MIT Synthetic Biology - **Designing and encoding models for synthetic biology**

Russell W. Hanson

Aug. 26, 2009

Designing and encoding models for synthetic biology

Lukas Endler, Nicolas Rodriguez, Nick Juty, Vijayalakshmi Chelliah, Camille Laibe, Chen Li and Nicolas Le Novère

J. R. Soc. Interface 2009 **6**, S405-S417 first published online 1 April 2009
doi: 10.1098/rsif.2009.0035.focus

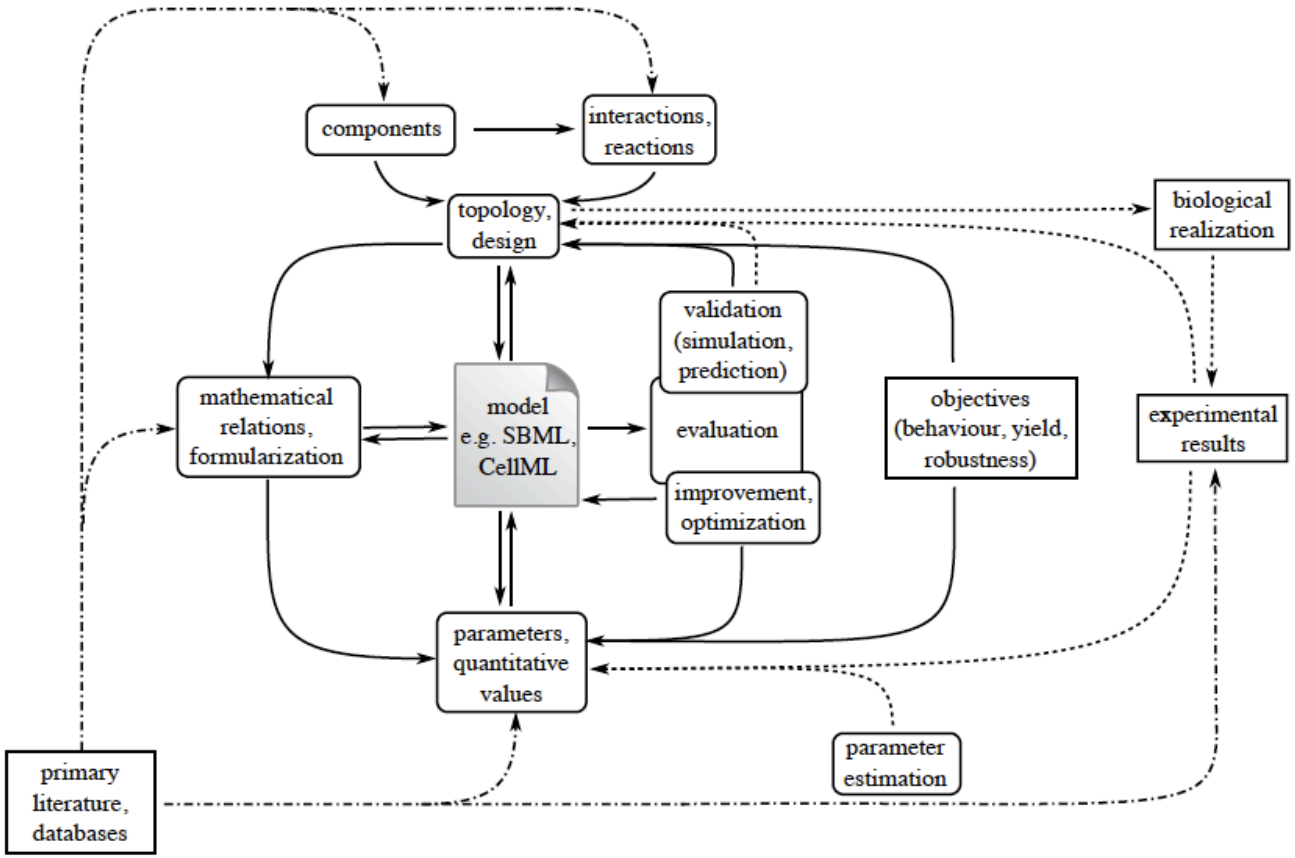


Figure 1. Diagram describing the model creation process, centred around the model file. External inputs come from literature, databases or experiments. Validation and parameter estimation give direct feedbacks on the model, while the predictions can also guide experimental design. The desired objectives for the system to be designed are part of a mostly *in silico* design cycle, the biological implementation and the fitting and comparison of the model to observations are part of the standard systems biology model development cycle.

Model creation for synthetic bio.

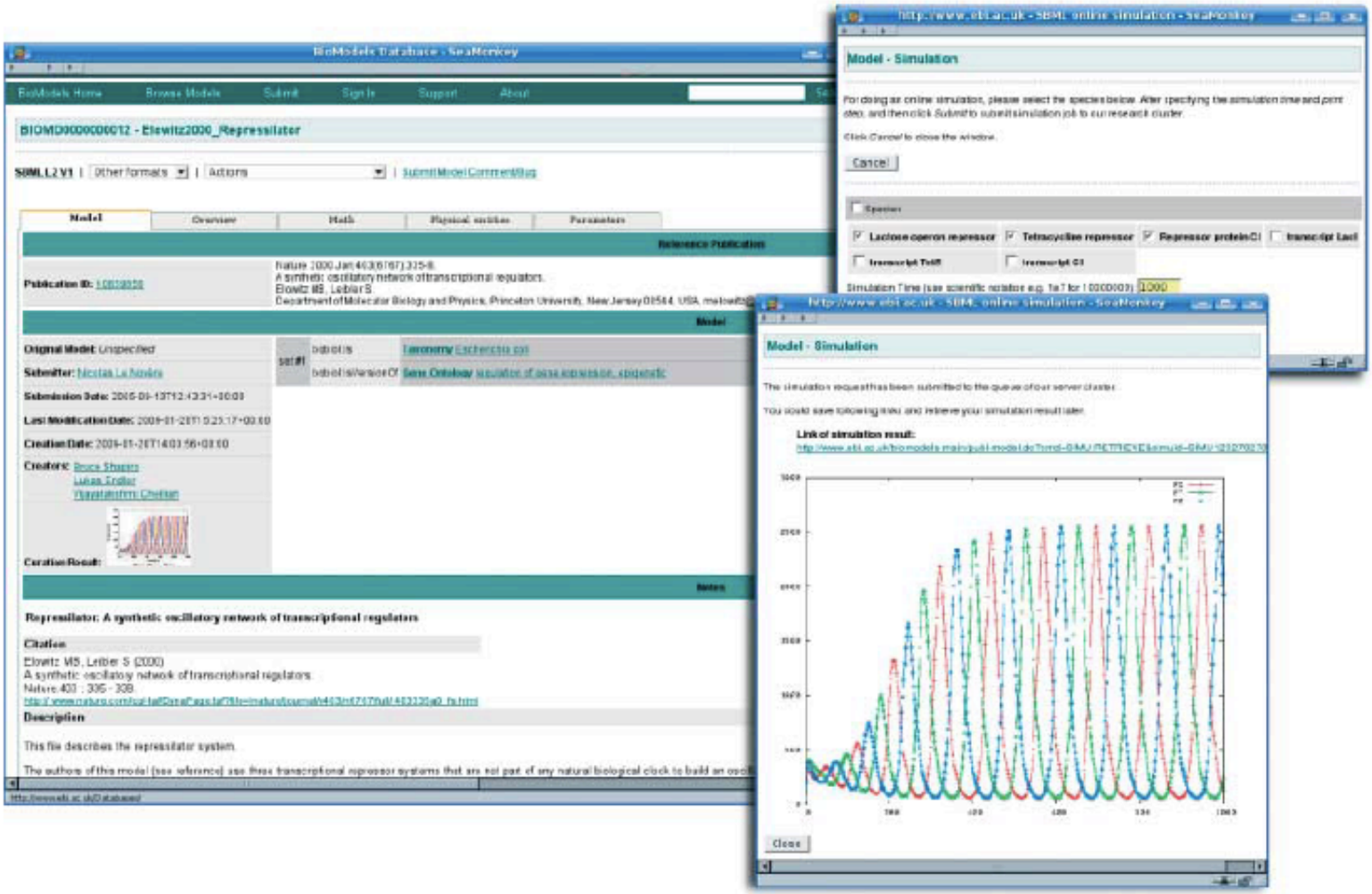


Figure 2. Biomodels database is a resource offering curated and annotated versions of models published in peer-reviewed literature. The models can be browsed, downloaded or directly simulated online, as shown here for the repressilator.

Graphical layout, design and ‘programming’, *CellDesigner*

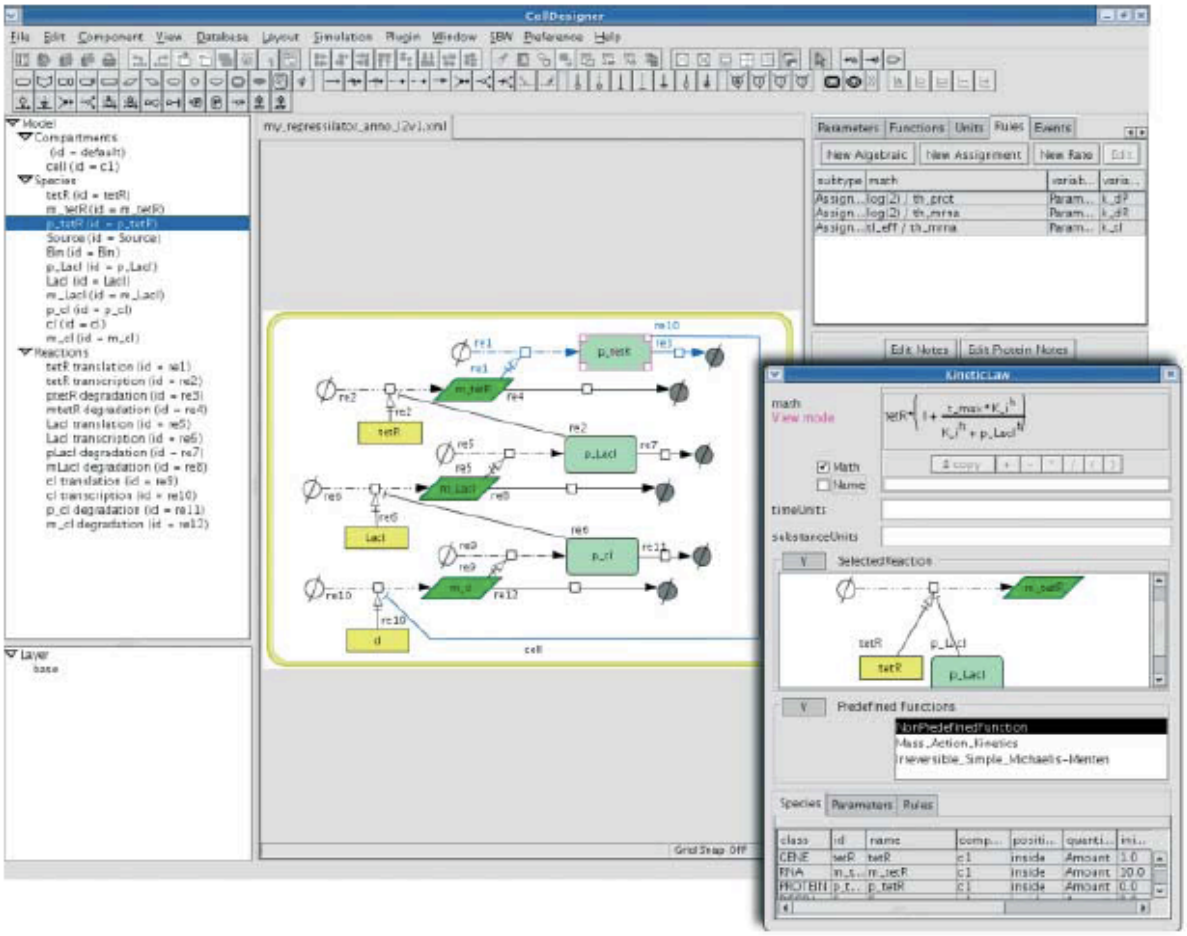


Figure 3. The CELLDESIGNER interface provides an ample selection of graphical elements and editing options to design biochemical and mathematical models. The repressilator is represented in CELLDESIGNER's graphical notation, a derivative of SBGN. The raised window is the editing form for the kinetic law of a single reaction.

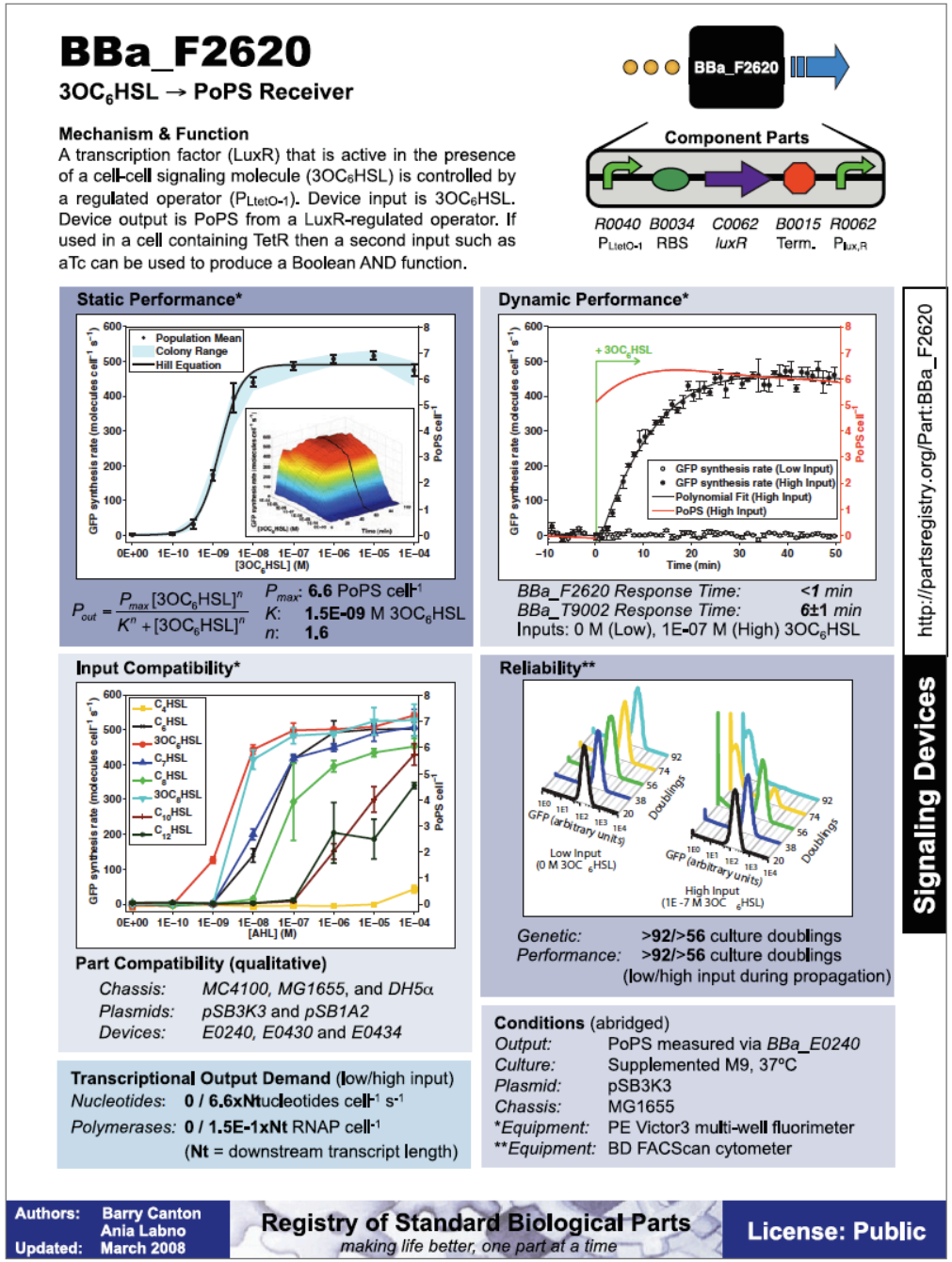


Figure 3 A prototypical 'datasheet' that summarizes current knowledge of the behavior of the receiver BBa_F2620. The datasheet, which includes a general description and a summary of relevant performance characteristics, is designed to support rapid reuse of the device. The description of the receiver is also available in electronic format[21]. A glossary for the datasheet is provided in **Box 3**.

"… represent quantitative characteristics of biological devices in the form of a **datasheet** and demonstrate it on a device composed of BioBrick parts. (PartsRegistry.org)"

(Canton, Labno, Endy 2008 Refinement and standardization of synthetic biological parts and devices.)

Data Supplement – input files

SBML l2v1 CellDesigner - Level 2 version 1 SBML file of the repressilator with SBGN graphics in CellDesigner annotations

2009-01-22T17:26:46+00:00 2009-01-22T18:55:30+00:00
2th_prot 2th_mrna tl_effth_mrna k_tlm_tetR tetRlt_maxK_ihK_ihp_LacIh
k_dPp_tetR k_dRm_tetR k_tlm_LacI LacIlt_maxK_ihK_ihp_cIh k_dPp_LacI
k_dRm_LacI k_tlm_cI cIlt_maxK_ihK_ihp_tetRhk_dPp_cI k_dRm_cI

SBMLl2v3 SBO annotations - Fully annotated SBML level 2 version 3 file of the repressilator

4.0 SQUARE inactive inactive inactive inactive inactive inactive inactive inactive inactive
inactive inactive inactive inactive inactive inactive inactive inactive inactive inactive
inactive inactive 2009-01-22T17:26:46+00:00 2009-01-22T18:55:30+00:00
cell inside GENE gn1 inside RNA rn2 inside PROTEIN pr1 inside DEGRADED Source
inside DEGRADED Bin inside PROTEIN pr2 inside GENE gn2 inside RNA rn3 inside
PROTEIN pr3 inside GENE gn3 inside RNA rn4 2th_prot 2th_mrna tl_effth_mrna
TRANSLATION sa7 sa2 sa5 k_tlm_tetR TRANSCRIPTION sa4 sa5 sa6 sa14
tetRlt_maxK_ihK_ihp_LacIh STATE_TRANSITION sa2 sa8 k_dPp_tetR
STATE_TRANSITION sa5 sa9 k_dRm_tetR TRANSLATION sa12 sa14 sa13
k_tlm_LacI TRANSCRIPTION sa10 sa13 sa11 sa20 LacIlt_maxK_ihK_ihp_cIh
STATE_TRANSITION sa14 sa16 k_dPp_LacI STATE_TRANSITION sa13 sa15 k_dRm_LacI
TRANSLATION sa18 sa20 sa17 TRANSCRIPTION sa19 sa17 sa21 sa2 cIlt_maxK_ihK_ih
STATE_TRANSITION sa20 sa23 k_dPp_cI STATE_TRANSITION sa17 sa22 k_dRm_cI

SynBioSS: the synthetic biology modeling suite

Anthony D. Hill[1,2], Jonathan R. Tomshine[1,2], Emma M. B. Weeding[1,2], Vassilios Sotiropoulos[1,2] and Yiannis N. Kaznessis[1,2,*]

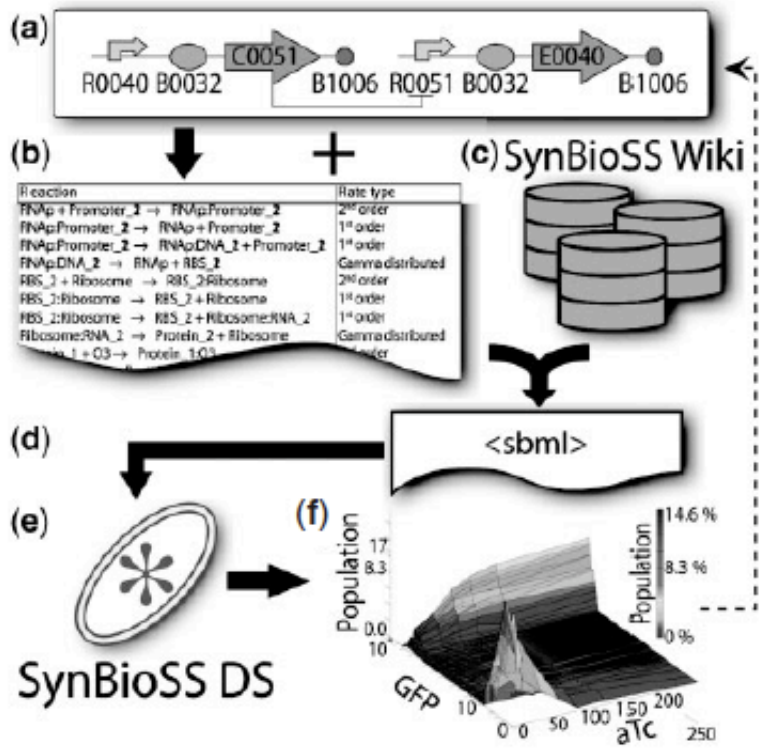


Fig. 1. Overview of the SynBioSS workflow. The user starts at (**a**) a synthetic gene construct (BioBrick nomenclature). Using the SynBioSS Designer creates (**b**) a kinetic model of the reaction network. Adding (**c**) the SynBioSS Wiki, (**d**) a kinetic model in SBML is generated. This model is then simulated with (**e**) the SynBioSS simulator, resulting in (**f**) a population distribution of each species over time. The process can then be iterated until the desired phenotype is achieved.

Modular cell biology: retroactivity and insulation

Domitilla Del Vecchio[1,*], Alexander J Ninfa[2] and Eduardo D Sontag[3]

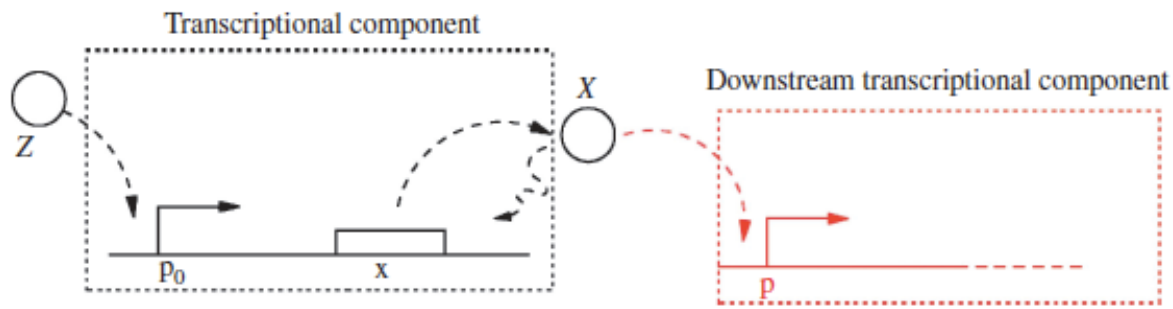


Figure 2 The transcriptional component takes as input *u* protein concentration *Z* and gives as output *y* protein concentration *X*. The transcription factor *Z* binds to operator sites on the promoter. The red part belongs to a downstream transcriptional block that takes protein concentration *X* as its input.

the dashed box. The activity of the promoter controlling gene x depends on the amount of Z bound to the promoter. If $Z=Z(t)$, such an activity changes with time. We denote it by $k(t)$. By neglecting the mRNA dynamics, we can write the dynamics of X as

$$\frac{dX}{dt} = k(t) - \delta X, \tag{2}$$

in which δ is the decay rate of the protein. We refer to equation (2) as the isolated system dynamics. For the current study, the

BMC Bioinformatics

BioMed Central

Software

Open Access

Multiscale Hy3S: Hybrid stochastic simulation for supercomputers

Howard Salis, Vassilios Sotiropoulos and Yiannis N Kaznessis*

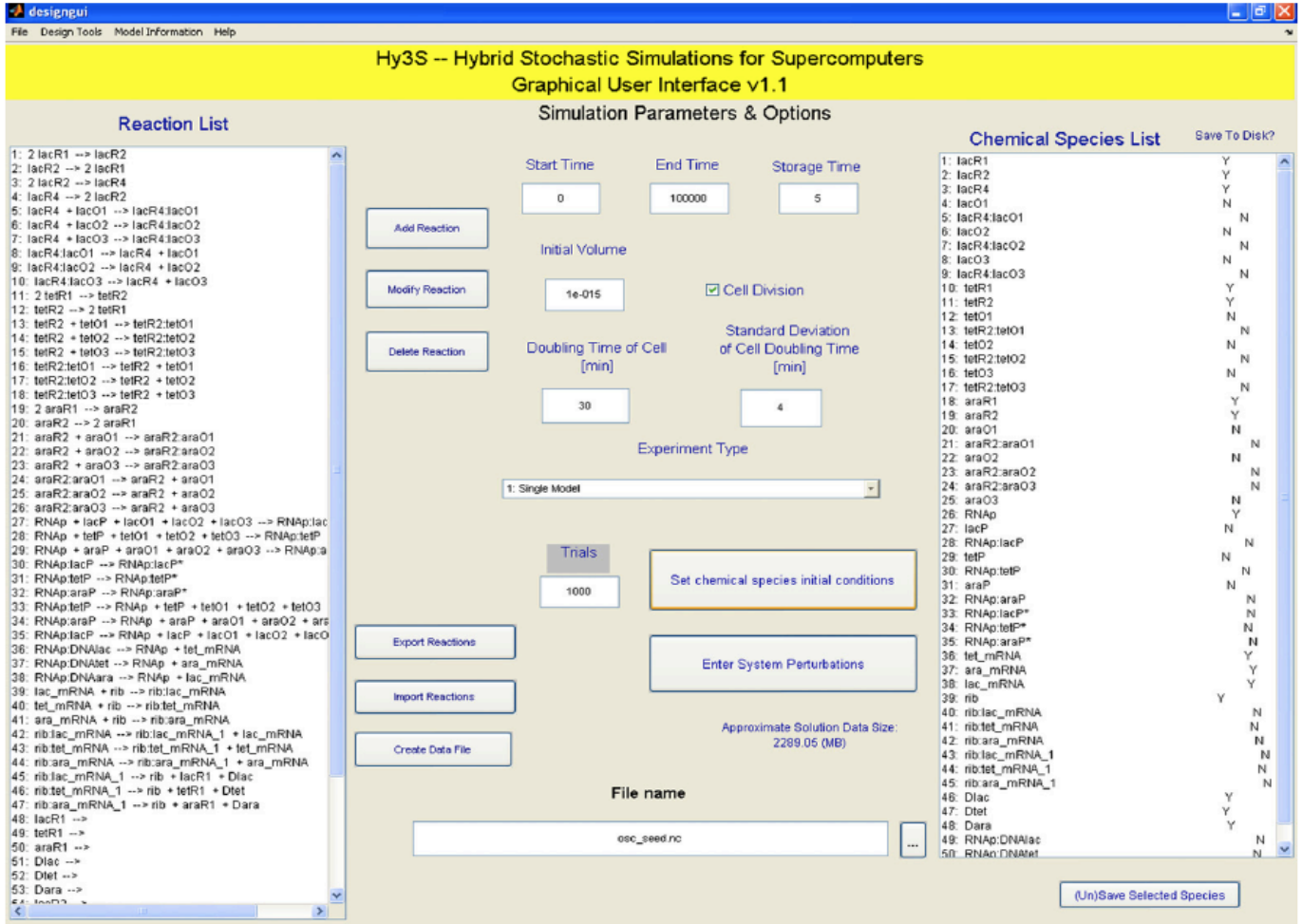


Figure 1
The main window of the graphical user interface.

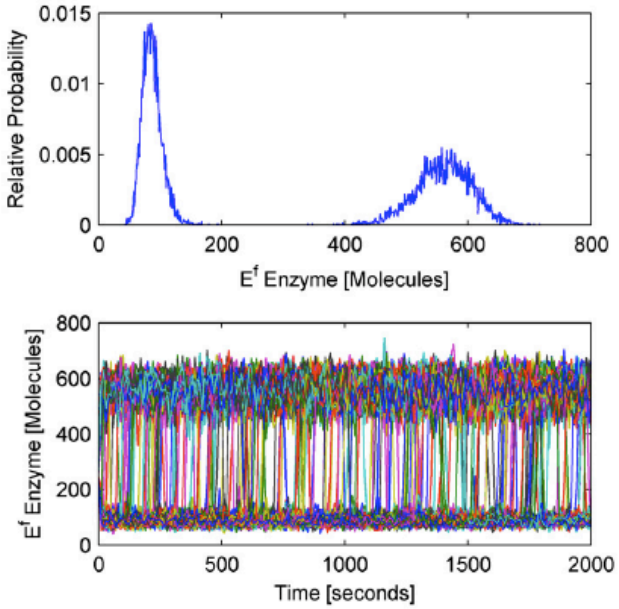


Figure 8
Distribution and trajectories of the Schlogl reaction module(Top) The relative probability distribution of the number of E^f molecules at 50 seconds. (Bottom) Out of 10 000 independent trajectories, the 225 shown here exhibit spontaneous transitions from low to high or high to low numbers of E^f molecules.

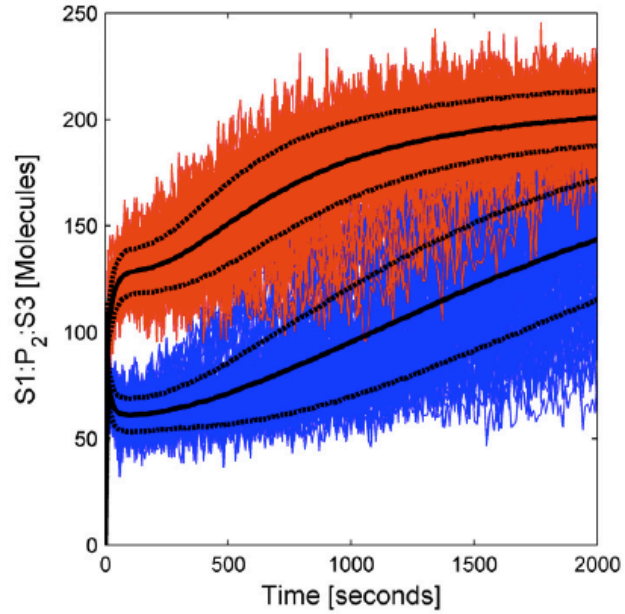
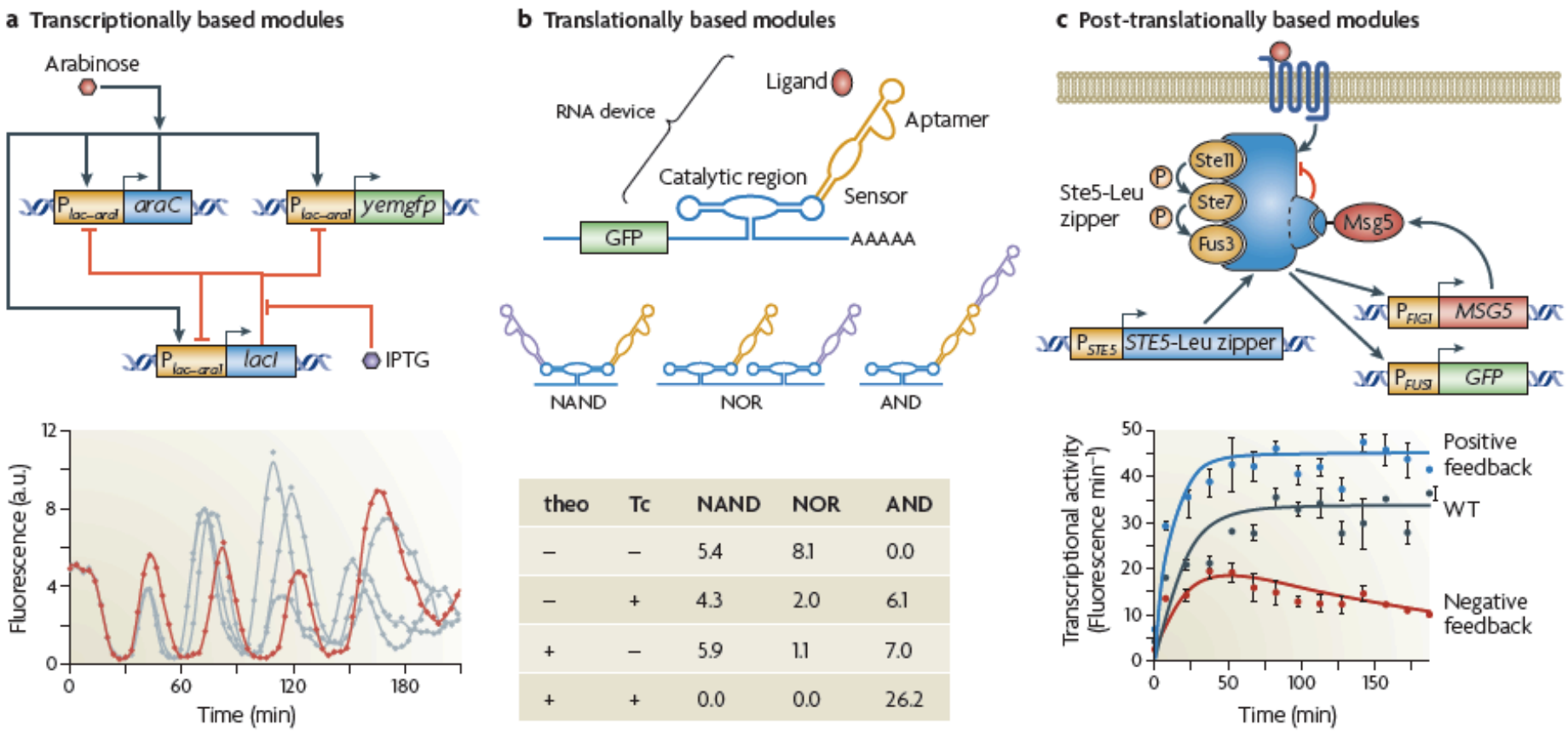


Figure 9
Branching of solution affects bound scaffold complexes. An ensemble of 10 000 trajectories of the $S1{:}P_2{:}S3$ scaffold complex, where trajectories are colored according to the branch of the solution. The number of E^f molecules resides in either the (red) low or (blue) high stable state. Both the (black solid lines) mean and (black dashed lines) mean ± standard deviation are shown for both branches.

(Rao CV, Arkin AP: Stochastic chemical kinetics and the quasi steady-state assumption: application to the Gillespie algorithm. J Chem Phys 2003, 118:4999-5010)

“The second wave of synthetic biology: from modules to systems”



theo	Tc	NAND	NOR	AND
–	–	5.4	8.1	0.0
–	+	4.3	2.0	6.1
+	–	5.9	1.1	7.0
+	+	0.0	0.0	26.2

Figure 1 | **Modules based on transcriptional, translational and post-translational control. a** | The dual-feedback

(“The second wave of synthetic biology: from modules to systems,” Purnick PEM, Weiss R, Nature Reviews Molecular Cell Biology, Volume: 10 Issue: 6 Pages: 410-422 Jun 2009.)

Design and analysis tools

Table 2 | **Currently available computational tools for the design and analysis of genetic networks**

Computational tool	Use	Website[‡]
21U-RNA	Scoring 21U-RNA-associated upstream motifs	Bartel laboratory introduction to 21U-RNAs
Antimony*	Programming language describing synthetic biological devices	Deepak laboratory syntax guide
Athena*	Build and simulate genetic circuits (implemented in C++)	Deepak laboratory downloads
BioJade*	Synthetic biology design and simulation (implemented in Java)	BioJade
CAD of modular protein devices	Modular protein device algorithm using a backbone of scaffold proteins[82]	None
ESSA	RNA secondary structure analysis	ESSA
Evolutionary design of genetic networks *in silico*	Algorithm to evolve small gene networks (modules) that perform basic tasks, such as toggle switches or oscillators[81]	None
GeneDesign*	Editing protein sequences and generating oligos for protein construction (implemented in Perl)	Gene Design
GeNetDes*	Transcriptional network design tool using simulated annealing optimization	Genetdes
GenoCAD*	Design of complex genetic constructs from standard parts library	GenoCAD
MiRscan	Scoring of hairpins versus some experimentally verified microRNAs from *Caenorhabditis elegans* or *Caenorhabditis briggsae*	MiRscan
OptCircuit	Identifies circuit components and suggests circuit topologies to attain desired outcome[85]	None
PCEnv*	Environment for simulating various types of CellML models	OpenCell
PROTDES*	Computational protein design	PROTDES
Random Sampling-High Dimensional Model Representation	Global sensitivity analysis algorithm that is useful in optimizing genetic circuit properties not available from experiments or modelling[86]	None
Registry of Standard Biological Parts and Clotho*	Creation, cataloguing and public availability of modular biological parts that are extensively characterized; Clotho is a database for managing these parts	Registry of Standard Biological Parts and Clotho Development
RNA world website	Compendium of RNA software	RNA world
RNAdraw	RNA secondary structure analysis	RNAdraw
RNAMotif	Database search for RNA sequences that match a secondary structure motif	Rutgers Case Group
RNAstructure	RNA secondary structure analysis	RNAstructure
RnaViz	RNA secondary structure images	RnaViz
Rosetta package	Design of protein-binding peptide sequences and protein engineering	Rosetta @ home and Rosetta Commons
RoVerGeNe*	Tool to analyse and tune gene networks	RoVerGeNe
SynBioSS*	Suite of programs to generate and simulate synthetic biological networks	SynBioSS
Tinkercell	Synthetic biology CAD program	Tinkercell
UNAFold software	Nucleic acid folding and hybridization	UNAFold software
Vienna RNA package	RNA secondary structure	Vienna RNA package

Biological “op-amps”

(*Ibid.*)

designed so as to attenuate the retroactivity to its output. We thus suggest a mechanism similar to that used to design non-inverting amplifiers employing operational amplifiers (OPAMPs; Schilling and Belove, 1968) to attenuate retroactivity. This simple mechanism employs a large input gain and a similarly large negative feedback. We then propose and analyze two biological instances of this mechanism, for gene and protein networks. The first one involves a strong, non-

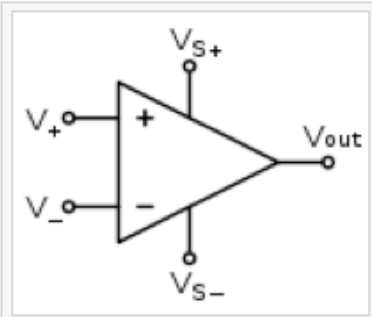
Circuit notation

[edit]

The circuit symbol for an op-amp is shown to the right, where:

- V_+: non-inverting input
- V_-: inverting input
- V_{out}: output
- V_{S+}: positive power supply
- V_{S-}: negative power supply

The power supply pins (V_{S+} and V_{S-}) can be labeled in different ways (*See IC power supply pins*). Despite different labeling, the function remains the same — to provide additional power for amplification of signal. Often these pins are left out of the diagram for clarity, and the power configuration is described or assumed from the circuit.



Circuit diagram symbol for an op-amp

Mathematical structures

(Marker, Model Theory, 2002)

Definition 1.1.6 Let ϕ be a formula with free variables from $\overline{v} = (v_{i_1}, \ldots, v_{i_m})$, and let $\overline{a} = (a_{i_1}, \ldots, a_{i_m}) \in M^m$. We inductively define $\mathcal{M} \models \phi(\overline{a})$ as follows.

i) If ϕ is $t_1 = t_2$, then $\mathcal{M} \models \phi(\overline{a})$ if $t_1^{\mathcal{M}}(\overline{a}) = t_2^{\mathcal{M}}(\overline{a})$.

ii) If ϕ is $R(t_1, \ldots, t_{n_R})$, then $\mathcal{M} \models \phi(\overline{a})$ if $(t_1^{\mathcal{M}}(\overline{a}), \ldots, t_{n_R}^{\mathcal{M}}(\overline{a})) \in R^{\mathcal{M}}$.

iii) If ϕ is $\neg\psi$, then $\mathcal{M} \models \phi(\overline{a})$ if $\mathcal{M} \not\models \psi(\overline{a})$.

iv) If ϕ is $(\psi \wedge \theta)$, then $\mathcal{M} \models \phi(\overline{a})$ if $\mathcal{M} \models \psi(\overline{a})$ and $\mathcal{M} \models \theta(\overline{a})$.

v) If ϕ is $(\psi \vee \theta)$, then $\mathcal{M} \models \phi(\overline{a})$ if $\mathcal{M} \models \psi(\overline{a})$ or $\mathcal{M} \models \theta(\overline{a})$.

vi) If ϕ is $\exists v_j \psi(\overline{v}, v_j)$, then $\mathcal{M} \models \phi(\overline{a})$ if there is $b \in M$ such that $\mathcal{M} \models \psi(\overline{a}, b)$.

vii) If ϕ is $\forall v_j \psi(\overline{v}, v_j)$, then $\mathcal{M} \models \phi(\overline{a})$ if $\mathcal{M} \models \psi(\overline{a}, b)$ for all $b \in M$.

If $\mathcal{M} \models \phi(\overline{a})$ we say that $\mathcal{M}$ *satisfies* $\phi(\overline{a})$ or $\phi(\overline{a})$ is *true* in $\mathcal{M}$.

Example 1.2.11 *Peano Arithmetic*

Let $\mathcal{L} = \{+, \cdot, s, 0\}$, where $+$ and $\cdot$ are binary functions, s is a unary function, and 0 is a constant. We think of s as the successor function $x \mapsto x + 1$. The Peano axioms for arithmetic are the sentences

$\forall x \; s(x) \neq 0,$

$\forall x \; (x \neq 0 \rightarrow \exists y \; s(y) = x),$

$\forall x \; x + 0 = x,$

$\forall x \; \forall y \; x + (s(y)) = s(x + y),$

$\forall x \;\; x \cdot 0 = 0,$

$\forall x \forall y \; x \cdot s(y) = (x \cdot y) + x,$

and the axioms $\mathrm{Ind}(\phi)$ for each formula $\phi(v, \overline{w})$, where $\mathrm{Ind}(\phi)$ is the sentence

Halting problem in biology

Halting problem

In computability theory, the halting problem is a decision problem which can be stated as follows: given a description of a program and a finite input, decide whether the program finishes running or will run forever, given that input.

Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. We say that the halting problem is undecidable over Turing machines.

Examples of halting problem in biological systems (http://www.google.com/search?q=halting+problem+in+biology ???):

-

-

Omega in Biology

The classical halting probability Ω introduced by Chaitin is generalized to quantum computations.

Chaitin's Ω [1, 2, 3] is a magic number. It is a measure for arbitrary programs to take a finite number of execution steps and then halt. It contains the solution for all halting problems, and hence to questions codable into halting problems, such as Fermat's theorem. It contains the solution for the question of whether or not a particular exponential Diophantine equation has infinitely many or a finite number of solutions. And, since Ω is provable "algorithmically incompressible," it is Martin-Löf/Chaitin/Solovay random. Therefore, Ω is both: a mathematicians "fair coin," and a formalist's nightmare.

In analogy to the fully classical case [1, 18, 3], the quantum halting amplitude Ω can be defined as a weighted expectation over all computations of C with classical input p_i ($|p_i|$ stands for the length of p_i)

$$\Omega \equiv \lim_{t\to\infty} \sum_{p_i} 2^{-|p_i|} \left[\langle HALT|C(t,p_i)\rangle - \langle GO|C(t,p_i)\rangle\right] \quad . \tag{7}$$

(K. Svozil, Halting probability amplitude of quantum computers, 1995)

Riemann zeta function

Implement arbitrary calculations on '*biological hardware*'

- Make slow numerical calculators
- Control, in a systems or robotics sense, of small machines

```
if wall
    turn 5*right + 2; %% procedure and function
end
```

- "Libraries" of functions and procedures, not physical parts (sometimes "DNA computing")
- … and of course, calculate the Riemann zeta function.

On the real line with $x > 1$, the Riemann zeta function can be defined by the integral

$$\zeta(x) \equiv \frac{1}{\Gamma(x)} \int_0^\infty \frac{u^{x-1}}{e^u - 1}\, du,$$

Finis