“Hybridizing” stochastic Concurrent Constraint Programming

Luca Bortolussi\textsuperscript{1}  Alberto Policriti\textsuperscript{2,3}

\textsuperscript{1} Dipartimento di Matematica ed Informatica 
Universit\`a degli studi di Trieste

\textsuperscript{2} Dipartimento di Matematica ed Informatica 
Universit\`a degli studi di Udine

\textsuperscript{3} Istituto di Genomica Applicata (IGA)

Edinburgh, June 18th, 2007
OUTLINE

1 stochastic Concurrent Constraint Programming for biological modeling;
2 Translation from sCCP to ODE;
3 behavioral equivalence issues;
4 being more “discrete” in the translation: sCCP to Hybrid Automata.
THE GENERAL PICTURE

SPA formalism
sCCP

CTMC

Hybrid Automata

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

Constraints can be added to a "container", 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 Stochastic CCP**

Program = D.A

\[ D = \varepsilon \mid D.D \mid p(\overrightarrow{x}) : -A \]

\[ A = 0 \mid \text{tell}_\infty(c).A \mid M \mid \exists x.A \mid A \parallel A \]

\[ M = \pi.G \mid M + M \]

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

\[ G = 0 \mid \text{tell}_\infty(c).G \mid p(\overrightarrow{y}) \mid M \mid \exists x.G \mid G \parallel G \]


**Stochastic Rates**

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

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

*Rates* can be thought as *speed* or *duration* of communications.
There are two transition relations, one instantaneous (finite and confluent) and one stochastic.

Traces are sequences of events with variable time delays among them.

The operational semantics is abstract w.r.t. the notion of time: we can map the labeled transition system into a discrete or a continuous time Markov Chain.

We have an interpreter written in Prolog, using the CLP engine of SICStus to manage the constraint store.

Quantities varying over time can be represented in sCCP as unbounded lists.

Hereafter: special meaning of $X = X + 1$. 
Biochemical Arrows to sCCP processes

\[ R_1 + \ldots + R_n \rightarrow_k P_1 + \ldots + P_m \]

\[ R_1 + \ldots + R_n \rightleftharpoons_{k_2}^{k_1} P_1 + \ldots + P_m \]

\[ S \rightarrow_{E, V_0}^k P \]

\[ S \rightarrow_{E, V_0, h}^k P \]

where \( r_{MA}(k, X_1, \ldots, X_n) = k \cdot X_1 \cdots X_n \); \( r_{MM}(K, V_0, S) = \frac{V_0 S}{S + K} \); \( r_{Hill}(k, V_0, h, S) = \frac{V_0 S^h}{S^h + K^h} \).
**Enzymatic Reaction**

\[ S + E \rightleftharpoons_{k_1}^{k_{-1}} ES \rightarrow_{k_2} P + E \]

**Mass Action Kinetics**

\[
\text{enz}_\text{reaction}(k_1, k_{-1}, k_2, S, E, ES, P) : - \\
\text{reaction}(k_1, [S, E], [ES]) || \\
\text{reaction}(k_{-1}, [ES], [E, S]) || \\
\text{reaction}(k_2, [ES], [E, P])
\]

**Mass Action Equations**

\[
\begin{align*}
\frac{d[ES]}{dt} &= k_1[S][E] - k_2[ES] - k_{-1}[ES] \\
\frac{d[E]}{dt} &= -k_1[S][E] + k_2[ES] + k_{-1}[ES] \\
\frac{d[S]}{dt} &= -k_1[S][E] \\
\frac{d[P]}{dt} &= k_2[ES]
\end{align*}
\]

**Michaelis-Menten Equations**

\[
\begin{align*}
\frac{d[P]}{dt} &= \frac{V_0 S}{S + K} \\
V_0 &= k_2[E_0] \\
K &= \frac{k_2 + k_{-1}}{k_1}
\end{align*}
\]

**Michaelis-Menten Kinetics**

\[
\text{mm}_\text{reaction} \left( \frac{k_2 + k_{-1}}{k_1}, k_2 \cdot E, S, P \right)
\]
ENZYMATIC REACTION

\[ S + E \rightleftharpoons k_1 ES \rightarrow k_2 P + E \]

**Mass Action Kinetics**

\[
\text{enz}\_\text{reaction}(k_1, k_{-1}, k_2, S, E, ES, P) :- \\
\text{reaction}(k_1, [S, E], [ES]) \parallel \\
\text{reaction}(k_{-1}, [ES], [E, S]) \parallel \\
\text{reaction}(k_2, [ES], [E, P])
\]

**Michaelis-Menten Kinetics**

\[
\text{mm}\_\text{reaction}\left(\frac{k_2 + k_{-1}}{k_1}, k_2 \cdot E, S, P\right)
\]
We defined a translation procedure that maps (a restricted version of) sCCP into PRISM, a probabilistic symbolic model checker. Hence, we can check if sCCP programs satisfy properties specified as Continuous Stochastic Logic formulae.
**Connecting SPA and ODE models**

- **SPA formalism**
  - sCCP
- **Two semantics**
- **CTMC**
- **ODE**

Relation between dynamics
**From sCCP to ODE**

**What?**

We want to associate a set of ODE to an sCCP program (written with a restricted syntax).

**Why?**

ODE can be numerically simulated faster than stochastic processes.

**On the market...**

There are (syntactic) methods to write set of ODEs for PEPA and stochastic $\pi$-calculus, looking at the speed of creation and destruction of terms (We did the same for sCCP).

However, the ODE can show a behavior different from that of SPA models.

---


L. Cardelli, From Processes to ODEs by Chemistry, 2006.

FROM sCCP to ODE: example

**Idea**
Collapse all instantaneous transitions following a stochastic one and add their updates to the edge’s label denoting such a transition.

**Reduced Transitions Systems**
- Associate a labeled graph to each sequential component of an sCCP program:
  - **Edges** are transitions and are labeled by a set of guards, a set of updates of variables of the store, and the corresponding rates;
  - **Nodes** are stochastic choices.
- Procedure calls are resolved by inserting a copy of the called procedure.
- Syntactic restrictions are necessary.
**Example**

A :- ask\(\lambda_1(\text{true})\).tell\(_\infty\)(X = X + 1).B
B :- tell\(\lambda_2(X = X - 1)\).A

**The RTS**

\[ [\ast; X = X + 1; \lambda_1] \quad [\ast; X = X - 1; \lambda_2] \]
**From SCCP to ODE: example**

**Interaction Matrix and Reaction Vector**

\[
I = \begin{pmatrix}
X & & \\
& t_1 & t_2 \\
A & 1 & -1 \\
B & -1 & 1
\end{pmatrix}
\]

\[
r = \begin{pmatrix}
\lambda_1 \cdot A \\
\lambda_2 \cdot B
\end{pmatrix}
\]

\[ode = I \cdot r\]

\[
\begin{align*}
\dot{X} &= \lambda_1 \cdot A - \lambda_2 \cdot B \\
\dot{A} &= -\lambda_1 \cdot A + \lambda_2 \cdot B \\
\dot{B} &= \lambda_1 \cdot A - \lambda_2 \cdot B
\end{align*}
\]
THE STRANGE BEAST OF REPRESSILATOR

MODELING 3 NEGATIVE GENE GATES

Neg(X, R) :- tell_{kp}(X = X + 1).Neg(X, R) + ask_{kb}(R \geq 1).ask_{ku}(true).Neg(X, R)

Degrade(X) :- ask_{kd}(X > 0).tell_{\infty}(X = X - 1).Degrade(X)

Neg(A, C) \parallel Neg(B, A) \parallel Neg(C, B) \parallel Degrade(A) \parallel Degrade(B) \parallel Degrade(C)
ODE for sCCP-Repressilator

\[
\begin{align*}
\dot{A} &= k_p Y_A - k_d A \\
\dot{B} &= k_p Y_B - k_d B \\
\dot{C} &= k_p Y_C - k_d C \\
\dot{Y}_1 &= k_u Z_A - k_b Y_A C \\
\dot{Y}_2 &= k_u Z_B - k_b Y_B A \\
\dot{Y}_3 &= k_u Z_C - k_b Y_C B \\
\dot{Z}_1 &= k_b Y_A C - k_u Z_A \\
\dot{Z}_2 &= k_b Y_B A - k_u Z_B \\
\dot{Z}_3 &= k_b Y_C B - k_u Z_C
\end{align*}
\]
THE STRANGE BEAST OF REPRESSILATOR

Repressilator with gene gates

Repressilator: ODE from sCCP

Repressilator in sCCP

Repressilator: average of sCCP model
Intuitively, a hybrid automaton is a *finite state automaton* $H$ with continuous variables $Z$ such that:

- $H$ evolves from $Z$ to $Z'$ in time $T$ according to *differential equations* $\dot{Z} = f(Z)$
- $Z$ must always satisfy *invariant conditions* in each state
- $H$ can cross an edge $e$ only when *guards* on $Z$ are true (activation conditions)
- when $H$ crosses $e$, some variables may be reset.
FROM sCCP TO HYBRID AUTOMATA

WHAT?
We want to associate an hybrid automaton to a sCCP network.

WHY?
The mixed discrete/continuous dynamics of HA is more natural, as it can preserve the logical structure of sCCP models.
Hybrid automata are equipped with well developed analysis methods.

HOW?
The separation between constraint store and logical description of agents makes easy to identify (discrete) modes of the automata
Activation conditions need to look at the temporal semantics of stochastic actions.

**Example: A “Distilled” Repressilator**

A :-
- `tell_{k_+}(X = X + 1).A`
- `ask_{k_-}(X > 0).`
- `tell_{\infty}(X = X - 1).A`
- `ask_{k_0}(true).B`

B :-
- `tell_{k_-}(X = X + 1).B`
- `ask_{k_+}(X > 0).`
- `tell_{\infty}(X = X - 1).B`
- `ask_{k_0}(true).A`
HA ASSOCIATED TO AN sCCP-NETWORK

**Ideas: States and Flows**

- States of the HA correspond to products of states of the RTS: \( \Sigma = (\sigma_1, \ldots, \sigma_n) \).
- Flows for system variables are obtained localizing the construction of ODEs to looping edges.
HA ASSOCIATED TO AN sCCP-NETWORK

IDEAS: **HA-edges**

- HA-edges change the state of just one RTS.
- Associate variables $Y_{ij}$ to edges for activation conditions.
- The flow for $Y_{ij}$ is given by the non constant rate: $\dot{Y}_{ij} = \lambda_{ij}(X)$
- Activation conditions take into account the guard of the RTS-edge plus $Y_{ij} \geq 1$.
- Resets correspond to updates of the RTS-edge.
**Why Y ≥ 1**

- Non-homogeneous Poisson process, with rate $\lambda = \lambda(t)$.
- Cumulative rate: $\Lambda(t) = \int_{t_0}^{t} \lambda(s)ds$
- Average number of firings at time $t$: $\Lambda(t)$.
- At least one firing on average: $\Lambda(t) \geq 1$.
- $\dot{\Lambda}(t) = \lambda(X(t))$
HA ASSOCIATED TO AN sCCP-NETWRK

\[ N = A_1 \parallel \ldots \parallel A_M \] be an sCCP-network.

**Definition (sketch)**

1. control modes \( \Sigma = (\sigma_1, \ldots, \sigma_M) \);
2. control edges corresponding to non-looping arcs \( t_{ij} \in T_i \) of \( RTS(A_i) \);
3. variables: stream variables \( X_1, \ldots, X_k \) of \( N \), plus one variable \( Y_{i,j} \) for each RTS-edge \( t_{ij} \);
4. flow conditions \( ode_{\Sigma} = \sum_{i=1}^{M} ode_{i,\sigma_i} \), where \( ode_{i,\sigma_i} = l_{i,\sigma_i} \cdot r_{i,\sigma_i} \).
   Moreover, if the label of \( t_{ij} \) is \((g_{ij}, c_{ij}, \lambda_{ij})\), \( \dot{Y}_{ij} = \lambda_{ij}(X_1, \ldots, X_k) \);
5. activation condition corresponding to \( t_{ij} \), is the predicate \( g_{ij} \land Y_{ij} \geq 1 \), where \( g_{ij} \) is the guard predicate of the transition;
6. resets corresponding to \( t_{ij} \), with \( c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij} \),
\[
\left( \bigwedge_{k=1}^{h_{ij}} X_{i_k}' = X_{i_k} + \delta_{ij} \right) \land \left( \bigwedge_{t_{ij} \in T_i} Y_{ij}' = 0 \right).\
\]
\[ \frac{\dot{X}}{k_+ - k_-} \quad \dot{Y}_1 = k_0 \quad \dot{Y}_2 = k_0 \]

\[ \dot{Y}_1 = 1 \quad \dot{Y}_2 = 0 \]

\[ \dot{X} = k_- - k_+ \quad \dot{Y}_1 = k_0 \quad \dot{Y}_2 = k_0 \]

\[ \dot{Y}_1' = 0 \quad \dot{Y}_1' = 0 \]

\[ \dot{Y}_2' = 0 \quad \dot{Y}_2' = 0 \]
HYBRID REPRESSILATOR

Repressilator with gene gates

Repressilator in sCCP

Repressilator: ODE from sCCP

Hybrid Repressilator from sCCP
Determinism vs non-determinism

Stability

Oscillations for different parameter values: $K_s$ and $K_p$, $K_s = 1.0$, $K_p = 0.1$

Oscillations for different parameter values: $K_s$ and $K_p$, $K_s = 1.0$, $K_p = 1.0$

Oscillations for different parameter values: $K_s$ and $K_p$, $K_s = 1.0$, $K_p = 15.0$
**Conclusions**

- sCCP for: biochemical reactions, genetic networks, etc.
- sCCP to ODE: problems (the stochastic component *averaged away*).
- Localize the above technique and keep a discrete portion from the network: Hybrid Automata (with the *right* control variables).

**Next steps**

- Relax the definition of sCCP-networks for which the technique works.
- Define a lattice of HAs.
- Formalize the behavioral properties to guide/determine the level of discreteness to maintain.
4th International School on Biology, Computation, and Information (BCI 2007).

2-7 July, Area Science Park, Trieste.
http://bci2007.cbm.fvg.it
Operational Semantic: Instantaneous Transition

**Theorem**

The instantaneous transition $\rightarrow$ is confluent and can be applied only a finite number of times to each configuration $C$. 

\[\begin{align*}
(IR1) & \quad \langle \text{tell}_\infty (c), d, V \rangle \rightarrow \langle 0, d \sqcup c, V \rangle \\
(IR2) & \quad \langle p(\overrightarrow{x}), d, V \rangle \rightarrow \langle A[\overrightarrow{x} / \overrightarrow{y}], d, V \rangle \quad \text{if} \ p(\overrightarrow{y}) : \neg A \\
(IR3) & \quad \langle \exists_x A, d, V \rangle \rightarrow \langle A[y/x], d, V \cup \{y\} \rangle \quad \text{with} \ y \in \mathcal{V}_2 \setminus V \\
(IR4) & \quad \frac{\langle A_1, d, V \rangle \rightarrow \langle A'_1, d', V' \rangle}{\langle A_1 \cdot A_2, d, V \rangle \rightarrow \langle A'_1 \cdot A_2, d', V' \rangle} \\
(IR5) & \quad \frac{\langle A_1, d, V \rangle \rightarrow \langle A'_1, d', V' \rangle}{\langle A_1 \parallel A_2, d, V \rangle \rightarrow \langle A'_1 \parallel A_2, d', V' \rangle}
\end{align*}\]
## Operational Semantic: Stochastic Transition

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(SR1)</td>
<td>$\langle \text{tell}_\lambda(c), d, V \rangle \xrightarrow{(1, \lambda(d))} \langle 0, d \uplus c, V \rangle$ if $d \uplus c \neq \text{false}$</td>
</tr>
<tr>
<td>(SR2)</td>
<td>$\langle \text{ask}_\lambda(c), d, V \rangle \xrightarrow{(1, \lambda(d))} \langle 0, d, V \rangle$ if $d \vdash c$</td>
</tr>
</tbody>
</table>
| (SR3) | $\langle \pi, d, V \rangle \xrightarrow{(p, \lambda)} \langle 0, d', V \rangle$  
  $\langle \pi \cdot A, d, V \rangle \xrightarrow{(p, \lambda)} \langle A, d', V \rangle$  
  with $\pi = \text{ask}$ or $\pi = \text{tell}$ |
| (SR4) | $\langle A_1, d, V \rangle \xrightarrow{(p, \lambda)} \langle A'_1, d', V' \rangle$  
  $\langle A_1 \cdot A_2, d, V \rangle \xrightarrow{(p, \lambda)} \langle A'_1 \cdot A_2, d', V' \rangle$ |
| (SR5) | $\langle M_1, d, V \rangle \xrightarrow{(p, \lambda)} \langle A'_1, d', V' \rangle$  
  $\langle M_1 + M_2, d, V \rangle \xrightarrow{(p', \lambda')} \langle A'_1, d', V' \rangle$  
  with $p' = \frac{p \lambda}{\lambda + \text{rate}(\langle M_2, d, V \rangle)}$ and $\lambda' = \lambda + \text{rate}(\langle M_2, d, V \rangle)$ |
| (SR6) | $\langle A_1, d, V \rangle \xrightarrow{(p, \lambda)} \langle A'_1, d', V' \rangle$  
  $\langle A_1 \parallel A_2, d, V \rangle \xrightarrow{(p', \lambda')} \langle A'_1 \parallel A_2, d', V' \rangle$  
  with $p' = \frac{p \lambda}{\lambda + \text{rate}(\langle A_2, d, V \rangle)}$ and $\lambda' = \lambda + \text{rate}(\langle A_2, d, V \rangle)$ |

Rate returns the sum of rates of all active agents.
Let $\langle A, d, V \rangle \in \mathcal{C}$ be the current configuration. Then the next stochastic transition executes one of the agents prefixed by a guard belonging to the set $\text{exec}(\langle A, d, V \rangle)$, call it $\overline{A}$. Moreover, the probability of the transition (i.e. the first label in $\xrightarrow{}$) is

$$\frac{\text{rate}(\langle \overline{A}, d, V \rangle)}{\text{rate}(\text{exec}(\langle A, d, V \rangle))}$$

and the rate associated to the transition (the second label in $\xrightarrow{}$) is

$$\text{rate}(\text{exec}(\langle A, d, V \rangle))$$
Rates can be interpreted as priorities or as frequencies.

**Rates as Priorities**
- A rate can represent the *priority of execution* of a process.
- There is a *global scheduler* choosing probabilistically between active processes, according to their priority.
- *Discrete time evolution.*

**Rates as Frequencies**
- A rate can represent the *frequency* or *speed* of a process.
- The higher the speed, the higher the probability of seeing a certain process executed.
- *Continuous time evolution.*
**Discrete and Continuous Time**

**Discrete time**

Discrete time transition can be recovered from stochastic transition $\rightarrow (\rho, \lambda)$ by dropping the second label. Hence we leave only the probability associated to transitions, obtaining a Discrete time Markov Chain.

**Continuous time**

Continuous time transition can be recovered from stochastic transition $\rightarrow (\rho, \lambda)$ by multiplying the two labels. Hence we consider the rate associated to the transition, obtaining a Continuous time Markov Chain.
**Discrete and Continuous Time Observables**

**Discrete time I/O observables**

\[ \mathcal{O}_d(\langle A, d \rangle) = \{(d', p) | p = \text{Prob}(\langle A, d \rangle \rightarrow \langle 0, d' \rangle)\} . \]

**Continuous time I/O observables**

\[ \mathcal{O}_c(\langle A, d \rangle)(t) = \{(d', p) | p = \text{Prob}(\langle A, d \rangle \rightarrow \langle 0, d' \rangle) (t)\} . \]

**Theorem**

\[ \lim_{t \rightarrow \infty} \mathcal{O}_c(\langle A, d \rangle)(t) = \mathcal{O}_d(\langle A, d \rangle). \]
Continuous Stochastic Logic

(state formulae) \( \phi ::= \text{true} | a | \neg \phi | \phi \land \phi | P_{\phi} \psi | S_{\phi} \psi | E_{\phi} \)

(path formulae) \( \psi ::= X\phi | \phi U \phi | \phi U^I \phi, \)

- **X**: “next”.
- **U**: “until”.
- **U^I**: “bounded until”, second formula must be true within interval \( I \).
- **P_\phi [\psi]**: path formula \( \psi \) must have probability \( \circ \circ \), with \( \circ \in \{<, \leq, \geq, >\} \).
- **S_\phi [\phi]**: states satisfying must have steady state probability \( \circ \circ \), with \( \circ \in \{<, \leq, \geq, >\} \).
- **E_\circ [\phi]**: expected reward of reaching a state satisfying \( \phi \) is \( \circ \circ \), with \( \circ \in \{<, <, >, >\} \).
PRISM: A PROBABILISTIC MODEL CHECKER

PROBABILISTIC MODEL CHECKING

To model check a formula in CSL, we need essentially to solve linear systems in the size of the state space. Hence model checking is polynomial (but no more linear).

PRISM

PRISM is a symbolic model checking for CSL (and other probabilistic logics), using both Algebraic Decision Diagrams and sparse matrices to speed up the basic algorithms. Models can be described using a concurrent language having variable update and comparison as basic primitives.

MODEL CHECKING sCCP

To model check (restricted) sCCP, we defined an encoding of sCCP programs into PRISM programs.