Progress in Biophysics and Molecular Biology xxx (2012) 1–8

Contents lists available at SciVerse ScienceDirect

Progress in Biophysics and Molecular Biology

journal homepage: www.elsevier.com/locate/pbiomolbio

Original research

Modelling, abstraction, and computation in systems biology: A view from computer science

Tom Melham*

Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Algorithmic biological modelling
Abstraction
Multi-scale modelling
Biological computation

ABSTRACT

Systems biology is centrally engaged with computational modelling across multiple scales and at many levels of abstraction. Formal modelling, precise and formalised abstraction relationships, and computation also lie at the heart of computer science—and over the past decade a growing number of computer scientists have been bringing their discipline's core intellectual and computational tools to bear on biology in fascinating new ways. This paper explores some of the apparent points of contact between the two fields, in the context of a multi-disciplinary discussion on conceptual foundations of systems biology.

1. Introduction

Systems biology can be characterised as an approach to the understanding of life through the study of how the properties of biological systems arise through interactions between components, as these are situated and organized within the system. What is regarded as the 'system' is, inevitably, bounded and is itself situated within some environment, with which the system (through its components) may have relevant interactions. It is an *open system*, whose boundary is to some extent drawn at our discretion. Of special interest is the characteristic property of the system being self-sustaining and, at the large scale, self-replicating. The aim of science is insight, and the kind of insight that systems biology seems to afford is explanation of how properties of the system arise by development of testable theories about the underlying mechanism for some system-wide phenomenon. Computer modelling and computational exploration of models are centrally involved, because of the complexity of the interactions and systems being studied. Experiments are central too, generating understanding through an intimate interplay with inductive theory formation embodied in models (Westerhoff and Kell, 2007) and, not least, driving insight forward by falsifying hypotheses (Kohl et al., 2010).

What does computer science offer this enterprise? In one sense the answer is obvious: systems biology in practice runs computer simulations of biological systems, often by computing numerical solutions to differential equations, and correlates the results of these 'insilico' experiments with laboratory data. Then there is also bioinformatics—the use of computer algorithms to process, manage, and analyse biological data, usually in large quantities. But I want to consider here the relevance of some of the deeper ideas of computer science—the intellectual framework and associated formal tools characteristic of so-called *computational thinking* (Wing, 2008).

The key idea is that computer science is centrally engaged with the design, analysis and justification of *abstractions*. Abstraction is the act of isolating for separate consideration the important aspects or properties of a complex object, and ignoring the remaining ones as being irrelevant to the task in hand. It is a fundamental strategy for taming complexity. Computer scientists work with especially rich worlds of abstractions, expressed in symbolic form as programs, algebras, logics—even diagrams—and supporting complex operations and constructions. The act of abstraction (or its reverse, refinement) structures our conceptualizations into layers, or 'levels'. Computer scientists are very used to working simultaneously at multiple different levels of abstraction and—crucially—to thinking about the relationship between pairs of abstraction layers.

Computing itself, says Wing (2008), is 'the automation of our abstractions'—the mechanised animation, analysis, or interpretation of abstractions represented in precise symbolic form. In practice, there is interplay between these two things. The abstraction you can achieve will be influenced by what can be mechanised efficiently, or which of its properties can be deduced algorithmically; the abstraction you want to design will drive the algorithmic innovations needed to automate it.

Denis Noble's first, and fundamental, principle of systems biology is that biological functionality is *multi-level*. From this flow the other nine major principles explored in (Noble, 2008) and

* Tel.: +44 1865 273824.
E-mail address: Tom.Melham@cs.ox.ac.uk.

0079-6107/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.pbiomolbio.2012.08.015

derived from his view of systems biology in *The Music of Life* (Noble, 2006). Accepting, for the time being, that what a physiologist means by 'level' is related to a computer scientist's 'layer of abstraction', it seems that computational thinking should have something fundamental to contribute. Certainly a growing number of computer scientists are bringing their discipline's core intellectual and computational tools to bear on biology in fascinating new ways (Cardelli, 2005; Fisher and Henzinger, 2007; Priami, 2007, 2009; Fisher et al., 2011).

In this paper I explore some of these connections from a conceptual point of view. I make no claim to novelty for these observations: much of what I say will be familiar to any educated computer scientist, not least those whose technical work in biological modelling is referenced in the reviews just cited. My aim here is to frame some of these ideas in the context of the multidisciplinary discussion represented by this focussed issue on *Conceptual Foundations of Systems Biology*.

2. Models

Not least among the kinds of abstractions computer scientists deal with are *abstract models*. Models of what? Fundamental to the subject, right from the beginning, is the idea of an abstract model of 'computation'. Turing machines are the archetypal example (Turing, 1937). A primary characteristic of such models is that they have a mechanistic, 'executable', flavour. We can *run* them. Usually, but not always, they are embodied as a mathematical theory of a rather formal character—and we can analyse their properties by deductive proof. Equally central to modern computer science are abstract models of actual computer hardware, for example as a deterministic finite-state automaton (Hopcroft et al., 2006), and models of many different kinds of software systems.

Automata are a classic example of a *state-based* abstract model—a formal representation of a system that is given by positing a set of 'states' that the system can be in, together with some rules that determine the succession of states through which the system moves over time. I will call such a succession a *trajectory* of the system through the state space. Usually, the state space as a whole is laid down in such a way that it will contain many individual states that do not observably correlate with any possible state of affairs in the system being modelled. We therefore designate a set of 'initial states', which the system may be assumed to be in at the beginning of the period of time over which its behaviour is being analysed or computed. The *reachable states* are those which can (in principle) occur along some trajectory beginning in the set of initial states; these are usually assumed to correlate with states of affairs that may obtain in the system being modelled.

For open systems, there will additionally be some representation of a mechanism of interaction with the surrounding environment; in a classic finite-state machine, for example, this is given by 'inputs', originating from outside the system and co-determining the trajectories followed within the system. Normally the environment cannot be assumed to engage freely in any and all interactions, but is constrained by its own make-up to behave in only certain ways. So an 'environment model' must be added, frequently articulated as a relational constraint on the interactions with the system the environment may engage in. This is almost invariably a *partial* model of the environment, characterizing just enough of its behaviour to relevantly constrain the trajectories that may arise within the model.

All this is commonplace in computer science—and indeed Science itself. State-based models, as causal theories of physical reality, go back at least to Newtonian mechanics. Computer science, however, is distinguished by its focus on *executable* models. A computer scientist will typically expect the rules determining trajectories in a model to provide the basis for an effective procedure to compute actual sequences of states. Fisher and Henzinger (2007) call the application of this algorithmic approach to biology 'executable cell biology' and contrast this to the classical approach based on differential equations. Priami (2009) contrasts the step-by-step operational descriptions of 'algorithmic systems biology' with modelling by equational constraints specifying system dynamics.

Many research efforts in these directions have focussed on the adaptation for biology of well-established frameworks for modelling computational systems—notably process calculi (Ciocchetta and Hillston, 2008; Calder and Hillston, 2009), rewriting and systems of rewrite rules (Danos et al., 2007; Talcott and Dill, 2006; Talcott, 2008), Petri nets (Heiner et al., 2003, 2008), and interacting state machines (Fisher et al., 2011). New formalisms in these broad families but specifically designed for biological modelling have also been proposed, such as Brane ('membrane') calculi (Cardelli, 2004) and, for higher level descriptions, Biocharts (Kugler et al., 2010). Published studies with these formalisms are commonly (but by no means exclusively) aimed at modelling of signalling pathways or other molecular networks—a mainstream abstraction of reductionist molecular biology.

A second characteristic of computer science approaches to modelling is a central focus on precisely defined, formal *languages* for defining models. Models expressed in these languages are commonly highly structured—constructed by composing together descriptions of parts using a variety of syntactic operators. The model's syntactic structure may correspond to a posited dissection of the system under study into physical parts, or (perhaps less commonly) functional components. The semantics of the operators defines the ways in which the modelled behaviours given by the semantics of the components may interact with each other to produce overall behaviour. Roughly speaking, such a model is said to be *compositional* if a satisfactory semantics of system-wide behaviour can be achieved in this way—if the expected system-wide behaviour 'emerges'. The concept is well-known in the philosophy of language and logic (Hodges, 2001) and pervasive in computer science.

Working within precisely-defined modelling languages, with rigorously-defined mathematical semantics, has both pragmatic and theoretical merits. A carefully designed language can make model descriptions more perspicuous and intuitive. The language may even be diagrammatic and still possess rigorous mathematical semantics—a *visual formalism* (Harel, 1988). Such languages can be richly supported by software tools for model execution and exploration (Efroni et al., 2005). An example is Biocharts, a fully executable language and tool based in part on well-proven, state-based software systems engineering formalisms (Kugler et al., 2010).

More significant, from a conceptual point of view, is the theoretical advantage that a well-defined modelling formalism circumscribes a space of all possible models, so that metatheorems about the modelling framework as a whole can be expressed, and even proved. Theoretical computer science abounds with such theorems—for example about the decidability of properties, closure under interesting constructions, minimality, expressiveness, computational complexity, and many others. A formal syntax also provides a 'syntactic purchase' on models, for example to prove properties by induction over the ways in which models are constructed, or syntactically to delineate interesting classes of models with special properties. Finally, having a *formal* language of models means we can entertain alternative interpretations of models, perhaps varying the level of abstraction. It remains, largely, to be seen what can be achieved with these ideas in the biological sphere.

A third—and perhaps distinctive—characteristic of formal, state-based models is their amenability to automated derivation of