Hybrid Systems and Systems Biology

Alberto Policriti

Dpt. of Mathematics and Informatics, University of Udine.

Applied Genomics Institute

IGA ISTITUTO DI GENOMICA APPLICATA

(joint work with Luca Bortolussi)

June 7th, 2008

Outline

1. Hybrid Systems: definition and sample applications to Systems Biology;
2. a Stochastic Process Algebra (SPA) for biological modeling: *sCCP*;
3. efficiency: from sCCP to ODE's;
4. being more “discrete”: the circadian clock.

Hybrid Systems

Many real systems have a double nature. They:

- evolve in a continuous way,
- are ruled by a discrete system.

Hybrid Systems

Many real systems have a double nature. They:

- evolve in a continuous way,
- are ruled by a discrete system.

Hybrid Systems

Many real systems have a double nature. They:

- evolve in a continuous way,
- are ruled by a discrete system.

Hybrid Systems

Many real systems have a double nature. They:

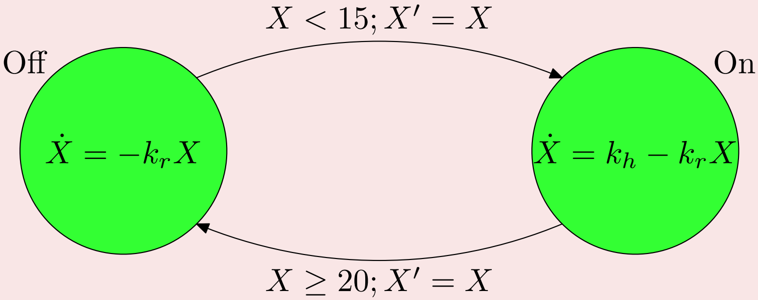
- evolve in a continuous way,
- are ruled by a discrete system.

Modeling?

hybrid systems/automata

The Example

A thermostat model



Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

Hybrid Automata - Syntax

Alur et al. 1992

Definition (Hybrid Automaton - Syntax)

A tuple $H = \langle Z, Z', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ where:

- Z and Z' are varibles in $\mathbb{R}^k$
- $\langle \mathcal{V}, \mathcal{E} \rangle$ is a graph
- Each $v \in \mathcal{V}$ is labelled by $Inv(v)[Z]$ and $Dyn(v)[Z, Z', T]$
- Each $e \in \mathcal{E}$ is labelled by $Act(e)[Z]$ and $Reset(e)[Z, Z']$

$Dyn(v)[Z, Z', T]$ is a formula of the form $Z' = p_v(Z, T)$, where p_v is the solution of the vectorial field $\mathcal{P}(v)$.

Hybrid Automata - Intuitively

Finite Automata *plus Time*

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - Intuitively

Finite Automata *plus Time*

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - Intuitively

Finite Automata *plus Time*

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - Intuitively

Finite Automata *plus Time*

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - Intuitively

Finite Automata *plus Time*

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - Intuitively

Finite Automata *plus Time*

Time flows when *within* states:

- in *mode* v, Z must always satisfy $Inv(v)[Z]$
- H evolves from Z to Z' in time T when $Dyn(v)[Z, Z', T]$
- H can cross e only if $Act(e)[Z]$
- when H crosses e, $Reset(e)[Z, Z']$

Hybrid Automata - States and Transitions

Definition (Hybrid Automaton State)

A state is a pair in $\mathcal{V} \times \mathbb{R}^k$.

Hybrid Automata - States and Transitions

Definition (Hybrid Automaton State)

A state is a pair in $\mathcal{V} \times \mathbb{R}^k$.

Definition (Continuous Transition)

$\langle v, r \rangle \xrightarrow{t}_C \langle v, s \rangle \iff$ $\exists f : \mathbb{R}^+ \mapsto \mathbb{R}^k$ continuous such that $r = f(0)$, $s = f(t)$, and $\forall t' \in [0, t]$ the formulæ $Inv(v)[f(t')]$ and $Dyn(v)[r, f(t'), t']$ hold.

Hybrid Automata - States and Transitions

Definition (Hybrid Automaton State)

A state is a pair in $\mathcal{V} \times \mathbb{R}^k$.

Definition (Continuous Transition)

$$\langle v, r \rangle \xrightarrow{t}_C \langle v, s \rangle \iff$$

$\exists f : \mathbb{R}^+ \mapsto \mathbb{R}^k$ continuous such that $r = f(0)$, $s = f(t)$, and $\forall t' \in [0, t]$ the formulæ $Inv(v)[f(t')]$ and $Dyn(v)[r, f(t'), t']$ hold.

Definition (Discrete Transition)

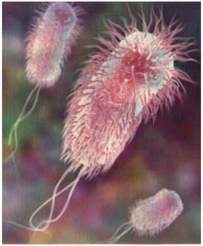
$$\langle v, r \rangle \xrightarrow{\langle v,u \rangle}_D \langle u, s \rangle \iff$$

$\langle v, u \rangle \in \mathcal{E}$ and $Inv(v)[r]$, $Act(\langle v, u \rangle)[r]$, $Reset(\langle v, u \rangle)[r, s]$, and $Inv(u)[s]$ hold.

Examples of use of HS for Systems Biology

Escherichia coli

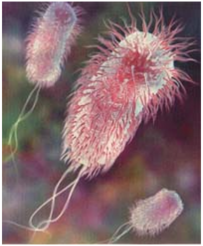
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Examples of use of HS for Systems Biology

Escherichia coli

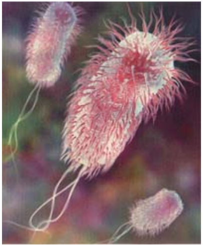
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Examples of use of HS for Systems Biology

Escherichia coli

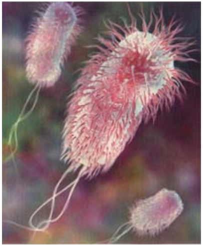
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Examples of use of HS for Systems Biology

Escherichia coli

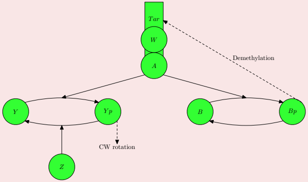
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Depending on the concentration of attractans and repellents, E. coli responds to stimuli in one of two ways:

- "RUNS" – it moves in a straight line by moving its flagella counterclockwise (CCW)
- "TUMBLES" – it randomly changes its heading by moving its flagella clockwise (CW)

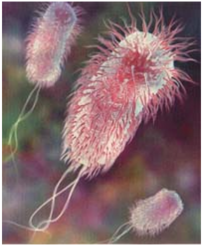
E. Coli Model



Examples of use of HS for Systems Biology

Escherichia coli

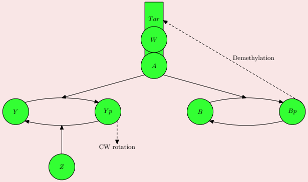
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Depending on the concentration of attractans and repellents, E. coli responds to stimuli in one of two ways:

- "RUNS" – it moves in a straight line by moving its flagella counterclockwise (CCW)
- "TUMBLES" – it randomly changes its heading by moving its flagella clockwise (CW)

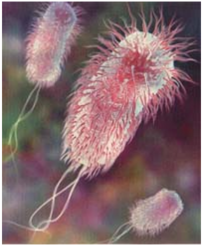
E. Coli Model



Examples of use of HS for Systems Biology

Escherichia coli

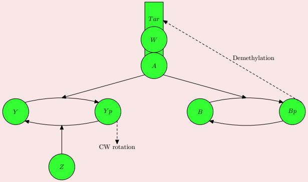
- a bacterium detecting the food concentration through a set of receptors;
- moving by flagellar rotations.



Depending on the concentration of attractans and repellents, E. coli responds to stimuli in one of two ways:

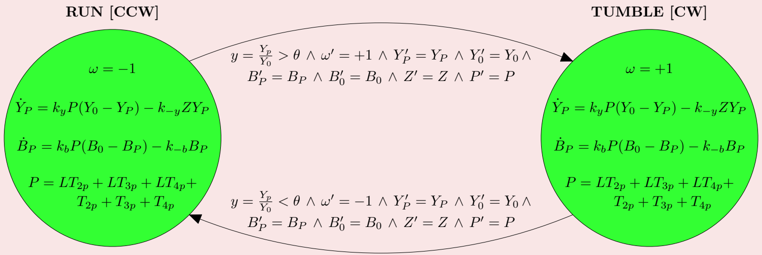
- "RUNS" – it moves in a straight line by moving its flagella counterclockwise (CCW)
- "TUMBLES" – it randomly changes its heading by moving its flagella clockwise (CW)

E. Coli Model



Examples of use of HS for Systems Biology

E. coli IDA Model



A. Casagrande et al., *Independent Dynamics Hybrid Automata in Systems Biology*, AB('05) Tokyo, 2005

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use “well behaving” differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters’space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use “well behaving” differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters’space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use “well behaving” differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters’space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use “well behaving” differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters’space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use "well behaving" differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters'space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

- Use "well behaving" differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters'space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

(Typical) Key Property

Theorem (Multiaffine functions on hyperrectangular polytopes) f multiaffine function and P hyperrectangular polytope:

$$f(P) \quad \subseteq \quad \mathrm{hull}(\{f(v) \mid v \in \mathcal{V}_P\}),$$

that is $\forall x \in P$, $f(x)$ is a linear combination of the values of f at vertices of P.

Examples of use of HS for Systems Biology

Parameters in Genetic Regulatory Networks

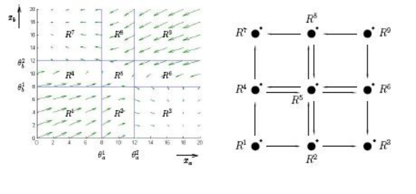
- Use "well behaving" differential equations (e.g. piece-wise multi affine functions);
- use temporal logic to express dynamical properties;
- partition the parameters'space in such a way to guarantee validity of temporal properties.
- PM-Systems *Model Checking Genetic Regulatory Networks with Applications to Synthetic Biology*, G. Batt and C. Belta

(Typical) Key Property

Theorem (Multiaffine functions on hyperrectangular polytopes) f multiaffine function and P hyperrectangular polytope:

$$f(P) \quad \subseteq \quad \text{hull}(\{f(v) \mid v \in \mathcal{V}_P\}),$$

that is $\forall x \in P$, $f(x)$ is a linear combination of the values of f at vertices of P.



Examples of use of HS for Systems Biology

Switching among simulation techniques

Use different simulation techniques as the number of molecule varies;

1. stochastic simulation for low numbers;
2. ode simulation for high numbers;

Alur et al. *Hybrid Modeling and Simulation of Biochemical Networks*

Modeling: Stochastic vs. Diff. Equations

Differential Equations

- mature
- computationally affordable (*one* run)

Stochastic *something*

- precise
- computationally costly (*many* runs!)

A *Bridge*: our attempt

First step

Use a Stochastic version of Concurrent Constraint Programming as Discrete Stochastic starting tool.

A *Bridge*: our attempt

First step

Use a Stochastic version of Concurrent Constraint Programming as Discrete Stochastic starting tool.

Second step

Introduce Hybrid Systems.

Modes of the HS $\Leftrightarrow$ Combinations of Stochastic choices

Dynamics $\Leftrightarrow$ *Ad-hoc* edge's variables with activations constrained by rates

Stochastic Concurrent Constraint Programming

What is

- A SPA with a computational “twist”.
- Maintains a form of local storage.
- Keeps separated the description of interactions and the management of data for computations.
- (Naturally) Introduce functional rates.

Concurrent Constraint Programming

Constraint Store

- 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

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)

V. Saraswat, *Concurrent Constraint Programming*, MIT press, 1993

Syntax of CCP

$Program = Decl.A$

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

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

Syntax of sCCP

Syntax of Stochastic CCP

$$Program = D.A$$

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

$$\begin{aligned} A &= \mathbf{0} \mid \text{tell}_\infty(c).A \mid M \mid \exists_x A \mid A \parallel A \\ M &= \pi.G \mid M + M \\ \pi &= \text{tell}_\lambda(c) \mid \text{ask}_\lambda(c) \\ G &= \mathbf{0} \mid \text{tell}_\infty(c).G \mid p(\vec{y}) \mid M \mid \exists_x G \mid G \parallel G \end{aligned}$$

L. Bortolussi, *Stochastic Concurrent Constraint Programming*, QAPL, 2006

Stochastic Rates

Rates are functions from the constraint store $\mathcal{C}$ to positive reals:

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

Rates can be thought as speed or duration of communications.

sCCP – technical details

Operational Semantics

- There are *two transition relations*, one instantaneous (finite and confluent) and one stochastic.
- Traces are sequences of events with variable time delays among them.

Discrete vs. Continuous Semantics

- 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.

Implementation

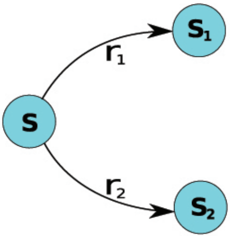
- We have an interpreter written in Prolog, using the *CLP engine of SICStus* to manage the constraint store.
- Efficiency issues.

Stream Variables

- *Quantities varying over time* can be represented in sCCP as unbounded lists.
- Hereafter: special meaning of $X = X + 1$.

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).

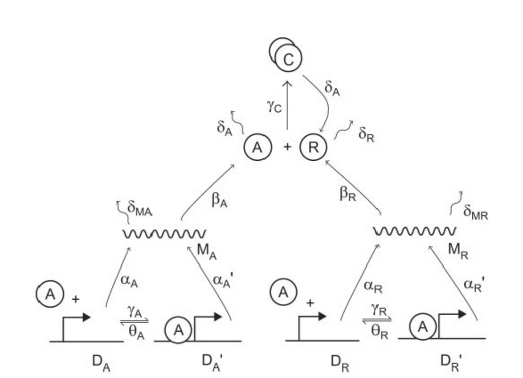


- In each state, we select the next state according to a *probability distribution* obtained normalizing rates (from S to S_1 with prob. $\frac{r_1}{r_1+r_2}$).
- The time spent in a state is given by an exponentially distributed random variable, with rate given by the *sum of outgoing transitions* from the actual node ($r_1 + r_2$).

Circadian Clock

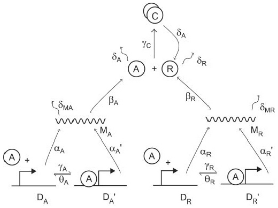
(J. M. G. Vilar, H. Yuan Kueh, N. Barkai, and S. Leibler. PNAS, 2002.)

A clock expressing proteins A and R with a stable period



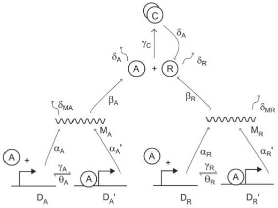
Circadian Clock

- transcription and the translation phases are modeled explicitl.
- *A* is an enhancer for both genes
- *R* represses *A* forming *AR* and making *A* inactive
- *R* can be degraded only if it is not in complexed form while *A* can be degraded in any form



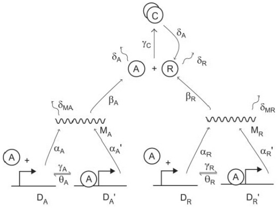
Circadian Clock

- transcription and the translation phases are modeled explicitl.
- A is an enhancer for both genes
- R represses A forming AR and making A inactive
- R can be degraded only if it is not in complexed form while A can be degraded in any form



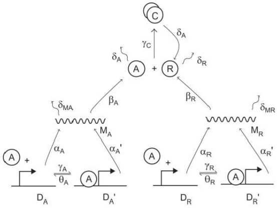
Circadian Clock

- transcription and the translation phases are modeled explicitl.
- *A* is an enhancer for both genes
- *R* represses *A* forming *AR* and making *A* inactive
- *R* can be degraded only if it is not in complexed form while *A* can be degraded in any form



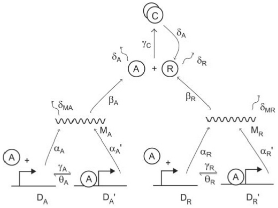
Circadian Clock

- transcription and the translation phases are modeled explicitl.
- A is an enhancer for both genes
- R represses A forming AR and making A inactive
- R can be degraded only if it is not in complexed form while A can be degraded in any form



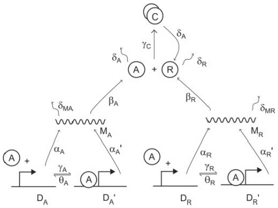
Circadian Clock

- transcription and the translation phases are modeled explicitl.
- A is an enhancer for both genes
- R represses A forming AR and making A inactive
- R can be degraded only if it is not in complexed form while A can be degraded in any form



Circadian Clock

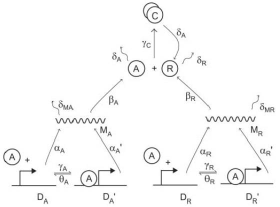
- transcription and the translation phases are modeled explicitl.
- A is an enhancer for both genes
- R represses A forming AR and making A inactive
- R can be degraded only if it is not in complexed form while A can be degraded in any form



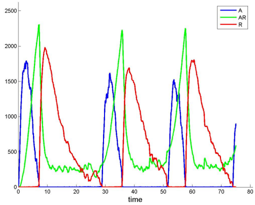
Robustness

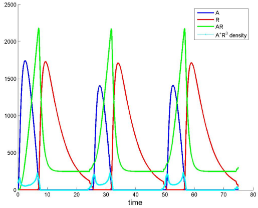
The stochastic model is more robust as if internal noise was exploited by Nature to increase stability of function (i.e., for a clock, oscillations).

Circadian Clock



$\mathsf{p\_gate}(\alpha_A, \alpha'_A, \gamma_A, \theta_A, M_A, A) \parallel$
$\mathsf{p\_gate}(\alpha_R, \alpha'_R, \gamma_R, \theta_R, M_R, A) \parallel$
$\mathsf{reaction}(\beta_A, [M_A], [A]) \parallel$
$\mathsf{reaction}(\delta_{MA}, [M_A], []) \parallel$
$\mathsf{reaction}(\beta_R, [M_R], [R]) \parallel$
$\mathsf{reaction}(\delta_{MR}, [M_R], []) \parallel$
$\mathsf{reaction}(\gamma_C, [A, R], [AR]) \parallel$
$\mathsf{reaction}(\delta_A, [AR], [R]) \parallel$
$\mathsf{reaction}(\delta_A, [A], []) \parallel$
$\mathsf{reaction}(\delta_R, [R], [])$





From sCCP to ODE

What?

We want to associate a set of ODE to an sCCP program (written in 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 π-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.

J. Hillston, Fluid Flow Approximation of PEPA models, *QEST*, 2005.

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

L. Bortolussi, A. Policriti. Connecting Process Algebras and Differential Equations for systems biology, 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.

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

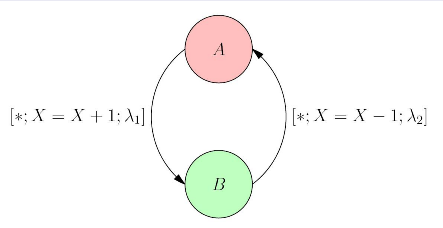
- Associate a labeled graph to each sequential component of an sCCP program:
  - EDGES are transitions and are labeled by a set of guards, a set of updates of variables of the store, and the corresponding rates;
  - NODES are stochastic choices.
- Procedure calls are resolved by inserting a copy of the called procedure.
- Syntactic restrictions are necessaries.

From sCCP to ODE: example

Example

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

The RTS



From sCCP to ODE: example

Interaction matrix andx reaction vector

$$I = \begin{array}{c|cc} & t_1 & t_2 \\ \hline X & 1 & -1 \\ A & -1 & 1 \\ B & 1 & -1 \end{array} \qquad r = \begin{pmatrix} \lambda_1 \cdot A \\ \lambda_2 \cdot B \end{pmatrix}$$

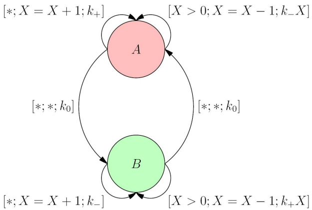
$ode = I \cdot r$

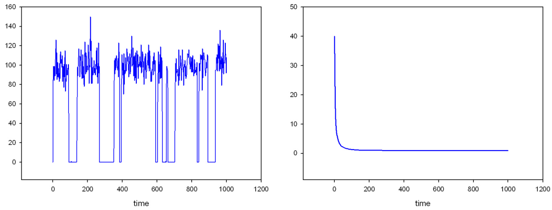
$$ode \begin{cases} \dot{X} = \lambda_1 \cdot A - \lambda_2 \cdot B \\ \dot{A} = -\lambda_1 \cdot A + \lambda_2 \cdot B \\ \dot{B} = \lambda_1 \cdot A - \lambda_2 \cdot B \end{cases}$$

Example: a "distilled" Repressilator

$$\begin{array}{lll}
A :- & & \mathsf{tell}_{k_+}(X = X + 1).A \\
& + & \mathsf{ask}_{k_- X}(X > 0). \\
& & \qquad \mathsf{tell}_{\infty}(X = X - 1).A \\
& + & \mathsf{ask}_{k_0}(\mathit{true}).B
\end{array}$$

$$\begin{array}{lll}
B :- & & \mathsf{tell}_{k_-}(X = X + 1).B \\
& + & \mathsf{ask}_{k_+ X}(X > 0). \\
& & \qquad \mathsf{tell}_{\infty}(X = X - 1).B \\
& + & \mathsf{ask}_{k_0}(\mathit{true}).A
\end{array}$$





HA associated to an sCCP-ntwrk

Ideas

- localize the construction to looping edges in order to determine flow conditions;
- use (non constant) rates to govern variables associated to edges;
- use variables associated to edges in activation conditions.

HA associated to an sCCP-ntwrk

Ideas

- localize the construction to looping edges in order to determine flow conditions;
- use (non constant) rates to govern variables associated to edges;
- use variables associated to edges in activation conditions.

HA associated to an sCCP-ntwrk

Ideas

- localize the construction to looping edges in order to determine flow conditions;
- use (non constant) rates to govern variables associated to edges;
- use variables associated to edges in activation conditions.

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,

$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,
$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,

$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,

$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,

$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

HA associated to an sCCP-ntwrk

$N = A_1 \parallel \ldots \parallel A_M$ 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 $ode_{i,\sigma_i} = I_{i,\sigma_i} \cdot r_{i,\sigma_i}$. 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 $g_{ij} \wedge Y_{ij} \geq 1$, where g_{ij} is the guard predicate of the transition;
6. resets corresponding to t_{ij}, with $c_{ij} = \bigwedge_{k=1}^{h_{ij}} X_{i_k} = X_{i_k} + \delta_{ij}$,

$$\left(\bigwedge_{k=1}^{h_{ij}} X'_{i_k} = X_{i_k} + \delta_{ij} \right) \wedge \left(\bigwedge_{t_{ij} \in T_i} Y'_{ij} = 0 \right).$$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s)ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s) ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s)ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s)ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s) ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

Time varying rates $\lambda = \lambda(t)$

- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

$$\Lambda(t) = \int_{t_0}^{t} \lambda(s)ds,$$

- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

On activation conditions

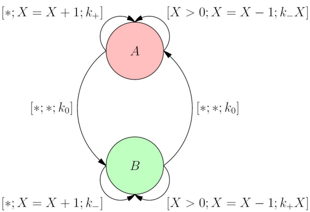
Time varying rates $\lambda = \lambda(t)$

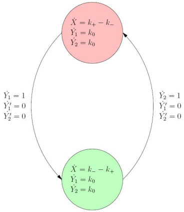
- introduce one variable Y_e for each edge e
- every transition constitute a *non-homogeneous Poisson process*
- we can define the *cumulative rate function*

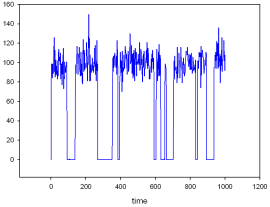
$$\Lambda(t) = \int_{t_0}^{t} \lambda(s) ds,$$

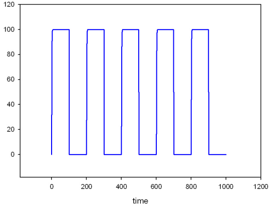
- theory of non-homogeneous Poisson processes $\Rightarrow$ number of firings at time t behaves like a Poisson variable with rate equal to $\Lambda(t)$
- we may activate the transition whenever $\Lambda(t) \geq 1$

$$\dot{Y}_e = \lambda(X_1, \ldots, X_k) \text{ and } Y_e \geq 1$$

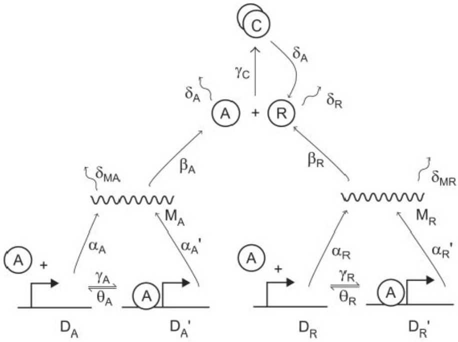

$[*; X = X+1; k_+]$
$[X > 0; X = X-1; k_- X]$
A
$[*; *; k_0]$
$[*; *; k_0]$
B
$[*; X = X+1; k_-]$
$[X > 0; X = X-1; k_+ X]$


$\dot{X} = k_+ - k_-$
$\dot{Y}_1 = k_0$
$\dot{Y}_2 = k_0$
$\dot{Y}_1 = 1$
$\dot{Y}_1' = 0$
$\dot{Y}_2' = 0$
$\dot{Y}_2 = 1$
$\dot{Y}_1' = 0$
$\dot{Y}_2' = 0$
$\dot{X} = k_- - k_+$
$\dot{Y}_1 = k_0$
$\dot{Y}_2 = k_0$


160
140
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
time


120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
time

Circadian Clock: robustness



Circadian Clock: robustness

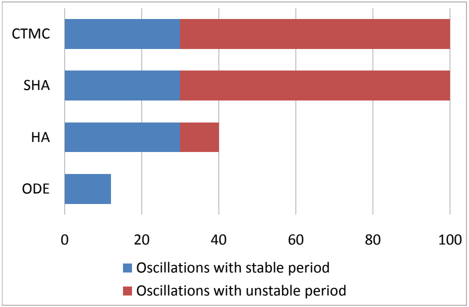


Figure: Stability with respect to β_R

Conclusions

- HS for: biochemical reactions, genetic networks, etc.
- SPA to ODE: problems (the stochastic component *averaged away*).
- Localize ODE's and maintain a discrete portion of the network: Hybrid Systems (with the *right* control variables).

Future

- Define a *lattice* of HSs.
- Formalize the behavioral properties to guide/determine the level of discreteness to maintain.