Biochemical Systems as Reactive Systems

Luca Cardelli
Microsoft Research

Cambridge 2016-02-08
http://lucacardelli.name
Outline

• Processes and Functions
  o The $\pi$-calculus modeling language

• Processes and Chemistry
  o Biochemical modeling in $\pi$-calculus

• Modeling Combinatorial Systems
  o Why $\pi$-calculus and other “agent-based” or “reactive” modeling languages are useful
Processes and Functions
Functions

\[ f(x) = \sqrt{x} \]

"read input into \( x \); then write \( \sqrt{x} \) to output"

function \hspace{1cm} read \hspace{1cm} write \hspace{1cm} channels

\[ f = \text{?input}(x); \text{!output}(\sqrt{x}) \]

(binding) input variable \hspace{1cm} output expression \hspace{1cm} (bound) variable occurrence
Composing Functions

\[ g(x) = (f \circ f)(x) \quad (= f(f(x))) \]

\[ g(x) = \sqrt{\sqrt{x}} \]

Diagram:

- Input: \( x \)
- Output: \( \sqrt{x} \)
- Intermediate: \( \sqrt{\sqrt{x}} \)

Diagram:

- Input: \( x \)
- Output: \( \sqrt{x} \)
- Intermediate: \( \sqrt{\sqrt{x}} \)

Diagram:

- Input: \( x \)
- Output: \( \sqrt{\sqrt{\sqrt{x}}} \)
- Intermediate: \( \sqrt{x} \)
- Intermediate: \( \sqrt{\sqrt{x}} \)
Composing Functions

\[ g(x) = (f \circ f)(x) \]

“create a \textit{new} channel and use it to compose two copies of \( f \)”

\[ g = (\nu \text{ temp}) \]
\[ \begin{array}{l}
\text{?input}(x); \text{!temp}(\sqrt{x}) |
\text{?temp}(y); \text{!output}(\sqrt{y})
\end{array} \]
Many inputs and outputs

\[
P = \text{?in}_1(x); \text{?in}_2(y); \text{!out}_1(x+y); P \oplus \text{?in}_3(z); \text{!out}_2(\sqrt{z}); \text{!out}_1(2z); 0
\]

The ‘skeleton’ automaton
That’s $\pi$–calculus

• To compose processes $P$ we need:
  - Composition: $P | P$ (with identity elem. 0)
  - Channel creation: $(\nu x) P$ (with $x$ bound in $P$)
  - Recursion: $*P$ (equal to $P | *P$)

• To execute actions we need:
  - Channel reading: $?c(x); P$ (with $x$ bound in $P$)
  - Channel writing: $!c(M); P$ (with message $M$)
  - Choice: $P \oplus P$ (with identity elem. 0)

• ... and channels can be sent as messages!
Generalizing Functions and Automata

• Unlike functions…
  o Processes have multiple, explicitly named, input and output channels.
  o Processes can run in parallel, can deadlock on their inputs, and can be nondeterministic in their outputs.

• Unlike automata (FSA)…
  o Processes can transmit data (not just change state).
  o While automata ‘talk’ to input strings, processes ‘talk’ to other processes: processes are communicating automata.
  o Processes are not “finite state”; they can express unbounded computation in time (divergence) and space (proliferation).
Algebraic Properties

• Functions have one binder and one rule:
  o Function application:
    If \( f(x) =_{\text{def}} M\{x\} \) then \( f(a) = M\{a/x\} \)

• Processes have two binders and two rules:
  o Communication (input ‘?’ binder)
    \((?c(x);P\{x\}) \oplus P' \mid (!c(a);Q) \oplus Q' = P\{a/x\} \mid Q\)
  o Scope extrusion (new ‘\(\nu\)’ binder)
    If \( x \) not occurring in \( Q \) then \(((\nu x)P) \mid Q = (\nu x)(P\mid Q)\)
Implementations

- SPiM (Stochastic Pi Machine)
  - Runs in a browser with Silverlight.
Processes and Chemistry
Continuous Chemical Systems

Reactions:

\[ \text{A} \rightarrow^r \text{B}_1 + \ldots + \text{B}_n \]  Degradation

\[ \text{A}_1 + \text{A}_2 \rightarrow^r \text{B}_1 + \ldots + \text{B}_n \]  Asymmetric Collision

\[ \text{A} + \text{A} \rightarrow^r \text{B}_1 + \ldots + \text{B}_n \]  Symmetric Collision

Continuous reaction kinetics, respectively:

\[ [\text{A}]^\cdot = -r[\text{A}] \]  Exponential Decay

\[ [\text{A}_i]^\cdot = -r[\text{A}_1][\text{A}_2] \]  Mass Action Law

\[ [\text{A}]^\cdot = -2r[\text{A}]^2 \]  Mass Action Law

(assuming \( A \neq B_i \neq A_j \) for all \( i,j \))
\(\pi\)-calculus for Chemistry

- Here we just need a *subset* of \(\pi\)-calculus
  - No new-channel (\(\nu\)) operator (except to define delays \(\tau_r\))
  - No value-passing (only synchronization/collision ?/!)

- To compose *soups* \(P\) we need:
  - Stochastic channels: \(X_r\) \(r\) is the rate of an exponential distribution: the rate of communication on that channel
  - Composition: \(P | P\) (with identity elem. \(0\))
  - Recursion: \(*P\) (equal to \(P | *P\))

- To execute *species* we need:
  - Collision: \(?x_r; P\) (with no input variables)
  - Co-collision: \(!x_r; P\) (with no output messages)
  - Delay: \(\tau_r; P\) (\(= (\nu x_r) ?x_r; P | !x_r; 0\) for any \(x\) not in \(P\))
  - Choice: \(P \oplus P\) (with identity elem. \(0\))
Discrete Chemical Systems (1)

Reaction:
\[ A \rightarrow^r B_1 + \ldots + B_n \]

Discrete reaction kinetics:
\[ A = \tau_r; (B_1|\ldots|B_n) \]

The mathematical meaning of that is a Continuous Time Markov Chain (for a specific set of initial conditions, e.g. a single A molecule), here represented as a transition graph:

Hence the \( \pi \)-calculus description abstracts from initial conditions (like ODEs). For each set of initial conditions, a CTMC can be systematically extracted from the stochastic \( \pi \)-calculus models.
Discrete Chemical Systems (2)

(Uniquely named) reaction:
\[ c: \ A_1 + A_2 \rightarrow^r B_1 + \ldots + B_n \]

Discrete reaction kinetics:
\[ A_1 = ?c_r; (B_1|\ldots|B_i) \]  \quad \text{(the name of the reaction becomes the channel)}
\[ A_2 = !c_r; (B_i|\ldots|B_n) \]  \quad \text{(splitting results is arbitrary: } 1 \leq i \leq n)\]

With initial conditions \( A_1|A_2 \) (single molecules), the CTMC is:

\[ r \]
\[ A_1|A_2 \quad B_1|\ldots|B_n \]
Discrete Chemical Systems (3)

(Uniquely named) reaction:

\[ c: \ A + A \rightarrow_r B_1 + \ldots + B_n \]

Discrete reaction kinetics:

\[ A = ?c_{r/2}; (B_1|\ldots|B_i) \oplus !c_{r/2}; (B_i|\ldots|B_n) \quad 1 \leq i \leq n \]

With initial conditions \( A|A \) (two molecules), the CTMC is as follows; note that each copy of \( A \) can do an input or an output, so there are two possible paths to the outcome:

That is:

\[ A|A \rightarrow_r B_1|\ldots|B_n \]
From Reactions to Processes

**Interaction Matrix**

<table>
<thead>
<tr>
<th>channels (1 per reaction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1(k1)</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Processes (1 per species)**

- Degradation reaction \( v_i: X \rightarrow^{k_i} P_i \)
  - add \( 0;P_i \) to \( <X,v_i> \).
- Asymmetric reaction \( v_i: X+Y \rightarrow^{k_i} P_i \)
  - add \( ?;P_i \) to \( <X,v_i> \) and \( !;0 \) to \( <Y,v_i> \).
- Symmetric reaction \( v_i: X+X \rightarrow^{k_i} P_i \)
  - add \( ?;P_i \) and \( !;0 \) to \( <X,v_i> \).

**Partial Processes**

- \( A = ?v_1(k_1);(C|C) \oplus ?v_2(k_2);D \)
- \( B = !v_1(k_1);0 \)
- \( C = !v_2(k_2);0 \oplus \tau_{k_3};(E|F) \)
- \( D = 0 \)
- \( E = 0 \)
- \( F = ?v_4(k_{4/2});B \oplus !v_4(k_{4/2});0 \)
That Chemical System in SPiM

\[
\begin{align*}
A &= \mathcal{v}_1(k_1);(C|C) \oplus \mathcal{v}_2(k_2);D \\
B &= \mathcal{v}_1(k_1);0 \\
C &= \mathcal{v}_2(k_2);0 \oplus \tau_{k_3};(E|F) \\
D &= 0 \\
E &= 0 \\
F &= \mathcal{v}_{4(k4/2)};B \oplus \mathcal{v}_{4(k4/2)};0
\end{align*}
\]

Gillespie-style stochastic simulation

directive sample 10.0

directive plot A(); B(); C(); D(); E(); F()

val k1 = 0.001  new v1@k1:chan
val k2 = 0.001  new v2@k1:chan
val k3 = 1.0
val k4 = 0.001  new v4@k4/2.0:chan

let A() = do ?v1;(C())|C()) or ?v2:D()
and B() = !v1
and C() = do !v2 or delay@k3;(E())|F())
and D() = ()
and E() = ()
and F() = do ?v4:B() or !v4

run 300 of (A()|B()|C()|D()|E()|F())
Model Reduction Techniques

- That is a *systematic* way to translate reactions to processes.
- But there can be *better* ways to do it.
- That is, ways that produce *more compact and/or modular models*, but with the *same kinetics*.
Ex: Catalysis

• Two reactions, same catalyst C
  
  o According to the general scheme the catalyst uses one channel for each reaction it catalyzes:

  a: \[ A + C \rightarrow^r C + B \]

  \[ C = !a_r; C \oplus !b_r; C \]

  b: \[ D + C \rightarrow^r C + E \]

  \[ A = ?a_r; B \]

  \[ D = ?b_r; E \]

  o Modularizing: the catalyst has its own catalysis channel c, used for all the reactions it catalyzes:

  A + C \rightarrow^r C + B

  \[ C = !c_r; C \]

  D + C \rightarrow^r C + E

  \[ A = ?c_r; B \]

  \[ D = ?c_r; E \]
Modeling Combinatorial (Biochemical) Systems
Molecules with State

- Explosion of species, reactions, and their state space.

\[ n \text{ modification sites} = 2^n \text{ molecular states} = 2^n \text{ ‘species’} = 2^n \text{ ODEs} \]
Connected Molecules

- Further combinatorial explosion

\[ n \text{ states} \rightarrow m \text{ states} = n \times m \text{ states} \]

\[ 2^{n_1} \times 2^{n_2} \times \cdots \times 2^{n_m} = \text{BIG} \]
Iterated Connections (Polymers)

- ‘Infinite’ explosion

An actually infinite number of species and ODEs

\[ p_1 \quad \text{(polymer of length 1)} \]
\[ p_2 \quad \text{(polymer of length 2)} \]
\[ p_3 \quad \text{(polymer of length 3)} \]

... 

Copolymer equation

An alternating copolymer has the formula \(-A-B-A-B-A-B-A-B\) or \(-A-B-A-B-A-B\)\(_\text{in}^\text{2}\). The molar ratios of the monomer in the polymer is close to one, which happens when the reactivity ratios \(r_1\) & \(r_2\) are close to zero, as given by the Mayo-Lewis equation also called the copolymerization equation.\([11]\)

\[
\frac{d[M_1]}{d[M_2]} = \frac{[M_1] (r_1 [M_1] + [M_2])}{[M_2] ([M_1] + r_2 [M_2])}
\]

where \(r_1 = k_{11}/k_{21} \) & \(r_2 = k_{12}/k_{21}\)
π–calculus for Biochemistry

• Biochemistry here means
  o *Direct* modeling of complexation and polymerization, which are fundamental biochemical features.
  o That is, a complex is not a “new species”: it is a structure formed by existing basic species, which can also break apart.

• We now need the *full* π–calculus
  o We need to create new channels to represent new *complexation bonds*.
  o We need value–passing so the components of a complex can operate on those bonds: we need to pass *channels over channels*.
Complexation

\[ \text{A} + \text{B} \xleftrightarrow{\text{s}, \text{r}} \text{A:B} \]

There is no good notation for this reaction in chemistry: \text{A:B} is considered as a separate species (which leads to combinatorial explosion of models).

But there is a way to write this precisely in \( \pi \)-calculus. Let there be a single public association channel \( a_r \) at rate \( r \), and many private dissociations channels \( d_s \) at rate \( s \), one for each complexation event (these are dynamically created by the new–channel operator \( \nu \)):

\[
\begin{align*}
\text{A}_{\text{free}} &= (\nu \ d_s) \ !a_r(d_s); \ \text{A}_{\text{bound}}(d_s) \\
\text{A}_{\text{bound}}(d_s) &= !d_s; \ \text{A}_{\text{free}} \\
\text{B}_{\text{free}} &= ?a_r(d_s); \ \text{B}_{\text{bound}}(d_s) \\
\text{B}_{\text{bound}}(d_s) &= ?d_s; \ \text{B}_{\text{free}}
\end{align*}
\]

Note that we are describing \text{A} independently of \text{B}: as in the catalysis example, \text{A} could form complexes with many different species over the \( a_r \) channel.

More compactly:

\[
\begin{align*}
\text{A} &= (\nu \ d_s) \ !a_r(d_s); \ !d_s; \ \text{A} \\
\text{B} &= ?a_r(d_s); \ ?d_s; \ \text{B}
\end{align*}
\]
Polymerization

- Polymerization is iterated complexation
  - It can be represented in π-calculus \textit{finitely}, with \textit{one process (definition)} for each monomer.
  - Note that polymerization cannot be described \textit{finitely} in chemistry (or ODEs) because there it needs one reaction for each \textit{length} of polymer.
  - The reason it works in π-calculus is because of the \( \nu \) operator. It enables the finite representation of systems of potentially unbounded complexity.
  - Like in real biochemistry, where the structure of each monomer is coded in a finite piece of DNA, and yet unbounded-length polymers happen.
Conclusions
π–Calculus

• A solution to combinatorial explosion
  o Pi–calculus does not have the typical combinatorial problems, at least not when you are writing a model.
  o Models are exponentially (for phosphorylation/complexation) or infinitely (for polymerization) more compact.
  o The combinatorial explosion still happens at simulation time, but can be handled ‘on demand’.
  o The state space is explored incrementally, and even if the state space is actually infinite (as with polymers) we can still simulate it with standard techniques.

• Shared by any “agent–based” modeling language
  o Provided it is sufficiently powerful to directly represent biochemical situations like complexation
  o I.e. NOT shared by chemical reactions (or ODE) languages
In Summary

• $\pi$–calculus
  o A mathematical notation for reactive systems
  o In stochastic form, suitable for representing discrete chemistry, biochemistry, etc.
  o Some unique properties: ability to finitely express systems of unbounded complexity, like networks of complexing proteins.

• Further Reading
  o R. Milner: Communicating and Mobile Systems: The Pi Calculus
  o L. Cardelli: From Processes to ODEs by Chemistry. TCS 2008, DOI: http://dx.doi.org/10.1007/978-0-387-09680-3_18