK[P] () \text{Phosphate})$
$KK[P] \} - KKK[*] \} - \{ \text{Phosphate}$	$(KK[PP] () \text{Phosphate}) KK[P] \text{Phosphate}$
$(KK[PP] () \text{Phosphate}) \} - KK \text{P'ase}$	$KK[P] \text{Phosphate} (KK[PP] () \text{Phosphate})$
$K \} - KK[PP] \} - \{ \text{Phosphate}$	$(K[P] () \text{Phosphate}) K \text{Phosphate}$
$(K[P] () \text{Phosphate}) \} - K \text{P'ase}$	$K \text{Phosphate} (K[P] () \text{Phosphate})$
$K[P] \} - KK[PP] \} - \{ \text{Phosphate}$	$(K[PP] () \text{Phosphate}) K[P] \text{Phosphate}$
$(K[PP] () \text{Phosphate}) \} - K \text{P'ase}$	$K[P] \text{Phosphate} (K[PP] () \text{Phosphate})$

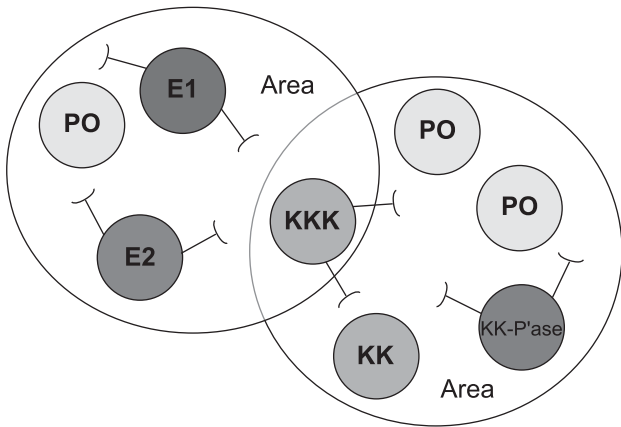


Figure 15: Distribution in scope areas of kinase and phosphates. Kinase systems are shared between areas where they can be activated/deactivated and areas where they can activate other kinase systems. Here two areas are shown, one with a KKK and the other one with a KK that can be activated in two steps by KKK provided KKK was previously activated by $E1$.

Model behaviour

For this study, the model is initialised with a configuration comparable to the one from Phillips *et al*²⁷: 10 KKK s, 100 KK s, 100 K s, one Enzyme1, one Enzyme2, one KK -Phosphatase and one K -Phosphatase systems. The amount of phosphate groups is deducted from equation (1): 410 phosphate systems. The amount of abstract local space areas is given by the amount of kinases (one area per kinase system): 210 area systems. All these systems are located within one universe system.

To ensure the model behaves like the MAPK cascade stochastic π -calculus model by Phillips *et al*,²⁷ the model was run with the same initial conditions: all proteins in a non-phosphorylated initial state (left side of the reactions in Table 5). The evolution over time of the amount of each activated kinase (simply phosphorylated KKK ,

double phosphorylated KK and K) in the system was recorded. Thirty runs were performed. Figure 18 shows the evolution of activated kinase quantities in the SC model and in the stochastic π -calculus model (results presented in Phillips *et al*²⁷). For the SC model the results are averaged over the 30 runs. The figure shows that the SC model, like the one from Phillips *et al*,²⁷ behaves as described in Huang and Ferrell:⁴⁰ the signal response is increasingly getting steeper as the cascade is traversed.

In addition, Figure 18(a) shows that in the SC model the amount of activated KKK systems averages around five. Considering that there are 10 KKK systems and that the probability of KKK to be phosphorylated or dephosphorylated is 0.5, we can write equation (2) (where KKK^* stands for *activated* KKK and t is a discrete time).

$$|KKK^*s_{t+1}| = |KKK^*s_t| + (|KKKs_t| \times 0.5) - (|KKK^*s_t| \times 0.5) \quad (2)$$

At initialisation there are 10 non-activated KKK s and no activated KKK . Iterating the equation over t then settles both values on 5.

Visualisations

In this section, the visual analysis performed on the MAPK cascade model enables to discover a non-obvious pattern that would be hard to discover any other way.

As shown experimentally in Figure 18, the MAPK cascade functions as an amplifier where the amount of phosphorylated kinases at a level in the cascade increases faster than the amount of phosphorylated kinases at its previous level. To understand and visualise how this happens the informational structure of the cascade can be visualised with its envelope to show what systems are interacting with which other systems and responsible for which changes.

Moreover, considering that kinases keep changing their state (non-phosphorylated, simply phosphorylated, doubly phosphorylated), the tree ring view showing the states over time of the systems is relevant to look at.

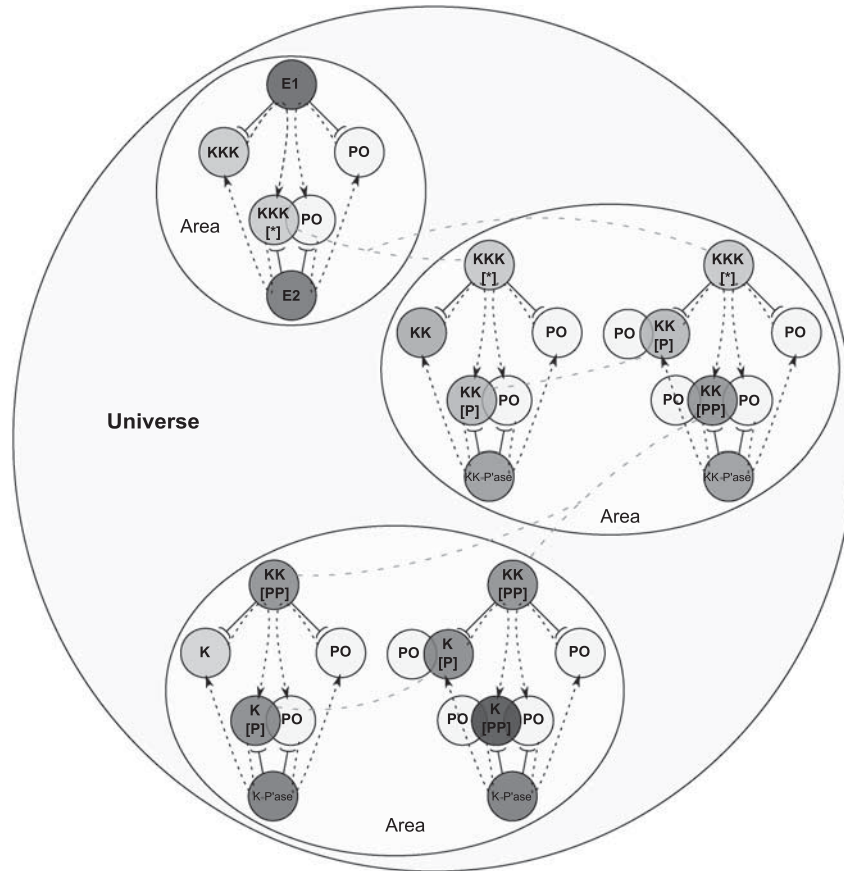


Figure 16: Structure of an MAPK cascade model. Kinase systems can be activated by phosphorylation in presence of the right activated kinase. Activated kinases can activate non-activated kinase of a different kind (KKK activates KK which activates K) thus creating the cascade. The black dashed arrows represent the transformation of systems over computation. The longer-dashed lines (in red in the coloured version) indicate systems that can potentially be the very same system but are represented twice or more in several places for drawing clarity.

Figure 19 shows an example of the three kinases in their activated state. The displayed states reveal through the changes in coloured rings that while the presented KK in Figure 19(c) went here successively from non-activated to simply phosphorylated and doubly phosphorylated, both KKK and K oscillated between phosphorylated states (including activated states) in the past before reaching their current activated state.

To focus on the kinases systems and their interactions only, the universe and the area systems are discarded from this set of visualisations. Figures 20–22 provide visualisations snapshots over time of a single run of the MAPK model.

Figure 20 shows three early stages, after respectively 10, 25 and 50 interactions, and displays the changes in systems states and the envelope. Only systems involved in at least one computation are shown, the remaining systems being hidden for clarity.

Figure 20(a) (after 10 interactions) shows the interactions that occurred since the very beginning of computation. Three KKKs got phosphorylated, among which two

were dephosphorylated and then phosphorylated again (as the envelope reveals by showing that more interactions happened with *Enzyme1* than with *Enzyme2* involving the two KKKs respectively located at the bottom and at the right). The activated KKK at the bottom phosphorylated a KK in turn dephosphorylated by the *KK-phosphatase*. The activated KKK at the right phosphorylated a KK. In Figure 20(b) (after 25 interactions), four KKKs are activated and phosphorylating KKs. Four groups of KKs and phosphates centred around an activated KKK at the four corners can be observed. Thus far four KKKs are active and seven KKs are phosphorylated. Finally, Figure 20(c) (after 50 interactions) shows the progression of the previous state with more and more KKs being phosphorylated, and the first Ks being phosphorylated. The amplification effect is visible with each activated KKK locally phosphorylating in turn several KKs (groups of phosphates and KKs gathering around activated KKKs).

From these first three stages we can observe a progression of the global phosphorylation. The amplification effect is observable with increasingly more KKs being

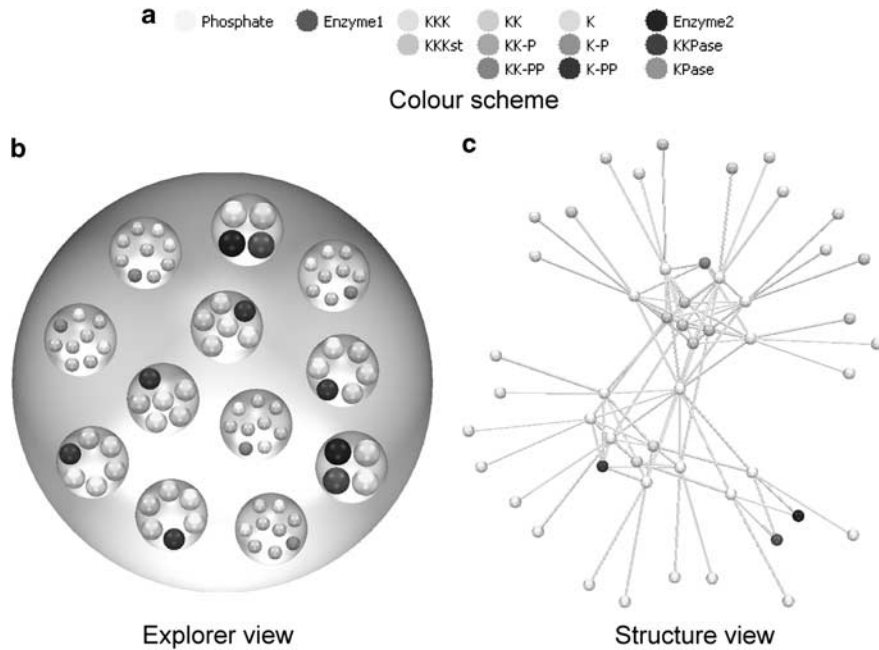


Figure 17: Small MAPK cascade model (Two KKKs, five Ks and five Ks) visualised with the 3D explorer and the abstract structure.

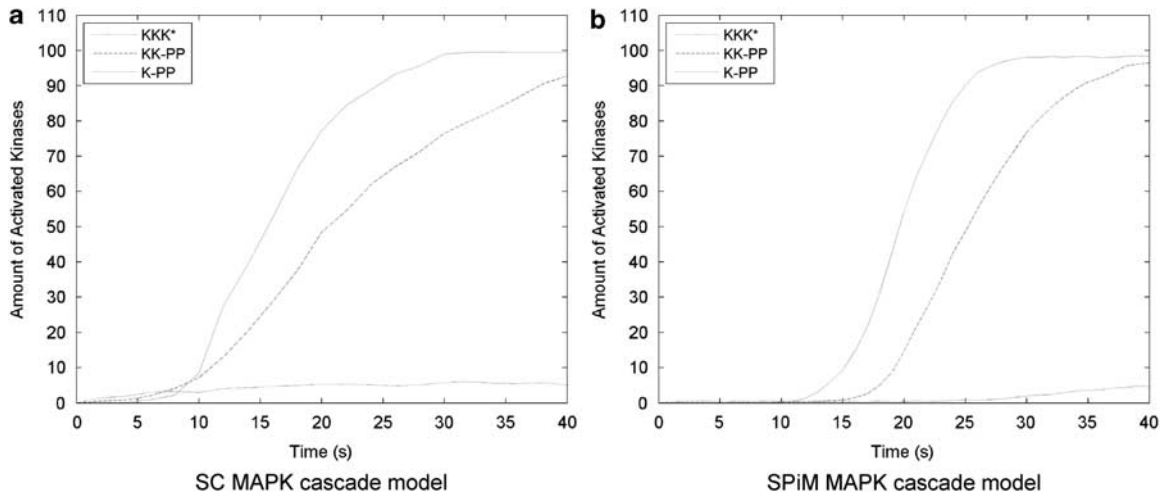


Figure 18: Evolution in the MAPK cascade of the amount of kinases over time (averaged over 30 runs) (a) in the SC model and (b) in the stochastic π -calculus model.

phosphorylated while the amount of activated *KKKs* remains stable. It is expected from equation (2) that half of the *KKKs* (five *KKKs*) on average would remain phosphorylated at a time. With only one *KK-phosphatase* to counterbalance the effect of several activated *KKKs*, phosphorylation of *KKs* is inevitably more likely to occur than their dephosphorylation. The same phenomenon is therefore expected to occur on the next cascade level with even more activated kinases (more activated *KKs* than activated *KKKs*), leading to a faster phosphorylation of *Ks*.

To investigate this, Figure 21 shows two later stages after, respectively, 100 and 150 interactions, following the stages from Figure 20, and illustrating the fast phosphorylation of *Ks*. Note that the changes in systems states (tree ring view) are no longer shown as the global view of the model is getting too large to make it readable at this scale.

Figure 21(a) (after 100 interactions) shows the evolution since Figure 20(c) with more phosphorylated *KKs*, several activated *KKs* (doubly phosphorylated) and *Ks* being phosphorylated in various places. Figure 21(b) (after 150

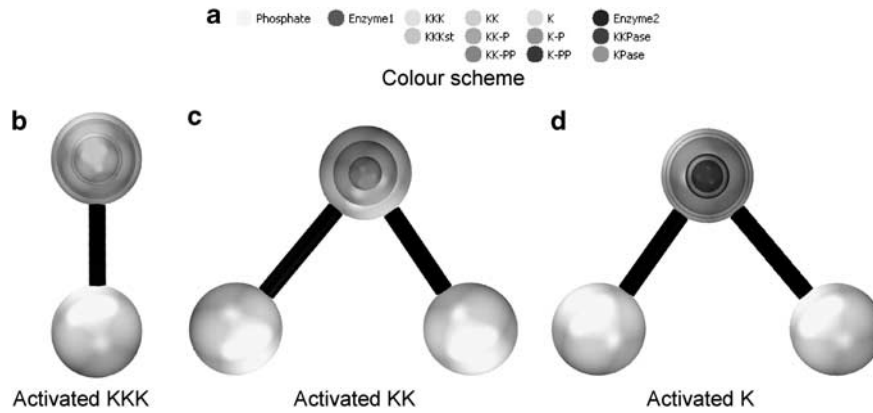


Figure 19: Tree ring view of the three kinases in their activated state.

interactions) shows the fast phosphorylation rate of *Ks* being now as numerous as phosphorylated *KKs*. The notion of local contribution to the global amplification effect well visible in Figure 20(c) for activated *KKKs* is again visible for the next cascade level as groups of *Ks* now also gather around activated *KKs*. As expected the activation of *KKKs* remains stable and the phosphorylation of *KKs* progresses but slower than the phosphorylation of *Ks*. With each activated kinase phosphorylating several kinases of the next level, which in turn phosphorylate several kinases of the level after, there seems to be an exponential phosphorylation effect. Further stages are expected to see a significant increase in phosphorylated *Ks* and final stages should contain a vast majority of doubly phosphorylated *Ks*. Figure 22 illustrates this by showing two late stages taken after 250 and 475 interactions.

Figure 22(a) (after 250 interactions) shows that more phosphorylated and especially doubly phosphorylated *KKs* and *Ks* appeared. At this point there is a similar amount of simply phosphorylated *KKs* (26) and *Ks* (27), and doubly phosphorylated *KKs* (21) and *Ks* (22). From the results presented in Figure 18(a), it is at approximately 250 interactions on average that phosphorylated *Ks* become more numerous over phosphorylated *KKs*. Finally, Figure 22(b) (after 475 interactions) shows a late stage where all (100) *Ks* are doubly phosphorylated whereas only 43 *KKs* (and five *KKKs*) are activated.

The visualisation of interactions over Figures 20–22 thus allowed a visual analysis of the local and emerging global behaviour of the cascade over time.

However, to reach the state of total phosphorylation shown in Figure 22(b), it is still unclear, for instance, what the respective roles of some kinases were (how many were involved doing what). Analysing the *envelope* after the 475 interactions can reveal which systems did not interact, which did and did what, and help to understand the requirements of the model’s behaviour. Previous captions like Figure 21(b) clearly showed that all *KKKs* took part in phosphorylation of *KKs* (the envelope shows

traces of interactions with *KKs* for all *KKKs*), but it is less clear regarding the involvement of *KKs*. Figure 23 shows envelopes after 475 interactions to investigate the implication of *KKs* in the building of the current *Ks* phosphorylation state. Phosphate systems are not displayed for clarity.

Figure 23(a) shows the non-activated *KKs* at the 475 interactions stage from Figure 22(b) and the ones that were involved at least once in a phosphorylation of *Ks* are marked with a red area around them (four of them). Figure 23(b) provides a zoom onto the marked one at the top-right of Figure 23(a). The violet envelope branch is going from red (colour of *KKs*) to blue (colour of *Ks*), revealing an interaction involving a *K*. Figure 23(c) shows the involvement of activated *KKs* (43 of them). The ones in the yellow areas (14 of them) did not contribute to phosphorylation of *Ks* (their envelope has no branch going towards a blue colour). From these two figures we can see that only $4 + (43 - 14) = 33$ *KKs* were actively involved in the current state. Looking at Figure 23(d) showing all systems but phosphates, we observe that only eight systems (in the grey areas on the side), all being *KKs*, were not involved in any computation. (Note that although Figure 23(d) may appear cluttered, the level of information can be reduced, for instance, by hiding some systems or interactions, zooming in, or spread out systems more.) Therefore, out of the 100 *KKs*, 8 were not involved, 33 contributed actively to the phosphorylation of *Ks* and thus $100 - (8 + 33) = 59$ *KKs* were alternatively phosphorylated and dephosphorylated without getting to phosphorylate any *K*. Computationally, the more *KKs* the higher the probability for each *KK* not to be dephosphorylated. Indeed the phosphatase system being here single and only able to interact with one kinase at a time, it is thus less and less likely to dephosphorylate a given kinase the more numerous they are. Therefore, it is noteworthy that the fact that a kinase is not involved in any phosphorylation does not mean it has no impact on the model’s behaviour.

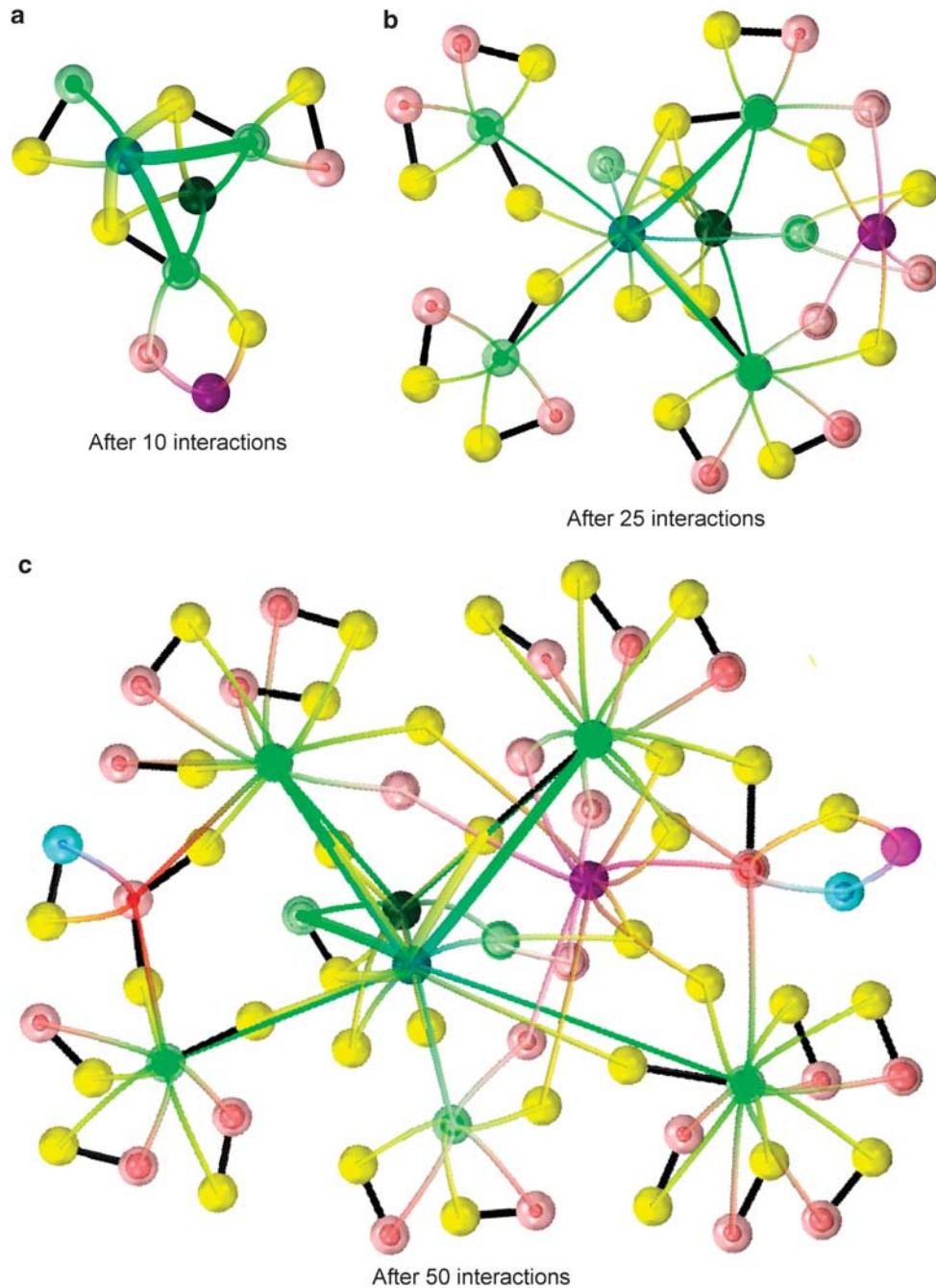


Figure 20: Visualisation over time of the MAPK model at three early stages: (a) after 10 interactions; (b) after 25 interactions and (c) after 50 interactions.

Other interesting situations

Thus far the visualisation framework was illustrated with models of fire chemistry, bistable gene network and mitogen-activated kinase cascade. Other modelling situations were not encountered and are interesting to look at. The following describes two more situations using special examples not related to the previous models.

Systems grouping

One case is the sorting of systems in a scope based on interaction, similarly to a crowd that would gather into clearly defined groups depending on, for instance, the debated subject.

In the explorer, interactions between systems create a momentary force that brings these interacting systems

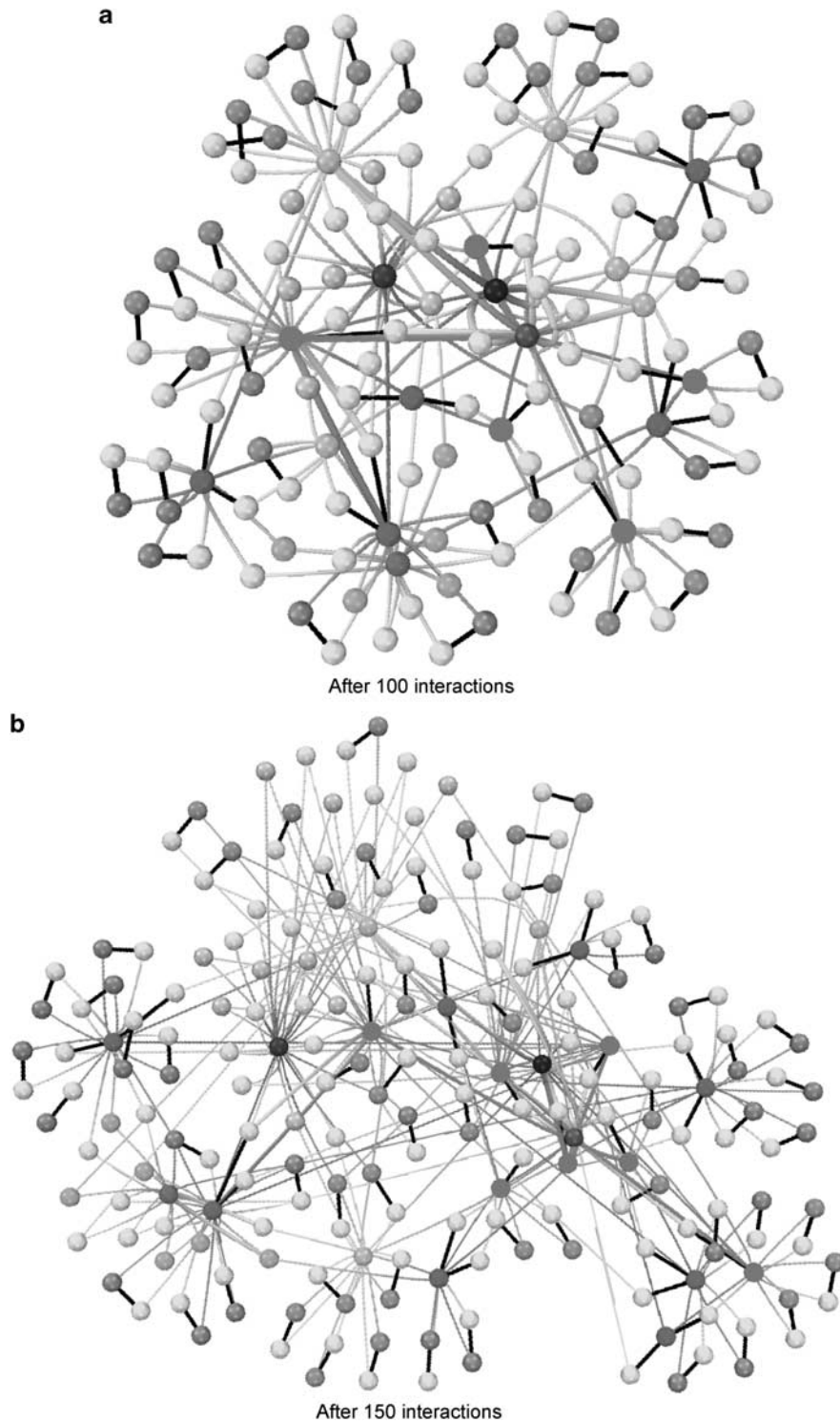


Figure 21: Visualisation over time of the MAPK model at two later stages (following those from Figure 20).

and the context closer towards their centre of mass. This enables to make appear groups of systems interacting with one another and to place these groups away from each other in the space of a scope. Figure 24 shows

an example in which five types of systems are initially randomly placed within a parent system and then are moved according to their interactions. If considering, for instance, people attending a research conference, the

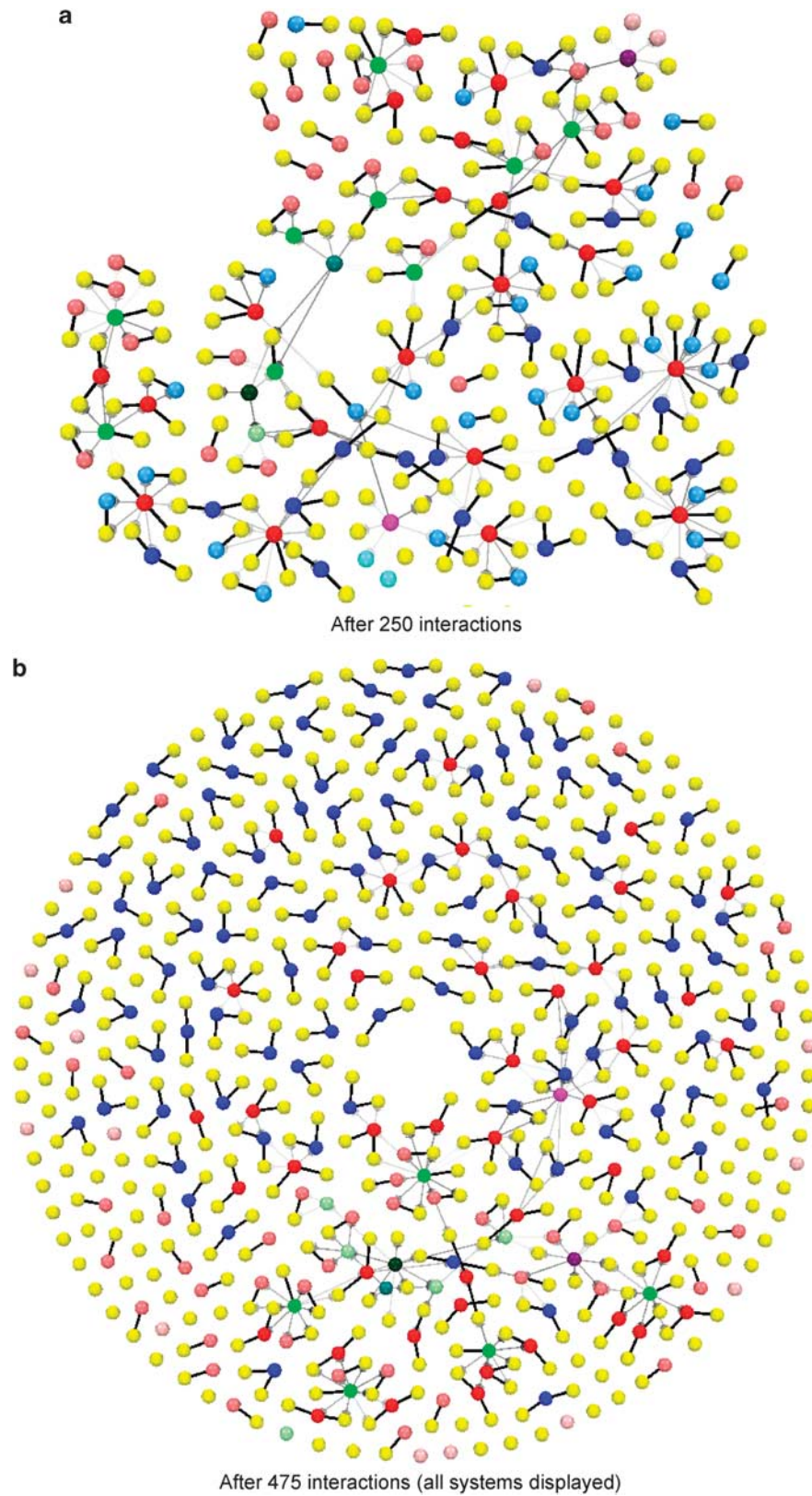


Figure 22: Visualisation over time of the MAPK model at an advanced and at a late stage (after those from Figures 20 and 21): (a) after 250 interactions and (b) after 475 interactions (all systems displayed).

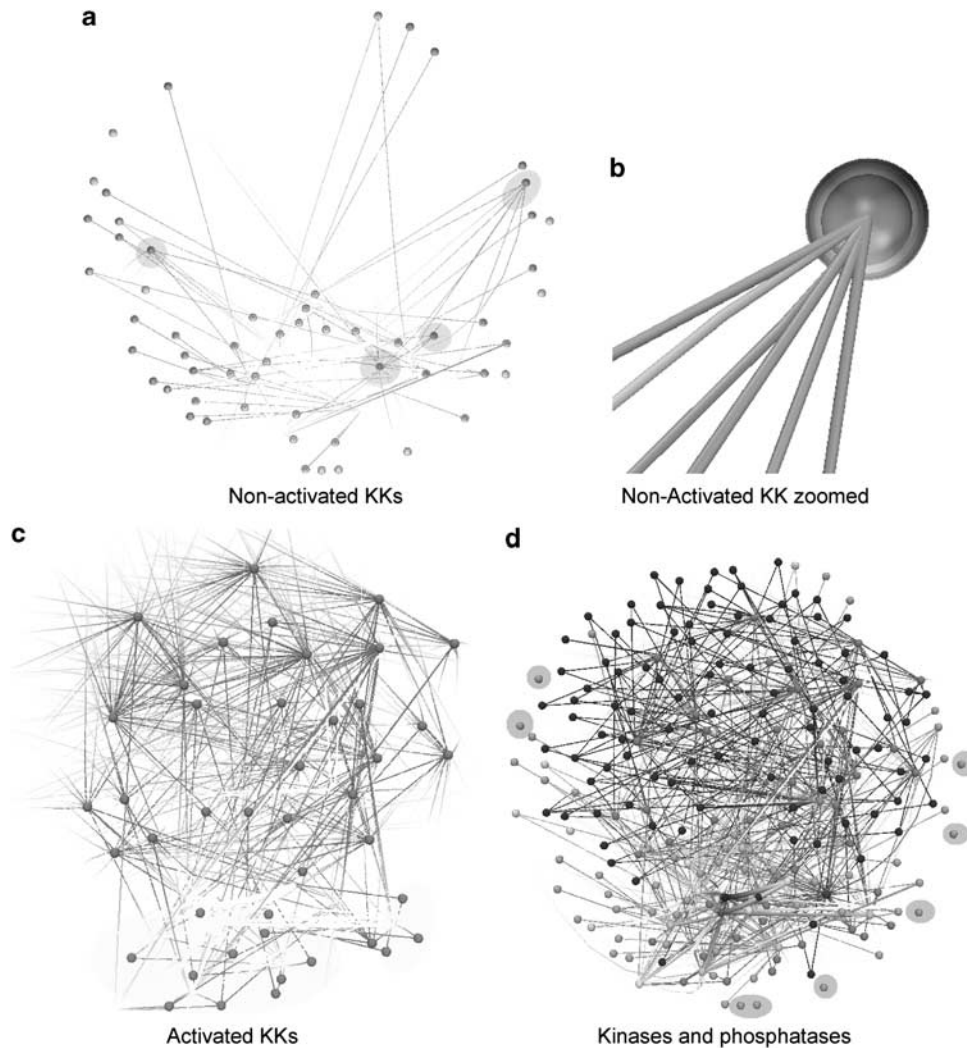


Figure 23: Involvement of kinases in phosphorylation after 475 interactions. The state is the same as in Figure 22(b), using here different views: (a) shows the non-activated *KKs* and their envelope, revealing that four only (highlighted in darkened areas, in red in the coloured version) were effectively involved in phosphorylation of *Ks*; (b) provides a zoom onto the top-right *KK* in darkened (or red) area on (a): the third branch of the envelope starting from the top (in violet in the coloured version) reveals an interaction (phosphorylation) with a *K* system (blue in the coloured version); (c) shows the activated *KKs* and their envelope. Similarly, activated *KKs* not involved in phosphorylation do not have an envelope branch going towards a *K* system (in blue in the coloured version), they are here highlighted in a yellow area. Finally, (d) shows all kinases at once. The most involved tend to be located towards the centre whereas the less involved *KK* are moved to the sides. Grey areas highlight the systems, all being *KKs*, that never interacted.

universe (in red in the coloured version) can represent the conference hall, the other systems represent people coming from various universities or companies (the colour can be the field of research).

In Figure 24(a) people randomly entered the room and in Figure 24(b) people are now chatting. The interactions can be people exchanging words in the context of another person momentarily leading a debate (as illustrated by yellow, violet and blue systems in the coloured version), or it can also be an interaction between people and the

room (assuming a presentation is going on in a corner of the room) in the context of a speaker. The three single systems (in green in the coloured version) represent bored people, having no interest in anything going on.

Figure 25 shows views of this example using the structure visualisation. Figure 25(a) and (b), respectively, show the structure of the systems after many interactions without and with time-based occlusion. The disappearance of the three single systems (green in the coloured version) on the edges systems (bored people) in

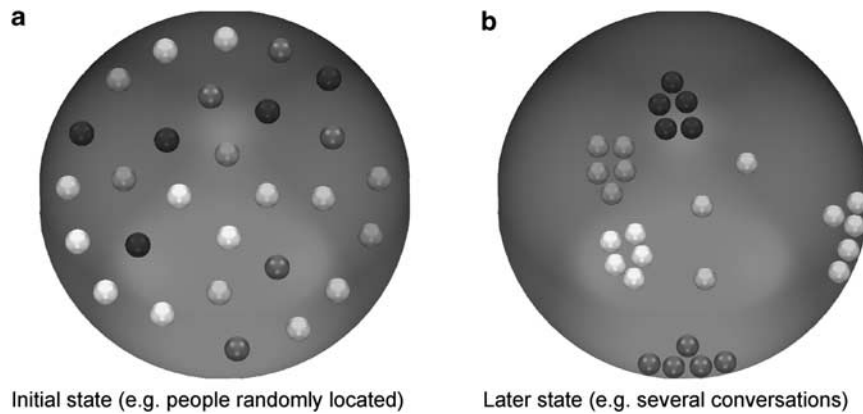


Figure 24: Example of systems grouping within a given scope using the 3D explorer. In (a) the systems (for example people, colour could represent their field of research) are randomly placed in an initial state. (b) shows the same systems after some interactions: the three groups of systems in the centre left interact only with systems of their own colours; two other groups of systems on the edge interact with systems of the same colour but also with the parent system the three remaining systems (in green in the coloured version) systems are not interacting at all.

Figure 25(a) shows that they did not interact recently, and Figure 25(d) shows, using the envelope, that they did not interact at all. Figure 25(d) also shows that the remaining people (in yellow, violet and sea blue in the coloured version) respectively interact with all people of their field whereas the two other groups (light blue and brown in the coloured version) of people also interact with the conference hall, suggesting that people are all debating a lot within their group.

Recursive hierarchies

Another interesting case can be related to the modelling of self-replication or reproduction. Considering a cell or an organism that has the potential to self-reproduce and assuming the modelling of the cycle of reproduction over time, the model should loop onto itself. When zooming in a model using the 3D explorer the user should therefore be able to explore even a recursive hierarchy. As the explorer shows a scope-wise representation of the hierarchy, it actually unrolls the recursive hierarchy to allow an in-depth exploration showing clearly the content of each scope. Figure 26 shows the 2D graph of an example involving a recursive hierarchy: A contains B which contains D which contains A . We can consider A as a fractal set containing subset B , itself containing subset D containing the set A itself.

Figure 27 shows the exploration of such model using the 3D explorer. Looking at Figure 27(d) it can be observed that D (the largest system, in violet in the coloured version) does indeed contain A and its sub-hierarchy, the exact same A sub-hierarchy that is visualised in Figure 27(b). Note that such exploration could in theory go on forever (in fact, only up to computer floating point precision).

Analysis

The two case studies showed how SC models can be analysed by exploiting the online and the overall computational past's visual information instead of reading and deciphering it from text. They illustrated how behaviour can be tracked and analysed by *looking* at models undergoing computation. Using SC for modelling and visualising offers several benefits, discussed below.

Improved naturalness

SC has been designed to imitate natural computation and enable computational models that reflect as much as possible their original concept. Part of this is achieved by using scopes for relative location and influences between systems, and interaction and transformations for the changes in systems' state. By visualising these concepts the visual output's interpretation benefits from intuitive cues such as systems' physical location to express the relations between them, and links between systems to represent hierarchy and interactions. The visualisation of an SC model is thus a concrete output representing an abstraction of the original concept that reveals its underlying computational processes while keeping a similar organisation.

The MAPK cascade model illustrated a network in which all entities are represented. Visualisations of these interacting entities provided an intuitive representation that shows kinases physically organised and linked depending on their phosphorylation state and located mostly near the other kinases they interacted with. The bistable gene network focussed on genes activity and thus modelled proteins of each type as one system (there was no need for this study to populate the model with many protein

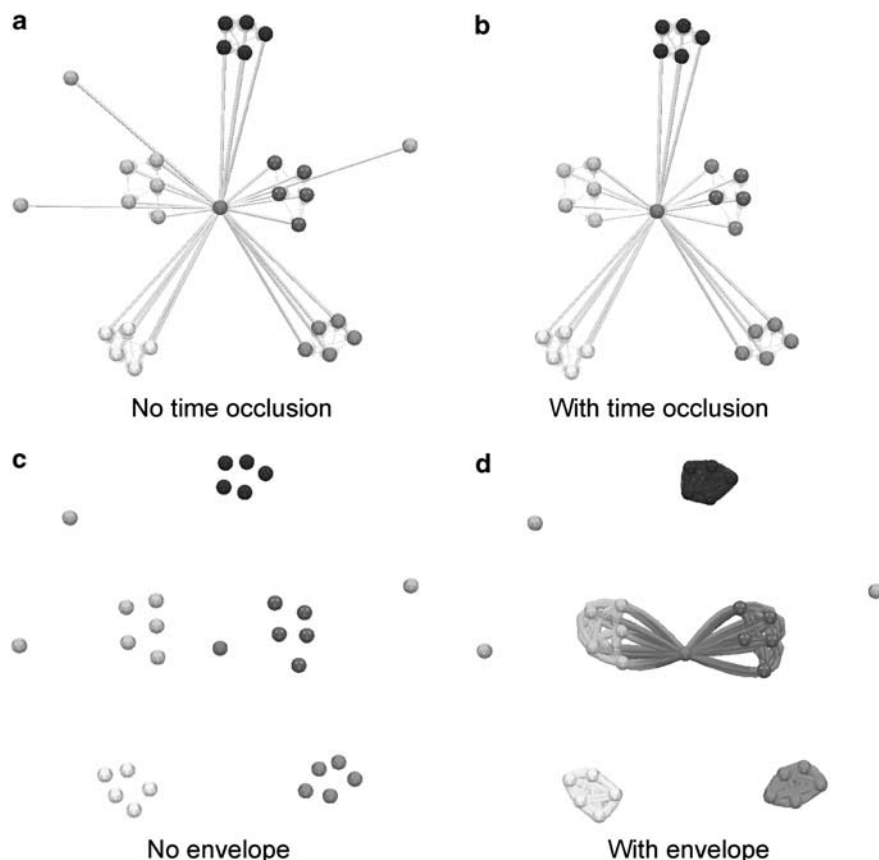


Figure 25: Example of systems presented in Figure 24 captured during execution: (a) shows the structure of the systems without usage occlusion and (b) shows the same state with occlusion; (c) shows the structure of the systems without hierarchy links and without the envelope and (d) shows the same state with the envelope.

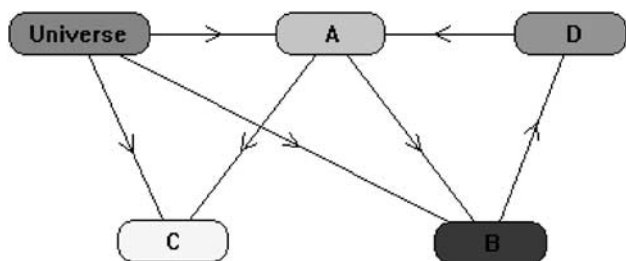


Figure 26: 2D graph of a model with a recursive hierarchy: A contains B which contains D which contains A.

systems). The visualisations thus represented the intensity of the genes’ activity where activity and state changes can be observed using intuitive clues such as proximity and links between interacting systems, and physical bounds for systems hierarchy.

In contrast, approaches such as π -calculus maintain separate processes in which one compound (for example a gene bound to a protein) is actually being modelled. In addition, the use of channels does not reflect a feature

of the model but a leftover from concurrent processes communication modelling. This is illustrated in Figure 28 providing the graphical representation for the stochastic π -calculus model of the bistable gene network from Phillips *et al.*²⁷

Conciseness, compactness and readability

As discussed in above and in previous work,²⁸ SC models provide advantages of clarity compared to other modelling approaches such as π -calculus. The binding of systems, whether genes and proteins in the gene network or phosphate groups and kinases in the MAPK cascade, is modelled in SC within the structure of the models as structural changes occurring over time and resulting from interactions.

Owing to its unnatural rules not suitable for biological processes, the stochastic π -calculus model requires two separate diagrams (see Figure 28) both representing a part of the model, leading to an unclear representation. Representing changes of structure in such model is therefore equally unclear.

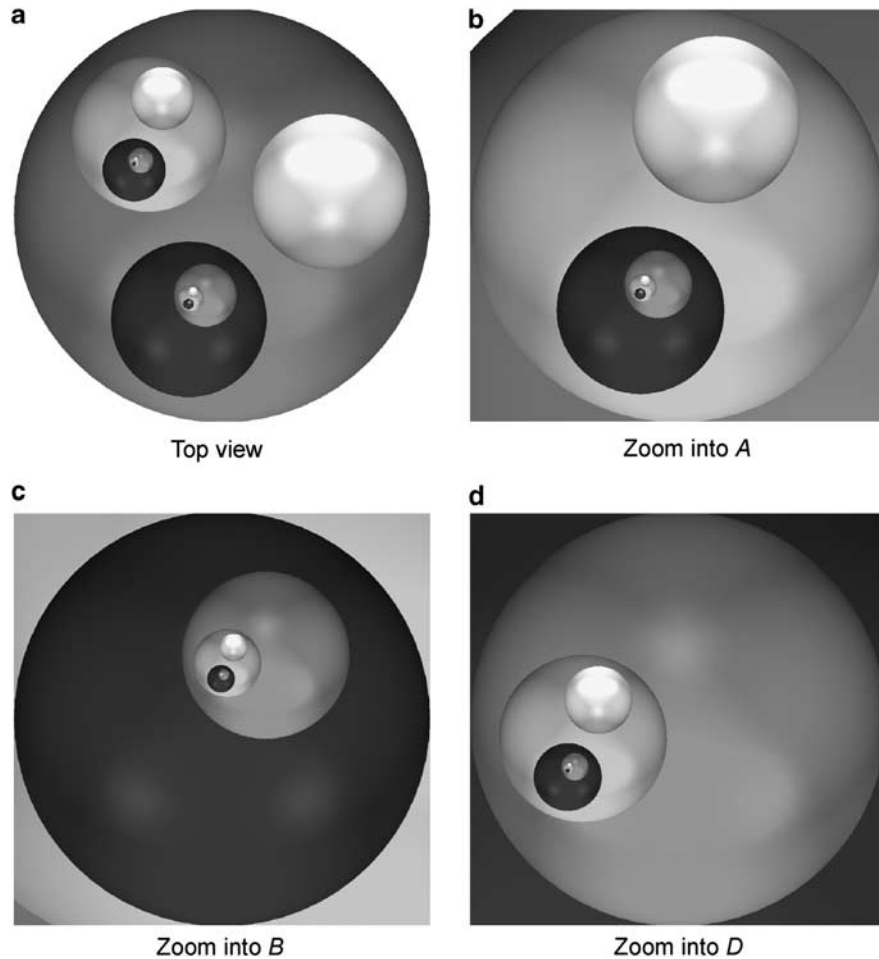


Figure 27: Example of exploration in depth starting from (a) the top view (universe), then (b) zooms from the universe into A (green), (c) zooms further into B (blue) and (d) zooms even further into D (violet).

By using the notions of scopes, interacting systems and transformations SC avoids these issues (for example, two interacting systems binding to each other are transformed so that they become bound by their scopes) and enables a better consistency between the model and its original concept. Visualising these features thus enables clearer and more intuitive representations of the underlying processes.

Natural properties exploitation ability

Rather than considering models as computer programs, SC considers them as processes in which the physical implementation is irrelevant and only interactions and structure matter. By considering these processes from such angle, emphasis is put on their behaviour. The visualisation of these processes hence becomes a graphical mapping of their behaviour, and therefore of their displayed properties. Specific systems can be shown or

hidden to highlight the activity of a subset of systems. The past states of systems can be visualised as a tree ring view. The overall interaction activity can be summarised by the abstract envelope. Computation can be broken down into steps to visualise progressively the changes occurring in the model, as was performed within the two case studies. Visualising SC is therefore another way in which the natural properties exploited in SC can be analysed.

Local, stochastic and distributed computation is clearly visualised with the ongoing interactions attracting interacting systems towards their context of interaction while the emerging behaviour can be observed within the progressive changes in structure, the respective systems location, the links between them and the envelope. The bistable gene network illustrated a network with several possible states depending upon stochastic interactions. These interactions could be visualised over time and summarised using the envelope, providing a different pattern for each potential final state. This pattern therefore reveals the distribution of interactions. Systems

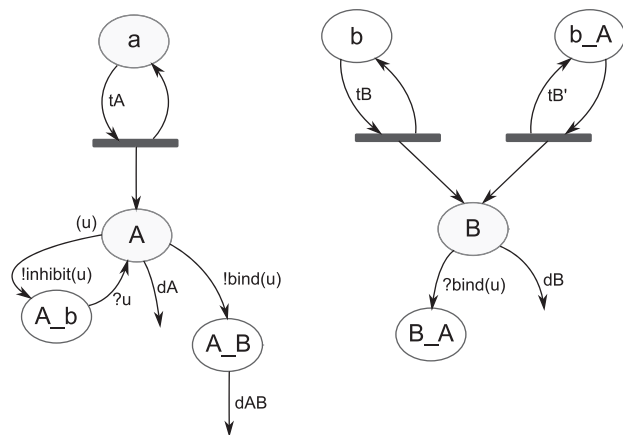


Figure 28: Bistable gene network implementation from Phillips *et al*²⁷ using the graphical representation for the stochastic π -calculus. Gene a transcribes a protein A by doing a stochastic delay at rate tA and then executing a new process A in parallel with the gene. Protein A can either degrade by doing a stochastic delay at rate dA , or bind to gene b by doing an output on channel $inhibit$, or bind irreversibly to protein B by doing an output on channel $bind$. When protein A binds to gene b it sends a private channel u and then executes the process A_b , which can unbind from the gene by doing an input on u . When protein A binds irreversibly to protein B it executes the process A_B , which can degrade by doing a stochastic delay at rate dAB . Thus, protein A is neutralised by protein B . Conversely, gene b can either transcribe a protein B by doing a stochastic delay at rate tB , or bind to protein A by doing an input on channel $inhibit$. When gene b binds to protein A it receives a private channel u and then executes the process b_A , which can either unbind from the protein by doing an output on u , or transcribe a protein B at a much slower rate tB' . Thus, gene b is inhibited by protein A . Protein B can either degrade by doing a stochastic delay at rate dB , or bind irreversibly to protein A by doing an input on channel $bind$.

occlusion, the tree ring view and the envelope were used to visualise how interactions happen in the MAPK cascade, displaying how kinases get phosphorylated and involved over time in the phosphorylation of other kinases.

Conclusion

This article presented a set of novel visualisation tools combined with systemic computation for the understanding of natural and complex systems as an on-line visualisation of models using the systemic computation paradigm. The framework involved a 2D graph, a 3D explorer and a 3D informational structure with a graphical online trace of the flow which provides a graphical output depending on models' behaviour. This trace reveals the quantity of interactions that occurred over a window of time or a whole run. The changes systems underwent are also recorded and presented as a tree ring view.

Two concrete bio-inspired models, a bistable gene network and an MAPK signalling cascade, respectively presented in François and Hakim³⁹ and in Huang and Ferrell⁴⁰ and both used in Phillips *et al*,²⁷ were studied. The bistable gene network, starting from a same initial state, showed after transformations a different shape depending on its final state. The MAPK cascade visualisation provided over time all the interactions information required to understand the inner mechanism for the evolution of activated kinase quantities, increasingly amplified as the cascade is traversed.

The visualisation provides a novel method that reveals the information flow within a dynamic system, allowing analysis of interactions, transformations and structure over time. It provides the control along the information flow to stop and look into a state, varying display options and camera views to analyse the model. The user can go back in time and try an alternative step to see what would happen next if another interaction was to happen. The bistable switch network showed that the final state it ends up in is because of a stochastic combination of interactions that can lead to one or another final state. Only an online analysis of these interactions can explain how and when the network switched to a particular behaviour that would lead to its final state. Similarly, only such visualisation of the MAPK cascade enabled the comprehension of the emerging organisation and the distribution of roles within kinases.

The graphical output provides the potential for in-depth analysis of more complex models in the future that may have non-intuitive behaviours. Different behaviours could be classified, grouped or identified based on the shape produced by the visualisation. Should two radically different models show the same or similar emergent shapes then both share the same kinds of interactions and information flow. It is anticipated that these kinds of analyses and comparisons could become essential tools for understanding commonalities in biological, natural and complex systems.

Availability: The systemic computation environment software which includes the work presented in this article is freely available from <http://code.google.com/p/sc-scope> and from <http://www.cs.ucl.ac.uk/staff/E.LeMartelot> with its documentation. Contact e.le_martelot@ucl.ac.uk for any information.

References

- Hochheiser, H., Baehrecke, E.H., Mount S.M. and Shneiderman, B. (2003) Dynamic querying for pattern identification in microarray and genomic data. In: Proceedings of the 2003 International Conference on Multimedia and Expo (ICME '03), Vol. 3. Washington DC: IEEE Computer Society, pp. 453–456.
- Holland, J.H. (1998) *Emergence: From Chaos to Order*. Oxford, UK: Oxford University Press.
- Kauffman, S.A. (1993) *The Origins of Order: Self-organization and Selection in Evolution*. USA: Oxford University Press, May.

- 4 Goldsby, R.A., Kindt, T.J., Kubly, J. and Osborne, B.A. (2002) *Immunology*. 5th edn., New York, NY: W.H. Freeman.
- 5 Meinhardt, H. (1995) *The Algorithmic Beauty of Sea Shells*. New York, NY: Springer-Verlag New York.
- 6 Wolfram, S. (2002) *A New Kind of Science*. USA: Wolfram Media.
- 7 Mitchell, M. (1996) *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.
- 8 Bentley, P.J. (2007) Systemic computation: A model of interacting systems with natural characteristics. *International Journal of Parallel, Emergent and Distributed Systems* 22(2): 103–121.
- 9 Eisen, M.B., Spellman, P.T., Brown, P.O. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America* 95(25): 14863–14868.
- 10 Hong, J., Jeong, D.H., Shaw, C.D., Ribarsky, W., Borodovsky, M. and Song, C. (2005) Gvis: A scalable visualization framework for genomic data. In: K. Brodlie, D. Duke and K. Joy (eds) EUROVIS 2005: Eurographics/IEEE VGTC Symposium on Visualization 2005, Leeds, UK: Eurographics/IEEE-CS, pp. 191–198.
- 11 Karp, P.D., Paley, S. and Romero, P. (2002) The pathway tools software. *Bioinformatics* 18(1): S1–S8.
- 12 Kanehisa, M. and Goto, S. (2000) Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research* 28(1): 27–30.
- 13 Gasteiger, E., Gattiker, A., Hoogland, C., Ivanyi, I., Appel, R.D. and Bairoch, A. (2003) ExPASy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Research* 31(13): 3784–3788.
- 14 Shannon, P. *et al.* (2003) Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research* 13(11): 2498–2504.
- 15 Van Ham, F., van de Wetering, H. and van Wijk, J.J. (2002) Interactive visualization of state transition systems. *IEEE Transactions on Visualization and Computer Graphics* 8(4): 319–329.
- 16 Pretorius, A.J. and van Wijk, J.J. (2005) Multidimensional visualization of transition systems. In: IV '05: Proceedings of the Ninth International Conference on Information Visualisation. Washington DC: IEEE Computer Society, pp. 323–328.
- 17 Pretorius, A.J. and van Wijk, J.J. (2006) Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics* 12: 685–692.
- 18 Gröller, M.E. (1998) Application of visualization techniques to complex and chaotic dynamical systems. In: Proceedings of the 5th Eurographics Workshop on Visualization in Scientific Computing, Eurographics Workshop in Scientific Computing; May.
- 19 Löffelmann, H., Doleisch, H. and Gröller, M.E. (1998) Visualizing Dynamical Systems Near Critical Points. Institute of Computer Graphics and Algorithms, Vienna University of Technology. Technical Report TR-186-2-98-09.
- 20 Viste, M. and Skartveit, H.-L. (2004) Visualization of complex systems – The two-shower model. *PsychNology Journal* 2(2): 229–241.
- 21 Hendley, R.J. and Drew, N.S. (1995) Visualisation of Complex Systems. Technical Report.
- 22 Wood, A., Beale, R., Drew, N.S. and Hendley, R.J. (1995) Hyperspace: A world-wide web visualiser and its implications for collaborative browsing and software agents. In: Poster Proceedings of the Third International World-Wide Web Conference; April; USA: Elsevier, pp. 21–25.
- 23 Benford, S. *et al.* (1999) Three dimensional visualization of the world wide web. *ACM Computing Surveys* 31: 25.
- 24 Bartram, L. (1998) Enhancing visualizations with motion. In: Proceedings of IEEE Information Visualization, USA: IEEE Computer Society Press, pp. 13–16.
- 25 Robertson, G.G., Mackinlay, J.D. and Card, S.K. (1991) Cone trees: animated 3d visualizations of hierarchical information. In: S.P. Robertson, G.M. Olson and J.S. Olson (eds.) CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; New York, NY: ACM, pp. 189–194.
- 26 Bosch, R., Stolte, C., Tang, D., Gerth, J., Rosenblum, M. and Hanrahan, P. (2000) Rivet: A flexible environment for computer systems visualization. *Computer Graphics* 34: 68–73.
- 27 Phillips, A., Cardelli, L. and Castagna, G. (2006) A graphical representation for biological processes in the stochastic pi-calculus. *Transactions on Computational Systems Biology VII*, Vol. 4230, Springer Series: LNCS, <http://www.springerlink.com/content/n218mu277hn50h3w>, pp. 123–152.
- 28 Le Martelot, E. and Bentley, P.J. (2009) Modelling biological processes naturally using systemic computation: Genetic algorithms, neural networks, and artificial immune systems. In: R. Choing (ed.) Nature-inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science and Engineering, Chapter 9, July, Hershey, PA: IGI Global, pp. 204–241.
- 29 Le Martelot, E., Bentley, P.J. and Beau Lotto, R. (2007a) A systemic computation platform for the modelling and analysis of processes with natural characteristics. In: D. Thierens *et al.* (eds.) Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007), July, USA: ACM Press, pp. 2809–2816.
- 30 Le Martelot, E., Bentley, P.J. and Beau Lotto, R. (2007b) Exploiting natural asynchrony and local knowledge within systemic computation to enable generic neural structures. In: S. Yasuhiro, H. Masami, U. Hiroshi and A. Andrew (eds) Natural Computing. Proceedings of 2nd International Workshop on Natural Computing (IWNC 2007), Vol. 1 of Proceedings in Information and Communications Technology; December, Japan: Springer, pp. 122–133.
- 31 Le Martelot, E., Bentley, P.J. and Beau Lotto, R. (2008) *Crash-proof systemic computing: A demonstration of native fault-tolerance and self-maintenance*. In: S. Sahni (ed) Proceedings of the Fourth IASTED International Conference on Advances in Computer Science and Technology (ACST 2008), April, USA, Canada and Switzerland: ACTA press, pp. 49–55.
- 32 Le Martelot, E. and Bentley, P.J. (2009) Metabolic systemic computing: Exploiting innate immunity within an artificial organism for on-line self-organisation and anomaly detection. *The Journal of Mathematical Modelling and Algorithms: Special Issue on Artificial Immune Systems* 8(2): 203–225.
- 33 Le Martelot, E. and Bentley, P.J. (2009) On-line systemic computation visualisation of dynamic complex systems. In: H.R. Arabnia and L. Deligiannidis (eds) Proceedings of the 2009 International Conference on Modeling, Simulation and Visualization Methods (MSV'09), July, Las Vegas, NV, CSREA Press, pp. 10–16.
- 34 Wheeler, M. and Clark, A. (1999) Genic representation: Reconciling content and causal complexity. *The British Journal for the Philosophy of Science* 50(1): 103–135.
- 35 Le Martelot, E. (2010) Investigating and analysing natural properties enabled by systemic computation within nature-inspired computer models. PhD thesis, University College London.
- 36 Fruchterman, T.M.J. and Reingold, E.M. (1991) Graph drawing by force-directed placement. *Software Practice & Experience* 21(11): 1129–1164.
- 37 Griffiths, D.F. (1999) Introduction to Electrodynamics, 3rd edn., B. Cummings (ed.) USA: Prentice Hall, p. 576.
- 38 Borelli, A.P. and Schmidt, R.J. (2002) *Advanced Mechanics of Materials*. 6th edn., USA: Wiley.
- 39 François, P. and Hakim, V. (2004) Design of genetic networks with specified functions by evolution in silico. *Proceedings of the National Academy of Sciences of the United States of America* 101(2): 580–585.
- 40 Huang, C.-Y.F. and Ferrell, J.E. (1996) Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proceedings of the National Academy of Sciences of the United States of America* 93(19): 10078–10083.
- 41 Pearson, G. *et al.* (2001) Mitogen-activated protein (map) kinase pathways: Regulation and physiological functions. *Endocrine Reviews* 22(2): 153–183.