MIT Synthetic Biology -
Designing and encoding models for synthetic biology

Russell W. Hanson
Aug. 26, 2009
Figure 1. Diagram describing the model creation process, centered 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.
“... represent quantitative characteristics of biological devices in the form of a datasheet and demonstrate it on a device composed of BioBrick parts. (PartsRegistry.org)”
Data Supplement – input files

SBML 12v1 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_Lacl LacIlt_maxK_ihK_ihp_clh k_dPp_Lacl
k_dRm_Lacl k_tlm_cl cllt_maxK_ihK_ihp_tetRhk_dPp_cl k_dRm_cl

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 inactive inactive inactive inactive inactive inactive inactive 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_Lacl TRANSCRIPTION sa10 sa13 sa11 sa20 LacIlt_maxK_ihK_ihp_clh
STATE_TRANSITION sa14 sa16 k_dPp_Lacl STATE_TRANSITION sa13 sa15 k_dRm_Lacl
TRANSLATION sa18 sa20 sa17 TRANSCRIPTION sa19 sa17 sa21 sa2 cllt_maxK_ihK_ih
STATE_TRANSITION sa20 sa23 k_dPp_cl STATE_TRANSITION sa17 sa22 k_dRm_cl
SynBioSS: the synthetic biology modeling suite

Anthony D. Hill\textsuperscript{1,2}, Jonathan R. Tomshine\textsuperscript{1,2}, Emma M. B. Weeding\textsuperscript{1,2}, Vassilios Sotiropoulos\textsuperscript{1,2} and Yiannis N. Kaznessis\textsuperscript{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 Vecchio¹*, Alexander J Ninfa² and Eduardo D Sontag³

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

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

(2)

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

**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 Schlögl reaction module. (Top) The relative probability distribution of the number of $E^1$ 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 $E^1$ molecules.

Figure 9
Branching of solution affects bound scaffold complexes. An ensemble of 10 000 trajectories of the $S1P_2S3$ scaffold complex, where trajectories are colored according to the branch of the solution. The number of $E^1$ 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.

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

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

## Design and analysis tools

<table>
<thead>
<tr>
<th>Computational tool</th>
<th>Use</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>21U-RNA</td>
<td>Scoring 21U-RNA-associated upstream motifs</td>
<td>Barrell laboratory introduction to 21U-RNAs</td>
</tr>
<tr>
<td>Antimony*</td>
<td>Programming language describing synthetic biological devices</td>
<td>Deepak laboratory syntax guide</td>
</tr>
<tr>
<td>Athena*</td>
<td>Build and simulate genetic circuits (implemented in C++)</td>
<td>Deepak laboratory downloads</td>
</tr>
<tr>
<td>Biode$e$*</td>
<td>Synthetic biology design and simulation (implemented in Java)</td>
<td>Biode$e$</td>
</tr>
<tr>
<td>CAD of modular protein devices*</td>
<td>Modular protein device algorithm using a backbone of scaffold proteins*</td>
<td>None</td>
</tr>
<tr>
<td>ESSA</td>
<td>RNA secondary structure analysis</td>
<td>ESSA</td>
</tr>
<tr>
<td>Evolutionary design of genetic networks in silico</td>
<td>Algorithm to evolve small gene networks (modules) that perform basic tasks, such as toggle switches or oscillators*</td>
<td>None</td>
</tr>
<tr>
<td>GeneDesign*</td>
<td>Editing protein sequences and generating alleles for protein construction (implemented in Perl)</td>
<td>GeneDesign</td>
</tr>
<tr>
<td>GeNetDes*</td>
<td>Transcriptional network design tool using simulated annealing optimization</td>
<td>GeNetDes</td>
</tr>
<tr>
<td>GenoCAD*</td>
<td>Design of complex genetic constructs from standard parts library</td>
<td>GenoCAD</td>
</tr>
<tr>
<td>MiRscan</td>
<td>Scoring of hairpins versus some experimentally verified microRNAs from Caenorhabditis elegans or Caenorhabditis briggsae</td>
<td>MiRscan</td>
</tr>
<tr>
<td>OptiCircuit</td>
<td>Identifies circuit components and suggests circuit topologies to attain desired outcome*</td>
<td>None</td>
</tr>
<tr>
<td>PCell*</td>
<td>Environment for simulating various types of CellML models</td>
<td>OpenCell</td>
</tr>
<tr>
<td>PROTDES*</td>
<td>Computational protein design</td>
<td>PROTDES</td>
</tr>
<tr>
<td>Random Sampling-High Dimensional Model Representation</td>
<td>Global sensitivity analysis algorithm that is useful in optimizing genetic circuit properties not available from experiments or modelling*</td>
<td>None</td>
</tr>
<tr>
<td>Registry of Standard Biological Parts and Clothes*</td>
<td>Creation, cataloguing and public availability of modular biological parts that are extensively characterized; Clothio is a database for managing these parts</td>
<td>Registry of Standard Biological Parts and Clothes Development</td>
</tr>
<tr>
<td>RNA world website</td>
<td>Compendium of RNA software</td>
<td>RNAworld</td>
</tr>
<tr>
<td>RNA2Norm</td>
<td>RNA secondary structure analysis</td>
<td>RNA2Norm</td>
</tr>
<tr>
<td>RNA modifier</td>
<td>Database search for RNA sequences that match a secondary structure motif</td>
<td>Rutgers Case Group</td>
</tr>
<tr>
<td>RNAstructure</td>
<td>RNA secondary structure analysis</td>
<td>RNAstructure</td>
</tr>
<tr>
<td>RnaViz</td>
<td>RNA secondary structure images</td>
<td>RnaViz</td>
</tr>
<tr>
<td>Rosetta package</td>
<td>Design of protein-binding peptide sequences and protein engineering</td>
<td>Rosetta@home and Rosetta Commons</td>
</tr>
<tr>
<td>RoVerGeNa*</td>
<td>Tool to analyse and tune gene networks</td>
<td>RoVerGeNa</td>
</tr>
<tr>
<td>SynBioSS*</td>
<td>Suite of programs to generate and simulate synthetic biological networks</td>
<td>SynBioSS</td>
</tr>
<tr>
<td>TinkerCell</td>
<td>Synthetic biology CAD program</td>
<td>TinkerCell</td>
</tr>
<tr>
<td>UNAFold software</td>
<td>Nucleic acid folding and hybridization</td>
<td>UNAFold software</td>
</tr>
<tr>
<td>Vienna RNA package</td>
<td>RNA secondary structure</td>
<td>Vienna RNA package</td>
</tr>
</tbody>
</table>
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

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.
Mathematical structures

Definition 1.1.6 Let $\phi$ be a formula with free variables from $\overline{v} = (v_{i_1}, \ldots, v_{i_n})$, 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 $\phi$ 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 $\phi$ 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 $\phi$ is $\neg \psi$, then $\mathcal{M} \models \phi(\overline{a})$ if $\mathcal{M} \nvDash \psi(\overline{a})$.

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

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

vi) If $\phi$ 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 $\phi$ 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 $\text{Ind}(\phi)$ for each formula $\phi(v, \overline{w})$, where $\text{Ind}(\phi)$ is the sentence

(Marker, Model Theory, 2002)
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):
The classical halting probability $\Omega$ introduced by Chaitin is generalized to quantum computations.

Chaitin’s $\Omega$ [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 $\Omega$ is provable “algorithmically incompressible,” it is Martin-Löf/Chaitin/Solovay random. Therefore, $\Omega$ 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 $\Omega$ can be defined as a weighted expectation over all computations of $C$ with classical input $p_i$ ($|p_i|$ stands for the length of $p_i$)

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

(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

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