Artificial Biochemistry

Luca Cardelli

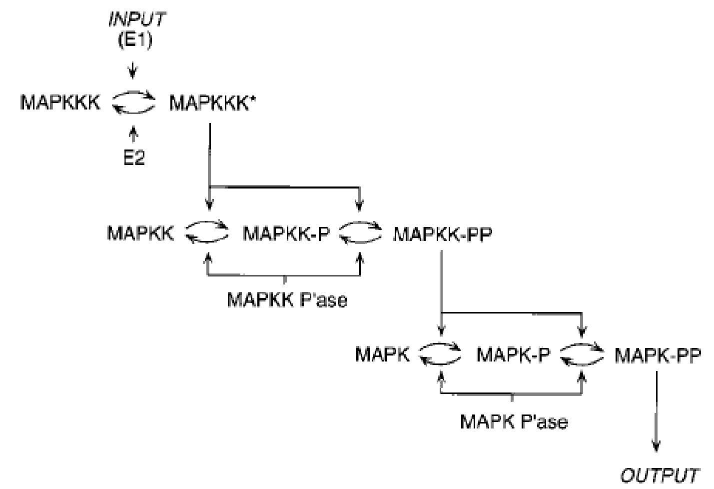
Microsoft Research

BIOWIRE 2007
Cambridge 2007-04-03

http://LucaCardelli.name

Intro

Ultrasensitivity in the mitogen-activated protein **cascade**, Chi-Ying F. Huang and James E. Ferrell, Jr., 1996, *Proc. Natl. Acad. Sci. USA*, 93, 10078-10083.



- Understanding how cells compute
  - How do signaling networks work?
  - Much is understood, and much is not.
- *An unusual computational paradigm*
  - By protein interactions (mostly)
  - Is it related to:
    - Electronic circuits?
    - Automata?
    - Process Algebra?
- Why study signaling networks?
  - It's "just chemistry", we should be able to cope with it.
    - Simpler than gene networks, neural networks, ants, and bees!
  - Yet non-trivial; general principles and algorithms may apply.

Stochastic Collectives

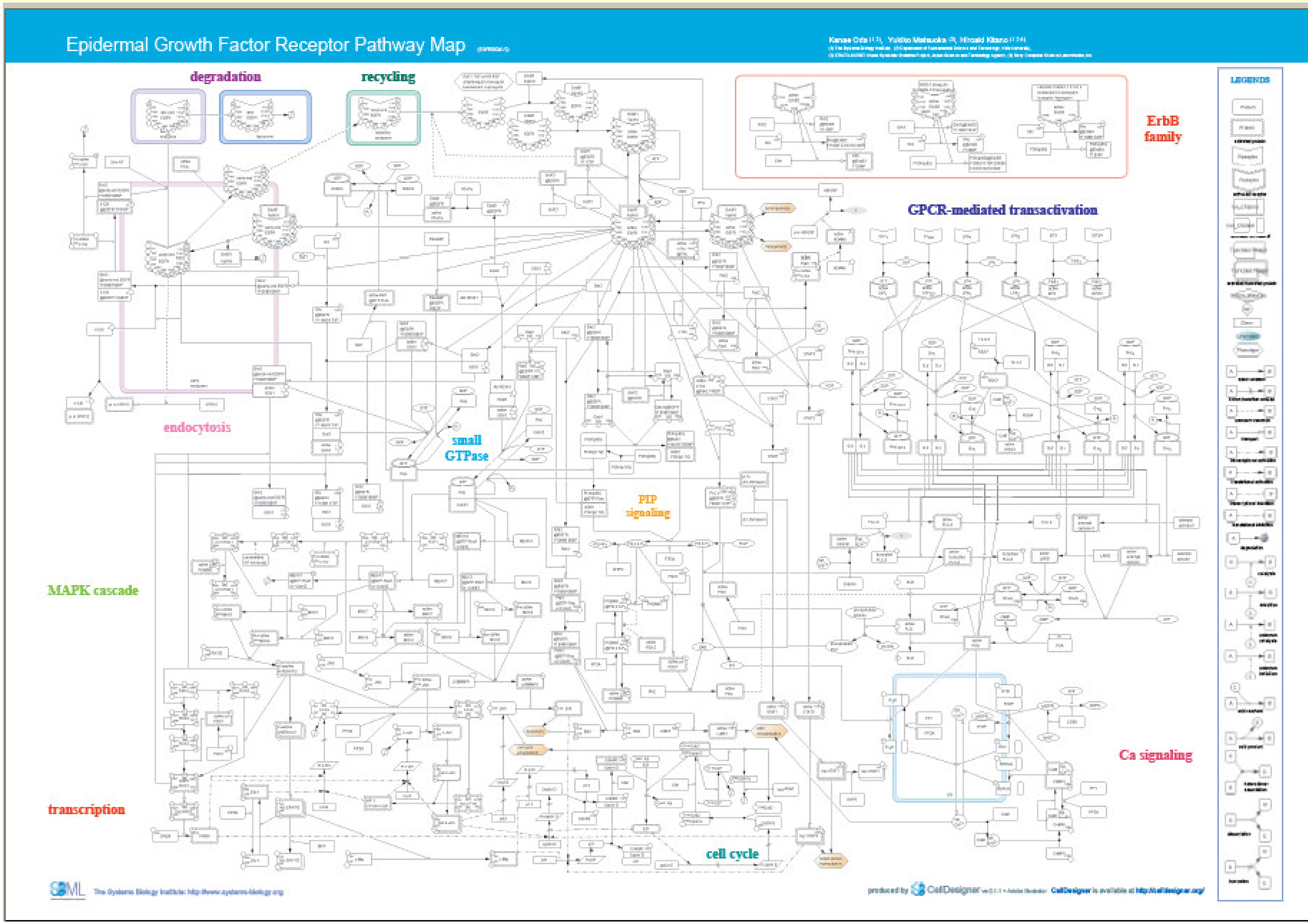
Computing by Stochastic Collectives

- "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

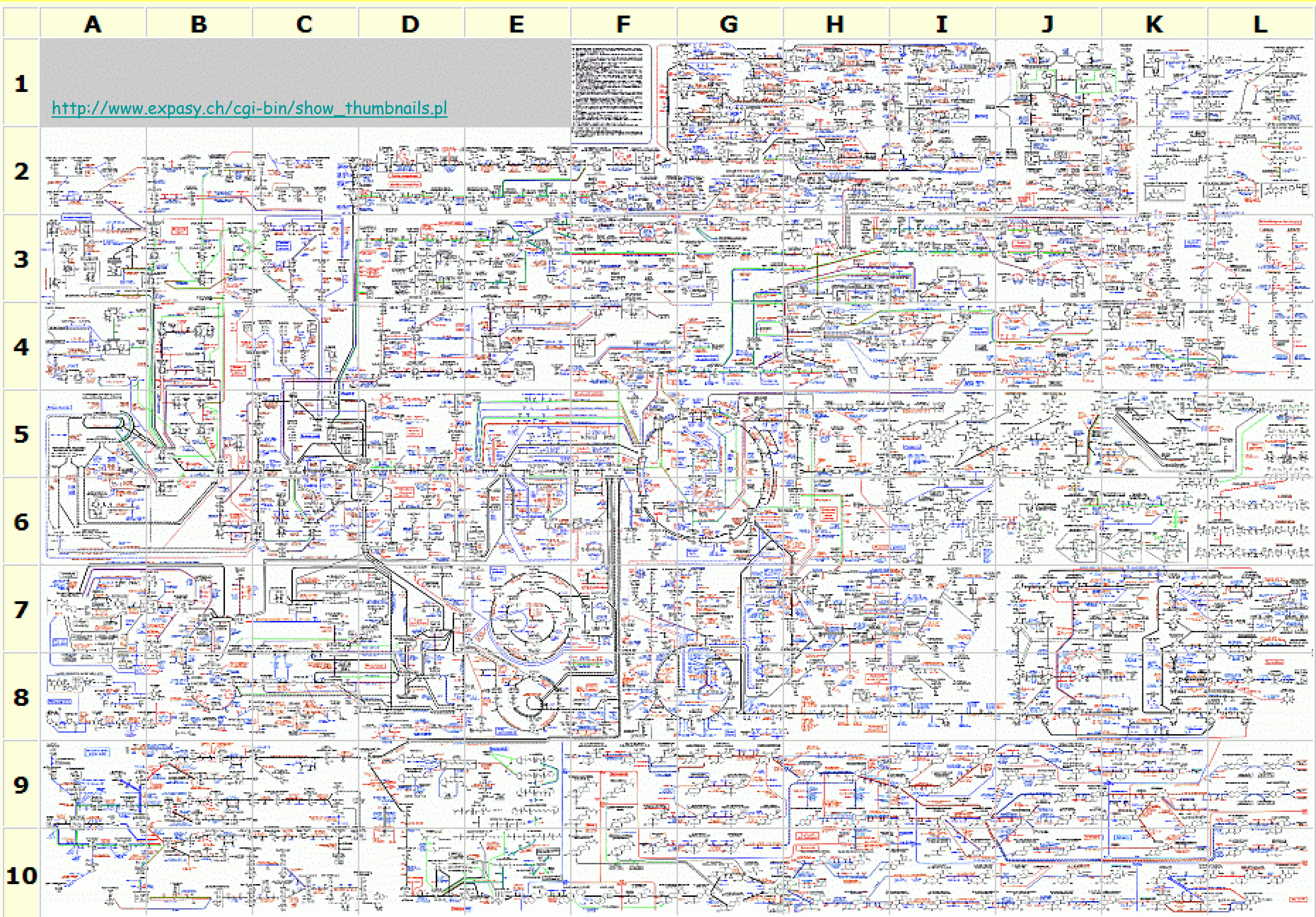
- "Stochastic":
  - Interactions have *rates*
    - Not quite discrete (hundreds or thousands of components)
    - Not quite continuous (non-trivial stochastic effects)
    - Not quite hybrid (no "switching" between regimes)

- 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].

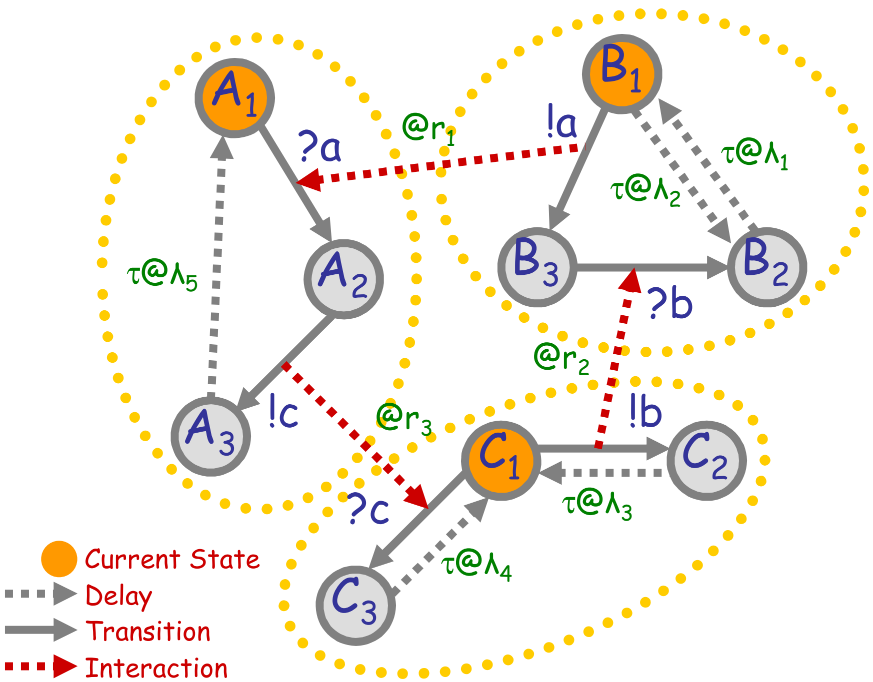
State Transitions



Compositionality (NOT!)



Interacting Automata



Communicating automata: a graphical FSA-like notation for "finite state restriction-free π-calculus processes".

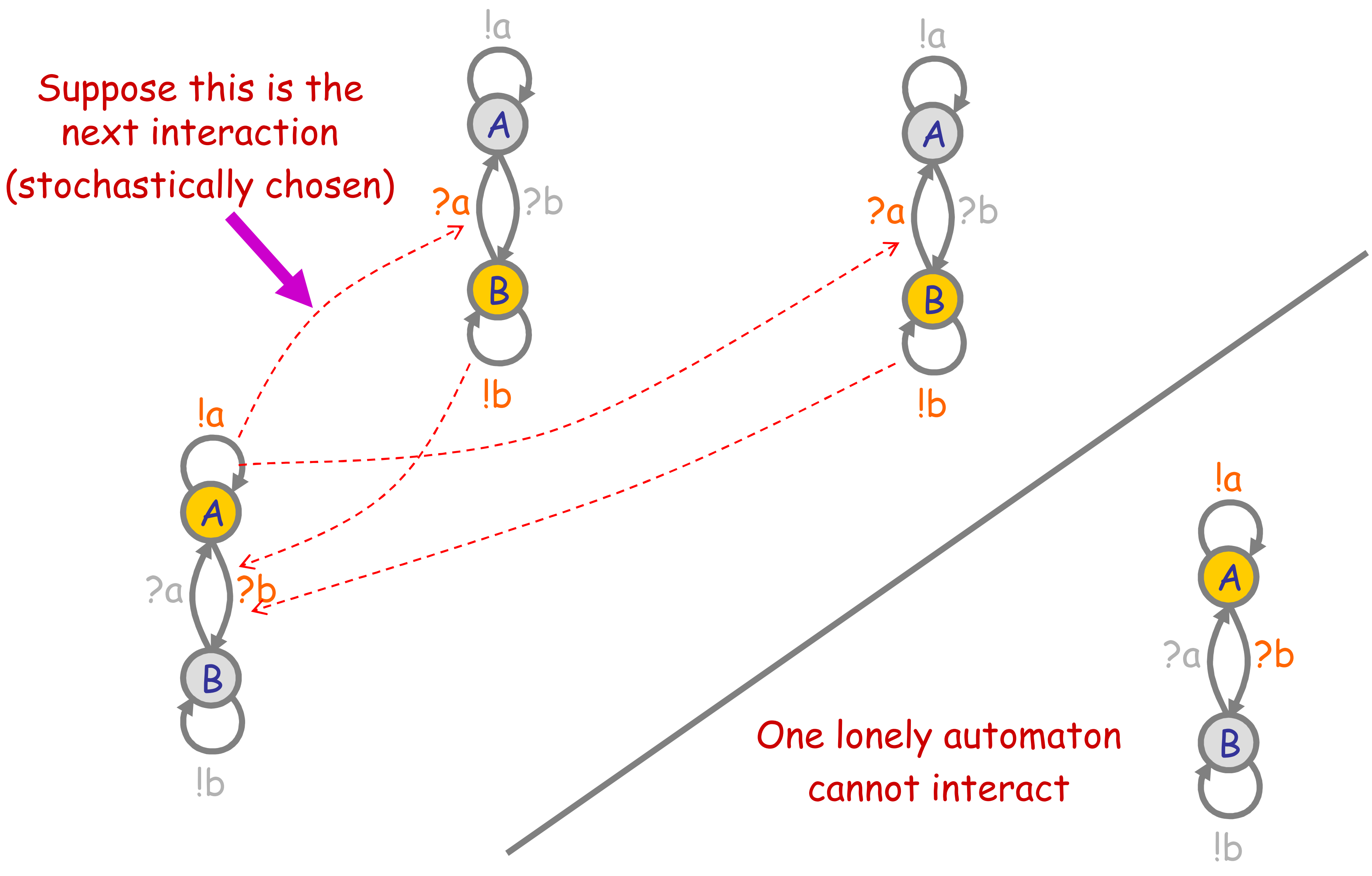
Interacting automata do not even exchange values on communication.

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

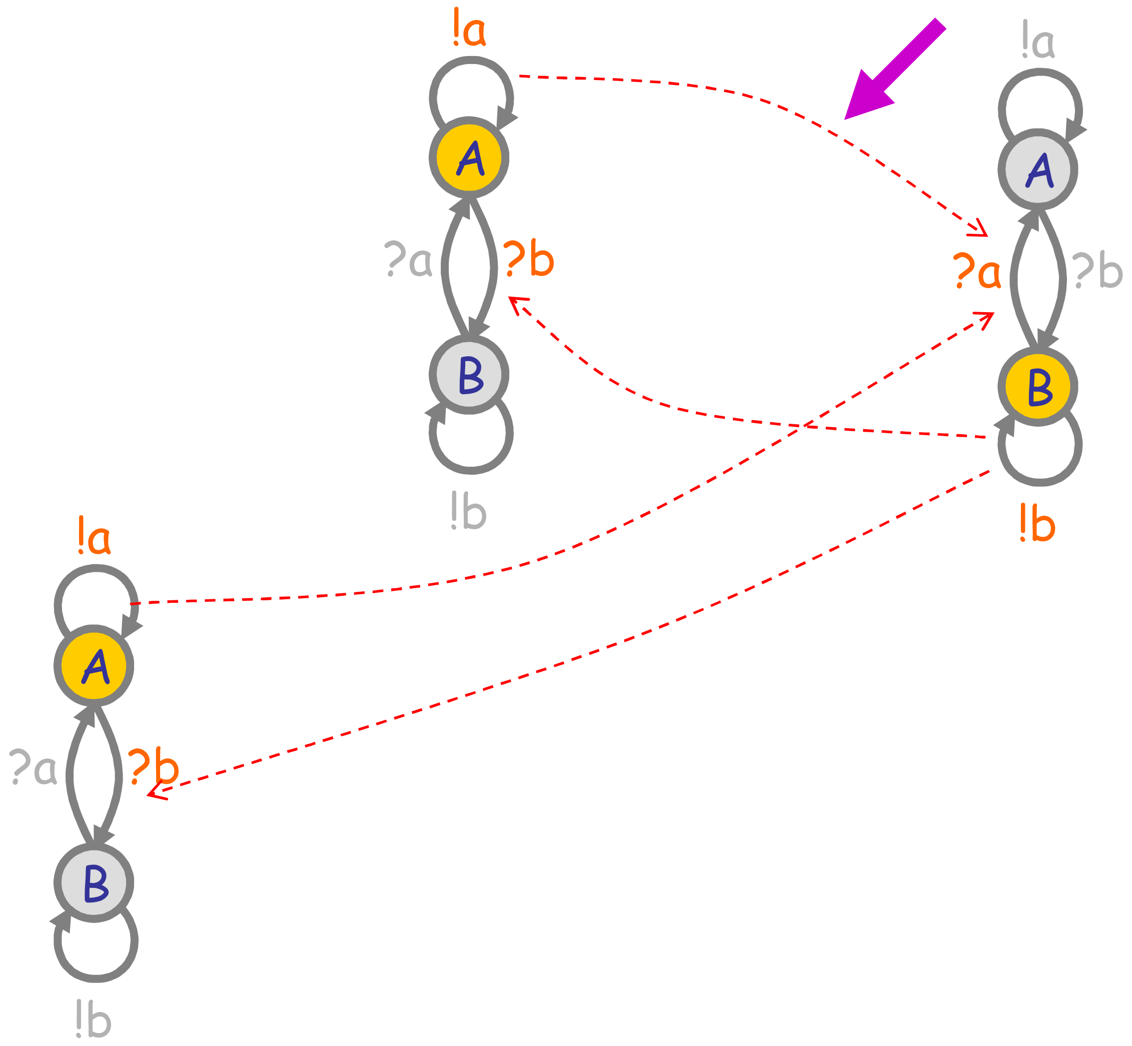
"Finite state" means: no composition or restriction inside recursion.

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

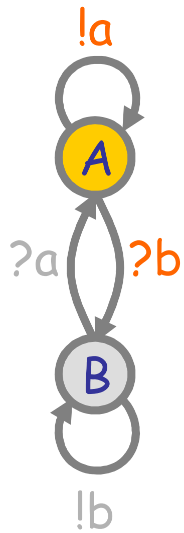
Interactions in a Population

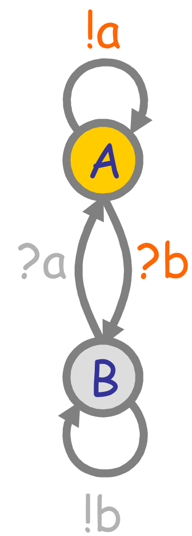


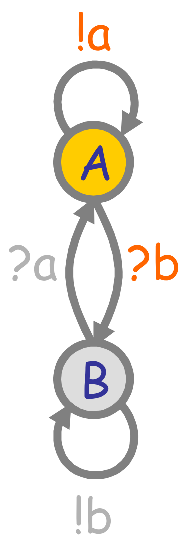
Interactions in a Population



Interactions in a Population

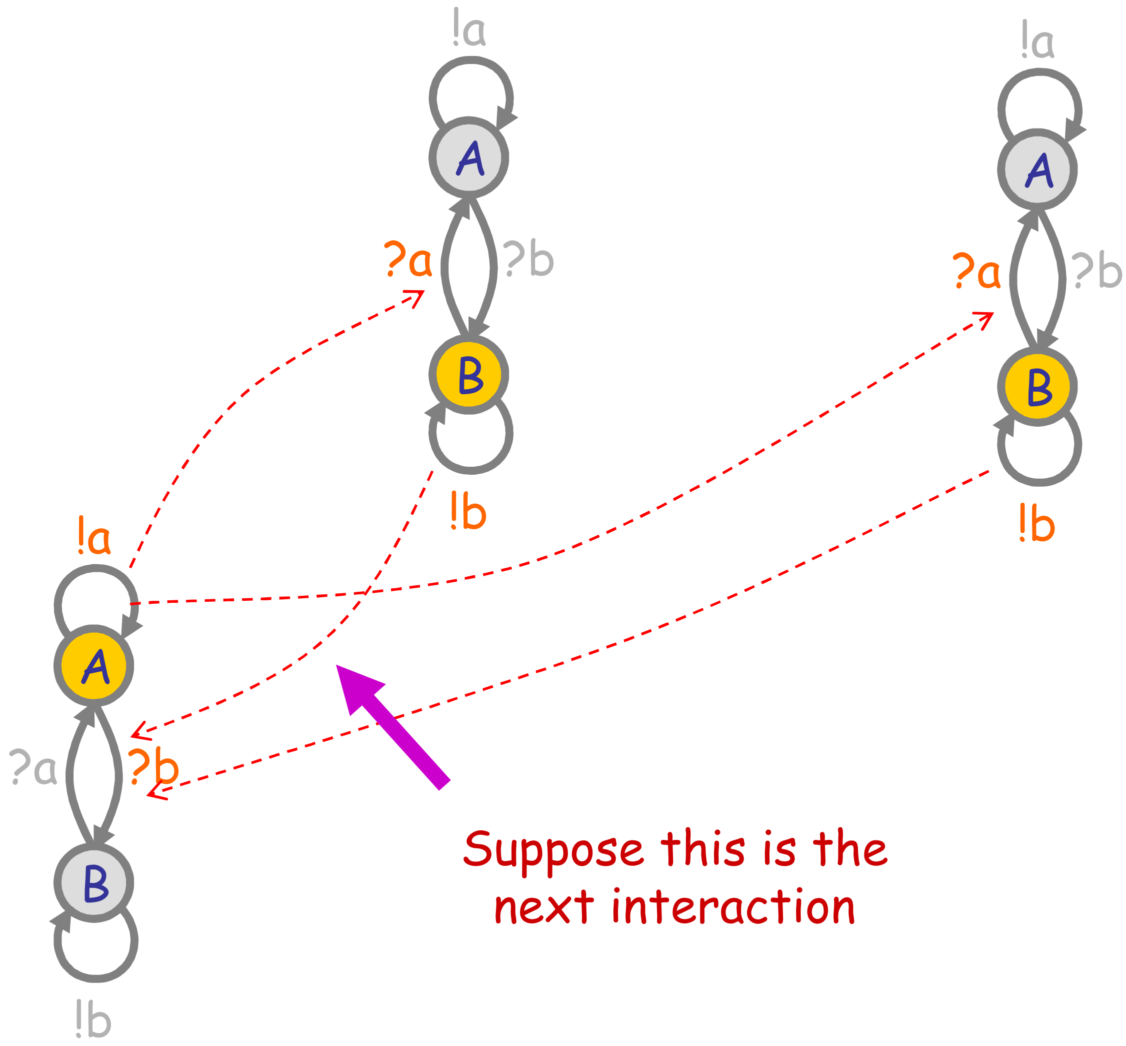






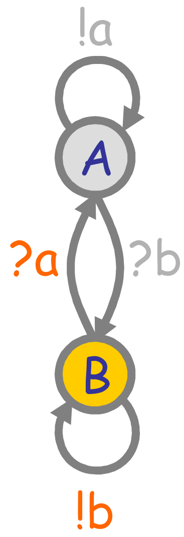
All-A stable population

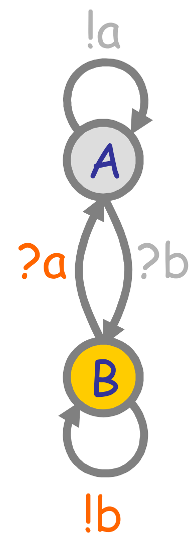
Interactions in a Population (2)

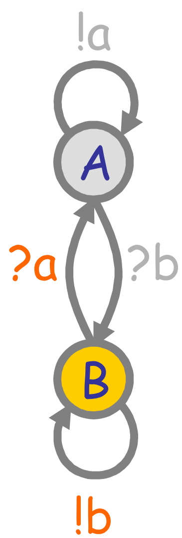


Luca Cardelli

Interactions in a Population (2)





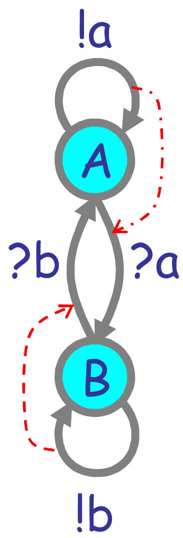


All-B stable
population

Nondeterministic
population behavior
("multistability")

Groupies and Celebrities

Celebrity
(does not want to be like somebody else)

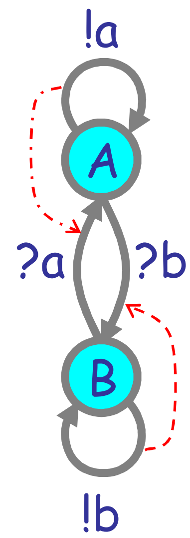



```
directive sample 0.1 200
directive plot A(); B()

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)

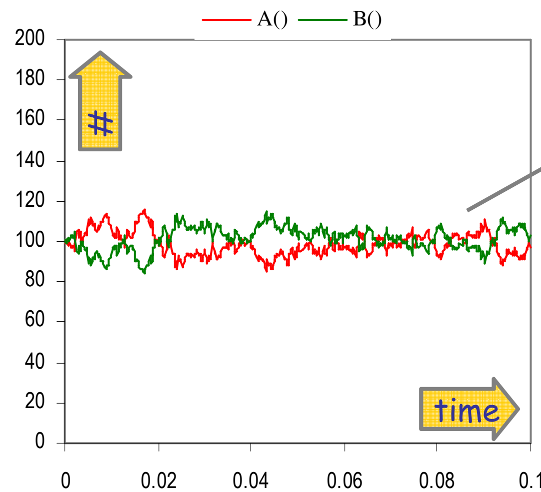



```
directive sample 0.1 200
directive plot A(); B()

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())
```

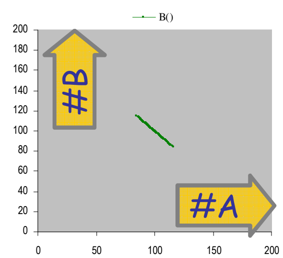
a@1.0

b@1.0

A stochastic collective of celebrities:

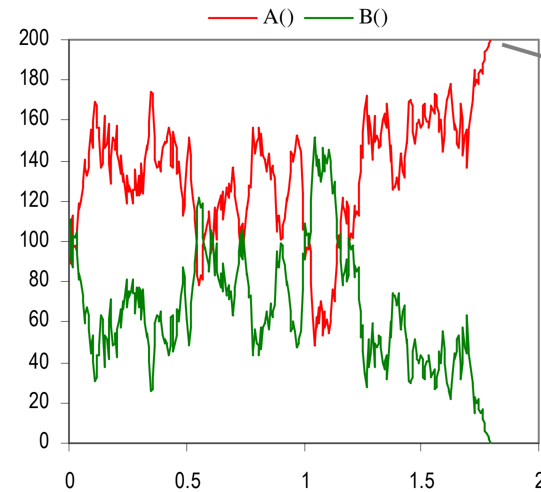


equilibrium

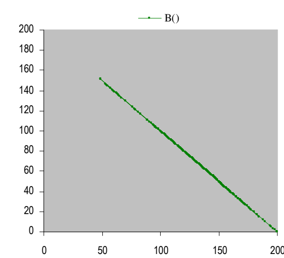


Stable because as soon as a A finds itself in the majority, it is more likely to find somebody in the same state, and hence change, so the majority is weakened.

A stochastic collective of groupies:



always eventually deadlock

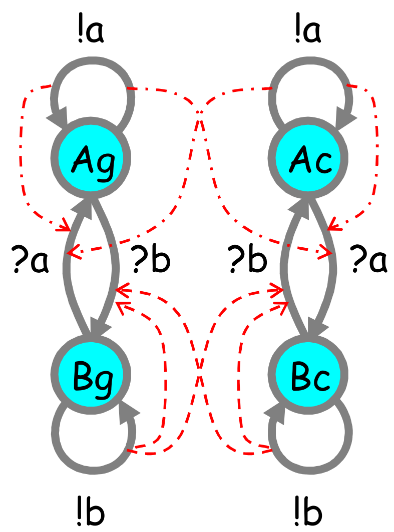


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

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

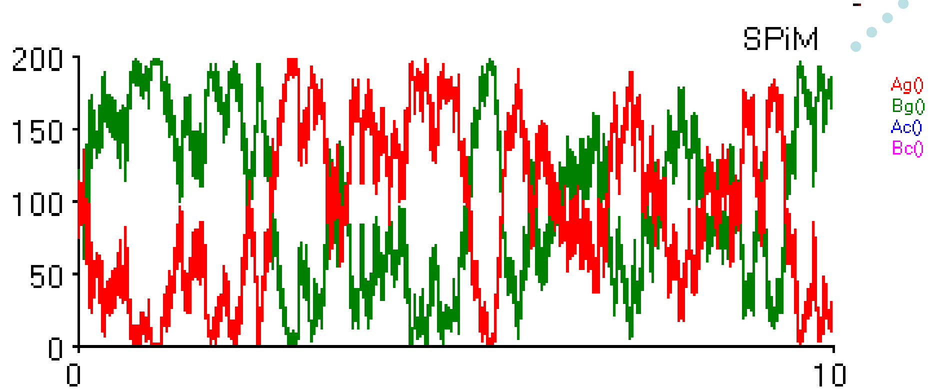
Many Groupies



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

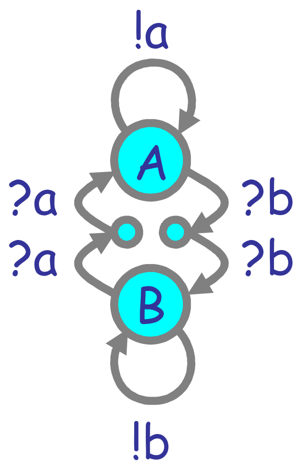


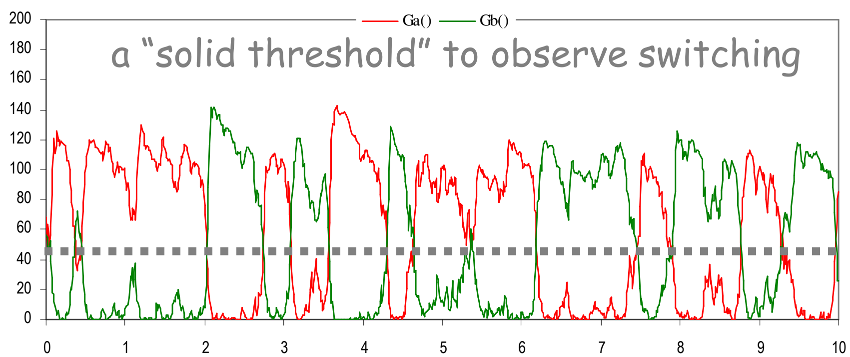
A tiny bit of "noise" can make a huge difference

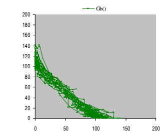
Hysteric Groupies

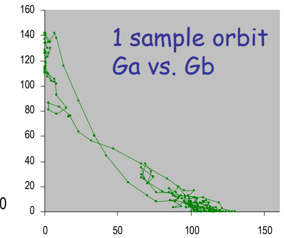
Regularity can arise not far from chaos

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

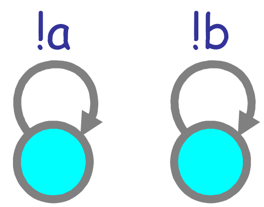








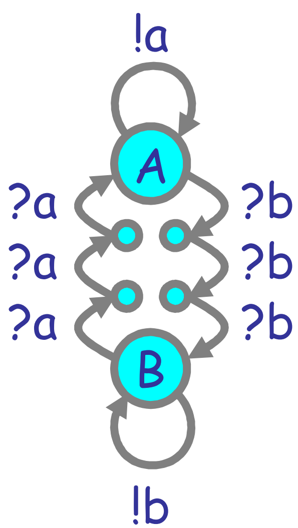

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

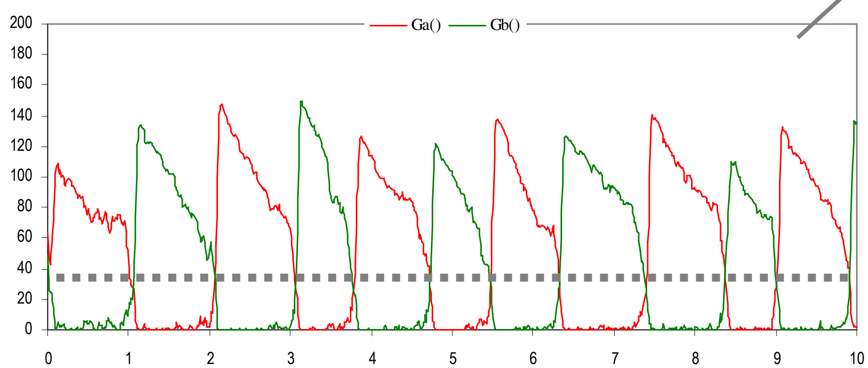


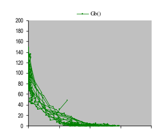
(With doping to break deadlocks)

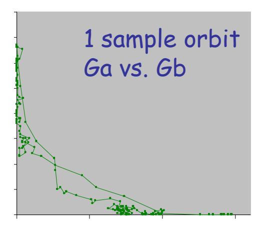
N.B.: It will not oscillate without doping (noise)

"regular" oscillation





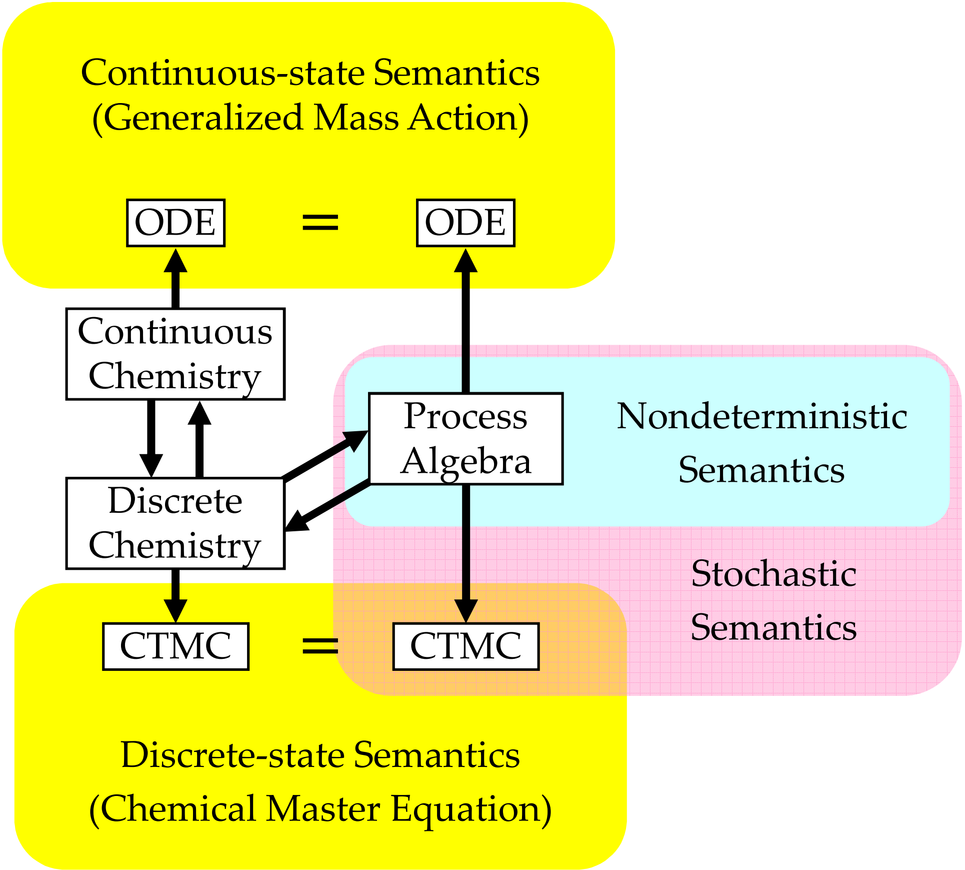





```
directive sample 10.0 1000
directive plot Ga(); Gb()

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

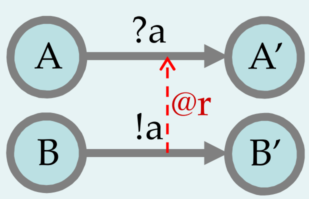
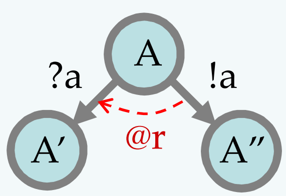
The Two Faces of Chemistry



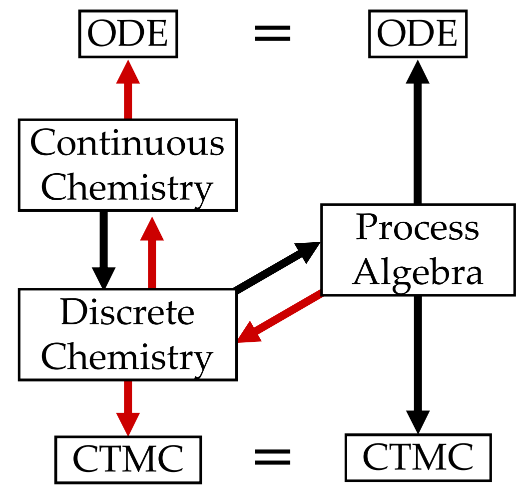
These diagrams commute via appropriate maps.

L. Cardelli: "On Process Rate Semantics"

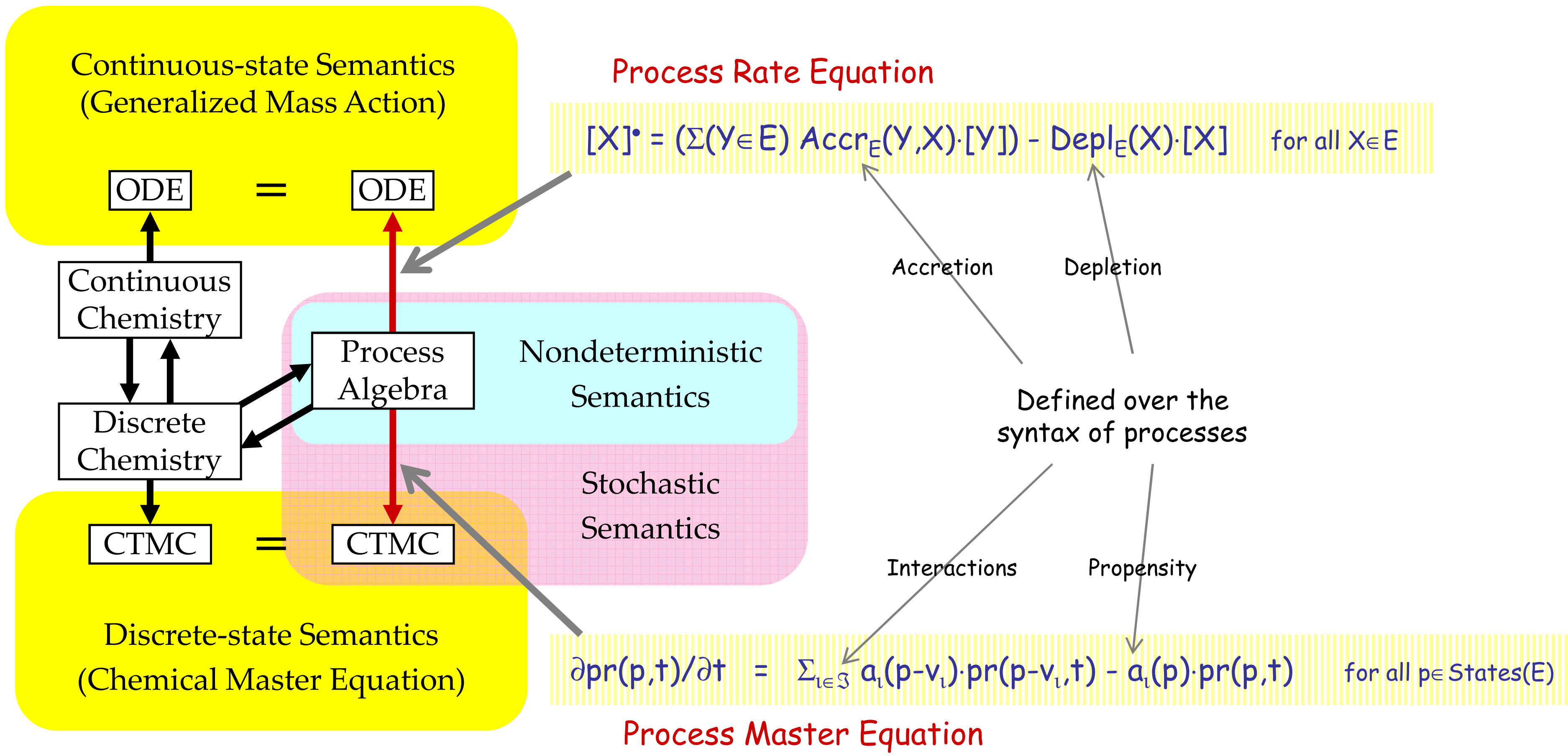
From Automata to Chemistry

Automata →	Discrete Chemistry →	Continuous Chemistry $\gamma = N_A V$
initial states A ǀ A ǀ ... ǀ A	initial quantities $\#A_0$	initial concentrations $[A]_0$ with $[A]_0 = \#A_0/\gamma$
	$A \overset{r}{\dashrightarrow} A'$	$A \xrightarrow{k} A'$ with $k = r$
	$A+B \overset{r}{\dashrightarrow} A'+B'$	$A+B \xrightarrow{k} A'+B'$ with $k = r\gamma$
	$A+A \overset{2r}{\dashrightarrow} A'+A''$	$A+A \xrightarrow{2k} A'+A''$ with $k = r\gamma/2$
	↓ CTMC	↓ ODE

Think $\gamma = 1$
i.e. $V = 1/N_A$

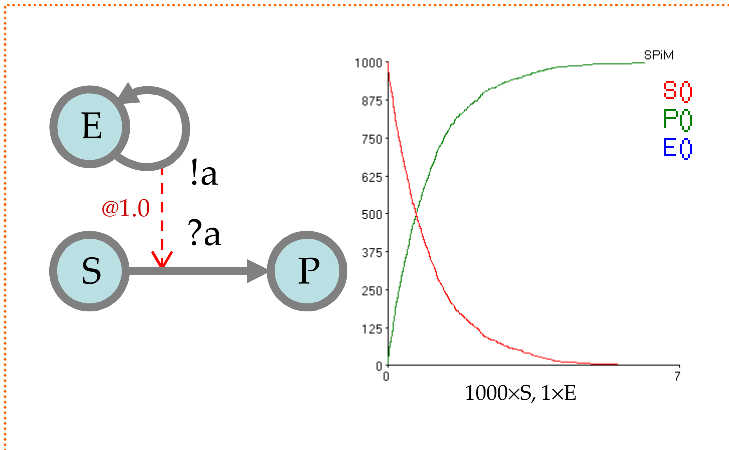


Quantitative Process Semantics



Further Examples

Second-order and Zero-order Regime

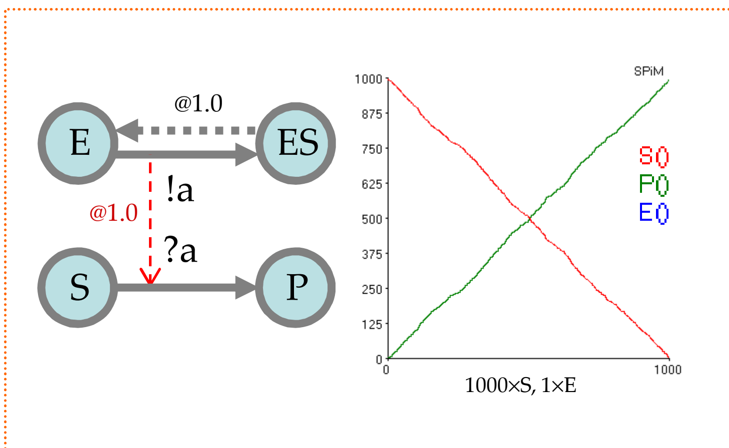


$E+S \rightarrow^{r} E+P$

```
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

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



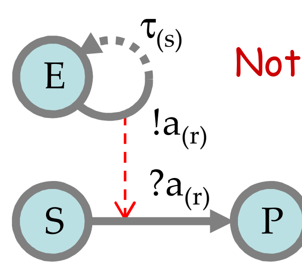
$E+S \rightarrow^{r} ES+P$

$ES \rightarrow^{s} E$

```
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())
```

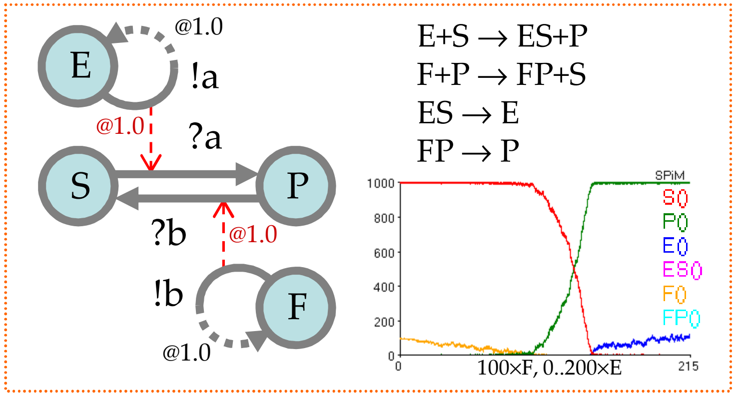
Zero-Order Regime

$[S]^{\bullet} \cong -1$ (by assuming $[ES]^{\bullet} = 0$)



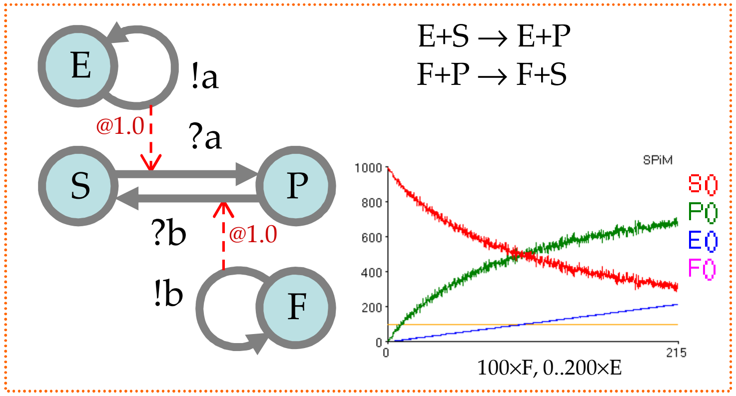
Notation

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)
```

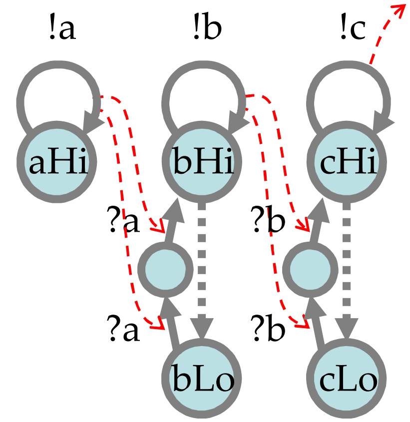
Zero-Order Regime
A small E-F inbalance causes
a much larger S-P switch.

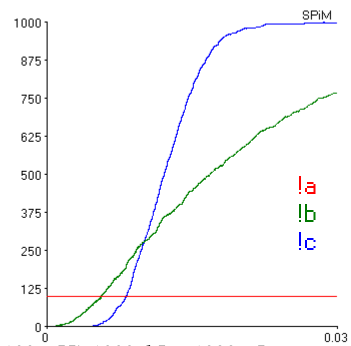



```
directive sample 215.0 1000
directive plot S(); P(); E(); F()

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



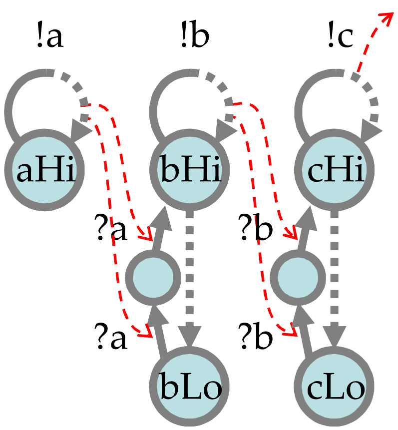


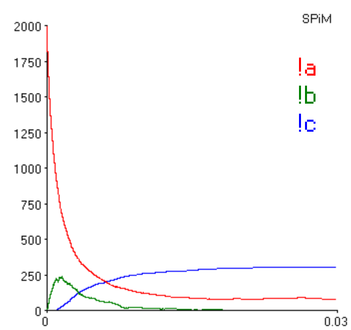
100×aHi, 1000×bLo, 1000×cLo, rates=1.0

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()
```





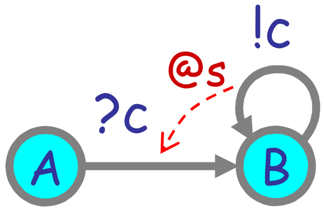
2000×aHi, 1000×bLo, 1000×cLo, rates=1.0

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()
```

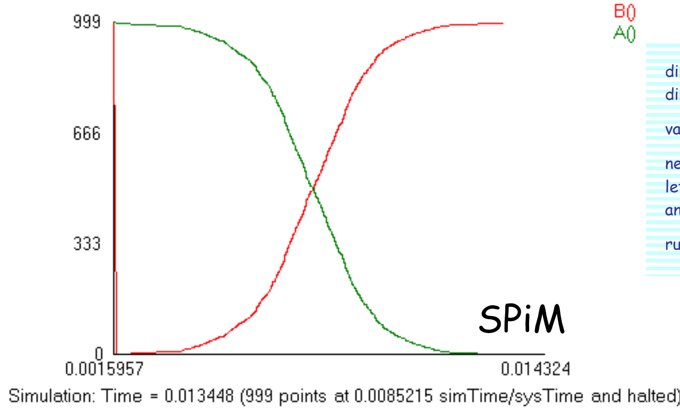
Nonlinear Transition (NLT)



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

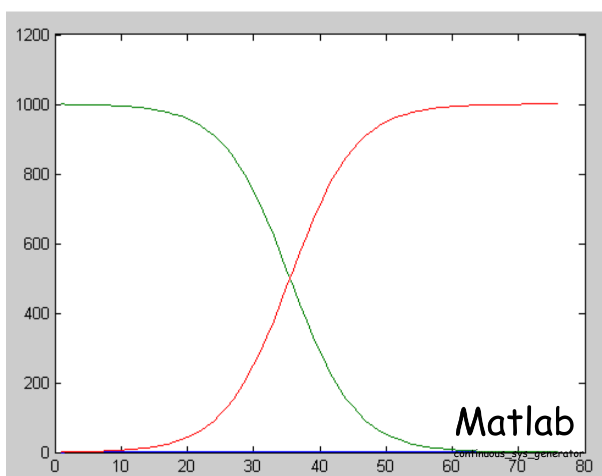
$A+B \rightarrow^{s} B+B$

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



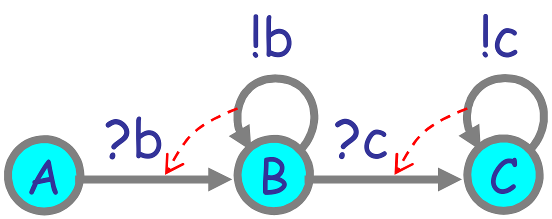

```
directive sample 0.02 1000
directive plot B(); A()

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



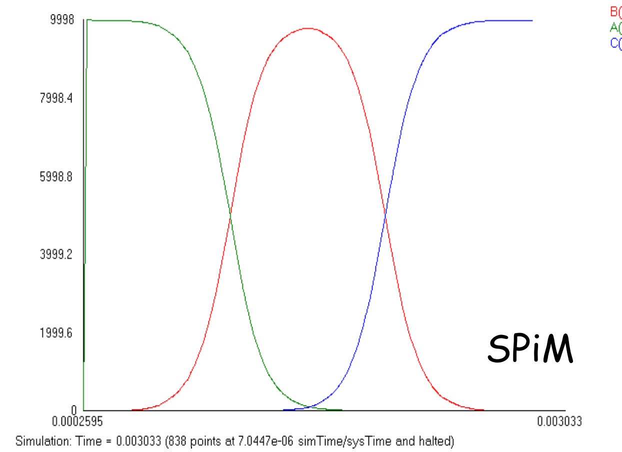
$[B]^\bullet = [B]([A]-[C])$

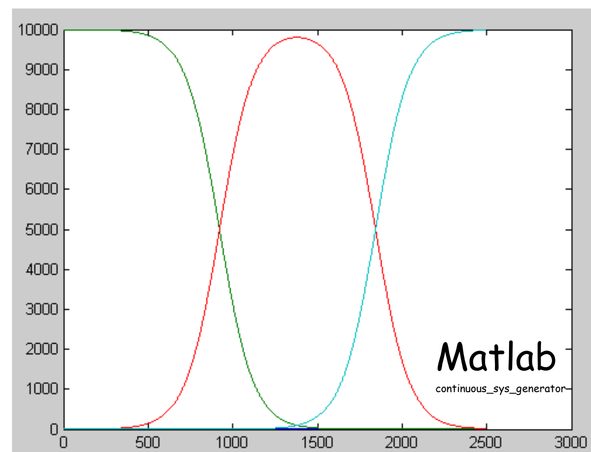
$A = ?b_{(1)};B$
$B = !b_{(1)};B \oplus ?c_{(1)};C$
$C = !c_{(1)};C$

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

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

```
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())
```

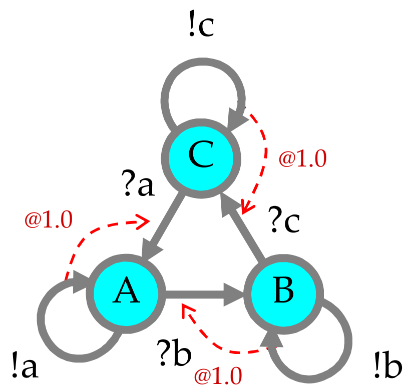


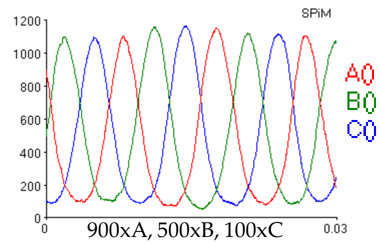


interval/step [0:0.000001:0.0025]

(A)	dx1/dt = -x1*x2	10000.0
(B)	dx2/dt = x1*x2 - x2*x3	1.0
(C)	dx3/dt = x2*x3	1.0

NLT in a Cycle: Oscillator



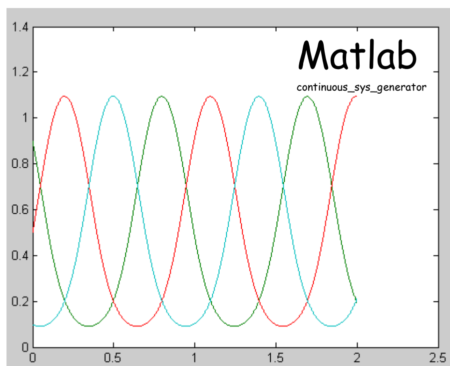



```
directive sample 0.03 1000
directive plot A(); B(); C()

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())
```

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

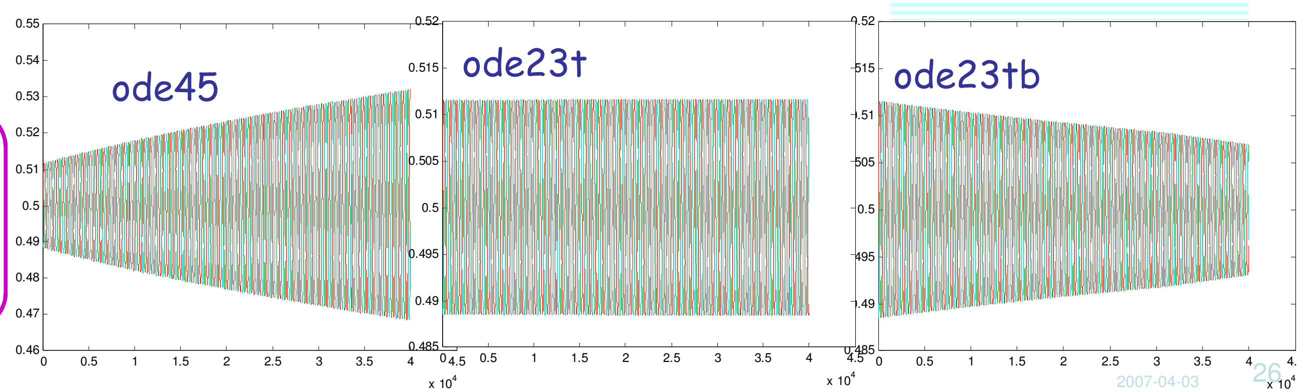
$A+B \to^{s} B+B$
$B+C \to^{s} C+C$
$C+A \to^{s} A+A$

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

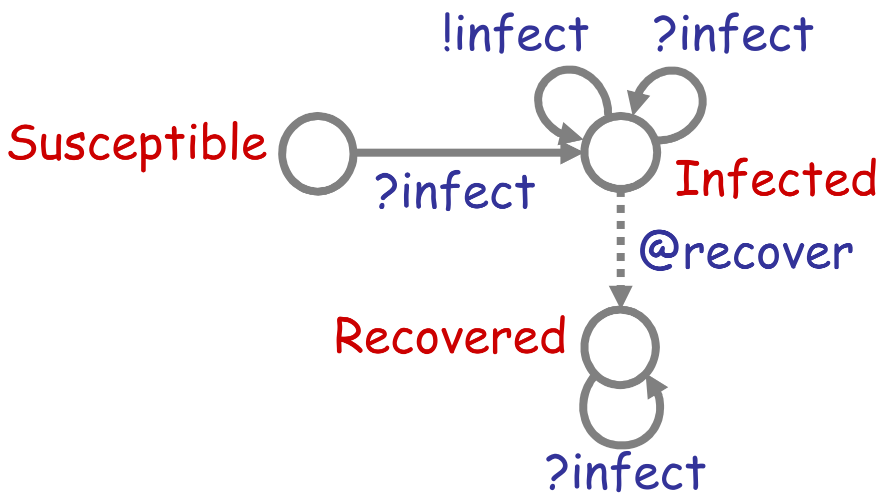


interval/step [0:0.001:20.0]		
(A)	dx1/dt = - x1*x2 + x3*x1	0.9
(B)	dx2/dt = - x2*x3 + x1*x2	0.5
(C)	dx3/dt = - x3*x1 + x2*x3	0.1

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



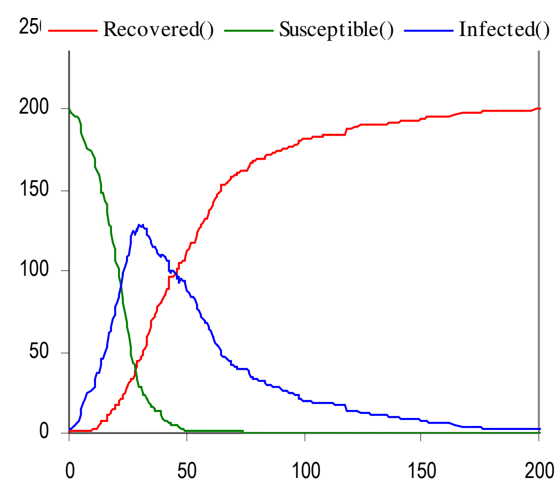

```
directive sample 500.0 1000
directive plot Recovered(); Susceptible(); Infected()

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!

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

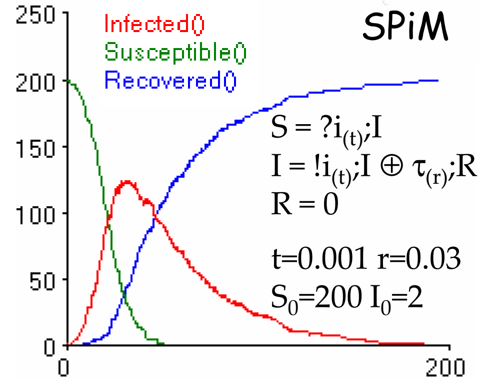
"useless" reactions

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

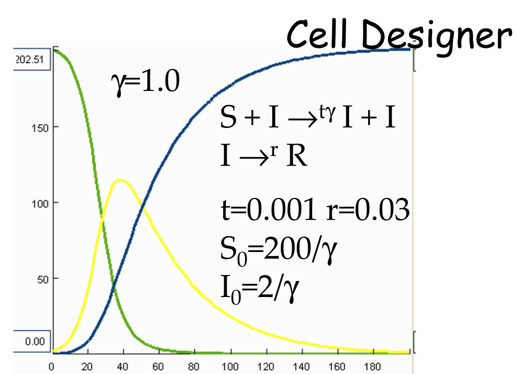
Automata produce the standard ODEs!

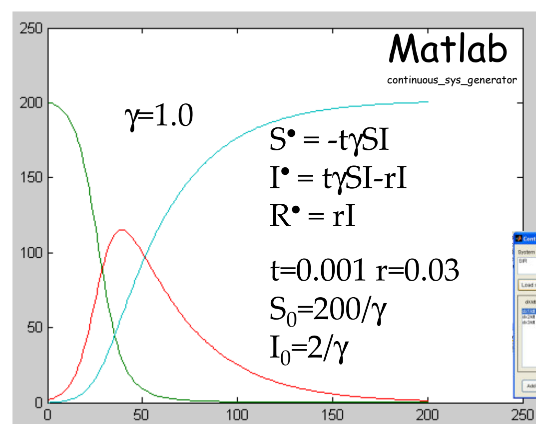
$$\frac{dS}{dt} = -aIS$$
$$\frac{dI}{dt} = aIS - bI$$
$$\frac{dR}{dt} = bI$$

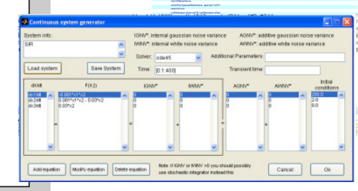
(the Kermack-McKendrick, or SIR model)




```
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())
```







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.

- An “artificial biochemistry”
  - A scalable mathematical and computational modeling framework.
  - To investigate “real biochemistry” on a large scale.

http://LucaCardelli.name

Q?