Biological Systems as Reactive Systems

Luca Cardelli

Microsoft Research
Cambridge UK

Cambridge 2005-05-04

www.luca.demon.co.uk/BioComputing.htm
50 Years of **Molecular Cell Biology**

- **Genes are made of DNA**
  - Store digital information as sequences of 4 different nucleotides
  - Direct protein assembly through RNA and the Genetic Code

- **Proteins (>10000) are made of amino acids**
  - Process signals
  - Activate genes
  - Move materials
  - Catalyze reactions to produce substances
  - Control energy production and consumption

- **Bootstrappping still a mystery**
  - DNA, RNA, proteins, membranes are today interdependent. Not clear who came first
  - Separation of tasks happened a long time ago
  - Not understood, not essential
Towards Systems Biology

- Biologists now understand many of the cellular components
  - A whole team of biologists will typically study a single protein for years
  - When each component and each reaction is understood, the system is understood (?)

- But this has not led to understand how “the system” works
  - Behavior comes from complex chains of interactions between components
  - Predictive biology and pharmacology still rare
  - Synthetic biology still unreliable

- New approach: try to understand “the system”
  - Experimentally: massive data gathering and data mining (e.g. Genome projects)
  - Conceptually: modeling and analyzing networks (i.e. interactions) of components

- What kind of a system?
  - Just beyond the basic chemistry of energy and materials processing...
  - Built right out of digital information (DNA)
  - Based on information processing for both survival and evolution

- Can we fix it when it breaks?
  - Really becomes: How is information structured and processed?
Storing Processes

• **Today we represent, store, search, and analyze:**
  - Gene sequence data
  - Protein structure data
  - Metabolic network data
  - Signalling pathway data
  - ...

• **How can we represent, store, and analyze biological processes?**
  - Scalable, precise, dynamic, highly structured, maintainable representations for *systems biology*.
  - Not just huge lists of chemical reactions or differential equations.

• **In computing...**
  - There are well-established scalable representations of dynamic reactive processes.
  - They look more or less like little, mathematically based, programming languages.
Eukaryotic Cell
(10~100 trillion in human body)
Membranes everywhere

H. Lodish et al.
Molecular Cell Biology
fourth edition p.1

Structural Architecture

- Nuclear membrane
- Mitochondria
- Golgi
- Vesicles
- E.R.
- Plasma membrane (<10% of all membranes)
Abstract Machines of Systems Biology

The “hardware” (biochemistry) is fairly well understood. But what is the “software” that runs on these machines?
Reactive Systems

- **Modeling biological systems**
  - Not as continuous systems (often highly nonlinear)
  - But as discrete *reactive systems*; abstract machines with:
    - States represent situations
    - Event-driven *transitions* between states represent dynamics
  - The adequacy of describing (discrete) complex systems as reactive systems has been argued convincingly [Harel]

- **Many biological systems exhibit feaures of reactive systems:**
  - Deep layering of abstractions
  - Complex composition of simple components
  - Discrete transitions between states
  - Digital coding and processing of information
  - Reactive information-driven behavior
  - High degree of concurrency and nondeterminism
  - “Emergent behavior” not obvious from part list
Chemistry vs. $\pi$-calculus

A process calculus (chemistry, or SBML)

$Na + Cl \rightarrow_{k_1} Na^+ + Cl^-$
$Na^+ + Cl^- \rightarrow_{k_2} Na + Cl$

This Petri-Net-like graphical representation degenerates into spaghetti diagrams: precise and dynamic, but not scalable, structured, or maintainable.

A compositional graphical representation, and the corresponding calculus.

$Na = !r_{k_1}; ?s_{k_2}; Na$
$Cl = ?r_{k_1}; !s_{k_2}; Cl$

A different process calculus ($\pi$)
Methods

- **Model Construction** *(writing things down precisely)*
  - Formalizing the notations used in systems biology.
  - Formulating description languages.
  - Studying their kinetics (semantics).

- **Model Validation** *(using models for postdiction and prediction)*
  - Simulation from compositional descriptions
    - Stochastic: quantitative concurrent semantics.
    - Hybrid: discrete transitions between continuously evolving states.
  - “Program” Analysis
    - Control flow analysis
    - Causality analysis
  - Model checking
    - Standard, Quantitative, Probabilistic
Basic Modeling Guidelines

- Regev-Shapiro: “Molecules as Computation”:

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction capability</td>
<td>Channel</td>
</tr>
<tr>
<td>Interaction</td>
<td>Communication</td>
</tr>
<tr>
<td>Modification (of chemical components)</td>
<td>State change (state-transition systems)</td>
</tr>
</tbody>
</table>

Cellular Abstractions: Cells as Computation
Regev&Shapiro NATURE vol 419, 2002-09-26, 343

- They chose $\pi$-calculus and adapted it with stochastic features
  - To match the stochastic aspects of (bio)chemistry
  - Many probabilistic process calculi predate them, but only Hillston (CSP) and Priami ($\pi$) had already studied stochastic calculi.
π-calculus Executive Summary

- **It’s for:**
  - The modular description of concurrent, nondeterministic systems
  - Study of such systems based on their descriptions

- **It’s got:**
  - Processes
  - Channels
    - A minimalistic syntax (it’s a language and also a model)

- **You can:**
  - Fork new processes
  - Create new channels
  - Do I/O over channels (synchronous and asynchronous)
    including passing channels over channels
  - Make nondeterministic choices
  - Define processes recursively

- **That’s it.**
  - Except for extensive model theory and metatheory.
  - Cannot pass processes over channels
    (simulated by passing channels to them)
  - Cannot define procedures
    (simulated by supplying reply channels)
π-calculus

Syntax

\[ \pi ::= x(y) \quad \text{receive } y \text{ along } x \]
\[ \overline{x}(y) \quad \text{send } y \text{ along } x \]

\[ P ::= 0 \mid \sum_{i \in I} \pi_i \cdot P_i \mid [x = y] \cdot P \mid P_1 \cdot P_2 \mid (\text{new } x) \cdot P \mid 1 \cdot P \]

Structural congruence

Renaming of bound variables

\[ x(y) \cdot P = x(z) \cdot (\{z/y\} \cdot P) \quad \text{if } z \notin FN(P) \]
\[ \text{new } y \cdot P = \text{new } z \cdot (\{z/y\} \cdot P) \quad \text{if } z \notin FN(P) \]

Structural congruence laws

\[ P \cdot Q \equiv Q \cdot P \quad \text{commutativity of parallel composition} \]
\[ (P \cdot Q) \cdot R \equiv P \cdot (Q \cdot R) \quad \text{associativity of parallel composition} \]
\[ P + Q \equiv Q + P \quad \text{commutativity of summation} \]
\[ (P + Q) + R \equiv P + (Q + R) \quad \text{associativity of summation} \]
\[ \text{new } x \cdot 0 \equiv 0 \quad \text{restriction of inert processes} \]
\[ (\text{new } x)(\text{new } y) \cdot P \equiv (\text{new } y)(\text{new } x) \cdot P \quad \text{polyadic restriction} \]
\[ (\text{new } x) \cdot P \mid Q \equiv (\text{new } x) \cdot (P \cdot Q) \quad \text{if } x \notin FN(Q) \quad \text{scope extrusion} \]
\[ !P \equiv P \cdot !P \quad \text{replication} \]

Reaction rules

\[ (\cdots + \overline{x}(z) \cdot Q)(\cdots + x(y) \cdot P) \rightarrow Q \cdot P \{z/y\} \quad \text{communication (COMM)} \]

\[ \frac{P \rightarrow P'}{P \cdot Q \rightarrow P' \cdot Q} \quad \text{reaction under parallel composition (PAR)} \]

\[ \frac{P \rightarrow P'}{(\text{new } x) \cdot P \rightarrow (\text{new } x) \cdot P'} \quad \text{reaction under restriction (RES)} \]

\[ Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q' \quad \text{structural congruence (STRUCT)} \]
Stochastic $\pi$-calculus Executive Summary

- A simple variant of $\pi$-calculus:
  - Channels have stochastic "firing" rates with exponential distribution.
  - Nondeterministic choice becomes stochastic race.
  - Cuts down to CTMCs (Continuous Time Markov Chains) in the finite case (not always). Then, standard analytical tools are applicable.
  - Can be given friendly automata-like scalable graphical syntax (work in progress: Andrew Phillips).
  - Is directly executable (e.g. via the Gillespie algorithm from physical chemistry).
  - Is analyzable (large body of literature, at least in the non-stochastic case).

Figure 2. Regulating Gene Expression by Positive Feedback [6]

Figure 3. Protein A molecules v.s. time in presence (left) and absence (right) of TF A. Phillips, L. Cardelli. BioConcur'04.
Importance of Stochastic Effects

- **A deterministic system:**
  - May get “stuck in a fixpoint”.
  - And hence **never oscillate**.

- **A similar stochastic system:**
  - May be “thrown off the fixpoint” by stochastic noise, entering a long orbit that will later bring it back to the fixpoint.
  - And hence **oscillate**.

Surprisingly enough, we have found that parameter values that give rise to a stable steady state in the deterministic limit continue to produce reliable oscillations in the stochastic case, as shown in Fig. 5. Therefore, the presence of noise not only changes the behavior of the system by adding more disorder but can also lead to marked qualitative differences.

**Mechanisms of noise-resistance in genetic oscillators**

José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, Stanislas Leibler

PNAS April 30, 2002 vol. 99 no. 9 p.5991

---

**Fig. 5.** Time evolution of $\dot{R}$ for the deterministic Eq. [1] (a) and stochastic (b) versions of the model. The values of the parameters are as in the caption of Fig. 1, except that now we set $\tau = 0.05\, \text{h}^{-1}$. For these parameter values, $\tau < 0$, so that the fixed point is stable.

**Fig. 6.** Phase portrait as in Fig. 4 but for a situation in which the system falls into the stable fixed point $(R_0, C_0)$. The dotted arrow to the left of the fixed point illustrates a perturbation that would initiate a single sweep of the (former) oscillatory trajectory.
Gene Networks
The Gene Machine

The “Central Dogma” of Molecular Biology

DNA

4-letter digital code

messenger RNA

4-letter digital code

PROTEIN

20-letter digital code

SYSTEMS

50,000 (?) shapes

regulation

transcription

translation

interaction

Lactose Operon

Operator

Z gene

S gene

R gene

β-galactosidase

Permease

Transacetylase

mRNA

RNA polymerase

Lactose

lac repressor

Gene space

Gene 1

Gene 2

Gene 3

Gene 4

Protein space

Protein 1

Protein 2

Protein 3

Protein 4

Complex 34

Meiabolite 1

Metabolite 2

DNA Tutorial
The Gene Machine “Instruction Set”

Regulation of a gene (positive and negative) influences transcription. The regulatory region has precise DNA sequences, but not meant for coding proteins: meant for binding regulators.

Transcription produces molecules (RNA or, through RNA, proteins) that bind to regulatory region of other genes (or that are end-products).

Human (and mammalian) Genome Size
3Gbp (Giga base pairs) 750MB @ 4bp/Byte (CD)
  Non-repetitive: 1Gbp 250MB
  In genes: 320Mbp 80MB
Coding: 160Mbp 40MB
  Protein-coding genes: 30,000-40,000

M.Genitalium (smallest true organism)
  580,073bp 145KB (eBook)
E.Coli (bacteria): 4Mbp 1MB (floppy)
Yeast (eukarya): 12Mbp 3MB (MP3 song)
Wheat 17Gbp 4.25GB (DVD)

cf. Hybrid Petri Nets [Matsuno, Doi, Nagasaki, Miyano]
Gene Composition

Is a shorthand for:

Under the assumptions [Kim & Tidor]
1) The solution is well-stirred
   (no spatial dependence on concentrations or rates).
2) There is no regulation cross-talk.
3) Control of expression is at transcription level only
   (no RNA-RNA or RNA-protein effects)
4) Transcriptions and translation rates monotonically
    affect mRNA and protein concentrations resp.

Ex: Bistable Switch

Ex: Oscillator

Expressed
Repressed
Expressing
Gene Regulatory Networks

http://strc.herts.ac.uk/bio/maria/NetBuilder/


(The Classical ODE Approach)

[Chen, He, Church]

\[
\frac{d r}{d t} = f(p) - Vr \\
\frac{d p}{d t} = Lr - Ur
\]

- \(r\) mRNA concentrations (n-dim vector)
- \(p\) protein concentrations (n-dim vector)
- \(f(p)\) transcription functions:
  (n-dim vector polynomials on \(p\))
**Nullary Gate**

- **null** (no input) → **b**
- **spontaneous** ("constitutive") output

**null(b) ≡ \( \tau_\epsilon; \ (tr(b) \mid null(b)) \)**

- **interaction site of output protein**
- **stochastic delay** (\( \tau \)) with rate \( \epsilon \) of constitutive transcription
- **(recursive, parametric) process definition**
- **output protein** (transcription factor), spawn out
- **and repeat**

---

*A stochastic rate \( r \) is always associated with each channel \( a_r \) (at channel creation time) and delay \( \tau_r \), but is often omitted when unambiguous.*
Production and Degradation

Degradation is extremely important and often deliberate; it changes unbounded growth into (roughly) stable signals.

\[ \text{transcription factor} \rightarrow \text{tr}(p) \triangleq (|p_r; \text{tr}(p)) + \tau_\delta \]

\[ \text{interaction site of transcription factor} \rightarrow \text{degradation} \]

\[ \text{stochastic choice (race between } r \text{ and } \delta) \rightarrow \text{degradation rate } \delta \]

\[ (\text{output, } l) \text{ interaction with rate } r \]

\[ (\text{input, } ?) \text{ is on the target gene} \]

A transcription factor is a process (not a message or a channel): it has behavior such as interaction on \( p \) and degradation.

\[ \text{null(b)} \triangleq \tau_c; (\text{tr(b)} \mid \text{null(b)}) \]

combined effect of production and degradation (without any interaction on \( b \))

interaction offers on \( b \)

(\( \geq \) number of tr processes)

null(b)
Unary Pos Gate

\[ \text{pos}(a, b) \triangleq \begin{cases} \ ? &; \tau_{r} \cdot (\text{tr}(b) \mid \text{pos}(a, b)) \ + \\ \tau_{\varepsilon} \cdot (\text{tr}(b) \mid \text{pos}(a, b)) \end{cases} \]

- (input, ?) interaction with rate \( r \)
- or constitutive transcription to always get things started
- parallel, not sequence, to handle self-loops without deadlock
- transcription delay with rate \( \eta \)
- race between \( r \) and \( \varepsilon \)
- output protein
- unlimited amount of

\[ *\text{tr}(a_{r}) \mid \text{pos}(a_{r}, b) \]

\[ \text{pos}(a, b) \]
Unary Neg Gate

Input (inhibitory) → \( a \) → \( b \) → Output (constitutive when not inhibited)

\( \text{neg}(a,b) \triangleq r: \tau_n: \text{neg}(a,b) + \tau_e: (\text{tr}(b) \mid \text{neg}(a,b)) \)

(inhibitory, \(?\)) interaction with rate \( r \)
or constitutive transcription to always get things started

Inhibition delay with rate \( \eta \)
Race between \( r \) and \( \varepsilon \)

Graph showing \( b \) over time with constitutive and inhibited states.

\( \text{neg}(a_r,b) \)
\( \ast \text{tr}(a_r) \mid \text{neg}(a_r,b) \)
## Signal Amplification

Let's consider a system where we have a sequence of reactions involving positions (a, b, c) corresponding to different states or molecules. The reactions can be described as follows:

- **pos(a,b)** and **pos(b,c)**

The dynamics of the system can be described through the following equations:

\[ \text{pos}(a,b) \triangleq \text{?}_{\alpha}; \tau_{\eta}; (\text{tr}(b) \mid \text{pos}(a,b)) + \tau_{\epsilon}; (\text{tr}(b) \mid \text{pos}(a,b)) \]

\[ \text{tr}(p) \triangleq (\text{!}_{p_r}; \text{tr}(p)) + \tau_{\delta} \]

With little degradation:

\[ r=1.0, \kappa=0.01, \eta=0.1, \delta=0.00001 \]

Even with no a input, constitutive production of b gets amplified to a high c signal.

E.g. 1 a that interacts twice before decay can produce 2 b that each interact twice before decay, which produce 4 c...
Signal Normalization

\[ \text{neg}(a,b) \mid \text{neg}(b,c) \]

\[ \begin{align*}
\text{neg}(a,b) & \triangleq \\
& ?a_r; \tau_h; \text{neg}(a,b) + \\
& \tau_{\varepsilon}; (\text{tr}(b) \mid \text{neg}(a,b))
\end{align*} \]

\[ \text{tr}(p) \triangleq (!p_r; \text{tr}(p)) + \tau_\delta \]

**30*-tr(a) | neg(a,b) | neg(b,c)**

- a non-zero input level, a, whether weak or strong, is renormalized to a standard level, c.
Self Feedback Circuits

**pos(a,a)**

\[ \text{pos}(a,b) \triangleq \]
\[ ?a_r; (\text{tr}(b) \mid \text{pos}(a,b)) + \]
\[ \tau_{e}; (\text{tr}(b) \mid \text{pos}(a,b)) \]

\[ \text{tr}(p) \triangleq (\lnot p_r; \text{tr}(p)) + \tau_\delta \]

*(Can overwhelm degradation, depending on parameters)*

**neg(a,a)**

\[ \text{neg}(a,b) \triangleq \]
\[ ?a_r; \tau_h; \text{neg}(a,b) + \]
\[ \tau_{e}; (\text{tr}(b) \mid \text{neg}(a,b)) \]

\[ \text{tr}(p) \triangleq (\lnot p_r; \text{tr}(p)) + \tau_\delta \]

*high, to raise the signal*

---

![Graph showing pos(a,a) and neg(a,a) with parameters](image)
Two-gate Feedback Circuits

Monostable:

For some degradation rates is quite stable:

\[ r = 1.0, \epsilon = 0.1, h = 0.01, \delta = 0.0005 \]

But with a small change in degradation, it goes wild:

\[ r = 1.0, \epsilon = 0.1, h = 0.01, \delta = 0.0001 \]

Bistable:

5 runs with \( r(a) = 0.1 \), \( r(b) = 1.0 \) shows that circuit is now biased towards expressing \( b \).
Repressilator

\[ \text{neg}(a,b) \mid \text{neg}(b,c) \mid \text{neg}(c,a) \]

Same circuit, three different degradation models by changing the tr component:

- \[ \text{tr}(p) \equiv !p_r \]
  - interact once and die
  - otherwise stick around
  - \( r=1.0, \varepsilon=0.1, h=0.04 \)

- \[ \text{tr}(p) \equiv !p_r + \tau_\delta \]
  - interact once and die
  - otherwise decay
  - \( r=1.0, \varepsilon=0.1, h=0.04, \delta=0.0001 \)

- \[ \text{tr}(p) \equiv (!p_r; \text{tr}(p)) + \tau_\delta \]
  - interact many times
  - and decay
  - \( r=1.0, \varepsilon=0.1, h=0.001, \delta=0.001 \)

Subtle... at any point one gate is inhibited and the other two can fire constitutively. If one of them fires first, nothing really changes, but if the other one fires first, then the cycle progresses.
Repressilator in SPiM

val dk = 0.001  (* Decay rate *)
val eta = 0.001  (* Inhibition rate *)
val cst = 0.1  (* Constitutive rate *)

let tr(p:chan()) =
  do !p; tr(p)
  or delay@dk

let neg(a:chan(), b:chan()) =
  do ?a; delay@eta; neg(a,b)
  or delay@cst; (tr(b) | neg(a,b))

(* The circuit *)
val bnd = 1.0  (* Protein binding rate *)
new a@bnd: chan()
new b@bnd: chan()
new c@bnd: chan()

run (neg(c,a) | neg(a,b) | neg(b,c))
Repressilator ODE Model and Simulation

Bruce E Shapiro
Cellerator

\[
\begin{align*}
\frac{d[X]}{dt} &= \alpha_0 + \frac{\alpha+\alpha_i[PY]^n}{K^n+[PY]^n} - k[X], \quad \frac{d[PY]}{dt} = \beta([X] - [PX]) \\
\frac{d[Y]}{dt} &= \alpha_0 + \frac{\alpha+\alpha_i[PZ]^n}{K^n+[PZ]^n} - k[Y], \quad \frac{d[PY]}{dt} = \beta([Y] - [PY]) \\
\frac{d[Z]}{dt} &= \alpha_0 + \frac{\alpha+\alpha_i[PX]^n}{K^n+[PX]^n} - k[Z], \quad \frac{d[PZ]}{dt} = \beta([Z] - [PZ])
\end{align*}
\]

We can model an inducer like aTc as something that competes for the transcription factor.

IPTG de-represses the lac operon, by binding to the lac repressor (the lac I gene product), preventing it from binding to the operator.
Protein Networks

MAPK Cascade - Huang&Ferrell

Table 2. Predicted Hill coefficients for MAP kinase cascade components. Varying the assumed $K_m$ values

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Range of assumed $K_m$ values</th>
<th>Range of effective Hill coefficients (nM) predicted for</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MAPKKE → MAPKKK$^*$</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7 → 4.9</td>
</tr>
<tr>
<td>2. MAPKKK$^*$ → MAPKKK</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7 → 4.9</td>
</tr>
<tr>
<td>3. MAPKK → MAPKK-P</td>
<td>60–1500 nM</td>
<td>1.0 → 1.3–2.3 → 4.0–5.1</td>
</tr>
<tr>
<td>4. MAPKK-P → MAPKK</td>
<td>60–1500 nM</td>
<td>1.0 → 1.5–1.9 → 3.6–6.7</td>
</tr>
<tr>
<td>5. MAPKK-P → MAPK-PP</td>
<td>60–1500 nM</td>
<td>1.0 → 1.2–2.4 → 3.8–5.2</td>
</tr>
<tr>
<td>6. MAPK-PP → MAPKK-P</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7–1.8 → 4.1–6.4</td>
</tr>
<tr>
<td>7. MAPK → MAPKK-P</td>
<td>60–1500 nM (300 nM$^3$)</td>
<td>1.0 → 1.7 → 3.6–6.2</td>
</tr>
<tr>
<td>8. MAPK-P → MAPKK</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7 → 4.3–5.2</td>
</tr>
<tr>
<td>9. MAPKK-P → MAPKK-PP</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7 → 3.4–6.1</td>
</tr>
<tr>
<td>10. MAPKK-PP → MAPKK</td>
<td>60–1500 nM</td>
<td>1.0 → 1.7 → 4.7–5.1</td>
</tr>
</tbody>
</table>

The assumed $K_m$ values for each reaction were individually varied over the ranges shown, with the assumed $K_m$ values for the other nine reactions held constant. The effective Hill coefficients were calculated from the steepness of the predicted stimulus/response curves, as described in the text.

The $K_m$ value for reaction 7 has been measured to be 300 nM for the phosphorylation of a mammalian MAPK by a MAPKK (N. Ahn, personal communication). All of the other $K_m$ values were initially assumed to be 300 nM as well.

Calculations. Eqs. 1-10 represent the reactions of the MAP kinase cascade, which are shown schematically in Fig. 1. We have used Goldbeter's and Keating's nomenclature for the rate constants—the letter a denotes association, d denotes dissociation without catalysis, and k denotes product formation (11). KKK denotes MAPKK; K denotes MAPK; and K denote MAPK.

\[
\begin{align*}
K_{11} + E_1 &\rightleftharpoons KKK+ E_1 & k_1 &\rightarrow  KKK+ E_1 & \text{(1)} \\
K_{12} + E_2 &\rightleftharpoons KKK+ E_2 & k_2 &\rightarrow  KKK+ E_2 & \text{(2)} \\
K + KKK &\rightleftharpoons KKK + K & k_3 &\rightarrow  KKK + K & \text{(3)} \\
KK + KKK &\rightleftharpoons KK+KK + K & k_4 &\rightarrow  KK+KK + K & \text{(4)} \\
K + P &\rightleftharpoons K + P & k_5 &\rightarrow  K + P & \text{(5)} \\
K + P &\rightleftharpoons K + P & k_6 &\rightarrow  K + P & \text{(6)} \\
K + KPP &\rightleftharpoons K + KPP & k_7 &\rightarrow  K + KPP & \text{(7)} \\
K + KPP &\rightleftharpoons K + KPP & k_8 &\rightarrow  K + KPP & \text{(8)} \\
K + KPP &\rightleftharpoons K + KPP & k_9 &\rightarrow  K + KPP & \text{(9)} \\
K + KPP &\rightleftharpoons K + KPP & k_{10} &\rightarrow  K + KPP & \text{(10)} \\
\end{align*}
\]

Fig. 1. Schematic view of the MAPK cascade. Activation of MAPK depends upon the phosphorylation of two conserved sites [Thr-183 and Tyr-185 in rat p42 MAPK/Erk2 (4, 5)]. Full activation of MAPKK also requires phosphorylation of two sites [Ser-218 and Ser-222 in mouse Mek-1/MKK1 (6–10)]. Detailed mechanisms for the activation of various MAPKKs (e.g., Raf-1, B-Raf, Mos) are not yet established; here we assume that MAPKKs are activated and inactivated by enzymes we denote E1 and E2. MAPKK$^*$ denotes activated MAPKK. MAPKK-P and MAPKK-PP denote singly and doubly phosphorylated MAPKK, respectively. MAPK-P and MAPK-PP denote singly and doubly phosphorylated MAPK. Pase denotes phosphatase.
As 18 Ordinary Differential Equations  
Plus 7 conservation equations

The 10 reactions described above give rise to 18 rate equations.

One equation for each species (8) and complex (10), but not for constant concentration enzymes (4)

In addition, there are seven conservation equations (Eqs. 29-35).

\[ \text{[KKK]} = [KKK] + [KKK^*] + [KKK\cdot E1] \]
\[ + [KKK^* \cdot E2] \]
\[ + [KKK^* \cdot K] + [KKK^* \cdot K-P] \]

in exactly one state

Each molecule

2005-05-04 35
The Circuit
Enzymatic Reactions

Reaction View

\[
\begin{aligned}
E & \xrightarrow{(c,d,e)} S \\
\end{aligned}
\]

\[
\begin{aligned}
E + S & \xrightarrow{c} ES \\
ES & \xrightarrow{d} E + S \\
ES & \xrightarrow{e} P + E
\end{aligned}
\]

Interaction View

\[
\begin{aligned}
S & \xrightarrow{a_c} E \\
E & \xrightarrow{u_d} P \\
E & \xrightarrow{k_e} S
\end{aligned}
\]

- bind
- unbind
- react

\[
\begin{aligned}
S() & \triangleq \text{new u@d;}
\!a_c(u); (\!u_d; S() + \!k_e; P())
\end{aligned}
\]

\[
\begin{aligned}
E() & \triangleq \?a_c(u); (\?u_d; E() + \?k_e; E())
\end{aligned}
\]

\[
\begin{aligned}
P() & \triangleq ...
\end{aligned}
\]

private binding between one S and one E molecule
let KKK() =
  (new u1@d1:Release
    !a1(u1); (do !u1;KKK() or !k1;KKKst())
  )
and KKKst() =
  (new u2@d2:Release
    do !a2(u2); (do !u2;KKKst() or !k2;KKK())
    or ?a3(u3); (do ?u3;KKKst() or ?k3;KKKst())
    or ?a5(u5); (do ?u5;KKKst() or ?k5;KKKst())
  )

let E1() =
  ?a1(u1); (do ?u1;E1() or ?k1;E1())

let E2() =
  ?a2(u2); (do ?u2;E2() or ?k2;E2())

let KK() =
  (new u3@d3:Release
    !a3(u3); (do !u3;KK() or !k3;KK_P())
  )
and KK_P() =
  (new u6@d6:Release
    do !a6(u6); (do !u6;KK_P() or !k6;KK_P())
    or ?a7(u7); (do ?u7;KK_P() or ?k7;KK_P())
    or ?a9(u9); (do ?u9;KK_P() or ?k9;KK_P())
  )

let K() =
  (new u7@d7:Release
    !a7(u7); (do !u7;K() or !k7;K_P())
  )
and K_P() =
  (new u8@d8:Release new u9@d9:Release
    do !a8(u8); (do !u8;K_P() or !k8;K())
    or !a9(u9); (do !u9;K_P() or !k9;K_PP())
  )
and K_PP() =
  (new u10@d10:Release
    do ?a10(u10); (do ?u10;K_PP() or ?k10;K_P())
  )
and KPse() =
  do ?a8(u8); (do ?u8;KPse() or ?k8;KPse())
  or ?a10(u10); (do ?u10;KPse() or ?k10;KPse())
type Release = chan()

new a1@1.0:Bond val d1=1.0 new k1@1.0:React
new a2@1.0:Bond val d2=1.0 new k2@1.0:React
new a3@1.0:Bond val d3=1.0 new k3@1.0:React
new a4@1.0:Bond val d4=1.0 new k4@1.0:React
new a5@1.0:Bond val d5=1.0 new k5@1.0:React
new a6@1.0:Bond val d6=1.0 new k6@1.0:React
new a7@1.0:Bond val d7=1.0 new k7@1.0:React
new a8@1.0:Bond val d8=1.0 new k8@1.0:React
new a9@1.0:Bond val d9=1.0 new k9@1.0:React
new a10@1.0:Bond val d10=1.0 new k10@1.0:React

... run 100 of KKK() run 100 of KK() run 100 of K()
run 1 of E2() run 1 of KKPse() run 1 of KPse()
run 1 of E1()
MAPK Cascade Simulation in SPiM

KKK
KK
K

KK-P
K-P

KKK* reacts
KK-PP rises quicker
K-PP flips up to 100!

KKK*
KK-PP
K-PP down slowly

1xE1 removed
1xE1 injected

1xE1 injected

All coefficients 1.0 !!!
100xKK, 100xKK, 100xK,
1xE2, 1xKPse, 1xKPse.

Input is 1xE1.
Output is 100xK-PP (ultrasensitivity).

2005-05-04 40
MAPK Cascade Simulation in SPiM

All coefficients 1.0 !!!
100xKKK, 100xKK, 100xK,
10xE2, 10xKPkse, 10xKPse.
(so 1xE1 is no longer sufficient to produce an output)
Model Validation
Model Validation: Simulation

- **Basic stochastic algorithm: Gillespie**
  - Exact (i.e. based on physics) stochastic simulation of chemical kinetics.
  - Can compute concentrations and reaction times for biochemical networks.

- **Stochastic Process Calculi**
  - **BioSPI** [Shapiro, Regev, Priami, et. al.]
    - Stochastic process calculus based on Gillespie.
  - **BioAmbients** [Regev, Panina, Silverma, Cardelli, Shapiro]
    - Extension of BioSPI for membranes.
  - **Case study: Lymphocytes in Inflamed Blood Vessels** [Lecca, Priami, Quaglia]
    - Original analysis of lymphocyte rolling in blood vessels of different diameters.
  - **Case study: Lambda Switch** [Celine Kuttler, IRI Lille]
    - Model of phage lambda genome (well-studied system).
  - **Case study: VICE** [U. Pisa]
    - Minimal prokaryote genome (180 genes) and metabolism of whole VIrtual CELL, in stochastic $\pi$-calculus, simulated under stable conditions for 40K transitions.

- **Hybrid approaches**
  - **Charon language** [UPenn]
    - Hybrid systems: continuous differential equations + discrete/stochastic mode switching.
  - Etc.
Model Validation: “Program” Analysis

- **Causality Analysis**
  - *Biochemical pathways*, ("concurrent traces" such as the one here), are found in biology publications, summarizing known facts.
  - This one, however, was automatically generated from a program written in BioSpi by comparing traces of all possible interactions. [Curti, Priami, Degano, Baldari]
  - One can play with the program to investigate various hypotheses about the pathways.

- **Control Flow Analysis**
  - Flow analysis techniques applied to process calculi.
  - Overapproximation of behavior used to answer questions about what “cannot happen”.
  - Analysis of positive feedback transcription regulation in BioAmbients [Flemming Nielson].

- **Probabilistic Abstract Interpretation**
  - [DiPierro Wicklicky].
Model Validation: Modelchecking

- **Temporal**
  - Software verification of biomolecular systems (NA pump) [Ciobanu]
  - Analysis of mammalian cell cycle (after Kohn) in CTL. [Chabrier-Rivier Chiaverini Danos Fages Schachter]
    - E.g. is state $S_1$ a necessary checkpoint for reaching state $S_2$?

- **Quantitative: Simpathica/xssys** [Antioniotti Park Policriti Ugel Mishra]
  - Quantitative temporal logic queries of human Purine metabolism model.
    - Eventually(Always (PRPP = 1.7 * PRPP1) implies steady_state() and Eventually(Always(IMP < 2 * IMP1)) and Eventually(Always(hx_pool < 10*hx_pool1)))

- **Stochastic: Spring** [Parker Normal Kwiatkowska]
  - Designed for stochastic (computer) network analysis
    - Discrete and Continuous Markov Processes.
    - Process input language.
    - Modelchecking of probabilistic queries.
What Reactive Systems Do For Us

We can write things down precisely
- We can modularly describe high structural and combinatorial complexity ("do programming").

We can calculate and analyze
- Directly support simulation.
- Support analysis (e.g. control flow, causality, nondeterminism).
- Support state exploration (modelchecking).

We can visualize
- Automata-like presentations.
- Petri-Net-like presentations.
- State Charts, Live Sequence Charts [Harel]
  - Hierarchical automata.
  - Scenario composition.

We can reason
- Suitable equivalences on processes induce algebraic laws.
- We can relate different systems (e.g. equivalent behaviors).
- We can relate different abstraction levels.
- We can use equivalences for state minimization (symmetries).

Disclaimers
- Some of these technologies are basically ready (medium-scale stochastic simulation and analysis, medium-scale nondeterministic and stochastic modelchecking).
- Others need to scale up significantly to be really useful. This is (has been) the challenge for computer scientists.

Many approaches, same basic philosophy, tools being built:
Conclusions

Q: “The data are accumulating and the computers are humming, what we are lacking are the words, the grammar and the syntax of a new language...”
D. Bray (TIBS 22(9):325-326, 1997)

A: “The most advanced tools for computer process description seem to be also the best tools for the description of biomolecular systems.”
E. Shapiro (Lecture Notes)
References


Papers

*BioAmbients*
- a stochastic calculus with compartments.

*Brane Calculi*
- process calculi with computation “on” the membranes, not inside them.

*Bitonal Systems*
- membrane reactions and their connections to “local” patch reactions.

*Abstract Machines of Systems Biology*
- the abstract machines implemented by biochemical toolkits.

www.luca.demon.co.uk/BioComputing.htm