Biological Networks in Stochastic $\pi$-Calculus

Luca Cardelli

Microsoft Research
Cambridge UK

Imperial College 2005-02-17

www.luca.demon.co.uk
50 Years of Molecular Cell Biology

- How cells work:
  - DNA stores information
  - DNA instructs Ribosomes via RNA to assemble Proteins
  - Proteins (>10000) do things:
    - Process signals, activate DNA
    - Catalyze reactions to produce substances
    - Control energy production and consumption
  - Bootstrappping still a mystery
    - Happened a long time ago; not understood, not essential.
Biologists now understand many cellular components, but do not yet understand how “the system” works.

**Molecular Biology**: Understanding the components of living matter.

**BioInformatics**: Mining -omics “high-throughput” whole-system data.

**Systems Biology**: Understanding the connectivity of the components.

**Aim**: Modeling **biological systems**

not as continuous systems (traditional)
but as **reactive systems** (information-processing)

**Because they have some similar features:**

- Deep layering of abstractions.
- Complex composition of simpler components.
- Discrete (non-linear) transitions.
- Digital coding of information.
- Reactive information-driven behavior.
- Very high degree of concurrency.
- "Emergent behavior" (not obvious from part list).
Methods

Model Construction (writing things down precisely)

Studying the notations used in systems biology.
Formulating description languages, for various purposes.
Studying their kinetics (semantics).

Model Validation (using models for postdiction and prediction)

Stochastic Simulation

Stochastic = Quantitative concurrent semantics.
Based on compositional descriptions.

“Program” Analysis

Control flow analysis
Causality analysis

Modelchecking

Standard, Quantitative, Probabilistic
Functional Architecture of Cellular Systems

Abstract Machines of Molecular Biology
- Biochemical Networks - The Protein Machine
- Gene Regulatory Networks - The Gene Machine
- Transport Networks - The Membrane Machine

Systems Biology
1. “How do components interact?”
2. “Gather high-throughput data.”

Glycan Machine
Sugars
Surface and Extracellular Features

Protein Machine
Aminoacids
Metabolism, Propulsion
Signal Processing
Molecular Transport

Gene Machine
Nucleotides
Regulation
Holds genome(s), confines regulators
Model Integration
Different time and space scales
Holds receptors, actuators, hosts reactions
Directs membrane construction and protein embedding
Makes proteins, where/when/how much
Signals conditions and events

Membrane Machine
Phospholipids
Confinement
Storage
Bulk Transport
Implements fusion, fission
Stochastic \( \pi \)-calculus Executive Summary

- **A process calculus:**
  - The modular representation of concurrent (and stochastic) processes of all kinds.
  - Cuts down to CTMCs in the finite case (not always), then standard tools are applicable.
  - Can be given friendly automata-like scalable graphical syntax (work in progress).
- Is directly executable (e.g. via Gillespie).
- Is analyzable (large body of literature, at least in the non-stochastic case).

![Diagram of stochastic \( \pi \)-calculus processes](image)

**Figure 2.** Regulating Gene Expression by Positive Feedback [8]

**Figure 3.** Protein A molecules v.s. time in presence (left) and absence (right) of TF
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</td>
<td>State change</td>
</tr>
<tr>
<td>(of chemical components)</td>
<td>(state-transition systems)</td>
</tr>
</tbody>
</table>

Cellular Abstractions: Cells as Computation
Regev&Shapiro NATURE vol 419, 2002-09-26, 343
\( \pi \)-calculus

**Syntax**

\[
\begin{align*}
\pi & ::= \pi(x) \quad \text{receive } y \text{ along } x \\
& \quad \pi(y) \quad \text{send } y \text{ along } x \\
\end{align*}
\]

\[
P ::= 0 \mid \sum_{x \in \pi} \pi_i.P_i \mid [x = y] P \mid P_1 | P_2 \mid (\text{new } x)P \mid IP
\]

**Structural congruence**

**Renaming of bound variables**

\[
\begin{align*}
x(y).P &= x(z).\{z/y\}P & \text{if } z \notin FN(P) \\
\text{(new } y).P &= \text{(new } z).\{z/y\}P & \text{if } z \notin FN(P)
\end{align*}
\]

**Structural congruence laws**

\[
\begin{align*}
P|Q &= Q|P & \text{commutativity of parallel composition} \\
(P|Q)|R &= P|(Q|R) & \text{associativity of parallel composition} \\
P + Q &= Q + P & \text{commutativity of summation} \\
(P + Q) + R &= P + (Q + R) & \text{associativity of summation} \\
0 + P &= P & \text{restriction of inert processes} \\
0 &= (\text{new } x)0 & \text{polyadic restriction} \\
\text{(new } x)(\text{new } y)P &= (\text{new } y)(\text{new } x)P & \text{scope extrusion} \\
\text{(new } x)P|Q &= (\text{new } x)(P|Q) & \text{if } x \notin FN(Q) \\
!P &= P!P & \text{replication}
\end{align*}
\]

**Reaction rules**

\[
\begin{align*}
&\text{(\cdots + } \pi(z).Q\mid (\cdots + x(y).P) \rightarrow Q|P\{z/y\} & \text{communication (COMM)} \\
\end{align*}
\]

\[
\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} & \text{reaction under parallel composition (PAR)} \\
\frac{P \rightarrow P'}{(\text{new } x)P \rightarrow (\text{new } x)P'} & \text{reaction under restriction (RES)} \\
\frac{Q \equiv P P' \rightarrow P'|P' \equiv Q'}{Q \rightarrow Q'} & \text{structural congruence (STRUCT)}
\]
Stochastic $\pi$-calculus

- Stochastic extension of p-calculus. [C.Priami]

  Associate a rate $r \in (0, \infty]$ of an exponential distribution to each activity $a$; it describes the stochastic behavior of the activity

  $$a;P \text{ is replaced by } a@r;P$$

  Exponential distribution guarantees the memoryless property: the time at which a change of state occurs is independent of the time at which the last change of state occurred.

  Race condition is defined in a probabilistic competitive context: all the activities that are enabled in a state compete and the fastest one succeeds.

Gene Networks
The Gene Machine

The “Central Dogma” of Molecular Biology

- 4-letter digital code
- 4-letter digital code
- 20-letter digital code
- 50,000(?) shapes

Lactose Operon

Metabolic space

Protein space

Gene space

Taken from Pedro Mendes
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)
Gene Regulatory Networks

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


(The Classical ODE Approach)

\[ \frac{dr}{dt} = f(p) - Vr \]
\[ \frac{dp}{dt} = L \cdot r - Ur \]

- \( n \): number of genes
- \( r \): mRNA concentrations (n-dim vector)
- \( p \): protein concentrations (n-dim vector)

\( f(p) \): transcription functions:
- (n-dim vector polynomials on \( p \))

[Chen, He, Church]
A stochastic rate $r$ is always associated with each channel $a_r$ (at 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.

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

and repeat

degradation

degradation rate $\delta$

(output, l) interaction with rate $r$
(input, ?, 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.

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

product

null($b$) $\triangleq \tau_e \cdot (\text{tr}(b) | \text{null}(b))$

interaction offers on $b$
($= \text{number of tr processes}$)

null($b$)

interaction on $b$

$\epsilon=0.1$, $\delta=0.001$

time

null($b$)
Unary Pos Gate

- **input (excitatory)**
  - $(a \rightarrow b)$

- **output (stimulated or constitutive)**
  - pos

**pos formula**

$\text{pos}(a,b) \triangleq \tau_{\eta} : (\text{tr}(b) | \text{pos}(a,b)) + \tau_{\varepsilon} : (\text{tr}(b) | \text{pos}(a,b))$

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

Graph showing the interaction between $a$ and $b$ with time progression and output levels.
Unary Neg Gate

- **Input (inhibitory)**: \( a \xrightarrow{\text{neg}} b \)
- **Output (constitutive when not inhibited)**: \( \text{neg}(a,b) \)

*(input, ?) interaction with rate \( r \)*

or constitutive transcription to always get things started

\( \text{neg}(a,b) \triangleq \)

\( ?q_r; \tau_{\eta}; \text{neg}(a,b) + \tau_{\epsilon}; (\text{tr}(b) | \text{neg}(a,b)) \)

Inhibition delay with rate \( \eta \)

\( r=1.0, \epsilon=0.1, \eta=0.01, \delta=0.001 \)

\( \text{Neg}(a_r,b) \)

\( \text{tr}(a_r) | \text{neg}(a_r,b) \)
Signal Amplification

pos(a,b) | pos(b,c)

\[
pos(a,b) \triangleq \quad \tau_r \cdot (\text{tr}(b) | \text{pos}(a,b)) + \tau_e \cdot (\text{tr}(b) | \text{pos}(a,b)) \]

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

With little degradation

\( r=1.0, \epsilon=0.01, \eta=0.1, \delta=0.00001 \)

pos(a,b) | pos(b,c)

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

even with no a input, consitutive production of b gets amplified to a high c signal
Signal Normalization

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

\[ a \quad \text{neg} \quad b \quad \text{neg} \quad c \]

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

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

---

![Graph showing signal normalization and renormalization](image)

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

- The renormalization occurs with parameters: \( r = 1.0, \ \epsilon = 0.1, \ \eta = 0.01, \ \delta = 0.001 \).

- The renormalization process is denoted as \( 30^*\text{tr}(a) \mid \text{neg}(a,b) \mid \text{neg}(b,c) \).
Self Feedback Circuits

\[ \text{pos}(a,a) \]

\[ \text{neg}(a,a) \]

\[ \begin{align*}
\text{pos}(a,b) & \triangleq \\
& \quad \text{?} a_r; (\text{tr}(b) \mid \text{pos}(a,b)) + \\
& \quad \tau_\epsilon; (\text{tr}(b) \mid \text{pos}(a,b)) \\
\text{tr}(p) & \triangleq (\text{!} p_r; \text{tr}(p)) + \tau_\delta \\
\end{align*} \]

(Can overwhelm degradation, depending on parameters)

\[ \begin{align*}
\text{neg}(a,b) & \triangleq \\
& \quad \text{?} a_r; \tau_h; \text{neg}(a,b) + \\
& \quad \tau_\epsilon; (\text{tr}(b) \mid \text{neg}(a,b)) \\
\text{tr}(p) & \triangleq (\text{!} p_r; \text{tr}(p)) + \tau_\delta \\
\end{align*} \]
Two-gate Feedback Circuits

Monostable:
For some degradation rates is quite stable:

\[ r=1.0, \delta=0.0005 \]

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

\[ r=1.0, \delta=0.0001 \]

Bistable:

\[ r=1.0, \epsilon=0.1, h=0.01, \delta=0.001 \]

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

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

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

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

- \( \text{tr}(p) \triangleq \neg p_r \)
  - interact once and die
  - otherwise stick around

- \( \text{tr}(p) \triangleq \neg p_r + \tau_\delta \)
  - interact once and die
  - otherwise decay

- \( \text{tr}(p) \triangleq (\neg p_r ; \text{tr}(p)) + \tau_\delta \)
  - interact many times
  - and decay

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

\[
\begin{align*}
\frac{d[X]}{dt} &= \alpha_0 + \frac{\alpha + \alpha_t [PY]^n}{K^n + [PY]^n} - k[X], \\
\frac{d[PY]}{dt} &= \beta ([X] - [PX]) \\
\frac{d[Y]}{dt} &= \alpha_0 + \frac{\alpha + \alpha_t [PZ]^n}{K^n + [PZ]^n} - k[Y], \\
\frac{d[PY]}{dt} &= \beta ([Y] - [PY]) \\
\frac{d[Z]}{dt} &= \alpha_0 + \frac{\alpha + \alpha_t [PX]^n}{K^n + [PX]^n} - k[Z], \\
\frac{d[PZ]}{dt} &= \beta ([Z] - [PZ])
\end{align*}
\]
**Guet et al.: D038/lac**


Diagram: aTc, TetR, LacI, IPTG, λcI, GFP

**experiment:**
- aTc: - + + +
- IPTG: - - + +
- GFP: - + - -
  (LacI: - + - -)

neg(TetR, TetR) | neg(TetR, LacI) | neg(LacI, λcI) | neg(λcI, GFP)

**r = 1.0, ε = 0.1, h = 1.0, δ = 0.001**

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.
Neg Gate Signal Response

Neg Gate

\[ neg(a, b) = \begin{cases} \text{inhibitory input} & \text{back to initial state} \\ \text{constitutive transcription} & \end{cases} \]

\[ \tau_\eta; \neg(a, b) \pm \tau_\varepsilon; (tr(b) \mid \neg(a, b)) \]

Gate Response

\( \eta = 0.1, \delta = 0.001 \)

\[
\begin{array}{c}
\eta: \\
100 \\
10 \\
1 \\
0 = 0.01 \\
0 \\
100 \\
\end{array}
\]

\[ tr(b) = !b; tr(b) + \tau_\delta \]

\[ \eta = 1: b \sim 100/a \text{ (at the fixpoint) matches Alon’s numbers} \]

\[ \eta = 0.01: b \sim 1/a \text{ is the self-feedback instability point} \]

\[ i\eta = 10: b \sim 900/a \]

hence \( b \sim 100 \times \eta / a \)
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. MAPKKK$^*$ → MAPKK</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.7, MAPKK$^*$: 1.0, MAPK: 4.9</td>
</tr>
<tr>
<td>2. MAPKK → MAPKK$^*$</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK$^*$: 1.0, MAPK: 4.9</td>
</tr>
<tr>
<td>3. MAPKK → MAPKK-P</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK-P: 1.3−2.3, MAPK: 4.0−5.1</td>
</tr>
<tr>
<td>4. MAPKK-P → MAPKK</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK-P: 1.5−1.9, MAPK: 3.6−6.7</td>
</tr>
<tr>
<td>5. MAPKK-P → MAPKK-PP</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK-P: 1.2−2.4, MAPK: 3.8−5.2</td>
</tr>
<tr>
<td>6. MAPKK-PP → MAPKK-P</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK-P: 1.7−1.8, MAPK: 4.1−6.4</td>
</tr>
<tr>
<td>7. MAP→ MAPKK</td>
<td>60−1500 nM (300 nM$^3$)</td>
<td>MAPKK: 1.0, MAPKK: 3.7−6.2</td>
</tr>
<tr>
<td>8. MAP-P → MAPKK</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK: 1.7, MAPK: 4.3−5.2</td>
</tr>
<tr>
<td>9. MAPKK-P → MAPKK-PP</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK-P: 1.7, MAPK: 3.4−6.1</td>
</tr>
<tr>
<td>10. MAPKK-PP → MAPKK</td>
<td>60−1500 nM</td>
<td>MAPKK: 1.0, MAPKK: 1.7, MAPK: 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 input/response curves, as described in the text.

The $K_m$ values 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

Equations 1-10 represent the reactions of the MAPK cascade, which are shown schematically in Fig. 1. We have used Goldbeter and Koshland'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; KK denotes MAPKK$^*$; and $K$ denotes MAPK.

1. $KKK + E1 \rightarrow KKK-E1 \rightarrow KKK + E1$
2. $KKK + E2 \rightarrow KKK-E2 \rightarrow KKK + E2$
3. $KK + KKK \rightarrow KK + KKK^*$
4. $KKP + KKP \rightarrow KK + KKP$
5. $KKP + KKP \rightarrow KKP + KKP^*$
6. $KKP + KK \rightarrow KKP + KK$
7. $K + KPP \rightarrow K + KPP$
8. $K + KPP \rightarrow K + KPP$
9. $K + KPP \rightarrow K + KPP$
10. $K + KPP \rightarrow K + KPP$

*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. P'ase denotes phosphatase.*
As 18 Ordinary Differential Equations

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

\[
\begin{align*}
\frac{d}{dt}[KKK] &= -a_1[KKK][E1] + d_1[KKK\cdot E1] \\
&+ k_2[KKK^* \cdot E2] \\
\frac{d}{dt}[KKK\cdot E1] &= a_1[KKK][E1] - (d_1 + k_1)[KKK\cdot E1] \\
\frac{d}{dt}[KKK^*] &= -a_2[KKK^*][E2] + d_2[KK^* \cdot E2] \\
&+ k_1[KKK \cdot E1] + (k_3 + d_3)[KK \cdot KKK^*] - a_3[KKK^*][KK] \\
&+ (k_5 + d_5)[KK-P \cdot KKK^*] - a_5[KK-P][KKK^*] \\
\frac{d}{dt}[KKK^* \cdot E2] &= a_2[KKK^*][E2] - (d_2 + k_2)[KKK^* \cdot E2] \\
\frac{d}{dt}[KK] &= -a_3KK][KKK^*] + d_3[KK-KKK^*] \\
&+ k_4[KK-P \cdot KPP^*] \\
\frac{d}{dt}[KK \cdot KKK^*] &= a_3[KK][KKK^*] \\
&- (d_3 + k_3)[KK \cdot KKK^*] \\
\frac{d}{dt}[KK-P] &= -a_4[KK-P][KK P'ase] + d_4[KK-P \cdot KPP^*] \\
&+ k_5[KK \cdot KKK^*] + k_6[KK-PP \cdot KKK^*] \\
&+ d_5[KK-P \cdot KKK^*] - a_5[KK-P][KKK^*] \\
\frac{d}{dt}[KPP] &= k_5[KK-P \cdot KKK^*] - a_6[KK-PP][KK P'ase] \\
&+ d_6[KK-PP \cdot KKK^*] - a_7[KK-PP][KK] \\
&+ (d_7 + k_7)[K \cdot KK-PP] \\
&+ (d_8 + k_8)[K \cdot KPP] \\
&- a_8[K \cdot KPP][KK-PP] \\
\frac{d}{dt}[KK-PP \cdot KPP^*] &= a_6[KK-PP][KK P'ase] \\
&- (d_6 + k_6)[KK-PP \cdot KPP^*] \\
\frac{d}{dt}[K] &= -a_7[K][KK-PP] + d_7[K \cdot KK-PP] \\
&+ k_8[K \cdot K P'ase] \\
\frac{d}{dt}[K \cdot KK-PP] &= a_7[K][KK-PP] - (d_7 + k_7)[K \cdot KK-PP] \\ 
\end{align*}
\]
In addition, there are seven conservation equations (Eqs. 29-35).

\[
\begin{align*}
\text{[KKK]_{tot}} & = [KKK] + [KKK^*] + [KKK\cdot E1] \\
& + [KKK^*\cdot E2] \\
& + [KKK^*\cdot K] + [KKK^*\cdot K\cdot P] \\
\text{[E1]_{tot}} & = [E1] + [KKK\cdot E1] \\
\text{[E2]_{tot}} & = [E2] + [KKK^*\cdot E2] \\
\text{[KK]_{tot}} & = [KK] + [KK\cdot P] + [KK\cdot PP] + [KK\cdot KKK^*] \\
& + [KK\cdot P\cdot KKK^*] + [KK\cdot P\cdot KK P'ase] \\
& + [KK\cdot PP\cdot KK P'ase] \\
& + [KK\cdot PP\cdot K] + [KK\cdot PP\cdot K\cdot P] \\
\text{[KK P'ase]_{tot}} & = [KK P'ase] + [KK P'ase\cdot KK\cdot P] \\
& + [KK P'ase\cdot KK PPP] \\
\text{[K]_{tot}} & = [K] + [K\cdot P] + [K\cdot PP] + [K\cdot PP\cdot K] \\
& + [KK\cdot PP\cdot K\cdot P] + [K\cdot P\cdot K'ase] + [K\cdot PP\cdot K'ase] \\
\text{[K P'ase]_{tot}} & = [K P'ase] + [K\cdot P\cdot K'ase] \\
& + [K\cdot PP\cdot K'ase] \\
\end{align*}
\]

These equations were solved numerically using the Runge–Kutta-based NDSolve algorithm in Mathematica (Wolfram Research, Champaign, IL). An annotated copy of the Mathematica code for the MAPK cascade rate equations can be obtained from J.E.F.
The Circuit

E1 (input)

KKK → KKK*

KK ← KK-P → KK-PP

KK-P'ase

K ← K-P → K-PP

K-P'ase

(output)
Enzymatic Reactions

**Reaction View**

\[ S \xrightarrow{(c,d,e)} E \xrightarrow{d} ES \xrightarrow{e} P+E \]

**Interaction View**

- **bind**
- **unbind**
- **react**

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

\[ E() \triangleq ?a_c(u); (?u_d; E()) + ?k_e; E() \]

\[ P() \triangleq ... \]
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()))

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 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_PP() =
(new u4@d4:Release new u5@d5:Release
  do !a4(u4); (do !u4;KK_P() or !k4;KK_P())
or !a5(u5); (do !u5;KK_P() or !k5;KK_PP()))

and KKPse() =
do ?a4(u4); (do ?u4;KKPse() or ?k4;KKPse())
or ?a6(u6); (do ?u6;KKPse() or ?k6;KKPse())

let K() =
(new u7@d7:Release
  !a7(u7); (do !u7;K() or !k7;K_P()))

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

KKK
KK
K

KK-P
K-P

1xE1 injected

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

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

Input is 1xE1.
Output is 100xK-PP
(ultrasensitivity).
MAPK Cascade Simulation in SPiM

All coefficients 1.0 !!!
100xKKK, 100xKK, 100xK, 10xE2, 10xKKPse, 10xKPse.
(so 1xE1 is no longer sufficient to produce an output)
Scaling up:
ODE vs Process Descriptions
**Chemistry vs. \( \pi \)-calculus**

A process calculus (chemistry, or SBML)

\[
\text{Na} + \text{Cl} \rightarrow_{k1} \text{Na}^+ + \text{Cl}^- \\
\text{Na}^+ + \text{Cl}^- \rightarrow_{k2} \text{Na} + \text{Cl}
\]

A compositional graphical representation, and the corresponding calculus.

\[
\begin{align*}
\text{Na} & \quad \text{Cl} \\
?e & \quad \text{k1} & \quad \text{k1} & \quad !e & \quad ?e & \quad !e \\
& \quad \text{Na}^+ & \quad \text{Cl}^- \\
\text{Na}^+ & \quad \text{k2} & \quad \text{k2} & \quad \text{Cl}^- \\
\end{align*}
\]

The same “model”

\[
\begin{align*}
\text{Na} = & \text{!}e_{k1}; ?e_{k2}; \text{Na} \\
\text{Cl} = & ?e_{k1}; \text{!}e_{k2}; \text{Cl}
\end{align*}
\]

A different process calculus (\( \pi \))

This Petri-Net-like graphical representation degenerates into spaghetti diagrams: precise and dynamic, but not scalable, structured, or maintainable.
From Reactions to ODE's

r₁: A+B →ₖ₁ C+C
r₂: A+C →ₖ₂ D
r₃: C →ₖ₃ E+F
r₄: F →ₖ₄ B

Write the coefficients by columns

<table>
<thead>
<tr>
<th>N</th>
<th>r₁</th>
<th>r₂</th>
<th>r₃</th>
<th>r₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Stoichiometric Matrix

species
x

Read the concentration changes from the rows

d[A]/dt = -v₁ - v₂

E.g. d[A]/dt = -k₁[A][B] - k₂[A][C]

Read the rate laws from the columns

vᵢ(x,eᵢ,kᵢ)

x: chemical species
[-]: concentrations
v: rate laws
k: kinetic parameters
N: stoichiometric matrix
e: catalysts (if any)
From Reactions to Processes

\[ r_1: A + B \rightarrow_{k_1} C + C \]
\[ r_2: A + C \rightarrow_{k_2} D \]
\[ r_3: C \rightarrow_{k_3} E + F \]
\[ r_4: F \rightarrow_{k_4} B \]

Write the coefficients by columns

\[
\begin{array}{c|cccc}
\text{N} & r_1 & r_2 & r_3 & r_4 \\
\hline
A & -1 & -1 & & \\
B & -1 & & 1 & \\
C & 2 & -1 & -1 & \\
D & 1 & & & \\
E & & 1 & & \\
F & & 1 & -1 & \\
\end{array}
\]

For binary reactions, first species in the column does an input and produces result, second species does an output. For unary reactions, species does a tau action and produces result. No ternary reactions.

\[ A = v_1 k_1 (C|C) + v_2 k_2 D + a \]
\[ B = v_1 k_1 + b \]
\[ C = v_2 k_2 + \tau k_3 (E|F) + c \]
\[ D = 0 + d \]
\[ E = 0 + e \]
\[ F = \tau k_3 B + f \]

Add a barb for counting

Read the process interactions from the rows

(Rate laws are implicit in stochastic semantics)
From Reactions to (join)Processes

\[ r_1: A + B \rightarrow_{k_1} C + C \]
\[ r_2: A + C \rightarrow_{k_2} D \]
\[ r_3: C \rightarrow_{k_3} E + F \]
\[ r_4: F \rightarrow_{k_4} B \]

Write the coefficients in the columns

\[
\begin{array}{c|cccc}
\text{N} & r_1 & r_2 & r_3 & r_4 \\
\hline
A & -1 & -1 & & \\
B & -1 & & 1 & \\
C & 2 & -1 & -1 & \\
D & 1 & & & \\
E & & 1 & & \\
F & & 1 & -1 & \\
\end{array}
\]

Would support arbitrary n-ary reactions.

Read the reactions from the columns

Stochastic join calculus ⚫?

Read the species from the rows

A = !v_1a + !v_2a
B = !v_1b
C = !v_2c + !v_3c
D = 0
E = 0
F = !v_4f
Not What We Want

- **Stoichiometric matrices**
  - It is standard to go from chemical equations to ODE’s via stoichiometric matrices.
  - It is possible to go from chemical equations to processes via stoichiometric matrices.

- **But there is a better way:**
  - Stoichiometric matrices blow-up exponentially for biochemical systems (unlike for ordinary chemical systems) because proteins have combinatorial state.
  - We should describe biochemical systems compositionally without going through stochiometric matrices (and hence without ODE’s).
Complexes: From Reactions to ODE’s

$n$ domains

A, B, C

$2n$ domain reactions

A $\Rightarrow$ A$_p$
B $\Rightarrow$ B$_p$
C $\Rightarrow$ C$_p$

1 complex

ABC

$2n$ species reactions (twice number of edges in n-dim hypercube)

ABC$_p$
A$_p$BC$_p$
AB$_p$C$_p$
A$_p$B$_p$C$_p$

The matrix is sparse, so the corresponding ODE system is not dense, but it still has $2^n$ equations, one per species, plus conservation equations ([ABC]+[A$_p$BC]=constant, etc.).

Stoichiometric Matrix

<table>
<thead>
<tr>
<th>N</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
<th>v7</th>
<th>v8</th>
<th>v9</th>
<th>v10</th>
<th>v11</th>
<th>v12</th>
<th>v13</th>
<th>v14</th>
<th>v15</th>
<th>v16</th>
<th>v17</th>
<th>v18</th>
<th>v19</th>
<th>v20</th>
<th>v21</th>
<th>v22</th>
<th>v23</th>
<th>v24</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ApBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABCp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ApBpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ApBpC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$2^n \times 2n(2^{n-1})$
Complexes: From Reactions to Processes

\[
\begin{align*}
A & \Leftrightarrow A_p \\
B & \Leftrightarrow B_p \\
C & \Leftrightarrow C_p
\end{align*}
\]

\[
\begin{align*}
2n & \\
2n & \\
2n &
\end{align*}
\]

\[
\begin{align*}
A & = \text{?kn;} A_p & A_p & = \text{?ph;} A \\
B & = \text{?kn;} B_p & B_p & = \text{?ph;} B \\
C & = \text{?kn;} C_p & C_p & = \text{?ph;} C
\end{align*}
\]

\[A \mid B \mid C \mid \text{kinase} \mid \text{phtase}\]

Where the local domain reactions are not independent, we can use lateral communication so that each component is aware of the relevant others.
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)