In-silico Biology: from virtual genomes to virtual cells

Jeremy Gunawardena

Department of Systems Biology
Harvard Medical School
200 Longwood Avenue
Boston, MA 02115

(617) 432 4839
jeremy@hms.harvard.edu
Virtual genomes

BIOLOGICAL OBJECT

MATHEMATICAL OBJECT
Virtual cells

system

BIOLOGICAL OBJECT

model

MATHEMATICAL OBJECT

\[ \frac{d}{dt} \text{ERK}[t] = d_7 \times \text{ERK} \circ \text{MEKpp}[t] + k_8 \times \text{ERKp} \circ \text{ERKphtase}[t] - a_7 \times \text{ERK}[t] \times \text{MEKpp}[t] \]
Models in virtual cells

informal model

- English
- ambiguity
- inconsistency
- incompleteness

formal model

- mathematics
- deducing conclusions from assumptions
Models in virtual cells

There is no “right” model. Models are contingent - they depend on

- the level of mathematical abstraction
- the biological assumptions
- the questions being asked
- the data available

It is better to think of this as “telling a better story” than as “predictive biology”

A model typically describes what happens in a single cell. This may not (and often does not) describe data acquired from cell populations.

We are data poor - we need more data at single cell resolution.
The central problem

Models, as currently constructed, are monolithic and private

They need to become modular and shareable
Other modelling languages, tools, etc

SBML - Systems Biology Markup Language

sbml.org

XML schema for enabling a model to be accessible to different analysis tools

but modularity is not a central concern
Design requirements

Separating biological assumptions from mathematical assumptions

Constructing implied entities

$A \xrightarrow{E} B$

enzymatic reaction

$A + E \xrightarrow{\text{underlying biochemistry}} EA \xrightarrow{} B + E$

Reasoning over existing knowledge

Specifying locations

Naming
little b

Language

Common Lisp macros

defcon
defprop
defmethod
defrule

biological classes and sub-classes

reaction
reaction-type
reactant
reactant-type
location
  compartment
  membrane
  membrane-surface

units

Aneil Mallavarapu
little b

Developed under Lispworks

To be released as open source software, under GPL or LGPL running on the GNU ANSI Common Lisp implementation (CLISP)
Hello World
little b program

(define tmr [reactant-type] :location-class membrane)
(define ligand [reactant-type] :location-class compartment)
(define tmr+ligand [reactant-type] :location-class membrane)
(define sigmol [reactant-type])
(define sigmol* [reactant-type])
(define tmr-ligand-binding [reaction-type {tmr + ligand.(required :c1)} {tmr+ligand} membrane] :k {50 / molar / seconds])
(define tmr-ligand-unbinding [reaction-type {tmr+ligand} {tmr + ligand.(required :c1)} membrane] :k {.05 / seconds])
(define tmr-sigmol-activation [enzymatic-reaction :e {tmr+ligand} :s {sigmol.(required :c2)} :p {sigmol*.required :c2}] :location-class membrane)
(define tmr-sigmol-activation.mechanism [enzymatic-steps] :kf (list {.0001 / molar / seconds} {1 / seconds})
 :kr (list {1000 / seconds} {.0000000000000001 / millimolar / seconds}) :rate-method 'mass-action)

(define test-tube [compartment] :#size [cvar] :val {10 milliliters}]
(define cell-a (with-object [membrane-enclosure]
   {@membrane.size #= [cvar] :@= {4 pi {30 micrometers} ^ 2}}
   {@inner.size #: [cvar] :@= {4/3 pi {30 micrometers} ^ 3}}
   {@membrane.c1 := @outer := test-tube} object))

(defun init-tmr-model (&optional (cell cell-a)
    cell.membrane.(contains tmr)
    cell.outer.(contains ligand)
    cell.inner.(contains sigmol)
    {tmr.t0 := 1 millimoles}
    {ligand.t0 := 1.2 millimoles}
    {sigmol.t0 := 1.4 millimoles}
    (format *standard-output* "use (b-matlab:create-ode-model \"model-name\") to generate a matlab model."
    t)
MATLAB program

function dy = transmembrane_odefn(t,y)
dy = zeros(6,1);
% (_id tmr+ligand.(in cell-a.membrane).MOLES)
dy(1) = 5000.0*y(3)*y(4) + (-884194.1282883076*y(6) + -8.841941282883076E-4*y(2) + -0.05)*y(1) + 1001.0*y(5);
% (_id sigmol*(in cell-a.inner).MOLES)
dy(2) = -8.841941282883076E-4*y(2)*y(1) + 1.0*y(5);
% (_id ligand.(in test-tube).MOLES)
dy(3) = -5000.0*y(3)*y(4) + 0.05*y(1);
% (_id tmr.(in cell-a.membrane).MOLES)
dy(4) = -5000.0*y(3)*y(4) + 0.05*y(1);
% (_id tmr-sigmol-activation.es.1.(in cell-a.membrane).MOLES)
dy(5) = (884194.1282883076*y(6) + 8.841941282883076E-4*y(2))*y(1) + -1001.0*y(5);
% (_id sigmol.(in cell-a.inner).MOLES)
dy(6) = -884194.1282883076*y(6)*y(1) + 1000.0*y(5);
MATLAB output

![Graph of sigmol* (moles) vs time (seconds)]
Summary

How do we formalise biological knowledge by centralisation?

by distribution?

little b is a preliminary attempt towards a distributed solution