Artificial Biochemistry

Luca Cardelli

Microsoft Research

CoSBi
Trento 2007-02-21

http://LucaCardelli.name

Stochastic Collectives

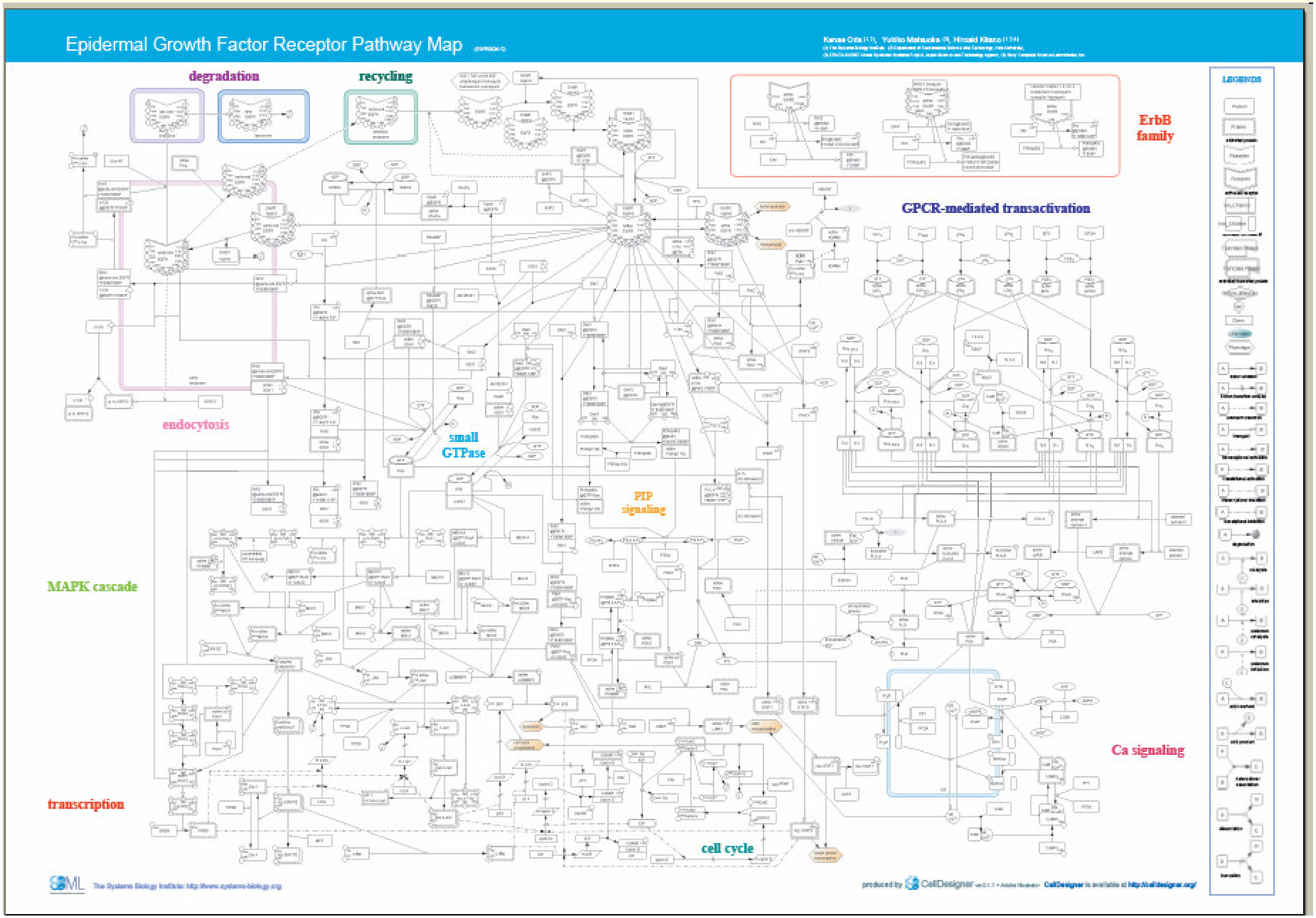
Stochastic Collectives

- "Collective":
  - A large set of interacting finite state automata:
    - Not quite language automata ("large set")
    - Not quite cellular automata ("interacting" but not on a grid)
    - Not quite process algebra ("collective behavior")
    - Cf. multi-agent systems and swarm intelligence

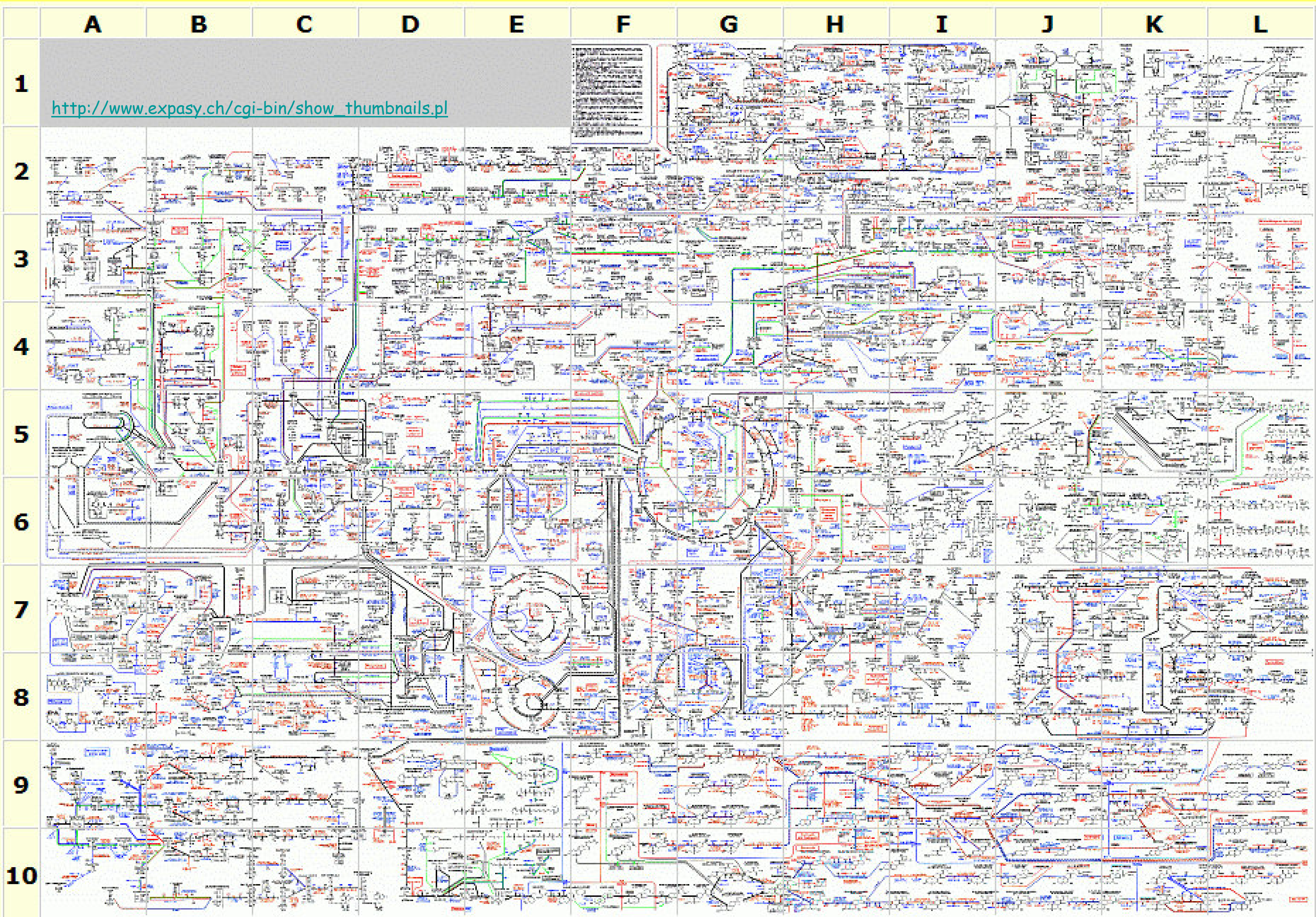
- "Stochastic":
  - Interactions have *rates*
    - Not quite discrete (hundreds or thousands of components)
    - Not quite continuous (non-trivial stochastic effects)
    - Not quite hybrid (no "switching" between regimes)

- Very much like biochemistry
  - Which is a large set of stochastically interacting molecules/proteins
  - Are proteins finite state and subject to automata-like transitions?
    - Let's say they are, at least because:
    - Much of the knowledge being accumulated in Systems Biology is described as state transition diagrams [Kitano].

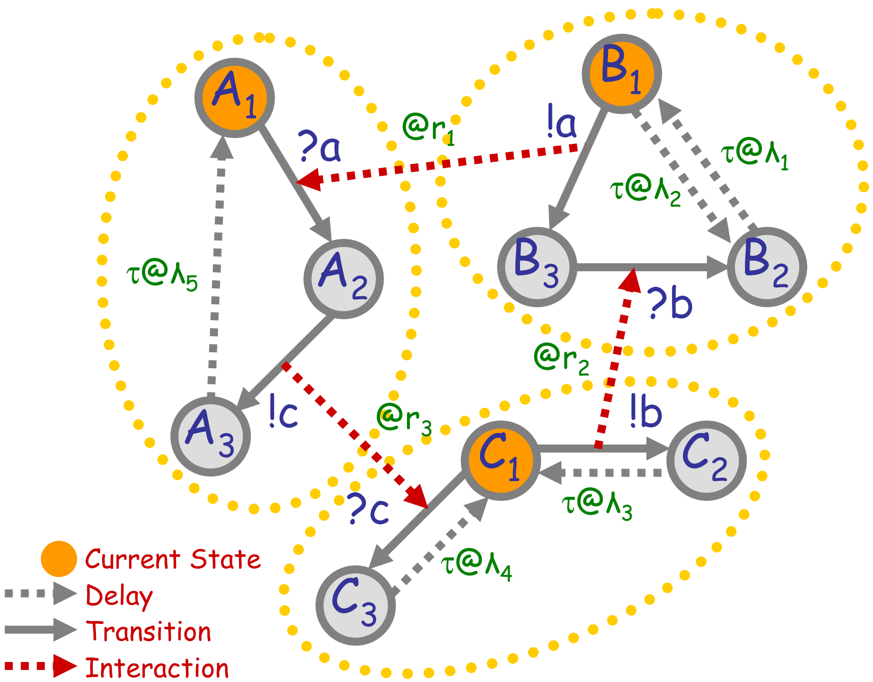
State Transitions



Compositionality (NOT!)



Interacting Automata



new $a@r_1$
new $b@r_2$
new $c@r_3$

Communication channels

$A_1 = ?a;\ A_2$
$A_2 = !c;\ A_3$
$A_3 = \tau@\lambda_5;\ A_1$

$B_1 = \tau@\lambda_2;\ B_2 + !a;\ B_3$
$B_2 = \tau@\lambda_1;\ B_1$
$B_3 = ?b;\ B_2$

$C_1 = !b;\ C_2 + ?c;\ C_3$
$C_2 = \tau@\lambda_3;\ C_1$
$C_3 = \tau@\lambda_4;\ C_2$

Automata

$A_1 \mid B_1 \mid C_1$

The system and initial state

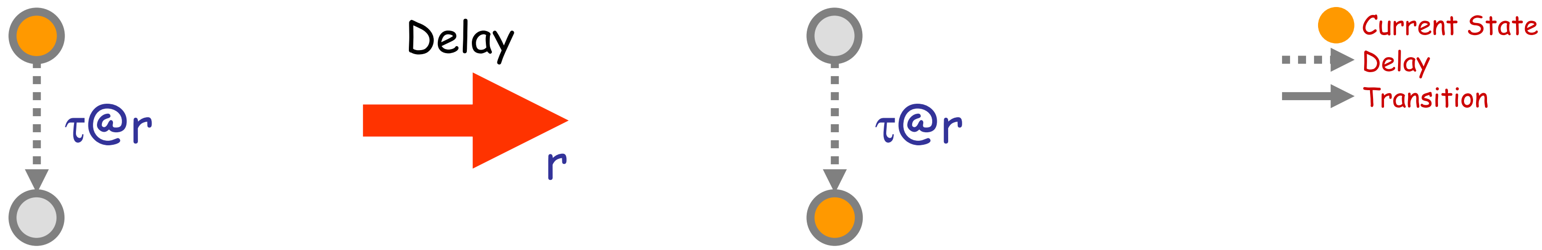
Communicating automata: a graphical FSA-like notation for "finite state restriction-free π-calculus processes". *Interacting* automata do not even exchange values on communication.

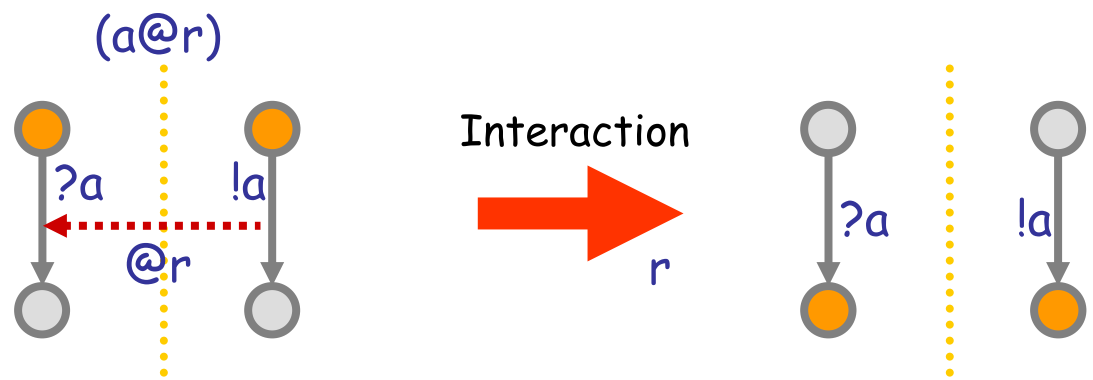
The stochastic version has *rates* on communications, and delays.

"Finite state" means: no composition or restriction inside recursion.

Analyzable by standard Markovian techniques, by first computing the "product automaton" to obtain the underlying finite Markov transition system. [Buchholz]

Interacting Automata Transition Rules



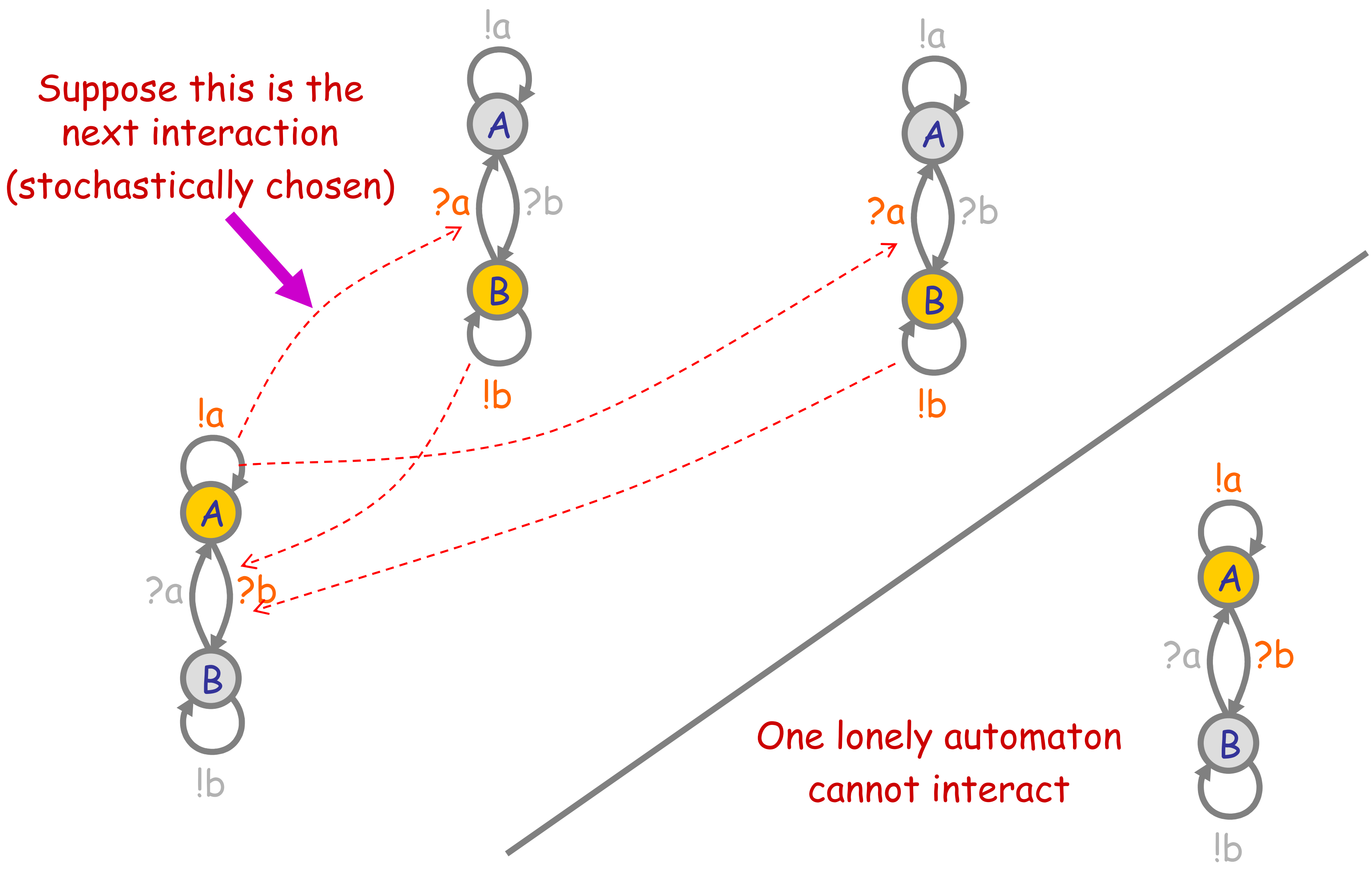


Interactions have rates. Actions DO NOT have rates.

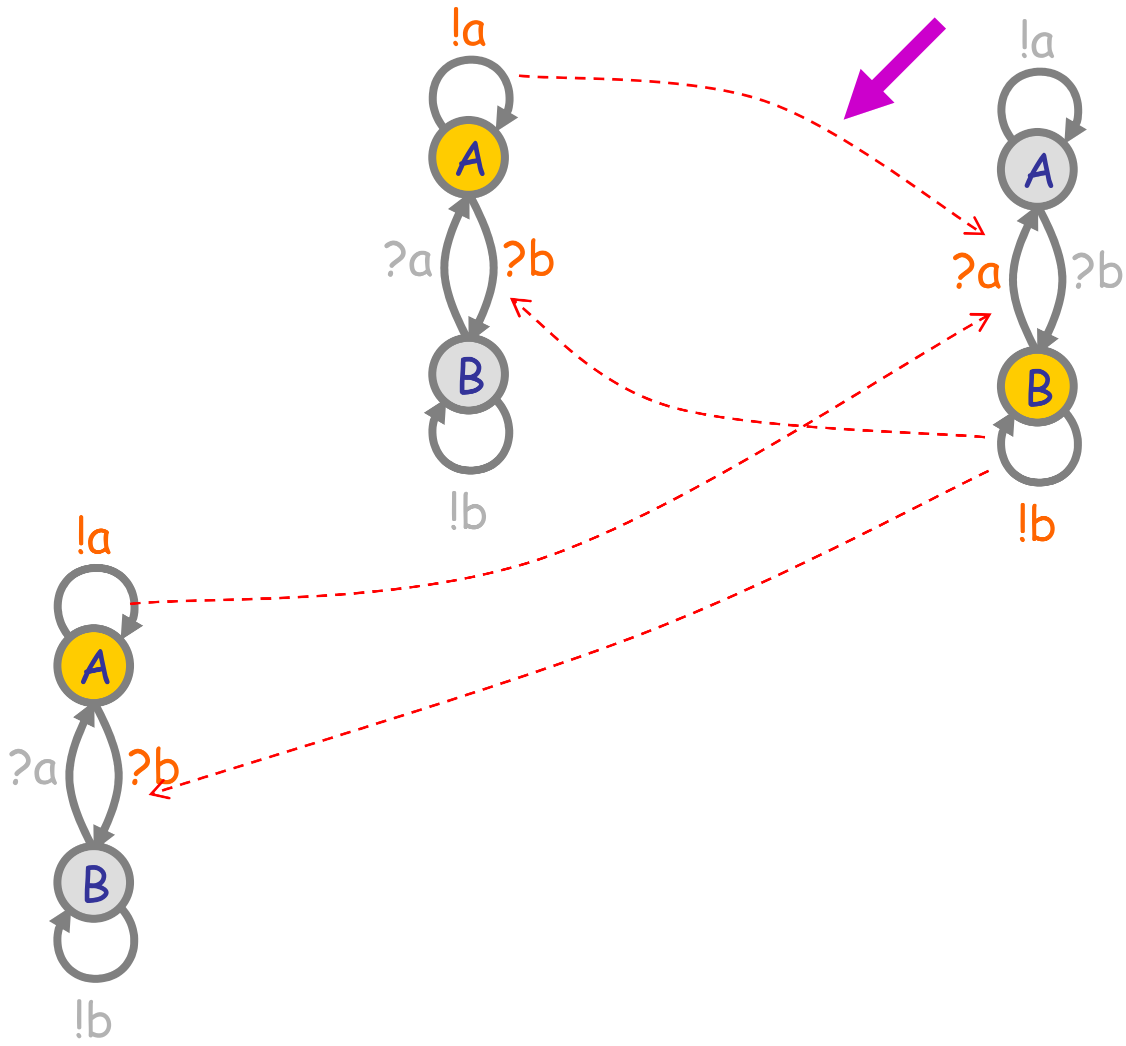
Q: What kind of mass behavior can this produce?

(We need to understand that if want to understand biochemical systems.)

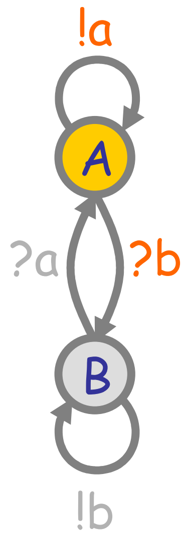
Interactions in a Population

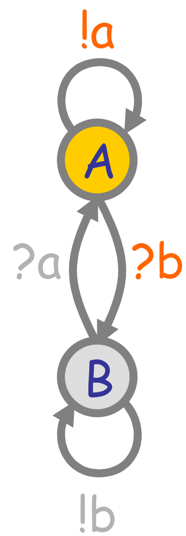


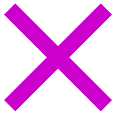
Interactions in a Population

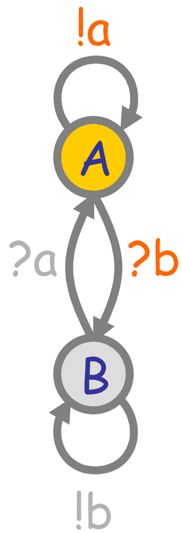


Interactions in a Population



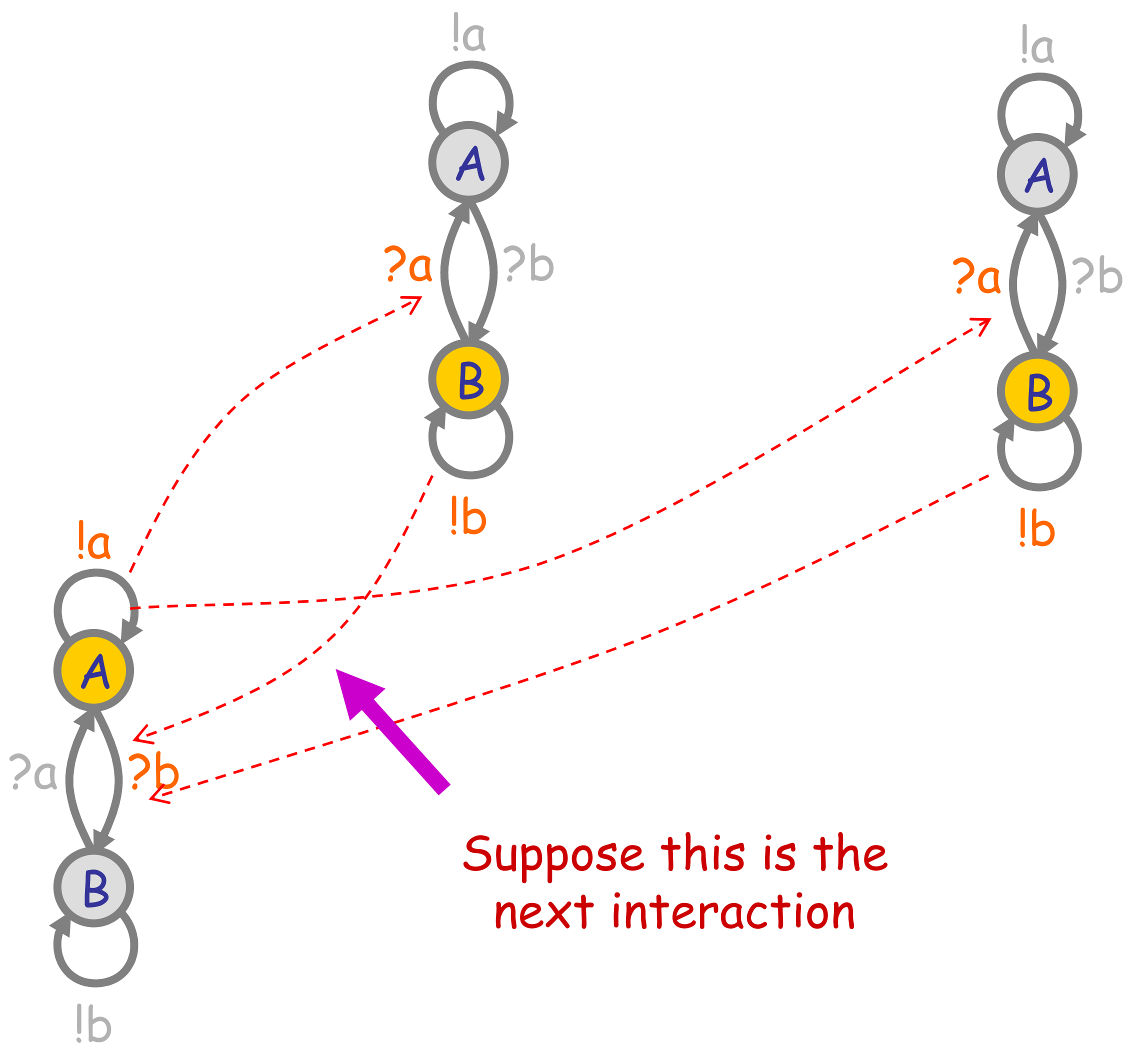






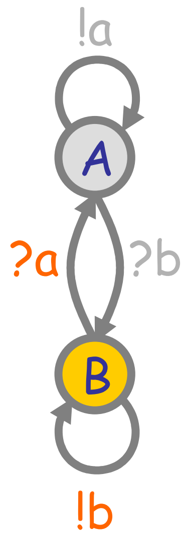
All-A stable
population

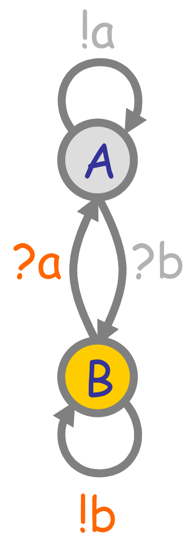
Interactions in a Population (2)

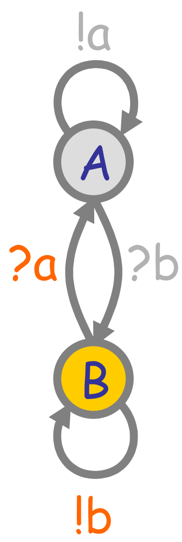


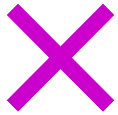
Suppose this is the next interaction

Interactions in a Population (2)





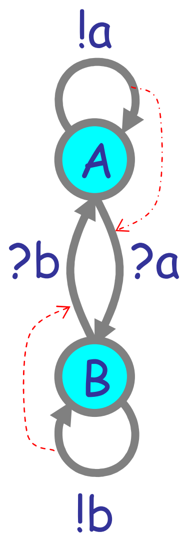




All-B stable population

Nondeterministic population behavior ("multistability")

Groupies and Celebrities



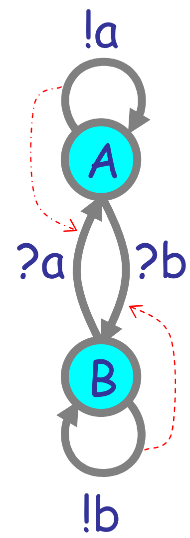
Celebrity

(does not want to be like somebody else)

```
directive sample 0.1 200
directive plot A(); B()

new a@1.0:chan()
new b@1.0:chan()

let A() = do !a; A() or ?a; B()
and B() = do !b; B() or ?b; A()

run 100 of (A() | B())
```

a@1.0

b@1.0



Groupie

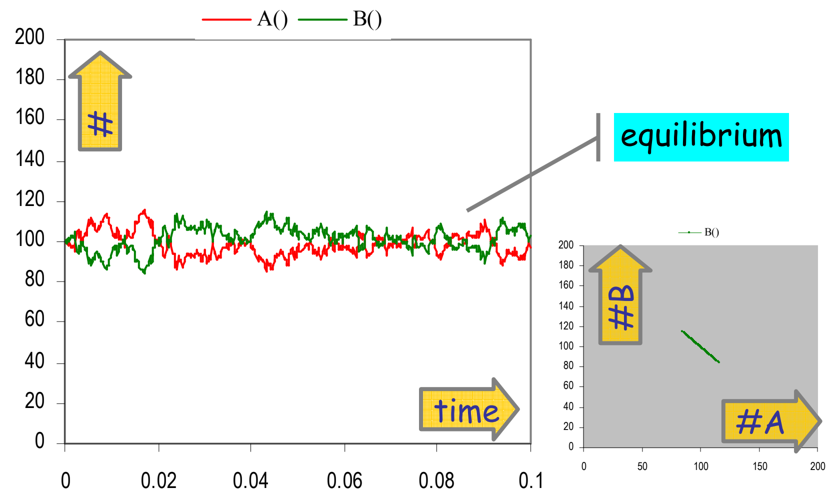
(wants to be like somebody different)

```
directive sample 0.1 200
directive plot A(); B()

new a@1.0:chan()
new b@1.0:chan()

let A() = do !a; A() or ?b; B()
and B() = do !b; B() or ?a; A()

run 100 of (A() | B())
```

a@1.0

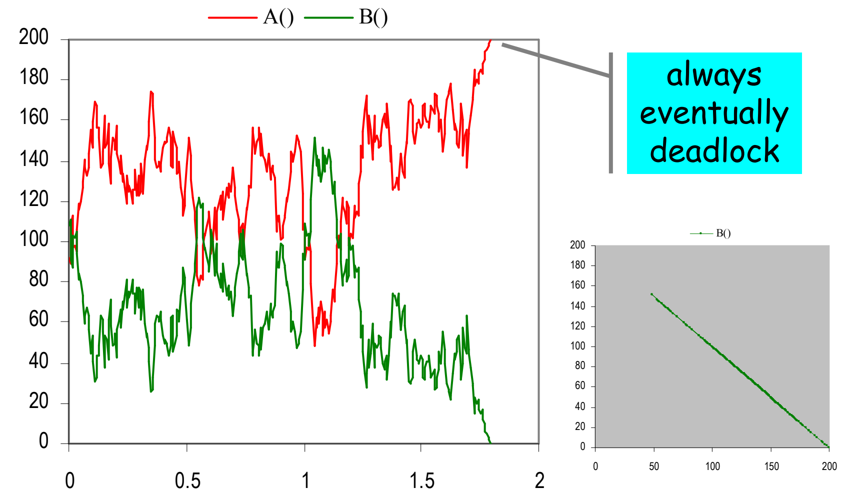
b@1.0

A stochastic collective of celebrities:



Stable because as soon as a A finds itself in the majority, it is more likely to find somebody in the same state, and hence change, so the majority is weakened.

A stochastic collective of groupies:

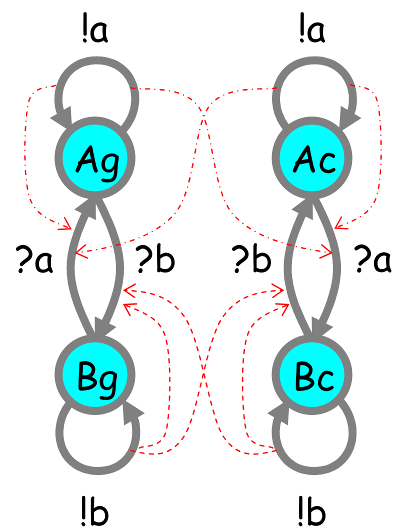


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

A way to break the deadlocks: Groupies with just a few Celebrities

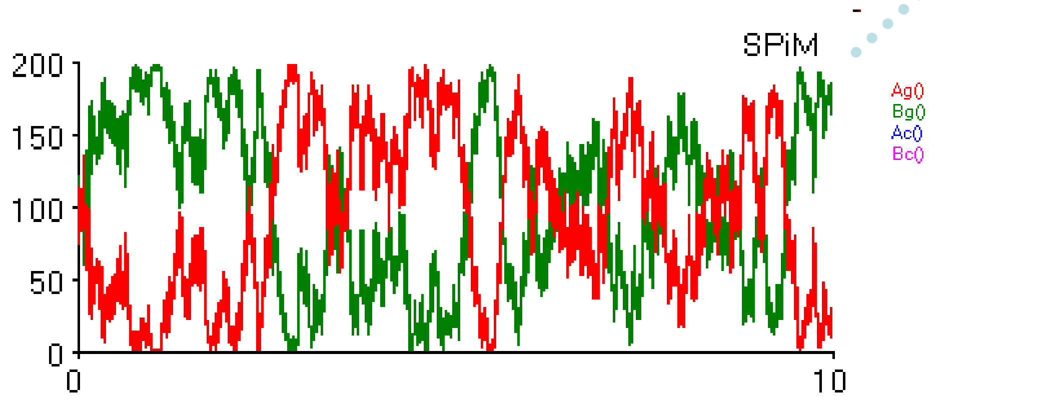
Many Groupies



A few Celebrities

```
directive sample 10.0
directive plot Ag(); Bg(); Ac(); Bc()

new a@1.0:chan()
new b@1.0:chan()

let Ac() = do !a; Ac() or ?a; Bc()
and Bc() = do !b; Bc() or ?b; Ac()

let Ag() = do !a; Ag() or ?b; Bg()
and Bg() = do !b; Bg() or ?a; Ag()

run 1 of Ac()
run 100 of (Ag() | Bg())
```

never deadlock

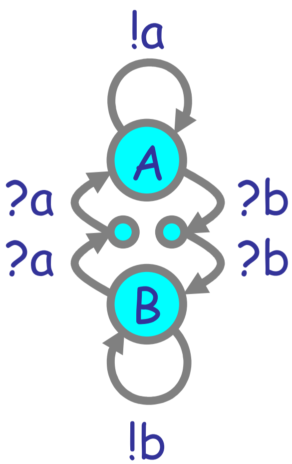


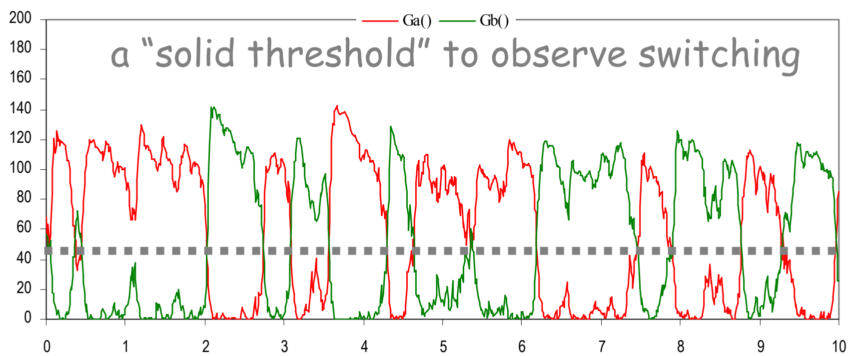
A tiny bit of "noise" can make a huge difference

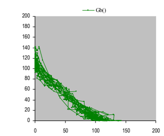
Regularity can arise not far from chaos

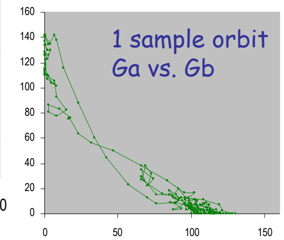
Hysteric Groupies

We can get more regular behavior from groupies if they "need more convincing", or "hysteresis" (history-dependence), to switch states.

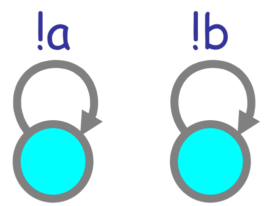








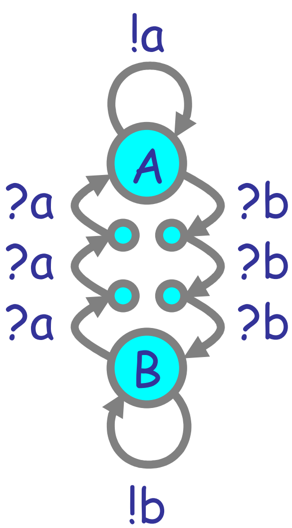

```
directive sample 10.0 1000
directive plot Ga(); Gb()

new a@1.0:chan()
new b@1.0:chan()

let Ga() = do !a; Ga() or ?b; ?b; Gb()
and Gb() = do !b; Gb() or ?a; ?a; Ga()

let Da() = !a; Da()
and Db() = !b; Db()

run 100 of (Ga() | Gb())
run   1 of (Da() | Db())
```

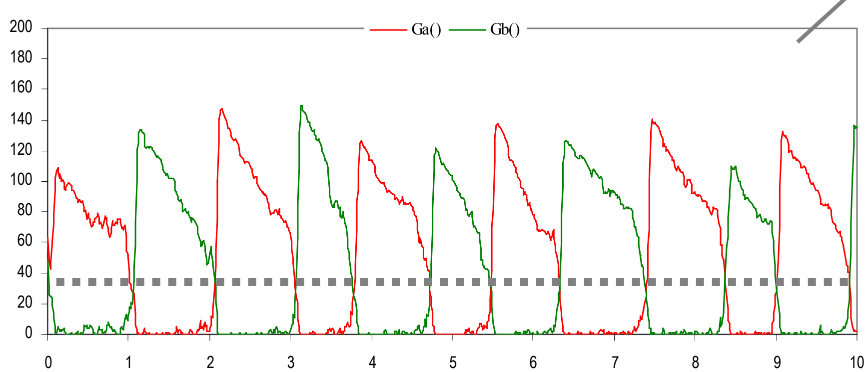


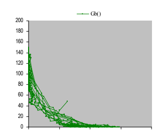
(With doping to break deadlocks)

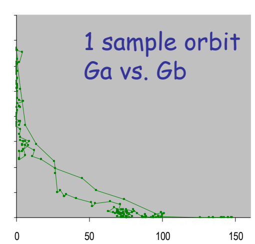
N.B.: It will not oscillate without doping (noise)

"regular" oscillation





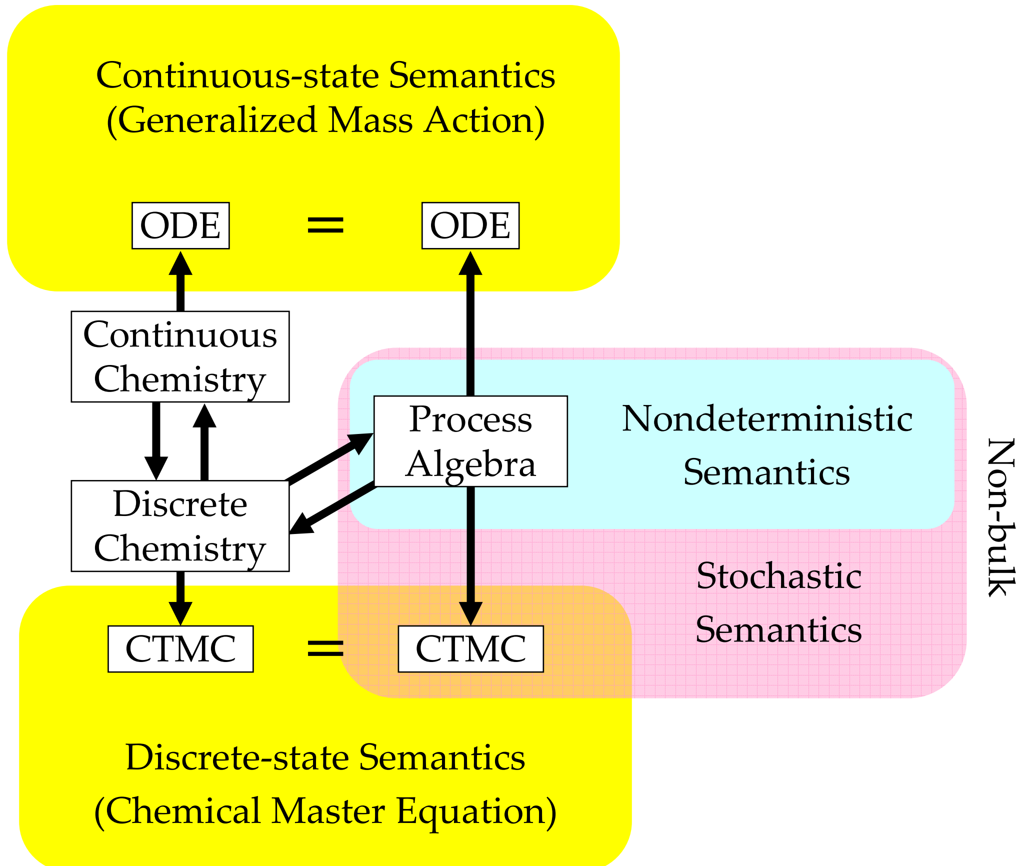





```
directive sample 10.0 1000
directive plot Ga(); Gb()

new a@1.0:chan()
new b@1.0:chan()

let Ga() = do !a; Ga() or ?b; ?b; ?b; Gb()
and Gb() = do !b; Gb() or ?a; ?a; ?a; Ga()

let Da() = !a; Da()
and Db() = !b; Db()

run 100 of (Ga() | Gb())
run   1 of (Da() | Db())
```

Semantics of Collective Behavior

The Two Semantic Faces of Chemistry



Luca Cardelli: "On Process Rate Semantics", showing that these diagrams commute.

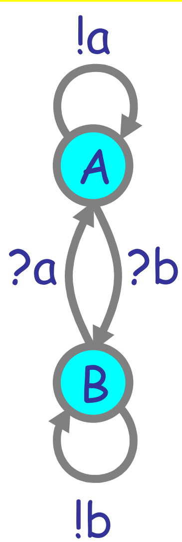
From Processes to Chemistry

Chemical Ground Form (CGF)

$E ::= X_1{=}M_1, \ldots, X_n{=}M_n$	Definitions	$(n \geq 0)$
$M ::= \pi_1;P_1 \oplus \ldots \oplus \pi_n;P_n$	Molecules	$(n \geq 0)$
$P ::= X_1 \mid \ldots \mid X_n$	Solutions	$(n \geq 0)$
$\pi ::= \tau_r \quad ?n_{(r)} \quad !n_{(r)}$	Interactions (delay, input, output)	
$CGF ::= E,P$	Definitions with Initial Conditions	

(To translate chemistry back to processes we need a bit more than simple automata: we may have "+" on the right of $\rightarrow$, that is we may need "|" after π.)

$\oplus$ is stochastic choice (vs. + for chemical reactions)
0 is the null solution ($P|0 = 0|P = P$)
and null molecule ($M\oplus 0 = 0\oplus M = M$) ($\tau_0;P = 0$)
X_i are distinct in E
Each name n is assigned a fixed rate r: $n_{(r)}$



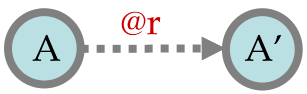
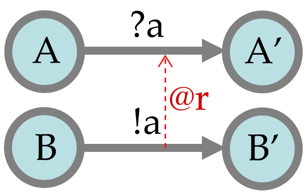
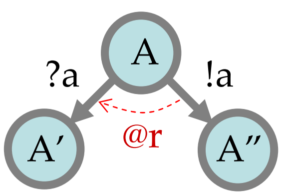
Ex: interacting automata
(which are finite-control CGFs: use "|" only in initial conditions):

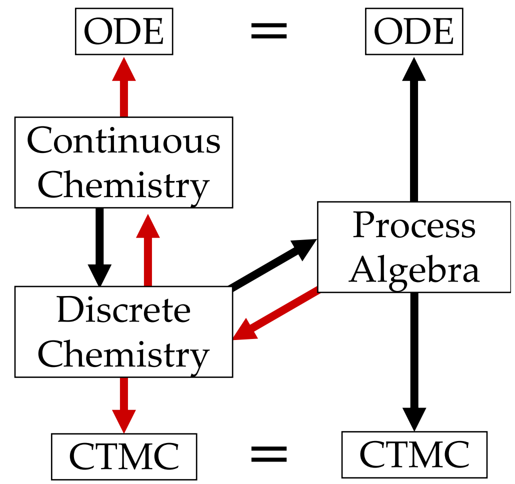
$A = !a;A \oplus ?b;B$ — Automaton in state A

$B = !b;B \oplus ?a;A$ — Automaton in state B

$A|A|B|B$ — Initial conditions: 2A and 2B

Processes to Chemistry

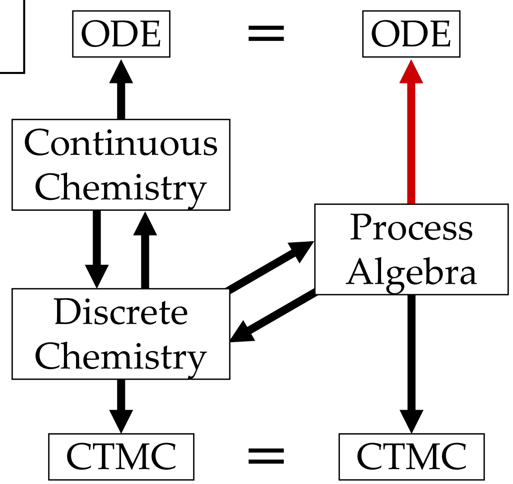
Automata	Discrete Chemistry	Continuous Chemistry	$\gamma = N_A V$
	$A \dashrightarrow^{r} A'$	$A \rightarrow^{k} A'$	with $k = r$
	$A+B \dashrightarrow^{r} A'+B'$	$A+B \rightarrow^{k} A'+B'$	with $k = r\gamma$
	$A+A \dashrightarrow^{2r} A'+A''$	$A+A \rightarrow^{2k} A'+A''$	with $k = r\gamma/2$
	$\#A_0$	$[A]_0$	with $[A]_0 = \#A/_0\gamma$
	↳ CTMC	↳ ODE	



Processes to GMA

Process Rate Equation for a CGF, E

$$[X]^{\bullet}_{E} = (\Sigma(Y{\in}E)\ Accr_{E}(Y,X)\cdot[Y]) - Depl_{E}(X)\cdot[X] \quad \text{for all } X{\in}E$$



$Depl_{E}(X) =$

$\quad \Sigma(i{:}\ E.X.i{=}\tau_{(r)};P)\ r\ +$

$\quad \Sigma(i{:}\ E.X.i{=}?a_{(r)};P)\ r\gamma\cdot OutsOn_{E}(a)\ +$

$\quad \Sigma(i{:}\ E.X.i{=}!a_{(r)};P)\ r\gamma\cdot InsOn_{E}(a)$

$Accr_{E}(Y, X) =$

$\quad \Sigma(i{:}\ E.Y.i{=}\tau_{(r)};P)\ \#X(P)\cdot r\ +$

$\quad \Sigma(i{:}\ E.Y.i{=}?a_{(r)};P)\ \#X(P)\cdot r\gamma\cdot OutsOn_{E}(a)\ +$

$\quad \Sigma(i{:}\ E.Y.i{=}!a_{(r)};P)\ \#X(P)\cdot r\gamma\cdot InsOn_{E}(a)$

$InsOn_{E}(a) = \Sigma(Y{\in}E)\ \#\{Y.i \mid E.Y.i{=}?a_{(r)};P\}\cdot[Y]$

$OutsOn_{E}(a) = \Sigma(Y{\in}E)\ \#\{Y.i \mid E.Y.i{=}!a_{(r)};P\}\cdot\cdot[Y]$

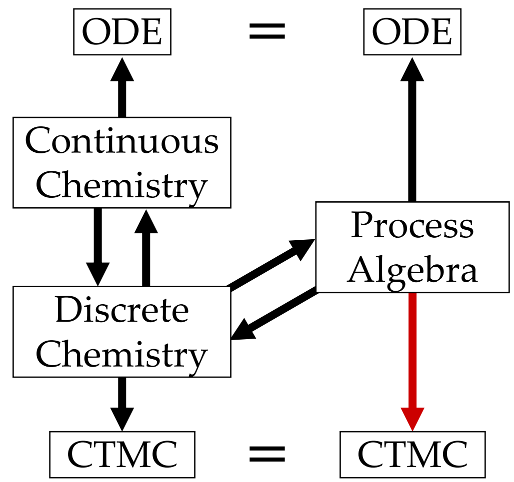
$X = \tau_{(r)};0$	$[X]^{\bullet} = -r[X]$
$X = ?a_{(r)};0$ $Y = !a_{(r)};0$	$[X]^{\bullet} = -r\gamma[X][Y]$ $[Y]^{\bullet} = -r\gamma[X][Y]$
$X = ?a_{(r)};0 \oplus !a_{(r)};0$	$[X]^{\bullet} = -2r\gamma[X]^2$

Processes to CME

Process Master Equation for a CGF, E

$$\partial pr(p,t)/\partial t \;=\; \Sigma_{\iota \in \Im}\; a_\iota(p-v_\iota)\cdot pr(p-v_\iota,t) - a_\iota(p)\cdot pr(p,t) \qquad \text{for all } p \in States(E)$$

$pr(p,t) = Pr\{\mathbf{S}(t)=p \mid \mathbf{S}(0)=p_0\}$ is the conditional probability of the system being in state p (a multiset of molecules) at time t given that it was in state p_0 at time 0.



$\Im = \{\{X.i\}$ *s.t.* $E.X.i = \tau_{(r)};Q\} \cup$
$\{\{X.i, Y.j\}$ *s.t.* $E.X.i = ?n_{(r)};Q$ and $E.Y.j = !n_{(r)};R\}$
is the set of possible interactions in E

v_ι is the *state change* caused by an interaction $\iota \in \Im$.

- $v_\iota = -X+Q$ if $\iota = \{X.i\}$ *s.t.* $E.X.i = \tau_{(r)};Q$
- $v_\iota = -X-Y+Q\_R$ if $\iota = \{X.i, Y.j\}$ *s.t.* $E.X.i = ?n_{(r)};Q$ and $E.Y.j = !n_{(r)};R$

a_ι is the *propensity* of interaction ι in state p. Here $p^{\#X}$ is the number of X in p.

- $a_\iota(p) = r\cdot p^{\#X}$ if $\iota = \{X.i\}$ *s.t.* $E.X.i = \tau_{(r)};Q$
- $a_\iota(p) = r\cdot p^{\#X}\cdot p^{\#Y}$ if $\iota = \{X.i, Y.j\}$ *s.t.* $X \neq Y$ and $E.X.i = ?a_{(r)};Q$ and $E.Y.j = !a_{(r)};R$
- $a_\iota(p) = r\cdot p^{\#X}\cdot(p^{\#X}-1)$ if $\iota = \{X.i, X.j\}$ *s.t.* $E.X.i = ?a_{(r)};Q$ and $E.X.j = !a_{(r)};R$

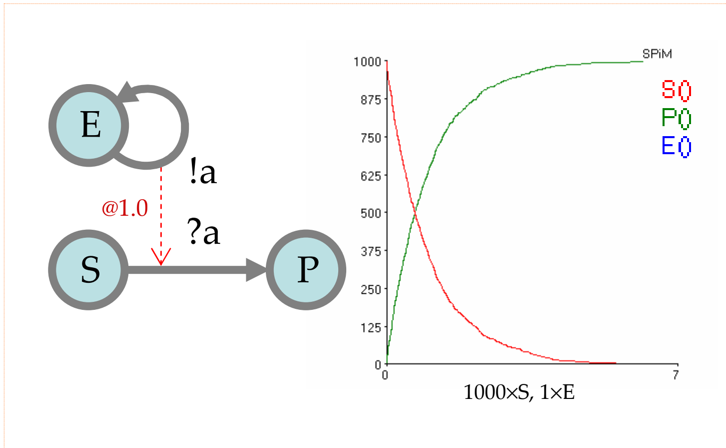
Examples of stochastic collectives where:

(1) Simulation is puzzling and ODE analysis is more useful.

(2) ODE analysis is puzzling and simulation is more useful.

Zero-Order Regime

Second-order and Zero-order Regime

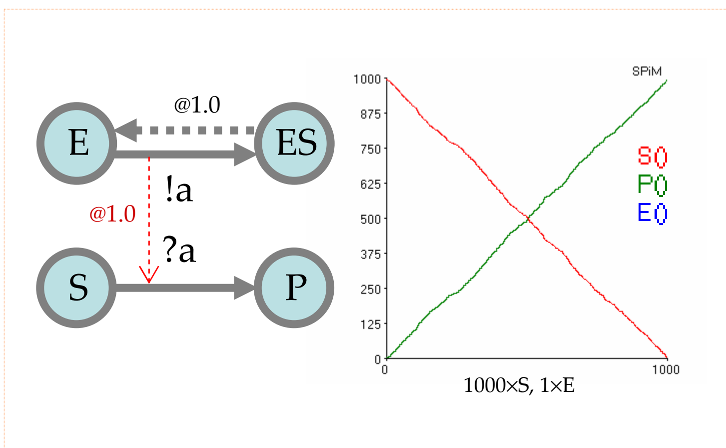


E+S $\rightarrow^{r}$ E+P

```
directive sample 1000.0
directive plot S(); P(); E()

new a@1.0:chan()

let E() = !a; E()
and S() = ?a; P()
and P() = ()

run (1 of  E() | 1000 of S())
```

Second-Order Regime

$[S]^{\bullet} = -r[E][S]$



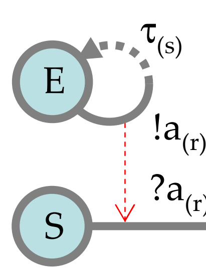
E+S $\rightarrow^{r}$ ES+P

ES $\rightarrow^{s}$ E

```
directive sample 1000.0
directive plot S(); P(); E()

new a@1.0:chan()

let E() = !a; delay@1.0; E()
and S() = ?a; P()
and P() = ()

run (1 of  E() | 1000 of S())
```

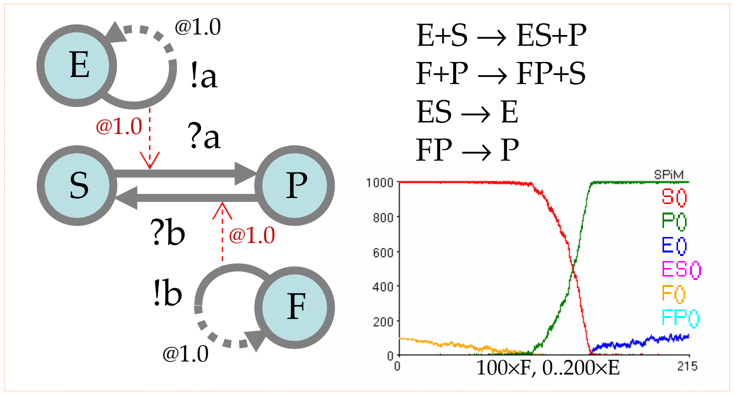
Zero-Order Regime

$[S]^{\bullet} \cong -1$



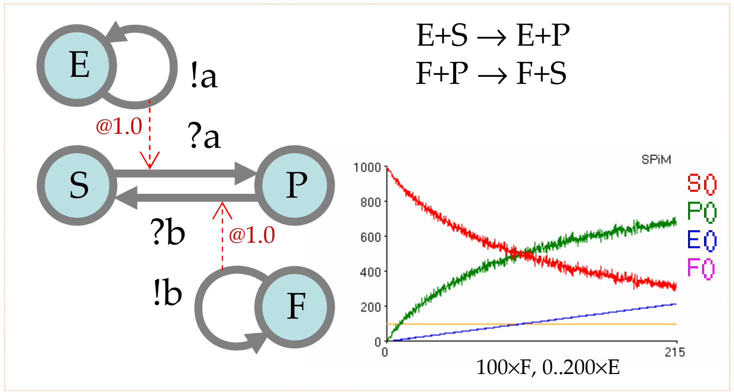
Notation

Ultrasensitivity




```
directive sample 215.0
directive plot S(); P(); E(); ES(); F(); FP()

new a@1.0:chan() new b@1.0:chan()

let S() = ?a; P()
and P() = ?b; S()

let E() = !a; delay@1.0; E()
and F() = !b; delay@1.0; F()

run 1000 of S()

let clock(t:float, tick:chan) =       (* sends a tick every t time *)
  (val ti = t/100.0 val d = 1.0/ti     (* by 100-step erlang timers *)
   let step(n:int) = if n<=0 then !tick; clock(t,tick) else delay@d; step(n-1)
   run step(100))
let Sig(p:proc(), tick:chan) = (p() | ?tick; Sig(p,tick))
let raising(p:proc(), t:float) =
  (new tick:chan run (clock(t,tick) | Sig(p,tick)))

run 100 of F()
run raising(E,1.0)
```

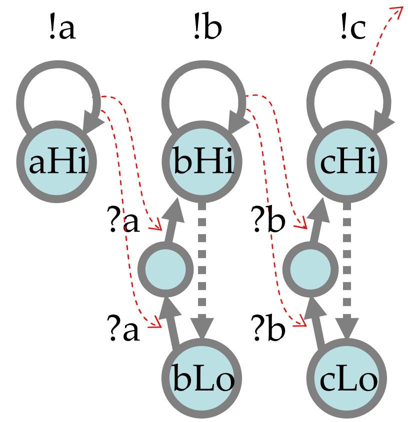
Zero-Order Regime
A small E-F inbalance causes
a much larger S-P switch.

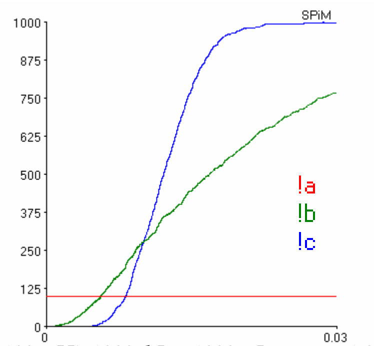



```
directive sample 215.0 1000
directive plot S(); P(); E(); F()

new a@1.0:chan() new b@1.0:chan()

let S() = ?a; P()
and P() = ?b; S()

let E() = !a; E()
and F() = !b; F()

run 1000 of S()

let clock(t:float, tick:chan) =       (* sends a tick every t time *)
  (val ti = t/100.0 val d = 1.0/ti     (* by 100-step erlang timers *)
   let step(n:int) = if n<=0 then !tick; clock(t,tick) else delay@d; step(n-1)
   run step(100))
let Sig(p:proc(), tick:chan) = (p() | ?tick; Sig(p,tick))
let raising(p:proc(), t:float) =
  (new tick:chan run (clock(t,tick) | Sig(p,tick)))

run 100 of F()
run raising(E,1.0)
```

Second-Order Regime

Cascades



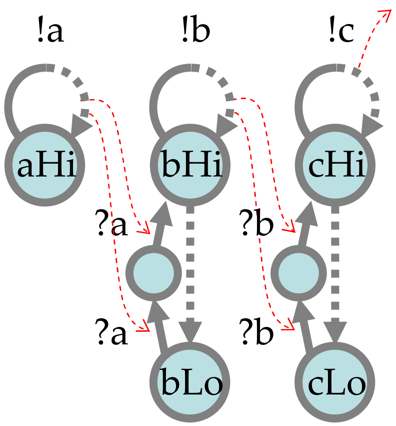


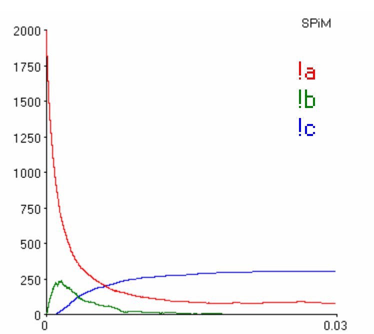
100×aHi, 1000×bLo, 1000×cLo, rates=1.0

Second-Oder Regime cascade:
a signal amplifier (MAPK)

aHi > 0 ⇒ cHi = max

```
directive sample 0.03
directive plot !a; !b; !c

new a@1.0:chan new b@1.0:chan new c@1.0:chan

let Amp_hi(a:chan, b:chan) =
  do !b; Amp_hi(a,b) or delay@1.0; Amp_lo(a,b)
and Amp_lo(a:chan, b:chan) =
  ?a; ?a; Amp_hi(a,b)

run 1000 of (Amp_lo(a,b) | Amp_lo(b,c))

let A() = !a; A()
run 100 of A()
```





2000×aHi, 1000×bLo, 1000×cLo, rates=1.0

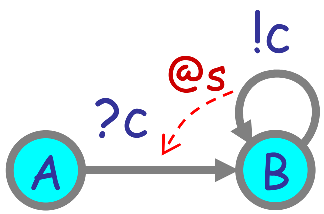
Zero-Oder Regime cascade:
a signal *divider!*

aHi = max ⇒ cHi = 1/3 max

```
directive sample 0.03
directive plot !a; !b; !c

new a@1.0:chan new b@1.0:chan new c@1.0:chan

let Amp_hi(a:chan, b:chan) =
  do !b; delay@1.0; Amp_hi(a,b) or delay@1.0; Amp_lo(a,b)
and Amp_lo(a:chan, b:chan) =
  ?a; ?a; Amp_hi(a,b)

run 1000 of (Amp_lo(a,b) | Amp_lo(b,c))

let A() = !a; delay@1.0; A()
run 2000 of A()
```

Nonlinear Transitions

Nonlinear Transition (NLT)



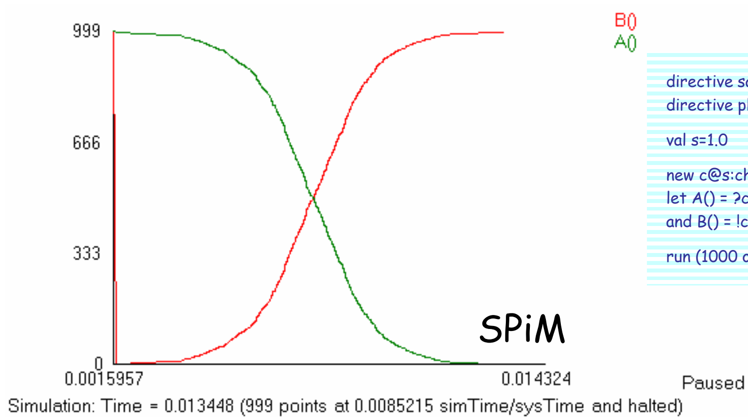
$A = ?c_{(s)};B$

$B = !c_{(s)};B$

$A+B \rightarrow^{s} B+B$

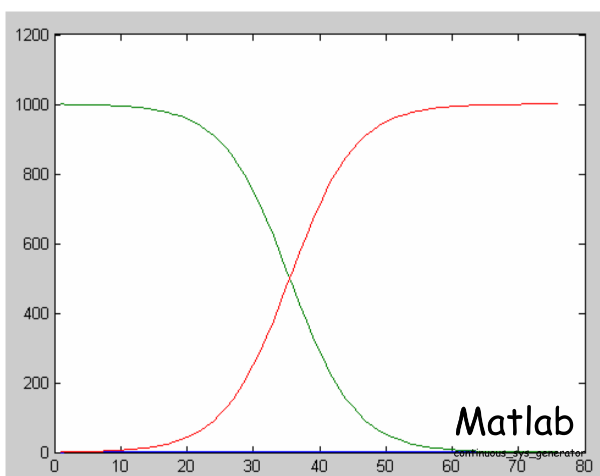
$[A]^{\bullet} = -s[A][B]$

$[B]^{\bullet} = s[A][B]$



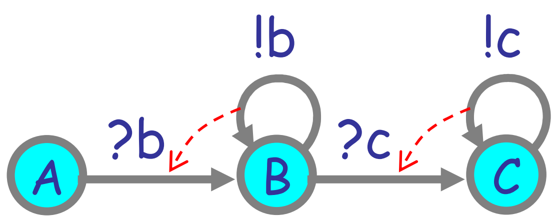

```
directive sample 0.02 1000
directive plot B(); A()

val s=1.0

new c@s:chan
let A() = ?c; B()
and B() = !c;B()

run (1000 of A() | 1 of B())
```

N.B.: needs at least 1 B to "get started".




```
interval/step [0:0.001:0.0]
(A)    dx1/dt = - x1*x2        1000.0
(B)    dx2/dt = x1*x2          1.0
```

Two NLTs: Bell Shape



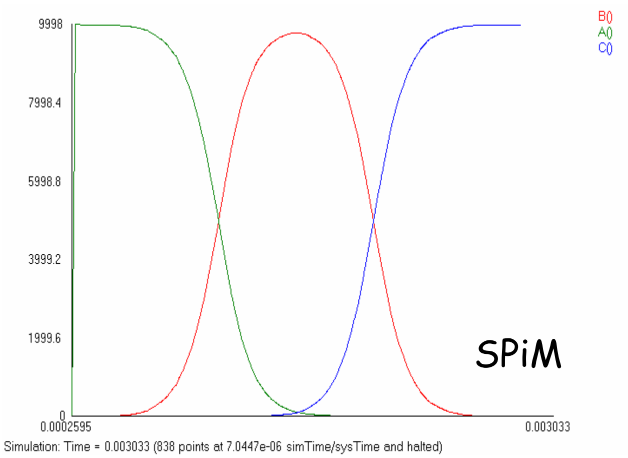
$[B]^\bullet = [B]([A]-[C])$

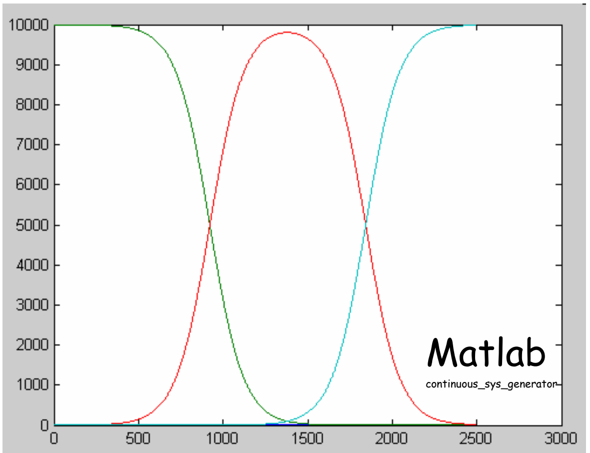
$A = ?b_{(1)};B$
$B = !b_{(1)};B \oplus ?c_{(1)};C$
$C = !c_{(1)};C$

$A+B \rightarrow^1 B+B$
$B+C \rightarrow^1 C+C$

$[A]^\bullet = -[A][B]$
$[B]^\bullet = [A][B]-[B][C]$
$[C]^\bullet = [B][C]$

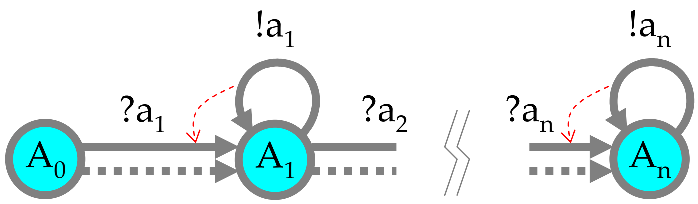
```
directive sample 0.0025 1000
directive plot B(); A(); C()

new b@1.0:chan new c@1.0:chan

let A() = ?b; B()
and B() = do !b;B() or ?c; C()
and C() = !c;C()

run ((10000 of A()) | B() | C())
```

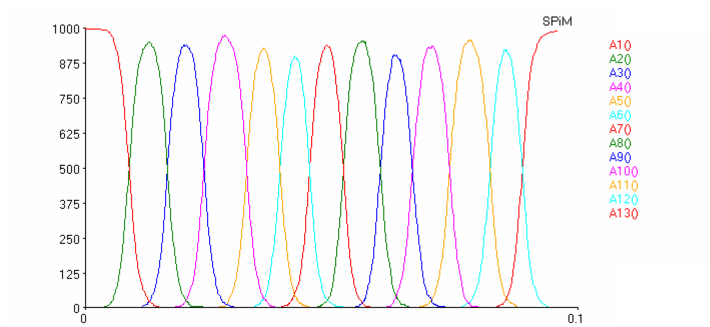




interval/step [0:0.000001:0.0025]			
(A)	dx1/dt = -x1*x2	10000.0	
(B)	dx2/dt = x1*x2 - x2*x3	1.0	
(C)	dx3/dt = x2*x3	1.0	

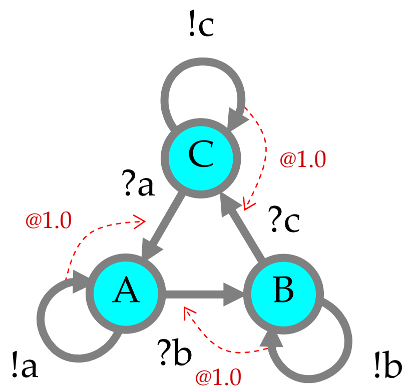
NLTs in Series: Soliton Propagation

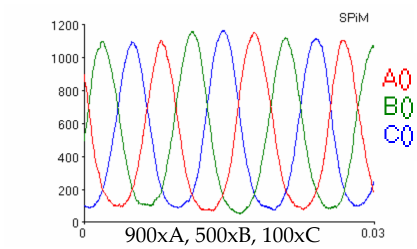





```
directive sample 0.1 1000
directive plot A1(); A2(); A3(); A4(); A5(); A6(); A7(); A8();
A9(); A10(); A11(); A12(); A13()

val r=1.0 val s=1.0

new a2@s:chan new a3@s:chan new a4@s:chan
new a5@s:chan new a6@s:chan new a7@s:chan
new a8@s:chan new a9@s:chan new a10@s:chan
new a11@s:chan new a12@s:chan new a13@s:chan
let A1() = do delay@r;A2() or ?a2; A2()
and A2() = do !a2;A2() or delay@r;A3() or ?a3; A3()
and A3() = do !a3;A3() or delay@r;A4() or ?a4; A4()
and A4() = do !a4;A4() or delay@r;A5() or ?a5; A5()
and A5() = do !a5;A5() or delay@r;A6() or ?a6; A6()
and A6() = do !a6;A6() or delay@r;A7() or ?a7; A7()
and A7() = do !a7;A7() or delay@r;A8() or ?a8; A8()
and A8() = do !a8;A8() or delay@r;A9() or ?a9; A9()
and A9() = do !a9;A9() or delay@r;A10() or ?a10; A10()
and A10() = do !a10;A10() or delay@r;A11() or ?a11; A11()
and A11() = do !a11;A11() or delay@r;A12() or ?a12; A12()
and A12() = do !a12;A12() or delay@r;A13() or ?a13; A13()
and A13() = !a13;A13()

run 1000 of A1()
```

NLT in a Cycle: Oscillator



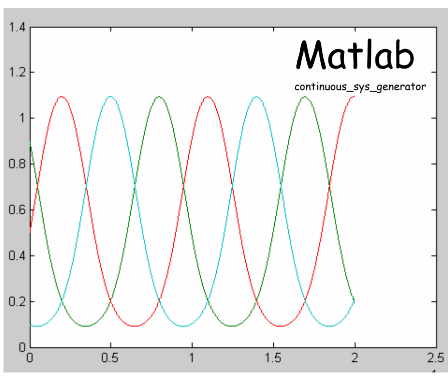



```
directive sample 0.03 1000
directive plot A(); B(); C()

new a@1.0:chan new b@1.0:chan new c@1.0:chan
let A() = do !a;A() or ?b; B()
and B() = do !b;B() or ?c; C()
and C() = do !c;C() or ?a; A()

run (900 of A() | 500 of B() | 100 of C())
```

$$A = !a_{(s)};A \oplus ?b_{(s)};B$$
$$B = !b_{(s)};B \oplus ?c_{(s)};C$$
$$C = !c_{(s)};C \oplus ?a_{(s)};A$$

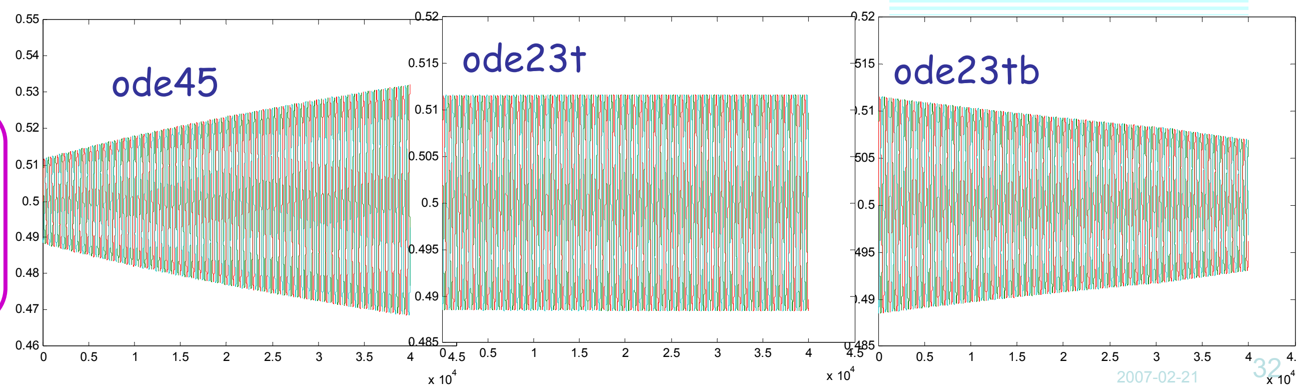
$$A+B \to^{s} B+B$$
$$B+C \to^{s} C+C$$
$$C+A \to^{s} A+A$$

$$[A]^{\bullet} = -s[A][B]+s[C][A]$$
$$[B]^{\bullet} = -s[B][C]+s[A][B]$$
$$[C]^{\bullet} = -s[C][A]+s[B][C]$$




```
interval/step [0:0.001:20.0]
(A)   dx1/dt = - x1*x2 + x3*x1   0.9
(B)   dx2/dt = - x2*x3 + x1*x2   0.5
(C)   dx3/dt = - x3*x1 + x2*x3   0.1
```

```
interval/step [0:0.01:400.0]
(A)   dx1/dt = - x1*x2 + x3*x1   0.51
(B)   dx2/dt = - x2*x3 + x1*x2   0.5
(C)   dx3/dt = - x3*x1 + x2*x3   0.49
```

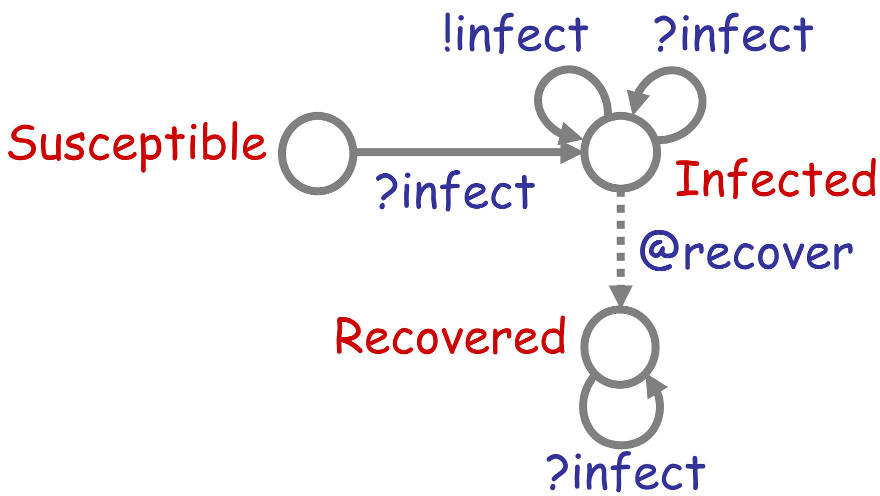


Epidemics

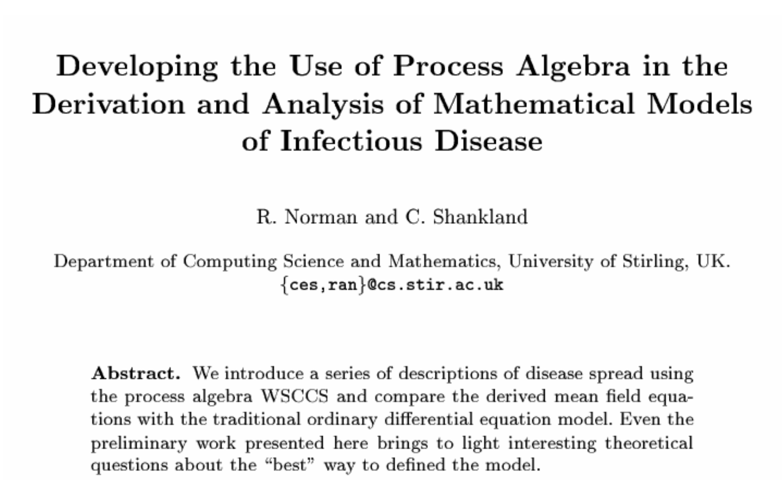
Kermack, W. O. and McKendrick, A. G. "**A Contribution to the Mathematical Theory of Epidemics.**" *Proc. Roy. Soc. Lond. A* **115**, 700-721, 1927.

http://mathworld.wolfram.com/Kermack-McKendrickModel.html

Epidemics




```
directive sample 500.0 1000
directive plot Recovered(); Susceptible(); Infected()

new infect @0.001:chan()
val recover = 0.03

let Recovered() =
  ?infect; Recovered()

and Susceptible() =
  ?infect; Infected()

and Infected() =
  do !infect; Infected()
  or ?infect; Infected()
  or delay@recover; Recovered()

run (200 of Susceptible() | 2 of Infected())
```

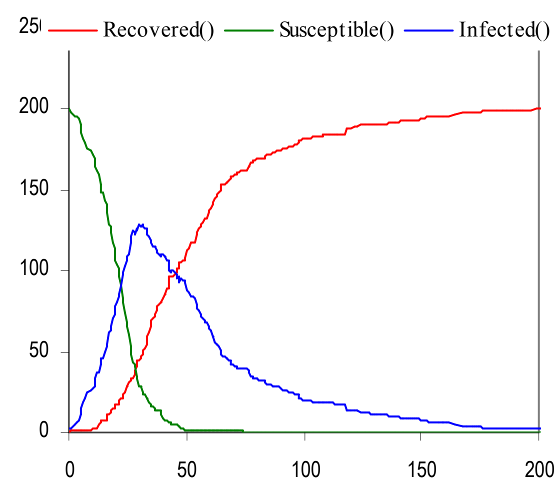


Developing the Use of Process Algebra in the Derivation and Analysis of Mathematical Models of Infectious Disease

R. Norman and C. Shankland

Department of Computing Science and Mathematics, University of Stirling, UK.
{ces,ran}@cs.stir.ac.uk

Abstract. We introduce a series of descriptions of disease spread using the process algebra WSCCS and compare the derived mean field equations with the traditional ordinary differential equation model. Even the preliminary work presented here brings to light interesting theoretical questions about the "best" way to defined the model.



ODE

$S = ?i_{(t)};I$
$I = !i_{(t)};I \oplus ?i_{(t)};I \oplus \tau_r;R$
$R = ?i_{(t)};R$

Differentiating Processes!

$S + I \rightarrow^{t\gamma} I + I$
$I + I \rightarrow^{t\gamma} I + I$
$I \rightarrow^{r} R$
$R + I \rightarrow^{t\gamma} R + I$

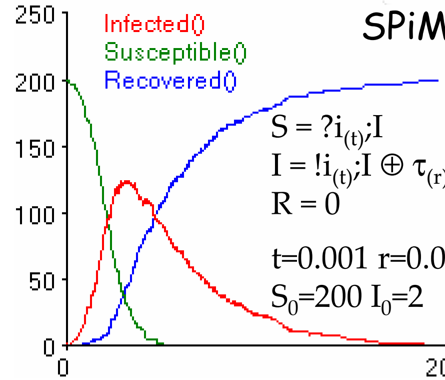
"useless" reactions

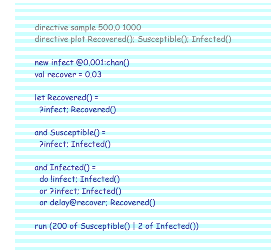
$[S]^\bullet = -t\gamma[S][I]$
$[I]^\bullet = t\gamma[S][I]-r[I]$
$[R]^\bullet = r[I]$

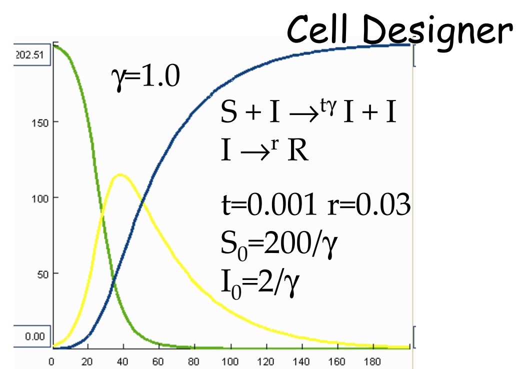
Automata produce the standard ODEs!

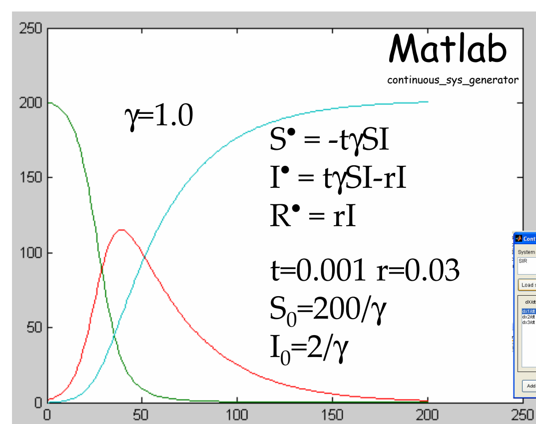
$$\frac{dS}{dt} = -aIS$$
$$\frac{dI}{dt} = aIS - bI$$
$$\frac{dR}{dt} = bI$$

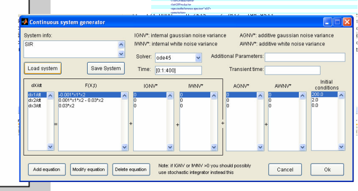
(the Kermack-McKendrick, or SIR model)



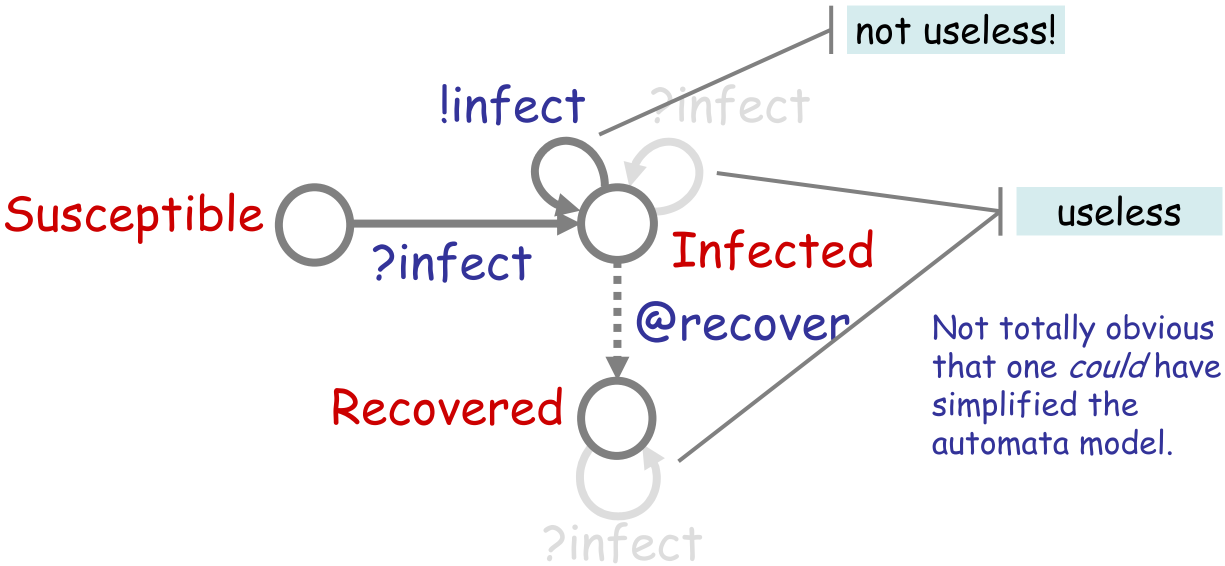








Simplified Model

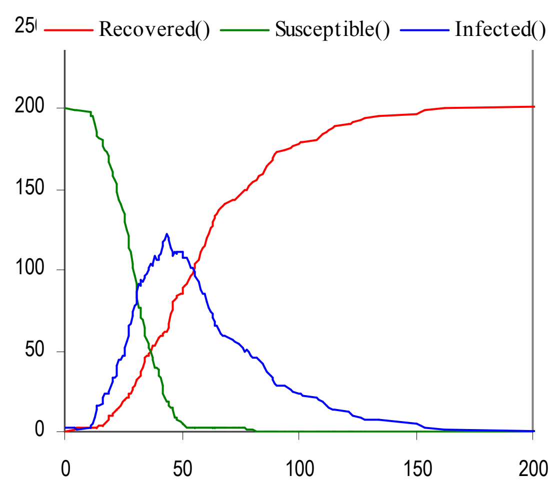


$$S = ?i_{(t)};I$$
$$I = !i_{(t)};I \oplus \tau_r;R$$
$$R = 0$$

$$S + I \rightarrow^{t\gamma} I + I$$
$$I \rightarrow^{r} R$$

$$[S]^\bullet = -t\gamma[S][I]$$
$$[I]^\bullet = t\gamma[S][I]-r[I]$$
$$[R]^\bullet = r[I]$$

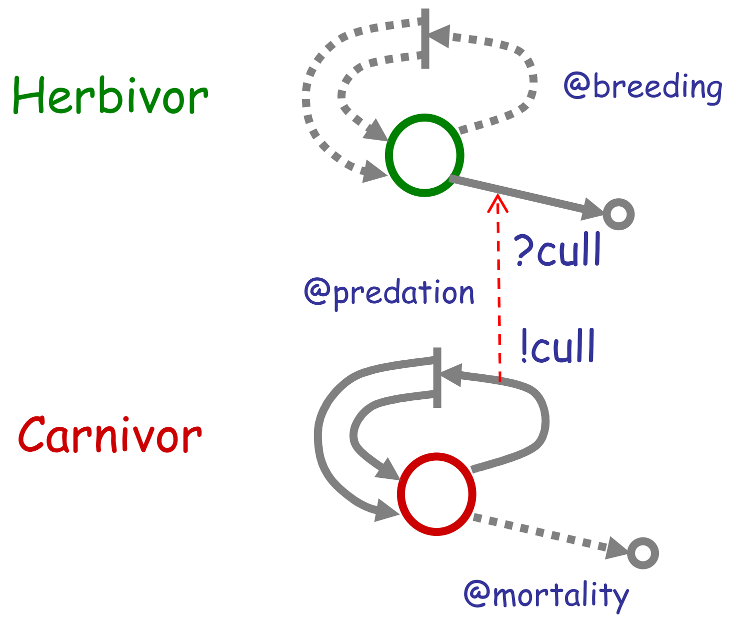
```
directive sample 500.0 1000
directive plot Recovered(); Susceptible(); Infected()

new infect @0.001:chan()
val recover = 0.03

let Recovered() =
  ()

and Susceptible() =
  ?infect; Infected()

and Infected() =
  do !infect; Infected()
  or delay@recover; Recovered()

run (200 of Susceptible() | 2 of Infected())
```



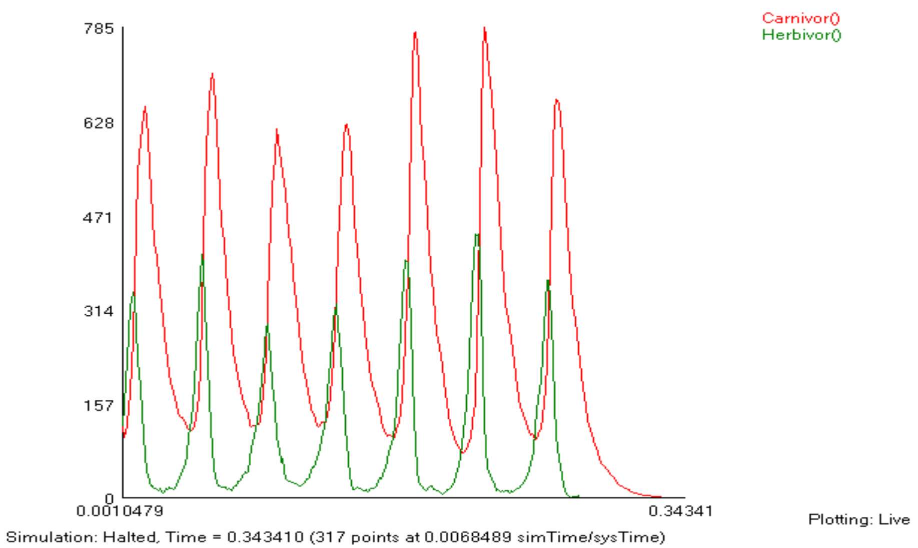
Same ODE, hence equivalent automata models.

Lotka-Volterra

Predator-Prey




```
directive sample 1.0 1000
directive plot Carnivor(); Herbivor()

val mortality = 100.0
val breeding = 300.0
val predation = 1.0
new cull @predation:chan()

let Herbivor() =
  do delay@breeding; (Herbivor() | Herbivor())
  or ?cull; ()

and Carnivor() =
  do delay@mortality; ()
  or !cull; (Carnivor() | Carnivor())

run 100 of Herbivor()
run 100 of Carnivor()
```



An unbounded state system!

Lotka-Volterra in Matlab

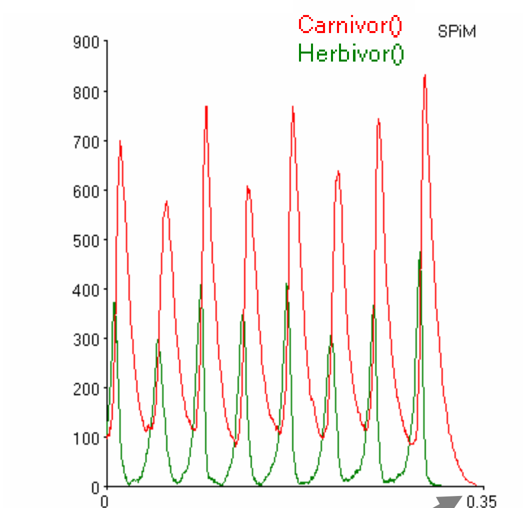
$H = \tau_b; (H|H) \oplus ?c_{(p)};0$
$C = \tau_m;0 \oplus !c_{(p)};(C|C)$
$\#H_0, \#C_0$

$H \to^b H + H$
$C \to^m 0$
$H + C \to^{p\gamma} C + C$
$[H]_0 = \#H_0/\gamma$
$[C]_0 = \#C_0/\gamma$

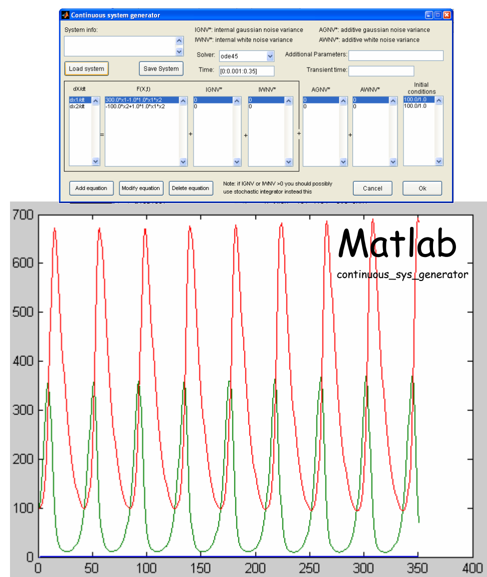
$[H]^\bullet = b[H]-p\gamma[H][C]$
$[C]^\bullet = -m[C]+p\gamma[H][C]$
$[H]_0 = \#H_0/\gamma$
$[C]_0 = \#C_0/\gamma$

$m=100.0$
$b=300.0$
$p=1.0$
$\gamma=1.0$
$\#H_0 = 100$
$\#C_0 = 100$

```
directive sample 0.35 1000
directive plot Carnivor(); Herbivor()

val mortality = 100.0
val breeding = 300.0
val predation = 1.0
new cull @predation:chan()

let Herbivor() =
  do delay@breeding; (Herbivor() | Herbivor())
  or ?cull; ()

and Carnivor() =
  do delay@mortality; ()
  or !cull; (Carnivor() | Carnivor())

run 100 of Herbivor()
run 100 of Carnivor()
```



Extinction



Parametric Processes

Chemical Parametric Form (CPF)

E	$::= X_1(\mathbf{p}_1)=M_1, \ldots, X_n(\mathbf{p}_n)=M_n$	Definitions	$(n \geq 0)$
M	$::= \pi_1;P_1 \oplus \ldots \oplus \pi_n;P_n$	Molecules	$(n \geq 0)$
P	$::= X_1(\mathbf{p}_1) \mid \ldots \mid X_n(\mathbf{p}_n)$	Solutions	$(n \geq 0)$
π	$::= \tau_r \quad ?n(\mathbf{p}) \quad !n(\mathbf{p})$	Interactions	
CPF	$::=$ E,P	with initial conditions	

$\oplus$ is stochastic choice (vs. + for chemical reactions)
0 is the null solution ($P|0 = 0|P = P$)
and null molecule ($M\oplus0 = 0\oplus M = M$) ($\tau_0;P = 0$)
X_i are distinct in E, **p** are vectors of names
p are vectors of distinct names when in binding position
Each free name n in E is assigned a fixed rate r:
written either $n_{(r)}$, or $\rho_{CPF}(n)=r$.

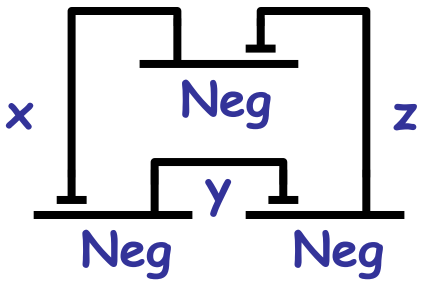
Not bounded-state systems.
Not finite-control systems.
But still finite-species systems.

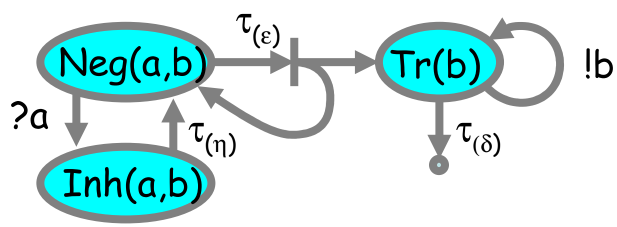
A translation from CPF to CGF exists
(expanding all possible instantiation of parameters from the initial conditions)

An incremental translation algorithm exists
(expanding on demand from initial conditions)

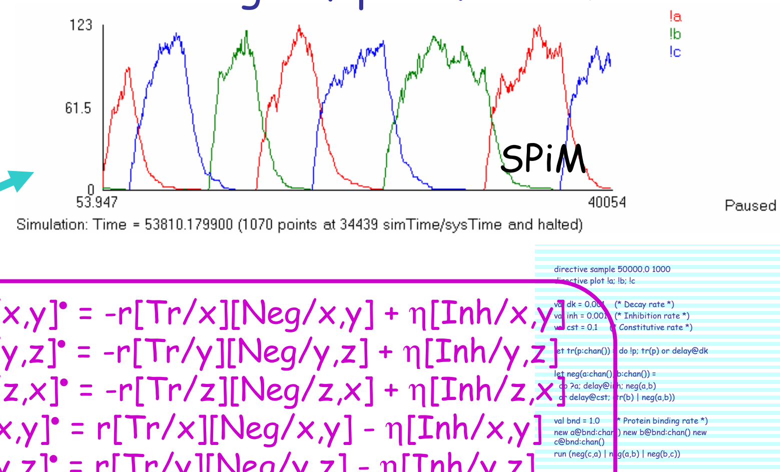
And Yet It Moves

The Repressilator





A fine stochastic oscillator over a wide range of parameters.



Parametric representation

$\text{Neg}(a,b) = ?a;\ \text{Inh}(a,b) \oplus \tau_\varepsilon;\ (\text{Tr}(b) \mid \text{Neg}(a,b))$

$\text{Inh}(a,b) = \tau_\eta;\ \text{Neg}(a,b)$

$\text{Tr}(b) = !b;\ \text{Tr}(b) \oplus \tau_\gamma;\ 0$

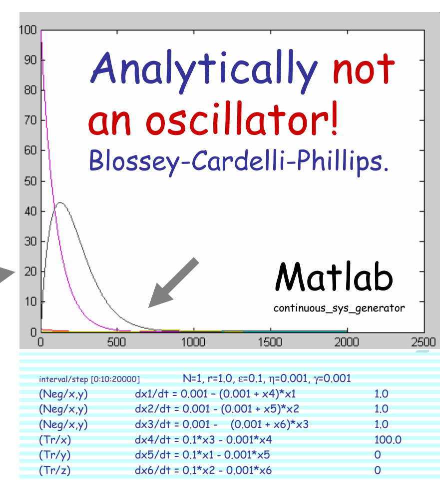
$\text{Neg}(x_{(r)},y_{(r)}) \mid \text{Neg}(y_{(r)},z_{(r)}) \mid \text{Neg}(z_{(r)},x_{(r)})$

$\text{Neg}/x,y \to^{\varepsilon} \text{Tr}/y + \text{Neg}/x,y$
$\text{Neg}/y,z \to^{\varepsilon} \text{Tr}/z + \text{Neg}/y,z$
$\text{Neg}/z,x \to^{\varepsilon} \text{Tr}/x + \text{Neg}/z,x$
$\text{Tr}/x + \text{Neg}/x,y \to^{r} \text{Tr}/x + \text{Inh}/x,y$
$\text{Tr}/y + \text{Neg}/y,z \to^{r} \text{Tr}/y + \text{Inh}/y,z$
$\text{Tr}/z + \text{Neg}/z,x \to^{r} \text{Tr}/z + \text{Inh}/z,x$
$\text{Inh}/x,y \to^{\eta} \text{Neg}/x,y$
$\text{Inh}/y,z \to^{\eta} \text{Neg}/y,z$
$\text{Inh}/z,x \to^{\eta} \text{Neg}/z,x$
$\text{Tr}/x \to^{\gamma} 0$
$\text{Tr}/y \to^{\gamma} 0$
$\text{Tr}/z \to^{\gamma} 0$
$\text{Neg}/x,y + \text{Neg}/y,z + \text{Neg}/z,x$

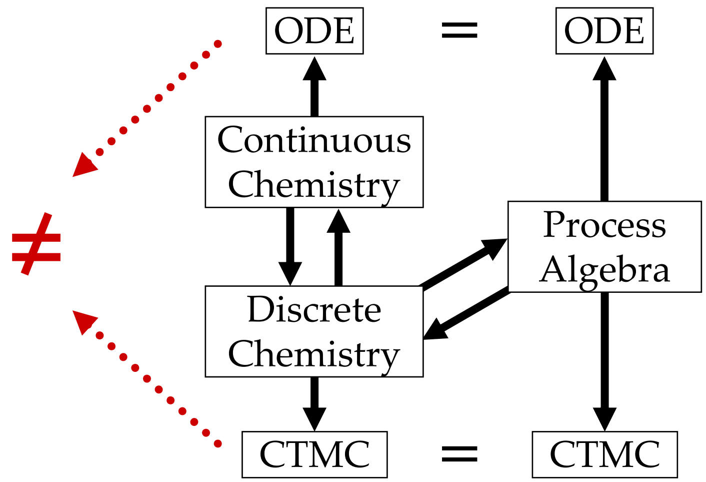
$[\text{Neg}/x,y]^\bullet = -r[\text{Tr}/x][\text{Neg}/x,y] + \eta[\text{Inh}/x,y]$
$[\text{Neg}/y,z]^\bullet = -r[\text{Tr}/y][\text{Neg}/y,z] + \eta[\text{Inh}/y,z]$
$[\text{Neg}/z,x]^\bullet = -r[\text{Tr}/z][\text{Neg}/z,x] + \eta[\text{Inh}/z,x]$
$[\text{Inh}/x,y]^\bullet = r[\text{Tr}/x][\text{Neg}/x,y] - \eta[\text{Inh}/x,y]$
$[\text{Inh}/y,z]^\bullet = r[\text{Tr}/y][\text{Neg}/y,z] - \eta[\text{Inh}/y,z]$
$[\text{Inh}/z,x]^\bullet = r[\text{Tr}/z][\text{Neg}/z,x] - \eta[\text{Inh}/z,x]$
$[\text{Tr}/x]^\bullet = \varepsilon[\text{Neg}/z,x] - \gamma[\text{Tr}/x]$
$[\text{Tr}/y]^\bullet = \varepsilon[\text{Neg}/x,y] - \gamma[\text{Tr}/y]$
$[\text{Tr}/z]^\bullet = \varepsilon[\text{Neg}/y,z] - \gamma[\text{Tr}/z]$

simplifying (N is the quantity of each of the 3 gates)

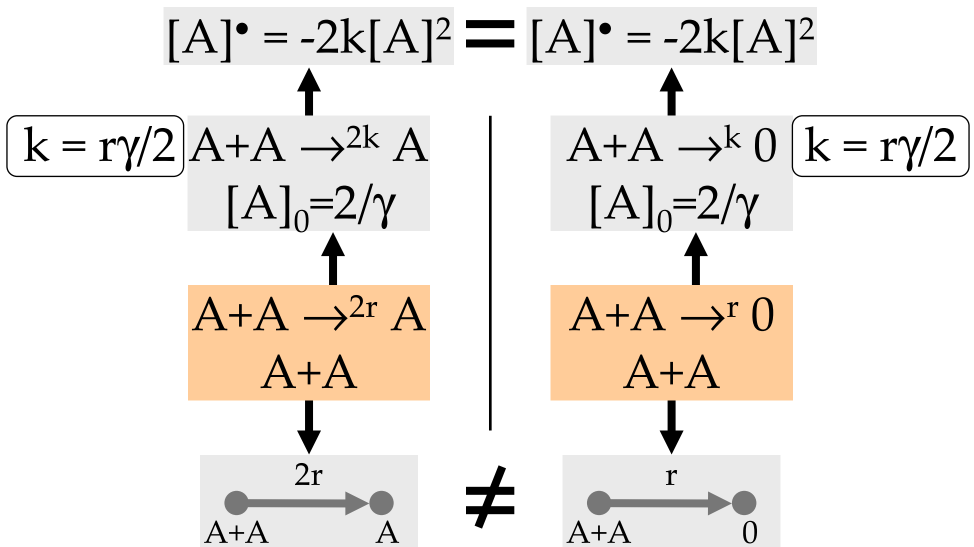
$[\text{Neg}/x,y]^\bullet = \eta N - (\eta + r[\text{Tr}/x])[\text{Neg}/x,y]$
$[\text{Neg}/y,z]^\bullet = \eta N - (\eta + r[\text{Tr}/y])[\text{Neg}/y,z]$
$[\text{Neg}/z,x]^\bullet = \eta N - (\eta + r[\text{Tr}/z])[\text{Neg}/z,x]$
$[\text{Tr}/x]^\bullet = \varepsilon[\text{Neg}/z,x] - \gamma[\text{Tr}/x]$
$[\text{Tr}/y]^\bullet = \varepsilon[\text{Neg}/x,y] - \gamma[\text{Tr}/y]$
$[\text{Tr}/z]^\bullet = \varepsilon[\text{Neg}/y,z] - \gamma[\text{Tr}/z]$



GMA ≠ CME


ODE
=
ODE
Continuous Chemistry
Process Algebra
≠
Discrete Chemistry
CTMC
=
CTMC

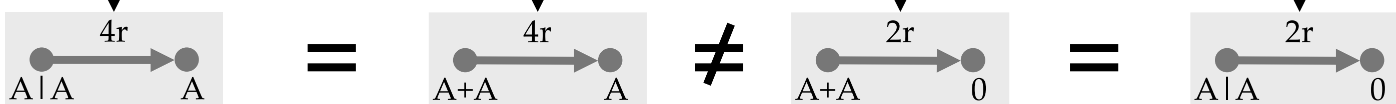
$A+A \rightarrow^{r} 0 \quad =? \quad A+A \rightarrow^{2r} A$

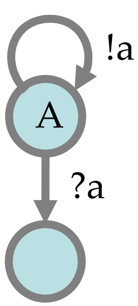


... as Automata

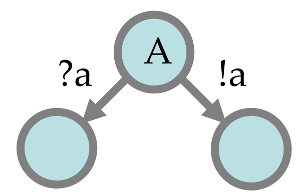
$[A]^\bullet = -2r\gamma[A]^2$ = $[A]^\bullet = -4k[A]^2$ = $[A]^\bullet = -4k[A]^2$ = $[A]^\bullet = -2r\gamma[A]^2$

$A = ?a_{(2r)};0 \oplus !a_{(2r)};A$ $A \mid A$	$k = r\gamma/2$ $A+A \rightarrow^{4k} A$ $[A]_0=2/\gamma$ $A+A \rightarrow^{4r} A$ $A+A$	$A+A \rightarrow^{2k} 0$ $[A]_0=2/\gamma$ $k = r\gamma/2$ $A+A \rightarrow^{2r} 0$ $A+A$	$A = ?a_{(r)};0 \oplus !a_{(r)};0$ $A \mid A$





(a@2r)

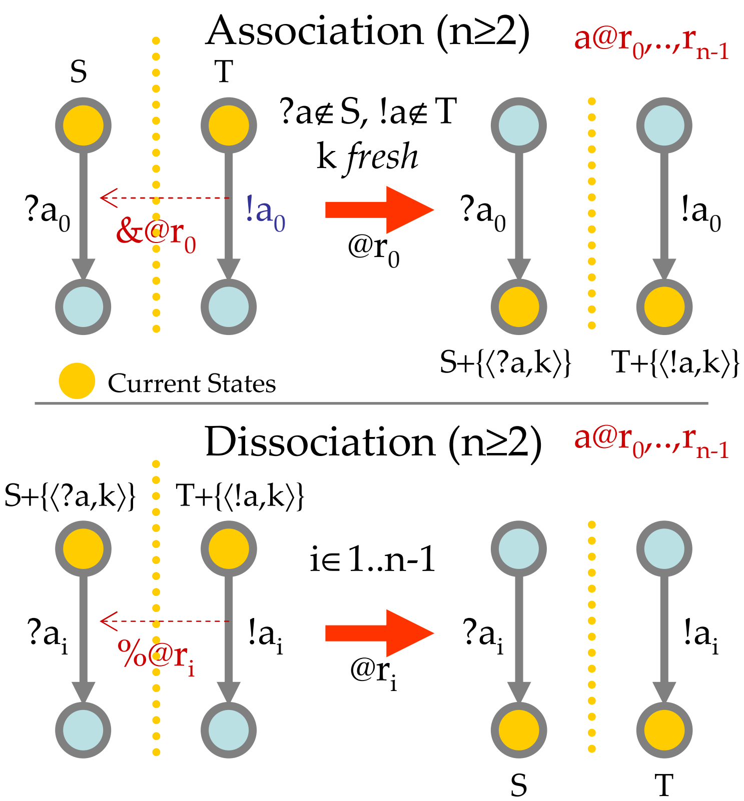


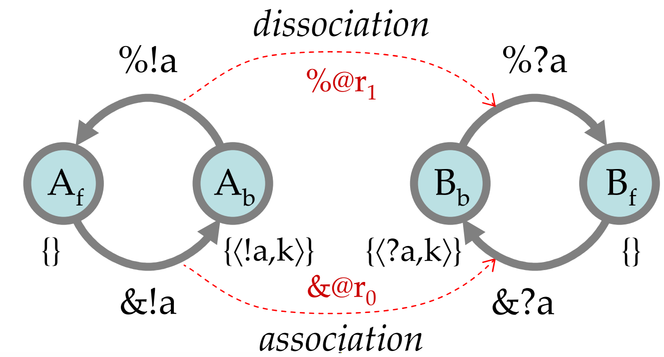
(a@r)

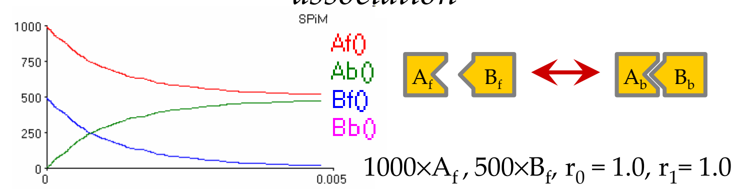
Biochemistry

Interaction+Complexation

Polyautomata



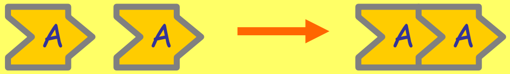




1000×A_f, 500×B_f, r_0 = 1.0, r_1 = 1.0

Can be encoded in π-calculus (and SPiM) by bound-output/bound-input.

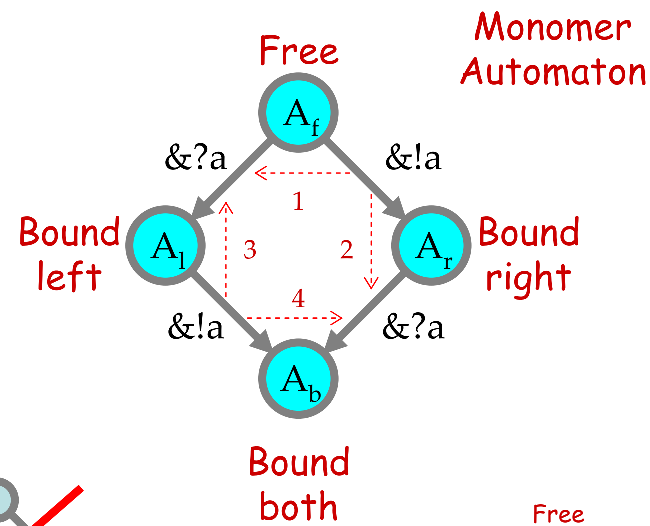
```
directive sample 0.005
directive plot Af(); Ab(); Bf(); Bb()

val mu = 1.0    val lam = 1.0
new a@mu:chan(chan)

let Af() = (new n@lam:chan run !a(n); Ab(n))
and Ab(n:chan) = !n; Af()

let Bf() = ?a(n); Bb(n)
and Bb(n:chan) = ?n; Bf()

run (1000 of Af() | 500 of Bf())
```



Bidirectional

Polymerization

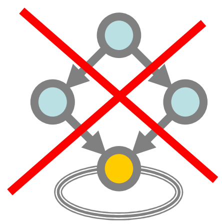
```
new c@μ  new stop@1.0

A_free =
   !c(νrht_λ); A_brht(rht)) +
   ?c(lft); A_blft(lft)

A_blft(lft) =
   !c(νrht_λ); A_bound(lft,rht))

A_brht(rht) =
   ?c(lft); A_bound(lft,rht)

A_bound(lft,rht) = ?stop
```

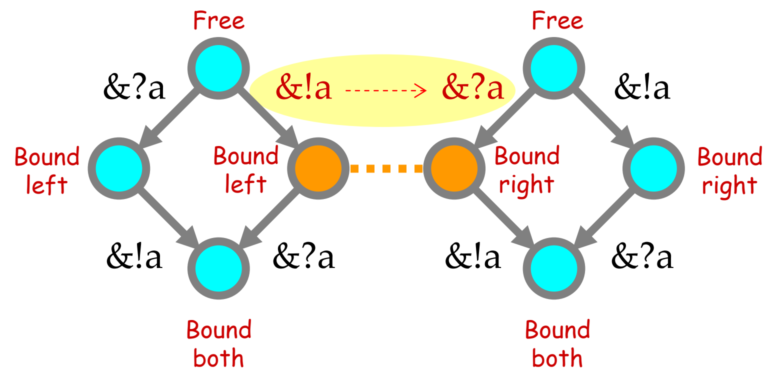


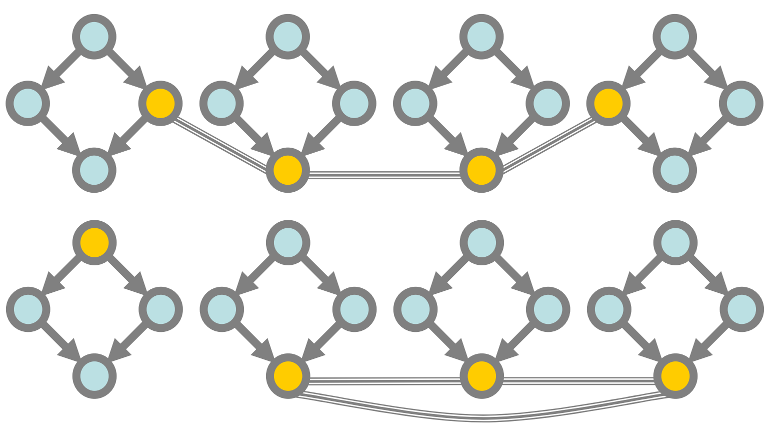
Polymerization is iterated complexation.

Polyautomata

Bound output !c(νr) and input ?c(l) on automata transitions to model complexation







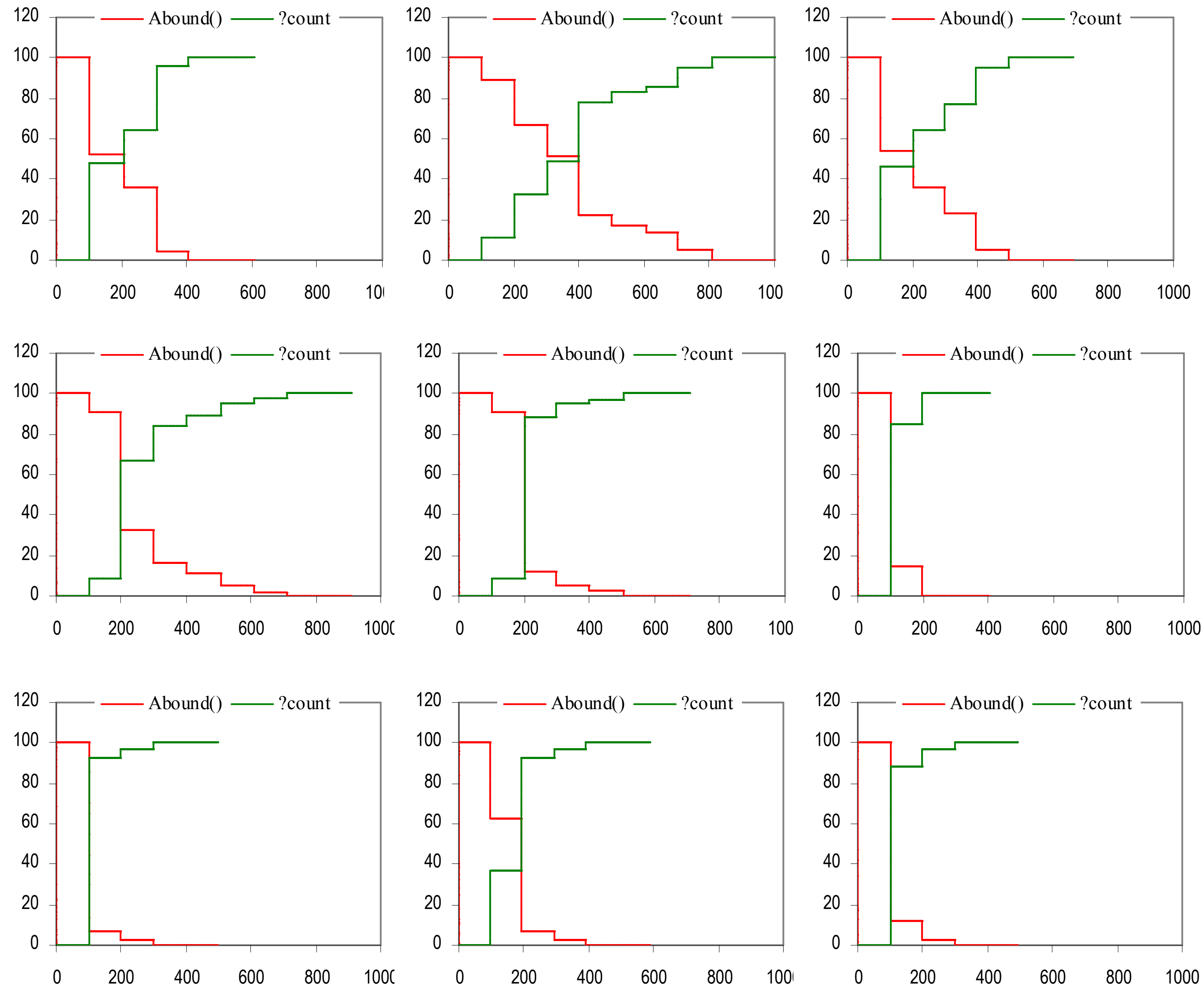
```
directive sample 10000.0
directive plot Afree(); Ablft(); Abrht(); Abound()

val lam = 1.0   val mu = 1.0
new c@mu:chan(chan)  new stop@1.0:chan

let Afree() =
  (new rht@lam:chan run
   do !c(rht); Abrht(rht)
   or ?c(lft); Ablft(lft))

and Ablft(lft:chan) =
  (new rht@lam:chan run
   !c(rht); Abound(lft,rht))

and Abrht(rht:chan) =
   ?c(lft); Abound(lft,rht)

and Abound(lft:chan, rht:chan) =
  ?stop

run (2 of Afree())
```

Bidirectional Polymerization

Circular Polymer Lengths

Scanning and counting the size of the circular polymers (by a cheap trick).

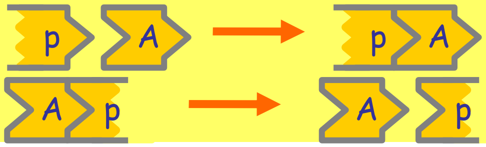
Polymer formation is complete within 10t; then a different polymer is scanned every 100t.




```
directive sample 1000.0
directive plot Abound(); ?count

type Link = chan(chan)
type Barb = chan

val lam = 1000.0 (* set high for better counting *)
val mu = 1.0
new c@mu:chan(Link)
new enter@lam:chan(Barb)
new count@lam:Barb

let Afree() =
  (new rht@lam:Link run
   do !c(rht); Abrht(rht)
   or ?c(lft); Ablft(lft))

and Ablft(lft:Link) =
  (new rht@lam:Link run
   !c(rht); Abound(lft,rht))

and Abrht(rht:Link) =
   ?c(lft); Abound(lft,rht)

and Abound(lft:Link, rht:Link) =
  do ?enter(barb); (?barb | !rht(barb))
  or ?lft(barb); (?barb | !rht(barb))
(* each Abound waits for a barb, exhibits it, and passes it to
   the right so we can plot number of Abound in a ring *)

let clock(t:float, tick:chan) =      (* sends a tick every t time *)
  (val ti = t/1000.0 val d = 1.0/ti
   let step(n:int) =
     if n<=0 then !tick; clock(t,tick) else delay@d; step(n-1)
   run step(1000))

new tick:chan
let Scan() = ?tick; !enter(count); Scan()

run 100 of Afree()
run (clock(100.0, tick) | Scan())
```

100xA_{free}, initially.

The height of each rising step is the size of a separate circular polymer. (Unbiased sample of nine consecutive runs.)



Actin-like
Poly/Depolymerization

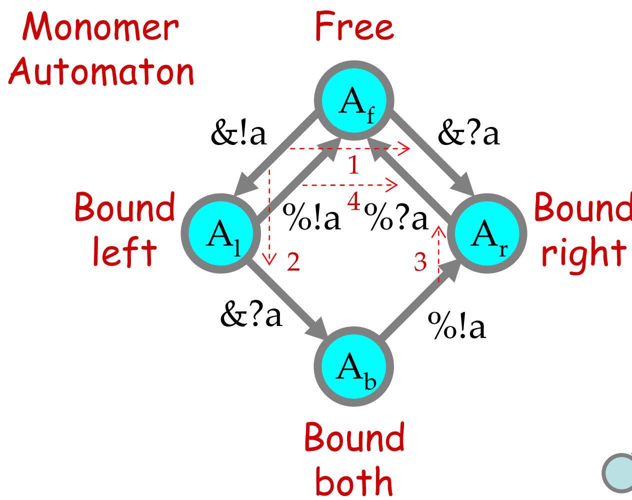
new c@μ

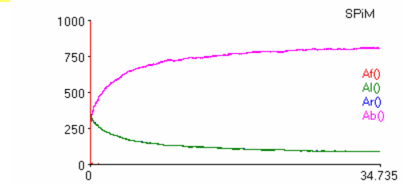
A_{free} =
  $!c(^{\nu}lft_{\lambda})$; $A_{blft}(lft)$ +
  $?c(rht)$; $A_{brht}(rht)$

$A_{blft}(lft)$ =
  $!!lft$; A_{free} +
  $?c(rht)$; $A_{bound}(lft,rht)$

$A_{brht}(rht)$ =
  $?rht$; A_{free}

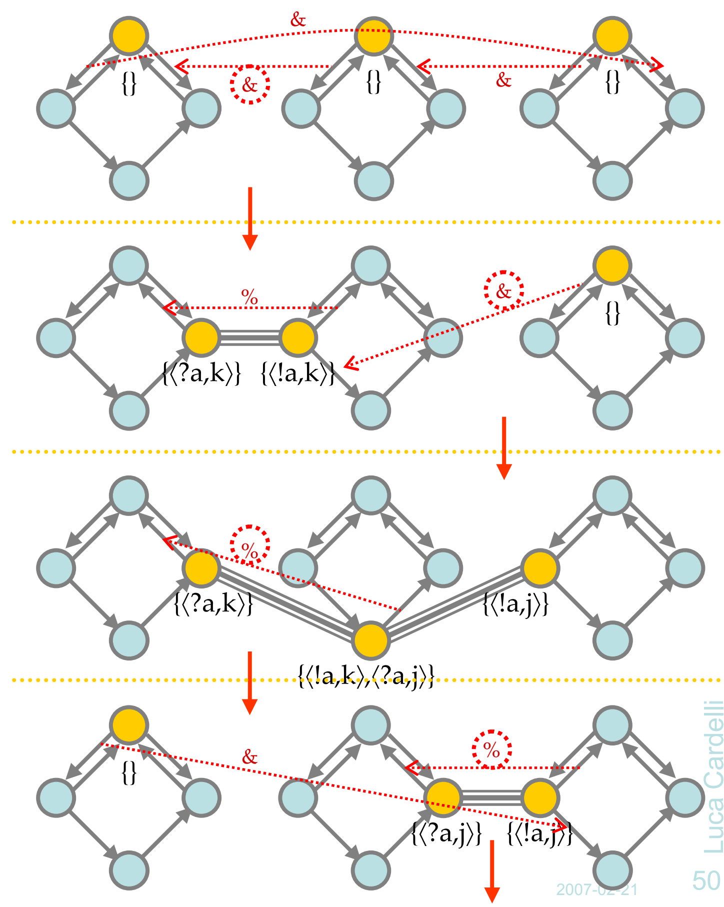
$A_{bound}(lft,rht)$ =
  $!!lft$; $A_{brht}(rht)$





1000 monomers settle to
~100 polymers of size ~10

```
directive sample 1000.0
directive plot Af(); Al(); Ar(); Ab()

val lam = 1.0 (* dissoc *)
val mu = 1.0 (* assoc *)
new c@mu:chan(chan)

let Af() =
  (new lft@lam:chan run
   do !c(lft); Al(lft)
   or ?c(rht); Ar(rht))

and Al(lft:chan) =
   do !!lft; Af()
   or ?c(rht); Ab(lft,rht)

and Ar(rht:chan) =
   ?rht; Af()

and Ab(lft:chan, rht:chan) =
   !!lft; Ar(rht)

run 1000 of Af()
```



Conclusions

Conclusions

- Compositional Models
  - Accurate (at the "appropriate" abstraction level).
  - Manageable (so we can scale them up by composition).
- Interacting Automata
  - Complex global behavior from simple components.
  - Bridging individual and collective behavior.
  - Connections to classical Markov theory, chemical Master Equation, and Rate Equation.
- Mapping out "the whole system"
  - Through an "artificial biochemistry" (a scalable mathematical and computational modeling framework) to investigate "real biochemistry" on a large scale.

http://LucaCardelli.name