Artificial Biochemistry

Luca Cardelli

Microsoft Research

Newcastle 2007-05-15

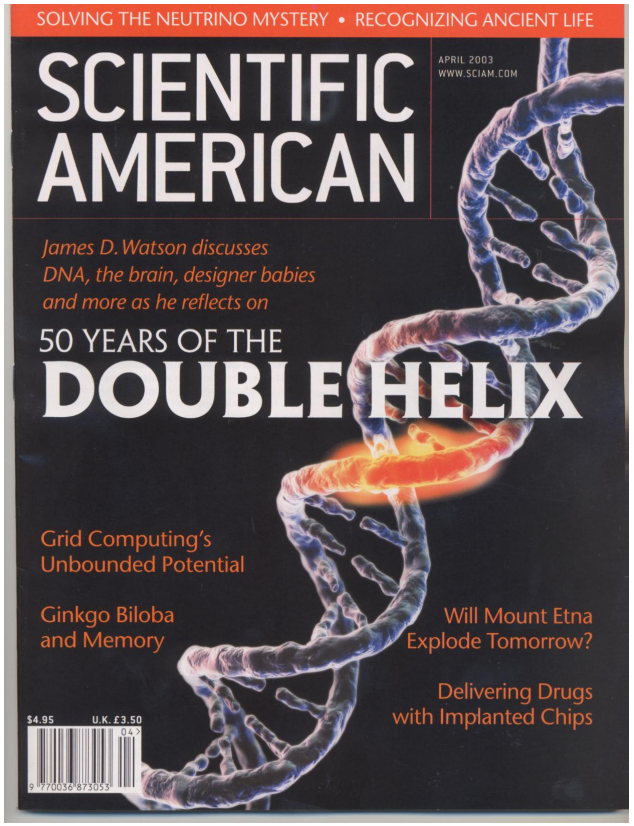
http://LucaCardelli.name

50 Years of Molecular Cell Biology

- The genome (3.2 GBases) is made of DNA
  - Stores digital information as sequences of 4 different nucleotides
  - Directs protein assembly through RNA and the Genetic Code

- Proteins (1M coded from 25K genes) are made of amino acids
  - Catalyze all biochemical reactions
  - Control metabolism (energy & materials)
  - Process signals, activate genes

- Bootstrapping still a mystery
  - DNA, RNA, proteins, membranes are today interdependent. Not clear who came first
  - Not understood, not essential for us



Cells Compute

- 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.

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.

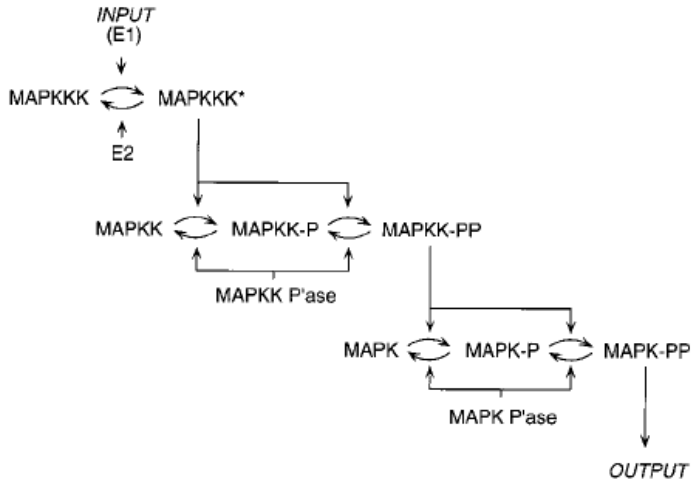


FIG. 1. Schematic view of the MAPK cascade. Activation of MAPK depends upon the phosphorylation of two conserved sites [Thr-183 and Tyr-185 in rat p42 MAPK/Erk2 (4, 5)]. Full activation of MAPKK also requires phosphorylation of two sites [Ser-218 and Ser-222 in mouse Mek-1/MKK1 (6–10)]. Detailed mechanisms for the activation of various MAPKKKs (e.g., Raf-1, B-Raf, Mos) are not yet established; here we assume that MAPKKKs are activated and inactivated by enzymes we denote E1 and E2. MAPKKK* denotes activated MAPKKK. MAPKK-P and MAPKK-PP denote singly and doubly phosphorylated MAPKK, respectively. MAPK-P and MAPK-PP denote singly and doubly phosphorylated MAPK. P'ase denotes phosphatase.

Stochastic Collectives

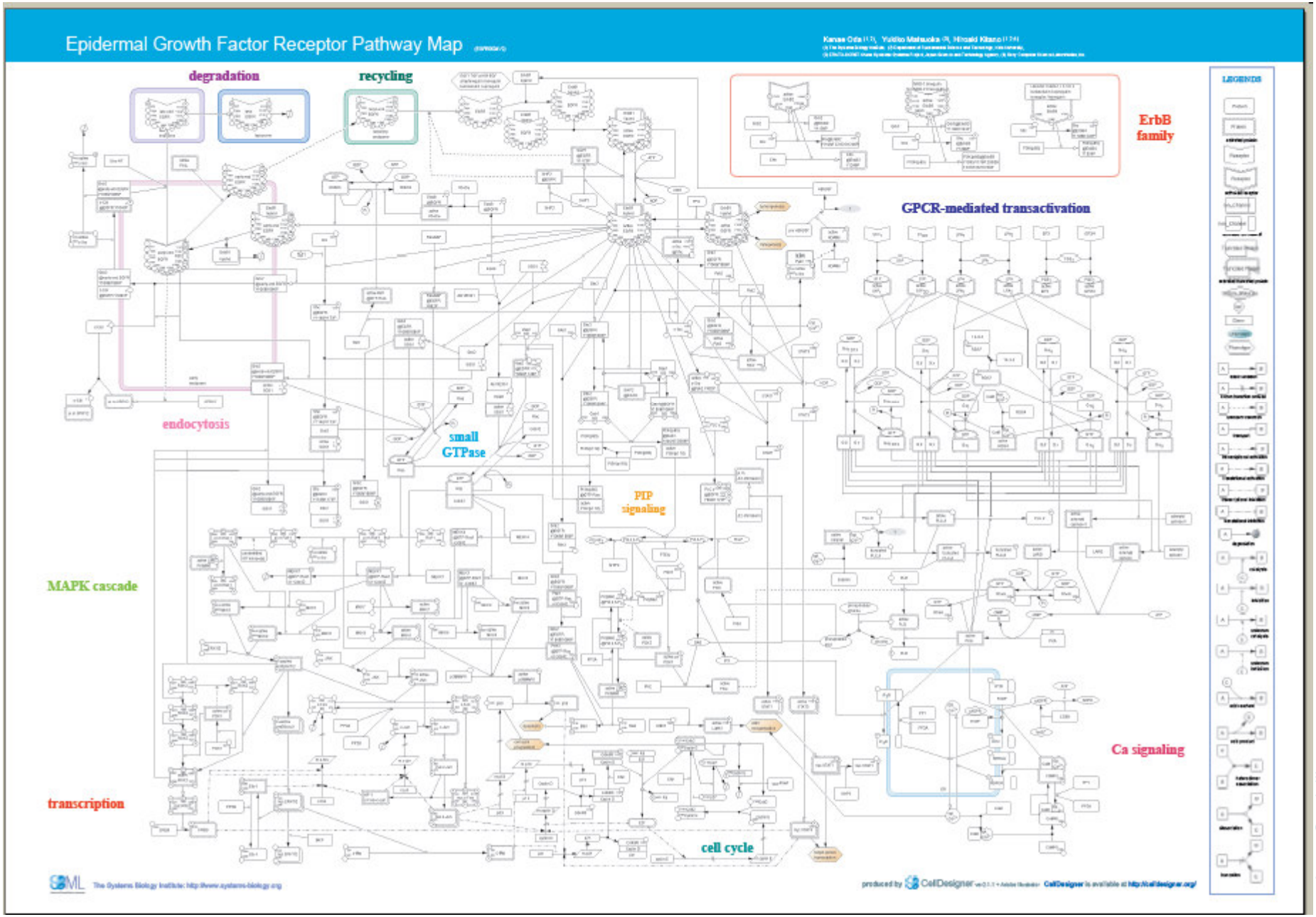
Stochastic Collectives

- "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

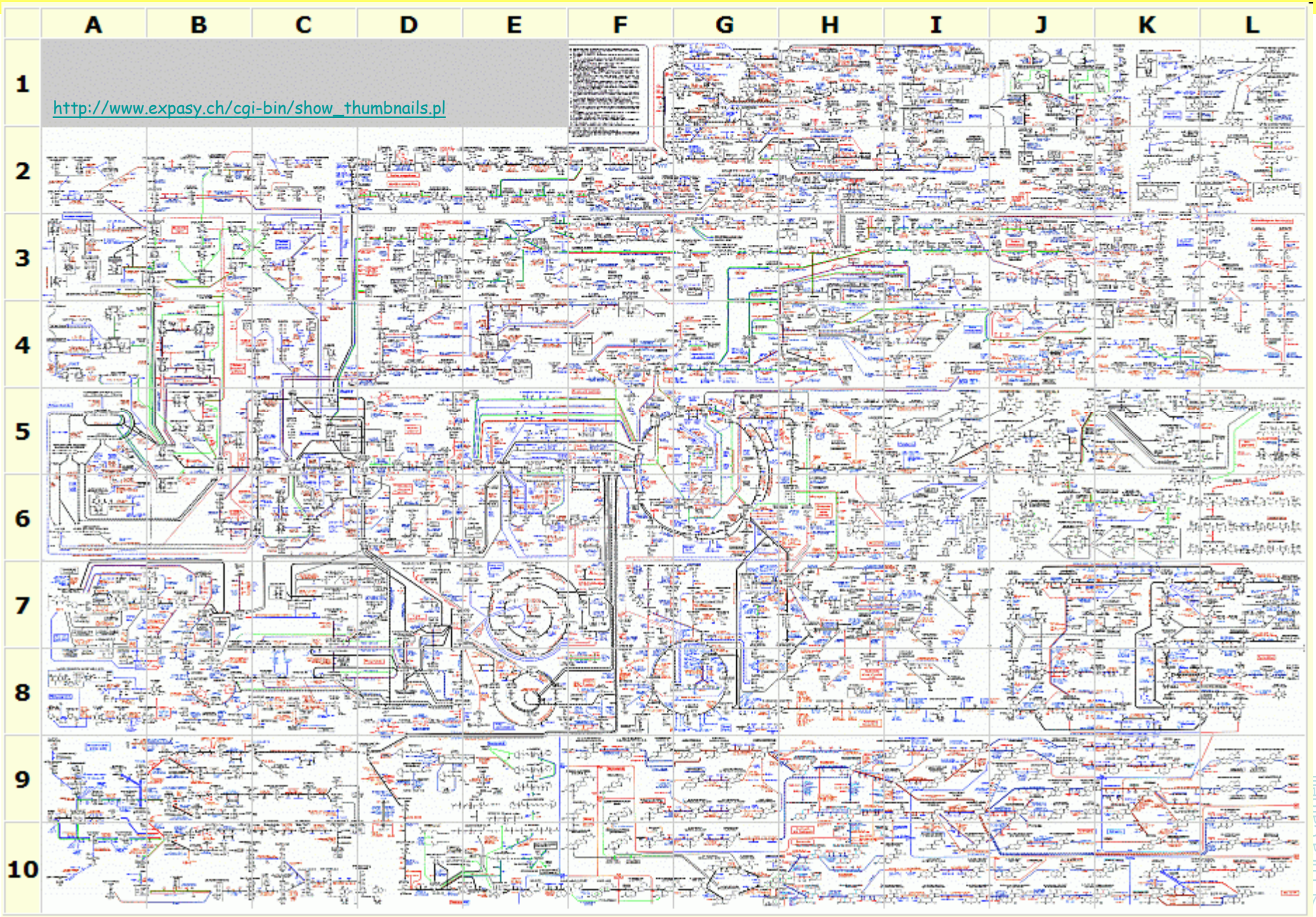
- "Stochastic":
  - Interactions have *rates*
    - Not quite discrete (hundreds or thousands of components)
    - Not quite continuous (non-trivial stochastic effects)
    - Not quite hybrid (no "switching" between regimes)

- 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].

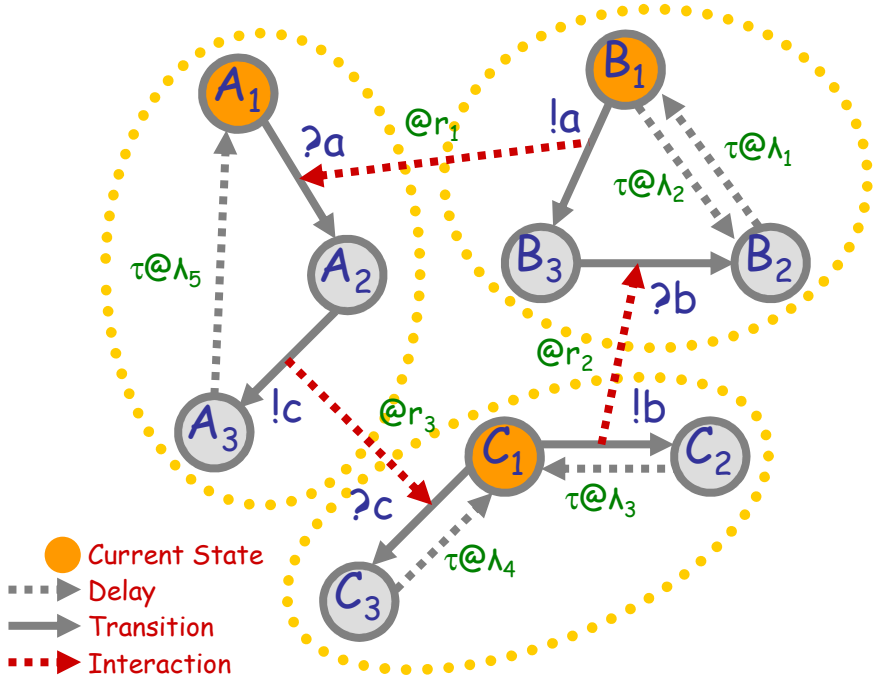
State Transitions



Compositionality (NOT!)



Interacting Automata



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.

$$\begin{aligned}
&\text{new } a@r_1 \\
&\text{new } b@r_2 \\
&\text{new } c@r_3
\end{aligned}$$ } Communication channels

$$\begin{aligned}
A_1 &= ?a;\ A_2 \\
A_2 &= !c;\ A_3 \\
A_3 &= \tau@\lambda_5;\ A_1
\end{aligned}$$

$$\begin{aligned}
B_1 &= \tau@\lambda_2;\ B_2 + !a;\ B_3 \\
B_2 &= \tau@\lambda_1;\ B_1 \\
B_3 &= ?b;\ B_2
\end{aligned}$$

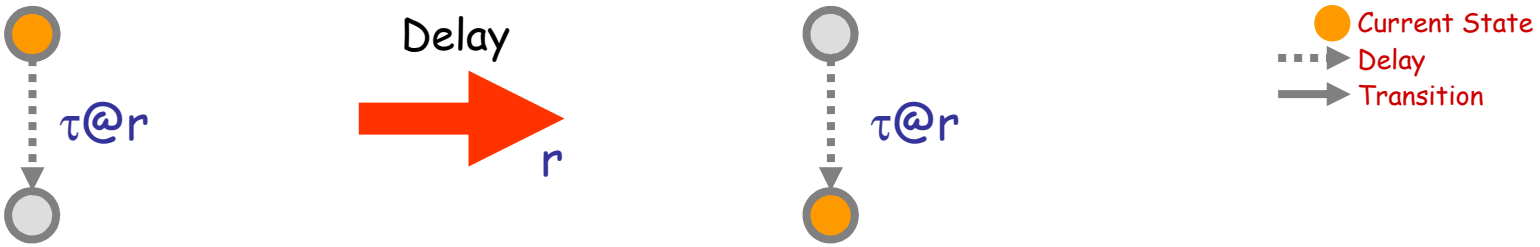
$$\begin{aligned}
C_1 &= !b;\ C_2 + ?c;\ C_3 \\
C_2 &= \tau@\lambda_3;\ C_1 \\
C_3 &= \tau@\lambda_4;\ C_2
\end{aligned}$$ } Automata

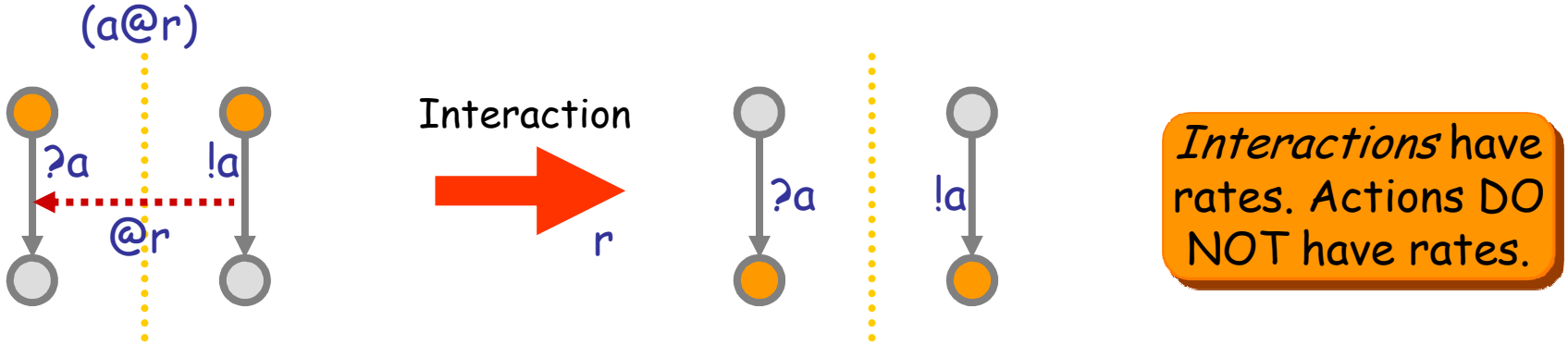
$$A_1 \mid B_1 \mid C_1$$ } The system and initial state

"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



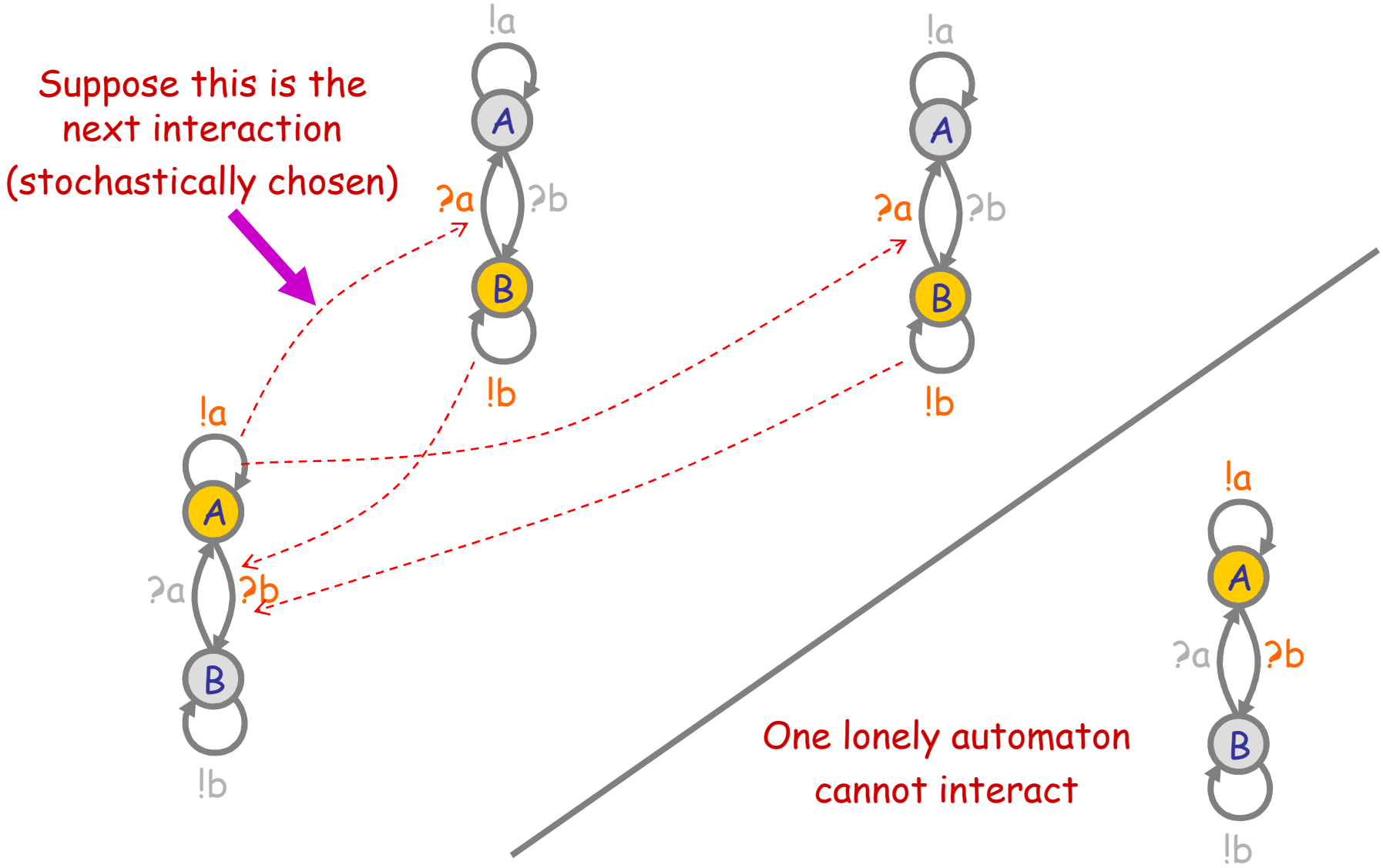


Interactions have rates. Actions DO NOT have rates.

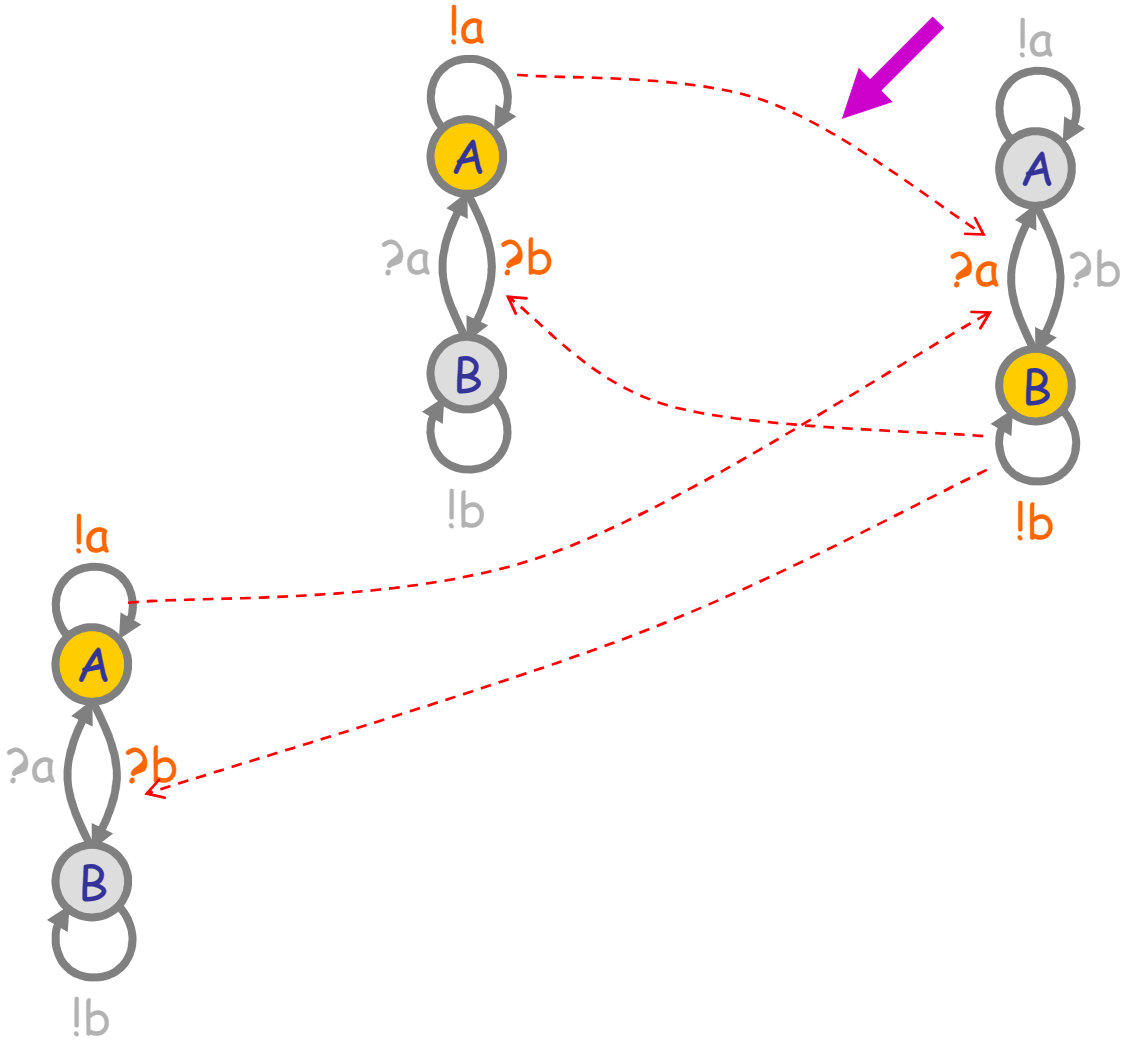
Q: What kind of mass behavior can this produce?

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

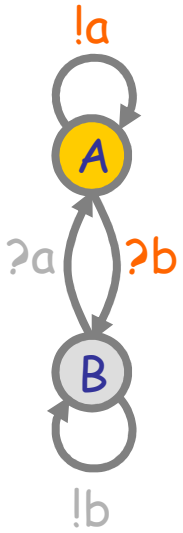
Interactions in a Population

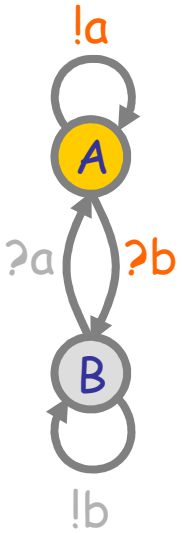


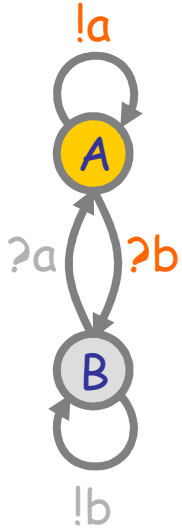
Interactions in a Population



Interactions in a Population

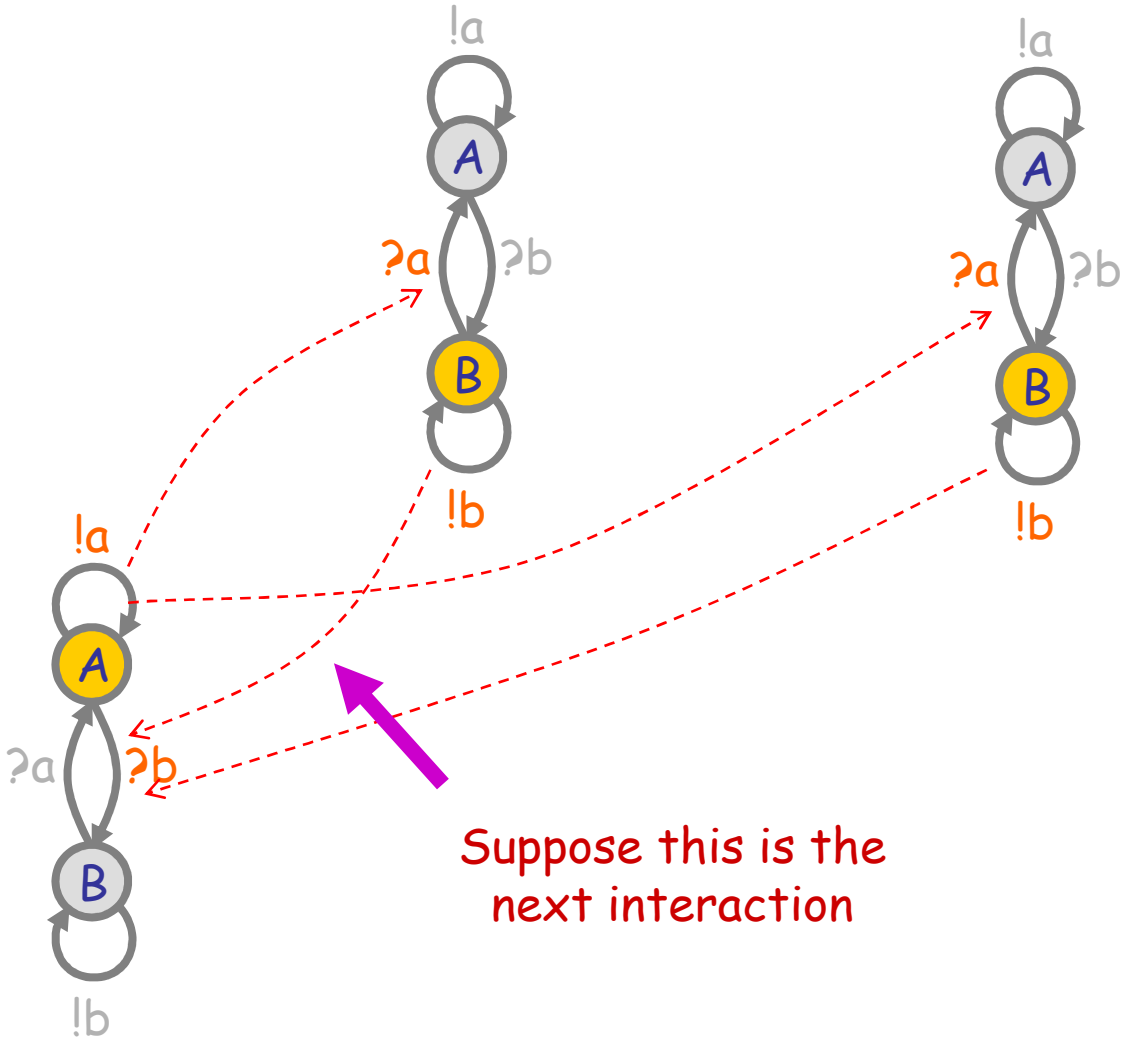




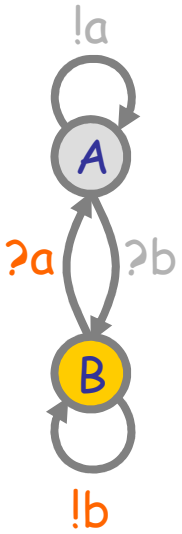


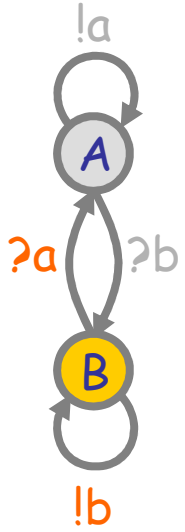
All-A stable population

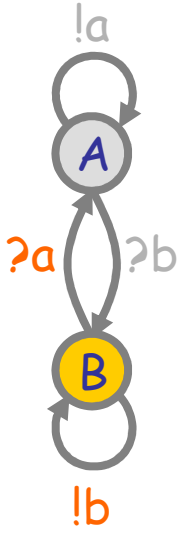
Interactions in a Population (2)



Interactions in a Population (2)







All-B stable population

Nondeterministic population behavior ("multistability")

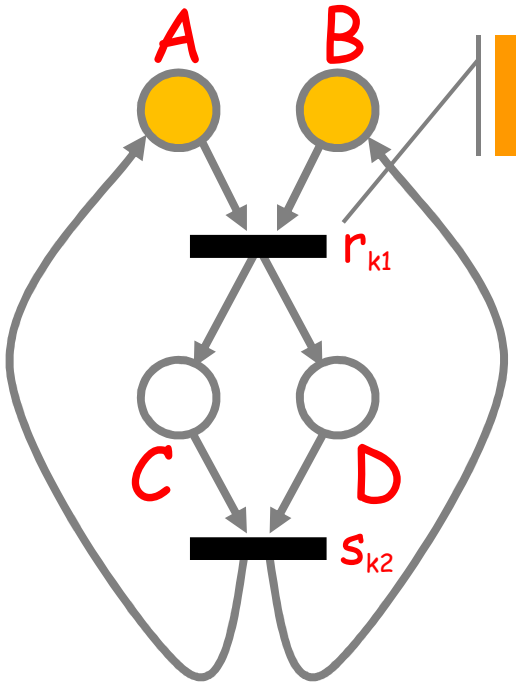
Chemistry vs. Automata

A process calculus (chemistry)

$r: A + B \rightarrow_{k1} C + D$

$s: C + D \rightarrow_{k2} A + B$

Does A become C or D?

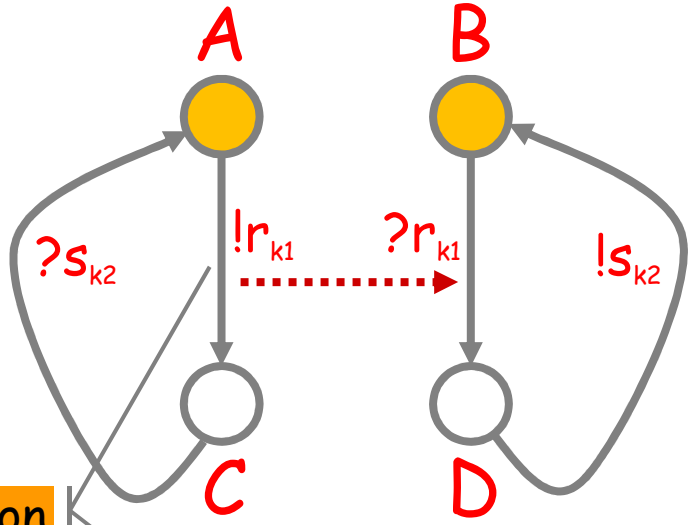


Reaction oriented

1 line per reaction

A Petri-Net-like representation. Precise and dynamic, but not modular, scalable, or maintainable.

A different process calculus (automata)



Interaction oriented

1 line per component

$A = !r_{k1}; C$

$C = ?s_{k2}; A$

$B = ?r_{k1}; D$

$D = !s_{k2}; B$

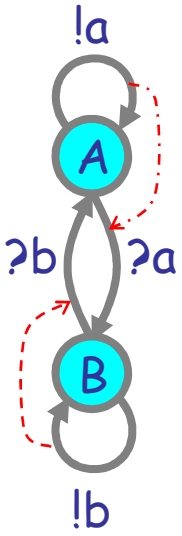
A becomes C not D!

The same "model"

Maps to a CTMC	Maps to a CTMC

A compositional graphical representation (precise, dynamic *and* modular) and the corresponding calculus.

Groupies and Celebrities



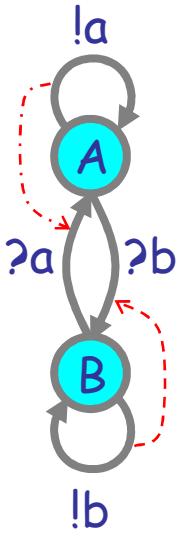
Celebrity

(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

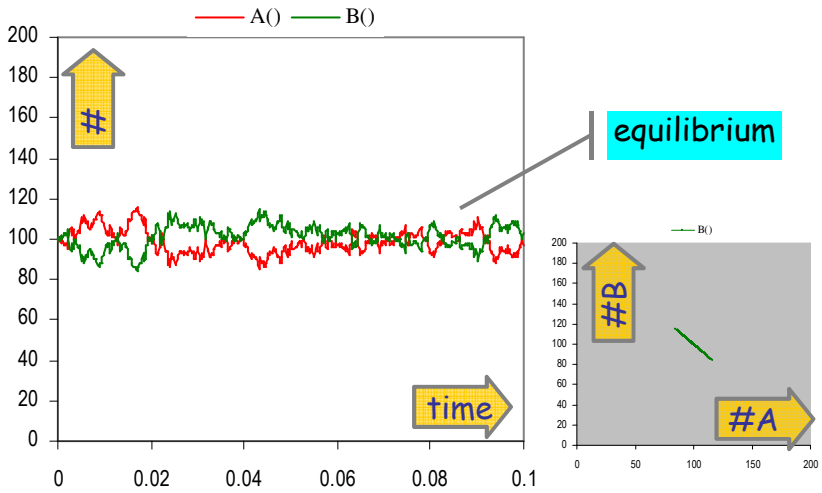
(wants to be like somebody different)

```
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

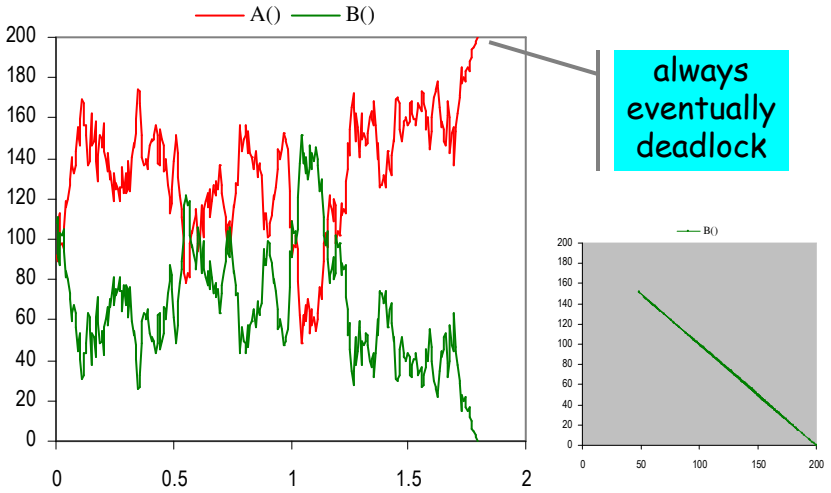
b@1.0

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:

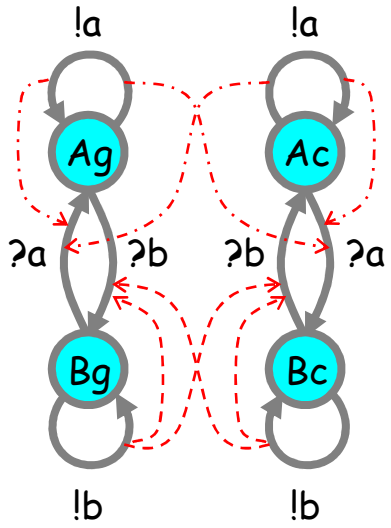


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

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

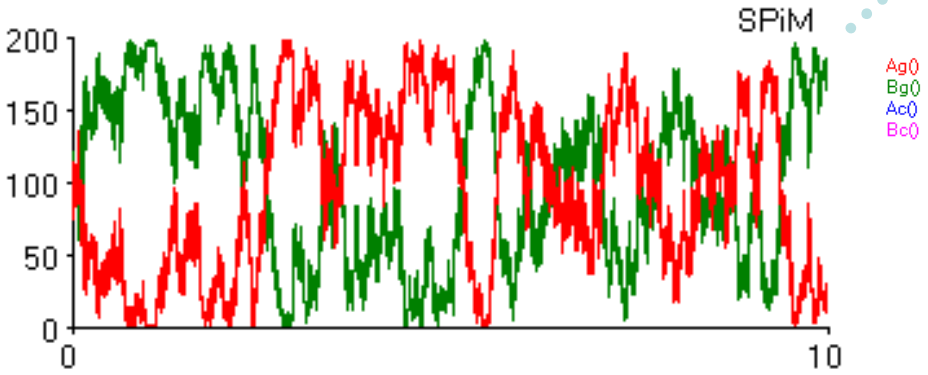
Many Groupies



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

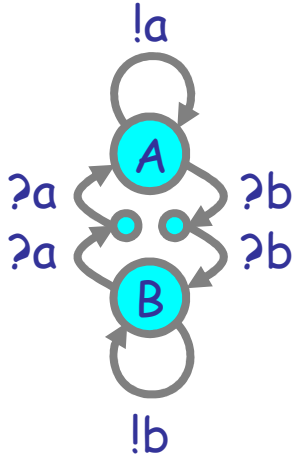


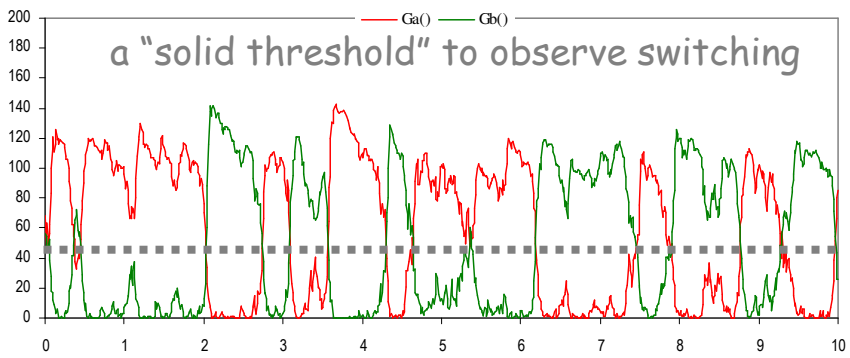
A tiny bit of "noise" can make a huge difference

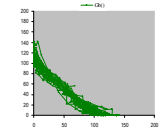
Regularity can arise not far from chaos

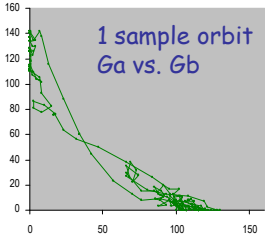
Hysteric Groupies

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

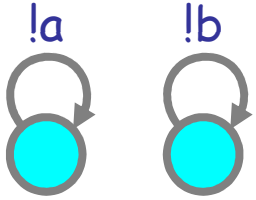








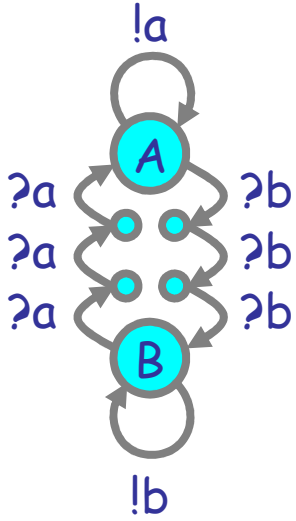

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

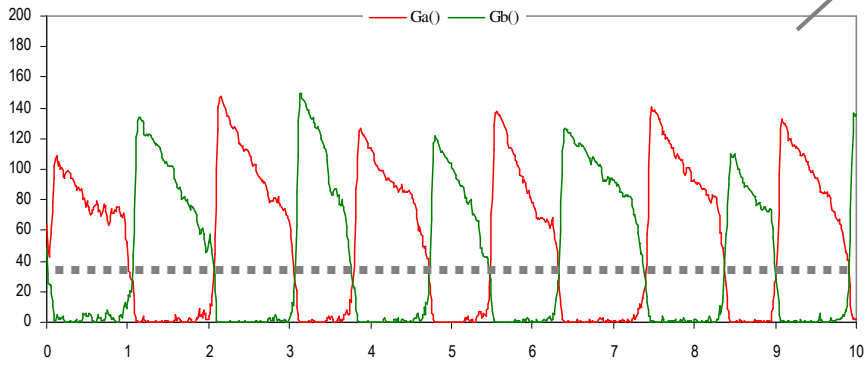


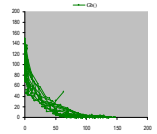
(With doping to break deadlocks)

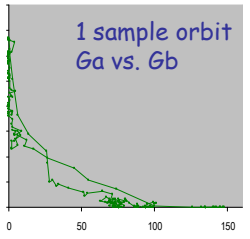
N.B.: It will not oscillate without doping (noise)

"regular" oscillation





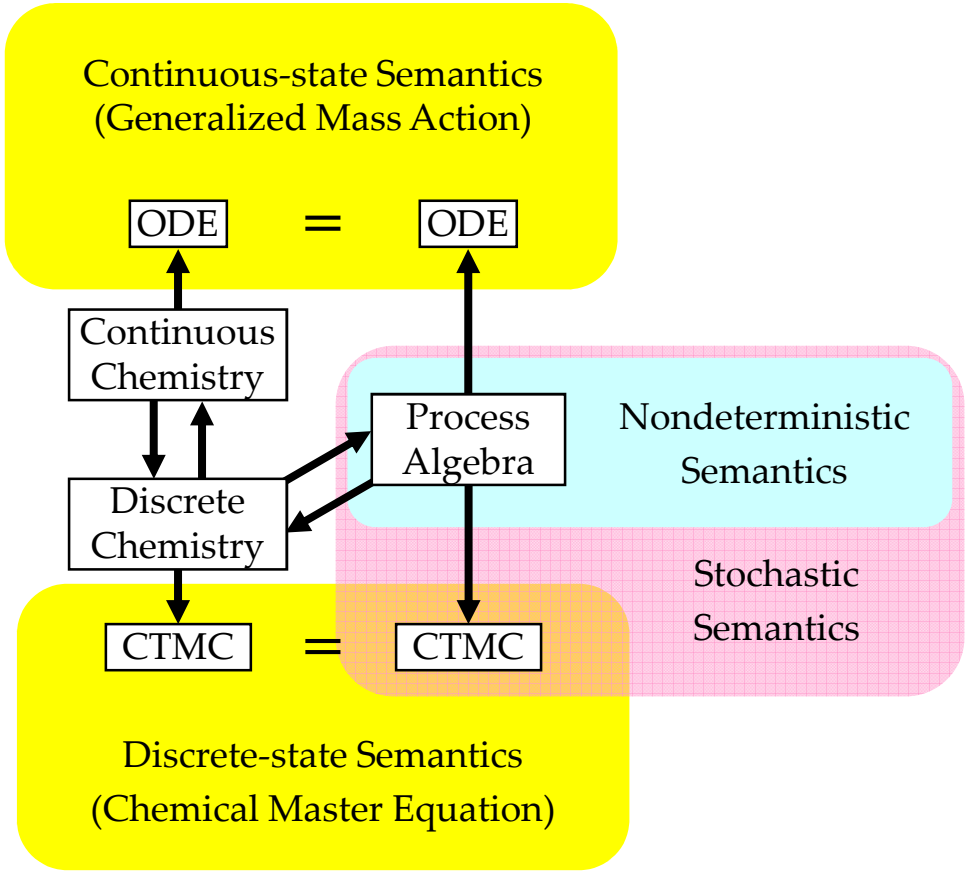





```
directive sample 10.0 1000
directive plot Ga(); Gb()

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



These diagrams commute via appropriate maps.

L. Cardelli: "On Process Rate Semantics"

From Processes to Chemistry

Chemical Ground Form (CGF)

$E ::= X_1=M_1, \ldots, X_n=M_n$ Reagents $(n \geq 0)$

$M ::= \pi_1;P_1 \oplus \ldots \oplus \pi_n;P_n$ Molecules $(n \geq 0)$

$P ::= X_1 \mid \ldots \mid X_n$ Solutions $(n \geq 0)$

$\pi ::= \tau_r \;\; ?n_{(r)} \;\; !n_{(r)}$ Interactions (delay, input, output)

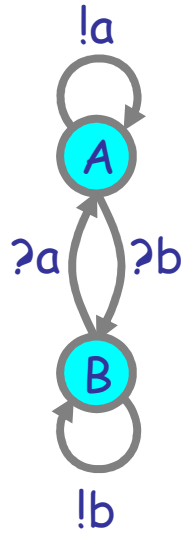
$CGF ::= E,P$ Reagents plus Initial Conditions

CGF = Interacting Automata + dynamic forking

Simplest process algebra *ever*

(To translate chemistry to processes we need a bit more than interacting 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):

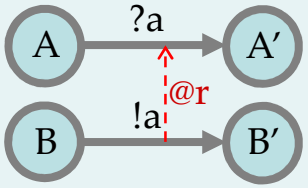
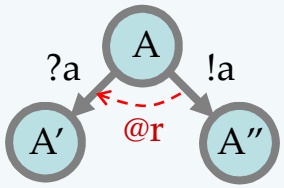
$A = !a;A \oplus ?b;B$ — Automaton in state A

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

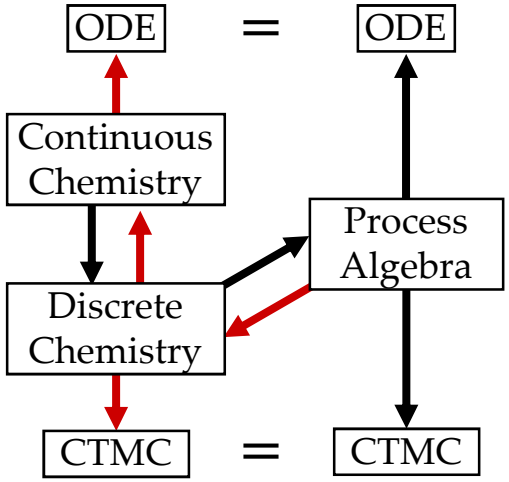
$A|A|B|B$ — Initial conditions: 2A and 2B

Processes to Chemistry

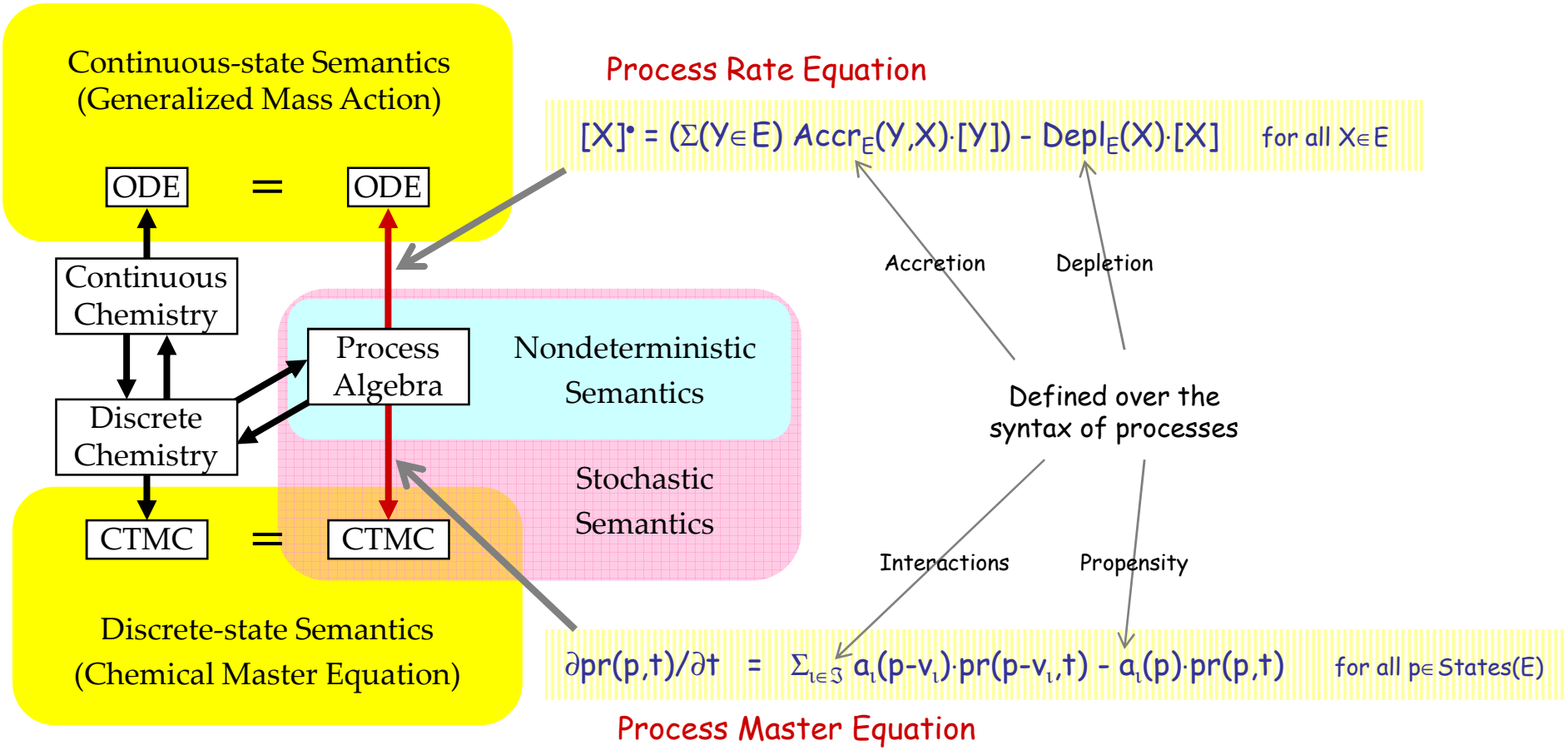
V = interaction volume
N_A = Avogadro's number

Automata →	Discrete Chemistry →	Continuous Chemistry	$\gamma = N_A V$
initial states A | A | ... | A	initial quantities $\#A_0$	initial concentrations $[A]_0$	with $[A]_0 = \#A_0/\gamma$
	$A \dashrightarrow^{r} A'$	$A \rightarrow^{k} A'$	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 $k = r\gamma/2$
	CTMC	ODE	

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



Quantitative Process Semantics



Processes to GMA Directly

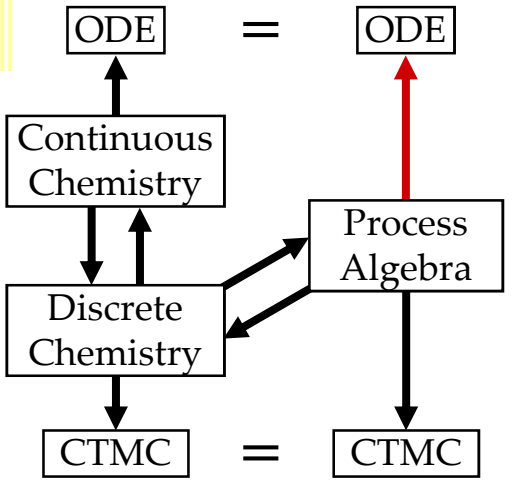
Process Rate Equation for Reagents E

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

$Depl_E(X) =$
$\quad \Sigma(i: E.X.i=\tau_{(r)};P)\ r +$
$\quad \Sigma(i: E.X.i=?a_{(r)};P)\ r\gamma\cdot OutsOn_E(a) +$
$\quad \Sigma(i: E.X.i=!a_{(r)};P)\ r\gamma\cdot InsOn_E(a)$

$Accr_E(Y, X) =$
$\quad \Sigma(i: E.Y.i=\tau_{(r)};P)\ \#X(P)\cdot r +$
$\quad \Sigma(i: E.Y.i=?a_{(r)};P)\ \#X(P)\cdot r\gamma\cdot OutsOn_E(a) +$
$\quad \Sigma(i: E.Y.i=!a_{(r)};P)\ \#X(P)\cdot r\gamma\cdot InsOn_E(a)$

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



$X = \tau_{(r)};0 \rightarrow [X]^\bullet = -r[X]$

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

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

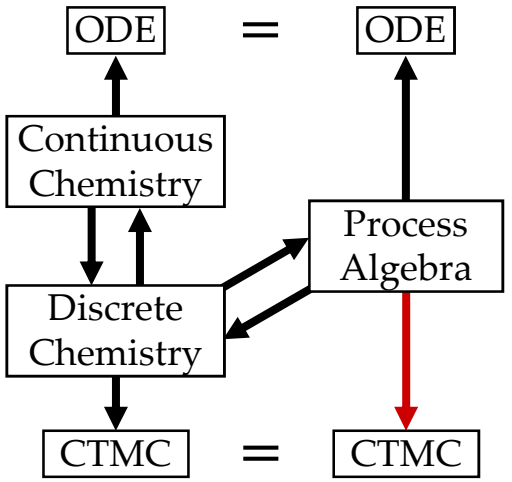
Processes to CME Directly

Process Master Equation for Reagents E

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

$pr(p,t) = Pr\{\mathbf{S}(t)=p \mid \mathbf{S}(0)=p_0\}$ 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 $E.X.j = !a_{(r)};R$

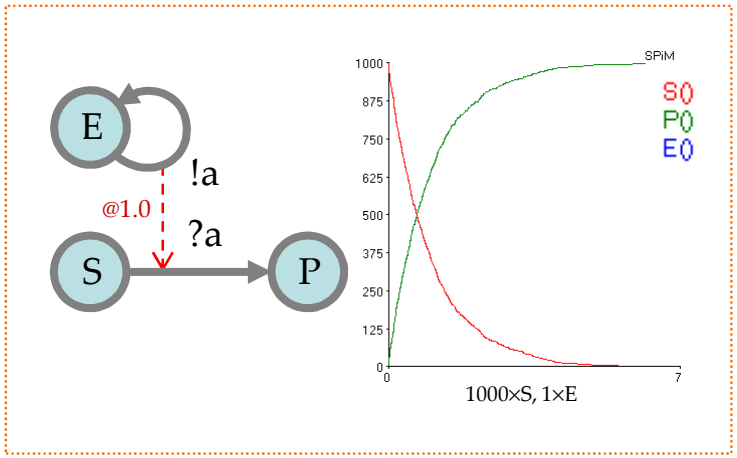
Examples of stochastic collectives where:

(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

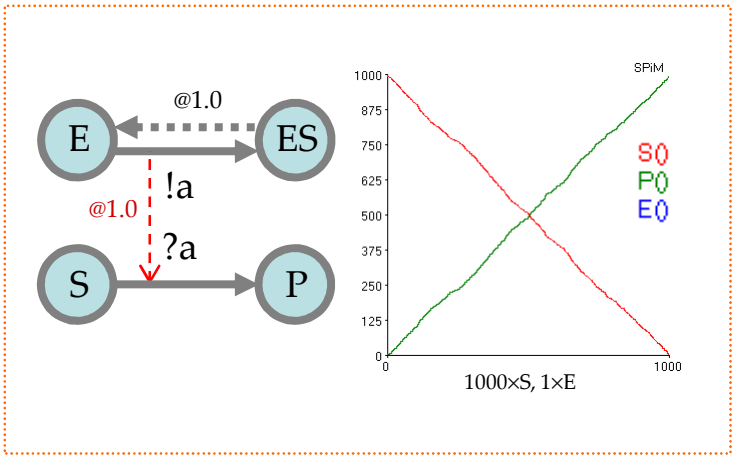


$E+S \rightarrow^{r} E+P$

```
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

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



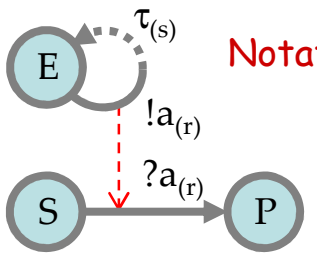
$E+S \rightarrow^{r} ES+P$

$ES \rightarrow^{s} E$

```
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())
```

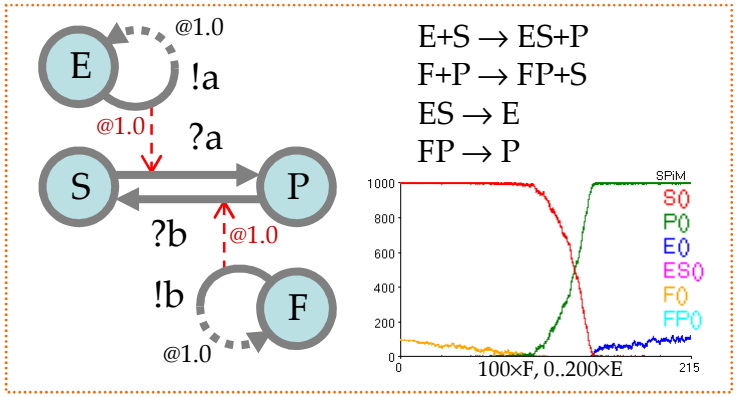
Zero-Order Regime

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



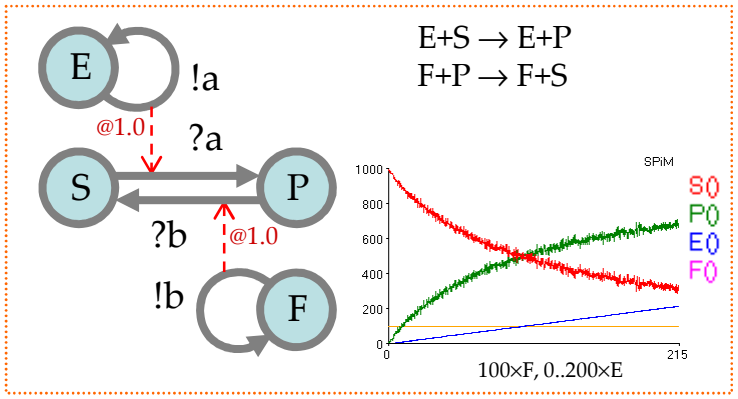
Notation

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)
```

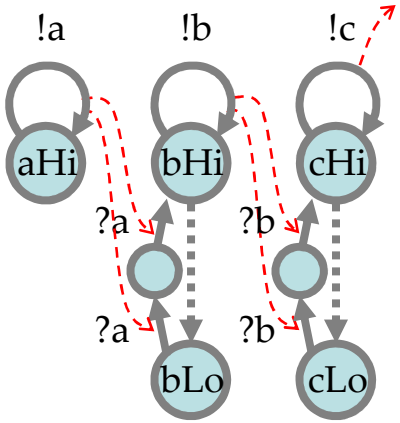
Zero-Order Regime
A small E-F inbalance causes a much larger S-P switch.

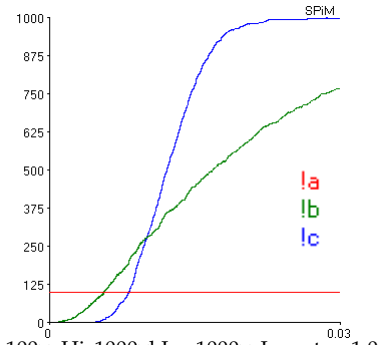



```
directive sample 215.0 1000
directive plot S(); P(); E(); F()

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



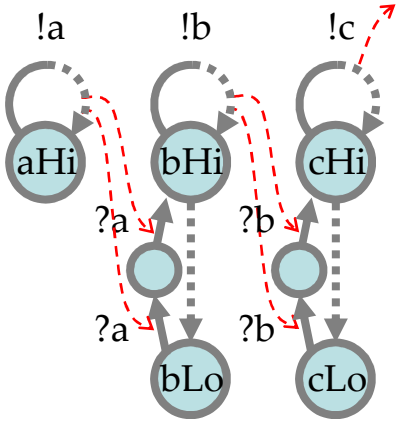


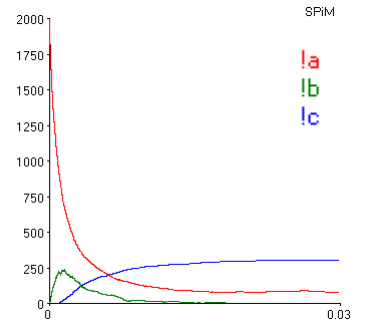
100×aHi, 1000×bLo, 1000×cLo, rates=1.0

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

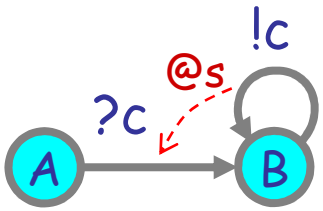
Zero-Oder Regime cascade:
a signal *divider!*

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 Transitions

Nonlinear Transition (NLT)



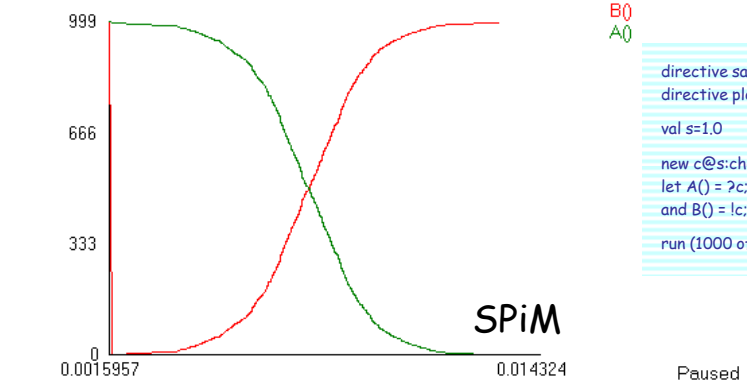
$A = ?c_{(s)};B$

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

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

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

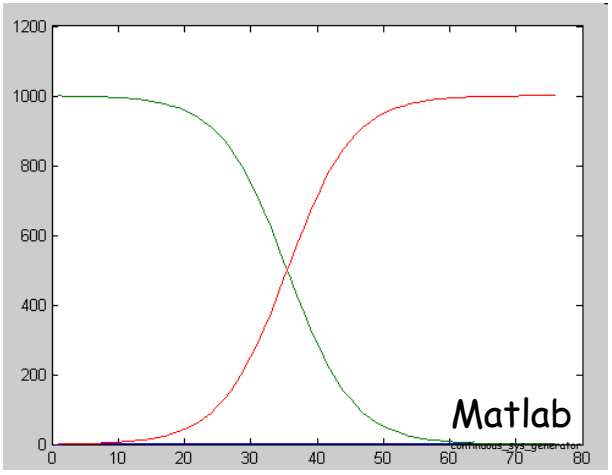
$[B]^{\bullet} = s[A][B]$



Simulation: Time = 0.013448 (999 points at 0.0085215 simTime/sysTime and halted)

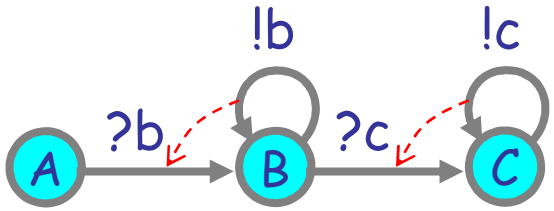
```
directive sample 0.02 1000
directive plot B(); A()

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



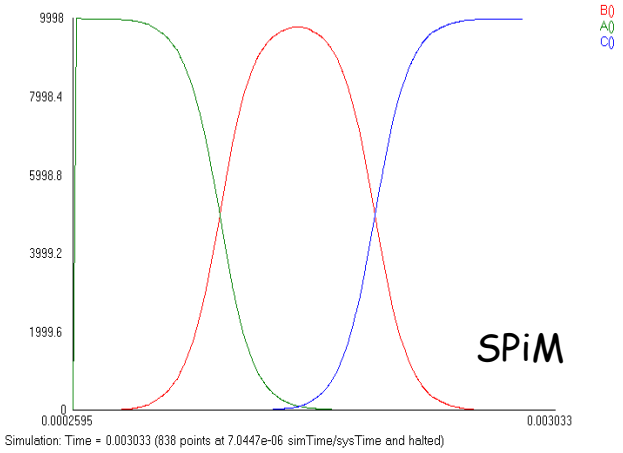
$[B]^\bullet = [B]([A]-[C])$

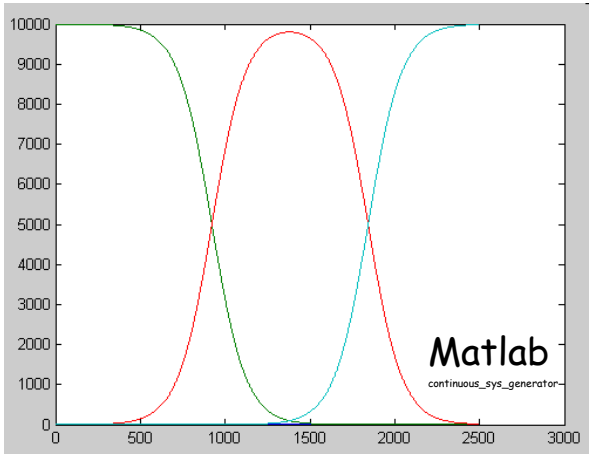
$A = ?b_{(1)};B$
$B = !b_{(1)};B \oplus ?c_{(1)};C$
$C = !c_{(1)};C$

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

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

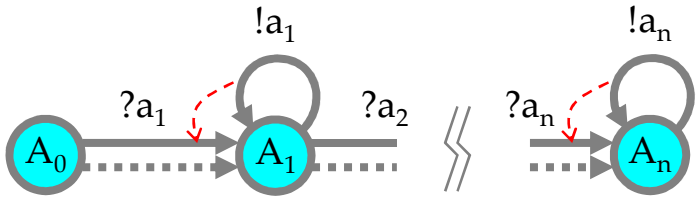
```
directive sample 0.0025 1000
directive plot B(); A(); C()

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())
```

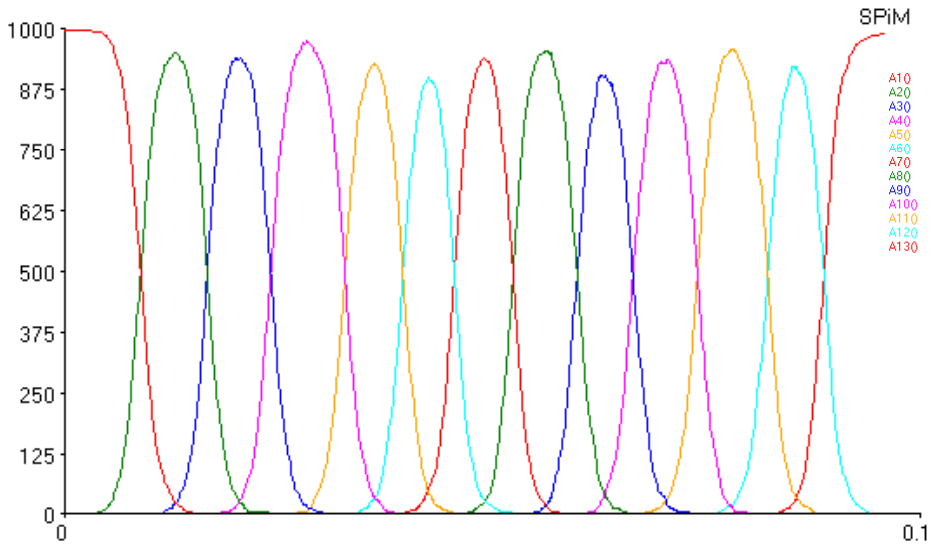





```
interval/step [0:0.000001:0.0025]
(A)   dx1/dt = -x1*x2          10000.0
(B)   dx2/dt = x1*x2 - x2*x3   1.0
(C)   dx3/dt = x2*x3           1.0
```

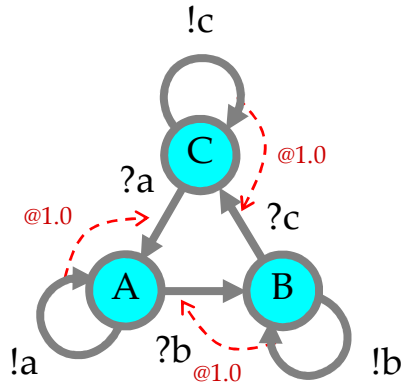
NLTs in Series: Soliton Propagation

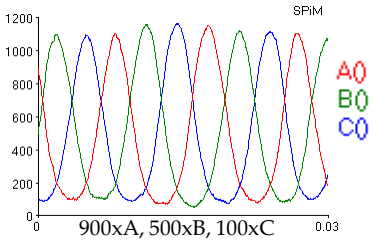





```
directive sample 0.1 1000
directive plot A1(); A2(); A3(); A4(); A5(); A6(); A7(); A8();
A9(); A10(); A11(); A12(); A13()

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



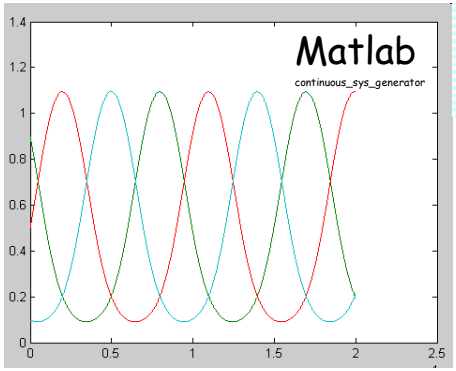



```
directive sample 0.03 1000
directive plot A(); B(); C()

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())
```

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

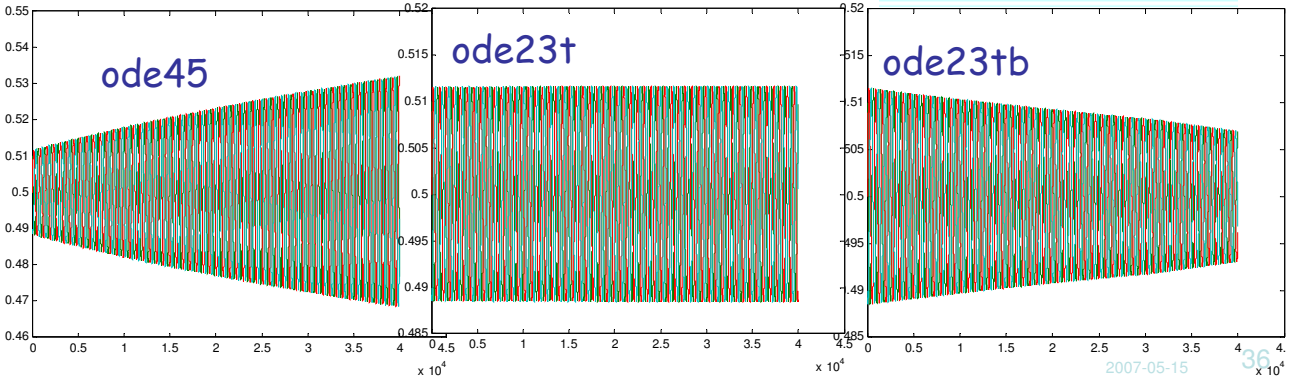
$$A+B \rightarrow^{s} B+B$$
$$B+C \rightarrow^{s} C+C$$
$$C+A \rightarrow^{s} A+A$$

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




```
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
```

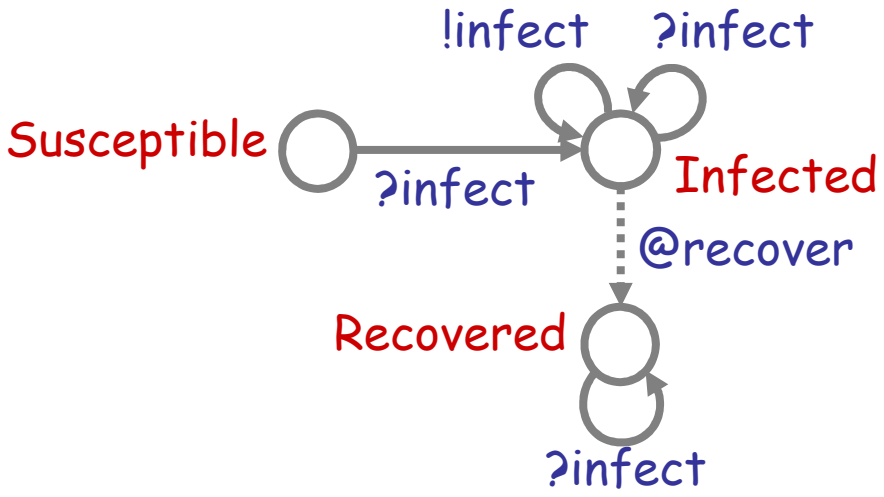


Epidemics

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



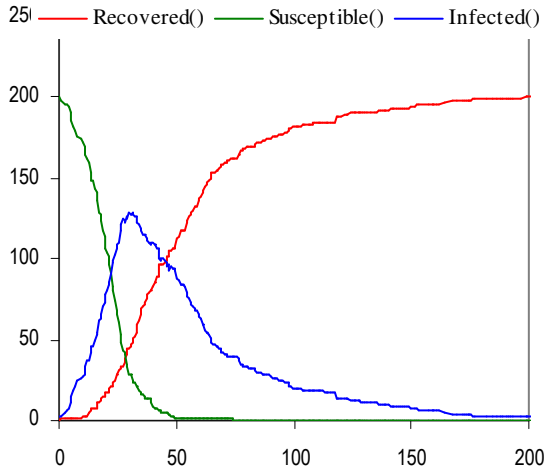

```
directive sample 500.0 1000
directive plot Recovered(); Susceptible(); Infected()

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

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

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

$[R]^\bullet = r[I]$

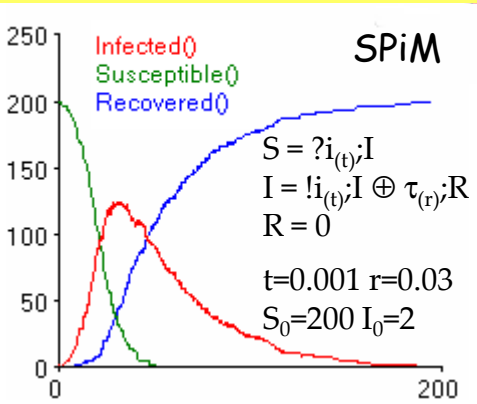
Automata produce the standard ODEs!

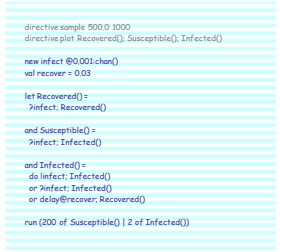
$$\frac{dS}{dt} = -aIS$$

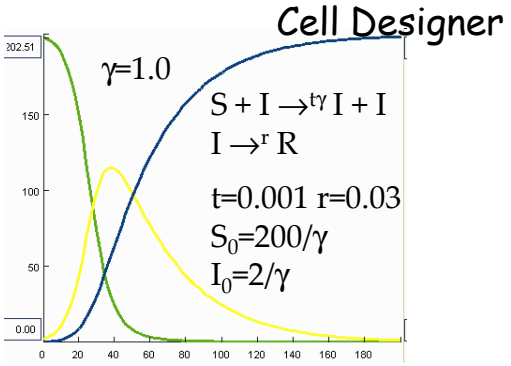
$$\frac{dI}{dt} = aIS - bI$$

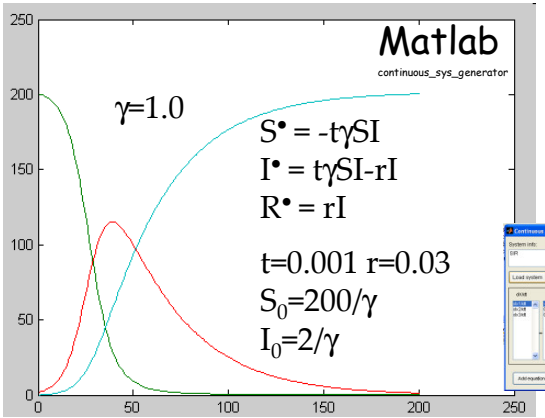
$$\frac{dR}{dt} = bI$$

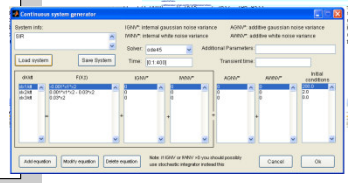
(the Kermack-McKendrick, or SIR model)



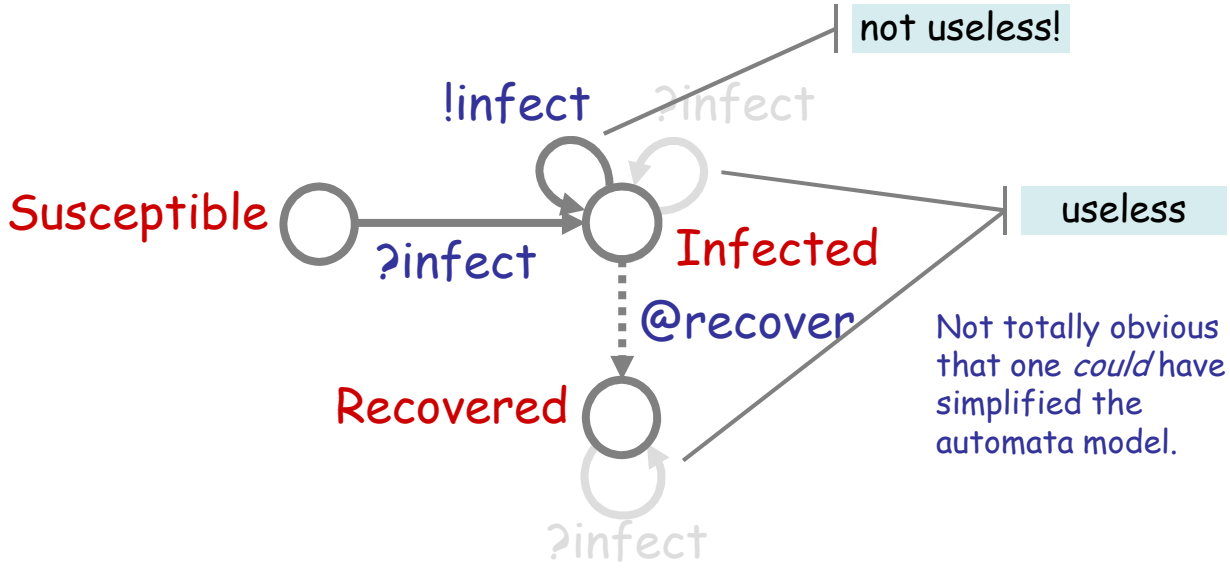








Simplified Model



Not totally obvious that one *could* have simplified the automata model.

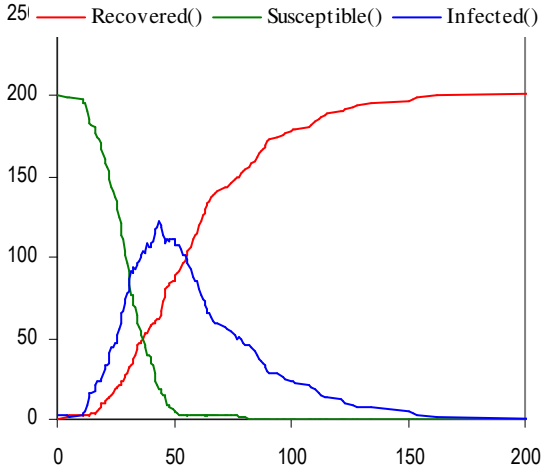
$$S = ?i_{(t)};I$$
$$I = !i_{(t)};I \oplus \tau_r;R$$
$$R = 0$$

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

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

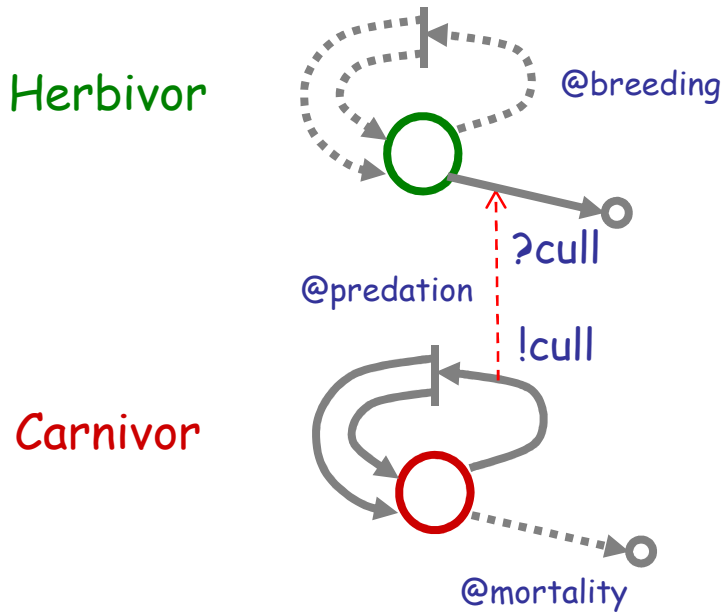
Same ODE, hence equivalent automata models.

```
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())
```

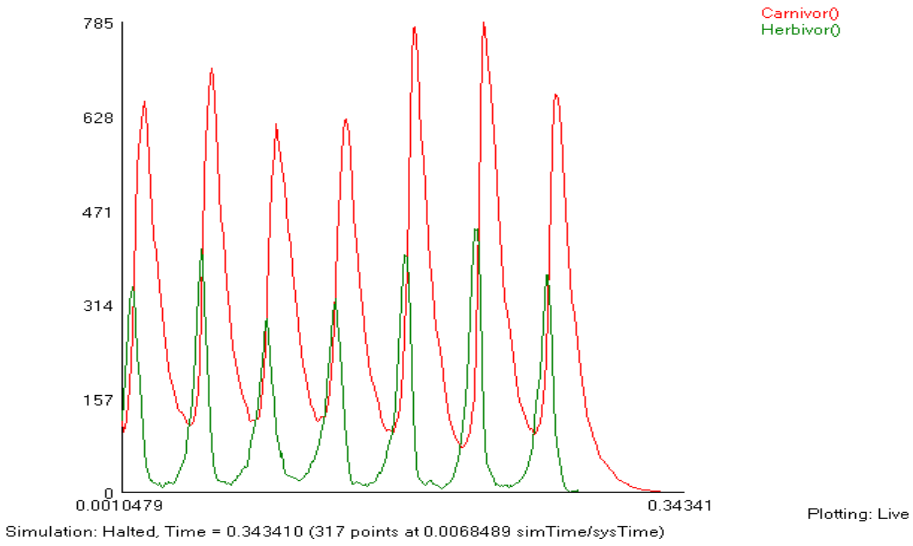


Lotka-Volterra

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

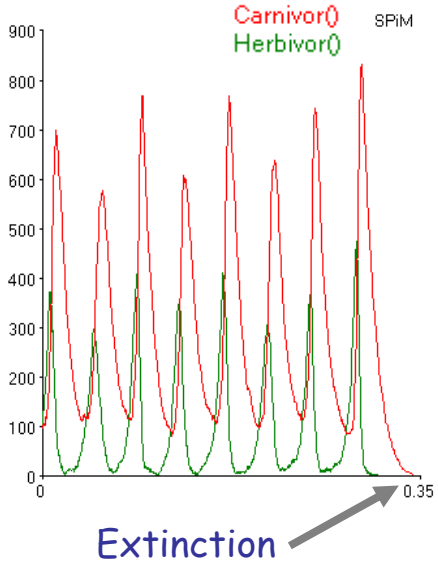
$H = \tau_b;\ (H|H) \oplus ?c_{(p)};0$
$C = \tau_m;0 \oplus !c_{(p)};(C|C)$
$\#H_0, \#C_0$

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

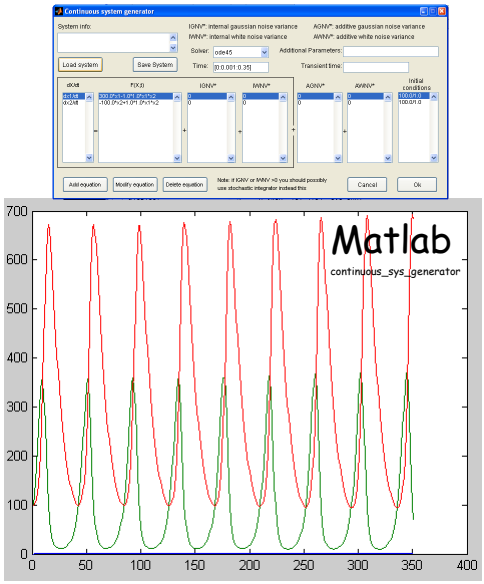
$[H]^{\bullet} = b[H]-p\gamma[H][C]$
$[C]^{\bullet} = -m[C]+p\gamma[H][C]$
$[H]_0 = \#H_0/\gamma$
$[C]_0 = \#C_0/\gamma$

m=100.0
b=300.0
p=1.0
γ=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()
```

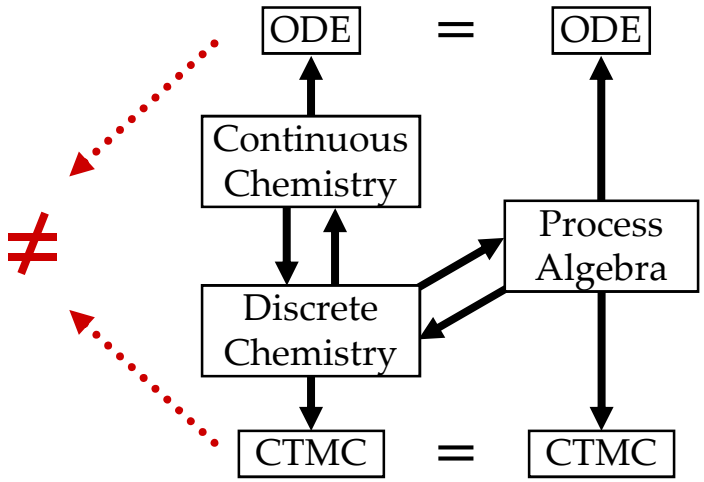


Extinction

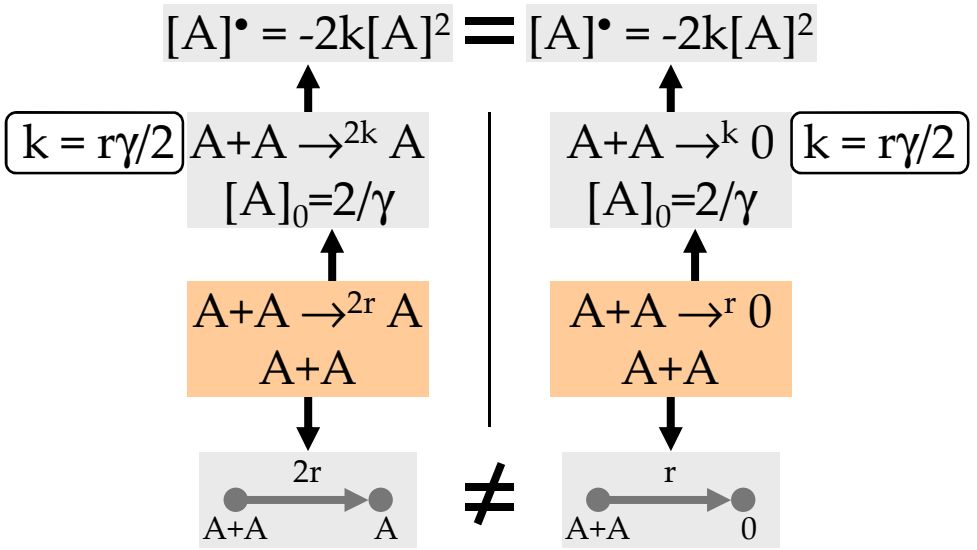


No extinction

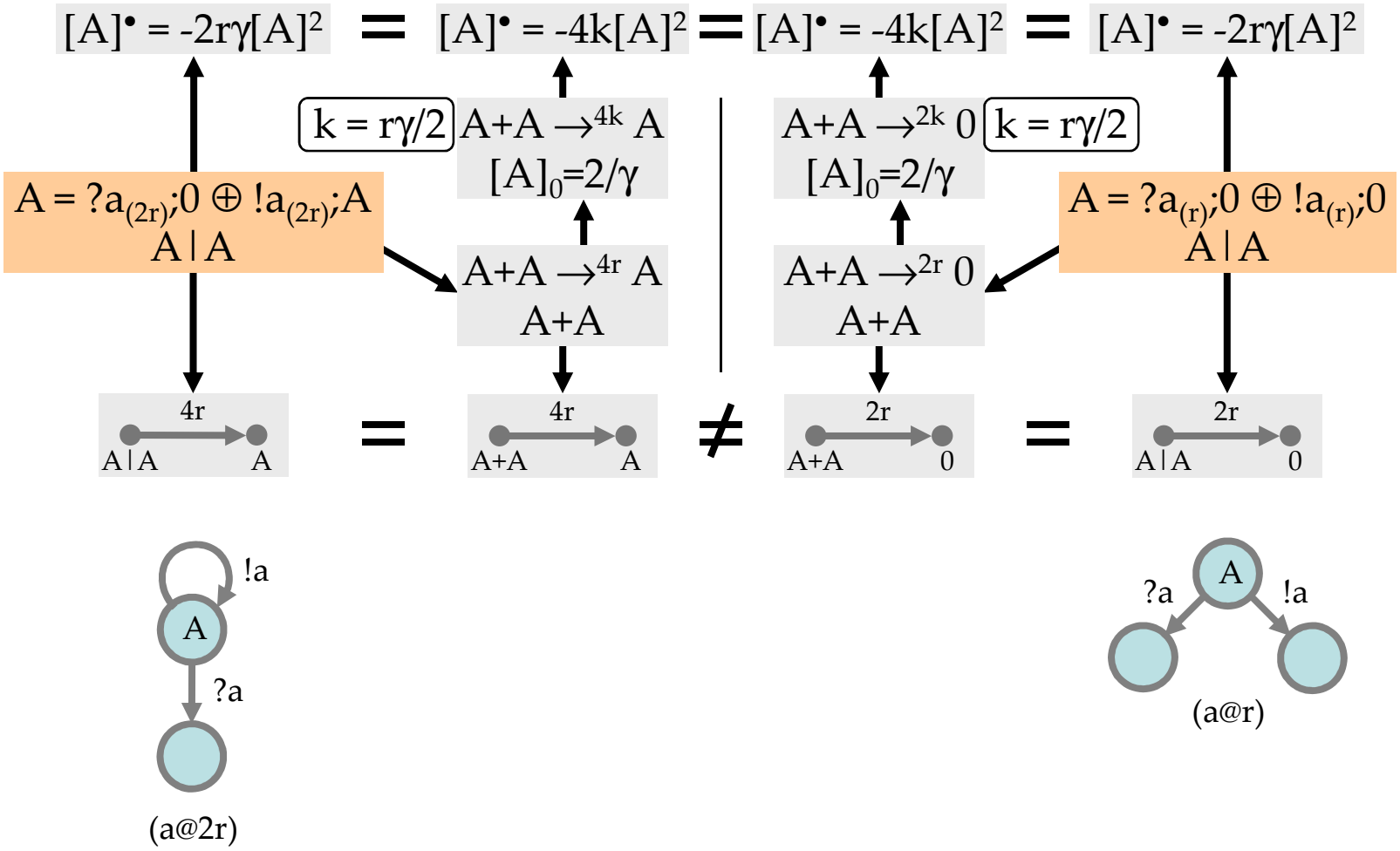
GMA ≠ CME


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

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



... as Automata


[A]• = -2rγ[A]²
=
[A]• = -4k[A]²
=
[A]• = -4k[A]²
=
[A]• = -2rγ[A]²
k = rγ/2
A+A →4k A
[A]0=2/γ
A+A →2k 0
[A]0=2/γ
k = rγ/2
A = ?a(2r);0 ⊕ !a(2r);A
A | A
A+A →4r A
A+A
A+A →2r 0
A+A
A = ?a(r);0 ⊕ !a(r);0
A | A
4r
A | A
A
=
4r
A+A
A
≠
2r
A+A
0
=
2r
A | A
0
!a
A
?a
(a@2r)
A
?a
!a
(a@r)

Conclusions

Conclusions

http://LucaCardelli.name

- Devising 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.

- Parametric Processes (not shown)
  - An standard extension for the modular description of components.

- PolyAutomata (not shown)
  - Artificial ***Bio***-Chemistry: complexation and polymerization.

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

Q?