Rosen's (M,R) System in Unified Modelling Language

Ling Zhang[1], Richard A. Williams[2] and Derek Gatherer[1]*

[1]Division of Biomedical & Life Sciences, Faculty of Health & Medicine, Lancaster University, Lancaster LA1 4YW, UK

[2]Department of Management Science, Management School, Lancaster University, Lancaster LA1 4YW, UK

*Corresponding author

Email: d.gatherer@lancaster.ac.uk

Abstract

Robert Rosen's (M,R) system is an abstract biological network architecture that is allegedly non-computable on a Turing machine. If (M,R) is truly non-computable, there are serious implications for the modelling of large biological networks in computer software. A body of work has now accumulated addressing Rosen's claim concerning (M,R) by attempting to instantiate it in various software systems. However, a conclusive refutation has remained elusive, principally since none of the attempts to date have unambiguously avoided the critique that they have altered the properties of (M,R) in the coding process, producing merely approximate simulations of (M,R) rather than true computational models. In this paper, we use the Unified Modelling Language (UML), a diagrammatic notation standard, to express (M,R) as a system of objects having attributes, functions and relations. We believe that this instantiates (M,R) in such a way than none of the original properties of the system are corrupted in the process. Crucially, we demonstrate that (M,R) as classically represented in the relational biology literature is implicitly a UML communication diagram. Furthermore, since UML is formally compatible with object-oriented computing languages, instantiation of (M,R) in UML strongly implies its computability in object-oriented coding languages.

1. Introduction

Relational biology is a school of thought within mathematical theoretical biology that claims that living systems can be expressed in valid models that are

nevertheless non-computable, thus placing a limitation on the analytical and predictive potential of mainstream systems biology. First devised by Robert Rosen (Rosen, 1958a, b, 1959, 1963, 1972, 1991, 2000) and subsequently developed by various others (Baianu, 2006; Casti, 1988 ; Cottam et al., 2007; Kineman, 2007; Kineman, 2011; Louie, 2005, 2007a, b; Louie, 2009, 2011; Louie, 2015; Louie and Kercel, 2007; Witten, 2007; Wolkenhauer and Hofmeyr, 2007), relational biology has been extensively reviewed as posthumous interest in Rosen's work has grown among systems biologists (Cardenas et al., 2010; Cornish-Bowden and Cardenas, 2005, 2007; Cornish-Bowden et al., 2007; Letelier et al., 2011; Wolkenhauer, 2007)

One of the bases of relational biology's critique of systems biology lies in the theory of computation in Turing machines, and how that theory relates to self-referential network architectures, meaning networks in which causal chains are circular. The Turing model of computation has provided the theoretical underpinning for the design of computers for over 70 years, but it was realised very early that there are certain problems that cannot be solved by Turing machines in any finite period of time, but rather continue processing data indefinitely (Radó, 1962; Turing, 1936). One major class of algorithms of this sort involve impredicative sets, meaning sets that are members of themselves (Whitehead and Russell, 1963 [1927]).

Recent work in relational biology has focussed on one particular theoretical model: a small abstract network architecture, the Metabolism-Repair – or alternatively Metabolism-Replacement (Letelier et al., 2006) – system, conventionally abbreviated to (M,R). Aloisius Louie has used the mathematics of Category Theory to demonstrate that (M,R) contains an impredicative set, and is

therefore non-computable on a Turing machine (Louie, 2005, 2007a, b; Louie, 2009, 2011). It should be emphasised that impredicativity is not the only obstacle to computability of (M,R) (see Rosen, 1989 for a possibly even more fundamental problem), but Louie has focussed attention on it as an important testable aspect of (M,R)'s properties. Illustration of how (M,R) can be expressed in Category Theory is beyond the scope of this paper - the best concise demonstration is Louie's 2005 paper (Louie, 2005) - but a more intuitive grasp of the self-referential nature of (M,R) can be achieved simply by contemplating its topology in either the original graphical representation (Rosen, 1991) or the reworking by Goudsmit designed to make it more comprehensible to biochemists (Goudsmit, 2007) by representing it as composed of metabolic and catalytic reactions (Fig. 1). In the Goudsmit representation (Fig. 1a), productive reactions are shown using the black arrows and catalytic requirements using the red dotted arrows. In the original (M,R) diagram of Rosen (Fig. 1b), the productive reactions are presented as open-headed arrows and causal processes as fill-headed arrows, with their arrowheads on the substrate of the productive reaction.

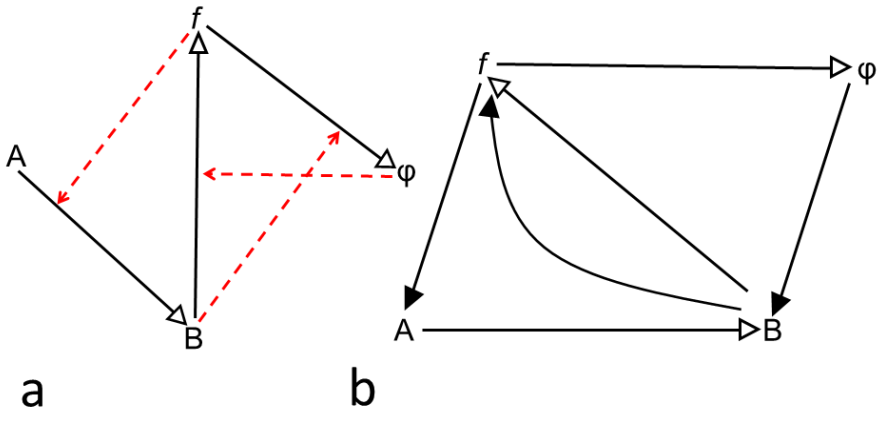


Fig. 1 a: The Goudsmit representation of the (M,R) system. b: the original (M,R) diagram of Rosen.

When (M,R) is considered in the terms proposed by Goudsmit, all of the catalytic components of the (M,R) network (*f*, φ, B) are themselves material products of the network, and all the causal relations within (M,R) – in the terminology of relational biology, its entailment structures – are internal. If one follows through a series of events within (M,R), one can see that there is an infinite loop. For instance, *f* catalyses the production of B from A, or as relational biologists say, *f* entails B. This in its turn, entails φ, which entails *f*, and so on. This is often expressed algebraically using an entailment operator, $\vdash$, as follows:

$$f \vdash B \vdash \varphi \vdash f \vdash B \ \ldots$$

Rosen intended (M,R) to be broadly representative of living systems, in that the production of B from A may be taken to represent the totality of metabolism in a cell, and the other reactions represent the totality of repair and replication components of the system. However, whether or not one chooses to see (M,R) as a generalized abstract description of a living system or rather as the basis for a specific example, as most of those who have attempted to compute it have done, the implications for systems biology are serious. If a small network instantiation of (M,R) is Turing non-computable, the existence of an (M,R)-like structure within a larger genetic or biochemical network would mean that it would also be non-computable. Correspondingly, if (M,R) is an adequate general model of a living system, artificial life is non-computable. The only way out of these problems would be to sacrifice

representational precision, creating a mere simulation of a network as opposed to a precise model. Relational biology defines a model as a computational or mathematical representation of an aspect of reality in which the entailment structures of the real world are mirrored in the entailment structures of the representation. A simulation by contrast, may have any entailment structures adequate to produce approximate behaviour corresponding to the real world. Simulations may be useful, but they rarely lead to true understanding. By virtue of being forced to substitute simulation for modelling, systems biology cannot fully capture the complex functional organisation of organisms (Rosen, 1991).

The responses to relational biology's critique of systems biology have been varied. The most direct attacks have been on the premises of (M,R) – either it is mathematically flawed or otherwise incomplete, it makes assumptions that are unjustified or it does not closely enough represent biological reality to be valuable (Chu and Ho, 2006, 2007; Goertzel, 2002; Gutierrez et al., 2011; Landauer and Bellman, 2002; Wells, 2006). These attacks have produced equally vigorous responses (Louie, 2004, 2007a; Louie, 2011), which have been summarised by Gwinn (2010). A second line of assault has been more indirect – to attempt to present (M,R) in a software format. The rationale of this second approach is to demonstrate that (M,R) is pragmatically computable, and thus to imply that there must be some error in the basic logic of relational biology, without formally identifying that error. This attritional offensive has also run into problems, principally with the need to show that the software instantiations of (M,R) do not, for software engineering

purposes, add or subtract elements from (M,R) that render them invalid as accurate models of what they purport to compute.

A summary of these previous attempts is given in Table 1. The relevance of the autopoietic system simulations on lines 1 to 3 is uncertain, as they were performed before publication of the paper of Letelier et al (2003) which posited that (M,R) is a variant of autopoietic systems. Since this has not been independently corroborated, the inclusion of autopoietic system simulations on the list must remain tentative. The remaining five lines of Table 1, however, all represent experiments carried out for the explicit purpose of testing the computability of (M,R).

Table 1. Summary of practical attempts to compute (M,R)

Type of simulation	Software system	Reference
Autopoietic	Tesselation automaton	(Varela et al., 1974)
Autopoietic	SWARM	(McMullin, 2004; McMullin and Varela, 1997)
Autopoietic	Assorted others	(Breyer et al., 1998; Ono and Ikegami, 2002; Suzuki and Ikegami, 2008; Zeleny, 1978)
Extended (M,R)	Hybrid automaton	(Cho et al., 2005)
Full (M,R)-consistent	MatLab/COPASI/MetaTool	(Piedrafita et al., 2012a;

example		Piedrafita et al., 2010; Piedrafita et al., 2012b)
Full (M,R)-consistent example	SPICE	(Prideaux, 2011)
Compact (*M,R*)	Bio-PEPA	(Gatherer and Galpin, 2013)
Verbatim (M,R)	UML	this paper

The latest published example, by Gatherer & Galpin (2013), may serve to illustrate the pitfalls that lie on this path. In that paper, we attempted to treat (M,R) as an individual network of four moieties and three catalysed reactions, from which we then derived reaction rate equations expressed in the Bio-PEPA process algebra engine (Hillston, 2005). This produced a clearly functioning system which exhibited some interesting behaviour, with output variation largely dependent on starting conditions. However, potential sources of error were pointed out by reviewers and recognised in the published paper. The first of these is the use of a stochastic mechanism for updating the (M,R) system state in Bio-PEPA. Since the original (M,R) is completely deterministic, introduction of stochasticity represents the application of an extra layer of causality to (M,R). We believe we successfully addressed the problem by also running the (M,R) system in a deterministic mode using a Runge-Kutta algorithm. However, this may also beg the question of the degree to which it is appropriate to use another algorithmic process with its own internal entailment structure (in this case one based on Runge-Kutta) to govern the processes occurring within (M,R).

The second problem is one common to all computational instantiations of (M,R) that attempt to translate the system into one resembling a small series of metabolic reactions governed by Michaelis-Menten kinetics or a similar set of rules (Prideaux, 2011), where entities *f* and φ are defined as concentrations of enzymes. This difficulty is too complex to explain in the present context, but can be found in detail in section 2 of Louie's 2011 paper (Louie, 2011).

The third problem is that the Bio-PEPA implementation of (M,R) would also, in some runs, continue beyond our patience to observe it, given regular replenishment of the input material A. Indeed, for many combinations of starting state parameters, we were unable to predict if the program would terminate, or when. We concluded that, although this might be taken to support the contention that (M,R) is not fully computable in finite time for all potential starting configurations on a Turing system, the Bio-PEPA instantiation of (M,R) was life-like, insofar as the life of any organism may be unpredictably short, long or indefinite. Therefore relational biology's insistence that incomplete computability necessarily renders artificial life uninformative about real life, is untenable. However, this merely undermines one of relational biology's corollaries, not its central argument.

Leaving aside these issues, a fourth and more serious problem was detected in the treatment of component B. In order to keep (M,R) compact, we assumed that B was capable of acting both as a metabolic substrate for production of *f* and also to catalyse the production of φ. This infringes the rules of (M,R), and indeed the

177 treatment of B has also been a problem in previous computational (Prideaux, 2011)
178 and theoretical (Landauer and Bellman, 2002; Mossio et al., 2009) approaches. This
179 issue has been elaborated on in some detail by other authors (Cardenas et al., 2010;
180 Letelier et al., 2006). A similar argument could be made for the dual role of f as
181 substrate for the production of φ and as a catalyst.
182
183 This illustrates the difficulty of encoding (M,R) without in some way corrupting
184 its structure. The use of Bio-PEPA, SPICE, MatLab, Copasi and MetaTool, and indeed
185 SWARM if autopoietic simulation can be regarded as relevant, necessarily impose
186 constraints and limitations emerging from the software tools themselves. These
187 may subtly alter the entailment structure of the computed representation of (M,R)
188 to the point where (M,R) is not being truly modelled but rather merely simulated -
189 the precise point that relational biology makes about systems biology in general.
190
191 Here, we once again attempt to computationally represent (M,R), this time
192 paying particular attention to doing so in a way that will not introduce any such
193 corruptions of (M,R)'s entailment structure. To do this, we choose the Unified
194 Modelling Language (UML) (Booch et al., 1998; Fowler, 2004), maintained by the
195 Object Management Group (2011). Although originally developed to document
196 technical requirements for the analysis and design of computer systems (Booch et
197 al., 1998), UML has recently been used to model complex biological systems (Read et
198 al., 2014; Roux-Rouquie et al., 2004; van Beijnum et al., 2010; Yan, 2010). Webb and
199 White (2005) and Bersini *et al* (2012) argue that the principles of object-oriented
200 analysis and design inherent in UML can be directly applied to the top-down

modelling of cells, and bottom-up modelling of metabolic pathways and cell signalling cycles. Crucially, UML allows computational structures to be represented entirely graphically, and therefore enables us to produce an instantiation of (M,R) which is completely transparent in its entailment structure without any hidden causal layers. We therefore produce a more verbatim encoding of (M,R) than has previously been achieved.

UML is compatible with any higher-order object-oriented (or class-based), computing language, such as Java, C++, and Objective-C. However, we do not at this stage take the obvious subsequent step of attempting to translate the UML representation into lines of code in any of these languages, which may be achieved via the intuitions of a programmer, or by using an automated UML-to-code application such as Poseidon (Gentleware AG, Hamburg). This would only introduce an added layer of potential error into the experiment, and once again raise the spectre of (M,R)'s corruption. We therefore present here only the graphical encoding of (M,R) in UML, in order first to establish beyond doubt that a genuine object-oriented realization of (M,R) is possible.

2. Methods

Object-oriented analysis was assisted by use of Class-Responsibility-Collaboration (CRC) cards (Beck and Cunningham, 1989) and table top simulation. By using real physical objects as tokens for software objects, the CRC method assists greatly in priming the programmer's intuitions concerning what objects to define

and what properties and functions they should have. As a result of the CRC process, the following types of UML diagram(Booch et al., 1998; Fowler, 2004; Object Management Group, 2011) were constructed in Visual Paradigm (2010):

a) Class Diagram – specifying the entities within the system, their features and relationships to each other
b) Activity Diagram - specifying the behaviour of the system
c) Communication Diagram – specifying how the entities within the system are connected, or how they interact.
d) State Machine Diagram – specifying how events within the system change the entities within the system

b) to d) are all examples of what are more generically termed UML behaviour diagrams, whereas a) is a UML structure diagram.

3. Results

3.1 Class Diagram

Object-oriented analysis is based on the notion that since the world is full of concrete objects that interact with each other, computer programs that attempt to address the real world should have a similar logical structure. The software world is therefore filled with software objects. Like objects in the real world, these software objects may be grouped by similarity. A software class in object-oriented analysis is an abstract term used to describe a set of software objects that share properties, in

other words, objects that are in some way the same kind of thing. Classes are deemed to have attributes which describe the properties of the objects in the class, and functions (also known as methods) which describe what the objects do. Classes can inherit attributes and functions from their parent classes. Fig. 2 shows inheritance from the class Biomolecule, which has two daughter classes, Substrate and Enzyme. The class Substrate has a single function: produceOtherBiomolecules(), indicating that this is what substrates do. Likewise the class Enzyme also has a single function: catalyseSubstrates(). From Substrate and Enzyme we then derive three more classes apiece which together represent the objects within (M,R). To take one of these as an example, class φ has the single function: catalyseRepair(B): *f*/*f'*, indicating that φ is the enzyme responsible for catalysis of the reaction which produces *f* or *f'* from B. We have avoided the error of Gatherer & Galpin (2013) by specifying *b* as a separate class to B, and also distinguishing between class *f* as substrate versus class *f'* as enzyme. This is equivalent to the conversion function on B in the previous instantiation of (M,R) in SPICE (Prideaux, 2011). It should be noted that none of our classes has any attributes. This is because the entities in (M,R) are defined entirely in terms of what they do, rather than what they look like, their size etc. This is entirely in keeping with relational biology's emphasis on abstract function. To quote Rosen: "The relation of analogy between natural systems is in fact independent of their material constitution." (Rosen, 1991, p119). It should be stressed that other object hierarchies may be possible, for instance to abolish the Substrate/Enzyme distinction and define classes *b* and *f'* as sub-classes of B and *f*, respectively. There is no single correct object-oriented instantiation of (M,R), but all

correct instantiations should allow the system to perform metabolism, repair and replication as specified by Rosen.

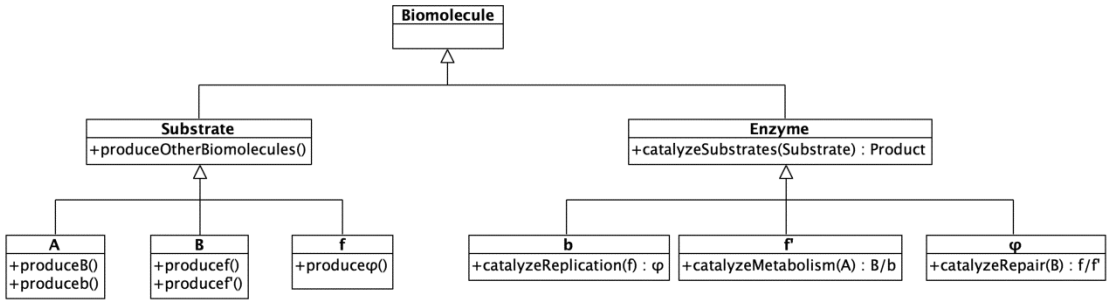
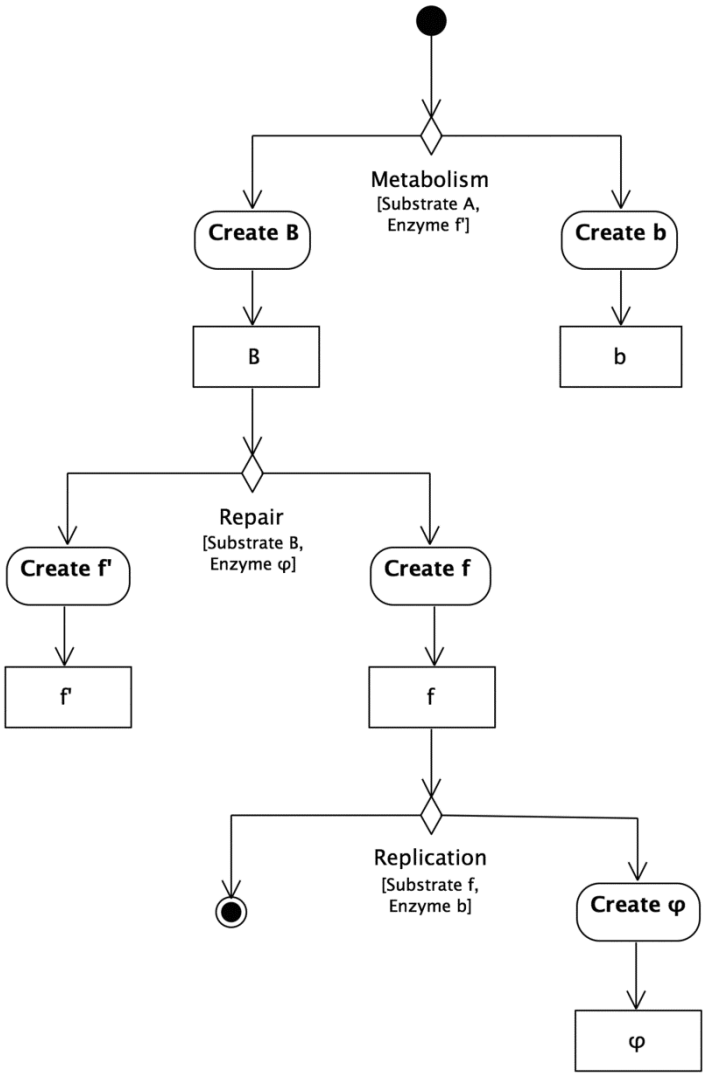


Fig. 2: A UML class diagram for (M,R) Class names are above the horizontal line, functions are below the horizontal line. Vertical arrows indicate inheritance. Class B, for instance, is a substrate and therefore inherits the functions of class Substrate, in addition to possessing its own, B-specific, functions.

3.2 Activity Diagram

The class diagram contains a great deal of implicit information. This is elaborated in more explicit form in the activity diagram (Fig. 3). The activities in this diagram often correspond to the functions listed in the class diagram. Their explicit effects, for instance "create B", are contained within lozenges and the objects resulting from these effects are contained within rectangles. The starting point of the activity diagram is an object of class A and the end-points are the non-metabolic objects of classes *b*, *f'* and φ. The activity diagram thus represents mass-flow within the (M,R) system, and illustrates the intuitively obvious fact that a continuous supply of A is required to maintain the life of the system. The activity diagram is also the part of UML that is most similar to the flowcharts of classic procedural programming

290 in languages such as Pascal and BASIC. In relational biology terminology, it is a
291 sequential composition (Louie, 2009, 2011), meaning that the circular entailments of
292 (M,R) have been unpicked and represented as a series of events with a beginning
293 and an end – there are no causal loops in the activity diagram. Crucially, relational
294 biology specifically rejects that such sequential compositions are full representations
295 of (M,R) but, conversely, admits they are computable. UML requires more than class
296 and activity diagrams to model (M,R).



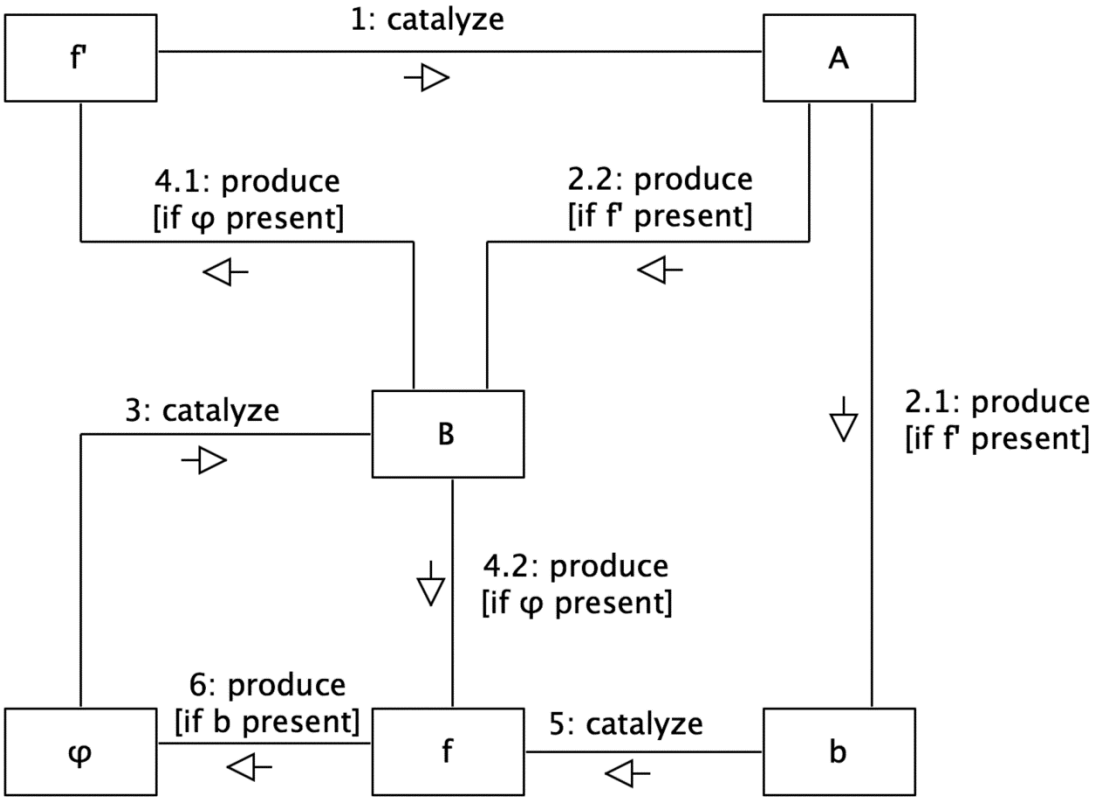
297

298 **Fig. 3: A UML activity diagram for (M,R)** An arbitrary initialization point is indicated
299 using the filled circle (●) and an arbitrary termination point using the filled circle

within another circle (⦿). Choices are shown as diamonds, with ensuing activities in lozenges. Arrows pointing out of activities show the products of that activity, and arrows pointing into activities show the requirements for the activity.

3.3 Communication diagram

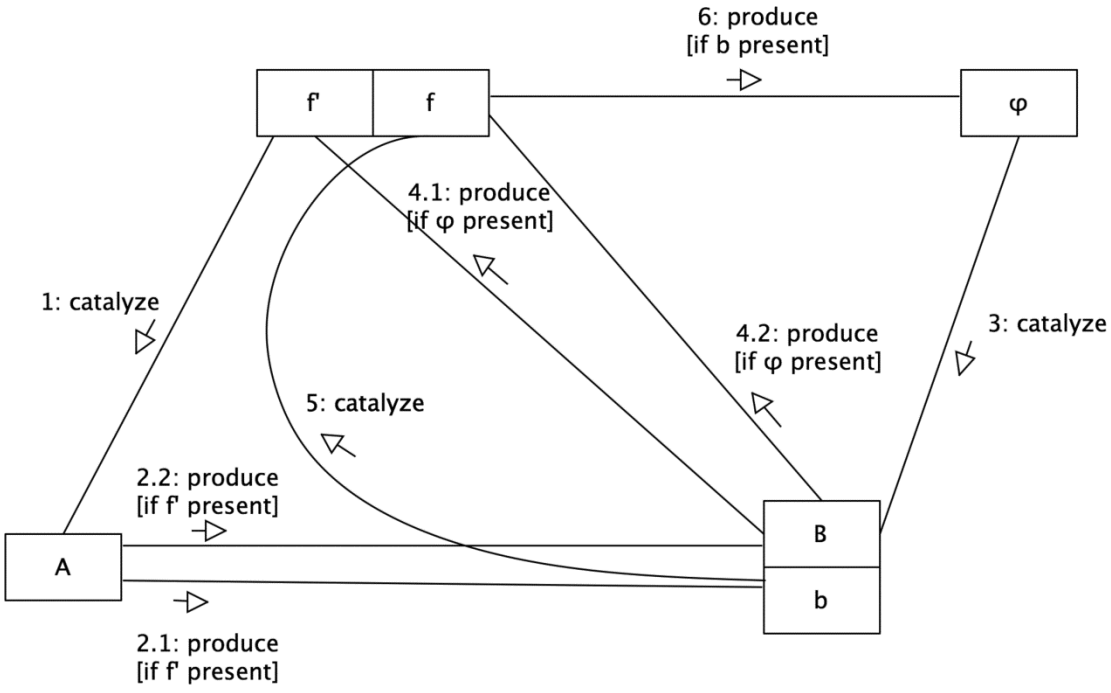
Showing how the loop-free sequential composition of the activity diagram can be developed into something closer to (M,R) requires specification not just of objects and their activities, but of the necessary links between objects. Just as the activity diagram makes explicit the functions pertaining to each class in the class diagram, the communication diagram shows how each object is connected with other objects. Each communication link is annotated as either productive or catalytic. Since the productive activities each result in two outcomes, with the exception of the $f \rightarrow \varphi$ reaction which only produces φ, these are annotated as 2:1, 2:2 etc. Crucially, UML syntax allows for the existence of loops in communication diagrams. The communication diagram is thus, in the terminology of relational biology, a hierarchical composition (Louie, 2009, 2011), meaning that the linear structure of the activity diagram is now circular. The communication diagram (Fig. 4) is of special interest as it may be manipulated in such a way that it strongly resembles the standard (M,R) diagram (Fig. 5, compare to Fig. 1b).



320

321 **Fig. 4: A UML communication diagram for** (M,R) Objects are shown in squares.

322



323

Fig. 5: A UML communication diagram for (M,R) **with classes repositioned to emphasise essential identity to original** (M,R) **diagram of Rosen (inset).** Numbers on communication lines correspond to those of Fig. 4.

3.4 State machine diagrams

(M,R) is often stated to be a state-free system (Louie, 2009, 2011; Rosen, 1991, 2000), so the use of state machine diagrams requires some further explanation. The state machines presented here imagine the fate of individual objects, undergoing biochemical modification under the effects of the various catalysts within the system. The fate of the catalytic objects (*b*, *f'* and φ) is not explicitly specified in classic representations of (M,R) (Louie, 2009, 2011; Rosen, 1991, 2000). If they are taken to be immortal, they will accumulate. In our previous Bio-PEPA realization of (M,R), a wear-and-tear function was incorporated to prevent this (Gatherer and Galpin, 2013). Here, we choose to use each catalytic object three times before removing it from the system. Recording the number of times each catalytic object has been used could be accomplished by the addition of a memory attribute to the class Enzyme, which would then be inherited by its three daughter classes (Fig. 2). The value held by this memory attribute would be increased by a private function activated each time the main function of the object – catalyseSubstrates(Substrate) - was activated. This has not been added to Fig. 2 in order to keep the Class diagram as generic as possible. Since (M,R) in its original form makes no provision for wear-and-tear on the catalysts, there can be no absolutely correct way to represent it when translating (M,R) into an alternative representation.

Metabolic objects (A, B and *f*) by contrast, are converted to other metabolic objects when the appropriate catalytic objects are available (Fig. 6). These conversions can be seen in the context of the whole system on the activity diagram (Fig. 3). The state machine diagrams make explicit how these activities relate to, and transform, individual objects. Just as relational biology allows for sequential compositions – analogous to the UML activity diagram (Fig. 3) – but denies that these constitute a full description of (M,R), it also allows for the individual components of (M,R) to have states, while denying that the (M,R) system as a whole can be represented as a state machine (Rosen, 1991, 2000).

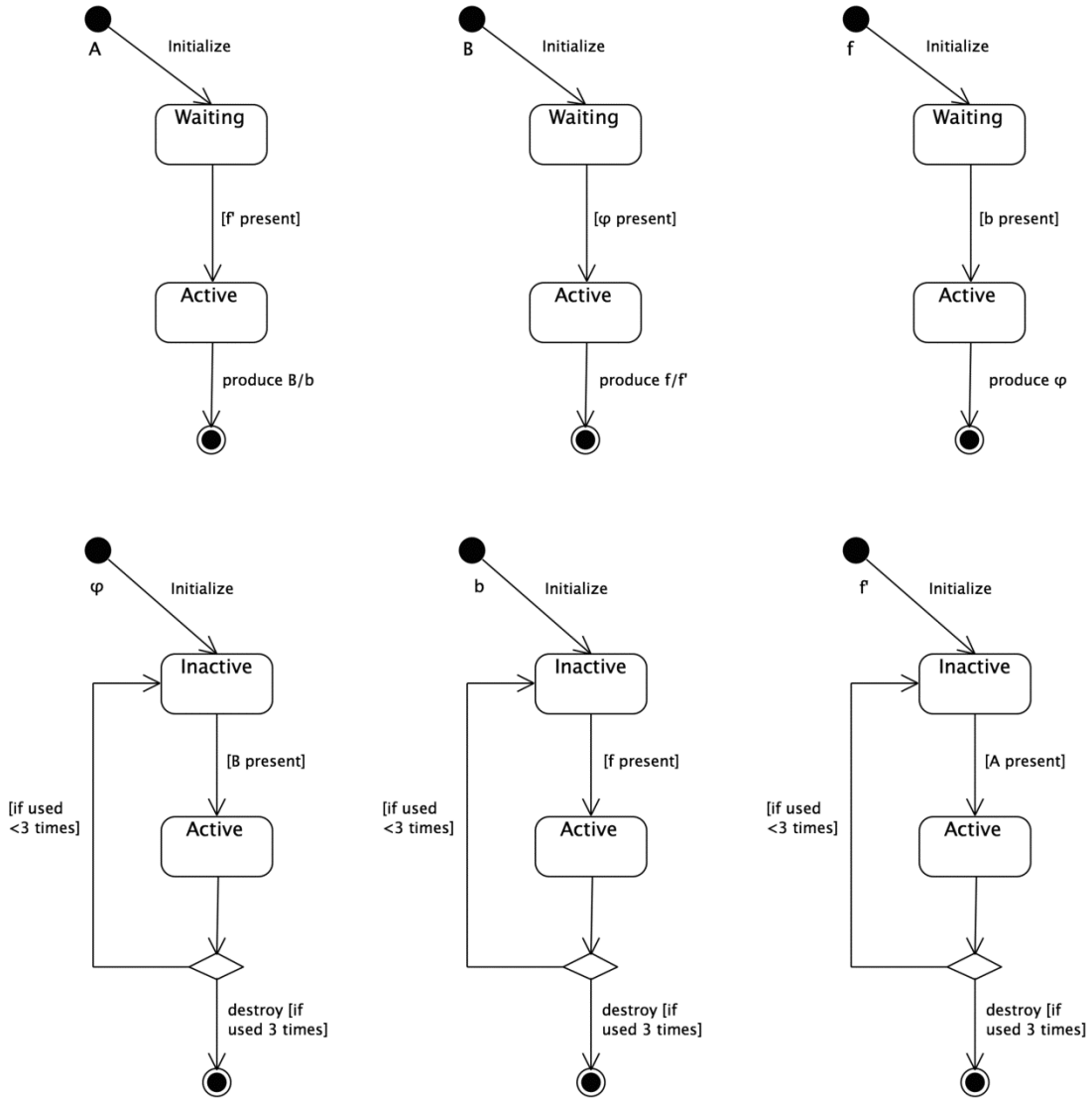


Fig. 6: UML state machine diagrams for individual classes in (M,R)**.** The initialization point is indicated using the filled circle (●) and the termination point using the filled circle within another circle (⦿). Choices are represented as diamonds.

As well as the issue of the computability of (M,R), relational biology also denies its reducibility to its component parts, in other words whether or not we can combine these individual state machine diagrams (Fig. 6) into a state machine diagram for the entire system. We attempt to do this in Fig. 7, in which we define

states of the whole system, positioned in a circular entailment structure. This is permissible within UML provided entry and exit points are specified. These are arbitrary and may be placed anywhere within the diagram. The reduction of our higher level states ("Metabolize", "Repair" and "Replicate") to the states of each individual component (Fig. 6) is assisted by the annotatory rectangles in Fig. 7. The system state "Metabolize", for instance, is achieved when object A is in its individual state "Active", and object *f'* is in its individual state "Active". System state "Metabolize" also initializes an object of class B, thus creating as output an object B in individual state "Waiting", and destroying an object A. The object *f'* will either be destroyed or enter individual state "Inactive" depending on its prior usage. The reduction of the other system states to their component object states is left to the reader.

Although we believe that it is possible to see how the system states of Fig. 7 are reducible to the individual object states of Fig. 6, it is admittedly less easy to see how Fig. 7 handles the concept of time. While the activity diagram (Fig. 3) and the object state diagrams (Fig. 6) can illustrate the effect of an individual object within the system over its life-cycle, they cannot convey the state of the entire system at any one point in time. Indeed, Fig. 7 implies that the three system states are mutually exclusive – that (M,R) is either in a state of metabolism or repair or replication, but only ever in one at a time. One might posit that (M,R) can cycle through the three states of Fig. 7 at such speed that they appear to be operating simultaneously. However, this is a contrived and unsatisfactory solution. At this point, UML has reached the boundaries of its usefulness for (M,R). Other authors

have also tested UML to the point of failure in modelling biological systems (Read et al., 2014) Handling system states within (M,R) may require the application of methods which can process concurrent states, such as Petri Nets (Chaouiya, 2007; Rohr et al., 2010).

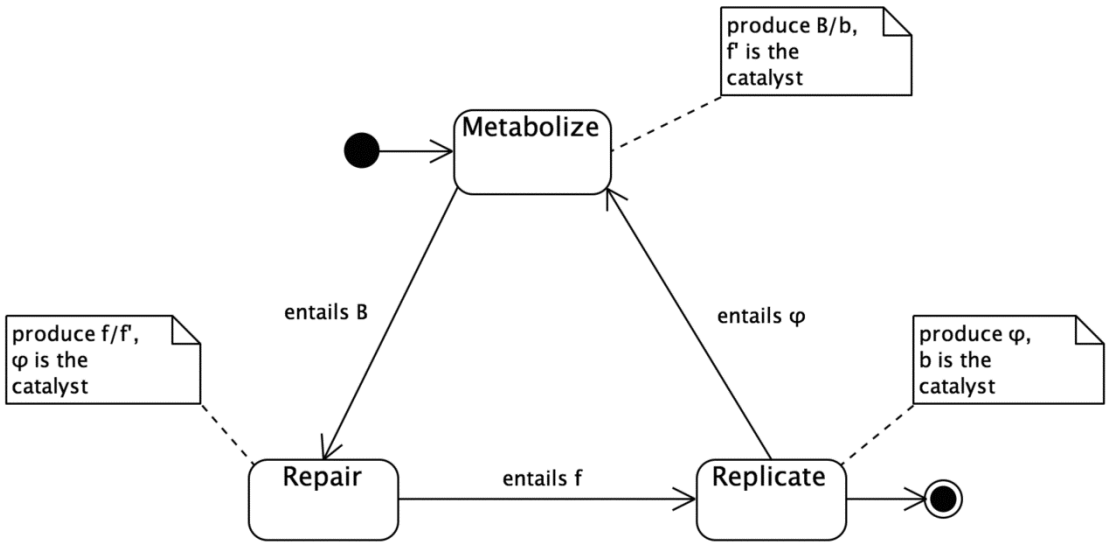


Fig. 7: A UML state machine diagram for the totality of (M,R) representing the entailment structure. The arbitrary initialization point is indicated using the filled circle (●) and the arbitrary termination point using the filled circle within another circle (⦿). Folded-corner rectangles with dotted lines are annotatory.

4. Discussion

Unified Modelling Language (UML) is a diagrammatic notation standard (maintained by the Object Management Group) that provides a set of rules for representing objects and their relationships within systems. UML was conceived as a preliminary tool to define the technical specification of an object-oriented computer application before its translation into computer code using an appropriate higher-

409 level language. Successful object-oriented analysis of a system strongly implies the
410 possibility of successful object-oriented computation of that system. We believe
411 that we have successfully produced an object-oriented analysis of (M,R) using UML.
412 It is acknowledged that some problems remain, which are discussed further below.
413 However, a compelling piece of evidence for the possibility of object-orientation of
414 UML lies in the close similarity of the classic (M,R) diagram (Fig. 1) to a UML
415 communication diagram (Figs. 4 and 5). Indeed we are tempted to advance the
416 opinion that the classic (M,R) diagram was an object-oriented communication
417 system *avant la lettre*. (M,R) therefore contains the seeds of object-orientation
418 within it, and the unfolding of these possibilities is both logical and necessary to a
419 full understanding of (M,R).
420
421 Previous attempts at computation of (M,R) have fallen short largely because of
422 doubts concerning the way that (M,R) has been coded, resulting in computational
423 systems that have either fewer or more components than (M,R), or that perform
424 certain operations in a way that (M,R) does not – in other words that alter (M,R)'s
425 entailment structure. We propose that object-oriented analysis enables us to
426 produce the most precise computational representation of (M,R) to date, one which
427 ought to enable us to progress to a precise computational realization of (M,R) in
428 terms of object-oriented code. Nevertheless, there are certain areas where we have
429 had to make decisions about how to represent (M,R) in UML, where the classic
430 relational biology literature does not provide much in the way of guidance. The
431 potential therefore exists for corruption of (M,R), resulting in yet another slip from
432 true model to mere simulation. We discuss these below.

1) The UML communication diagram (Fig. 4) may be rearranged without disturbing its topology to produce something very similar to the classic (M,R) representation (Fig. 5). However, we cannot claim complete identity, since our communication diagram therefore has objects *f* and *f'* where the original (M,R) diagram has *f*, and objects B and *b* where the original (M,R) diagram has entity B.
2) This distinction is maintained in the UML class diagram (Fig. 2) where we have a total of 6 classes within the system.
3) Our activity (Fig. 3) and state machine (Figs. 6 and 7) have starting and termination points specified. This is because the rules of UML require state machines to compute over time and to have strict rules about when certain processes will terminate or continue. We do not believe that the starting and termination points are controversial in Fig. 3 or Fig. 6 as these represent parts of (M,R) that are acknowledged to behave as mechanisms. In Fig. 7, it is admitted that the placement of the starting and termination points produces a certain awkwardness in the diagram, since the circular entailment structure clearly produces a circular state structure.

We believe it is clear on inspection that Fig. 3 is reducible to Fig. 6, or conversely that Fig. 3 is clearly also a larger machine composed of the six smaller machines in Fig. 6. We believe that Fig. 7 also represents a machine, although seeing how it is reducible to Fig. 3, and therefore by implication to Fig. 6, requires a little more careful scrutiny.

5. Conclusions

Rosen intended (M,R) to be broadly representative of living systems, in that the production of B from A may be taken to represent the totality of metabolic reactions in a cell. φ, *b* and *f'* are catalysts, for instance enzymes. B and *f* are the products of metabolism and substrates for further metabolic reactions. The only external necessity is the production of the basic foodstuff in the form of A, which is purely a substrate and neither product nor catalyst. (M,R) may also be treated more literally as a small network with three reactions and three catalysts. For further clarification of the subtle distinction between B and *f* as substrates and *b* and *f'* as catalysts see Letelier et al (2006) and section 8 of Cardenas et al (2010). The necessity of multifunctionality of the component parts of an *(M,R*) system is further discussed by Cornish-Bowden and Cardenas (2007), and on this basis we believe that division of our components into metabolic/catalytic objects – B/*b* and *f*/*f'* respectively is justified.

UML has the advantage that, by representing all elements of an analysis in a diagrammatic format, there are no hidden modifications of the system being realised. Seeing how one UML diagram is implied, indeed necessitated, by the others is self-evident once the principles of UML are understood. The entailment structures of the UML realization of (M,R) are the same as those of (M,R) itself, which is the crucial requirement for a model of a system as opposed to a simulation. Therefore, we have come closer to a computer model of (M,R) than has been

previously achieved. Since correctly formed UML enables the generation of object-oriented code which captures the object-oriented structure specified in the UML analysis, we believe that such code may fulfil the requirements for an accurate model of (M,R) on a Turing-architecture computer, thus subsuming relational biology into standard computational systems biology. First, however, we present the object-oriented UML analysis for the scrutiny of the relational biology and systems biology communities.

Acknowledgments

We thank various colleagues and reviewers for their comments and suggestions, some of which we have incorporated into our argument. No external funding bodies contributed to this work.

Data Access and Ethics Statement

No new data were created in this study. No ethical approval was required for this study.

References

Baianu, I.C., 2006. Robert Rosen's work and complex systems biology. Axiomathes 16, 25-34.

Beck, K., Cunningham, W., 1989. A laboratory for teaching object-oriented thinking, OOPSLA89. SIGPLAN Notices.

Bersini, H., Klatzmann, D., Six, A., Thomas-Vaslin, V., 2012. State-transition diagrams for biologists. PLoS One 7, e41165.

Booch, G., Rumbaugh, J., Jacobson, I., 1998. The Unified Modeling Language User Guide. Addison Wesley Longman, Reading, Massachusetts.
Breyer, J., Ackermann, J., McCaskill, J., 1998. Evolving reaction-diffusion ecosystems with self-assembling structures in thin films. Artificial Life 4, 25-40.
Cardenas, M.L., Letelier, J.C., Gutierrez, C., Cornish-Bowden, A., Soto-Andrade, J., 2010. Closure to efficient causation, computability and artificial life. J Theor Biol 263, 79-92.
Casti, J.L., 1988 The theory of metabolism-repair systems Applied Mathematics and Computation 28, 113-154.
Chaouiya, C., 2007. Petri net modelling of biological networks. Brief Bioinform 8, 210-219.
Cho, K.-H., Johansson, K.H., Wolkenhauer, O., 2005. A hybrid systems framework for cellular processes. Biosystems 80, 273-282.
Chu, D., Ho, W.K., 2006. A category theoretical argument against the possibility of artificial life: Robert Rosen's central proof revisited. Artificial Life 12, 117-134.
Chu, D., Ho, W.K., 2007. The localization hypothesis and machines. Artif Life 13, 299-302.
Cornish-Bowden, A., Cardenas, M.L., 2005. Systems biology may work when we learn to understand the parts in terms of the whole. Biochem Soc Trans 33, 516-519.
Cornish-Bowden, A., Cardenas, M.L., 2007. Organizational invariance in (M,R)-systems. Chemistry and Biodiversity 4, 2396-2406.
Cornish-Bowden, A., Cardenas, M.L., Letelier, J.C., Soto-Andrade, J., 2007. Beyond reductionism: metabolic circularity as a guiding vision for a real biology of systems. Proteomics 7, 839-845.
Cottam, R., Ranson, W., Vounckx, R., 2007. Re-mapping Robert Rosen's (M,R)-Systems. Chemistry and Biodiversity 4.
Fowler, M., 2004. UML distilled. A brief guide to the standard object modeling language., 3 ed. Addison-Wesley, Boston, MA.
Gatherer, D., Galpin, V., 2013. Rosen's (M,R) system in process algebra. BMC Systems Biology 7, 128.
Goertzel, B., 2002. Appendix 2. Goertzel versus Rosen: Contrasting views on the autopoietic nature of life and mind., Creating Internet Intelligence. Kluwer Academic/Plenum Publishers, New York.
Goudsmit, A.L., 2007. Some reflections on Rosen's conceptions of semantics and finality. Chemistry and Biodiversity 4, 2427-2435.
Gutierrez, C., Jaramillo, S., Soto-Andrade, J., 2011. Some thoughts on A.H. Louie's "More Than Life Itself: A Reflection on Formal Systems and Biology". Axiomathes 21, 439-454.
Gwinn, T., 2010. Critiques of critiques.
Available: http://www.panmere.com/?cat=18, In: Gwinn, T. (Ed.), Panmere. Rosennean complexity and other interests.
Hillston, J., 2005. Process algebras for quantitative analysis, 20th Annual Symposium on Logic in Computer Science. IEEE Computer Society, pp. 1-10.
Kineman, J.J., 2007. Modeling relations in nature and eco-informatics: a practical application of rosennean complexity. Chemistry and Biodiversity 4, 2436-2457.
Kineman, J.J., 2011. Relational Science: A Synthesis. Axiomathes 21, 393-437.
Landauer, C., Bellman, K., 2002. Theoretical biology: Organisms and mechanisms. AIP Conference Proceedings 627, 59-70.
Letelier, J.C., Cardenas, M.L., Cornish-Bowden, A., 2011. From L'Homme Machine to metabolic closure: Steps towards understanding life. J Theor Biol 286, 100-113.

555 Letelier, J.C., Marin, J., Mpodozis, J., 2003. Autopoietic and (M,R)-systems. Journal
556 of Theoretical Biology 222, 261-272.
557 Letelier, J.C., Soto-Andrade, J., Guinez Abarzua, F., Cornish-Bowden, A., Cardenas,
558 M.L., 2006. Organizational invariance and metabolic closure: analysis in terms of
559 (M,R) systems. J Theor Biol 238, 949-961.
560 Louie, A.H., 2004. Rosen 1, Goertzel 0: Comments on the appendix "Goertzel versus
561 Rosen", Available: http://panmere.com/rosen/Louie%20-%20GoetzelvsRosen.pdf.
562 Louie, A.H., 2005. Any material realization of the (M,R)-systems must have
563 noncomputable models. J Integr Neurosci 4, 423-436.
564 Louie, A.H., 2007a. A living system must have noncomputable models. Artificial Life
565 13, 293-297.
566 Louie, A.H., 2007b. A Rosen etymology. Chemistry and Biodiversity 4, 2296-2314.
567 Louie, A.H., 2009. More than Life Itself. A Synthetic Continuation in Relational
568 Biology. Ontos Verlag, Frankfurt.
569 Louie, A.H., 2011. Essays on More Than Life Itself Axiomathes 21, 473-489.
570 Louie, A.H., 2015. A metabolism–repair theory of by-products and side-effects.
571 International Journal of General Systems 44, 26-54.
572 Louie, A.H., Kercel, S.W., 2007. Topology and Life redux: Robert Rosen's relational
573 diagrams of living systems. Axiomathes 17, 109-136.
574 McMullin, B., 2004. Thirty years of computational autopoiesis: a review. Artificial
575 Life 10, 277-295.
576 McMullin, B., Varela, F.J., 1997. Rediscovering computational autopoiesis, SFI
577 Working Paper 97-02-012. Santa Fe Institute.
578 Mossio, M., Longo, G., Stewart, J., 2009. An expression of closure to efficient
579 causation in terms λ-calculus. Journal of Theoretical Biology 257, 489-498.
580 Object Management Group, 2011. Unified modeling language superstructure
581 specification v2.4.
582 Ono, N., Ikegami, T., 2002. Selection of catalysts through cellular reproduction, In:
583 Standish, R., Bedau, M.A., Abbass, H.A. (Eds.), 8th International Conference on
584 Artificial Life. MIT Press, pp. 57-64.
585 Piedrafita, G., Cornish-Bowden, A., Moran, F., Montero, F., 2012a. Size matters:
586 influence of stochasticity on the self-maintenance of a simple model of metabolic
587 closure. J Theor Biol 300, 143-151.
588 Piedrafita, G., Montero, F., Moran, F., Cardenas, M.L., Cornish-Bowden, A., 2010. A
589 simple self-maintaining metabolic system: robustness, autocatalysis, bistability. PLoS
590 Computational Biology 6, pii: e1000872.
591 Piedrafita, G., Ruiz-Mirazo, K., Monnard, P.A., Cornish-Bowden, A., Montero, F.,
592 2012b. Viability conditions for a compartmentalized protometabolic system: a semi-
593 empirical approach. PLoS One 7, e39480.
594 Prideaux, J.A., 2011. Kinetic models of (M,R)-systems. Axiomathes 21, 373-392.
595 Radó, T., 1962. On non-computable functions. Bell System Technical Journal 41,
596 877–884.
597 Read, M., Andrews, P.S., Timmis, J., Kumar, V., 2014. Modelling biological
598 behaviours with the unified modelling language: an immunological case study and
599 critique. Journal of the Royal Society. Interface 11, DOI: 10.1098/rsif.2014.0704
600 Rohr, C., Marwan, W., Heiner, M., 2010. Snoopy--a unifying Petri net framework to
601 investigate biomolecular networks. Bioinformatics 26, 974-975.
602 Rosen, R., 1958a. A relational theory of biological systems. Bull. Math. Biophys. 20,
603 245-260.

Rosen, R., 1958b. The representation of biological systems from the standpoint of the theory of categories. Bull. Math. Biophys. 20, 317-341.
Rosen, R., 1959. A relational theory of biological systems II. Bull. Math. Biophys. 21, 109-128.
Rosen, R., 1963. Some results in graph theory and their application to abstract relational biology. Bulletin of Mathematical Biophysics 25, 231-241.
Rosen, R., 1972. Some Relational Cell Models: The Metabolism-Repair System, In: Rosen, R. (Ed.), Foundations of Mathematical Biology. Academic Press, New York.
Rosen, R., 1989. The roles of necessity in biology, In: Casti, J.R., Karlqvist, A. (Eds.), Newton to Aristotle: Toward a theory of models for living systems. Birkhauser, New York, pp. 11-37.
Rosen, R., 1991. Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life. Columbia University Press, New York.
Rosen, R., 2000. Essays on Life Itself. Columbia University Press, New York.
Roux-Rouquie, M., Caritey, N., Gaubert, L., Rosenthal-Sabroux, C., 2004. Using the Unified Modelling Language (UML) to guide the systemic description of biological processes and systems. Biosystems 75, 3-14.
Suzuki, K., Ikegami, T., 2008. Shapes and self-movement in protocell systems. Artificial Life 15, 59-70.
Turing, A.M., 1936. On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. 42, 230-265.
van Beijnum, B.J., Widya, I.A., Marani, E., 2010. Modeling the vagus nerve system with the Unified Modeling Language. J Neurosci Methods 193, 307-320.
Varela, F., Maturana, H., Uribe, R., 1974. Autopoiesis: the organization of living systems, its characterization and a model. Biosystems 5, 187-196.
Visual Paradigm, 2010. Visual paradigm for UML. UML tool for software application development.
Webb, K., White, T., 2005. UML as a cell and biochemistry modeling language. Biosystems 80, 283-302.
Wells, A.J., 2006. In defense of mechanism. Ecological Psychology 18, 39-65.
Whitehead, A.N., Russell, B., 1963 [1927]. Principia Mathematica, 2 ed. Cambridge University Press, Cambridge.
Witten, T.M., 2007. (M,R)-systems, (P,M,C)-nets, hierarchical decay, and biological aging: reminiscences of Robert Rosen. Chemistry and Biodiversity 4, 2332-2344.
Wolkenhauer, O., 2007. Interpreting Rosen. Artificial Life 13, 291-292.
Wolkenhauer, O., Hofmeyr, J.-H., 2007. An abstract cell model that describes the self-organization of cell function in living systems. Journal of Theoretical Biology 246, 461-476.
Yan, Q., 2010. Bioinformatics for transporter pharmacogenomics and systems biology: data integration and modeling with UML. Methods in Molecular Biology 637, 23-45.
Zeleny, M., 1978. APL-AUTOPOIESIS: Experiments in self-organization of complexity., Progress in Cybernetics and Systems Research. Hemisphere Publishing Corp., Washington, pp. 65-84.