CONSTRAINT-BASED SIMULATION OF BIOLOGICAL SYSTEMS DESCRIBED BY MOLECULAR INTERACTION MAPS

Luca Bortolussi¹    Simone Fonda⁴    Alberto Policriti²,³

¹Dipartimento di Matematica ed Informatica
   Università degli studi di Trieste
   luca@DMI.units.it
²Dipartimento di Matematica ed Informatica
   Università degli studi di Udine
³Istituto di Genomica Applicata
   Parco Scientifico Tecnologico, Udine.
⁴Dipartimento di Informatica
   Università degli studi di Pisa.

BIBM 2007, Silicon Valley, 3th November 2007
Views of Computational Systems Biology

Description Language

- Graphical Languages: Molecular Interaction Maps
- Stochastic Process Algebras: sCCP
- Basic math. language

Mathematical model
**Outline**

1. **Middle Level Language: Stochastic Concurrent Constraint Programming**

2. **Higher Level Language: Molecular Interaction Maps**

3. **Encoding MIMs in sCCP**
1. **Middle Level Language: Stochastic Concurrent Constraint Programming**

2. **Higher Level Language: Molecular Interaction Maps**

3. **Encoding MIMs in sCCP**
**STOCHASTIC CONCURRENT CONSTRAINT PROGRAMMING**

**CCP = Constraints + Agents**

- **Constraints** are formulae over an interpreted first order language (i.e. $X = 10, Y > X - 3$); they can be added to a "container", the constraint store, but can never be removed.

- Agents can perform two basic operations on this store (asynchronously): **tell** or **ask** a constraint.

```plaintext
rw(X):- ask(X > 0).
       tell(X' = X - 1).rw(X)
       + tell(X' = X + 1).rw(X)
```

**STOCHASTIC CCP**

Each **ask** and **tell** instruction has a rate (function) attached to it:

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

The semantics of the language is given in terms of a Continuous Time Markov Chain.

Stochastic Concurrent Constraint Programming

CCP = Constraints + Agents

- **Constraints** are formulae over an interpreted first order language (i.e. \( X = 10, \ Y > X - 3 \)); they can be added to a "container", the constraint store, but can never be removed.

- Agents can perform two basic operations on this store (asynchronously): **tell** or **ask** a constraint.

\[
\text{rw}(X) : \begin{array}{l}
\text{ask}_{\lambda(X)}(X > 0).
\text{tell}_{\infty}(X' = X - 1).\text{rw}(X)
+ \text{tell}_{\lambda(X)}(X' = X + 1).\text{rw}(X)
\end{array}
\]

**STOCHASTIC CCP**

Each **ask** and **tell** instruction has a rate (function) attached to it:

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

The semantics of the language is given in terms of a Continuous Time Markov Chain.

MODELING IN sCCP

MODELING BIOCHEMICAL REACTIONS

\[ R_1 + \ldots + R_n \rightarrow_{f(R, X; k)} P_1 + \ldots + P_m \]

\[ f\text{-reaction}(R, X, P, k) :- \]
\[ \text{tell}_{f(R, X; k)}(R' = R - 1 \land P' = P + 1). \]
\[ f\text{-reaction}(R, X, P, k) \]

ANALYSIS TOOLS

- Stochastic simulation (Gillespie algorithm)
- Stochastic model checking and CTMC analysis
- Approximation with ODE’s and Hybrid Automata

OREGONATOR

\[
\begin{align*}
B & \rightarrow_{k_1} A \\
A + B & \rightarrow_{k_2} \emptyset \\
A & \rightarrow_{k_3} 2A + C \\
2A & \rightarrow_{k_4} \emptyset \\
C & \rightarrow_{k_5} B
\end{align*}
\]

**Outline**

1. **Middle Level Language: Stochastic Concurrent Constraint Programming**

2. **Higher Level Language: Molecular Interaction Maps**

3. **Encoding MIMs in sCCP**
Maps and their Interpretation

Explicit
- A:B
- B:C
- pB

Combinatorial
- pB
- A:B
- B:C
- A:B:C
- A:pB
- pB:C
- A:pB:C

**Combinatorial Explosion**

**Explicit**
1 reaction

\[ A + B \rightarrow A:B \]

**Combinatorial**
4 reactions

\[ A + B \rightarrow A:B \]
\[ A + pB \rightarrow A:pB \]
\[ A + B:C \rightarrow A:B:C \]
\[ A + pB:C \rightarrow A:pB:C \]
CONTINGENCIES

Interpretation $\neq$ formal semantic
CONTINGENCIES

Interpretation $\neq$ formal semantic
1. **Middle Level Language: Stochastic Concurrent Constraint Programming**

2. **Higher Level Language: Molecular Interaction Maps**

3. **Encoding MIMs in sCCP**
Proteins and complexes are represented as graphs, suitably encoded by predicates of the constraint store.

Complexes are manipulated by predicates acting on their representations in the store.

Contingencies are represented as list of logical rules.

Reactions and interactions are associated to different sCCP-agents.

In the encoding, complexes are created at run-time. Hence the simulation is implicit.
**ENCODING — ENTITIES IN THE STORE**

- **interaction sites** = **ports** *(boolean state)*;
- **molecules** = collection of ports;
- **complexes** = graphs:
  - *vertices* are molecules;
  - *edges* connect two ports;

```python
molecular_type(molecular_type_id, port_list, contingency_list)
node(molecular_type_id, mol_id)
edge([mol_id1, port_id1], [mol_id2, port_id2])
complex_type(complex_id, node_list, edge_list, contingency_list)
complex_number(complex_type_id, Num)
port_number(port_id, Num)
```
ENCODING — ENTITIES IN THE STORE

- interaction sites = ports (boolean state);
- molecules = collection of ports;
- complexes = graphs:
  - vertices are molecules;
  - edges connect two ports;

molecular_type(molecular_type_id, port_list, contingency_list)
node(molecular_type_id, mol_id)
edge([mol_id1, port_id1], [mol_id2, port_id2])
complex_type(complex_id, node_list, edge_list, contingency_list)
complex_number(complex_type_id, Num)
port_number(port_id, Num)
ENCODING — CONTINGENCIES

Contingencies are logical rules:

IF (there are some edges) THEN (inhibit or allow some other ports of edges)

IF (there is y) THEN (inhibit z)
IF (there is y) THEN (allow x)
1. **choose reaction**
   - Interaction agents compete stochastically to determine next reaction
   - Reactions act on port (types)
2. **choose actual complexes involved**
   - Each port type has a port manager agent doing this
3. **build product and apply enabled contingencies**
SIMULATION IN sCCP (IMPLICIT)

1. choose reaction
   - Interaction agents compete stochastically to determine next reaction
   - Reactions act on port (types)
2. choose actual complexes involved
   - Each port type has a port manager agent doing this
3. Build product and apply enabled contingencies
**SIMULATION IN sCCP (IMPLICIT)**

1. **choose reaction**
   - Interaction agents compete stochastically to determine next reaction
   - Reactions act on port (types)

2. **choose actual complexes involved**
   - Each port type has a port manager agent doing this

3. **build product and apply enabled contingencies**
**Simulation in sCCP (Implicit)**

1. **Choose reaction**
   - Interaction agents compete stochastically to determine next reaction
   - Reactions act on port (types)

2. **Choose actual complexes involved**
   - Each port type has a port manager agent doing this

3. **Build product and apply enabled contingencies**
**Simulation in sCCP (Implicit)**

1. **Choose reaction**
   - Interaction agents compete stochastically to determine next reaction
   - Reactions act on port (types)

2. **Choose actual complexes involved**
   - Each port type has a port manager agent doing this

3. **Build product and apply enabled contingencies**
choose reaction

- Interaction agents compete stochastically to determine next reaction
- Reactions act on port (types)

choose actual complexes involved

- Each port type has a port manager agent doing this

build product and apply enabled contingencies
A simple example

Mammalian G1/S cell cycle phase transition
A SIMPLE EXAMPLE
CONCLUSIONS

- sCCP allows an implicit simulation of MIMs
- The key ingredient is the use of the constraint store to represent and manage graph-based representation of complexes.
- The encoding is compositional and linear in the size of MIMs. This is possible only due to the implicit encoding: expliciting reactions causes an exponential increase the description.
- The stochastic simulation is a natural consequence of the semantics of sCCP.
- Future work: a more efficient implementation and an automatic compiler from MIMs.