Simulating Biological Systems in the Stochastic Pi-calculus

Andrew Phillips
with
Luca Cardelli

Microsoft Research, Cambridge UK
Biological Computing
Modelling Biology

The Human Genome project:
- Map out the complete genetic code in humans
- To understand and predict gene and protein behaviour
- Like reading the source code of a computer program...
- But functional meaning of the code is still a mystery!

Systems Biology:
- Understand and precisely describe the behaviour of biological systems

Two complementary approaches:
- Look at experimental results and infer general system properties
- Build detailed models of systems and test these in the lab

Biological Modelling:
- Conduct virtual experiments, saving time and money.
- Need tools for modelling complex parallel systems.
- Should also scale up to very large systems.
- The beginnings of a biological programming language...
Programming Biology

Languages for complex, parallel computer systems:

Languages for complex, parallel biological systems:

stochastic π-calculus
Reactions vs. Components

Traditional modelling: model individual reactions
Reactions vs. Components

Traditional modelling: model individual reactions
Large, Connected Reaction Graphs

http://www.celldesigner.org/
Reactions vs. Components

Traditional modelling: model the individual reactions

Stochastic $\pi$-calculus: model the components
Compositional Modelling

Build complex models incrementally, by direct composition of simpler components:
A formal programming language

\[
P, Q ::= \quad M \quad \text{Choice} \quad \quad E ::= \quad \emptyset \quad \text{Empty}
\]
\[
\quad | \quad X(\tilde{n}) \quad \text{Instance} \quad \quad | \quad E, X(\tilde{m}) = P \quad \text{Definition}
\]
\[
\quad | \quad P \mid Q \quad \text{Parallel} \quad \quad | \quad \text{fn}(P) \subseteq \tilde{m}
\]
\[
\quad | \quad \nu x \ P \quad \text{Restriction} \quad \quad | \quad \pi ::= \quad ?x(\tilde{n}) \quad \text{Input}
\]
\[
\quad | \quad !x(\tilde{n}) \quad \text{Output} \quad \quad | \quad \tau_r \quad \text{Delay}
\]

\[
\tau_r. P + M \xrightarrow{r} P \quad \quad (1)
\]
\[
!x(\tilde{n}). P + M \mid ?x(\tilde{m}). Q + N \xrightarrow{r} P \mid Q_{\{\tilde{n}/\tilde{m}\}} \quad \quad (2)
\]
\[
P \xrightarrow{r} P' \quad \Rightarrow \quad \nu x \ P \xrightarrow{r} \nu x \ P' \quad \quad (3)
\]
\[
P \xrightarrow{r} P' \quad \Rightarrow \quad P \mid Q \xrightarrow{r} P' \mid Q \quad \quad (4)
\]
\[
Q \equiv P \xrightarrow{r} P' \equiv Q' \quad \Rightarrow \quad Q \xrightarrow{r} Q' \quad \quad (5)
\]

Analysis techniques (types, equivalences, model-checking)

Could help provide insight into fundamental properties of biological systems
Equivalent Models

Can we replace one model with another?
Related Work

- Stochastic $\pi$-calculus proposed by [Priami, 1995]

- Used to simulate a range of biological systems:
  - RTK MAPK pathway [Regev et al., 2001]
  - Gene Regulation by positive Feedback [Priami et al., 2001]
  - Cell Cycle Control in Eukaryotes [Lecca and Priami, 2003]

- First simulator for stochastic $\pi$-calculus [BioSPI]
  - A subset of $\pi$-calculus with limited choice
  - Compiles a calculus process to an FCP procedure
  - Executed by the FCP Logix platform [Silverman et al., 1987]
Graphical Stochastic $\pi$-calculus

- Display stochastic $\pi$-calculus models as graphs
  - [Phillips and Cardelli, 2005]
- Helps make the stochastic $\pi$-calculus more accessible
- Defined a graphical calculus and graphical execution model
- Proved equivalent to the stochastic $\pi$-calculus
  - [Phillips, Cardelli and Castagna, 2006]
The **Stochastic Pi Machine**

- A simulation algorithm for stochastic $\pi$-calculus
  - [Phillips and Cardelli, 2004]
- Based on standard theory of chemical kinetics [Gillespie, 1977]
- The probability of a reaction is proportional to its rate
- Proved correct with respect to the stochastic $\pi$-calculus.

---

**Definition 7. Syntax of SPiM**

\[
\begin{align*}
V, U & ::= \nu x \nu V \\
\text{Restriction} & \quad A, B ::= || \\
\text{Empty} & \quad (\Sigma ; A) \quad \text{Summation} \quad (38)
\end{align*}
\]

**Definition 8. Construction in SPiM**

\[
\begin{align*}
[\Phi] & \equiv P : || \\
\text{Definition 9. Encoding SPi to SPiM} & \quad (36)
\end{align*}
\]

**Definition 10. Reduction in SPiM**

\[
\begin{align*}
V & \equiv V' \quad \Rightarrow \quad \nu x V \quad \Rightarrow \quad \nu x V' \\
\text{Definition 11. Selecting in SPiM} & \quad (37)
\end{align*}
\]

1. For all $x \in \text{fn}(A)$ calculate $a_x = \text{Act}_x(A) \times \text{rate}(x)$
2. Sort non-zero values of $a_x$ in a list $(a_1, a_2, \ldots, a_M)$, where $a_1 \geq 1 \ldots a_M$.
3. Calculate $a_0 = \sum_{i=0}^{M-1} a_i$.
4. Generate two random numbers $r_1, r_2 \in [0, 1]$ and calculate $\tau, m$ such that:
   \[
   \tau = \frac{1}{a_0} \ln(1/m) \\
   \sum_{i=1}^{m-1} a_i < a_0 \leq \sum_{i=1}^{m} a_i
   \]
5. $\text{Next}(A) = x$ and $\text{Delay}(A) = \tau$.

**Definition 12. Calculating Next(A) and Delay(A) according to Gillespie [6].**
The **SPiM Simulator**

- Simulation algorithm mapped to functional code (F#)
- Used as the basis for implementing the SPiM simulator.
- [Phillips, 2006] GUI by James Margetson, MSRC
- Close correspondence between formal algorithm and functional code
- Correct specification improves confidence in simulation results
- Used in various research centres (UK, France, Italy, Sweden...)

http://research.microsoft.com/~aphillip/spim
Visualising Simulations in 3D

Generate a 3D view of the interactions

Software by Rich Williams, MSRC
Course Outline

- The Stochastic Pi-calculus
- Gene Networks
- Signalling Networks
- Immune System Pathway
The Stochastic Pi-calculus

Introductory Tutorial
Calculus Syntax

\[ \pi ::= \ ?x(m) \]  
\[ \ !x(n) \]  
\[ \tau_r \]  
\[ \text{Input value } m \text{ on channel } x \]  
\[ \text{Output value } n \text{ on channel } x \]  
\[ \text{Delay at rate } r \]  

\[ P ::= \pi_1.P_1 + ... + \pi_N.P_N \]  
\[ P_1 | ... | P_M \]  
\[ X(n) \]  
\[ \text{Choice between actions} \]  
\[ \text{Parallel composition of processes} \]  
\[ \text{Instance of } X \text{ with arguments } n \]  
\[ \text{Restriction of names } x_1, ..., x_N \text{ to } P \]  

\[ E ::= X(m) = P \]  
\[ E_1, \ldots, E_N \]  
\[ \text{Definition of } X, \text{ where } fn(P) \subseteq m \]  
\[ \text{Union of definitions} \]
## Graphical Syntax

<table>
<thead>
<tr>
<th>Choice</th>
<th>Parallel</th>
<th>Instance</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \quad \pi_1.P_1 + \ldots + \pi_N.P_N$</td>
<td>$P_1 \mid \ldots \mid P_M$</td>
<td>$X(n)$, if $X(m) = P$</td>
<td>new $x_1, \ldots, x_N$</td>
</tr>
<tr>
<td>![Diagram of Choice]</td>
<td>![Diagram of Parallel]</td>
<td>![Diagram of Instance]</td>
<td>![Diagram of Restriction]</td>
</tr>
<tr>
<td>$P$</td>
<td>$P_1$</td>
<td>$X$</td>
<td>$P$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$P_M$</td>
<td>${n/m}$</td>
<td>$(x_1, \ldots, x_N)$</td>
</tr>
</tbody>
</table>

### Definition

<table>
<thead>
<tr>
<th>$E$</th>
<th>$X(m) = P$</th>
</tr>
</thead>
</table>

### Union

<table>
<thead>
<tr>
<th>$E_1, \ldots, E_N$</th>
<th>$E_1 \quad \cdots \quad E_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$P$</td>
</tr>
<tr>
<td>$E$</td>
<td>$E_1 \quad \cdots \quad E_N$</td>
</tr>
</tbody>
</table>
Execution: Stochastic Delay

\[ \tau_r P_1 + \ldots + \pi_N P_N \]
Execution: Stochastic Delay

\[ \tau_r P_1 + \cdots + \pi_N P_N \quad \rightarrow \quad P_1 \]
Execution: Interaction

\[!x(n).P_1 + \ldots + \pi_N . P_N \quad \mid \quad ?x(m).Q_1 + \ldots + \pi_M . Q_M\]
Execution: Interaction

\[ !x(n).P_1 + \ldots + \pi_N.P_N \quad \mid \quad ?x(m).Q_1 + \ldots + \pi_M.Q_M \]

\[ \rightarrow \quad P_1 \quad \mid \quad Q_1\{n/m\} \]
new \( n \) \((!x(n).P_1 + \ldots + \pi_N P_N) \mid ?x(m).Q_1 + \ldots + \pi_M Q_M\)
Execution: Binding Interaction

\[
\text{new } n \ (!x(n).P_1 + \ldots + \pi_N.P_N) \ | \ ?x(m).Q_1 + \ldots + \pi_M.Q_M
\]

\[\longrightarrow\]

\[
\text{new } n \ (P_1 \ | \ Q_1\{n/m\})
\]
**Ionization:** \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

Let \( Na() = \) !ionize; \( Na\_plus() \)
and \( Na\_plus() = \) ?deionize; \( Na() \)
run \( Na() \)

Let \( Cl() = \) ?ionize; \( Cl\_minus() \)
and \( Cl\_minus() = \) !deionize; \( Cl() \)
run \( Cl() \)

- \( Na \) can ionize \( Cl \) at \( \text{rate}(\text{ionize}) = 100 \text{s}^{-1} \)
- \( Cl^- \) can deionize \( Na^+ \) at \( \text{rate}(\text{deionize}) = 10 \text{s}^{-1} \)
Ionization: \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

```
let Na() = !ionize; Na_plus()
and Na_plus() = ?deionize; Na()
run Na()

let Cl() = ?ionize; Cl_minus()
and Cl_minus() = !deionize; Cl()
run Cl()
```

- \( Na \) can ionize \( Cl \) by an output on the \textit{ionize} channel
Ionization: \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

let Na() = !ionize; Na_plus()
and Na_plus() = ?deionize; Na()
run Na_plus()

let Cl() = ?ionize; Cl_minus()
and Cl_minus() = !deionize; Cl()
run Cl_minus()

- \( Cl^- \) can deionize \( Na^+ \) by an output on the deionize channel
**Ionization:** \[ Na + Cl \leftrightarrow Na^+ + Cl^- \]

let Na() = !ionize; Na_plus()
and Na_plus() = ?deionize; Na()
run Na()

let Cl() = ?ionize; Cl_minus()
and Cl_minus() = !deionize; Cl()
run Cl()

- *Na* and *Cl* are no longer charged
Ionization: \[ Na + Cl \leftrightarrow Na^+ + Cl^- \]

- A number of \textit{Na} and \textit{Cl} atoms can be composed in parallel.
Ionization: \( \text{Na} + \text{Cl} \leftrightarrow \text{Na}^+ + \text{Cl}^- \)

One of the Na atoms can ionize one of the Cl atoms.
Ionization:  \[ Na + Cl \leftrightarrow Na^+ + Cl^- \]

Additional Na and Cl atoms can interact in parallel.
Ionization: \[ Na + Cl \leftrightarrow Na^+ + Cl^- \]

- A \( Cl^- \) ion can deionize any of the \( Na^+ \) ions.
Ionization: \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

- These reactions can continue indefinitely...
Virtual Experiment: \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

What happens if we mix 100\( \times Na \) and 100\( \times Cl \)?

Use a more compact representation to count populations.

The colour is proportional to the number of atoms:
Virtual Experiment: \( \text{Na} + \text{Cl} \leftrightarrow \text{Na}^+ + \text{Cl}^- \)

One of the \textit{Na} atoms can ionize one of the \textit{Cl} atoms.
Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$

![Diagram showing the interaction of Na and Cl atoms.]

- Additional $Na$ and $Cl$ atoms can interact in parallel.
Virtual Experiment: \( Na + Cl \leftrightarrow Na^+ + Cl^- \)

A \( Cl^- \) ion can deionize any of the \( Na^+ \) ions.
Virtual Experiment: \[ Na + Cl \leftrightarrow Na^+ + Cl^- \]

Eventually an Equilibrium is reached...
Virtual Experiment:  \( \text{Na} + \text{Cl} \leftrightarrow \text{Na}^+ + \text{Cl}^- \)

- At equilibrium: \(100 \times [\text{Na}][\text{Cl}] = 10 \times [\text{Na}^+][\text{Cl}^-]\)
- Approximately \(76 \times \text{Na}\) and \(24 \times \text{Na}^+\)
Binding:  $H + Cl \leftrightarrow HCl$

let $H() = \text{new } e@10.0 \ (\ !\text{share}(e); \ H\_\text{Bound}(e))$
and $H\_\text{Bound}(e) = !e; \ H()$

let $Cl() = ?\text{share}(e); \ Cl\_\text{Bound}(e)$
and $Cl\_\text{Bound}(e) = ?e; \ Cl()$
run ( $H() \ | \ Cl()$ )

- $H$ has a private electron $e$.
- $H$ can share its electron with $Cl$ at $rate(\text{share}) = 100 \text{s}^{-1}$
- $HCl$ can break its private bond at $rate(e) = 10 \text{s}^{-1}$
Binding: \[H + Cl \leftrightarrow HCl\]

let H() = new e@10.0 ( 
    !share(e); H_Bound(e))

and H_Bound(e) = !e; H()

let Cl() = ?share(e); Cl_Bound(e)

and Cl_Bound(e) = ?e; Cl()

run ( H() | Cl() )

- \(H\) can share its electron with \(Cl\) on the share channel.
Binding: \[ H + Cl \Leftrightarrow HCl \]

let H() = new e@10.0 (  
  !share(e); H_Bound(e))
and H_Bound(e) = !e; H()

let Cl() = ?share(e); Cl_Bound(e)  
and Cl_Bound(e) = ?e; Cl()
run new e (H_Bound(e) | Cl_Bound(e))

\[ \text{HCl can break its private bond by synchronising on } e. \]
Binding: \( H + Cl \leftrightarrow HCl \)

Let \( H() = \text{new e@10.0 (} \) !share(e); HBound(e)) and HBound(e) = !e; H()

Let Cl() = ?share(e); ClBound(e) and ClBound(e) = ?e; Cl()

Run \( H() | Cl() \)

- \( H \) and \( Cl \) are no longer bound
Binding: \( H + Cl \leftrightarrow HCl \)

- A number of \( H \) and \( Cl \) atoms can be composed in parallel.
Binding: \[ H + Cl \leftrightarrow HCl \]

- One of the \( H \) atoms can bind with one of the \( Cl \) atoms.
Binding: \[ H + Cl \leftrightarrow HCl \]

- Additional \( H \) and \( Cl \) atoms can bind in parallel.
A single $HCl$ molecule can split into $H$ and $Cl$ atoms.
Binding: \[ H + Cl \leftrightarrow HCl \]

- These reactions can continue indefinitely...
Virtual Experiment: \[ H + Cl \leftrightarrow HCl \]

As with the previous reaction, we mix 100\(\times\)H and 100\(\times\)Cl
Virtual Experiment: \( H + Cl \leftrightarrow HCl \)

One of the \( H \) atoms can bind with one of the \( Cl \) atoms
Virtual Experiment: $H + Cl \leftrightarrow HCl$

Additional $H$ and $Cl$ atoms can bind in parallel.
Virtual Experiment: $H + Cl \leftrightarrow HCl$

A single $HCl$ molecule can split into $H$ and $Cl$ atoms.
Virtual Experiment: $H + Cl \leftrightarrow HCl$

Eventually an Equilibrium is reached...
Virtual Experiment: \( H + Cl \rightleftharpoons HCl \)

- At equilibrium: \( 100[H][Cl] = 10[HCl] \)
- Approximately \( 3H \) and \( 97HCl \)
Calculus Syntax

\(\pi\)-calculus:

\[
P, Q ::= \nu x P \quad \text{Restriction} \quad E ::= X(m) = P \quad \text{Definition} \\
| P \mid Q \quad \text{Parallel} \quad | E_1, E_2 \quad \text{Union} \\
| M \quad \text{Choice} \quad | \emptyset \quad \text{Empty} \\
| X(n) \quad \text{Instance} \\
\]

\[
M ::= \pi.P + M \quad \text{Action} \quad \pi ::= ?x(m) \quad \text{Input} \\
| 0 \quad \text{Null} \quad | !x(n) \quad \text{Output} \\
| \tau_r \quad \text{Delay} \\
\]

Graphical \(\pi\)-calculus:

\[
P, Q ::= \nu x P \quad \text{Restriction} \quad E ::= X(m) = D \quad \text{Definition} \\
| P \mid Q \quad \text{Parallel} \quad | E_1, E_2 \quad \text{Union} \\
| 0 \quad \text{Null} \quad | \emptyset \quad \text{Empty} \\
| X(n) \quad \text{Instance} \\
\]

\[
D ::= P \quad \text{Process} \\
M ::= \pi.P + M \quad \text{Action} \quad | M \quad \text{Choice} \\
| 0 \quad \text{Null} \quad | \nu x D \quad \text{Restriction} \\
\]
# Calculus Semantics

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Description</th>
<th>$\text{fn}(\alpha)$</th>
<th>$\text{bn}(\alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$?x(n)$</td>
<td>Receive a value $n$ on channel $x$</td>
<td>${x, n}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>!$x(n)$</td>
<td>Send a value $n$ on channel $x$</td>
<td>${x, n}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>!$x(\nu y)$</td>
<td>Send a private channel $y$ on channel $x$</td>
<td>${x}$</td>
<td>${y}$</td>
</tr>
<tr>
<td>$r$</td>
<td>Perform an action with rate $r$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
!x(n).P + M & \xrightarrow{!x(n)} P \\
?x(m).P + M & \xrightarrow{?x(n)} P_{\{n/m\}} \\
\tau_r.P + M & \xrightarrow{r} P \\
P \xrightarrow{!x(n)} P' & \quad Q \xrightarrow{?x(n)} Q' \quad \Rightarrow \quad P \parallel Q \xrightarrow{\text{rate}(x)} P' \parallel Q' \\
n \notin \text{fn}(Q) & \quad P \xrightarrow{!x(\nu n)} P' \quad Q \xrightarrow{?x(n)} Q' \quad \Rightarrow \quad P \parallel Q \xrightarrow{\text{rate}(x)} \nu n \,(P' \parallel Q') \\
x \neq y & \quad P \xrightarrow{!x(y)} P' \quad \Rightarrow \quad \nu y \, P \xrightarrow{!x(\nu y)} P' \\
x \notin \text{fn}(\alpha) \cup \text{bn}(\alpha) & \quad P \xrightarrow{\alpha} P' \quad \Rightarrow \quad \nu x \, P \xrightarrow{\alpha} \nu x \, P' \\
M \xrightarrow{\alpha} P' & \quad \Rightarrow \quad \pi . P + M \xrightarrow{\alpha} P' \\
\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset & \quad P \xrightarrow{\alpha} P' \quad \Rightarrow \quad P \parallel Q \xrightarrow{\alpha} P' \parallel Q \\
X(m) = P & \quad P_{\{n/m\}} \xrightarrow{\alpha} P' \quad \Rightarrow \quad X(n) \xrightarrow{\alpha} P' 
\end{align*}
\]
Gene Networks

with
Luca Cardelli (MSR Cambridge)
Ralf Blossey (IRI Lille)
Programming a Biological Clock
Gene with Negative Control

- **Neg(a,b)** produces protein b and is blocked by protein a.

```
val transcribe = 0.1  val degrade = 0.001
val unblock = 0.0001  rate(a,b) = 1.0
new a@1.0:chan  new b@1.0:chan

let Neg(a:chan,b:chan) =
do delay@transcribe;
    (Protein(b) | Neg(a,b))
or ?a; Blocked(a,b)
and Blocked(a:chan,b:chan) =
delay@unblock; Neg(a,b)
and Protein(b:chan) =
do !b; Protein(b)
or delay@degrade
run Neg(a,b)
```
Gene Simulation

- Simulation results show evolution over time.
- Level of protein $b$ fluctuates around 100.
Gene Simulation: 0s

- A protein $b$ can be transcribed at rate 0.1
- $P(\text{transcribe}) = 1$

<table>
<thead>
<tr>
<th>reaction</th>
<th>rate (s$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>transcribe</td>
<td>0.1</td>
</tr>
<tr>
<td>degrade</td>
<td>0</td>
</tr>
<tr>
<td>total</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Another protein $b$ can be transcribed

$P(\text{transcribe}) = 0.1 / 0.101$
Gene Simulation: 19.7126s

- And another...
- \( P(\text{transcribe}) = \frac{0.1}{0.102} \)

<table>
<thead>
<tr>
<th>reaction</th>
<th>rate (s(^{-1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>transcribe</td>
<td>0.1</td>
</tr>
<tr>
<td>degrade</td>
<td>0.001*2</td>
</tr>
<tr>
<td>total</td>
<td>0.102</td>
</tr>
</tbody>
</table>
Gene Simulation: 26.80166s

- A protein $b$ can be degraded at rate 0.001
- $P(\text{degrade}) = 0.003 / 0.103$

<table>
<thead>
<tr>
<th>reaction</th>
<th>rate (s$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>transcribe</td>
<td>0.1</td>
</tr>
<tr>
<td>degrade</td>
<td>$0.001 \times 3$</td>
</tr>
<tr>
<td>total</td>
<td>0.103</td>
</tr>
</tbody>
</table>
Eventually reach an equilibrium between transcription and degradation
Gene Simulation: 2980.631s

Equilibrium at about 100 proteins.
P(transcribe) = 0.1 / 0.2 = P(degrade)

<table>
<thead>
<tr>
<th>reaction</th>
<th>rate (s^{-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>transcribe</td>
<td>0.1</td>
</tr>
<tr>
<td>degrade</td>
<td>0.001*100</td>
</tr>
<tr>
<td>total</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Repressilator [Elowitz and Leibler, 2000]

- A gene network engineered in live bacteria.
- Modelled as a simple combination of Neg gates:

\[
\text{Neg}(\text{lac}, \text{tet}) \\
\text{Neg}(\text{tet}, \text{lambda}) \\
\text{Neg}(\text{lambda}, \text{lac}) \\
\text{Neg}(\text{tet}, \text{gfp})
\]

Repressilator Simulation

- Alternate oscillation of proteins: tet, lac, lambda, tet, ...
- Why are the oscillations in a particular order?

![Graph showing alternate oscillation of proteins: tet, lac, lambda, tet, ...]

Results
Repressilator: Debugging

- Understand how the oscillations are produced.
- \( \text{Neg}(\text{lac}, \text{tet}) \mid \text{Neg}(\text{tet}, \text{lambda}) \mid \text{Neg}(\text{lambda}, \text{lac}) \)
Initially there is one copy of each gene

Any one of the proteins can be transcribed at rate 0.1

\[ P(\text{transcribe}) = \frac{0.3}{0.3} \]
Repressilator: 5.568177s

- The *tet* protein can block the *lambda* gene at rate 1.0
- \( P(\text{tet}) = 1.0 \div 1.301 \)
Now no lambda protein can be transcribed.

But lac protein can still be transcribed at rate 0.1

\[ P(\text{transcribe}) = \frac{0.2}{0.2011} \]
Repressilator: 11.62149s

The *lac* protein can block the *tet* gene at rate 1.0

\[ P(\text{lac}) = \frac{1.0}{1.2021} \]
Now no *tet* or *lambda* protein can be transcribed.

A *tet* protein can degrade at rate 0.001

P(degrade) = 0.002 / 0.1022
Repressilator

Meanwhile, lots of lac protein is transcribed
Repressilator

- Represents one oscillation cycle
- Equilibrium between transcription and degradation
- Eventually, \( \text{lambda} \) or \( \text{lac} \) gene unblocks at rate 0.0001
- \( P(\text{unblock}) = 0.0002 / 0.2002 \)
Suppose the lac gene unblocks
There is a high probability that it will block immediately
\[ P(\text{lac}) = 100.0 / 100.3001 \]
Eventually, the \textit{lambda} gene unblocks at rate 0.0001

\[ P(\text{unblock}) = \frac{0.0002}{0.2002} \]
There is nothing to block the lambda gene.
The *lambda* protein can now take over...
A high probability of oscillating in a particular order.
Repressilator Simulation in 3D
We observe irregular oscillations when degrade << unblock

degrade = 0.0001, unblock = 0.001, transcribe = 0.1
Repressilator with Dimers

- Refined model of a gene with cooperative binding
- Different implementation, same main program
- Neg(lac,tet) | Neg(tet,lambda) | Neg(lambda,lac)

```plaintext
let Neg(a:chan,b:chan) = (  
    new b2@dimerize:chan Neg2(a,b2,b)  
)  
and Neg2(a:chan,b2:chan,b:chan) =  
    do delay@transcribe;  
    (Protein(b2,b) | Neg2(a,b2,b))  
or ?a; Blocked(a,b2,b)  
and Blocked(a:chan,b2:chan,b:chan) =  
    delay@unblock; Neg2(a,b2,b)  
and Protein(b2:chan, b:chan) =  
    do ?b2; Protein2(b) or !b2  
or delay@degrade  
and Protein2(b:chan) =  
    do !b; Protein2(b)  
or delay@degrade
```

Refined model of a gene with cooperative binding
Different implementation, same main program

Neg(lac,tet) | Neg(tet,lambda) | Neg(lambda,lac)
Dimers Improve Regularity

- Proteins form dimers first
- Improves regularity of oscillations

Investigating stochasticity, cooperativity, ODEs
[Blossey, Cardelli, Phillips, 2007]
Gene with additional Inhibitor

val transcribe = 0.1  val degrade = 0.001
val unblock = 0.0001  val rate(a,b,r) = 1.0
new a@1.0:chan    new b@1.0:chan
new r@1.0:chan

let Negp(a:chan,b:chan,r:chan) =
  do delay@transcribe;
       (Proteinp(b,r) | Negp(a,b,r))
  or ?a; Blockedp(a,b,r)

and Blockedp(a:chan,b:chan,r:chan) =
  delay@unblock; Negp(a,b,r)

and Proteinp(b:chan,r:chan) =
  do !b; Proteinp(b,r)
  or ?r
  or delay@degrade

let Inh(r:chan) = !r; Inh(r)

run Negp(a,b,r)
**Bacteria Logic Gates** [Guet et al., 2002]


- 3 genes: tetR, lacI, λcl
- 5 promoters: PL1, PL2, PT, Pλ-, Pλ+
- 125 possible networks consisting of 3 promoter-gene units
- 2 inputs: IPTG (represses Lac), aTc (represses Tet)
- 1 output: GFP (linked to Pλ-)
Combinatorial Library of Genes

- Can model 125 networks using just 2 modules (Neg, Negp).
- Used simulation to investigate system behaviour.
- Can easily refine the modules without rewiring the networks.
- [Blossey, Cardelli, Phillips, 2006]

```plaintext
let D038() =
    ( Negp(TetR,TetR,aTc) | Negp(TetR,LacI,IPTG) | Neg(LacI,LambdacI) | Neg(LambdacI,GFP) )

D038()
D038() | Inh(aTc)
D038() | Inh(IPTG)
D038() | Inh(aTc) | Inh(IPTG)
```
Signalling Networks

with
Luca Cardelli (MSR Cambridge)
Jasmin Fisher (EPFL Lausanne)
Programming a Biological Switch
Enzymatic Reactions

An enzyme can bind to a substrate
Enzymatic Reactions

The enzyme and substrate can unbind
Enzymatic Reactions

The enzyme can bind to the substrate again.
The enzyme can react with the substrate
Enzymatic Reactions

- The enzyme is restored and the substrate is transformed into a product
Mapk Cascade  [Huang and Ferrel, 1996]

- Signals the presence of an input enzyme
Mapk Cascade: Simulations

- Rates as in paper
- All rates set to 1.0!

![Graph 1](image1)

![Graph 2](image2)
Mapk Simulation in 3D
Pi-calculus v Reaction Equations

- Pi-calculus
- Reaction Equations
Mapk is a module of EGFR
Immune System Modelling: MHC I Antigen Presentation

with
Luca Cardelli (MSR Cambridge)
Leonard Goldstein (Cambridge University)
Tim Elliott (Southampton University)
Joern Werner (Southampton University)
MHC: A Biological Virus Scanner
MHC: A Biological Virus Scanner

©2005 from Immunobiology, Sixth Edition by Janeway et al.
Reproduced by permission of Garland Science/Taylor & Francis LLC.
Investigate the Role of Tapasin
Peptide Loading Model

- Each graph represents a component
MHC can load a peptide
Peptide Loading Model

- The loaded peptide can escape
Peptide Loading Model

- The loaded peptide can escape
Peptide Loading Model

- MHC can bind tapasin
Peptide Loading Model

- The bound tapasin can unbind
Peptide Loading Model

- The bound tapasasin can unbind
Assume low, medium and high affinity peptides

Experimental Setup
**Model Parameters**

- **MHC spends < 2h on average in the ER.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Rate (min⁻¹)</th>
<th>Time (min)</th>
<th>Range (min⁻¹)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpep</td>
<td>50</td>
<td>0.02</td>
<td></td>
<td>Active transport of peptides into the ER</td>
</tr>
<tr>
<td>dpep</td>
<td>10</td>
<td>0.1</td>
<td></td>
<td>Degradation of free peptides inside the ER</td>
</tr>
<tr>
<td>bind</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Binding of peptides to MHC (per molecule)</td>
</tr>
<tr>
<td>low</td>
<td>3</td>
<td>0.33</td>
<td></td>
<td>Unbinding of low affinity peptides from MHC</td>
</tr>
<tr>
<td>med</td>
<td>1.2</td>
<td>0.83</td>
<td></td>
<td>Unbinding of medium affinity peptides from MHC</td>
</tr>
<tr>
<td>high</td>
<td>0.5</td>
<td>2</td>
<td></td>
<td>Unbinding of high affinity peptides from MHC</td>
</tr>
<tr>
<td>gMHC</td>
<td>10</td>
<td>0.1</td>
<td></td>
<td>Assembly of MHC complexes inside the ER</td>
</tr>
<tr>
<td>dMHCo</td>
<td>0.01</td>
<td>100</td>
<td>0.01 - 100</td>
<td>Degradation of free MHC inside the ER</td>
</tr>
<tr>
<td>dMHCe</td>
<td>0.01</td>
<td>100</td>
<td></td>
<td>Degradation of loaded MHC at the cell surface</td>
</tr>
<tr>
<td>egress</td>
<td>1</td>
<td>1</td>
<td>0.01 - 1</td>
<td>Egression of loaded MHC from the ER</td>
</tr>
<tr>
<td>gTPN</td>
<td>10</td>
<td>0.1</td>
<td></td>
<td>Production of tapasin inside the ER</td>
</tr>
<tr>
<td>dTPN</td>
<td>0.01</td>
<td>100</td>
<td></td>
<td>Degradation of free tapasin inside the ER</td>
</tr>
<tr>
<td>bindT</td>
<td>100</td>
<td>0.01</td>
<td>1 - 1000</td>
<td>Binding of tapasin to MHC (per molecule)</td>
</tr>
<tr>
<td>uT</td>
<td>1</td>
<td>1</td>
<td>0.01 - 1</td>
<td>Unbinding of tapasin from loaded MHC</td>
</tr>
</tbody>
</table>
Tapasin Hypotheses

- Tapasin:
  - Can increase peptide loading at ER entrance
  - Can destabilise loaded MHC

- Peptide:
  - Can increase tapasin unbinding from MHC

<table>
<thead>
<tr>
<th>Factor</th>
<th>Value</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>1 - 10</td>
<td>Binding of peptides to MHC in presence of tapasin</td>
</tr>
<tr>
<td>q</td>
<td>10</td>
<td>1 - 100</td>
<td>Unbinding of peptides from MHC in presence of tapasin.</td>
</tr>
<tr>
<td>v</td>
<td>0.01</td>
<td>0 - 0.01</td>
<td>Unbinding of tapasin from MHC in absence of peptide.</td>
</tr>
</tbody>
</table>
Peptide Editing Simulation
Peptide Filtering

- How does MHC egress the right peptides?
- Consider a loaded peptide with unbinding rate $u$
- Competition between unbinding and egression

\[
P(\text{egress}, u) = \frac{\text{egress}}{(u + \text{egress})}
\]

\[
P(\text{egress}, u) = \frac{1}{(1 + \frac{u}{\text{egress}})}
\]

$\text{egress} \rightarrow \infty : \quad P(\text{egress}, u) \rightarrow 1$

$\text{egress} \rightarrow 0 : \quad P(\text{egress}, u) \rightarrow \frac{1}{u}$
Peptide Discrimination

- Consider 3 loaded MHC complexes
- Stable peptides are more likely to egress

\[ P_i(\text{egress}) = P(\text{egress}, u_i) / \sum_k P(\text{egress}, u_k) \]

\[ \text{egress} \rightarrow \infty: \quad P_i(\text{egress}) \rightarrow 1/\sum_k 1 = 1/3 \]

\[ \text{egress} \rightarrow 0: \quad P_i(\text{egress}) \rightarrow (1/ u_i) / \sum_k (1/ u_k) \]
Discrimination Upper Bound

- Assume low = 6/2, med = 6/5, high = 6/12
- Upper bound as egress tends to 0

<table>
<thead>
<tr>
<th>affinity</th>
<th>low</th>
<th>med</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_i$</td>
<td>6/2</td>
<td>6/5</td>
<td>6/12</td>
</tr>
<tr>
<td>$1/u_i$</td>
<td>2/6</td>
<td>5/6</td>
<td>12/6</td>
</tr>
<tr>
<td>$P_i(\infty)$</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>$P_i(0)$</td>
<td>2/19</td>
<td>5/19</td>
<td>12/19</td>
</tr>
</tbody>
</table>

$$P_i(\text{egress}) \rightarrow (1/ u_i) / \Sigma_k (1/ u_k)$$

$$= (1/ u_i) / (2/6 + 5/6 +12/6)$$

$$= (1/ u_i) \cdot (6/19)$$
Discrimination vs Egression

- Calculate peptide discrimination for different values of egress.
- Assume 1000 uniformly loaded MHC complexes
Peptide Editing Simulations

Simulate peptide editing for different values of egress.
Peptide Editing Results

Simulation results are comparable to predictions
Peptide Filtering with Tapasin

- Tapasin adds a second filtering stage.

\[
P(uT,u) = uT / (u \cdot q + uT)
\]
\[
P(uT,u) = 1 / (1 + u \cdot q/uT)
\]
\[
P(uT, egress, u) = P(uT,u) \cdot P(egress,u)
\]

\[
uT, egress \rightarrow \infty : \quad P(uT, egress, u) \rightarrow 1 \cdot 1
\]
\[
uT, egress \rightarrow 0 : \quad P(uT, egress, u) \rightarrow (1/u \cdot q) \cdot (1/u)
\]
Peptide Discrimination

- Tapasin improves upper bound on discrimination

\[ P_i(uT, \text{egress}) = \frac{P(uT, \text{egress}, u_i)}{\sum_k P(uT, \text{egress}, u_k)} \]

\( uT, \text{egress} \to \infty: P_i(uT, \text{egress}) \to 1 \cdot 1 / \sum_k 1 \cdot 1 = 1/3 \)

\( uT, \text{egress} \to 0: P_i(uT, \text{egress}) \to (1/u_i \cdot q) \cdot (1/u_i) / \sum_k (1/u_k \cdot q) \cdot (1/u_k) \)
Discrimination vs Egression

- Calculate peptide discrimination for different values of $uT/q$ and egress.
- Assume 1000 uniformly loaded MHC complexes

### Table: Peptide Discrimination with Tapasin

<table>
<thead>
<tr>
<th>$uT/q$; egress</th>
<th>low</th>
<th>med</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>1;1</td>
<td>88</td>
<td>290</td>
<td>623</td>
</tr>
<tr>
<td>0.1;1</td>
<td>52</td>
<td>227</td>
<td>721</td>
</tr>
<tr>
<td>0.01;1</td>
<td>47</td>
<td>213</td>
<td>740</td>
</tr>
<tr>
<td>0.1;0.1</td>
<td>30</td>
<td>170</td>
<td>800</td>
</tr>
<tr>
<td>0.01;0.1</td>
<td>27</td>
<td>159</td>
<td>815</td>
</tr>
<tr>
<td>0.01;0.01</td>
<td>24</td>
<td>147</td>
<td>829</td>
</tr>
<tr>
<td>0;0</td>
<td>23</td>
<td>145</td>
<td>832</td>
</tr>
</tbody>
</table>
Tapasin Improves Discrimination

- Max discrimination without tapasin: 105:263:632
- Max discrimination with tapasin: 23:145:832

Peptide Discrimination

Peptide Discrimination with Tapasin

[Bar charts showing discrimination levels with and without tapasin]
Peptide Editing Simulations

Simulate peptide editing with tapasin, for different values of $q$ and egress.
Peptide Editing Results

Simulation results are comparable to predictions
**MHC Alleles: Model Predictions**

- **Peptide discrimination in absence of tapasin**: Depends on egress.
- **Peptide discrimination in presence of tapasin**: Depends on \( q \) and ratio of complexes that follow tapasin pathway.
- **Cell surface expression in absence of tapasin**: Depends on \( dMHC_{o} \).

<table>
<thead>
<tr>
<th>Rate</th>
<th>Range</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>egress</td>
<td>0.1 - 1.0</td>
<td>Lower egress gives better filtering in absence of tapasin.</td>
</tr>
<tr>
<td>( dMHC_{o} )</td>
<td>0.01 - 100</td>
<td>Lower degradation gives higher throughput of MHC</td>
</tr>
<tr>
<td>( q )</td>
<td>1 - 100</td>
<td>Higher ( q ) gives a better filtering in presence of tapasin.</td>
</tr>
</tbody>
</table>
MHC Alleles: Model Simulations

- Explain varying dependence on tapasin

- B4402 (Dependent)
  - egress = 0.1 uT = 1.0, 
  - dMHC = 100.0, TPN = 0

- B2705 (Partially)
  - egress = 1.0 uT = 10.0, 
  - dMHC = 0.01, TPN = 0

- B4405 (Independent)
  - egress = 0.1 uT = 100.0, 
  - dMHC = 0.01, TPN = 0

No TPN

1000 TPN
Peptide Loading: Flytrap Model

- MHC I captures peptides like a Venus Flytrap.

- Peptide enters open MHC
- Unstable peptide escapes
- Stable peptide is captured and presented at cell surface
Flytrap Model

- MHC undergoes a conformational change after peptide loading
Model Equivalence
Compositional Modelling

- Core model
Compositional Modelling

Adding two new peptides
Compositional Modelling

- Adding a conformational change to MHC
Next Steps

- A functional model of MHC I Antigen presentation [Cardelli, Elliott, Goldstein, Phillips, Werner]
- Medical Research Grant (MRC 2006-2007) to complete a more detailed MHC model.
- Medical Research Grant (MRC 2007-2010) to perform targeted experiments.
Conclusions

- Biological systems work surprisingly well, though we don’t fully understand why.
  - We still have a lot to learn from nature.

Long-term benefits for medical research:
- Better understand disease.
- Speed up the design of a cure.

Long-term benefits for computing:
- Write more robust concurrent / distributed programs
- Design and verification of biological computers
- Smart drugs? Computers in a test tube...?

Biological modelling is pushing the boundaries of concurrent programming.
Outlook

Senior executives of pharmaceutical companies:
“a real need for a modular biological programming language”
Thanks.
References


References


