A computational study of liposome logic: towards cellular computing from the bottom up

James Smaldon, Francisco J. Romero-Campero, Francisco Fernández Trillo, Marian Gheorghe, Cameron Alexander, Natalio Krasnogor

Abstract In this paper we propose a new bottom-up approach to cellular computing, in which computational chemical processes are encapsulated within liposomes. This "liposome logic" approach (also called vesicle computing) makes use of supra-molecular chemistry constructs, e.g. protocells, chells, etc. as minimal cellular platforms to which logical functionality can be added. Modeling and simulations feature prominently in "top-down" synthetic biology, particularly in the specification, design and implementation of logic circuits through bacterial genome reengineering. The second contribution in this paper is the demonstration of a novel set of tools for the specification, modelling and analysis of "bottom-up" liposome logic. In particular, simulation and modelling techniques are used to analyse some example liposome logic designs, ranging from relatively simple NOT gates and NAND gates to SR-Latches, D Flip-Flops all the way to 3 bit ripple counters. The approach we propose consists of specifying, by means of P systems, gene regulatory network-like systems operating inside proto-membranes. This P systems specification can be automatically translated and executed through a multiscaled pipeline composed of dissipative particle dynamics (DPD) simulator and Gillespie's stochastic simulation algorithm (SSA). Finally, model selection and analysis can be performed through a model checking phase. This is the first paper we are aware of that brings to bear formal specifications, DPD, SSA and model checking to the problem of modeling target computational functionality in protocells. Potential chemical routes for the laboratory implementation of these simulations are also discussed thus for the first time suggesting a potentially realistic physiochemical implementation for membrane computing from the bottom-up.

Keywords Simulation and modelling · Vesicle computing · Cellular computing · Synthetic biology · Dissipative particle dynamics · Stochastic simulation · Model checking · Logic gates · Chells · Protocells

J. Smaldon · F. J. Romero-Campero · N. Krasnogor (✉)
School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
e-mail: nxk@cs.nott.ac.uk

J. Smaldon
e-mail: jqs@cs.nott.ac.uk

F. J. Romero-Campero
e-mail: fxc@cs.nott.ac.uk

M. Gheorghe
Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, UK

F. Fernández Trillo · C. Alexander
School of Pharmacy, University of Nottingham, University Park, Nottingham NG7 2RD, UK

Introduction

Just as an electrical engineer can construct circuits from modules with common inputs and outputs without consideration of internal module construction, the standardisation of biological components proposed by T.F. Knight, D. Endy, R. Weiss and others (Endy 2005; Knight 2003; Heinemann and Panke 2006; Serrano 2007), and exemplified in the MIT biobricks project (Shetty et al. 2008), may allow a bioarchitect to construct biological systems with prespecified phenotypes in a more scalable way. One important application within the field of Synthetic Biology is Cellular Computing. Cellular Computing (Amos 2004) seeks the construction of genes, signals and metabolic

regulation networks within organisms[1] which, by implementing boolean logic gate circuits, can accomplish specific computational tasks (Tan et al. 2007).

In this computing paradigm, individual cells perform a small part of a computation in a highly asynchronous fashion with communication taking place only between cells which are within a short distance from one another [so called amorphous computation (Abelson et al. 2000)]. In particular, cellular logic NOT and AND gates were first characterised in detail by Ron Weiss et al. (Weiss and Basu 2002) in vitro and with "bioSPICE", an ODE based modelling technique. Since then, several small scale systems have been constructed, either in the lab or in simulation (Amos 2004). Other examples of in vitro implementations of cellular computing systems include band detectors, coupled oscillations (Basu et al. 2004, 2005) and, more recently, a solution to the three vertices Hamiltonian path problem (Baumgardner et al. 2009).

By creating *modular* logic gates that behave in well characterised ways, it might be possible to abstract away some of the biological detail when designing more complex synthetic biology systems (Andrianantoandro et al. 2006). In doing so, the behaviour of a composite system becomes more predictable and designs can be constructed and prototyped in silico before attempting to implement them in the lab.

Thus far cellular computing has only been investigated from the "top down" perspective, that is, by modifying existing organisms through the incorporation of synthetic biological regulatory netoworks (BRNs). Little, if any, attention has been paid to how such distributed computation might be implemented from the "bottom up" perspective, that is, by using protocells (Rasmussen et al. 2008) or "chells" [artificial chemical cells (Cronin et al. 2006)] rather than fully fledged biological cells. Although highly innovative, trying to use bacteria to perform amorphous computation is like trying to build a glider by knocking out parts out of a jumbo jet. The problem is, both technically and philosophically, one of managing complexity: bacterial cells have evolved for millions of years and they carry too much evolutionary "baggage". Before any useful and general computation, rather than specific as exemplified by the above mentioned references, can be achieved, this unwanted complexity must be tamed.

In contrast, the approach we suggest in this paper retains all the advantages of amorphous computing at the nanoscale (e.g., redundancy, massive parallelism, asynchronous local processes, self-organisation, etc.) but by starting from the bottom up with engineered components of prespecified and limited complexity, one avoids unnecessary biological nuisances from the start. Moreover, by turning to chemical cellular-like constructs, compartmentalisation and orthogonality are maximised while crosstalk minimised, thus the approach we propose might provide a route to, e.g., more reliable programmable chemical comminication and drug delivery systems (Pasparakis and Alexander 2008; Gardner et al. 2009).

In this paper we present a bottom-up approach to cellular computing, *liposome logic* (or vesicle computing), which involves the encapsulation of logical functionalities within vesicles. Liposome logic makes use of supramolecular chemistry constructs, e.g. protocells, chells, etc., to encapsulate logical functionality. The vesicle computing approach proposed is related to the effort to build a semi-synthetic protocell (Luisi et al. 2006), however, the goal in this case is to create a chemical automaton that is a useful platform for design and implementation of cellular computing circuits, rather than attempting to reproduce exactly the properties of living systems (Although there will, of course, be some synergy between the two). Liposome logic could then be used for designing the next generation of medical devices going beyond of the current state of the art (MacDiarmid et al. 2009). Constructing a cellular computing device from the bottom-up, enables the designer of a vesicle computing system to benefit from design abstractions not available in existing microbiological organims. For example, vesicles could be used for encapsulation and implementation hiding, as hierarchical (i.e. nested) structures of membranes could be created with clearly defined inputs and outputs, that create boundaries around functionality just as organelles contain specific functions in eukaryotic cells. This kind of compartmentalisation will enable interference between BRNs to be minimized, and the need for multiple promoter sequences/transcription factors to be reduced, which may help to overcome one of the key problems in cellular computing and top-down synthetic biology, that of unexpected interactions between synthetic regulatory networks and the underlying cell processes.

As a first step towards the realisation of vesicle computers in vitro, a new simulation and modelling framework, enabling the formal specification, multi-scale simulation and model checking of liposome logic designs is employed to investigate the behaviour and dynamics of systems of example BRN-like logic gates encapsulated within liposomes. The modelling approach we propose consists of specifying by means of P systems, BRN-like systems operating inside proto-membranes. This P system specification can be automatically translated and executed through a multiscaled pipeline composed of a dissipative particle dynamics (DPD) simulator and Gillespie's stochastic simulation algorithm (SSA). Finally, model selection and analysis can be performed through a model

[1] In what follows we will call these networks biological regulatory networks (BRN).

checking phase. We analyse the behaviour of liposome logic systems increasing in complexity from relatively simple NOT and NAND gates to SR-Latches, D Flip-Flops all the way to 3 bit ripple counters. This is the first paper we are aware of that brings to bear formal specifications, DPD, SSA and model checking into the problem of modeling target functionality in chells. Potential chemical routes for the laboratory implementation of these Liposome logic simulations are also discussed.

Methods

In this section we describe the proposed modeling pipeline as depicted in Fig. 1. Liposome logic modeling starts with the specification of the logic circuitry using P systems. The P system specification is then executed through DPD or an advanced Gillespie's stochastic algorithm implementation. The decision of whether to execute the model in DPD or directly through SSA depends on the time and lengths scales of interest and also on whether physical volumes should be modelled explicitly, i.e. geometrically, or implicitly, i.e. topologically. If the length/time scales are large and the volume is only topologically represented then SSA is used. A further analytical level is afforded by the use of model checking techniques. In what follows we describe the P system specification formalism, DPD, SSA and model checking.

P systems as a specification framework

P systems (Păun 2002) constitute a recently developed specification framework bringing into systems and synthetic biology methodologies from formal rewriting systems distributed over multicompartmentalised regions. Our P systems based approach to modelling falls within the classification of computational, rule-based, modular and discrete stochastic modelling frameworks. In this work, we use a variant called *stochastic P systems* specially suitable for the scalable and parsimonious specification of cellular systems exhibiting evident levels of stochasticity (Pérez-Jiménez and Romero-Campero 2006).

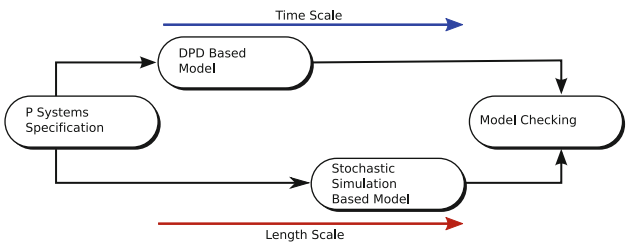


Fig. 1 Computational pipeline for liposome logic. Computing circuitry is specified through P systems that could then be interpreted and simulated either through a DPD simulator or, if the time and length scales are larger, through a stochastic simulation algorithm (e.g. Gillespie's SSA). A further analysis can be performed using model checking

The main components of a stochastic P systems are objects, representing molecular species; compartments defined by membranes containing multisets of objects and rewriting rules specifically associated with each compartment describing the molecular interactions taking place in and between different compartments. The simulation component of our framework does not currently support the fission or fusion of the vesicle membranes, but the provision of such functionality is already included through the formal language we used to describe liposome based logical systems, namely, P systems. Indeed, the controlled fusing and splitting of vesicles may be used for computations, and so the addition of this functionality will be added to the simulation algorithms in the near future.

Formally, a stochastic P system is a construct

$$\Pi = (O, L, \mu, M_{l_1}, M_{l_2}, \ldots, M_{l_n}, R_{l_1}, \ldots, R_{l_n})$$

where

- O is a finite alphabet of objects specifying the molecular species in the system.
- $L = \{\ l_1, ..., l_n\ \}$ is a finite set of labels identifying compartment types.
- μ is a membrane structure containing $n \geq 1$ membranes defining compartments arranged in a hierarchical manner. Each membrane is identified in a one to one manner with a label in L which determines its type.
- M_{l_i} for each $1 \leq i \leq n$, is the initial multiset of objects over O placed inside the compartment defined by the membrane with label l_i in the initial state of the system.
- $R_{l_i} = \{r_1^{l_i}, \ldots, r_{k_{l_i}}^{l_i}\}$, for each $1 \leq i \leq n$, is a finite set of rewriting rules associated with the compartment with label $l_i \in L$ and of the following general form:

$$o_1[o_2]_{l_i} \xrightarrow{c} o_1'[o_2']_{l_i} \tag{1}$$

with o_1, o_2, $o_1{}'$, $o_2{}'$ multisets (potentially empty) of objects over O representing the molecular species and the stoichiometries involved in the molecular interaction represented in the rule. The label $l_i \in L$ identifies the compartment where the interaction takes place. These multiset rewriting rules can potentially change both the inside and outside of (proto)membranes. An application of a rule of this form replaces simultaneously the multisets o_1 outside membrane l_i and o_2 inside membrane l_i by the multisets $o_1{}'$ and $o_2{}'$, respectively. A stochastic constant c is associated specifically with each rule in order to compute how often the rules are applied and the time elapsed between rule applications according to Gillespie's theory of stochastic kinetics (Gillespie 2007). Specifically, rewriting rules are selected

according to an extension of Gillespie's well known *stochastic simulation algorithm* (SSA) (Gillespie 2007) to the multicompartmental structure of P system models (Romero-Campero et al. 2009).

The P systems are intended to describe a functional specification of the behaviour of computational liposomes, where logic gate based computation arises as a result of interactions between these elements. The specification language is deliberately general and does not reference any particular chemical or biological system. It is, however, possible to speculate on the physical interpretation of the processes described by rules specified as in definition 1. For example, a rule in which a multiset of objects (labeled o_2 in 1) inside a compartment are replaced with the multiset specified by o_2' could be interpreted as a simple chemical interaction, such as reaction, the binding of two transcription factors, or the binding of a ribosome to a DNA sequence. Rules which specify multisets of objects external to the compartment (multiset o_1 in Definition 1) altering the multiset of objects inside the compartment might be used to represent the diffusion of small molecules into or out of the vesicle accross the membrane or the action of signal transduction mechanisms via the expression of receptors on the membrane surface of the vesicle. Currently the implementations of the DPD and SSA simulation techniques used in this paper support simple chemical interactions within or outside of the membrane and the diffusion of objects accross the membrane.

Cellular phenotypes arise from the orchestration of the interactions between different molecular modules acting as discrete entities whose functionalities are up to certain point separable from one another (Hartwell et al. 1999). The interaction modality in cellular systems is an intense research field in systems and synthetic biology which is unraveling specific modular patterns in BRNs (Alon 2007). Biological modularity is thus one of the cornerstones of synthetic biology (Andrianantoandro et al. 2006) and its relevance for systems and synthetic biology has been recently emphasized by Mallavarapu et al. (2009). In this work we follow a modular modelling approach whereby models are incrementally, parsimoniously and hierarchically built by combining virtual parts that are available from a library. This library comprises a set of elementary modules that specify biological regulatory-like networks as well as modules describing the regulation of specific gene promoters widely used in synthetic biology.

A *P system module* is defined as a set of rewriting rules, each of the form in Eq. 1, for which some of the objects, stochastic constants or the labels of the compartments involved might be *variables*. This facilitates reusability and parsimony in the development of models. Large models can be specified by integrating commonly found modules that are then further instantiated with specific values obtained experimentally. Formally, a P system module M is represented as $M(V, C, L)$ where V specifies object variables, which can be instantiated using specific names of molecular species like genes and proteins, C are variables for the stochastic constants associated to the rewriting rules, which can be instantiated using specific affinities between genes and proteins, half lifes for degradation processes, etc. and finally L are variables for the labels of the compartments involved in the rules that might represent different cell compartments, e.g., cytoplasm, lysosome, cellular membrane, etc., or different (proto-)cells altogether.

In the next section the DPD and SSA simulation techniques are described in detail.

Dissipative particle dynamics

Simulations at small length and timescales were performed using a self-developed mesoscopic modelling framework based on the dissipative particle dynamics (DPD) technique. Our framework enables easy specification of large scale models in DPD, and vesicles and other structures that form over the course of the simulation can be extracted and stored for later recombination into new initial states for further simulations. This feature allows for the combinatorial bootstrap of computationally expensive simulations. For example, it is possible to (1) simulate the formation of vesicles and (2) save these emergent structures as to then (3) create a new initial state for a simulation containing those vesicles with the internal volume, perhaps modified to contain particles representing genes, proteins etc.

Dissipative Particle Dynamics is a coarse grained particle simulation technique, in which each particle represents several molecules of a given molecular species, rather than a single atom. By dispensing with the details of individual atoms, the short length and timescale processes can be averaged out, allowing simulation for much larger length and time-scales than is possible with other particle dynamics methods[2]. Simulations are formed by filling a volume with particles and integrating the equations of motion to calculate the particle positions and velocities at each time step. Three forces act between particles in a symmetric pairwise fashion, the dissipative and random forces act as the thermostat in DPD, with the dissipative force removing energy from the system (whilst conserving

[2] Note that the term "dissipative" refers to the fact that energy is not conserved in a DPD simulation, and does not relate to the "dissipative structures" proposed by Ilya Prigogine and others. We also note in passing, that the name "dissipative particle dynamics" is the way in which this family of algorithms is referred to in the literature, and this predates this paper.

momentum) and the random force introducing energy in the system by producing a brownian style motion between particles, Eq. 2 shows the forces acting between two particles i and j. The conservation of momentum in the system means that the hydrodynamics are represented correctly.

$$F_{ij} = F_{ij}^C + F_{ij}^D + F_{ij}^R \tag{2}$$

The conservative force F_{ij}^C simply introduces a parameterisable repulsion between particles types which decreases linearly with distance and is zero at the cut-off distance r_c.

$$F_{ij}^C = \alpha(1 - \frac{|r_{ij}|}{r_c}) \tag{3}$$

The α parameter sets the maximum repulsion for a given pair of types so for example the α parameter will be set to a high value for the interaction between an oil and water particle, as these would be immiscible, but to a smaller value for two water particles.

The dissipative force acts as a drag force, slowing down particles that are approaching one another:

$$F_{ij}^D = -\gamma w^D(r_{ij})(\hat{r}_{ij} \cdot v_{ij})\hat{r}_{ij} \tag{4}$$

where γ is the dissipative force parameter which controls the magnitude of the force, r_{ij} is the distance between particle i and particle j, $\hat{r}_{ij}$ is the unit vector point from particle j to particle i and v_{ij} is the relative velocity between particle i and j and w^D is a weighting function described below.

The random force introduces a random force between each particle pair

$$F_{ij}^R = \frac{\sigma w^R(r_{ij})\theta_{ij}\sqrt{3}\hat{r}_{ij}}{\sqrt{dt}} \tag{5}$$

where σ is the random force parameter controlling the magnitude of the force, θ_{ij} is a uniformly distributed random number with unit variance and w^R is the random weighting function described below. Polymers can also be represented in DPD with harmonic bonding and angle potentials.

$$F_{ij}^S = k(r_{ij} - r_0) \tag{6}$$

where k is the bond strength parameter, and r_0 is the preferred bond length. preferred angles between two bonds can be included with a harmonic 3-body potential

$$U_\theta = \frac{1}{2}k_\theta(\theta - \theta_0)^2 \tag{7}$$

where k_θ is the angle force strength parameter, θ is the angle between the two bonds and θ_0 is the preferred angle.

In Español and Warren (1995) the authors investigated the statistical mechanics of DPD and found that in order to maintain a correct and stable temperature, the σ and γ force parameters should be set according to the following relation:

$$\sigma^2 = \sqrt{2\gamma K_b T} \tag{8}$$

where $k_B T$ is the required particle kinetic energy.

The dissipative and random forces are coupled with weighting functions $w^D(r_{ij})$ and $w^R(r_{ij})$. One of these functions may be chosen arbitrarily, and we use the weighting proposed by Groot and Warren (1997) for the random force:

$$W^R(r) = \begin{cases} (1-r) & \text{when}(r<1) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The dissipative weight function is then derived from the following relation:

$$W^D(r) = [W^R(r)]^2 \tag{10}$$

Our implementation of DPD contains a collision based artificial chemistry supporting first and second order reactions. A collision is considered to have occurred if two particles come within a parameterisable collision radius of one another. In the simulations presented here the collision radius is set to the force interaction radius r_c. Each reaction is assigned a rate c_{dpd}, which is the rate at which colliding particles will react as a result of that collision per DPD time unit (the per collision rate is therefore $c_{dpd} * dt$. If the types of the colliding particles match the reactant types for a reaction, then a pseudo random number is generated, and if this number is less than the rate then the reaction occurs and the types of the particles are changed to represent the products of the reaction. In the case of first order reactions, for each particle the reaction is attempted once per timestep, with a rate $c_{dpd}*dt$.

Groot and Warren gave a thorough explanation of the correct setting of the DPD parameters (Groot and Warren 1997) and described a method for parameterising the conservative force based on the immiscibility of fluids, as well as a new integration method more suitable to the larger timesteps taken in DPD. The work by Groot and Rabone (2001) showed simulations of poration in a phospholipid membrane and described the coarse graining procedure. These papers define the de facto standard implementation of DPD, and our implementation of the algorithm is based on these works.

Despite the increased simulated length and time scales that DPD method permits, calculation of explicit particle forces and velocities is computationally very expensive, and so simulations are typically limited to milliseconds of simulated time and volumes in the order of 0.1 cubic micrometres. In order to simulate the formation of DMPC vesicles, an implementation of DPD was created using the general purpose GPU CUDA programming environment from Nvidia, producing a 50x speedup over the single

processor implementation using an Nvidia Tesla C1060 card.

Vesicles can be formed via a variety of different methods, including microfluidics (Tan et al. 2006), centrifugation (Noireaux and Libchaber 2004), sonication and spontaneous formation. Regardless of the route to formation, all vesicles are composed of amphiphiles which have a hydrophobic section which does not dissolve in water and a hydrophilic section which is polar. In the presence of a polar solvent such as water, the hydrophobic sections of the molecules move together such that the disruption to the structure of the solvent is minimized. This hydrophobic effect is the cause of spontaneous formation of micelles, vesicles and bilayers.

Clearly there is a large difference in the time and length scales in which the BRN are normally simulated and the timescales which can be captured using the DPD method. However, the use of the DPD method has some clear advantages over other less detailed techniques. Firstly, the vesicle container and the emergent dynamics of the system are a result of the application of simple rules, e.g. the vesicle does not form due to any prespecified design, but as a result of minimization of configurational energy of the lipids, just as real vesicles do. If the reaction rates of BRN models can be scaled so that the processes occur within timescales that can be simulated in DPD, then this allows an exploration, at least in a qualitative sense, of systems where the BRN may produce proteins which affect the membrane, either by production of proteins that have hydrophobic moeties that could embed within the membrane (such as α-hemolysin) or by producing enzymes that catalyse the formation of other lipids (which may form domains in the vesicle membrane, eventually leading to fission). Also, as every particle in the system has an explicit position, and the system is not assumed to be mixing, unlike what occurs with the stochastic simulation algorithm (see next subsection), concentration gradients can arise and are captured within the model.

Moreover, as suggested in Cronin et al. (2006), both the P systems specification of a model and its execution through DPD adhere to the abstraction that programmable living matter can be engineered through clearly identifying the compartment (C) that delimits the self from non-self, information (I) storage and processing that helps guide the manufacturing of the compartment's building blocks and the orchestration of metabolism (M) processes as the arbiters of energy and waste management. That is, neither the P system nor the DPD simulations require that C, I, M be implemented in the way biology does but can indeed, follow a more chemical (rather than biological) route for liposome logic (Pasparakis et al. 2009a).

Figure 2 shows the formation of a vesicle from model DMPC amphiphiles in DPD. The dynamics of formation are as follows: the amphiphiles initially aggregate into small micelles, which then aggregate into larger micelles. Once the micelles reach a critical size, they become oblate (flattened) patches of bilayer membrane. If the bilayer membrane is large enough, then the membrane will begin to fold inwards into a bowl shape, which continues to curve until fusing at the top to form the spherical vesicle.

Stochastic simulation algorithm

Simulations of cellular logic systems for longer time scales were performed with the multicompartment stochastic simulator (MCSS) toolkit (Romero-Campero et al. 2008, 2009). This toolkit has at its core an optimised implementation of the Gillespie SSA, and supports simulation of stochastic P system models specified in an XML format. Stochastic discrete simulation techniques for biological systems have a number of advantages over ODEs at the cellular scale. Firstly, as ODEs are continuous and the concentrations of chemical species within cell volumes can be very small, integration of the ODEs could result in concentrations which represent fractions of a molecule. Secondly, as ODEs represent the dynamics of concentrations of molecules rather than individual molecules themselves, they do not capture the stochastic nature of the cellular volume and thirdly ODE models are typically more

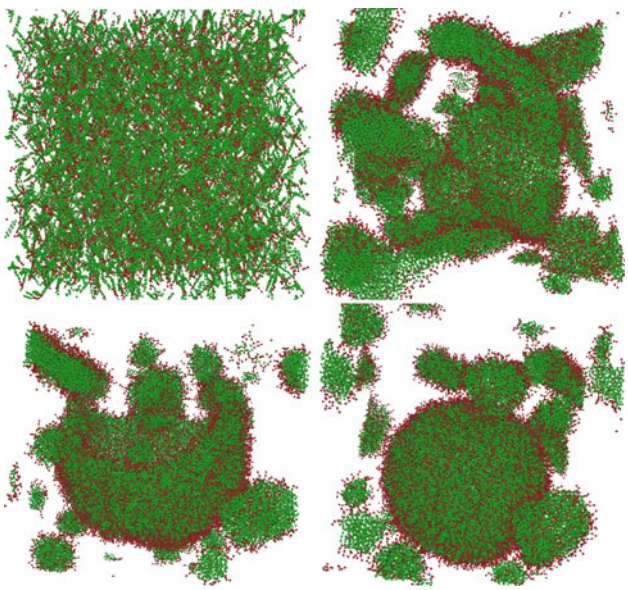


Fig. 2 The figure shows the process of vesicle formation from the initial state of the system were amphiphiles are distributed randomly in solvent (*top left*, solvent not shown). The amphiphiles are pushed together into micelles (*top right*) which in turn join together to form large planar bilayers (*bottom left*), these bilayers then begin to curl at the edges and fold over into a spherical vesicle (*bottom right*)

difficult to create and understand in comparison with executable (Fisher and Henzinger 2007) or algorithmic (Priami 2009) systems biology methodologies. Moreover, stochastic models specified in, e.g., P systems are more amenable to formal computational analysis such as model checking.

Model checking

Model checking is a well established formal method for analysing the behaviour of various systems. It normally requires a computational model of the system, provided as a high-level formalism (such as a Petri net, process algebra or P system), and a set of properties of the same system, expressed usually in temporal logic (LTL or CTL) (Kwiatkowska et al. 2009). A computational model associated to a system may consist of distinct parts, modules in the case of the P system formalism (see section "P system specification of liposomelogic models"), each one with a complex behaviour and generating many states. The model allows to test and verify certain hypotheses by executing the model and comparing the outcome with experimental data (Fisher and Henzinger 2007). Knowing that some systems are non-deterministic or probabilistic, the conclusions obtained are just limited to the number of executions performed. In order to ascertain more general properties, model checking techniques are employed. These properties can be validated for the entire system or for some components of it.

In probabilistic model checking, which will be used in this paper, the models are extended with quantitative information regarding the likelihood that some events will occur and the time they do so (Kwiatkowska et al. 2009). The models referred to in this paper are *continuous-time Markov chains* (CTMCs), where rates of negative exponential distributions are assigned. The properties are still expressed in temporal logic, but they show now some quantitative aspects. So, rather than verifying that for the NOT gate (see section "Liposome logic in DPD") "the protein output always eventually reaches a certain level" we may check "what is the probability that the protein output eventually reaches a certain level". More than this, using rewards we can ask questions like "what is the maximum protein output of the NOT gate". Such questions will be formulated in a specific temporal logic called continuous stochastic logic (CSL) (Kwiatkowska et al. 2009).

Model checking is very effective in verifying certain hypotheses regarding the system when more than one execution is possible or incomplete data is available. In this case through the new characteristics revealed, the model checking approach may suggest new experiment to confirm or reject hypotheses (Fisher and Henzinger 2007).

P system specification of liposome logic models

In this section we describe the P sytems specifications for liposome logic circuits. These specifications are the first step in the proposed methodological pipeline shown in Fig. 1.

A P system specification for the repressilator

Logic gates in cellular computing are constructed from networks of gene regulation in prokaryotic genomes. In prokaryotes, genes are arranged into operons, sequences of DNA containing a promoter region which is recognised by RNA polymerase enzymes, an operator region which is recognised by gene transcription factors, and one or more gene sequences (see Fig. 3).

The liposome logic simulations in this paper are based on the repressilator reported by Elowitz and Leibler (2000). The repressilator is a ring oscillator built from three genes. Figure 4 shows a schematic diagram of the repressilator network. The system includes three different genes, LacI, λcI and TetR, with the protein expressed from each gene acting as a repressor which binds to the promoter of the next gene, and reduces the rate of transcription.

The authors present a stochastic model of the repressilator, in which all three genes and promoters have identical properties in terms of the rates of binding etc. The repression of the gene is represented by a cooperative binding of the repressor protein to the gene promoter, (reactions 11, 12):

$$G + R \xrightarrow{1\,\mathrm{nm^{-1}\,s^{-1}}} GR \tag{11}$$

$$GR + R \xrightarrow{1\,\mathrm{nm^{-1}\,s^{-1}}} GRR \tag{12}$$

where G is the NOT gate promoter and gene, R is the repressor protein which binds to the gene operator and represses transcription of the gene, M is transcribed mRNA from the gene G, and O is the expressed protein from G, translated from M.

The repressor proteins also decomplex from the gene promoter, and these are modelled with reactions 13 and 14. It should be noted that the rate of decomplexation when both repressor proteins are bound to the sequence is greatly decreased when compared with the rate when only a single protein is bound.

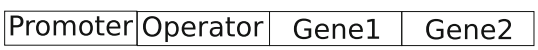


Fig. 3 The operon in prokaryote genomes, the promoter region is recognised by RNA polymerase, which binds to the promoter to initial transcription. The operator is recognised by transcription factor proteins which alter the rate of gene expression, the operator may control the expression of multiple genes

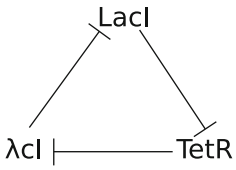


Fig. 4 The repressilator, the system is composed of three different genes, LacI, λ *cI* and TetR. The protein expressed from each gene inhibits the next, so for example the LacI proteins inhibit TetR expression, and TetR proteins inhibit λ *cI* expression

$$GR \xrightarrow{224\,\mathrm{s}^{-1}} G + R \tag{13}$$

$$GRR \xrightarrow{9\,\mathrm{s}^{-1}} GR + R \tag{14}$$

Reactions 15–18 represent transcription and translation. Transcription when the gene promoter is unrepressed occurs 1,000 times more frequently than when the gene is repressed.

$$G \xrightarrow{0.5\,\mathrm{s}^{-1}} G + M \tag{15}$$

$$M \xrightarrow{0.167\,\mathrm{s}^{-1}} M + O \tag{16}$$

$$GR \xrightarrow{5*10^{-4}\,\mathrm{s}^{-1}} GR + M \tag{17}$$

$$GRR \xrightarrow{5*10^{-4}\,\mathrm{s}^{-1}} GRR + M \tag{18}$$

mRNA and protein degradation occurs with a half-life of 120 and 600 s, respectively (reactions 19, 20).

$$O \xrightarrow{0.0012\,\mathrm{s}^{-1}} \tag{19}$$

$$M \xrightarrow{0.0058\,\mathrm{s}^{-1}} \tag{20}$$

These reactions specify a stochastic model of the behaviour of one gene in the repressilator. If the repressing protein is considered as the input to the system, and the expressed protein the output, then the behaviour of the model mimics that of a NOT logic gate, which outputs a high signal when the input is low, and a low signal when the input is high. The gene operon has two operator regions. A repressor protein can then bind to these regions and repress the gene. When only one operator is occupied by a repressor protein, the repressor is more likely to decomplex from the gene than when both promoters are occupied, and it is this cooperative binding which causes a "switch-like" transition between the high and low output states of the logic gate (Amos 2004), as a certain threshold of repressor concentration must be reached within the cell volume before both operators become occupied. The effect can be magnified by increasing the number of operators which cooperatively bind repressors, or by using oligomer proteins which must bind together before being able to bind to the gene. As we are interested in the modular assembly of variable depth logic circuits, we define a P system module (see formal definition in section "P systems as a specification framework") that, by using the repressilator circuitry, encodes a NOT gate module:

$$NOTGate(\{c1, c2, c3, c4, c5, c6, c7, c8, c9, c10\}) = \left\{ \begin{array}{l} [G+R] \xrightarrow{c1} [GR],\ [GR+R] \xrightarrow{c2} [GRR], \\ [GR] \xrightarrow{c3} [G+R],\ [GRR] \xrightarrow{c4} [GR+R], \\ [GR] \xrightarrow{c5} [GR+M],\ [G] \xrightarrow{c6} [G+M], \\ [GRR] \xrightarrow{c7} [GRR+M],\ [M] \xrightarrow{c8} [M+O], \\ [M] \xrightarrow{c9} [],\ [O] \xrightarrow{c10} [] \end{array} \right\}$$

The module's variables $\{R, G, M, O\}$, including the continuous ones $\{c_1, ..., c_{10}\}$, can be instantiated with different promoters, genes, proteins and kinetic constants as to represent specific systems. Also note that the square brackets indicate that the reactions take place inside a specific (proto) membrane or compartment. Indeed to simplify the notation we have taken out the compartment name variables as we use only one compartment in this study.

To construct the P system module representing the repressilator, we start by deriving from the *NOTGate* module a specific instantiation named *NG*, as to avoid specifying the stochastic rate constants each time (which are the same unless otherwise specified):

$$NG(\{R, G, M, O\}) = \left\{ \begin{array}{l} NOTGate(\{R, G, M, O\}, \\ \{1, 1, 224, 9, 5*10^{-4}, 0.5, 5*10^{-4}, 0.167, \\ 0.0058, 0.0012\}) \end{array} \right\}$$

Three or more *NG* modules can be connected together in sequence, with the output of the last connected as the input of the first gate, to produce a ring oscillator.

Ring oscillators made from N gates can be constructed as follows, for any odd integer N greater than or equal to three:

$$RON(\{G_1, \ldots, G_N, M_1, \ldots, M_N, O_1, \ldots, O_N\}) = \left\{ \begin{array}{l} NG(O_N, G_1, M_1, O_1), \\ NG(O_1, G_2, M_2, O_2), \\ \ldots \\ NG(O_{N-1}, G_N, M_N, O_N) \end{array} \right\}$$

Therefore when $N = 3$ the original Repressilator model, named RO3, is reproduced.

A NAND gate

By creating two copies of the same gene, with different promoter regions, a NAND gate can be created (Fig. 6).

The NAND gate is defined by the following module, note that the gene, mRNA and output protein are the same for both *NG* modules, but the input repressor is different (R1 for one gene and R2 for the other).

$$NAND(\{R1, R2, G1, M1, O1\} = \left\{ \begin{matrix} NG(\{R1, G1, M1, O1\}) \\ NG(\{R2, G1, M1, O1\}) \end{matrix} \right\}$$

It should be noted that constructing a NAND gate in this way produces two distinct output levels when the gate output is high, in the first case when neither input to the gate is present, both genes are transcribed. However, when the gate is presented with a single input one gene is repressed and the gate output, whilst still representing a logic value of True or high, produces roughly half the amount of protein than it does when no input is present.

A set-reset latch

Two NAND gates can then be connected to create a Set-Reset Latch (Fig. 7), the output of each gate is connected to the input of the other, and the state of the latch can be switched by holding the remaining set or reset inputs high for a short period. The Latch acts as a simple one bit memory which can be set or reset by expressing the appropriate protein that represses the gene of the relevant NAND gate. The Latch module is built from two NAND gates

$$SR - Latch(\{R1, R2, G1, G2, O1, O2\}) = \left\{ \begin{matrix} NAND(\{R1, O2, G1, M1, O1\}) \\ NAND(\{R2, O1, G2, M2, O2\}) \end{matrix} \right\}$$

A D type flip-flop

Latches can then be connected to NAND and NOT gates to construct a D type flip-flop, as shown in Fig. 8.

A D Flip-Flop takes a data input, indicating whether the flip-flop should be set or reset, and a clock input. The output of the flip-flop will be the last active input when the clock was still high, and so the output is fixed when the clock input goes from high to low. Each flip-flop stores a single bit, and can be coupled in sequence to make larger memories. The D-Flip Flop can also be converted to a

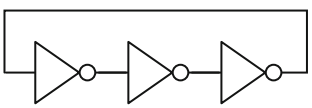


Fig. 5 Ring oscillator built from three not gates

Fig. 6 A Nand Gate built from two NOT gates. The inputs to the gate are two repressor proteins labelled X and Y, and the output protein is labelled Z

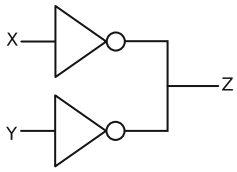
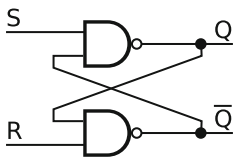


Fig. 7 Set Reset Latch constructed from two NAND gates



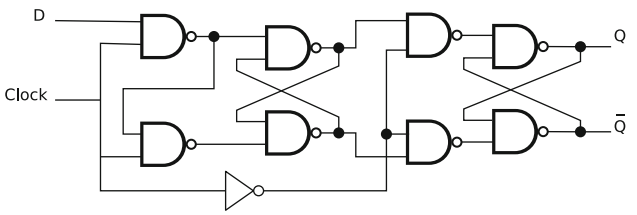


Fig. 8 The D Flip-Flop built from two latches, four NAND gates and a NOT gate

toggle flip-flop by connecting the $\bar{Q}$ output to the D input. Therefore each time a clock pulse occurs, the gate will toggle between the Set and Reset states. The module configuration for the D flip flop is shown below:

$$DFlipFlop(\{G1, \ldots, G9, DInput, ClockInput, M1, \ldots, M8, O1, O2\}) = \left\{ \begin{matrix} NG(\{ClockInput, G9, M9, O9\}) \\ NAND(\{Dinput, ClockInput, G5, M5, O5\}) \\ NAND(\{O5, ClockInput, G6, M6, O6\}) \\ SR - Latch(\{O5, O6, G1, G2, O1, O2\}) \\ NAND(\{O1, O9, G7, M7, O7\}) \\ NAND(\{O9, O2, G8, M8, O8\}) \\ SR - Latch(\{O7, O8, G3, G4, O3, O4\}) \end{matrix} \right\}$$

A 3 bit ripple counter

A toggle flip-flop can in turn be used to build ripple counters, simple counters in which the the Q output of one flip flop is connected to the clock input of the next flip flop. Figure 9 shows a diagram of three flip flops connected together to form a 3 bit ripple counter, with a 5 NOT gate ring oscillator acting as the system clock, when the clock is high, the state of the first flip-flop is toggled to produce a high output, connected to the clock input of the next flip-flop, which is then also toggled. The first flip-flop remains in the logic high state until the next clock pulse, upon which it toggles to the low state, causing the state of the

second flip-flop to be fixed high. As the output of each flip flop toggles at half the rate of it's clock input, the output of the first flip flop is high for one clock cycle, and then low for one clock cycle. the second bit high for two cycles and low for two cycles, and the third bit high for four cycles and low for four cycles.

The counter module is constructed from the following modules:

$$
\begin{aligned}
&Counter - 3bit(\{G1,\ldots,G28,\\
&M1,\ldots,M35,O1,\ldots,O8\}) =\\
&\left\{\begin{array}{l}
DFlipFlop(\{G1,\ldots,G9,\\
ClockInput,O2,M1,\ldots,M9,O1,O2\})\\
DFlipFlop(\{G10,\ldots,G18,\\
O1,O4,M10,\ldots,M18,O3,O4\})\\
DFlipFlop(\{G19,\ldots,G27,\\
O3,O6,M19,\ldots,M27,O5,O6\})\\
5GateClock(\{G28,\ldots,G35,\\
O5,O8,M28,\ldots,M35,O7,O8\})
\end{array}\right\}
\end{aligned}
$$

Experimental results

The liposome logic circuits specified in the previous section were simulated under various different conditions using DPD, SSA or both (see Fig. 10) which shows a diagram of the relationship between the different modules and indicates which were simulated with DPD and which were simulated using SSA. The results of these simulations are presented below.

Dissipative particles dynamic results

Forming the vesicles

Our model amphiphiles in DPD are based on parameters from Kranenberg et al. (2004) in which the authors investigate a number of different coarse grainings of DMPC like amphiphiles. Figure 11 shows the structure of the model amphiphiles. Each amphiphile has two hydrophobic tails and a hydrophilic headgroup of three particles. Angle forces maintain a rigidity in the tail chains with a preferred angle of 180° and a force strength of $6K_bT$, and

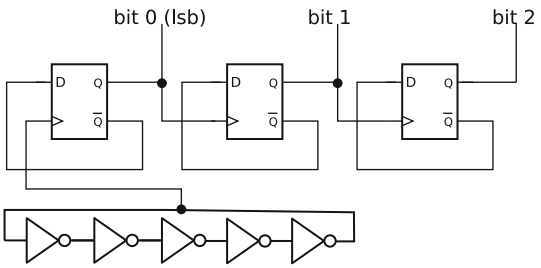


Fig. 9 A 3 bit counter connected to a 5 gate ring oscillator

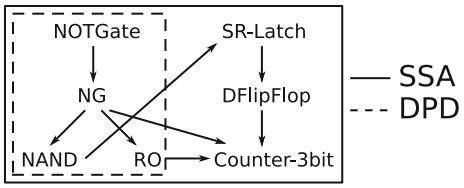


Fig. 10 The diagram indicates which modules were simulated using which simulation technique (e.g. DPD or SSA), and the relationship between the different models simulated using the pipeline, an *arrow* pointing from one module to another indicates that the module at the *arrow's source* is used by the other module

hold the tail particles apart with a preferred angle of 90° and force strength of $3K_bT$.

The alpha parameter matrix for the hydrophobic, hydrophilic and water particles is shown in Table 1.

The physical length and timescales in the DPD simulation can be ascertained by performing the mapping described by Groot and Rabone. The unit length in the simulation is set to the force interaction radius, and all beads have the same mass in the simulation and occupy the same volume, equivalent to three water molecules ($\sim$90 Å^3). Since the unit cube density parameter ρ is set to 3, meaning on average there will be three particles in each unit cube, the side length of each cube, and therefore the physical interpretation of the unit length is $\sqrt{[3]}270\,\text{Å}^3 = 6.4633$ Å. The physical interpretation of the time step is based on calculation of the self-diffusion constant of the simulated DPD fluid, which is then mapped to the same value for water at room temperature resulting in a time unit length τ of $\sim$88 ps. Typical simulated times are 2,500 τ (220 ns) to 100,000 τ (8.8 μs) in volumes of $\sim$34 nm^3.

All DPD simulations in this work were performed with $\sigma = 3, \gamma = 4.5$ and $\rho = 3$ and the Groot Warren intergrator was used with $\lambda = 0.65$ and the timestep length $dt = 0.05$

For the vesicle computation simulations in this work a vesicle was formed which was composed of 5,825 DMPC molecules, encapsulating a core of 58,550 solvent particles.

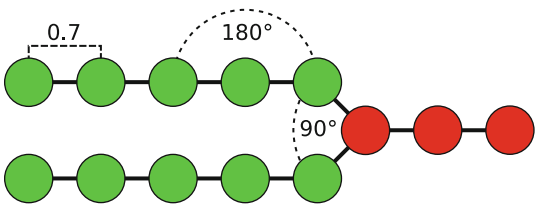


Fig. 11 Schematic diagram of the DMPC amphiphile. The particles in the hydrophobic tail chains of the amphiphile are held together by Hookean spring forces with a preferred distance of 0.7 units, and a bond angle force maintaining the angle between bonds are 180°. The tail chains are held in close proximity by a bond angle force with a preferred angle of 90°. The Hydrophilic head particles (righthand side) also have a preferred bond distance of 0.7 units, but no bond angle force is imposed between head particles

Table 1 The alpha parameters type matrix for the DMPC polymer

	Water	Head	Tail
Water	78	75.8	110
Head	75.8	78	110
Tail	110	110	78

A value of 78 produces the correct compressibility for water at room temperature, larger values indicate a repulsion between particle types

The vesicle was then placed within a simulation space of $50r_c^3$, and the volume which was external to the vesicle membrane was filled with solvent particles such that the correct density (3 particles per r_c^3) was achieved. For each NOT gate in the module a solvent particle within the vesicle core was chosen at random and replaced with a particle representing the gene.

Liposome logic in DPD

The NOT gate model can be implemented in DPD as a set of first and second order reactions. However, the rates in the original model are specified over timescales of the order of seconds, with the dynamics of the system only observable over minutes/hours. In order to observe the model dynamics within DPD timescales, the reaction rates are rescaled to occur within the DPD timescale. For the first order reactions this is a straightforward process, as long as all the first order reaction rates are scaled equivalently. However, for the second order reactions, the situation is more complex, as the stochastic rate constant is the rate at which a reactant pair will collide and react. This rate is determined by the physical properties of the system (e.g. temperature, reactant mass etc.) and the probability that the colliding particles will be in the correct orientation for the reaction to occur. Therefore to scale the second order reactions correctly, it is necessary to determine the rate at which particle pairs collide within the vesicle. The collision rate was determined directly from simulation (data not shown) and was found to be 0.0002 collisions per DPD time unit. Thus the NOT gate model is instantiated as

$$NOTGate(\{P,R,G,M,O\},\{1\tau^{-1},1\tau^{-1},1\tau^{-1},0.0402\tau^{-1},5*10^{-4}\tau^{-1},0.5\tau^{-1},5*10^{-4}\tau^{-1},0.167\tau^{-1},0.0012\tau^{-1},0.0058\tau^{-1}\})$$

For the other reactions, the rate constants were rescaled by changing the unit of time from seconds to DPD time units (τ). The consequence of this for the second order reactions representing binding of repressor to gene, is that the rate of collisions is reduced as the number of collisions per time unit is much smaller in DPD. Therefore the actual reaction rate in DPD is shown below.

$$[R][G]*0.0002 \quad (21)$$

where $[R]$ and $[G]$ indicates the number of repressors and genes in the simulation respectively. The consequence of this is that the rate at which repressors bind to the gene is reduced by a factor of 5,000 in comparison with the other scaled rates. Note that the rates of the decomplexation rules which represent the repressor protein unbinding from the gene have been rescaled to $1\tau^{-1}$ and $0.0402\tau^{-1}$ (the rates in the original model were 224 and 9 s^{-1}, respectively), as the original rates were too fast to be represented in the rescaling, and so were reduced by a factor of 224.

The effect of this alteration somewhat mitigates the reduction of the complexation rate, and the reduction of the binding rate relative to the adjusted decomplexation rate is reduced by a factor of 22.3. The effect of these changes will be that the repression of the gene occurs more slowly, and both the decomplexation and complexation reactions occur more slowly in comparison to the first order reactions, but the qualitative structure of the model should be maintained.

The effect of encapsulation on logic gate dynamics

The first experiment involved placing the NOT gate inside the vesicle membrane, and comparing the results of simulation in which the NOT gate was not encapsulated within the membrane (e.g. allowed to diffuse freely within the full $50r_c$ volume.) to show the effect that the encapsulation has on the second order reaction rates.

The result of simulating the NOT gate within a vesicle with no input signal connected to the gate, for 5,000 τ is shown in Fig. 12. The NOT gate gene is expressed when the gate has no input and the amount of protein rises until an equilibrium between expression and degradation is reached, at an output level of $\sim$10,000 proteins.

Figure 13 shows the results of simulating the NOT gate with a high input signal (i.e. a gene producing the NOT gate input repressor protein was added to the system) the gate and input gene particles were encapsulated within a vesicle and the number of expressed proteins recorded each time unit to produce a time series (continuous black line). Simulations were also performed in which the NOT gate and input gene particles were not placed within the vesicle, but instead were able to diffuse freely within the entire simulated volume (dotted black line).

At the start of the simulation the NOT gate gene is initially expressed, until the amount of repressor (which is concurrently expressed from the input gene) reaches the threshold required to fully repress the NOT gate gene, causing the amount of output protein to drop as the protein

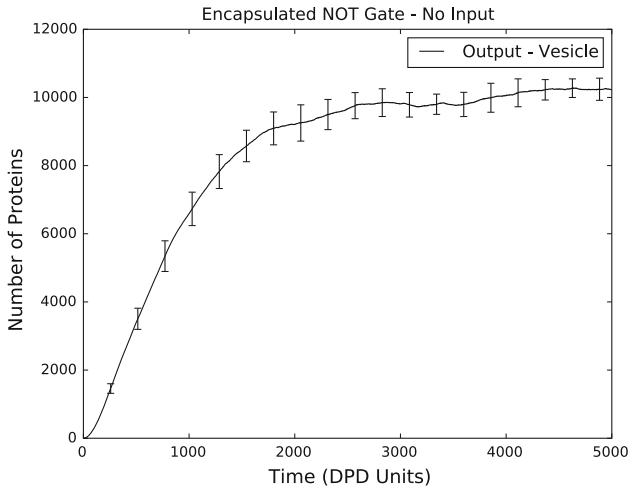


Fig. 12 Time series of the protein output from a gene representing a NOT gate, encapsulated within a vesicle showing the mean number of proteins present in the volume over the course of the simulation, with the *error bars* showing the estimated standard error

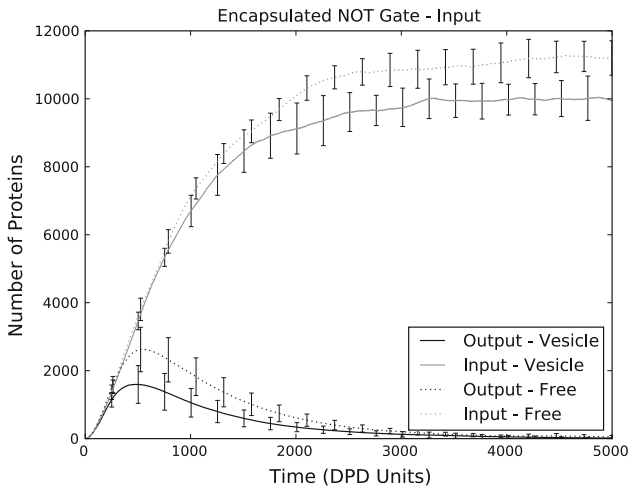


Fig. 13 Output protein levels for simulation of NOT gate placed within a vesicle and diffusing freely in the simulated volume, averaged over 10 runs (*error bars* indicate estimated standard error). The continuous *grey and black lines* show the time series for the input and output proteins for the NOT gate placed within a vesicle. The *dotted grey and black lines* show the input and output proteins of the NOT gate with an input present when system is not encapsulated within a vesicle

and mRNA degrade and are not replenished, so that typically less than 50 proteins remained by the end of the simulation.

The dotted line in Fig. 13 shows the result of simulating the high input model for the case were the input and NOT gate genes were not encapsulated within the vesicle. The mean number of proteins expressed at the peak of expression was greater by $\sim$1,000 particles when compared to the output time series for the encapsulated gate, and the peak was reached later in the simulation, indicating the transition of the NOT gate (module NG) from the high to low output state occurred more slowly. The mean time series for the input repressor protein in the encapsulated and unencapsulated NOT gate simulations are shown by the grey continuous and grey dotted lines, respectively. Once the repressor protein levels have reached an equilibrium, there is a difference of over $\sim$1,000 proteins between the encapsulated and unencapsulated equilibrium value. Correspondingly there is a difference between the levels of outputted mRNA when the NOT gate was and was not encapsulated. Figure 14 shows that the mean mRNA output when the gate was not encapsulated peaked at slightly less than 60 molecules, whereas the mRNA output for the encapsulated gate peaked at around 43 molecules. As the collisions occur more frequently in the vesicle volume, the gene becomes fully repressed more quickly, and so the peak level of mRNA output is reduced.

Figure 15 shows the output protein levels from the NAND gate model built from two NOT gates (model NAND in section "A NAND gate"). Four time series are shown, one for each possible combination of signal inputs to the gate. The continuous line shows the output for the gate when both of the genes for the input signals (labelled X and Y in the figure) were present, the output level rises initially until enough of the input proteins is present to fully repress both genes in the NAND gate, at which point the output signal drops to zero. The dotted line shows the case where there was no input signal to the NAND gate, the level of output protein reaches an equilibrium value of around 17,500 proteins. The dashed and dot-dashed lines show the case where one of the input signal genes was

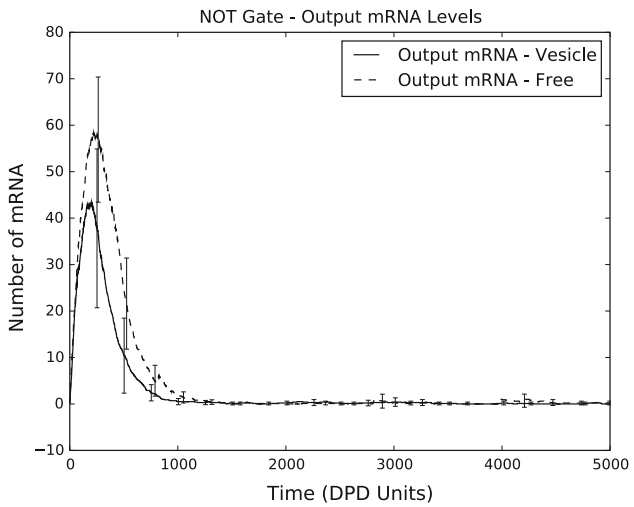


Fig. 14 The time series for the transcribed mRNA from the NOT gate gene, averaged over 10 runs. The *continuous black line* shows the output level of transcribed mRNA when the NOT gate was encapsulated within the vesicle, whereas the *dashed line* shows the mRNA time series when the NOT gate was diffusing freely throughout the entire volume

present, in both of these cases, one of two NOT gates which make up the NAND is repressed, and so the output protein levels reaches an equilibrium value of around 8,000 proteins, which is roughly half the output level when neither input was present.

The encapsulated repressilator

The second experiment involved the encapsulation of the repressilator (module RO3) within the core of a self assembled vesicle. Time series from simulations of the increased decomplexation rate repressilator model, encapsulated within a vesicle are shown in Fig. 16, the expressed protein levels for each of the three NOT gates in the repressilator (shown for for three runs of the simulation) can be seen to oscillate. The increased decomplexation rate of the repressors from the gene when compared to the original repressilator model means that the period of oscillation is not quite long enough to allow all of the transcription factor to degrade, and so the amount of each transcription factor drops to around 1,000 proteins.. Figure 17 shows the results of simulating the model where the decomplexation rates were only scaled, and not increased. The decomplexation of repressor from gene occurs less frequently in this model and so the oscillations have a longer period, allowing the transcription factors to degrade completely before the next cycle of the oscillation and the period of the oscillation is increased. Figure 18 shows the images from the inner volume of vesicle, with the particles representing the different output proteins from each of the three NOT gates given different colours, each of the images are captured at the point in the simulation were the respective protein is being expressed.

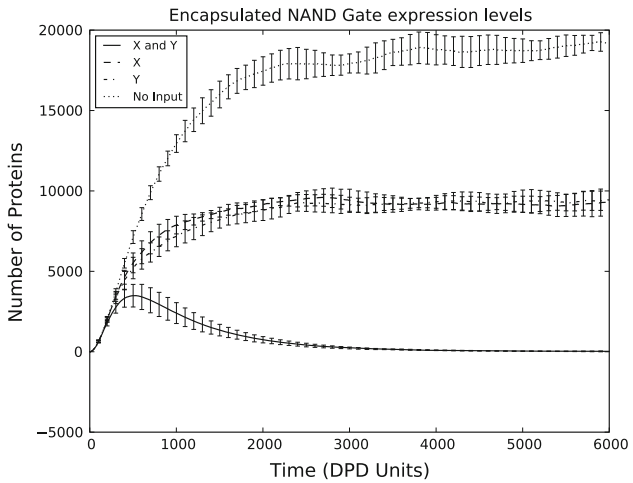


Fig. 15 The time series of protein output levels from the simulation of the NAND gate, the output of the gate is shown in response to 4 different combinations of inputs labelled X and Y

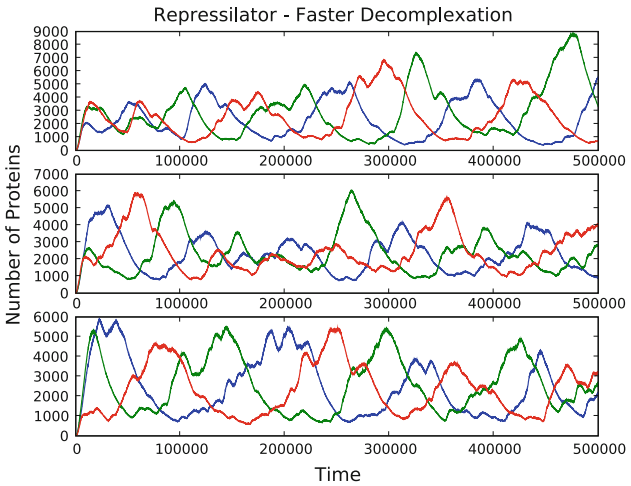


Fig. 16 The result of three simulations of the repressilator model with increased rate constants for the decomplexation of repressors from the promoter, each plot shows the time-series for the three NOT gate output proteins

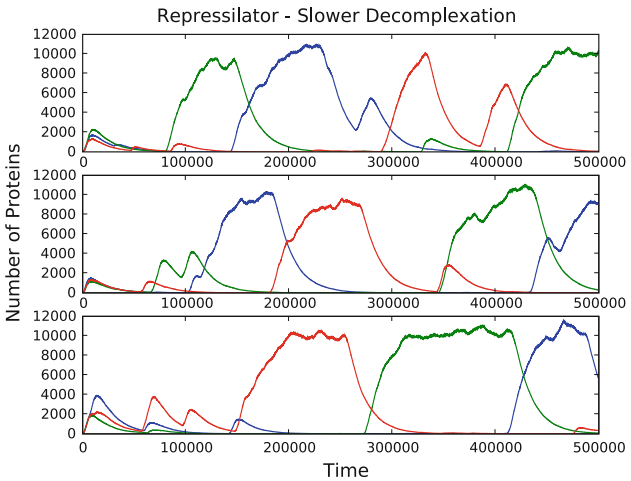


Fig. 17 The result of three simulations of the repressilator model within a vesicle in DPD, the parameters are rescaled versions of those from the elowitz model such that the dynamics can be examined within DPD timescales, each plot shows the time-series for the three NOT gate output proteins

Immiscible repressors

The third vesicle computing experiment involved the same initial configuration as the previous repressilator experiment (module RO3), but the alpha parameters for the proteins were modified slightly to examine the case where the output protein is slightly hydrophobic, and also less miscible with other proteins. The effect of this should be to create three distinct protein phases, which may mean the dynamics of the repressilator will be altered due to the non-uniform concentrations of repressors. Table 2 shows the alpha parameter vector for each repressor protein in the system.

Table 2 α parameters for immiscible repressors

	Solvent	Gene	R1	R2	R3
Solvent	78	78	85	85	85
Gene	78	78	78	78	78
mRNA	78	78	78	78	78
R1	85	78	78	85	85
R2	85	78	85	78	85
R3	85	78	85	85	78

The results from simulations of the repressilator with increased α parameters between the repressor proteins expressed from each NOT gate gene are shown in Fig. 19. Because the transcription factors are now hydrophobic and do not mix with the solvent, the volume is no longer homogeneous, causing the dynamics of the repressilator to be altered. The period of the oscillations is no longer steady as the gene might not diffuse into an area that contains a high concentration of proteins that repress it. The repressor proteins also form distinct phases which tend to move towards the boundary between the vesicle membrane and the solvent, so that contact between hydrophobic repressor and solvent is minimised. The result of this movement was a bulging deformation of the normally spherical vesicle shape, this effect is shown in Fig. 20. Deformation of the membrane may be interesting to those working on the problem of causing vesicle fusion, as the deformation of the membrane will create areas of increased tension due to the elasticity of the membrane, which may increase the likelihood of fusion if two such vesicles were to come into close contact (Shillcock and Lipowsky 2005). This result also illustrates the sort of system dynamics that can be observed in DPD rather than in other less detailed simulation techniques.

Stochastic simulation algorithm results

If more complex logical circuits need to be simulated, or simulations for long length/timescales are required, we can abstract away the molecular and three dimensional detail of DPD and use instead a stochastic simulation algorithm to simulate deeper logic circuits that capture compartments' topologies but ignores their detailed geometries. The results of the SSA experiments are now described.

Oscillator frequency

We extended the Elowitz models with increasing numbers of NOT gates, to investigate whether increasing clock periods would match the theoretical estimates for silicon gates, and if there are limits to the number of gates which can be connected together in this way. The oscillator models were constructed from 5,7,9,11,21,31,41 and 51 gates modules (RO3, RO5, etc.) and simulation of each oscillator was performed for 2 days of simulated time.

The formula for calculating the frequency of a electronic ring oscillators built from any odd number of NOT gates is shown in Eq. 22:

$$\frac{1}{2nTp} \tag{22}$$

where n is the number of logic gates, and Tp is the propagation delay of each gate. We determine if this formula accurately calculates the frequency of the oscillators built from logic gates by calculating the propagation delay for the gates, and calculating the oscillator frequency from the output data, and then compare with the value from the formula.

The propogation delay of the NOT gate was determined to be 766.46 $\pm$ 1.95 s, by simulating an NOT gate with the initial number of input repressor proteins set to the mean equilibrium output for the gate (11,983 $\pm$ 47.29), with a constant input of repressor protein also present. The propogation delay was determined as the mean number of seconds for the NOT gate output to fall to half of it's original level.

The results from simulation of oscillators with 1,3, 5,7,9,11,21,31,41 and 51 NOT gates are shown in Fig. 21,

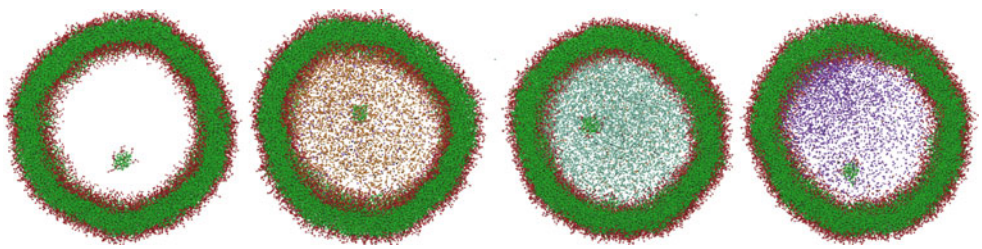


Fig. 18 Snapshots from a simulation of the repressilator within a vesicle were taken every 2,500 τ, a small micelle was trapped within the vesicle when it formed and is visible in each image. The vesicle was sliced so that the inner volume is visible (note that solvent particles are not shown). The images show (from *left* to *right*) the initial vesicle condition, high concentrations of the output protein expressed from the first NOT gate, the second NOT gate, and the third NOT gate (note the concentration gradient visible in the last image)

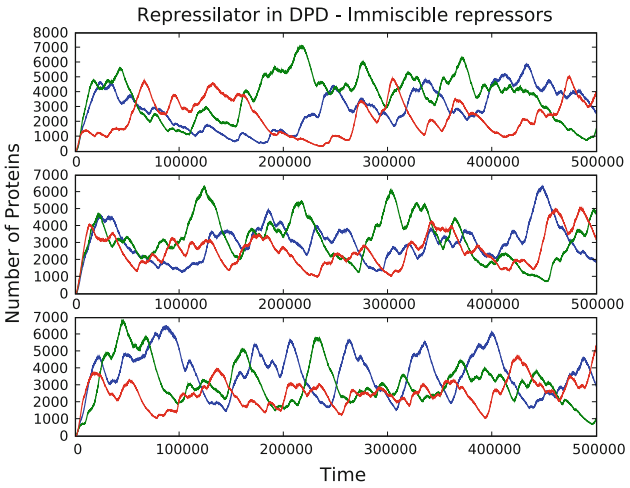


Fig. 19 The result of three simulations of the repressilator model with hydrophobic repressor proteins. In each run, the time series indicate the expression levels of the different NOT gate output proteins

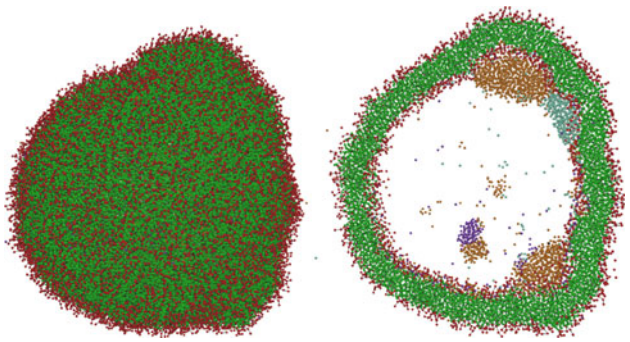


Fig. 20 Hydrophobic repressor domains form within the vesicle, and deform the membrane: The image on the *left* shows the surface of a vesicle which has been deformed by the formation of phases within it. The image on the right shows a slice through the same vesicle, the output proteins have formed phases in the vesicle core and are pressing against the membrane (the darker and lighter coloured particles between the vesicle core and the membrane)

the figure shows that the relationship between the number of NOT gates and oscillator frequency is similar to Eq. 22 until the number of NOT gates is 11 although the frequency is reduced by between 0.3 and 1 microhertz. For oscillators with more than 11 NOT gates the standard deviation of the frequency is increased, and the shape of the curve no longer follows the predictions from Eq. 22. Looking at the data for each individual run showed that for 21 NOT gates and above, the oscillator was decreasingly likely to settle into a stable oscillation.

Table 3 shows the number of oscillators in the 10 runs which were unstable for the different numbers of NOT gates.

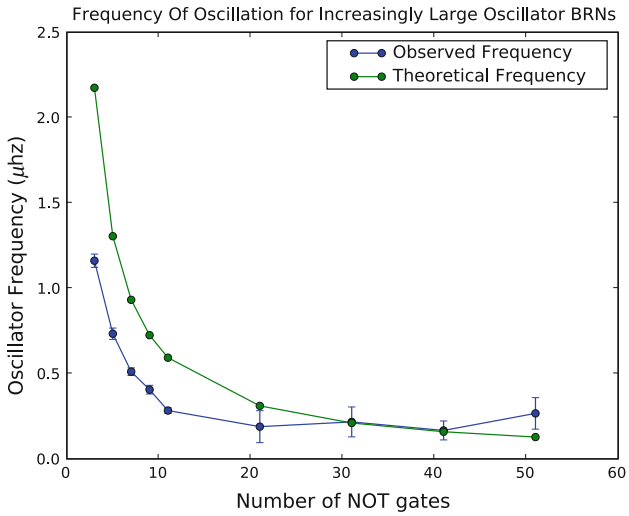


Fig. 21 The figure shows the frequency of oscillation in μhz for oscillators constructed from 1,3,5,7,9,11,21,31,41 and 51 NOT gates. The observed frequency line shows the the oscillator frequencies observed in simulation, each point is the mean frequency of 10 simulations of the oscillator, the *error bars* show the standard deviation. The theoretical frequency line shows the frequency calculated from equation 22 for the different numbers of gates

Table 3 The number of unstable oscillations observed during 10 runs of oscillators composed of different numbers of NOT gates

Number of NOT gates	Unstable count
3	0
5	0
7	0
9	0
11	0
21	1
31	5
41	7
51	9

The effect of RNAP and ribosomes

The behaviour of the RO51 oscillator was also examined in a more detailed model where the transcription and translation explicitly included polymerase and ribosomes, The number of polymersomes and ribosomes were at realistic levels for a bacterial or large vesicle volume.

When RNAP and ribosome interactions are included explicitly in the model, the effect is that there is a global constraint on the rate of transcription and translation. Figure 22 shows the levels of free RNAP and ribosomes for a simulation of the 51 gate oscillator model modified to include RNAP and ribosome interactions explicitly, the model was initialised with 35 RNAP and 350 ribosomes. The result shows that when the oscillator is functioning the average number of RNAP in use is slightly less than one,

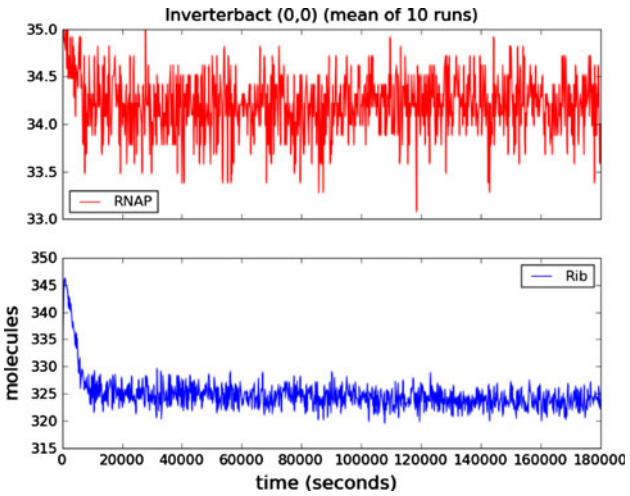


Fig. 22 Time series for free RNA polymerase (*RNAP*) and Ribosome (*Rib*) proteins in a simulation of the 51 gate oscillator model

and the average number of ribosomes in use is around 25. However, the inclusion of the RNAP and Ribosomes did not alter the transcription rate significantly.

The 3-bit ripple counter

The counter models were simulated in MCSS for simulated time periods of either 2 or 3 days, with the number of molecules of each chemical species recorded at every 3 min of simulated time to produce a time series for each chemical species in the simulation.

Figure 23 shows the time series for the simulation of the 3-bit counter with a 3 gate ring oscillator as the clock, and Fig. 24 shows the results of simulating the same counter with a 5 gate ring oscillator.

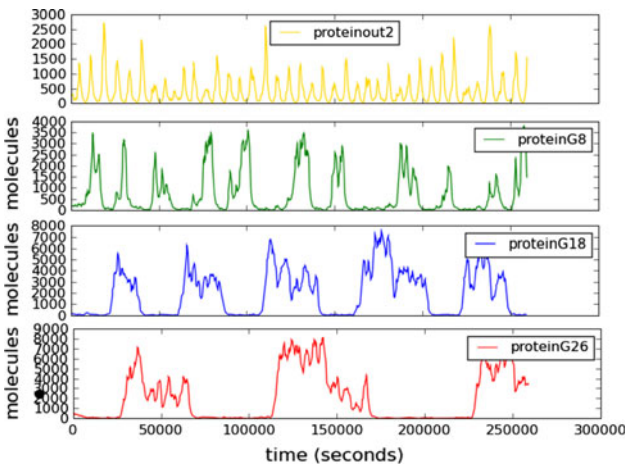


Fig. 23 Time series for 3-bit counter model with 3-gate clock as input, **proteinout2** is the clock signal, **proteinG8** is the output of the first bit of the counter, **proteinG18** the output of the second bit of the counter and **proteinG26** is the output of the third bit

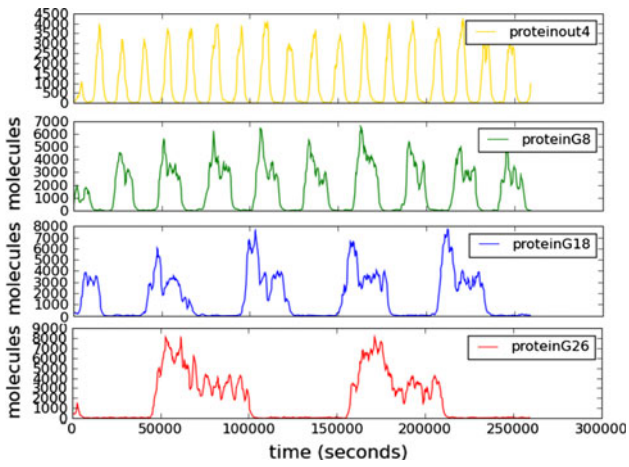


Fig. 24 Time series for 3-bit counter model with 5-gate clock as input, **proteinout4** is the clock signal, **proteinG8** is the output of the first counter bit, **proteinG18** the output of the second bit of the counter and **proteinG26** the output of the third bit

The time series show the output protein levels for each bit of the 3-bit counter. In the case of the counter connected to a 3-gate clock, it is likely that the propagation delay of the flip flops is greater than the time between clock pulses, and so the output of the first counter bit (proteinG18) does not always indicate that the flip flop was correctly toggled by the clock input. When the counter is connected to a lower frequency clock (constructed from 5 NOT gates), the dynamics of the output of the first counter bit have a much more consistent period and number of period of high output is roughly 1/2 the number of input clocks as expected. Figure 25 shows the clock input and first bit output overlayed for the 5 gate clock model.

The figures shows that there is a clear correspondence in each case between the high level of each bit and the triggering of the output of the next bit, the counter is therefore functioning as intended. Note that when the counter reaches its limit (7 in this case) it simply overflows and the counter starts from zero again.

Model checking

We focus our analysis on two of the simplest parts in our study, namely the NOT gate and the NAND gate, that are subsequently used to construct the rest of the models. In order to asses their perfomances we applied formal analysis on their dynamics using *simulative probabilistic model checking*. More specifically, the behaviour of our P system models were translated into CTMCs and then analysed using the PRISM probabilistic model checker (Kwiatkowska et al. 2002). Due to the complexity of the models under study the complete state space was not constructed,

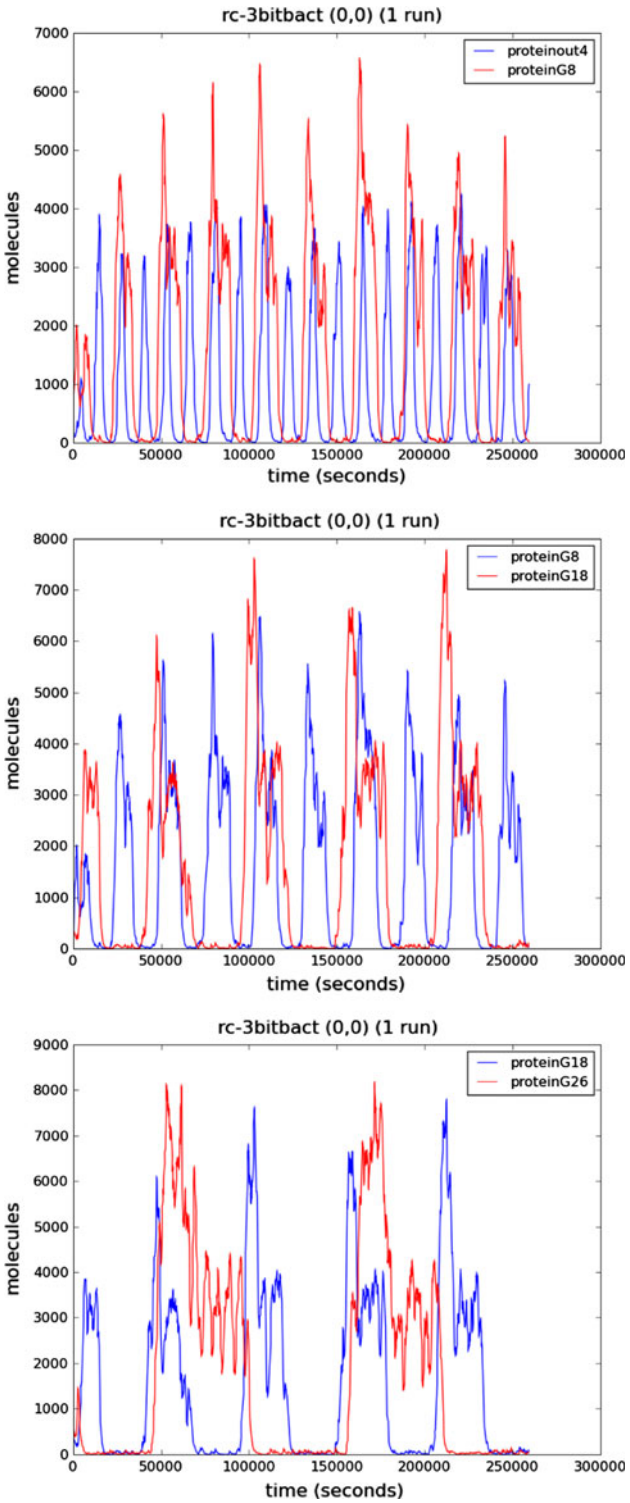


Fig. 25 Overlayed time series for protein output levels, the *top figure* shows the clock input level overlayed with the bit-0 output for the counter, the *middle figure* shows the bit-0 output overlayed with the bit-1 output, and the *bottom figure* shows the bit-1 output overlayed with the bit-2 output

but, instead, ensembles of multiple simulations or trajectories in the state space were generated and the corresponding properties, expressed in the temporal logic CSL (Kwiatkowska et al. 2002), were checked against them.

In the analysis that follows 1,000 simulations were used to produce an estimate $\hat{p}$ of the answer p to a query. This resulted in a *precision* of 0.1 with a *confidence* of 0.01 which determines the accuracy of the estimate according to the following formula.

$$P[\ |p - \hat{p}| > precision\ ] < confidence$$

NOT gate

In the case of our molecular NOT gate we studied the accuracy of its behaviour with respect to the general specification of a NOT gate and the speed of its reponse when provided with some input molecules.

Expected number of output proteins in the long run for different values of input proteins

We examine whether or not this basic building block behaves as expected. That is, in the presence of low values of input proteins, high levels of output proteins should be produced and viceversa, when high amounts of input proteins are provided, no output protein should be synthesized.

In order to investigate this, the following *instantaneous reward formula* was formulated and a reward corresponding to the number of output proteins was associated to each state in the corresponding continuous-time Markov chain.

$$\mathtt{R} = ?[\mathtt{I} = 6,000]$$

The property was analysed at the time instant I = 6,000 seconds. Figure 26 shows that for low numbers of CI proteins the number of output proteins in the long run is high. Whereas an increase in the number of input proteins produces a sharp decrease to zero in the number of output proteins. The transition from high to low output occurs at

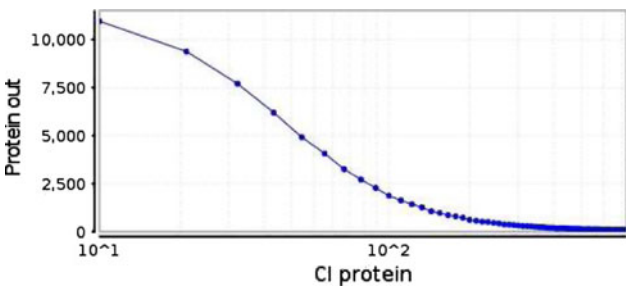


Fig. 26 Expected number of output proteins for different number of initial input proteins in the NOT gate (logarithmic scale)

around 150 input proteins. These results are in agreement with the general specification of a NOT gate.

Expected propagation time or response time

We analyse how fast our molecular device responds to its input by determining the time expected to reach half way between the initial and the final state once input proteins are introduced in the system. This property is normally termed *propagation time* or *response time*.

The following *reachability reward formula* was considered in order to investigate the propagation time of the NOT gate.

$\mathtt{R} = ?[\mathtt{F}\ \mathtt{proteinOut} < 5,000]$

This type of query accumulates, over a trajectory, the rewards associated with each state times the time spent in that state until a state fulfilling the corresponding formula is reached. Since we want to accumulate the time spent in each state over a given trajectory a reward equal to one is associated to each state in the corresponding CTMC.

The property whose reachability needs to be analysed is the output protein descending below the threshold of 5,000 molecules, which is half of the the initial number of output proteins, 10^4 molecules. In Fig. 27 we can observe that a low number of input proteins leads to a very slow response, whereas an increase in the number of input molecules produces a fast decay in the propagation time. Interestingly, our study shows the existence of a threshold for the input proteins around 150 for which any further increase does not produce an acceleration in the response.

From these two properties we can conclude that for our NOT gate there exists a threshold of around 150 input proteins. Below this number our molecular device produces a high number of output proteins. By contrast, if a number of input proteins above this threshold is provided to the system, then no output proteins are synthesised. Moreover, this threshold of 150 proteins provides the optimal input value with respect to the propagation time, as an increase in the input beyond this level does not produce a faster response.

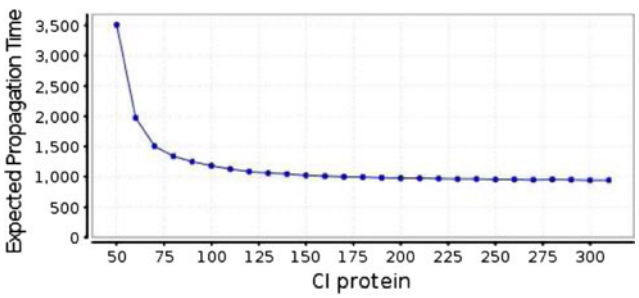


Fig. 27 Expected propagation time for the NOT gate with different number of initial input proteins

NAND gate

Similar to the previous case for the NOT gate we study properties that determine the accuracy of the behaviour of our genetic design when compare to the general specification of a NAND gate.

Expected behaviour of the NAND gate

In the presence of both inputs our molecular device should synthesise no output proteins whereas in any other case, that is, presence of only one input or absence of both inputs, output proteins should be detectable.

The following *instantaneous reward property* is used to determine the number of output proteins in the long run, time instant I = 6,000, for different values of the two input proteins.

$\mathtt{R} = ?[\mathtt{I} = 6,000]$

Note that since the NAND gate is a composition of two identical NOT gates with the same parameters as the one analysed above the threshold of 150 input molecules is also evident in the behaviour of this gate, Fig. 28. This determines four different regimes in the behaviour of the gate. When $INPUT_1$ and $INPUT_2$ are less than 150 the output is maximal. When $INPUT_1$ is less than 150 and $INPUT_2$ is greater than 150 (similarly when $INPUT_2$ is less than 150 and $INPUT_1$ is greater than 150) the output is produced at a half maximal level. Finally, no output proteins are synthesised when both $INPUT_1$ and $INPUT_2$ are greater than 150.

Probability of the absence of a detectable level of output proteins

In order to get a more detailed intuition of the behaviour of the NAND gate we estimated the probability of a non-detectable level of output proteins in the long run for different values of the two input proteins. The detectable level

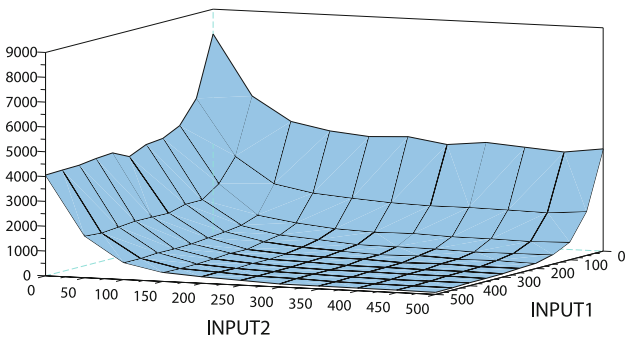


Fig. 28 Expected number of output proteins in the long run for the NAND gate with different numbers of input proteins

was fixed to 500 output proteins. For this we used the following *transient probability formula*.

$$\mathtt{P} = ?\,[\,\mathtt{true}\,\mathtt{U}[6,000,\ 6,000]\,\mathtt{proteinOut} < 500\,]$$

Figure 29 shows the sharp transition around the threshold of 150 input proteins from a detectable level of output proteins to an undetectable one.

Potential routes to a chemical implementation

The potential power of the vesicle computation method and the use of compartmentalisation in the DPD simulations offer intriguing possibilities within a chemical context. The "bottom-up" approach allows for many further molecular systems to be invoked than those currently used in biology, sophisticated though these already are. For example, logic gates have been constructed from a variety of non-biological systems and have used inputs/processes ranging from photoelectron transfer and fluorescence through to gel swelling and electrical signals (Asoh and Akashi 2009; de Silva and Uchiyama 2007; Gunnlaugsson et al. 2000; Yoshida and Yokobayashi 2007; Magri 2009; Pischel 2007). Abiotic small molecule systems generally rely for their logic processing on binding events such as host-guest interactions, which lead to a perturbation in the electronic or conformational state of the molecule, which in turn are converted to signals. Combinations of different inputs (e.g. pH, ion binding) on to molecules with more than one potential host-guest interaction or conformational change lead to multiple logic operations and functions such as Adders and Subtractors built from AND, XOR, INH and OR gates. Small molecule logic systems of this type can also be coupled to "non-chemical" inputs, such as light, enabling their use in energy interconversion and signal transduction. In this way, a number of processor elements in the size range of a few nm have been developed, with obvious advantages in miniaturisation compared to "top-down" machining or lithographic fabrication methods.

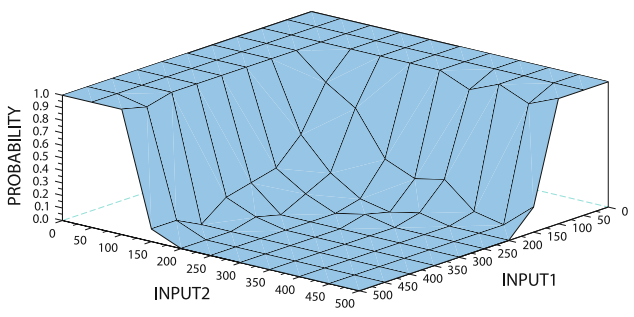


Fig. 29 Probability of the absence of a detectable level of output proteins from the NAND gate for different levels of both inputs

However, potentially much more powerful operations are possible when multicomponent cooperative or interfering interactions are used. Introduction of multiple binding or reporter elements onto polymer chains enables a further level of sophistication in processing information. This is because each interaction, for example at a receptor site, on a polymer chain is inherently coupled to its nearest neighbour on the chain. This can be positive or negative in terms of the next interaction, and thus enhancement or thresholding effects can become apparent. Natural logic systems such as DNA, RNA already exploit these effects in binding or repression of binding as described above, but recent studies have also shown simple logic circuits can be derived from host-guest interactions in synthetic polymers (Pasparakis et al. 2009b). Conformational changes in these polymers resulting from temperature-driven phase transitions cause changes in functional group accessibility which result in "switching" of signalling. The system can be reset with pH or temperature, leading to AND and INH functions. The cooperativity of hydrogen-bonding solvent interactions drives the phase transition, and this is a property related to the balance of entropic and enthalpic factors governing polymer solubility and is fundamentally "polymeric" in origin. These factors combine to produce the overall effect, i.e. switching of binding "on" or "off", but because the phase transition is tuneable through the choice of chemistries in the polymer, other "states" of switching are possible. For example, by connecting polymer chains together in such a way that one component undergoes a phase transition while another does not, a simple "on–off" solubility change can become a unimer-to micelle or unimer-to vesicle switch (Sundararaman et al. 2008). This can be considered as an alteration in the symmetry of the system, as chemical species able to interact with the unimers in isotropic solution become distinct from each other dependent on whether they are inside or outside the micellar or vesicular compartments which form during the polymer phase transition. Functionality that before the transition was identical becomes strongly directional on the inner and outer surfaces of the vesicle, while concentration and diffusion gradients are generated.

In our previous work, a prototype version of the P systems and DPD based simulation and modelling framework was used to investigate the effect of poration on the rate of diffusion of encapsulants through vesicle membranes (Smaldon et al. 2008). The results from simulation indicated a linear relationship between the number of pores present in the membrane and the rate of diffusion. As a step towards a physical implementation of the liposome logic approach, vesicles were prepared which encapsulated a flourescent reporter. Prior literature has shown that several amphiphilic moieties can be employed for encapsulation of

nucleic acids and proteins, some of which have proven useful as bioreactors, drug delivery vehicles or as early-stage protocells (Noireaux and Libchaber 2004; Meng et al. 2009; Roodbeen and van Hest 2009; van Dongen et al. 2009). For instance, some of the amphiphiles employed for the encapsulation of DNA show interesting properties such as the ability to respond to an external stimuli, such as pH (Lomas et al. 2007; Kim et al. 2009). Among those, we have focussed attention on the PEG-PLA system developed by Discher et al. (Kim et al. 2009). for which, the vesicle cargo can be gradually released as a function of the pH, due to the formation of pores in the vesicle membrane, as a result of the hydrolysis of the polyester block (Ahmed and Discher 2004). To this end, an EO_{50}-b-LA_{50} copolymer was synthesized in our laboratories, and EO_{50}-b-LA_{50}-LMVs of approximately 125 nm in size could be synthesized (Fig. 30).

The vesicle formation experiments were performed as follows: Poly(ethylene glycol) monomethyl ether [MW 1,900] was purchased from Polysciences Inc. DL-Lactide (3,6-Dimethyl-1,4-dioxane-2,5-dione) and Tin(II) 2-ethylhexanoate were purchased from Sigma–Aldrich. EO_{50}-b-LA_{50} was prepared from these chemicals according to the procedure described by Discher et al. (Ahmed and Discher 2004). Block composition, purity and final molecular weight were confirmed by nuclear magnetic resonance (NMR) spectra and Gel Permeation Chromatography (GPC). EO_{50}-b-LA_{50} large multilamellar vesicles (EO_{50}-b-LA_{50}-LMVs) were prepared by suspending EO_{50}-b-LA_{50} (2 mg) in H_2O (2 mL) and the suspension stirred overnight at room temperature. Dynamic light scattering (DLS) analysis of these EO_{50}-b-LA_{50}-LMVs was measured using a Viscotek Model 802 instrument equipped with an internal laser (825–832 nm) with a maximum radiation power of 60 mW. At least five measurements of each sample were taken. The mean and standard deviation were calculated. Data processing was performed with the software program OmniSize2. Cryo-TEM analysis of the EO_{50}-b-LA_{50}-LMVs was performed using a JEOL JEM-2100F transmission electron microscope with a Gatan Orius 4K camera and Digital Micrograph imaging software. Frozen hydrated samples were prepared by plunging into liquid ethane using a Gatan Cryoplunge3 plunge freezing system. 5(6)-carboxyfluorescein (CF), 4-(2-Hydroxyethyl)piperazine-1-ethanesulfonic acid, N-(2-Hydroxyethyl)piperazine-N′-(2-ethanesulfonic acid) (HEPES) and NaCl were purchased from Sigma-Aldrich. EO_{50}-b-LA_{50} large unilamelar vesicles containing CF (EO_{50}-b-LA_{50}-LUVs ⊂ CF) were prepared as follows: A thin lipid film was prepared by evaporating a solution of EO_{50}-b-LA_{50} (10 mg) in CHCl3 (1 ml) on a rotary evaporator (30°C, 180 mBar) and then overnight in high vacuo. After hydration (>30 min) with buffer (1 ml; 10 mM HEPES, 10 mM NaCl, 50 mM CF, pH 7.4), the resulting suspension was subjected to 7 freeze-thaw cycles (liquid N2; 30°C, water bath), and 20 times extruded through a polycarbonate membrane (pore size 100 nm). Extravesicular components were removed by size exclusion chromatography (Sephadex G-50) with 10 mM HEPES, 100 mM NaCl, pH 7.4. Fluorescence spectra were recorded on a Cary Eclipse fluorimeter. EO_{50}-b-LA_{50}-LUVs ⊂ CF (25 µl, inside: 10 mM HEPES, 10 mM NaCl, 50 mM CF, pH 7.0) were added to 1,975 µl gently stirred, thermostated buffer (10 mM HEPES, 100 mM NaCl, pH 12) in a disposable plastic cuvette. The time-dependent changes in fluorescence intensity It (λexc = 492 nm, λem = 517 nm) were monitored for 290 min.

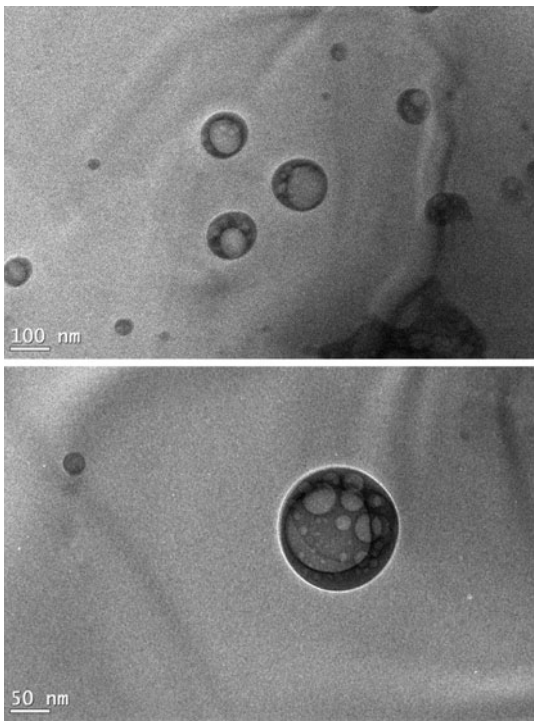


Fig. 30 Cryogenic transmission electron microscope Images of EO_{50}-b-LA_{50} LMVs

Preliminary data has shown that these systems can be used to trigger a response as a function of pH, EO_{50}-b-LA_{50} LUVs loaded with 5(6)-Carboxyfluorescein as a fluorescent reporter were prepared, diluted into pH 12 HEPES buffer and heated to 45°C. Evaluation of the fluorescence as a function of time showed that the vesicle cargo was slowly released, indicating a measurable response to an external stimulus. These results can be compared with those from simulations presented in our previous work (Smaldon et al. 2008), in which the diffusion rate of encapsulated particles within vesicles containing different numbers of membrane pores was determined in simulation. The results are presented in Fig. 31, and relative intensity of the flourescent reporter can be seen to be qualitatively similar to the

change in concentration of encapsulated particle within the simulated vesicle, although occuring over a much longer time-scale.

Ultimately then, it is possible to envisage sophisticated information processing circuits that could be formed using synthetic polymer vesicles in exactly analogous ways to the repressilators described in section "The encapsulated repressilator" for proteins and gene circuits inside liposomes. Interactions of a synthetic polymer with a ligand (a "repressor") in the aqueous interior of a vesicle could lead to a change in solubility of the polymer which drives it towards the hydrophobic interior of the vesicular membrane. Incorporation in the membrane of a reagent capable of, for example, reacting with or sequestering the ligand, will return the polymer back into the vesicle interior, but only in the case where the ligand remains accessible. This can be a function of the degree of binding as the interaction of multiple weakly hydrophobic ligands will, eventually, lead to an overall change in solubility of the polymer-ligand complexes. If the synthetic polymer solubility is tuned such that at a certain binding threshold it then becomes membrane-inserting or membrane-traversing, a "flip-flop" operation becomes possible, dependent on starting concentrations. These in turn will be set by the conversion of unimers to vesicles, generating multiple feedback loops. Thus even for quite simple synthetic polymer constructs, it is theoretically feasible, if not yet fully experimentally tractable, to put in place chemical implementations of the computational simulations and molecular logic operations described earlier.

In the above discussions, we have focussed on water as the solvent in which the chemistries take place. It is also possible to consider other solvents in which micelle, vesicles and other containers form, thus the metabolism and information processing could be far removed from existing biological entities. If the rules of macromolecular phase transitions, vesicle formation and molecular association/dissociation can be derived for other solvent systems, there is no reason why sophisticated logics and synthetic biologies should not emerge in non-aqueous environments.

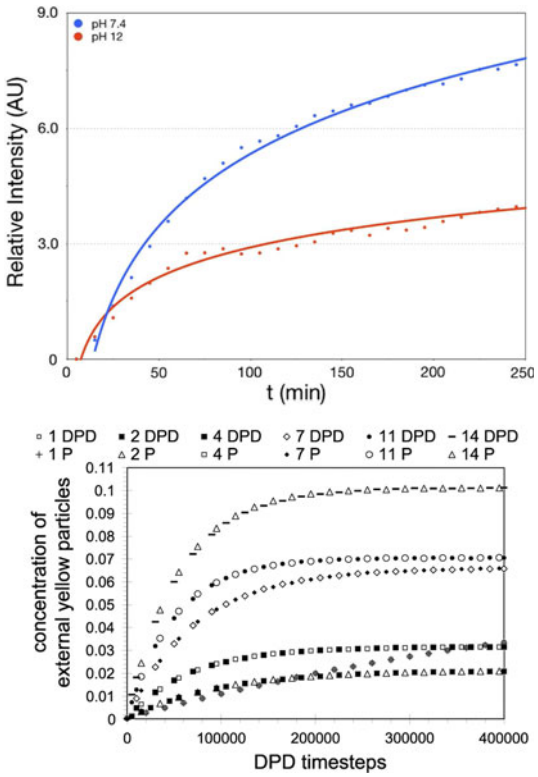


Fig. 31 The *top image* shows the release of CF from EO_{50}-*b*-LA_{50}-*LUVS* as a function of time, measured at 45°C, and two different pH values. The *bottom image* shows the diffusion of encapsulated particles from inside vesicles with 1,2,4,7,11 and 14 pores included in the membrane. The increase in acidity results in a greater number of pores in the membrane which increases the rate at which the flourescent reporter diffuses out of the vesicle. The *bottom image* shows the result of modelling this effect by varying the number of pores in a self-assembled vesicle in DPD

Conclusions

In this paper we have presented our investigations of vesicle and cellular computing which were performed using a novel simulation pipeline. The input to this pipeline is a formal specification of the vesicle computing model in stochastic P systems. This formal specification language is independent of the simulation paradigm used to study the dynamics of the model. Specifically, in this work we have used DPD and SSA to simulate the behaviour of our models. Moreover, our P system specifications enable automatic reasoning, with model checking, of systems and synthetic biology designs at a high level of abstraction.

Modularity in P system models allowed us to develop our models in a parsimonious manner. We started by designing a model of a NOT logic gate, based on the rates and reactions specified in Elowitz's stochastic model of the repressilator. The expressive power of the P systems specification was then illustrated by combining NOT gate modules to make NAND gates, which were in turn used to create a Set-Reset latch, which formed the basic component required to create more complicated flip-flop and counter modules.

In principle any applicable simulation technique could then be chosen to investigate the models, in this paper we focused our analysis of the logic gate models at a very high level of detail in DPD, enabling qualitative understanding to be gained about the behaviour of the model logic gates when encapsulated within a self-assembled liposome, simulations of these models indicated that encapsulation within the liposome had the effect of increasing the rate of

reaction. The SSA method was chosen to explore the behaviour of models which would be too computationally expensive to investigate with DPD, and simulations using this technique showed that complicated logic designs such as a 3-bit counter functioned correctly for several days of simulated time.

Designing a BRN using our pipeline might involve several iterations of the model specification and simulation phases, with non-working prototypes reworked after each iteration. Once the simulations indicate a working design, the designer can move to the next stage in the pipeline, and perform a more robust analysis into the behaviour of the design using model checking. For our liposome logic designs, model checking was performed on the NOT and NAND gates, and temporal logic CSL queries used to determine the propogation delay and minimum input for the gates.

We used the proposed simulation pipeline to illustrate and investigate the concept of performing simple computation in liposomes or vesicles rather than biological cells. The benefit of this approach is that starting from the bottom up and creating systems containing only the required functionality will likely make the system more predictable. The vesicle membranes could also act as a barrier between the implementation of a logic component and the external environment. This provides a second level of modularity as computing systems could be built as a collection of vesicles encapsulating modules that only interact through a well defined interface, the membranes. Potential viable routes for a chemical implementation of vesicle computing have been discussed, but important challenges remain, such as the incorporation of a rudamentanty metabolism to provide the raw materials required to keep the encapsulated logic components functioning. It is our hope that protocell research in synthetic biology may provide some possible solutions to these problems.

Acknowledgments Natalio Krasnogor and Cameron Alexander would like to acknowledge the EPSRC for funding projects *EP/E*017215/1, *EP/G*042462/1 and *EP/H*024905/1, and the BBSRC for their support for the "SynBioNT: A Synthetic Biology Network for Modelling and Programming Cell-Chell Interactions" (BB/F01855X/1).

References

Abelson H, Allen D, Coore D, Hanson C, Homsy G, Knight TF Jr, Nagpal R, Rauch E, Sussman GJ, Weiss R (2000) Amorphous computing. Commun ACM 43(5):74–82

Ahmed F, Discher DE (2004) Self-porating polymersomes of peg-pla and peg-pcl: hydrolysis-triggered controlled release vesicles. J Control Release 96(1):37–53. doi:10.1016/j.jconrel.2003.12.021

Alon U (2007) An introduction to systems biology. CRC Press, Boca Raton

Amos M (ed) (2004) Cellular computing. Oxford University Press, Oxford

Andrianantoandro E, Basu S, Karig DK, Weiss R (2006) Synthetic biology: new engineering rules for an emerging discipline. Mol Syst Biol 2

Asoh Ta, Akashi M (2009) Hydrogel logic gates using gradient semi-ipns. Chem Commun 24(24):3548–3550

Basu S, Mehreja R, Thiberge S, Chen MT, Weiss R (2004) Spatiotemporal control of gene expression with pulse-generating networks. Proc Natl Acad Sci 101(17):6355–6360

Basu S, Gerchman Y, Collins CH, Arnold FH, Weiss R (2005) A synthetic multicellular system for programmed pattern formation. Nature 434(7037):1130–1134

Baumgardner J, Acker K, Adefuye O, Crowley S, DeLoache W, Dickson J, Heard L, Martens A, Morton N, Ritter M, Shoecraft A, Treece J, Unzicker M, Valencia A, Waters M, Campbell A, Heyer L, Poet J, Eckdahl T (2009) Solving a hamiltonian path problem with a bacterial computer. J Biol Eng 3(1):11

Cronin L, Krasnogor N, Davis BG, Alexander C, Robertson N, Steinke JHG, Schroeder SLM, Khlobystov AN, Cooper G, Gardner PM, Siepmann P, Whitaker BJ, Marsh D (2006) The imitation game-a computational chemical approach to recognizing life. Nat Biotechnol 24(10):1203–1206

de Silva AP, Uchiyama S (2007) Molecular logic and computing. Nat Nano 2(7):399–410

Elowitz MB, Leibler S (2000) A synthetic oscillatory network of transcriptional regulators. Nature 403(6767):335–338

Endy D (2005) Foundations for engineering biology. Nature 438(7067):449–453

Español P, Warren P (1995) Statistical-mechanics of dissipative particle dynamics. Europhys Lett 30:191–196

Fisher J, Henzinger T (2007) Executable cell biology. Nat Biotechnol 25:1239–1249

Gardner P, Winzer K, Davis B (2009) Sugar synthesis in a protocellular model leads to a cell signalling response in bacteria. Nat Chem 1:377–383

Gillespie DT (2007) Stochastic simulation of chemical kinetics. Annu Rev Phys Chem 58:35–55

Groot RD, Rabone KL (2001) Mesoscopic simulation of cell membrane damage, morphology change and rupture by nonionic surfactants. Biophys J 81:725–736

Groot RD, Warren PB (1997) Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulation. J Chem Phys 107(11):4423–4435

Gunnlaugsson T, Donail DAM, Parker D (2000) Luminescent molecular logic gates: the two-input inhibit (inh) function. Chem Commun 1(1):93–94

Hartwell LH, Hopfield JJ, Leibler S, Murray AW (1999) From molecular to modular cell biology. Nat Impacts 402:47–52

Heinemann M, Panke S (2006) Synthetic biology—putting engineering into biology. Bioinformatics 22(22):2790–2799

Kim Y, Tewari M, Pajerowski JD, Cai S, Sen S, Williams J, Sirsi S, Lutz G, Discher DE (2009) Polymersome delivery of sirna and antisense oligonucleotides. J Control Release 134(2):132–140. doi:10.1016/j.jconrel.2008.10.020

Knight T (2003) Idempotent vector design for standard assembly of biobricks. http://hdl.handle.net/1721.1/21168

Kranenburg M, Nicolas JP, Smit B (2004) Comparison of mesoscopic phospholipid-water models. Phys Chem Chem Phys 6(16): 4142–4151

Kwiatkowska M, Norman G, Parker D (2002) Prism: probabilistic symbolic model checker

Kwiatkowska M, Norman G, Parker D (2009) Probabilistic model checking for systems biology. Symb Syst Biol (to appear)

Lomas H, Canton I, MacNeil S, Du J, Armes S, Ryan A, Lewis A, Battaglia G (2007) Biomimetic ph sensitive polymersomes for efficient dna encapsulation and delivery. Adv Mater 19(23): 4238–4243. 10.1002/adma.200700941

Luisi P, Ferri F, Stano P (2006) Approaches to semi-synthetic minimal cells: a review. Naturwissenschaften 93(1):1–13

MacDiarmid J, Amaro-Mugridge N, Madrid-Weiss J, Sedliarou I, Wetzel S, Kochar K, Brahmbhatt V, Phillips L, Pattison S, Petti C, Stillman B, Graham R, Brahmbhatt H (2009) Sequential treatment of drug-resistant tumors with targeted minicells containing sirna or a cytotoxic drug. Nat Biotechnol 27(7): 643–651

Magri DC (2009) A fluorescent and logic gate driven by electrons and protons. New J Chem 33(3):457–461

Mallavarapu A, Thomson M, Ullian B, Gunawardena J (2009) Programming with models: modularity and abstraction provide powerful capabilities for system biology. J R Soc Interface 6:257–270

Meng F, Zhong Z, Feijen J (2009) Stimuli-responsive polymersomes for programmed drug delivery. Biomacromolecules 10(2):197–209 10.1021/bm801127d

Noireaux V, Libchaber A (2004) A vesicle bioreactor as a step toward an artificial cell assembly. PNAS 101(51):17669–17674

Pasparakis G, Alexander C (2008) Sweet talking double hydrophilic block copolymer vesicles. Angew Chem Int Ed Engl 47(26): 4847–4850

Pasparakis G, Krasnogor N, Cronin L, Davis B, Alexander C (2009a) Controlled polymer synthesis—from biomimicry towards synthetic biology. Chem Soc Rev (in press). doi:10.1039/B809333B

Pasparakis G, Vamvakaki M, Krasnogor N, Alexander C (2009b) Diol-boronic acid complexes integrated by responsive polymers—a route to chemical sensing and logic operations. Soft Matter (in press). doi:10.1039/b911341

Pérez-Jiménez MJ, Romero-Campero FJ (2006) P systems, a new computational modelling tool for systems biology. Trans Comput Syst Biol VI:176–197

Pischel U (2007) Chemical approaches to molecular logic elements for addition and subtraction. Angew Chem Int Ed 46(22): 4026–4040

Priami C (2009) Algorithmic systems biology. Commun ACM 52(5):80–88

Păun G (2002) Membrane computing: an introduction. Springer, New York

Rasmussen S, Bedau MA, Chen L, Deamer D, Krakauer DC, Packard NH, Stadler PF (eds) (2008) Protocells: bridging nonliving and living matter. MIT Press, Cambridge

Romero-Campero FJ, Cao H, Camara M, Krasnogor N (2008) Structure and parameter estimation for cell systems biology models. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008), ACM Publisher, pp 331–338

Romero-Campero FJ, Twycross J, Camara M, Bennett M, Gheorghe M, Krasnogor N (2009) Modular assembly of cell systems biology models using p systems. Int J Found Comput Sci 20(3):427–442

Roodbeen R, van Hest JCM (2009) Synthetic cells and organelles: compartmentalization strategies. BioEssays 31(12):1299–1308 doi:10.1002/bies.200900106

Serrano L (2007) Synthetic biology: promises and challenges. Mol Syst Biol 3

Shetty R, Endy D, Knight T (2008) Engineering biobrick vectors from biobrick parts. J Biol Eng 2(1):5

Shillcock J, Lipowsky R (2005) Tension-induced fusion of bilayer membranes and vesicles. Nat Mater 4:225–228

Smaldon J, Blakes J, Krasnogor N, Lancet D (2008) A multi-scaled approach to artificial life simulation with p systems and dissipative particle dynamics. In: GECCO '08: proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA, pp 249–256. doi: 10.1145/1389095.1389134

Sundararaman A, Stephan T, Grubbs RB (2008) Reversible restructuring of aqueous block copolymer assemblies through stimulus-induced changes in amphiphilicity. J Am Chem Soc 130(37): 12264–12265

Tan C, Song H, Niemi J, You L (2007) A synthetic biology challenge: making cells compute. Mol Biosyst 3:343–353

Tan YC, Hettiarachchi K, Siu M, Pan YR, Lee AP (2006) Controlled microfluidic encapsulation of cells, proteins, and microbeads in lipid vesicles. J Am Chem Soc 128(17):5656–5658

van Dongen SFM, Nallani M, Cornelissen JJLM, Nolte RJM, van Hest JCM (2009) A three-enzyme cascade reaction through positional assembly of enzymes in a polymersome nanoreactor. Chem Eur J 15(5):1107–1114. 10.1002/chem.200802114

Weiss R, Basu S (2002) The device physics of cellular logic gates. In: The first workshop on non silicon computing

Yoshida W, Yokobayashi Y (2007) Photonic boolean logic gates based on dna aptamers. Chem Commun 2(2):195–197