A Graphical Representation for the Stochastic Pi-Calculus

Andrew Phillips             Luca Cardelli

27th August 2005
Microsoft Research
7 J J Thomson Avenue
Cambridge, UK
Introduction

➢ Stochastic pi-calculus used to model and simulate a range of biological systems [Lecca and Priami, 2003, Priami et al., 2001, Regev et al., 2001]:

- Able to model independent system components, which can be composed to predict emergent system behaviour.
- Mathematical definition supports useful analysis techniques: type systems, behavioural equivalences, model checking.

➢ Mathematical syntax and semantics can limit accessibility to a wider audience:

- Useful to present an alternative graphical view
- Particularly welcomed by experimental systems biologists.
Outline

➢ Stochastic pi-calculus

➢ Graphical stochastic pi-calculus:
  ■ Graphical syntax
  ■ Graphical execution model

➢ Graphical biological examples:
  ■ Evolved gene network [Francois and Hakim, 2004]
  ■ Mapk signalling cascade [Huang and Ferrel, 1996]

➢ Automatic graph generation:
  ■ Encoding pi-calculus to an open graph syntax
  ■ Front end to a graphical simulator/debugger (ongoing)
The Stochastic Pi-Calculus (SPi)

Each channel $x$ is associated with a stochastic rate given by $rate(x)$

$$\pi ::= \ ?x(\vec{m}) \quad \text{Input}$$

$$\quad | \quad !x(\vec{n}) \quad \text{Output}$$

$$\quad | \quad \tau_r \quad \text{Delay}$$

$$P, Q ::= \pi_1.P_1 + \ldots + \pi_N.P_N \quad \text{Choice}$$

$$\quad | \quad P_1 | \ldots | P_N \quad \text{Parallel}$$

$$\quad | \quad \nu x \ P \quad \text{Restriction}$$

$$\quad | \quad X(\vec{n}) \quad \text{Instance}$$

$$\Gamma ::= X_1(\vec{m}_1) \triangleq P_1, \ldots, X_N(\vec{m}_N) \triangleq P_N \quad \text{Definitions, fn}(P_i) \subseteq \vec{m}_i$$
The SPiM Programming Language (v0.04)

\[ Dec ::= \] new \( x \{ @r \} : t \] \hspace{1cm} \text{Channel Declaration}
\[ \text{type } n = t \] \hspace{1cm} \text{Type Declaration}
\[ \text{val } m = v \] \hspace{1cm} \text{Value Declaration}
\[ \text{run } P \] \hspace{1cm} \text{Process Declaration}
\[ \text{let } D_1 \text{ and } \ldots \text{ and } D_N \] \hspace{1cm} \text{Definitions, } N \geq 1

\[ D ::= X(m_1, \ldots, m_N) = P \] \hspace{1cm} \text{Definition, } N \geq 0

\[ P ::= () \] \hspace{1cm} \text{Null Process}
\[ (P_1 \mid \ldots \mid P_M) \] \hspace{1cm} \text{Parallel, } M \geq 2
\[ X(v_1, \ldots, v_N) \] \hspace{1cm} \text{Instantiation, } N \geq 0
\[ \pi \{ ; P \} \] \hspace{1cm} \text{Action}
\[ \text{do } \pi_1 \{ ; P_1 \} \text{ or } \ldots \text{ or } \pi_M \{ ; P_M \} \] \hspace{1cm} \text{Choice, } M \geq 2
\[ (Dec_1 \ldots Dec_N P) \] \hspace{1cm} \text{Declarations, } N \geq 0

\[ \pi ::= !x \{ (v_1, \ldots, v_N) \} \] \hspace{1cm} \text{Output, } N \geq 0
\[ ?x \{ (m_1, \ldots, m_N) \} \] \hspace{1cm} \text{Input, } N \geq 0
\[ \text{delay} @r \] \hspace{1cm} \text{Delay}
The Graphical Stochastic Pi-Calculus (GSPi)
A normal form for SPi, with each summation or guarded process as a definition:

\[
\pi ::= \quad ?x(\vec{m}) \quad \text{Input} \\
\quad | \quad !x(\vec{n}) \quad \text{Output} \\
\quad | \quad \tau_r \quad \text{Delay} \\
\]

\[P, Q ::= P_1 \mid \ldots \mid P_N \quad \text{Parallel} \]

\[\quad | \quad \nu x P \quad \text{Restriction} \]

\[\quad | \quad X(\vec{n}) \quad \text{Instance} \]

\[
\Gamma ::= \quad X(\vec{m}) \triangleq \nu x_1 \ldots \nu x_M (\pi_1.X_1(\vec{n}_1) + \ldots + \pi_N.X_N(\vec{n}_N)) \quad \text{Summation} \\
\quad | \quad X(\vec{m}) \triangleq \nu x_1 \ldots \nu x_M (X_1(\vec{n}_1) \mid \ldots \mid X_N(\vec{n}_N)) \quad \text{Composition} \]
A collection of mutually recursive definitions:

\[ X_1(m_1) \triangleq C_1, \ldots, X_N(m_N) \triangleq C_N \]

Displayed as a directed graph with nodes \( X_1 \ldots X_N \) and with edges between these nodes.

Each definition \( X(m) \triangleq C \) displayed as a node \( X \) with zero or more edges to subsequent nodes.
Graphical Representation: Definitions

\[ X(\vec{m}) \triangleq \nu x_1...\nu x_M (\pi_1.X_1(\vec{n}_1) + \ldots + \pi_N.X_N(\vec{n}_N)) \]

\[ X(\vec{m}) \triangleq \nu x_1...\nu x_M (X_1(\vec{n}_1) | \ldots | X_N(\vec{n}_N)) \]
Graphical Representation: Processes

\[ X(\vec{n}) \]

\[ \nu x_1 \ldots \nu x_M (P_1 \mid \ldots \mid P_N) \]

Restriction as Complexation:

A complex of \( P \) and \( Q \) modelled as a restriction \( \nu x (P \mid Q) \)

\[ \nu x_1 \nu x_2 (P_1 \mid P_2 \mid P_3) , x_1 \not\in \text{fn}(P_3), x_2 \not\in \text{fn}(P_1) \]
Graphical Reduction: Execution Model

Reduction in SPi:

\[ !x(\vec{n}).P + \Sigma \mid ?x(\vec{m}).Q + \Sigma' \xrightarrow{rate(x)} \ P \mid Q_{\{\vec{n}/\vec{m}\}} \]  
(1)

\[ \tau_r.P + \Sigma \xrightarrow{r} P \]  
(2)

\[ P \xrightarrow{r} P' \Rightarrow P \mid Q \xrightarrow{r} P' \mid Q \]  
(3)

\[ P \xrightarrow{r} P' \Rightarrow \nu x \: P \xrightarrow{r} \nu x \: P' \]  
(4)

\[ Q \equiv P \xrightarrow{r} P' \equiv Q' \Rightarrow Q \xrightarrow{r} Q' \]  
(5)

Reduction in GSPi $\subset$ SPi:

**Proposition 1.** $\forall P \in GSPi. P \xrightarrow{r} P' \Rightarrow \exists P'' \in GSPi. P' \equiv P''$
Graphical Reduction: Communication

\[
X(\vec{z}) \triangleq !x(\vec{n}).X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z}) , \quad Y \triangleq ?x(\vec{m}).Y_1(\vec{z}) + \ldots + \pi_M.Y_M(\vec{z})
\]

\[
\frac{X(\vec{z}) | Y(\vec{z}) \xrightarrow{\text{rate}(x)} X_1(\vec{z}) | Y_1(\vec{z})\{\vec{n}/\vec{m}\}}
\]
Graphical Reduction: Communication

\[
X(\vec{z}) \triangleq !x(\vec{n}).X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z}) , \quad Y \triangleq ?x(\vec{m}).Y_1(\vec{z}) + \ldots + \pi_M.Y_M(\vec{z})
\]

\[
X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{\text{rate}(x)} X_1(\vec{z}) \mid Y_1(\vec{z})\{\vec{n}/\vec{m}\}
\]
Graphical Reduction: Communication

\[
X(\vec{z}) \triangleq \! x(\vec{n}).X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z}) , \quad Y \triangleq \? x(\vec{m}).Y_1(\vec{z}) + \ldots + \pi_M.Y_M(\vec{z})
\]

\[
X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})\{\vec{n}/\vec{m}\}
\]
\[
X(\vec{z}) \triangleq !x(\vec{n}).X_1(\vec{z}) + ... + \pi_N.X_N(\vec{z}) ,
\]
\[
Y \triangleq ?x(\vec{m}).Y_1(\vec{z}) + ... + \pi_M.Y_M(\vec{z})
\]
\[
\begin{align*}
X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{rate(x)} & \quad X_1(\vec{z}) \mid Y_1(\vec{z})\{\vec{n}/\vec{m}\}
\end{align*}
\]
Inline Graphical Reduction: Communication

\[ X(\vec{z}) \triangleq !x(\vec{n}).X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z}) \ , \ Y \triangleq ?x(\vec{m}).Y_1(\vec{z}) + \ldots + \pi_M.Y_M(\vec{z}) \]

\[ X(\vec{z}) \mid Y(\vec{z}) \xrightarrow{rate(x)} X_1(\vec{z}) \mid Y_1(\vec{z})_{\{\vec{n}/\vec{m}\}} \]
\[
X(\vec{z}) \triangleq !x(\vec{n}) . X_1(\vec{z}) + \ldots + \pi_N . X_N(\vec{z}) , \ Y \triangleq ?x(\vec{m}) . Y_1(\vec{z}) + \ldots + \pi_M . Y_M(\vec{z})
\]

\[
X(\vec{z}) | Y(\vec{z}) \xrightarrow{rate(x)} X_1(\vec{z}) | Y_1(\vec{z}) \{\vec{n}/\vec{m}\}
\]
Inline Graphical Reduction: Delay

\[
X(\vec{z}) \triangleq \tau_r.X_1(\vec{z}) + \ldots + \pi_N.X_N(\vec{z})
\]

\[
X(\vec{z}) \xrightarrow{r} X_1(\vec{z})
\]
Inline Graphical Reduction: Delay

\[
X(z) \triangleq \tau_r X_1(z) + \ldots + \pi_N X_N(z)
\]

\[
X(z) \xrightarrow{r} X_1(z)
\]
Inline Graphical Reduction: Delay

\[ X(\vec{z}) \triangleq \tau_r X_1(\vec{z}) + \ldots + \pi_N X_N(\vec{z}) \]

\[ X(\vec{z}) \xrightarrow{r} X_1(\vec{z}) \]
Graphical Representation: Benefits

➢ Static graphical representation used to:
   - Clarify the connectivity between process definitions
   - Highlight cycles, which are key to many biological systems.

➢ Dynamic graphical representation used to:
   - Visualise the execution trace of a model
   - Clarify the overall system function
   - Graphically debug pi-calculus code.
Evolved Gene Network [Francois and Hakim, 2004]

➢ Gene networks are evolved in silico to perform specific functions, e.g.:

Gene networks are evolved in silico to perform specific functions, e.g.:

Genes $a$ and $b$ can produce proteins $A$ and $B$ respectively:

- $A$ and $B$ can bind irreversibly to produce $AB$, which eventually degrades.
- $A$ can also bind reversibly to gene $b$, slowing the transcription of $B$.

➢ What is the function of this system?
Evolved Gene Network: Definitions

\[
a(\vec{z}) \triangleq \tau_{\text{transcribeA}} \cdot (A() | a())
\]
\[
A(\vec{z}) \triangleq \nu u (\tau_{\text{degradeA}}
+ !\text{bind}(u) \cdot AB(u)
+ !\text{inhibit}(u) \cdot Ab(u))
\]
\[
Ab(u) \triangleq ?u.A()
\]
\[
AB(u) \triangleq \tau_{\text{degradeAB}}
\]
\[
b(\vec{z}) \triangleq \tau_{\text{transcribeB}} \cdot (B() | b())
+ ?\text{inhibit}(u) \cdot bA(u)
\]
\[
bA(u) \triangleq \tau_{\text{transcribeB'}} \cdot (B() | bA(u))
+ !u.b()
\]
\[
B(\vec{z}) \triangleq \tau_{\text{degradeB}}
+ ?\text{bind}(u) \cdot BA(u)
\]
Evolved Gene Network: SPiM Code

let a() = delay@transcribeA; ( A() | a() )
and A() = (  
    new u@0.42:chan  
    do delay@degradeA  
    or !bind; A_B()  
    or !inhibit(u); A_b(u))
and A_b(u:chan) = ?u; A()
and A_B() = delay@degradeAB
let b() =  
    do delay@transcribeB; ( B() | b() )  
    or ?inhibit(u); b_A(u)
and b_A(u:chan) =  
    do !u; b()  
    or delay@transcribeB'; B(); b_A(u)
and B() = do delay@degradeB or ?bind
run (a() | b())
Evolved Gene Network

Initially there is one copy of each gene, $a$ and $b$
Evolved Gene Network

Represent the behaviour of each gene as a separate graph
Evolved Gene Network

Gene $a$ can transcribe a new protein $A$ at rate $\text{transcribe}A$
A new protein $A$ is transcribed
Protein $A$ can bind to gene $b$ to inhibit production of protein $B$
Protein $A$ is bound to gene $b$ by the private channel $u$
Evolved Gene Network

Protein $A$ can unbind from gene $b$ using channel $u$
Evolved Gene Network

Protein $A$ is no longer bound to gene $b$
Gene $b$ can transcribe a new protein $B$ with rate $\text{transcribeB}$
A new protein $B$ is transcribed
Evolved Gene Network

Protein A can bind with protein B
Evolved Gene Network

Protein A and B are irreversibly bound
Evolved Gene Network

Complex $AB$ can be degraded
Evolved Gene Network

Complex $AB$ has been degraded
Evolved Gene Network: Simulation Results
Mapk Cascade [Huang and Ferrel, 1996]

- System originally described using a set of reaction equations
  - Converted to ordinary differential equations, which were solved numerically
  - Response curves shown to be steeply sigmoidal (≃Hill 5).

- System functions as follows:
  - The enzyme E1 drives the transformation from KKK to KKK*
  - KKK* drives the transformation from KK to KK-P to KK-PP
  - KK-PP drives the transformation from K to K-P to K-PP

Andrew Phillips - Bioconcur 2005
Mapk Cascade: Equations

Reaction Equation:

\[ E + K_d^{\rightarrow a} \ E.K \rightarrow^k E + P \]

Pi-calculus Processes:

\[ E(a) \triangleq \nu d \nu k !a(d, k). (?d.E(a) + ?k.E(a)) \]
\[ K(a) \triangleq ?a(d, k). (!d.K(a) + !k.P()) \]

Graphical Representation:
let E1() = (new k1@rk1:chan new d1@rd1:chan!

  a1(d1,k1); do ?d1;E1() or ?k1;E1()
)

let E2() = (new k2@rk2:chan new d2@rd2:chan!

  a2(d2,k2); do ?d2;E2() or ?k2;E2()
)

let KKK() = ?a1(d,k); KKK_E(d,k)
and KKK_E(d:chan,k:chan) =
do !d;KKK() or !k;KKKst()
and KKKst() = (new d3@rd3:chan new k3@rk3:chan

  new d5@rd5:chan new k5@rk5:chan
do ?a2(k,d); KKKE(d,k)
or !a3(d3,k3); (do ?d3;KKKst() or ?k3;KKKst())
or !a5(d5,k5); (do ?d5;KKKst() or ?k5;KKKst())
)

let K() = ?a3(d,k); KK_E(d,k)
and KK_E(d:chan,k:chan) = do !d;KK() or !k;KK_P()
and KK_P() =
do ?a4(k,d); KK_E(d,k) or ?a5(d,k); KK_P_E(d,k)
and KK_P_E(d:chan,k:chan) =
do !d;KK_P() or !k;KK_PP()
and KK_PP() = (new d7@rd7:chan new k7@rk7:chan

  new d9@rd9:chan new k9@rk9:chan
do ?a6(k,d); KK_PP_E(d,k)
or !a7(d7,k7); (do ?d7;KK_PP() or ?k7;KK_PP())
or !a9(d9,k9); (do ?d9;KK_PP() or ?k9;KK_PP())
)

let K() = ?a7(d,k); K_E(d,k)
and K_E(d:chan,k:chan) = do !d; K() or !k; K_P()
and K_P() = do ?a8(k,d); K_E(d,k) or ?a9(d,k); K_P_E(d,k)
and K_P_E(d:chan,k:chan) = do !d;K_P() or !k;K_PP()
and K_PP() = ?a10(k,d); K_P_E(d,k)

let KK() = ?a3(d,k); KK_E(d,k)
and KK_E(d:chan,k:chan) =
do !d;KK() or !k;KK_P()
and KK_P() = (new d4@rd4:chan new k4@rk4:chan

  new d6@rd6:chan new k6@rk6:chan
do !a4(d4,k4); (do ?d4;KKPase() or ?k4;KKPase())
or !a6(d6,k6); (do ?d6;KKPase() or ?k6;KKPase())
)

let KPase() = (new d8@rd8:chan new k8@rk8:chan

  new d10@rd10:chan new k10@rk10:chan
do !a8(d8,k8); (do ?d8;KPase() or ?k8;KPase())
or !a10(d10,k10); (do ?d10;KPase() or ?k10;KPase())
)

run 100 of (KKK() | KK() | K())
run ( E2() | KKPase() | KPase() | E1())

Andrew Phillips - Bioconcur 2005 40
Mapk Cascade

![Mapk Cascade Diagram]
Enzyme $E_1$ can bind to substrate KKK using channel $a_1$
Mapk Cascade

$E_1$ is bound to KKK by private channels $d_1$ and $k_1$
Mapk Cascade

$E_1$ can react with KKK using channel $k_1$
Mapk Cascade

KKK is transformed to KKK*

Andrew Phillips - Bioconcur 2005
KKK* can bind with KK using channel $a_3$
KKK* is bound to KK by private channels $d_3$ and $k_3$
KKK* can react with KK using channel $k_3$
Mapk Cascade

KK is transformed to KK-P

Andrew Phillips - Bioconcur 2005
KKK* can bind to KK-P using channel $a_5$
KKK* is bound to KK-P by channels $d_5$ and $k_5$
KKK* can react with KK-P using channel $k_5$
Mapk Cascade

KK-P is transformed to KK-PP
KK-PP can bind to K using channel $a_7$
KK-PP is bound to K by channels $d_7$ and $k_7$
Mapk Cascade

KK-PP can react with K using channel $k_7$
Mapk Cascade

K is transformed to KK-P
Mapk Cascade

KK-PP can bind to KK-P using channel $a_9$
KK-PP is bound to KK-P by channels $d_9$ and $k_9$. 
Mapk Cascade

KK-PP can react with K-P using channel $k_{9}$
K-P is transformed to K-PP, completing the cascade
Mapk Cascade: Results
Open Graph Syntax (DOT)

\[
G ::= I[label] \quad \text{Labelled node with id } I
\]
\[
I \stackrel{label}{\rightarrow} J \quad \text{Labelled edge from node } I \text{ to node } J
\]
\[
G; G' \quad \text{Sequence of graph declarations}
\]
Graph Generation

Proposition 2. \( \forall \Gamma. \Gamma \in \text{GSPi}_\Gamma \Rightarrow (|\Gamma|)_I \in \text{DOT} \)

\[
\begin{align*}
(\emptyset)_I & \triangleq \emptyset \\
(|X(\vec{m}) \triangleq C, \Gamma|)_I & \triangleq (|C|)_X; (|\Gamma|)_I \\
(\nu x_1 \ldots \nu x_N C|)_I & \triangleq I[x_1, \ldots, x_N]; (|C|)_I \\
X(\vec{m}) \triangleq C & \Rightarrow (|X(\vec{n})|)_I \triangleq I \xrightarrow{\{\vec{n}/\vec{m}\}} X \\
X(\vec{m}) \triangleq C & \Rightarrow (|X(\vec{n})| \bigoplus |\Pi|)_I \triangleq I \xrightarrow{\{\vec{n}/\vec{m}\}} X; (|\Pi|)_I \\
(0|)_I & \triangleq \emptyset \\
X(\vec{m}) \triangleq C & \Rightarrow (\pi.X(\vec{n}) + \Sigma|)_I \triangleq I \xrightarrow{\pi \{\vec{n}/\vec{m}\}} X; (|\Sigma|)_I
\end{align*}
\]
Generated Graphs: Evolved Network
Related Work

➢ Statecharts [Harel, 1987] highlighted the need for a scalable, self-contained graphical representation of concurrent systems.

➢ Synchronous variant to Statecharts allows concurrent processes to synchronise on shared labels [Andre, 1995].

➢ Foundational graphical representations for pi-calculus use elaborate graph rewriting rules [Milner, 1994].


➢ Preliminary informal ideas on a graphical representation for the stochastic pi-calculus in [Phillips and Cardelli, 2004].
Conclusion

➢ Presented a graphical representation for the stochastic pi-calculus.

➢ Used to model a Mapk signalling cascade and an evolved gene network.

➢ Highlights the existence of cycles, which are key to many biological systems.

➢ Able to animate interactions between biological system components.

➢ Able to clarify the overall system function and to debug changes in the system.
Future Work

➢ Observation: within a collection of mutually recursive definitions, applied arguments often the same as the formal parameters.

➢ Investigate additional design patterns to improve modelling and visualisation techniques.

➢ Use graph generation tool to implement a graphical debugger for SPiM.

➢ Develop a tool for drawing graphical models, which automatically generates SPiM code.

➢ Make modelling and simulation of biological systems more accessible to non computer scientists.
References


