# Stochastic Concurrent Constraint Programming

Luca Bortolussi[1]

[1]Department of Mathematics and Computer Science
University of Udine, Italy.

PRIN meeting, Padova, 10$^{\mathrm{nd}}$ April 2006

# Outline

# Outline

1. Introduction
   - Concurrent Constraint Programming
   - Continuous Time Markov Chains
2. Syntax and Operational Semantic
   - Syntax and Rates
   - Operational Semantic
3. Examples
   - Random Walk
   - Modeling Biochemical Reactions
   - Modeling Gene Regulatory Networks

# Concurrent Constraint Programming

## Constraint Store

- 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.

## Syntax of CCP

$$Program = Decl.A$$

$$D = \varepsilon \mid Decl.Decl \mid p(x) : -A$$

$$\begin{aligned} A = \; & \mathbf{0} \\ \mid \; & \mathrm{tell}(c).A \\ \mid \; & \mathrm{ask}(c_1).A_1 + \mathrm{ask}(c_2).A_2 \\ \mid \; & A_1 \parallel A_2 \mid \exists_x A \mid p(x) \end{aligned}$$

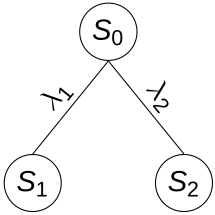
## Agents

Agents can perform two basic operations on this store:

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

# Continuous Time Markov Chains

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* 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, with rate given by the *sum of outgoing transitions* from the actual node ($\lambda_1 + \lambda_2$).

# Outline

# Syntax of sCCP

## Syntax of Stochastic CCP

$$Program = Decl.A$$

$$D = \varepsilon \mid Decl.Decl \mid p(\mathbf{x}) : -A$$

$$A = \mathbf{0} \mid \mathrm{tell}_\lambda(c).A \mid \mathrm{ask}_\lambda(c).A \mid [p(\mathbf{x})]_\lambda \mid \exists_x A \mid (A_1 + A_2) \mid (A_1 \parallel A_2)$$

Each basic instruction (tell, ask, procedure call) has a rate attached to it. *Rates are functions from the constraint store* $\mathcal{C}$ *to positive reals*: $\lambda : \mathcal{C} \longrightarrow \mathbb{R}^+$.

# Operational Semantic

## SOS

$$\langle \text{tell}_\lambda(c).A, d, V\rangle \longrightarrow_{(1,\lambda(d))} \langle A, d \sqcup c, V\rangle$$

$$\langle \text{ask}_\lambda(c), d, V\rangle \longrightarrow_{(1,\lambda(d))} \langle A, d, V\rangle \qquad \text{if } d \vdash c$$

$$\langle [p(\mathbf{y})]_\lambda, d, V\rangle \longrightarrow_{(1,\lambda(d))} \langle A[\mathbf{y}/\mathbf{x}], d, V\rangle \qquad \text{if } p(\mathbf{x}) : -A$$

$$\frac{\langle A_1, d, V\rangle \longrightarrow_{(p,\eta)} \langle A_1', d', V\rangle}{\langle A_1 + A_2, d, V\rangle \longrightarrow_{(p',\eta')} \langle A_1', d', V\rangle}$$

$$\frac{\langle A_1, d, V\rangle \longrightarrow_{(p,\eta)} \langle A_1', d', V\rangle}{\langle A_1 \parallel A_2, d, V\rangle \longrightarrow_{(p',\eta')} \langle A_1' \parallel A_2, d', V\rangle}$$

$$\text{with } p' = \frac{p\eta}{\eta + \text{rate}(A_2, d)} \text{ and } \eta' = \eta + \text{rate}(A_2, d)$$

rate returns the *sum of rates of all active agents, evaluated w.r.t. the current configuration of the store*.

# Examples

## Example

$$\langle \mathrm{tell}_1(c), \top, \emptyset \rangle \longrightarrow_{(1,1)} \langle \mathbf{0}, c, \emptyset \rangle$$

$$\begin{aligned} &\langle \mathrm{ask}_1(c).\mathrm{tell}_{100}(d) \parallel \mathrm{tell}_1(c), \top, \emptyset \rangle \\ &\longrightarrow_{(1,1)} \langle \mathrm{ask}_1(c).\mathrm{tell}_{100}(d), c, \emptyset \rangle \\ &\longrightarrow_{(1,1)} \langle \mathrm{tell}_{100}(d), c, \emptyset \rangle \\ &\longrightarrow_{(1,100)} \langle \mathbf{0}, c \sqcup d, \emptyset \rangle. \end{aligned}$$

## Example

$$\langle tell_1(c) + tell_1(d), \top, \emptyset \rangle \longrightarrow_{(0.5,2)} \langle 0, c, \emptyset \rangle$$
$$\langle tell_1(c) + tell_1(d), \top, \emptyset \rangle \longrightarrow_{(0.5,2)} \langle 0, d, \emptyset \rangle$$

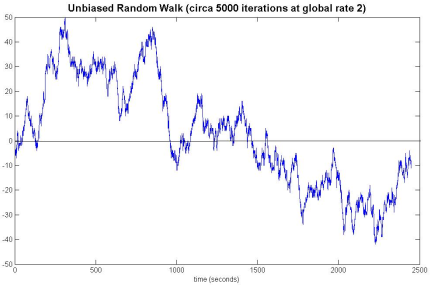
# Outline

# Random Walk

The language has been implemented by writing a meta-interpreter in SICStus Prolog.

We can model random walk as a stochastic process increasing or diminishing of one unit the value of a variable $X$.

random($X$) :-
$\exists_Y$ ( tell$_1$($Y = X + 1$)
+ tell$_1$($Y = X - 1$)
).random($Y$)



# CCP for System Biology

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

## $\pi$-calculus for system biology

| | |
|---|---|
| Molecule | Process |
| Interaction capability | Channel |
| Interaction | Communication |
| Modification<br>(of cellular components) | State change<br>(state transition systems) |

# CCP for System Biology

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

## CCP for Biological Simulation

| | | |
|---|---|---|
| Molecule (Type) / Reaction | $\longleftrightarrow$ | Process |
| Modification | $\longleftrightarrow$ | State change |
| (of cellular components) | | (state transition systems and memory) |
| Environment | $\longleftrightarrow$ | Constraint Store |
| Interaction with Environment | $\longleftrightarrow$ | Asynchronous Communication |
| *Direct Interaction capability* | $\longleftrightarrow$ | *Channel* |
| *Interaction* | $\longleftrightarrow$ | *Synchronous Communication* |

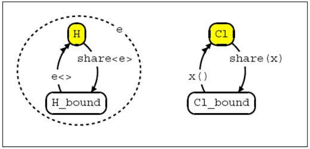
We need to extend the concept of rate: a rate needs to be a function $\lambda : \mathcal{C} \longrightarrow \mathbb{R}^+$.

# A simple reaction: H + Cl $\leftrightharpoons$ HCl

## $\pi$-calculus

We model atom H and atom Cl. Reaction happens by a synchronous communication of these two processes. We need several copy of these processes.

Covalent Bonding: $H + Cl \rightleftharpoons HCl$




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

A. Phillips - Feb. 2005

# A simple reaction: H + Cl $\leftrightharpoons$ HCl

**sCCP**

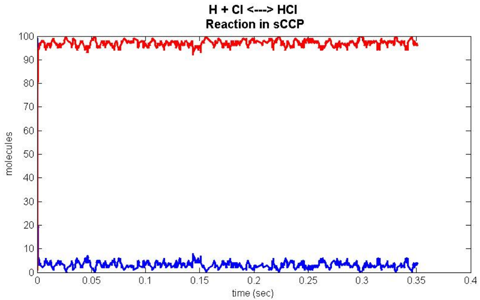
We write a reaction agent, while the reagents and the product are modeled in the constraint store (put down in the environment). Independent on the number of agents.

$$
\begin{aligned}
&\text{reaction}(H, CL, HCL) \text{ :-}\\
&\qquad (\ \text{tell}_{\text{shareRate}(H,Cl)}(\text{share}(H, CL, HCL)) +\\
&\qquad\ \ \ \text{tell}_{\text{relRate}(H,Cl)}(\text{rel}(H, CL, HCL))\\
&\qquad ).\text{reaction}(H, CL, HCL)
\end{aligned}
$$

# A simple reaction: H + Cl ⇋ HCl

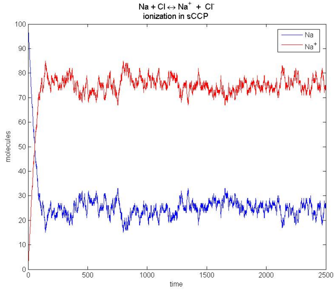
## sCCP

We write a reaction agent, while the reagents and the product are modeled in the constraint store (put down in the environment). Independent on the number of agents.

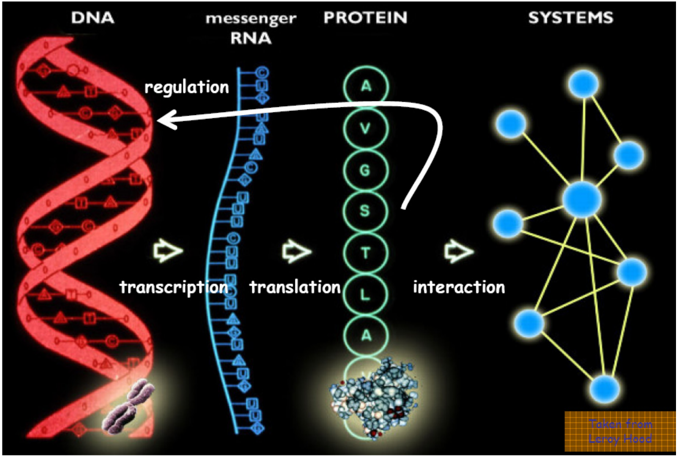


# Another reaction: Na + Cl $\leftrightharpoons$ $Na^{+}$ + $Cl^{-}$

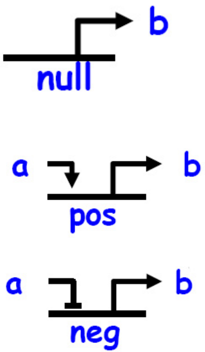
$$
\begin{aligned}
&\text{ionization}(Na, Cl, Na^{+}, Cl^{-}) \text{ :-}\\
&\qquad (\ \text{tell}_{\text{ionizeRate}(Na^{+}, Cl^{-})}(\text{ionize}(Na, Cl, Na^{+}, Cl^{-})) +\\
&\qquad\ \ \ \text{tell}_{\text{deionizeRate}(Na+, Cl-)}(\text{deionize}(Na, Cl, Na^{+}, Cl^{-}))\\
&\qquad ).\text{ionization}(Na, Cl, Na^{+}, Cl^{-})
\end{aligned}
$$



# The gene machine



# The instruction set




degradator(X) :- $\text{tell}_{\text{degRate}(X)}$(degrade($X$)).degradator($X$)

null($X$) :- $\text{tell}_{\text{prodRate}(X)}$(produce($X$)).null($X$)

pos($X$, $Y$) :- ( $\text{tell}_{\text{prodRate}(X)}$(produce($X$))
+ $\text{ask}_{\text{enhanceRate}(Y)}$($Y > 0$).$\text{tell}_{\text{enhProdRate}(X)}$(produce($X$))
).pos($X$, $Y$)

neg($X$, $Y$) :- ( $\text{tell}_{\text{prodRate}(X)}$(produce($X$))
+ $\text{ask}_{\text{inibitRate}(Y)}$($Y > 0$).$\text{ask}_{\text{delayRate}(X)}$(true)
).neg($X$, $Y$)

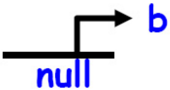
null_gate($X$) :- null($X$) || degradator($X$)

pos_gate($X$, $Y$) :- pos($X$, $Y$) || degradator($X$)

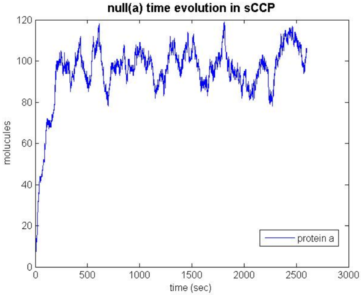
neg_gate($X$, $Y$) :- neg($X$, $Y$) || degradator($X$)

L. Cardelli, A. Phillips, 2005.

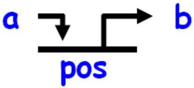
# Null Gate



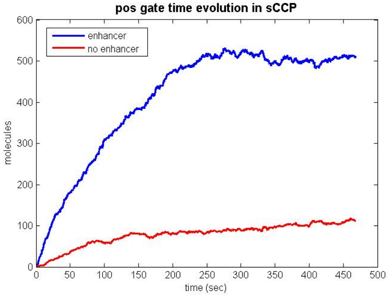

neg_gate($A$)



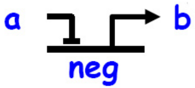
# Pos Gate



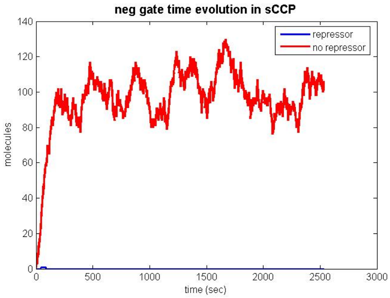

$\text{pos\_gate}(A, B)$



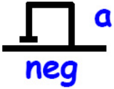
# Neg Gate



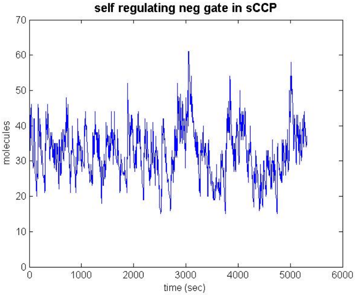

$\text{neg\_gate}(A, B)$



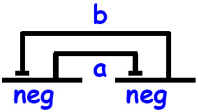
# Self Repression



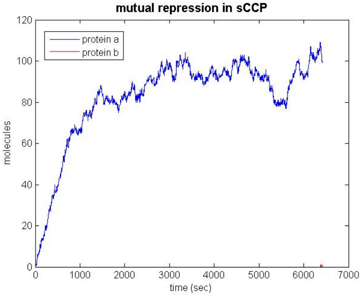

$\text{neg\_gate}(A, A)$



# Mutual Repression



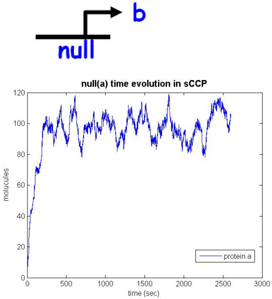

$$\text{neg\_gate}(A, B) \parallel \text{neg\_gate}(B, A)$$



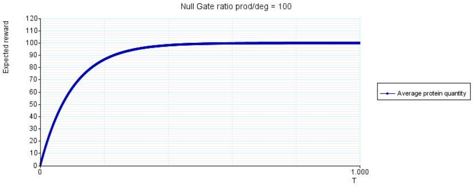
# Towards verification of models

## PRISM

We have defined a mapping from a sublanguage of sCCP (restriction on parallel operators) to the modeling language of PRIMS, a symbolic probabilistic model checker.




Expected value of protein $b$ at time $T$.



# Conclusions

- We have introduced a stochastic version of CCP.
- We showed that sCCP may be used for modeling biological systems, via examples.
- We showed first examples of verifying properties of these systems, using PRISM

# The End

**THANKS FOR THE ATTENTION!**

**QUESTIONS?**