Perspectives on Concurrency and Biological Sciences

Luca Bortolussi

1Department of Mathematics and Informatics
University of Udine, Italy.

SISSA, Trieste, 12th December 2005
Outline

1. Concurrency and System Biology
   - Concurrency and Computations
   - $\pi$-calculus
   - Modeling Biological Systems
   - Simulation and Semantics

2. Increasing Expressiveness
   - Concurrent Constraint Programming
   - Encoding Protein Folding
   - A Glance on Concurrent Optimization
   - Again on Modeling Biological Systems

3. Beyond Simulations
   - Behaviours, Syntax and Dynamical Systems
   - Checking the Models
Outline

1. Concurrency and System Biology
   - Concurrency and Computations
   - $\pi$-calculus
   - Modeling Biological Systems
   - Simulation and Semantics

2. Increasing Expressiveness
   - Concurrent Constraint Programming
   - Encoding Protein Folding
   - A Glance on Concurrent Optimization
   - Again on Modeling Biological Systems

3. Beyond Simulations
   - Behaviours, Syntax and Dynamical Systems
   - Checking the Models
Old and New *Computation*


**Old Computing**
- (formal) analysis of the (informal) notion of *computability*
- Sequential

**New Computing**
- (formal) evolution of the (informal) notion of *information passing*
- Concurrent

Prescription
Hierarchical design
Determinism
End-result
(Extension)

... Description
... Heterarchical phenomena
... Non-determinism
... Continuing interaction
(Extension)
## Computation vs. Interaction

<table>
<thead>
<tr>
<th></th>
<th>COMPUTATION</th>
<th>INTERACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACTIVE ENTITY P:</strong></td>
<td>Program</td>
<td>Active object, Agent</td>
</tr>
<tr>
<td><strong>ITS MEANING:</strong></td>
<td>Function</td>
<td>Process</td>
</tr>
<tr>
<td><strong>STATICS (COMBINATION):</strong></td>
<td>Sequential Composition</td>
<td>Parallel Composition</td>
</tr>
<tr>
<td></td>
<td>$P_1; P_2$</td>
<td>$P_1 \parallel P_2$</td>
</tr>
<tr>
<td><strong>DYNAMICS (ACTION):</strong></td>
<td>Operation on datum</td>
<td>Message</td>
</tr>
</tbody>
</table>
Computing by Exchanging Information

There are two basic entities: *agents* and *names*

Agents interact by *exchanging names* through communication channels (*message-based computation*).

Computation resides in the evolution of the status of the agents, by means of message passing (*behavioral computation*).

That’s all!

---

\( \pi \)-calculus: Graphical Syntax

\[
P, Q ::= \quad P^n \\
\begin{array}{c}
P \\
\hline
Q
\end{array}
\quad \text{Restriction}
\]

\[
\sum ::= \quad \bigcirc \\
\quad \text{Null}
\]

\[
\begin{array}{c}
P \\
\hline
Q
\end{array}
\quad \text{Parallel}
\]

\[
\sum \\
\quad \text{Summation}
\]

\[
\pi ::= \quad x^{(n)} \\
\quad \text{Output}
\]

\[
x^{(m)} \\
\quad \text{Input}
\]

**Stochastic π-calculus (C. Priami)**

Can we use π-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>
</tbody>
</table>

(of cellular components) (state transition systems)


Quantitative aspects must enter the picture: interaction "rates" assigned to channels.

Example: $H + Cl$

Covalent Bonding: $H + Cl \rightleftharpoons HCl$

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

Covalent Bonding: $H + Cl \rightleftharpoons HCl$

This produces additional $HCl$ molecules.
Another Example: Repressilator

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

Luca Bortolussi
Perspectives on Concurrency and Biological Sciences
Modeling Biological Systems

Another Example: Repressilator

```
L. Cardelli - Sept. 2005
```
How does it work?

The Execution Model

- A state is the collection of all agents present at a given point in the simulation.
- In every state consider all pairs of agents ready to communicate on channels. These are the active transitions.
- Channel’s rates are associated to an exponential probability distribution over non-negative reals (time).
- A graph can be build (state-transition graph), where the vertices are the states and each enabled transition is an edge, labeled by the corresponding rate.
How does it work?

The Simulation

- In every state, one transition is chosen among the active ones by running a race condition: the quickest communication wins and it is executed.
- The execution of a program corresponds to the execution of a Continuous Time Markov Chain.
- It’s a linear time evolution in the state space.
- SPiM is an Abstract Machine that implements this mechanism to perform simulations.

Pro and Cons

Pro

- The modeling technique is **modular and compositional**. The same model (of a protein, of a gene) can be used in different contexts.
- It’s based on a well developed theory, and on an attractive metaphor.
Pro and Cons

Cons

- The behaviour of the system depends on the rates, and it is not so clear how to assign them.
- There is the possibility of modeling only the interaction among biological entities. It is not natural to introduce a form of memory or some (algorithmical) computational mechanism.
- Modeling complex interactions can be very difficult.
Outline

1. Concurrency and System Biology
   - Concurrency and Computations
   - $\pi$-calculus
   - Modeling Biological Systems
   - Simulation and Semantics

2. Increasing Expressiveness
   - Concurrent Constraint Programming
   - Encoding Protein Folding
   - A Glance on Concurrent Optimization
   - Again on Modeling Biological Systems

3. Beyond Simulations
   - Behaviours, Syntax and Dynamical Systems
   - Checking the Models
How to Increase Expressiveness

- Add algorithmical features to $\pi$-calculus.
- Add a form of local storage.
- Keep separated the description of interactions and the description of memory and computations.
Constraint Programming is a declarative programming methodology where problems are encoded by sets of constraints (e.g., $X \leq Y$, $|X - Y| \geq 10$, \{X + Y\} $\subseteq A \cup B$).

Optimization problems are naturally encoded and in this case computations are traversing of search trees pruned by unsatisfiability of constraints.

In general, variables do not have necessarily a unique value but a set of values that makes the constraints satisfiable. Computations are, in a sense, manipulations of sets of values.

Classical computations can be reobtained by using (list of) sets containing just one point.
Concurrent Constraint Programming combines together a calculus of interactions and the algorithmical power of constraints.

Agents interact by telling constraints in a store (constraint store), and by asking if certain relation are verified (entailed) by the current set of constraints.

Communication proceeds via the store, and it is asynchronous.

The store is a form of memory, and it has algorithmical capabilities.

Syntax of Concurrent Constraint Programming

\[ \text{Decl} ::= \varepsilon \mid \text{Decl}.\text{Decl} \mid p(x) : \neg A \]

\[ \pi ::= \text{tell}(c) \mid \text{ask}(c) \]

\[ A ::= \pi.A \mid A + A \mid A \parallel A \mid \exists x A \mid p(x) \]

\( c \) is a constraint belonging to a Constraint System, which is a mathematical structure encoding the algorithmical aspects of constraints.
The basic language must be extended with a synchronous form of communication (message-based interaction), and with stochastic features, in order to be usable for modeling biological systems.


The stochastic version of CCP is a powerful language. In fact, *we can even simulate (a simplified version of) the Protein Folding.*

We use a **coarse-grained model**: each aminoacid (aa) is a single center of interaction, energy function is simple.

The idea behind a concurrent encoding is to see if we can have some help in predicting the structure of a protein by “mimicking” the concurrent nature of the real folding process.


The Encoding

- Each aa is associated to an independent agent.
- Each aa has a memory: it stores its spatial position.
- Each aa has algorithmical power: it can compute the energy of the system (known to it).
- The interaction is obtained by letting aa exchange information about their position.
The Encoding

- aa move in the space following a **Monte Carlo dynamics**.
- The **environment** contains information about the temperature of the system.
- There are also **“strategic” agents**, with the task of **coordinating** the dynamics, and making the aa **cooperate** form “good” local structures.
Does It Work?

- There is a CCP version of the program, and also a low level implementation (faster!). The program is intrinsically parallel!
- The system was tested on several proteins, and on different energy functions.
- Results are encouraging, though the use of better strategies may improve the simulation.

1PG1, native structure.

1PG1, predicted structure.
The protein folding encoding can be seen as a concurrent optimization algorithm. In particular, it is a hybrid form of simulated annealing.

In general, CCP is particularly adapt to describe concurrent optimization heuristics: the search space can be modeled independently from the search strategy.

The same programs can be used to search different spaces!
The ultimate goal is to use *state-space analysis techniques* to automatically *understand some properties* of different concurrent exploration heuristics for a given search space. In this way we hope to *suggest the best heuristic* (or combination of heuristics), and *tune automatically parameters*.
The stochastic extension of Concurrent Constraint Programming can be used to model biological systems, similarly to $\pi$-calculus.

$\pi$-calculus for system biology

<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>
The stochastic extension of Concurrent Constraint Programming can be used to model biological systems, similarly to $\pi$-calculus.

**CCP for Biological Simulation**

- Molecule $\leftrightarrow$ Process
- Direct Interaction capability $\leftrightarrow$ Channel
- Interaction $\leftrightarrow$ Synchronous Communication
- Modification $\leftrightarrow$ State change
- Environment (of cellular components) $\leftrightarrow$ Constraint Store
- Interaction with Environment $\leftrightarrow$ Asynchronous Communication

(state transition systems and memory)
Pro and Cons

Pros

- **Compositionality and modularity.**
- There is a form of environment, and processes have a direct form of memory and some direct computational capabilities.
- Model building should be more flexible.

Cons

- *The system evolution is still linear* (in the state space), though non-linearity can be inserted at the level of (the operations on) the constraint store.
- Working with constraints may not be simple. A form of graphical interface is needed.
Outline

1. Concurrency and System Biology
   - Concurrency and Computations
   - $\pi$-calculus
   - Modeling Biological Systems
   - Simulation and Semantics

2. Increasing Expressiveness
   - Concurrent Constraint Programming
   - Encoding Protein Folding
   - A Glance on Concurrent Optimization
   - Again on Modeling Biological Systems

3. Beyond Simulations
   - Behaviours, Syntax and Dynamical Systems
   - Checking the Models
Searching for Limits

Expressivity of $\pi$

- The modeling approach with $\pi$-calculus (or with CCP) seems very attractive. But *is it able to reproduce (qualitatively) all the emergent behaviours we see in biological systems?*
- And how we can *compare the expressivity* of these methods?
Searching for Limits

Symbolic Dynamical Systems

- Idea: all these concurrent languages can be seen as symbolic dynamical systems.
- This is closely related to the “behavioral” idea of computation as information exchange.
- It focuses on the relation between syntax and dynamical behaviors.
- It can be seen as a general methodology for studying the expressiveness of concurrent languages.
What can we do with a discrete and stochastic model of a biological system?

We can run some simulations and have an idea of the general behaviour.

But can we do more?

We would like to “solve” the model, understand some of its emerging properties in an exact way. This can be done using *model checking* techniques.
The need of a Formal Language

We need a way to express formally the properties we want to verify. They involve both behaviors in time, quantitative information and probabilistic information.

Using Temporal Logic

We can use a temporal logic, a logic with (modal) operators to talk about time (linear or branching). This must be extended with operators capable of expressing probabilistic information (probability, averages), and quantitative information.
Model Checking

State Space Exploration

- The logical formulas can be verified (checked) in the state transition graph of the model, by essentially an exhaustive exploration.

- The problem is the huge number of those states (state space explosion). There are, however, a lot of techniques to tackle this: we can check state spaces of size $10^{50}$!

- There are versions of probabilistic model checking, that must be adapted to the languages and the context under consideration.
## Checking the Models

### An Example

**What Properties?**

Here are some examples of queries that can be asked to a model checking system:

- With which probability the level of expression of gene \( a \) is always above \( x \)?
- With which probability the expression level of genes \( a, b, c \) will eventually stabilize into a limit cycle?
- Is it true that the average expression level of gene \( d \) is above \( x \)?
THANKS FOR THE ATTENTION!

QUESTIONS?