Simulating Biological Systems in the Stochastic Pi-Calculus

Andrew Phillips       Luca Cardelli

29th November 2005
Microsoft Research
7 J J Thomson Avenue
Cambridge, UK
{andrew.phillips,luca}@microsoft.com
Summary of Molecular Biology

DNA

messenger RNA

PROTEIN

SYSTEMS

transcription

translation

interaction

regulation

4-letter digital code

4-letter digital code

20-letter digital code

50,000 (?) shapes
Challenge: Understanding Biological Systems

➢ The Human Genome project has generated vast quantities of data.

➢ Challenge: understand and precisely describe the behaviour of Biological Systems

➢ Traditional modelling tools are non-compositional:
  - if a small part of the system changes, we need to change the entire model.
  - e.g. Differential Equations, Chemical Notations...

➢ Need a more scalable approach.
How Process Calculi Can Help

➢ Ongoing Experiment:
   - Use process calculi to model biological systems

➢ Features of process calculi:
   - *Compositional* modelling, analysis and simulation of systems.

➢ Potential Benefits:
   - *Decompose* complex systems into simpler subsystems.
   - *Analyse* properties of subsystems using established theory.
   - *Predict* behaviour of subsystems by running stochastic simulations.
   - Predict properties and behaviour of *composed* systems.

➢ Pi-calculus: one of the simplest and most well-studied calculi.
Modelling Biological Systems

➢ 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

➢ Chemical examples

➢ Biological examples:
  □ Evolved gene network [Francois and Hakim, 2004]
  □ Mapk signalling cascade [Huang and Ferrel, 1996]
The Stochastic Pi-Calculus (SPi)

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

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

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

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

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

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

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

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

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

\[\begin{align*}
Dec & ::= \text{new } x\{@r\}:t & \text{Channel Declaration} \\
& \mid \text{type } n = t & \text{Type Declaration} \\
& \mid \text{val } m = v & \text{Value Declaration} \\
& \mid \text{run } P & \text{Process Declaration} \\
& \mid \text{let } D_1 \text{ and } \ldots \text{ and } D_N & \text{Definitions, } N \geq 1 \\
D & ::= X(m_1, \ldots, m_N) = P & \text{Definition, } N \geq 0 \\
P & ::= () & \text{Null Process} \\
& \mid (P_1 \mid \ldots \mid P_M) & \text{Parallel, } M \geq 2 \\
& \mid X(v_1, \ldots, v_N) & \text{Instantiation, } N \geq 0 \\
& \mid \pi\{;\ P\} & \text{Action} \\
& \mid \text{do } \pi_1\{;\ P_1\} \text{ or } \ldots \text{ or } \pi_M\{;\ P_M\} & \text{Choice, } M \geq 2 \\
& \mid (Dec_1 \ldots Dec_N \ P) & \text{Declarations, } N \geq 0 \\
\pi & ::= !x\{v_1, \ldots, v_N\} & \text{Output, } N \geq 0 \\
& \mid ?x\{m_1, \ldots, m_N\} & \text{Input, } N \geq 0 \\
& \mid \text{delay } @r & \text{Delay}
\end{align*}\]
The Graphical Stochastic Pi-Calculus (GSPi)

A normal form for SPi, with each summation or guarded process as a definition:

\[ \pi ::= ?x(\tilde{m}) \quad \text{Input} \]
\[ | !x(\tilde{n}) \quad \text{Output} \]
\[ | \tau_r \quad \text{Delay} \]
\[ P, Q ::= P_1 | \ldots | P_N \quad \text{Parallel} \]
\[ | \nu xP \quad \text{Restriction} \]
\[ | X(\tilde{n}) \quad \text{Instance} \]

\[ \Gamma ::= X(\tilde{m}) \triangleq \nu x_1 \ldots \nu x_M (\pi_1.X_1(\tilde{n}_1) + \ldots + \pi_N.X_N(\tilde{n}_N)) \quad \text{Summation} \]
\[ | X(\tilde{m}) \triangleq \nu x_1 \ldots \nu x_M (X_1(\tilde{n}_1) | \ldots | X_N(\tilde{n}_N)) \quad \text{Composition} \]
Graphical Representation: Definitions

➢ 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

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

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

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

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

Restriction as Complexation:

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

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

Reduction in SPi:

\[ !x(\tilde{n}).P + \Sigma | ?x(\tilde{m}).Q + \Sigma' \xrightarrow{\text{rate}(x)} P | Q_{\tilde{n}/\tilde{m}} \]  
\[ \tau_r.P + \Sigma \xrightarrow{r} P \]  
\[ P \xrightarrow{r} P' \Rightarrow P | Q \xrightarrow{r} P' | Q \]  
\[ P \xrightarrow{r} P' \Rightarrow \forall x P \xrightarrow{r} \forall x P' \]  
\[ Q \equiv P \xrightarrow{r} P' \equiv Q' \Rightarrow Q \xrightarrow{r} Q' \]

Reduction in GSPi \( \subset \) SPi:

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

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

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

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

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

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

\[
\begin{array}{c}
X(\tilde{z}) \quad \text{rate}(x) \quad \rightarrow \quad X_1(\tilde{z}) \quad | \quad Y_1(\tilde{z})_{\{n/m\}}
\end{array}
\]
Inline Graphical Reduction: Delay

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

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

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

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

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

\[ X(\tilde{z}) \xrightarrow{r} X_1(\tilde{z}) \]
**Ionization:** \[ Na + Cl \rightleftharpoons Na^+ + Cl^- \]

- Na can ionize Cl by sending its electron, with \( \text{rate}(\text{ionize}) = 100s^{-1} \)
- Cl\(^-\) can deionize Na\(^+\) by sending its electron, with \( \text{rate}(\text{deionize}) = 10s^{-1} \)
- State names \( Na, Na^+, Cl, Cl^- \) are merely annotations
Ionization: $Na + Cl \iff Na^+ + Cl^-$

- $Na$ can ionize $Cl$ by sending its electron on the ionize channel

Andrew Phillips - Imperial College 2005
**Ionization:** $Na + Cl \rightleftharpoons Na^+ + Cl^-$

- $Na^+$ is positively charged and $Cl^-$ is negatively charged.
**Ionization:** \( Na + Cl \rightleftharpoons Na^+ + Cl^- \)

- \( Cl^- \) can deionize \( Na^+ \) by sending its electron on the *deionize* channel

Andrew Phillips - Imperial College 2005
**Ionization**: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

➢ *Na* and *Cl* are no longer charged
Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

A number of $Na$ and $Cl$ atoms can be composed in parallel.
**Ionization:** $Na + Cl \rightleftharpoons Na^+ + Cl^-$

One of the $Na$ atoms can ionize one of the $Cl$ atoms by sending its electron.
**Ionization:** \( Na + Cl \rightleftharpoons Na^+ + Cl^- \)

This produces \( Na^+ \) and \( Cl^- \) ions.
**Ionization:** \( Na + Cl \rightleftharpoons Na^+ + Cl^- \)

Additional \( Na \) and \( Cl \) atoms can interact in parallel.
**Ionization:** \( Na + Cl \rightleftharpoons Na^+ + Cl^- \)

This produces additional \( Na^+ \) and \( Cl^- \) ions.
**Ionization:** $Na + Cl \rightleftharpoons Na^+ + Cl^-$

A $Cl^-$ ion can deionize any of the $Na^+$ ions.
Ionization: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

These reactions can continue indefinitely...

Andrew Phillips - Imperial College 2005
Virtual Experiment: $Na + Cl \rightleftharpoons Na^+ + Cl^-$

Simulations for 100 $Na$ and $Cl$ atoms show that an equilibrium is reached.
Covalent Bonding: \( H + Cl \rightleftharpoons HCl \)

- **H** has a *private* electron \( e \).
- **H** can share its electron with **Cl** to form **HCl**, with \( rate(share) = 100s^{-1} \).
- **HCl** can break its private bond, with \( rate(e) = 10s^{-1} \).
Covalent Bonding: \( H + Cl \rightleftharpoons HCl \)

- \( H \) has a private electron \( e \) that is not accessible from outside.
Covalent Bonding: \( H + Cl \xrightarrow{\text{\(\rightleftharpoons\)}} HCl \)

- \( H \) can share its electron with \( Cl \) on the share channel.
Covalent Bonding: \( H + Cl \rightleftharpoons HCl \)

➢ \( H \) and \( Cl \) share a private electron, to form \( HCl \).
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

➢ $HCl$ can break its private bond by synchronising on channel $e$. 

Andrew Phillips - Imperial College 2005
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

➢ $H$ and $Cl$ are no longer bound
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

A number of $H$ and $Cl$ atoms can be composed in parallel.
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

One of the $H$ atoms can bind with one of the $Cl$ atoms by sharing its electron.
Covalent Bonding: \[ H + Cl \rightleftharpoons HCl \]

This produces an \( HCl \) molecule.
Covalent Bonding: \(H + Cl \rightleftharpoons HCl\)

Additional \(H\) and \(Cl\) atoms can interact in parallel.
Covalent Bonding: \( H + Cl \rightleftharpoons HCl \)

This produces additional \( HCl \) molecules.
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

An $HCl$ molecule can split into $H$ and $Cl$ atoms.
Covalent Bonding: $H + Cl \rightleftharpoons HCl$

These reactions can continue indefinitely...
Virtual Experiment: $H + Cl \rightleftharpoons HCl$

Simulations for 100 $H$ and $Cl$ atoms show that an equilibrium is reached.
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

\[ a(\bar{z}) \triangleq \tau_{\text{transcribeA}} \cdot (A(\bar{z}) \mid a(\bar{z})) \]
\[ A(\bar{z}) \triangleq \forall u \left( \tau_{\text{degradeA}} + !\text{bind}(u) \cdot AB(u) + !\text{inhibit}(u) \cdot Ab(u) \right) \]
\[ Ab(u) \triangleq ?u.A(\bar{z}) \]
\[ AB(u) \triangleq \tau_{\text{degradeAB}} \]
\[ b(\bar{z}) \triangleq \tau_{\text{transcribeB}} \cdot (B(\bar{z}) \mid b(\bar{z})) + !\text{inhibit}(u) \cdot bA(u) \]
\[ bA(u) \triangleq \tau_{\text{transcribeB'}} \cdot (B(\bar{z}) \mid bA(u)) + !u.b(\bar{z}) \]
\[ B(\bar{z}) \triangleq \tau_{\text{degradeB}} + !\text{bind}(u) \cdot BA(u) \]
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())
Initially there is one copy of each gene, $a$ and $b$
Represent the behaviour of each gene as a separate graph
Evolved Gene Network

Gene $a$ can transcribe a new protein $A$ at rate $\text{transcribe}A$
Evolved Gene Network

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 $b$
Evolved Gene Network

Gene $b$ can transcribe a new protein $B$ with rate $\text{transcribe}_B$
A new protein $B$ is transcribed
Evolved Gene Network

Protein $A$ can bind with protein $B$
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
Protein $A$ activates protein $TF$, which stimulates the production of $A$ and $TF$. 
Gene Regulation by Positive Feedback: Definitions

Behaviour of the proteins $A$ and $TF$, and of the DNA and RNA of a protein $P$. 
let Gate(a:chan, b:chan) =
  do delay@transcribe;
      (Protein(b) | Gate(a,b))
  or ?a; Blocked(a,b)
and Blocked(a:chan, b:chan) =
  delay@activate; Gate(a,b)
and Protein(b:chan) =
  do !b; Protein(b)
  or delay@degrade

run ( Gate(a,b) | Gate(b,c) | Gate(c,a) )
Repressilator
Mapk Cascade [Huang and Ferrel, 1996]

➢ System originally described using a set of reaction equations

❑ Converted to ordinary differential equations, 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
Mapk Cascade: Equations

Reaction Equation:

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

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:
Mapk Cascade: SPiM Code

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); KKK_E(d,k)
or !a3(d3,k3); (do ?d3;KKKst() or ?k3;KKKst())
or !a5(d5,k5); (do ?d5;KKKst() or ?k5;KKKst())
)

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() = (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_P_E(d,k)
or !a7(d7,k7); (do ?d7;KK_P() 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 KKPase() = (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 (10 of KKK() | 100 of KK() | 100 of K())
run ( E2() | KKPase() | KPase() | E1() )
Mapk Cascade

!d5
?k5
!a5(d5,k5) d5,k5,d3,k3 !a3(d3,k3) !k

!d3
?k3
!a3(d3,k3)

!d
?a2(k,d)
!k

!k

!d
?a2(k,d)

!d
?a4(k,d)

!d
?a6(k,d)

!d
?a8(k,d)

!d
?a10(k,d)

!a1(d,k)
d1, k1

!a2(d2,k2)
d2, k2

!a4(d4,k4)
d4, k4, d6, k6

!a6(d6,k6)
d6, k6

!a7(d,k)

!a9(d,k)

!a8(d,k)

!a10(d10,k10)

E1

E2

KK

KK_P

KK_PP

KKst

KK

KK

KK

KK

KK

KK

KK

KK

Mapk Cascade
Mapk Cascade

Enzyme $E_1$ can bind to substrate KKK using channel $a_1$
$E_1$ 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*
KKK* can bind with KK using channel $a_3$
Mapk Cascade

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
Mapk Cascade

KKK* can bind to KK-P using channel \( a_5 \)
Mapk Cascade

KKK* is bound to KK-P by channels $d_5$ and $k_5$
Mapk Cascade

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 $a_7$
Mapk Cascade

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 $k_9$
Mapk Cascade

K-P is transformed to K-PP, completing the cascade
Mapk Cascade: Results
Related Work

➢ BioSPI [Priami et al., 2001] is an existing simulator for a biochemical variant of the stochastic \( \pi \)-calculus. The system compiles a \( \pi \)-calculus process to an FCP procedure, which is then executed by the FCP Logix platform [Silverman et al., 1987].

➢ An abstract machine for the stochastic pi-calculus is presented in [Phillips and Cardelli, 2004]. The abstract machine is proved both sound and complete with respect to the calculus, and then used as the basis for implementing the SPiM simulator [Phillips, 2005]. The simulator is implemented in OCaml, which is compiled to native code.

➢ 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 re-writing rules [Milner, 1994].


➢ A graphical representation for the stochastic pi-calculus is presented in [Phillips and Cardelli, 2005]. This is used as the basis for the examples in SPiM.
References


