From Process Algebras to Differential Equations and return

Luca Bortolussi\textsuperscript{1}  
Alberto Policriti\textsuperscript{1,2}

\textsuperscript{1}Department of Mathematics and Computer Science  
University of Udine, Italy.  
\textsuperscript{2}Center for Biomolecular Medicine  
Trieste, Italy.

Simulation and Verification of Dynamic Systems, Schloss Dagstuhl, 17\textsuperscript{th} – 22\textsuperscript{nd} April 2006
Outline

1. **Introduction**
   - Motivations
   - S-Systems
   - Stochastic $\pi$-calculus
   - Stochastic Concurrent Constraint Programming

2. **From Process Algebras to Differential Equations**
   - Gillespie Algorithm as a Continuous Time Markov Chain
   - Observables and Traces
   - Parameters of S-Systems

3. **From Differential Equations to Process Algebras**
   - S-Systems as rates
Outline

1. Introduction
   - Motivations
   - S-Systems
   - Stochastic $\pi$-calculus
   - Stochastic Concurrent Constraint Programming

2. From Process Algebras to Differential Equations
   - Gillespie Algorithm as a Continuous Time Markov Chain
   - Observables and Traces
   - Parameters of S-Systems

3. From Differential Equations to Process Algebras
   - S-Systems as rates
Introduction
From Process Algebras to Differential Equations
From Differential Equations to Process Algebras

Motivations

Discrete vs. Continuous

- Stochastic process algebras ($\pi$-calculus)
  - stochastic discrete “internal” modeling

- (Ordinary) Differential Equations ($S$-Systems)
  - deterministic continuous “external” modeling

Luca Bortolussi, Alberto Policriti
Motivations

Long Term Goal

Studying the relationship between the two modeling techniques (behavioral expressivity, interchangeability, hybrid modeling).

Short Term Goal

Studying the parameters, their connection and their determination (stochastic rates, differential equations parameters).
A biological system usually involves

- dependent reactants (variables) whose values are affected by the system
- independent reactants that are not affected by the system

S-systems describe the dynamical behavior of a biological system by a set of differential equations over reactants

- non-linear, time-invariant, DAE systems;
- biologically plausible and expressive (see Savageau, Voit, ...);
- they are an interchangeable component of the approach


**Definition**

An **S-system** is a tuple $S = (DV, IV, DE, C)$ such that:

- $DV = \{X_1, \ldots, X_n\}$ dependent variables
- $IV = \{X_{n+1}, \ldots, X_{n+m}\}$ independent variables
- $DE$ is a set of differential equations of the form
  \[ \dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}} \]
  with $\alpha_i, \beta_i \geq 0$ called *rate constants*;
- $C$ is a set of algebraic constraints
S-Systems — the Repressilator

The Repressilator:
a cyclic, three-repressor, transcriptional network

TetR

λ cI ——|— LacI

PC gene A PA gene B PB gene C

mRNA A mRNA B mRNA C

protein A protein B protein C
S-Systems — the Repressilator equations

\[ \dot{X}_1 = X_4 X_3^{-1} - X_1^{0.5}, \quad X_4 = 0.2, \]
\[ \dot{X}_2 = X_5 X_1^{-1} - X_2^{0.578151}, \quad X_5 = 0.2, \]
\[ \dot{X}_3 = X_6 X_2^{-1} - X_3^{0.5}, \quad X_6 = 0.2. \]
Biology and $\pi$-calculus

The syntax of $\pi$-calculus

$P, Q ::= \nu x P \quad $ Restriction $\quad \Sigma ::= 0 \quad $ Null
$\mid P|Q \quad $ Parallel $\quad | \pi.P + \Sigma \quad $ Action
$\mid \Sigma \quad $ Summation $\quad \pi ::= x\langle n \rangle \quad $ Output
$\mid !\pi.P \quad $ Replication $\quad | x(m) \quad $ Input, $x \neq m$

Quantitative aspect: interaction “rates” assigned to channels.


Can we use $\pi$-calculus for (biological) simulation?

<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 cellular components)</td>
<td>(state transition systems)</td>
</tr>
</tbody>
</table>

In this process algebra, the main object are **constraints**, which are formulae over an interpreted first order language (i.e. $X = 10$, $Y > X - 3$).

Constraints can be added to a "pot", called the **constraint store**, but can never be removed.

Agents can perform two basic operations on this store (**asynchronously**):

- Add a constraint (**tell** ask)
- Ask if a certain relation is entailed by the current configuration (**ask** instruction)

**Syntax of CCP**

$$
\begin{align*}
Program &= Decl.A \\
D &= \epsilon \mid Decl.Decl \mid p(x) : \neg A \\
A &= 0 \\
&\mid \text{tell}(c).A \\
&\mid \text{ask}(c_1).A_1 + \text{ask}(c_2).A_2 \\
&\mid A_1 \parallel A_2 \mid \exists x.A \mid p(x)
\end{align*}
$$

Stochastic Concurrent Constraint Programming

Stochastic CCP

Syntax of Stochastic CCP

\[
\begin{align*}
\text{Program} &= \text{Decl}.A \\
D &= \varepsilon \mid \text{Decl}.\text{Decl} \mid p(x) : -A \\
A &= 0 \mid \text{tell}_\lambda(c).A \mid \text{ask}_\lambda(c).A \mid [p(x)]_\lambda \\
&\quad \mid \exists x A \mid (A_1 + A_2) \mid (A_1 \parallel A_2)
\end{align*}
\]


Stochastic information

Each basic instruction (tell, ask, procedure call) has a rate attached to it.

Rates are functions from the constraint store \(C\) to positive reals:

\[
\lambda : C \rightarrow \mathbb{R}^+.
\]

Why another Process Algebra?

- Constraints are powerful and easy to program.
- Easy to simulate (up to now, prototype engine in Prolog).
- We can use “clever” stochastic rates.
An example — Repressilator
Repressilator in $\pi$

L. Cardelli - Sept. 2005
Repressilator in sCCP

degradator(X) :- tell_{degRate}(X)(degrade(X)).degradator(X)

neg(X, Y) :- ( tell_{prodRate}(X)(produce(X))
+ ask_{inhibitRate}(Y)(Y > 0).ask_{delayRate}(X)(true)
).neg(X, Y)

neg_gate(X, Y) :- neg(X, Y) || degradator(X)
Outline

1. Introduction
   - Motivations
   - S-Systems
   - Stochastic π-calculus
   - Stochastic Concurrent Constraint Programming

2. From Process Algebras to Differential Equations
   - Gillespie Algorithm as a Continuous Time Markov Chain
   - Observables and Traces
   - Parameters of S-Systems

3. From Differential Equations to Process Algebras
   - S-Systems as rates
A **Continuous Time Markov Chain** (CTMC) is a direct graph with edges labeled by a real number, called the **rate of the transition** (representing the speed or the frequency at which the transition occurs).

- In each state, we select the next state according to a *probability distribution* (*jumping chain*) obtained normalizing rates (from $S_0$ to $S_1$ with prob. $\frac{\lambda_1}{\lambda_1 + \lambda_2}$).

- The *time* spent in a state is given by an exponentially distributed random variable (*holding time*), with rate given by the *sum of outgoing transitions* from the actual node ($\lambda_1 + \lambda_2$).
Gillespie Algorithm as a Continuous Time Markov Chain

Gillespie Algorithm

The approach of Gillespie is based on an estimate of the average probability that a particular pair of molecules interact in the next vanishingly small $\delta t$.

$X_1, \ldots, X_n$: reactant quantities
$R_1, \ldots, R_m$: reactions
$c_\mu$: basic rate of reaction $\mu$
$h_\mu$: product of reactants for $\mu$
$a_\mu = c_\mu h_\mu$: actual rate of $\mu$
$a = \sum_\mu a_\mu$.

The probability density function $P(\tau, \mu)$ is the probability that, given the state $(X_1, \ldots, X_N)$ at time $t$, the next reaction will occur in the infinitesimal time interval $(t + \tau, t + \tau + dt)$, and will be an $R_\mu$ reaction.

$$P(\tau, \mu) = P_1(\tau)P_2(\mu);$$
$$P_1(\tau) = ae^{-a\tau}; P_2(\mu) = \frac{a_\mu}{a}.$$  

Given $r_1$ and $r_2$ uniform random numbers in $[0, 1]$, $\tau$ and $\mu$ are determined by:

$$\tau = \frac{1}{a_0} \log \left( \frac{1}{r_1} \right)$$
$$\sum_{\nu=1}^{\mu-1} a_\nu < r_2 a_0 \leq \sum_{\nu=1}^{\mu} a_\nu.$$
Gillespie Algorithm as a Continuous Time Markov Chain

**Gillespie Algorithm as a CTMC**

- **State space** $S$: the set of all possible reactant quantities.
- **Edges**: one for each active reaction $R_\mu$, with associated rate $a_\mu$.

- The probability of taking an edge of type $R_\mu$ is $\frac{a_\mu}{a}$ (*jumping chain*).
- The time spent in a state is distributed exponentially with rate $a$ (*holding time*).
Observables and Traces

**Observables in \( \pi \)-calculus simulations**

**Configurations**

\( C \) is the set of different configurations in which a (stochastic) \( \pi \)-calculus simulation can be.

**Observable Terms**

\( T = \{ \tau_1, \ldots, \tau_n \} \) is the set of terms (states of processes) we want to count.

**Observable Function**

\[
\Phi : C \times T^k \rightarrow \mathbb{N}^k; \quad \Phi(c, \langle \tau_{i_1}, \ldots, \tau_{i_k} \rangle) = \langle n_1, \ldots, n_k \rangle,
\]

\( \Phi \) returns the number of instances of \( \tau_{i_1}, \ldots, \tau_{i_k} \) \((k \leq n)\) in the configuration \( c \).

**Traces and Observable Traces**

\[
T = \{ c_0 t_0 c_1 t_1 \ldots c_n t_n \ldots \mid c_i \in C, t_i \in \mathbb{R}^+, c_i \rightarrow c_{i+1} \}.
\]

\( T \) is the set of (timed) traces of a \( \pi \)-calculus program. It can be lifted to a set of observable traces \( T_{\Phi, \bar{\tau}} \) (where \( \bar{\tau} = \langle \tau_{i_1}, \ldots, \tau_{i_k} \rangle \)).

\[
T_{\Phi, \bar{\tau}} = \{ \Phi(c_0, \bar{\tau}) t_0 \ldots \Phi(c_n, \bar{\tau}) t_n \ldots \mid c_0 t_0 \ldots c_n t_n \ldots \in T \}.
\]
Parameters of S-Systems can be identified from a set of temporal series of measurements, using well established techniques.

The same biological network can be modeled both in $\pi$ calculus (or sCCP) and with S-Systems. *If we have few experiments, it may be easier to build a process algebra model* (less parameters involved, more stable systems).

We can use the $\pi$ calculus model to generate observable traces that can be used to train the S-System parameters. *We are seeking a mathematical law describing the $\pi$-model.*

After rescaling, we can train the S-System with both $\pi$-traces and real measurements.
Jane and Jeremy showed us how one can associate to a PEPA or a $\pi$-calculus model (under some restrictions) a set of ODEs describing its behaviour.

What’s the difference?

- Their approach is “exact”, ours is “approximate”.
- Their approach needs to associate a variable to each different syntactical term present in the model (also to control terms without any biological meaning), while we can focus only on a subset of “key” terms.
- The class of differential equations involved in their approach is the mass action one, while we can use different classes (e.g. S-Systems).

Outline

1. Introduction
   - Motivations
   - S-Systems
   - Stochastic \( \pi \)-calculus
   - Stochastic Concurrent Constraint Programming

2. From Process Algebras to Differential Equations
   - Gillespie Algorithm as a Continuous Time Markov Chain
   - Observables and Traces
   - Parameters of S-Systems

3. From Differential Equations to Process Algebras
   - S-Systems as rates
Using S-Systems to determine rate functions

A generic form for S-System equations is

\[ \dot{X}_i = V^+(X_1, \ldots, X_{m+n}) - V^-(X_1, \ldots, X_{m+n}). \]

production speed degradation speed

Rate equations have the same format:

\[ \dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_{ij}^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_{ij}^{h_{ij}}. \]

This is the format of rates in Gillespie algorithm!

A generic S-System equation has non-linear dependencies on variables.

\[ \dot{X}_i = \alpha_i \prod_{j=1}^{n+m} X_{ij}^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_{ij}^{h_{ij}}. \]

We can use these expression as rates.

Using sCCP, we can associate to each dependent variable an agent:

\[ \text{substance}(X_i) ::= (\text{produce}(X_i) \prod_{j=1}^{n+m} X_{ij}^{g_{ij}} + \text{degrade}(X_i) \prod_{j=1}^{n+m} X_{ij}^{h_{ij}}).\text{substance}(X_i). \]
S-Systems as rates

An example — Repressilator (Again)

\[ \dot{X}_1 = X_4 X_3^{-1} - X_1^{0.5}, \]
\[ \dot{X}_2 = X_5 X_1^{-1} - X_2^{0.5}, \]
\[ \dot{X}_3 = X_6 X_2^{-1} - X_3^{0.5}. \]

\[
\text{substance}(X_1) ::= (\text{produce}(X_1)_\alpha X_3^{-1} + \text{degrade}(X_1)_X^{0.5} \cdot \text{substance}(X_1))
\]
\[
\text{substance}(X_2) ::= (\text{produce}(X_2)_\alpha X_1^{-1} + \text{degrade}(X_2)_X^{0.5} \cdot \text{substance}(X_2))
\]
\[
\text{substance}(X_3) ::= (\text{produce}(X_3)_\alpha X_2^{-1} + \text{degrade}(X_3)_X^{0.5} \cdot \text{substance}(X_3))
\]
S-Systems as rates

Repressilator gone wild

S-System’s model of repressilator suffers from an high sensitivity from parameters, differently from the usual PA models. sCCP model with variable rates has the same “wild” behaviour!
Conclusions

- We are focusing on the relationships between two different modeling techniques for biological systems: SPA from one side and ODE (S-Systems) on the other.
- We introduced a stochastic version of Concurrent Constraint Programming, that has complete freedom in defining rates of transitions.
- We are studying a method for identifying an equivalent S-System starting from a SPA model.
- We are studying methods to map differential equations directly into SPA description.
Up to now, the passage from DE to PA is based on a generalization of the concept of rates, but the structure of the agents is simple (control mechanism hidden on rates).

THE BIG QUESTION

Is it possible to identify the logical mechanism regulating a system, starting from a set of differential equations? (find a set of agents with an interesting internal structure, that exhibit the same behaviour)

A possible solution may pass through symbolic dynamical systems.