Artificial Biochemistry

Luca Cardelli

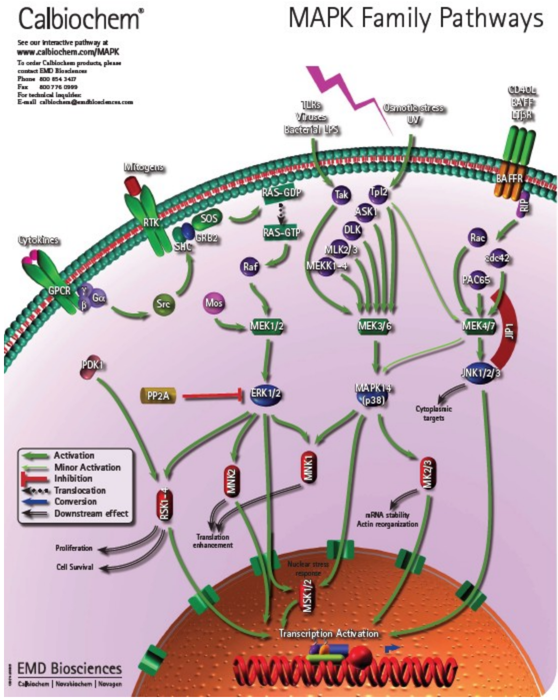
Microsoft Research

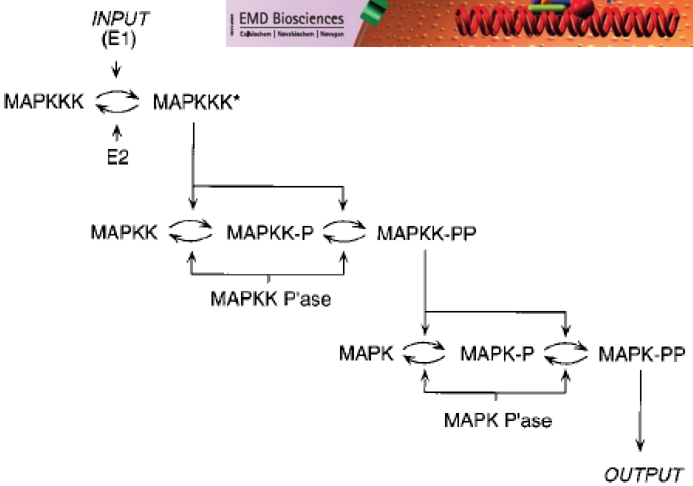
Algorithmic Bioprocesses
Leiden, 2007-12-04

http://LucaCardelli.name

Cells Compute

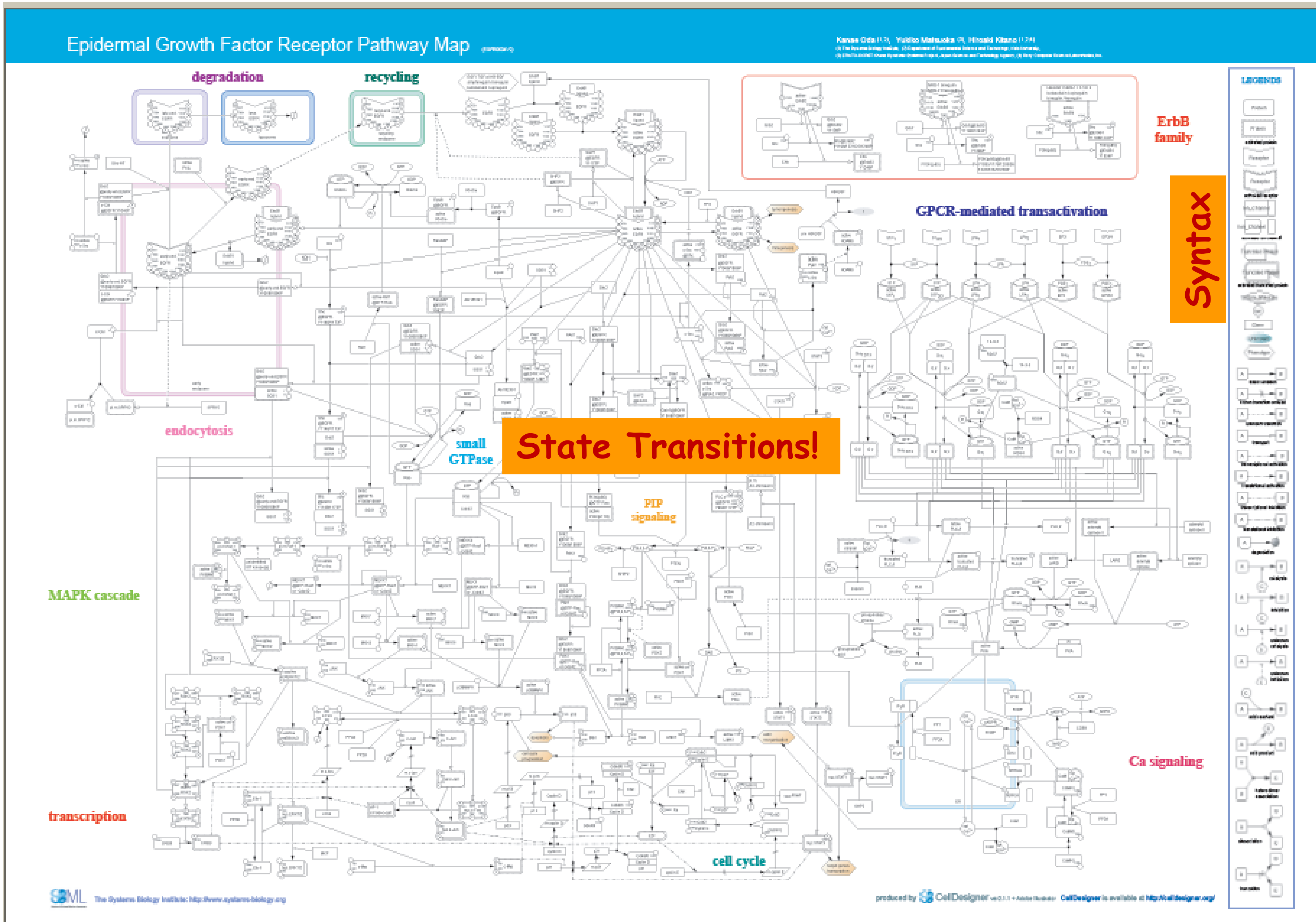
- No survival without computation!
  - Finding food
  - Avoiding predators
- How do they compute?
  - Unusual computational paradigms.
  - Proteins: do they work like electronic circuits? or process algebra?
  - Genes: what kind of software is that?
- Signaling networks
  - Clearly "information processing"
  - They are "just chemistry": molecule interactions
  - But what are their principles and algorithms?
- Complex, higher-order interactions
  - MAPKKK = MAP Kinase Kinase Kinase: that which operates on that which operates on that which operates on protein.





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.

The View from Systems Biology

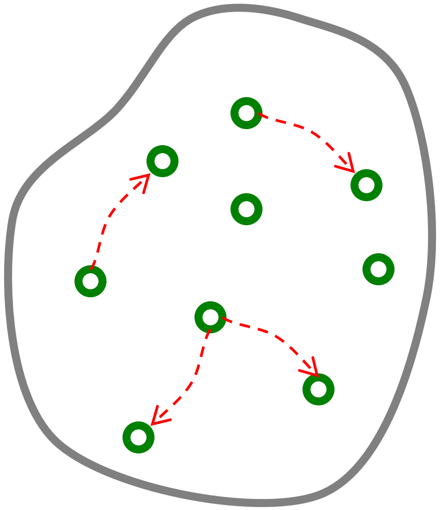


Stochastic Collectives

Stochastic Collectives

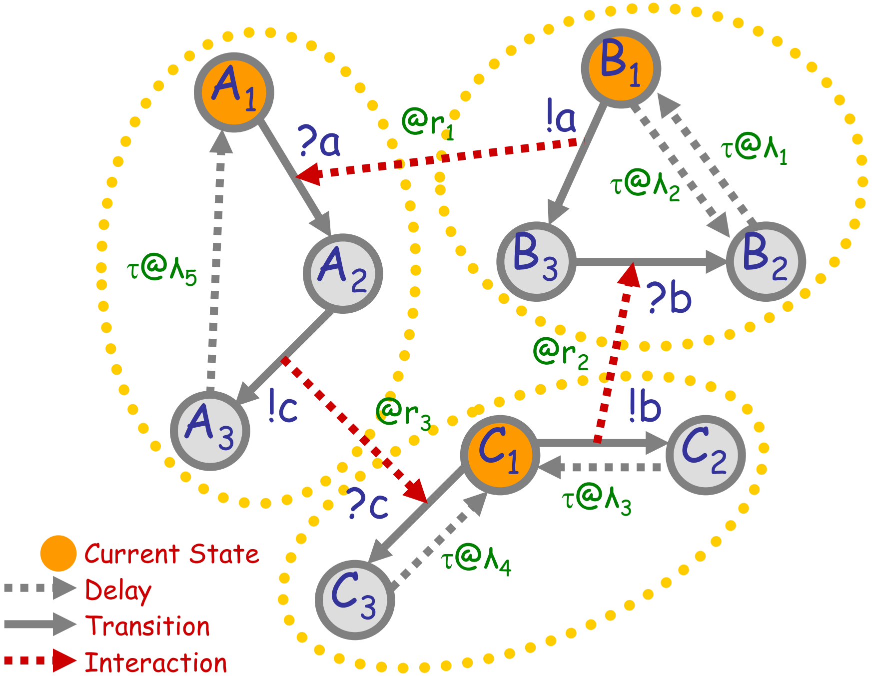
[Regev-Shapiro]

- "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].

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.

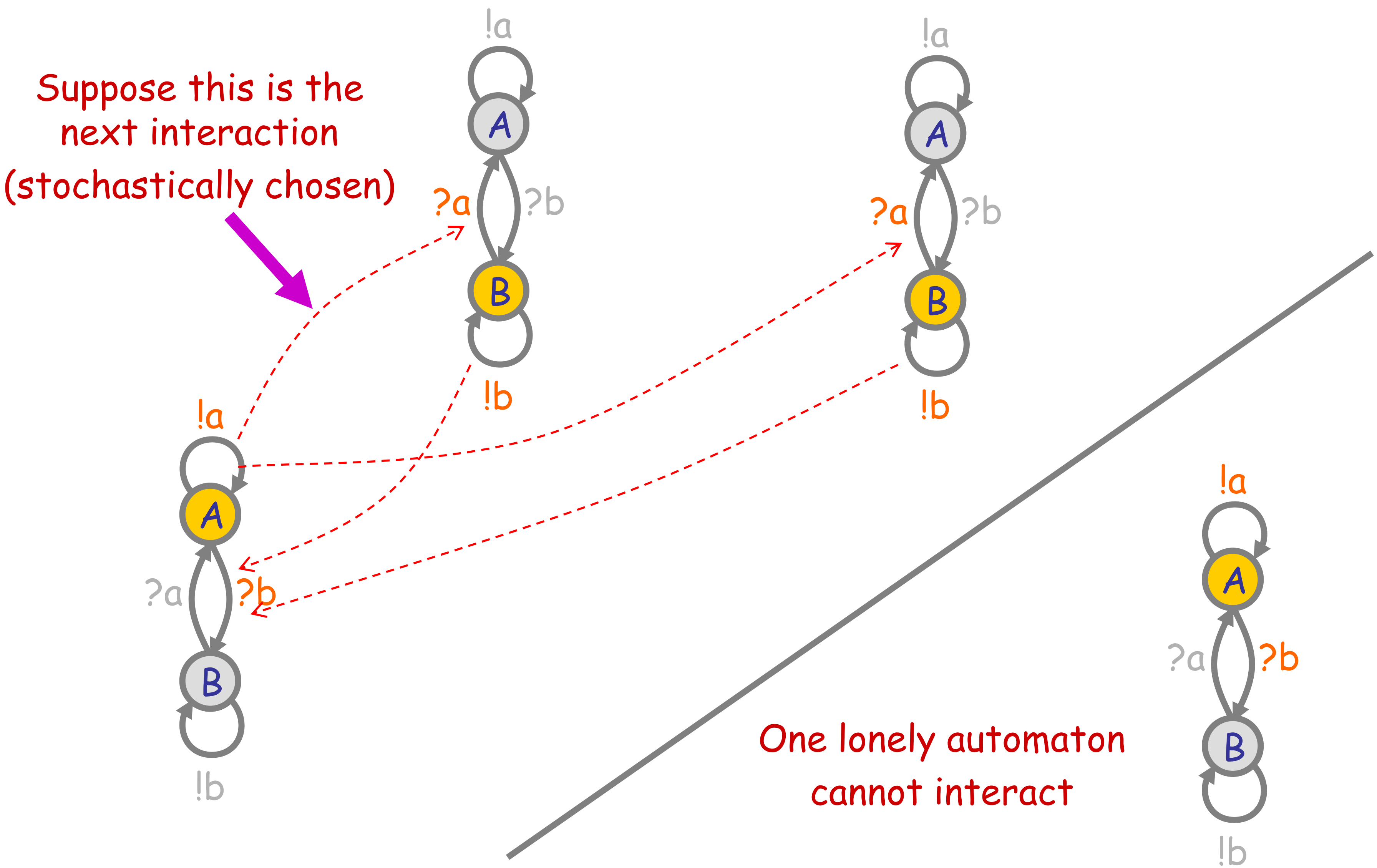
The stochastic version has *rates* on communications, and delays.

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

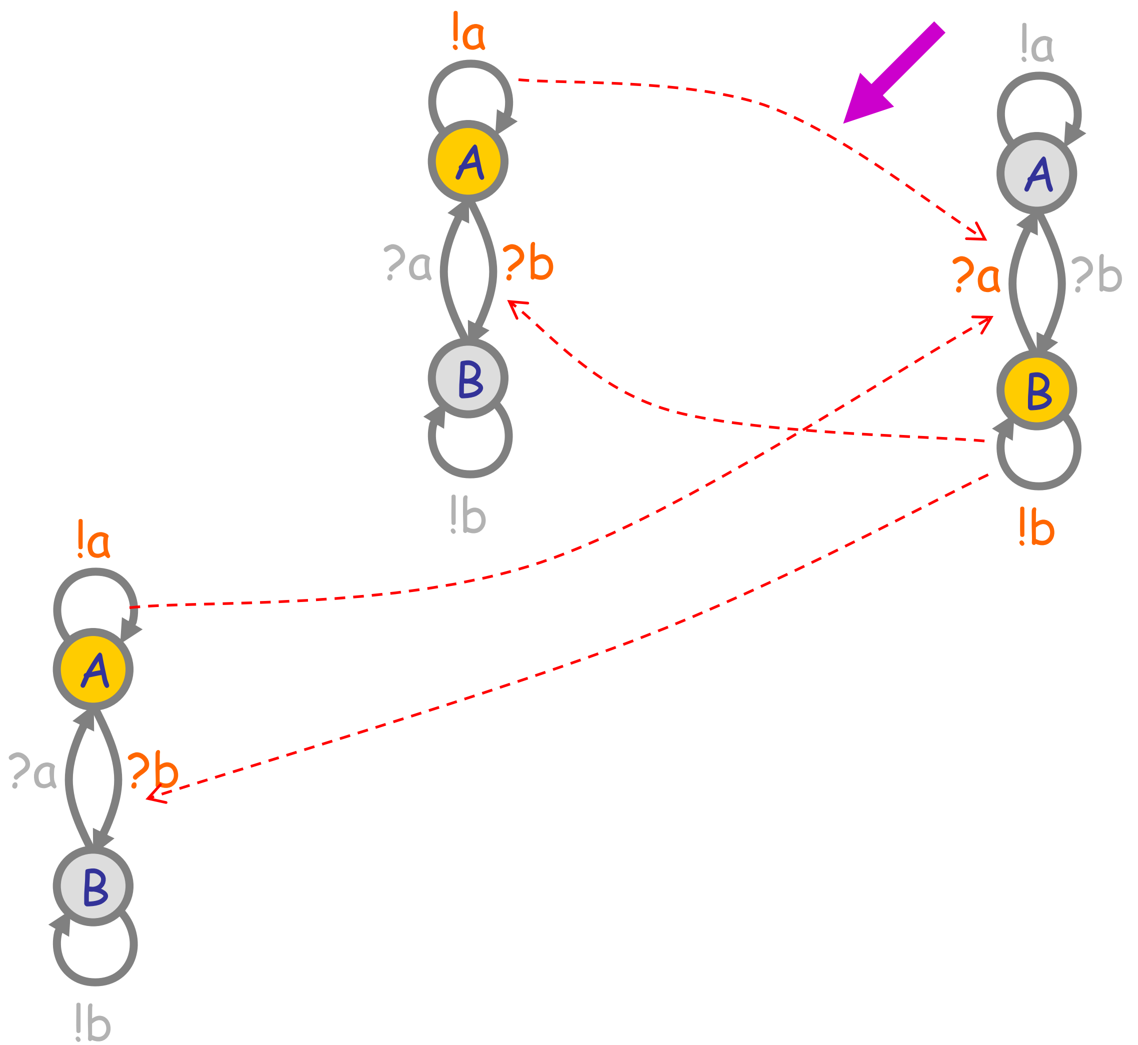
Analyzable by standard Markovian techniques, by first computing the "product automaton" to obtain the underlying finite Markov transition system. [Buchholz]

Interactions have rates. Actions DO NOT have rates.

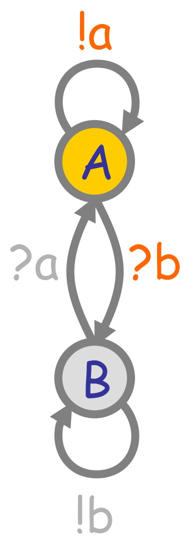
Interactions in a Population

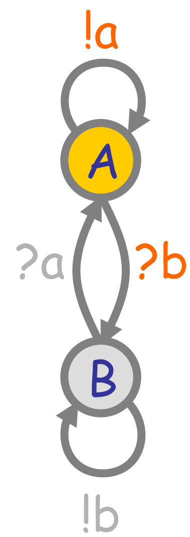


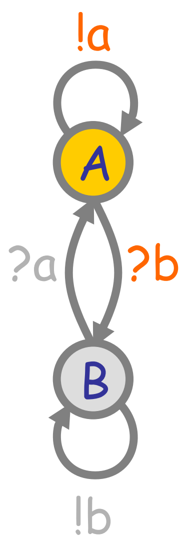
Interactions in a Population



Interactions in a Population

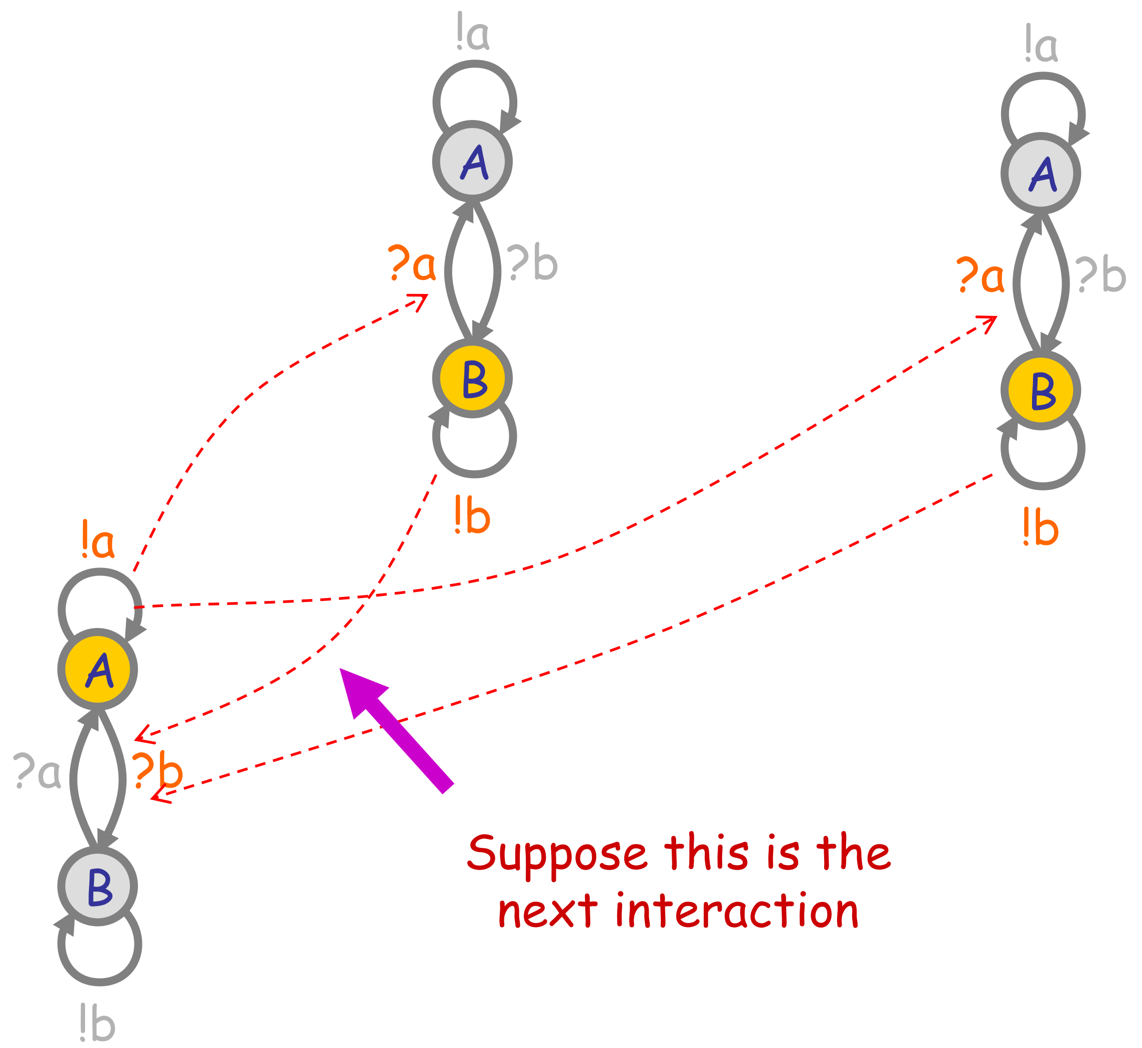






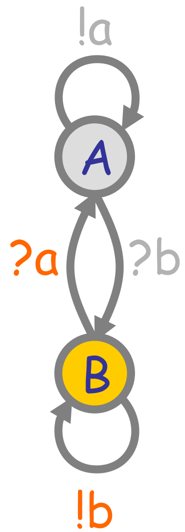
All-A stable population

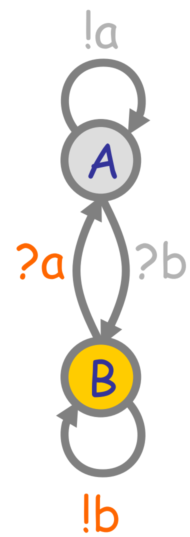
Interactions in a Population (2)

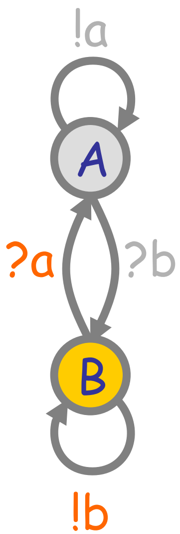


Suppose this is the next interaction

Interactions in a Population (2)



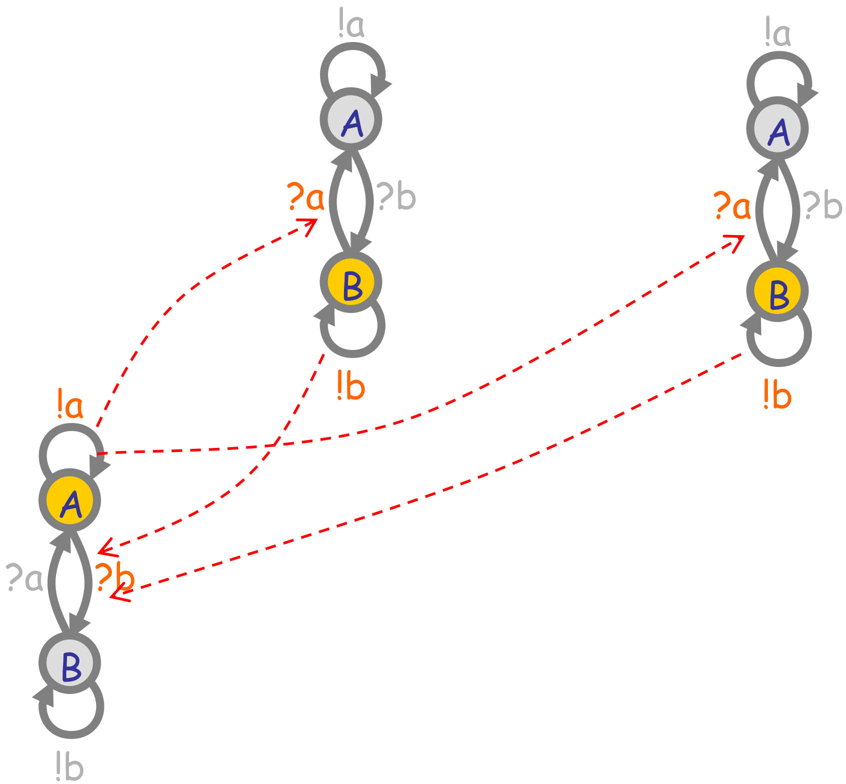


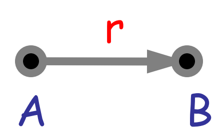


All-B stable population

Nondeterministic population behavior ("multistability")

CTMC Semantics



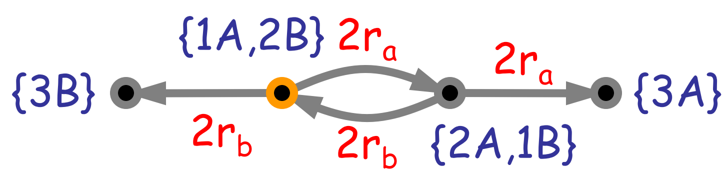


CTMC
(homogeneous) Continuous Time Markov Chain
- directed graph with no self loops
- nodes are system states
- arcs have transition rates

Probability of holding in state A:

$$Pr(H_A > t) = e^{-rt}$$

in general, $Pr(H_A > t) = e^{-Rt}$ where R is the sum of all the exit rates from A



CTMC

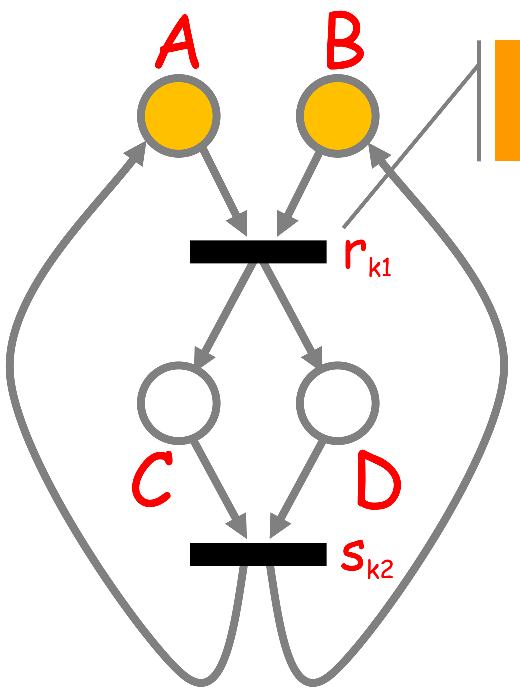
Chemistry vs. Automata

A process algebra (chemistry)

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

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

Does A become C or D?

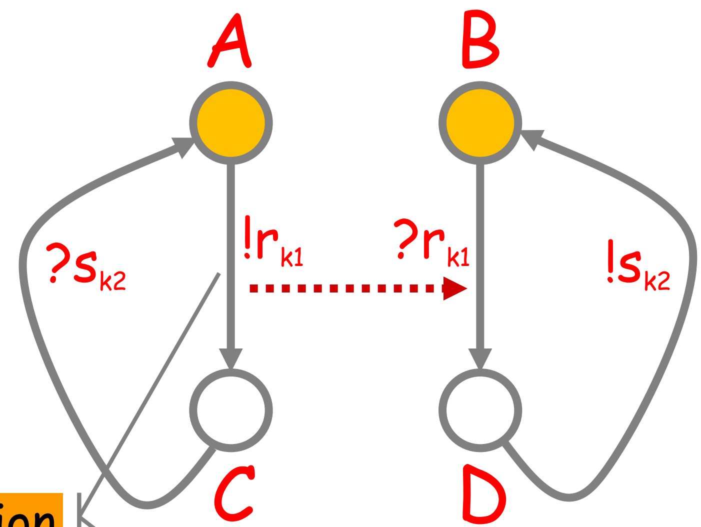


Reaction oriented

1 line per reaction

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

A different process algebra (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!

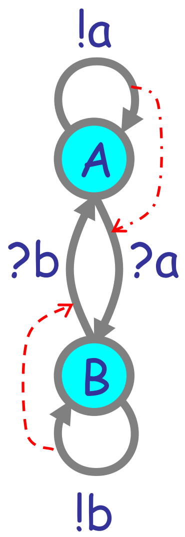
The same "model"

Maps to a CTMC	Maps to a CTMC

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

Emergent Collective Behavior

Groupies and Celebrities



Celebrity

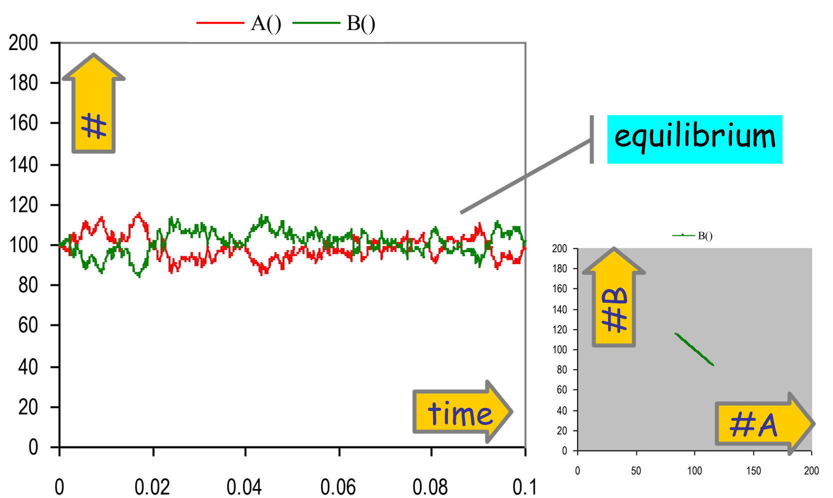
(does not want to be like somebody else)

```
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())
```

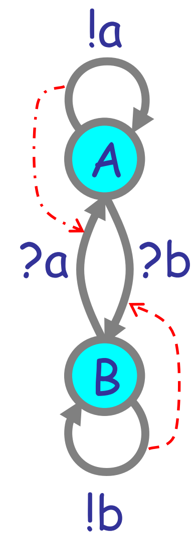
a@1.0

b@1.0

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.



Groupie

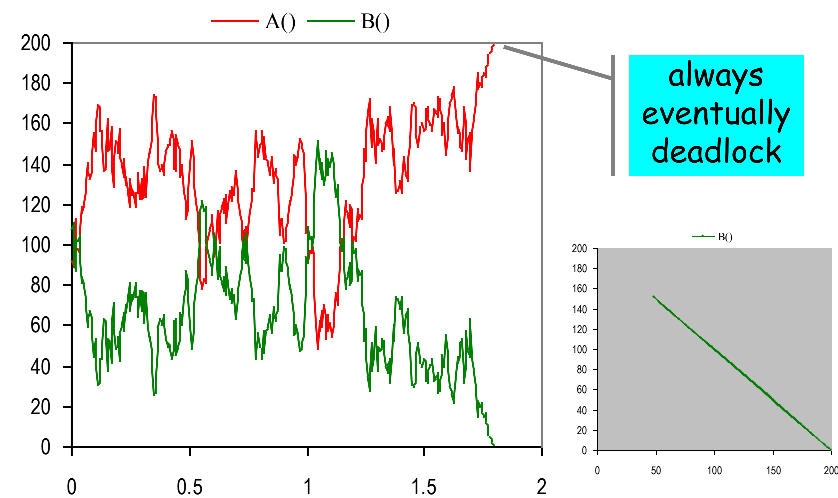
(wants to be like somebody different)

```
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 groupies:

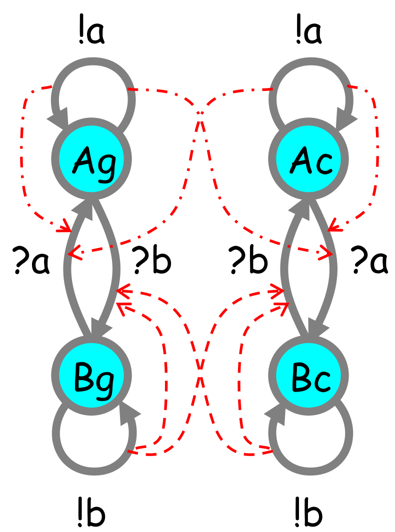


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

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

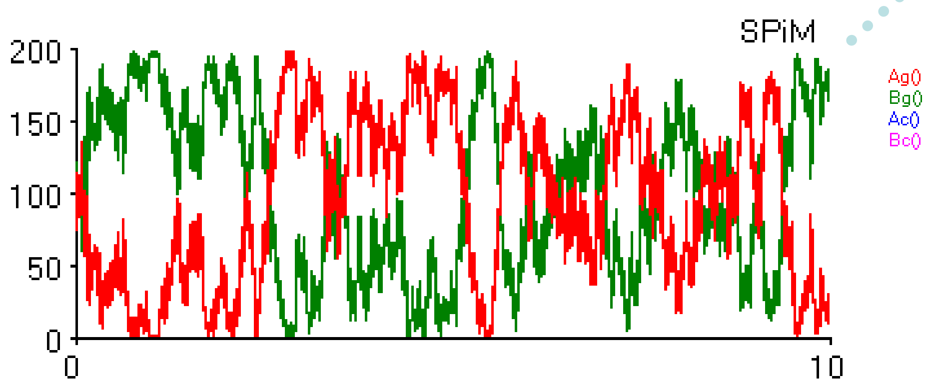
Many Groupies



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

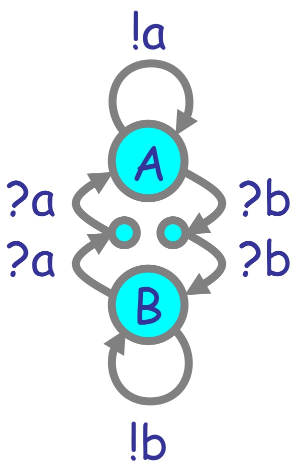


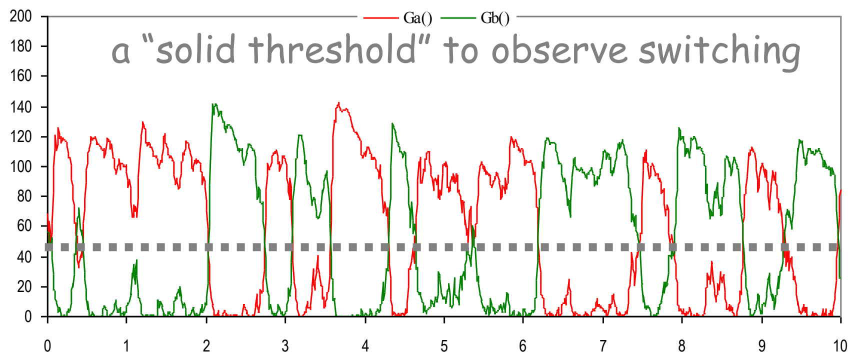
A tiny bit of "noise" can make a huge difference

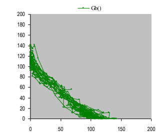
Regularity can arise not far from chaos

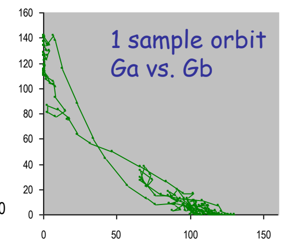
Hysteric Groupies

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

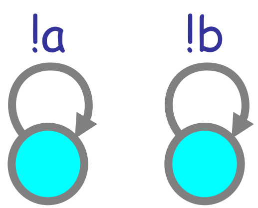








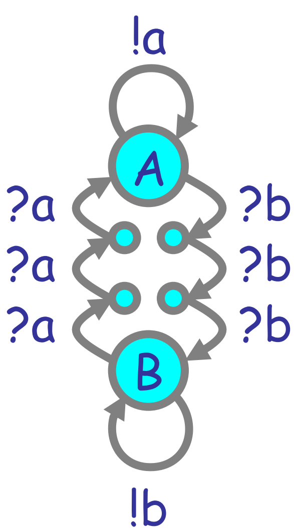

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

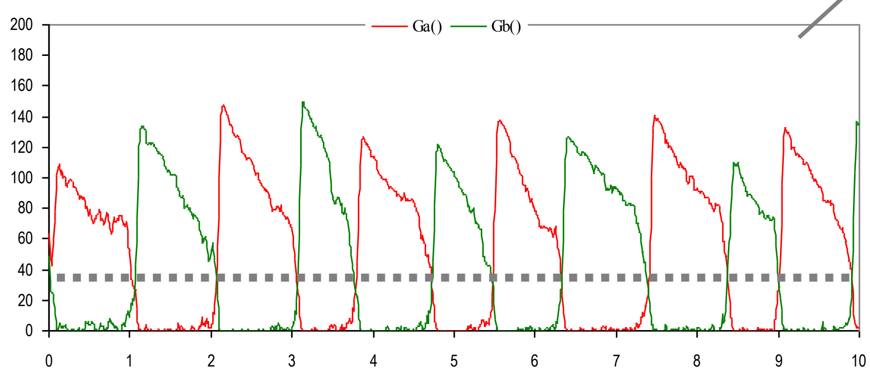


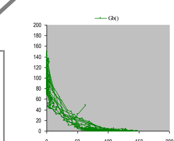
(With doping to break deadlocks)

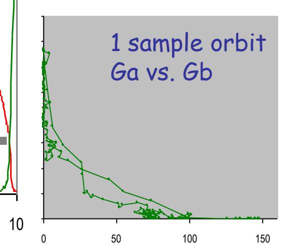
N.B.: It will not oscillate without doping (noise)

"regular" oscillation





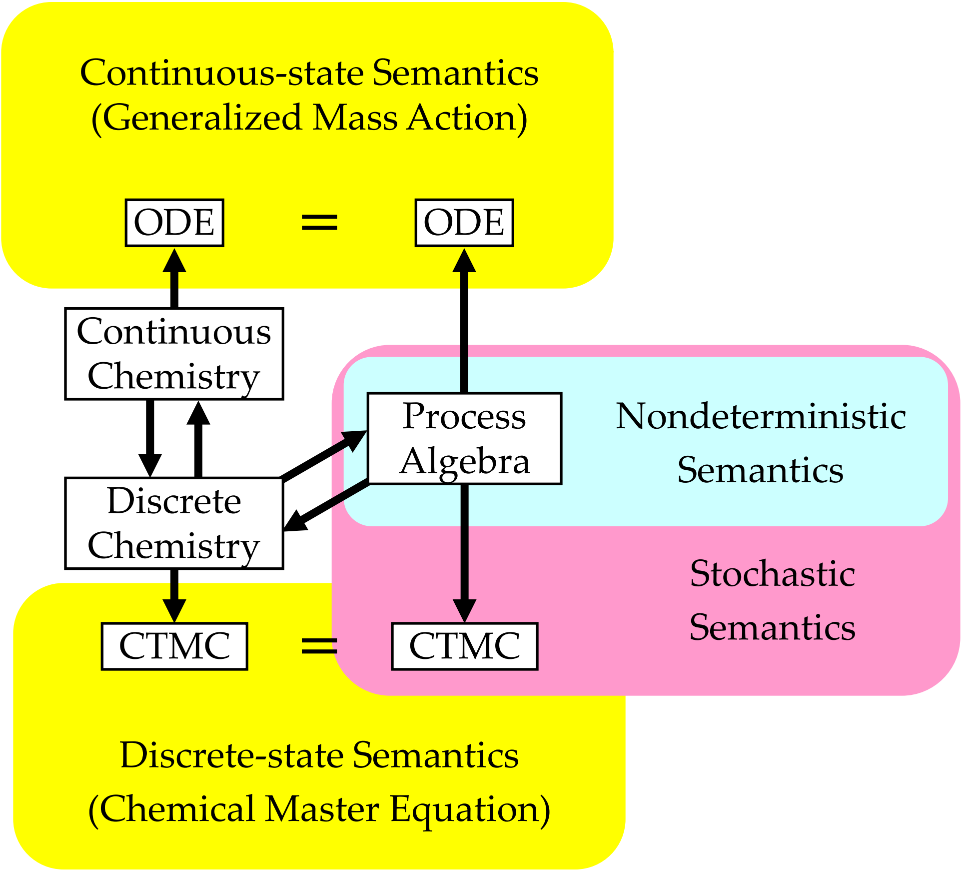





```
directive sample 10.0 1000
directive plot Ga(); Gb()

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 Sides of Chemistry

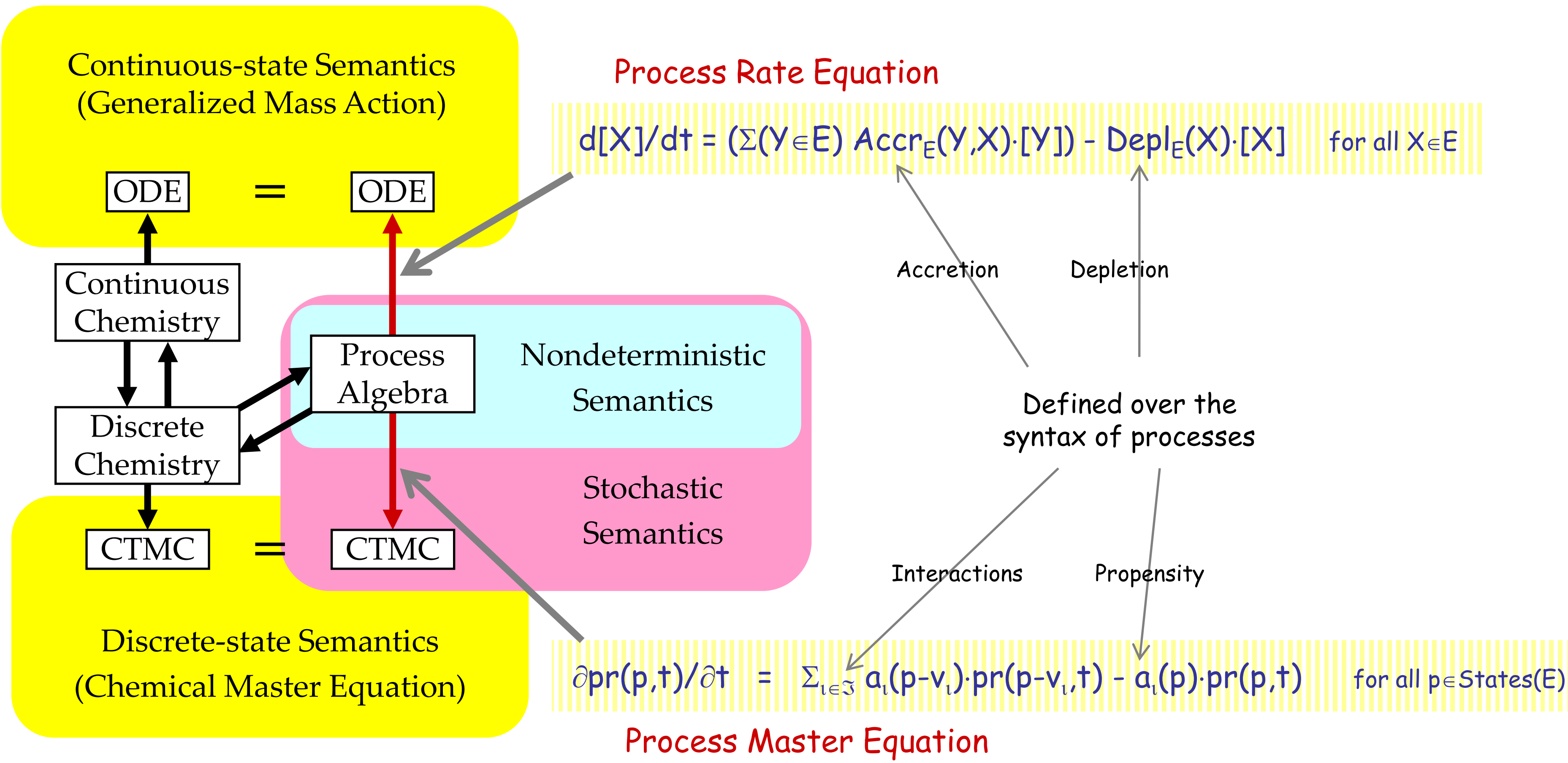


These diagrams commute
(for the "Chemical Ground Form" process algebra).

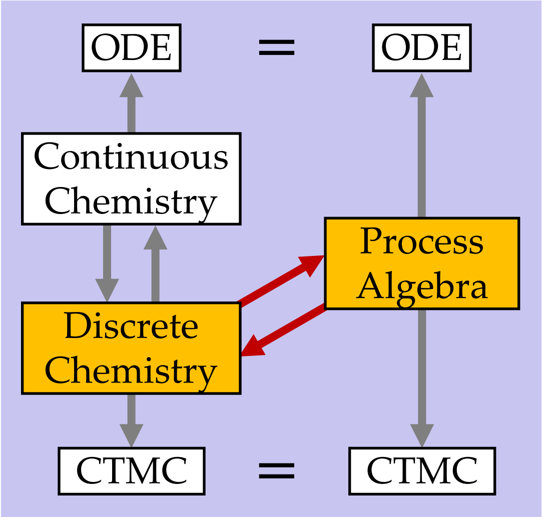
L. Cardelli: "On Process Rate Semantics" (TCS)

L. Cardelli: "A Process Algebra Master Equation" (QEST'07)

Quantitative Process Semantics



Stochastic Processes & Discrete Chemistry



Chemical Reactions

Elementary Reactions:

$A \to^r B_1 + \ldots + B_n$ ($n \geq 0$) Unary Reaction

$A_1 + A_2 \to^r B_1 + \ldots + B_n$ ($n \geq 0$) Hetero Reaction

$A + A \to^r B_1 + \ldots + B_n$ ($n \geq 0$) Homeo Reaction

Reaction kinetics: [A] = concentration of A

$d[A]/dt = -r[A]$ Exponential Decay

$d[A_i]/dt = -r[A_1][A_2]$ Mass Action Law

$d[A]/dt = -2r[A]^2$ Mass Action Law

(assuming $A \neq B_i \neq A_j$ for all i,j)

No other reactions!

JOURNAL OF CHEMICAL PHYSICS VOLUME 113, NUMBER 1

The chemical Langevin equation

Daniel T. Gillespie[a)]
Research Department, Code 4T4100D, Naval Air Warfare Center, China Lake, California 93555

Genuinely *trimolecular* reactions do not physically occur in dilute fluids with any appreciable frequency. *Apparently* trimolecular reactions in a fluid are usually the combined result of two bimolecular reactions and one monomolecular reaction, and involve an additional short-lived species.

Chapter IV: Chemical Kinetics
[David A. Reckhow , CEE 572 Course]
... reactions may be either elementary or non-elementary. Elementary reactions are those reactions that occur exactly as they are written, without any intermediate steps. These reactions almost always involve just one or two reactants. ... Non-elementary reactions involve a series of two or more elementary reactions. Many complex environmental reactions are non-elementary. In general, reactions with an overall reaction order greater than two, or reactions with some non-integer reaction order are non-elementary.

THE COLLISION THEORY OF REACTION RATES
www.chemguide.co.uk
The chances of all this happening if your reaction needed a collision involving more than 2 particles are remote. All three (or more) particles would have to arrive at exactly the same point in space at the same time, with everything lined up exactly right, and having enough energy to react. That's not likely to happen very often!

Reactions have rates. Molecules *do not* have rates.

Trimolecular reactions:

$A + B + C \to^r D$

the measured "r" is an (imperfect) aggregate of e.g.:

$A + B \leftrightarrow AB$

$AB + C \to D$

Enzymatic reactions:

$S \xrightarrow{E}{}^r P$

the "r" is given by Michaelis-Menten (approximated steady-state) laws:

$E + S \leftrightarrow ES$

$ES \to P + E$

Chemical Ground Form (CGF)

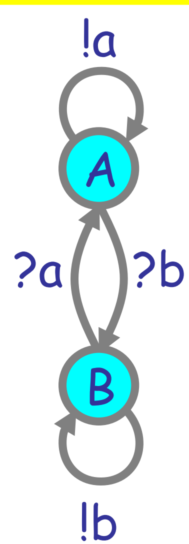
$E ::= 0 \;:\; X{=}M, E$		Reagents
$M ::= 0 \;:\; \pi;P \oplus M$		Molecules
$P ::= 0 \;:\; X \mid P$		Solutions
$\pi ::= \tau_{(r)} \;:\; ?a_{(r)} \;:\; !a_{(r)}$		Interactions (delay, input, output)
$CGF ::= E,P$		Reagents plus Initial Conditions

A stochastic subset of CCS
(no values, no restriction)

Interacting Automata
+ dynamic forking

(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 ($M\oplus 0 = 0\oplus M = M$)
Each X in E is a distinct *species*
Each name a is assigned a fixed rate r: $a_{(r)}$



Ex: Interacting Automata
(= finite-control CGFs: they use "|" only in initial conditions):

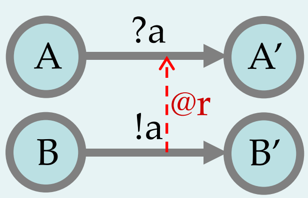
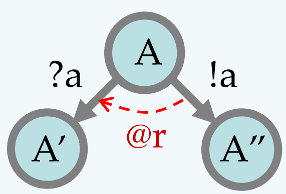
$A = !a;A \oplus ?b;B$ — Automaton in state A

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

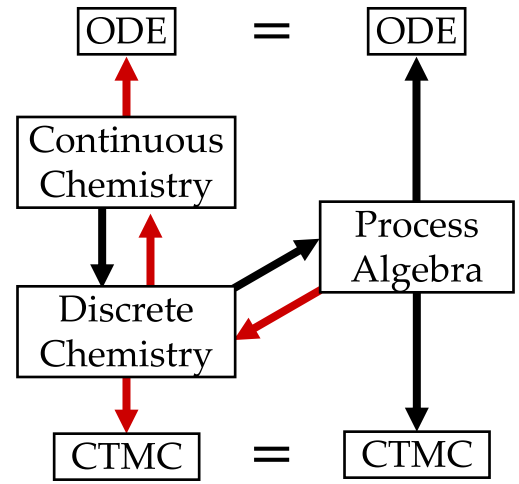
$A|A|B|B$ — Initial conditions: 2A and 2B

Automata to Chemistry

V = interaction volume
N_A = Avogadro's number

Automata	→ Discrete Chemistry (molecule counts)	→ Continuous Chemistry (concentrations) $\gamma = N_A V$
initial states A | A | ... | A	initial quantities $\#A_0$	initial concentrations $[A]_0$ with $[A]_0 = \#A_0/\gamma$
	$A \overset{r}{\dashrightarrow} A'$	$A \rightarrow^{k} A'$ with $k = r$
	$A+B \overset{r}{\dashrightarrow} A'+B'$	$A+B \rightarrow^{k} A'+B'$ with $k = r\gamma$
	$A+A \overset{2r}{\dashrightarrow} A'+A''$	$A+A \rightarrow^{2k} A'+A''$ with $k = r\gamma/2$
	↓ CTMC	↓ ODE ($[A]^\bullet \equiv d[A]/dt$ change of concentration over time)

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

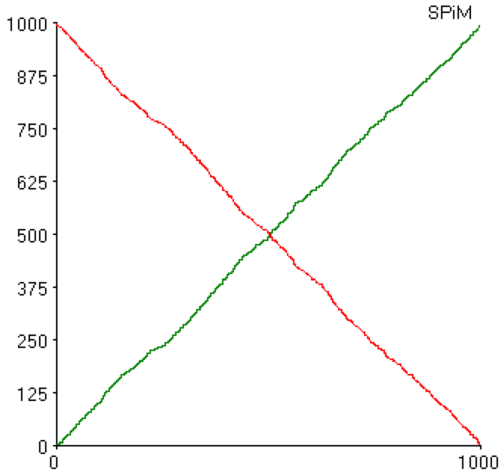


Automata are n^2 more compact!

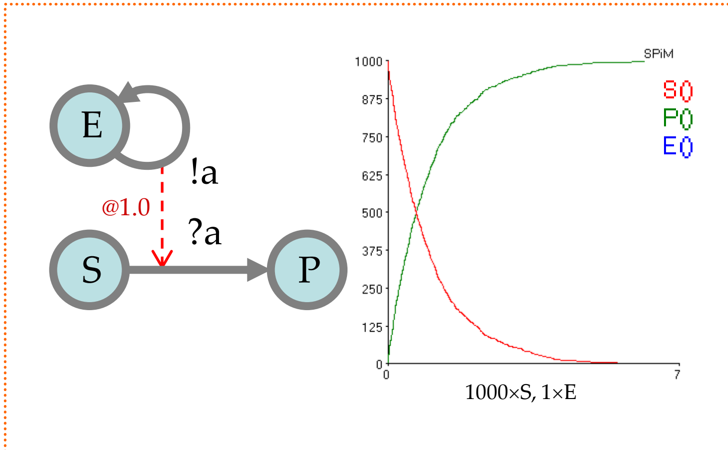
Examples of Chemical Kinetics by Interacting Automata

Zero-Order Regime

Or: build me a population like this:



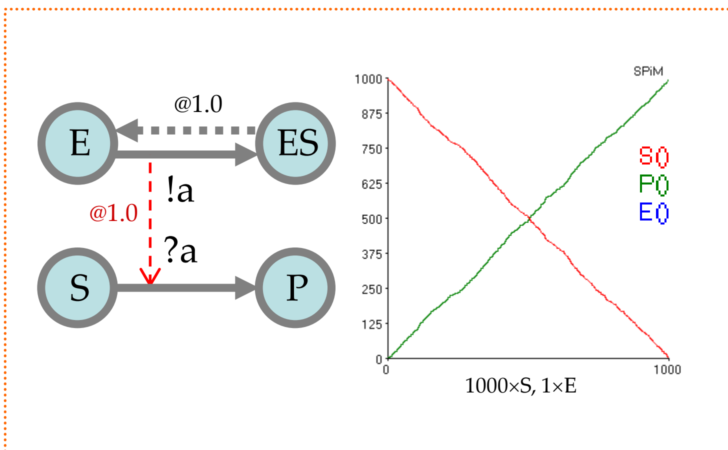
Second-order and Zero-order Regime



E+S →^{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
d[S]/dt = -r[E][S]

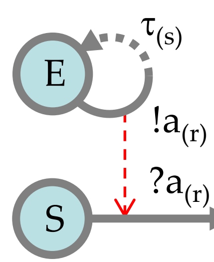


E+S →^{r} ES+P

ES →^{s} E

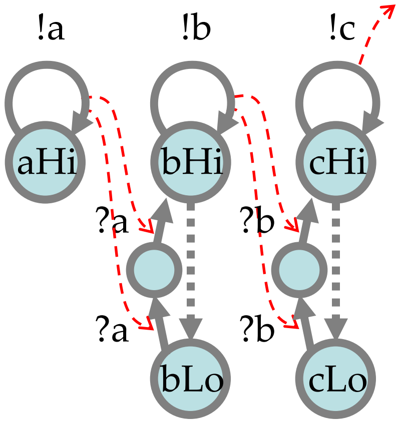
```
directive sample 1000.0
directive plot S(); P(); E()

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())
```

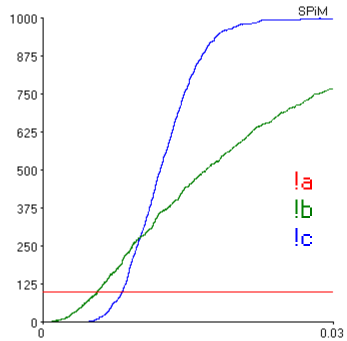
Zero-Order Regime
d[S]/dt ≅ -1 (by assuming d[ES]/dt=0)



Notation

Cascades



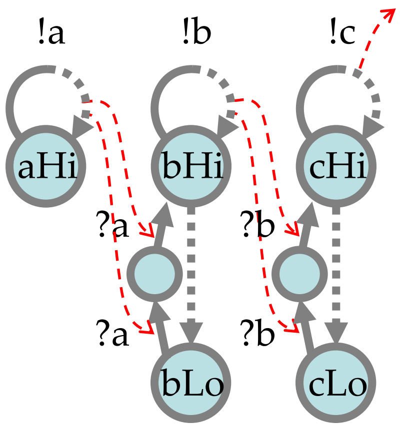


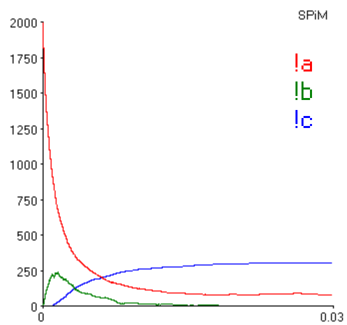
100×aHi, 1000×bLo, 1000×cLo, rates=1.0

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()
```





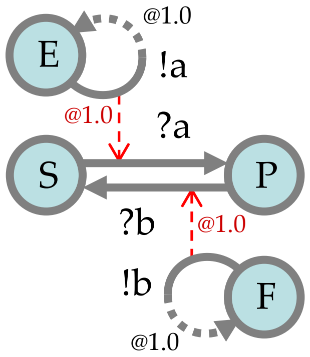
2000×aHi, 1000×bLo, 1000×cLo, rates=1.0

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

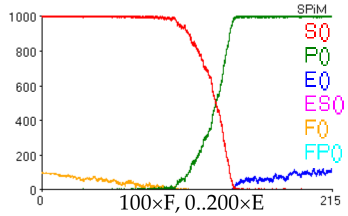
aHi = max ⇒ cHi = 1/3 max

```
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()
```

Ultrasensitivity



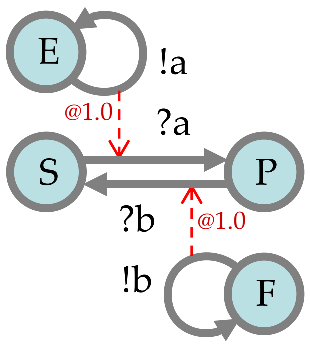
$E+S \to ES+P$
$F+P \to FP+S$
$ES \to E$
$FP \to P$



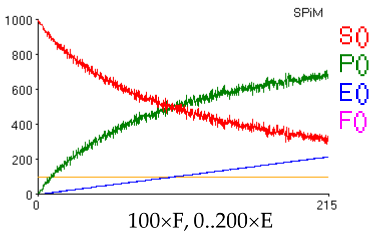

```
directive sample 215.0
directive plot S(); P(); E(); ES(); F(); FP()

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.



$E+S \to E+P$
$F+P \to F+S$



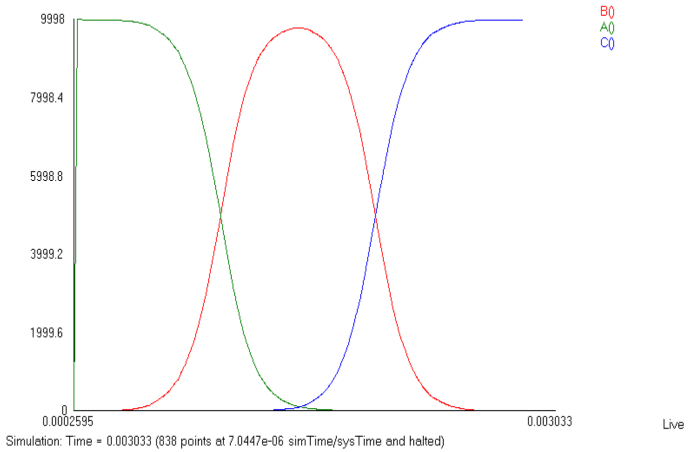

```
directive sample 215.0 1000
directive plot S(); P(); E(); F()

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

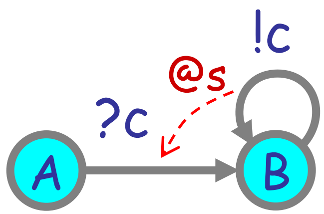
No switching behavior

Waves

Or: build me a population like this:



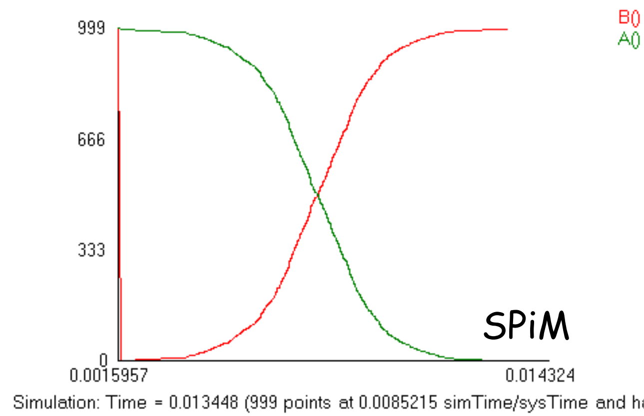
Nonlinear Transition (NLT)



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

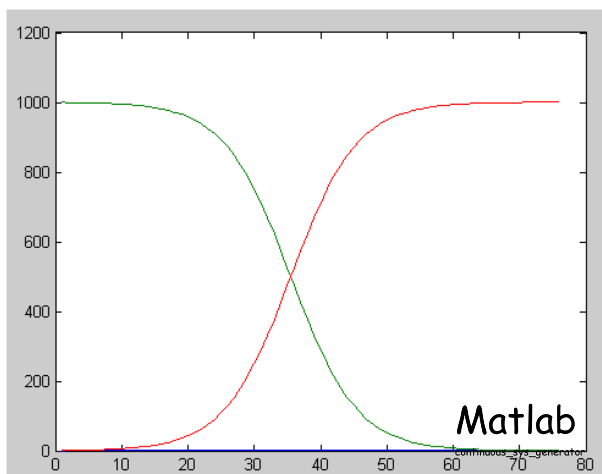
$A+B \rightarrow^{s} B+B$

$d[A]/dt = -s[A][B]$
$d[B]/dt = s[A][B]$



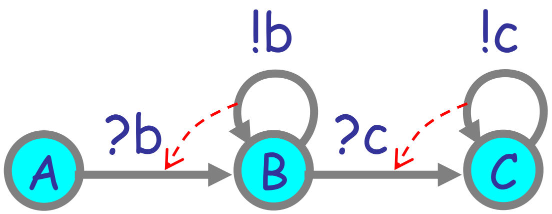

```
directive sample 0.02 1000
directive plot B(); A()

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



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

```
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())
```

$A = ?b_{(1)};B$

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

$C = !c_{(1)};C$

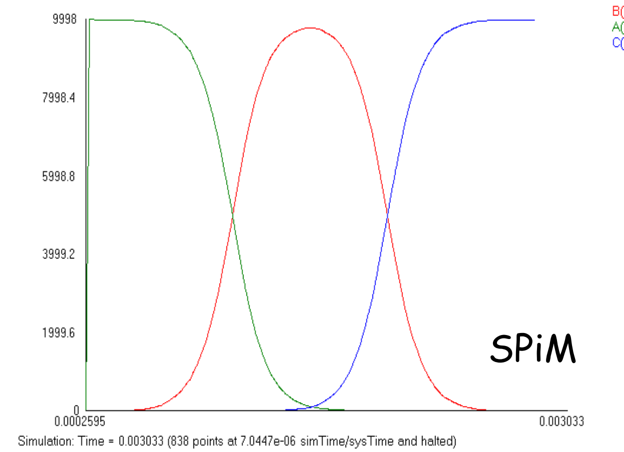
$A+B \to^1 B+B$

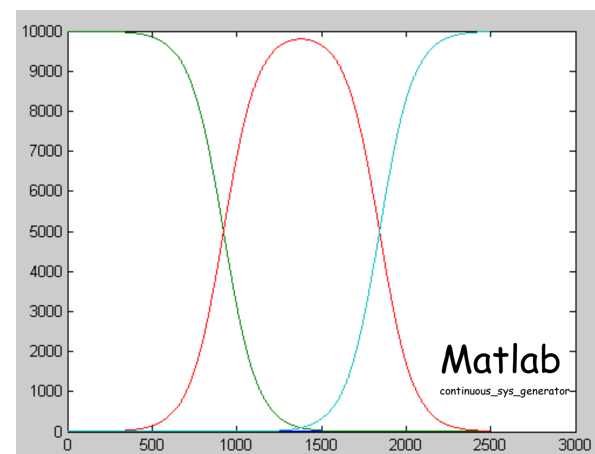
$B+C \to^1 C+C$

$d[A]/dt = -[A][B]$

$d[B]/dt = [A][B]-[B][C]$

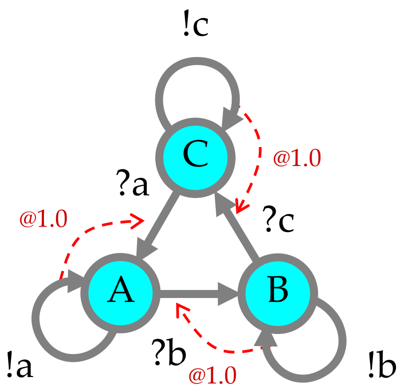
$d[C]/dt = [B][C]$

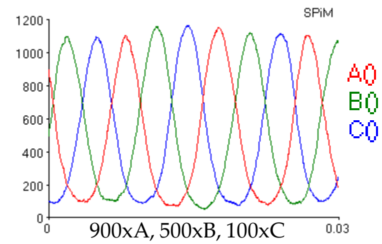





```
interval/step [0:0.000001:0.0025]
(A)    dx1/dt = -x1*x2              10000.0
(B)    dx2/dt = x1*x2 - x2*x3       1.0
(C)    dx3/dt = x2*x3               1.0
```

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())
```

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

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

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

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

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

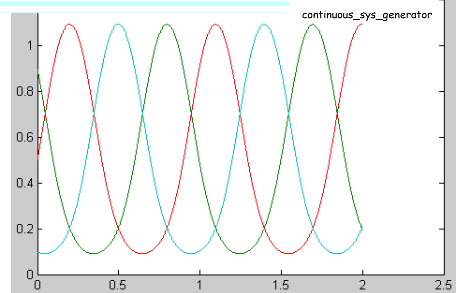
$C+A \rightarrow^{s} A+A$

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

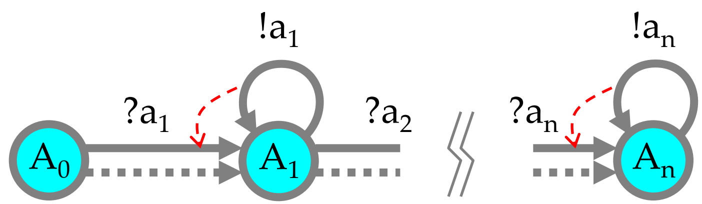
$[B]^{\bullet} = -s[B][C]+s[A][B]$

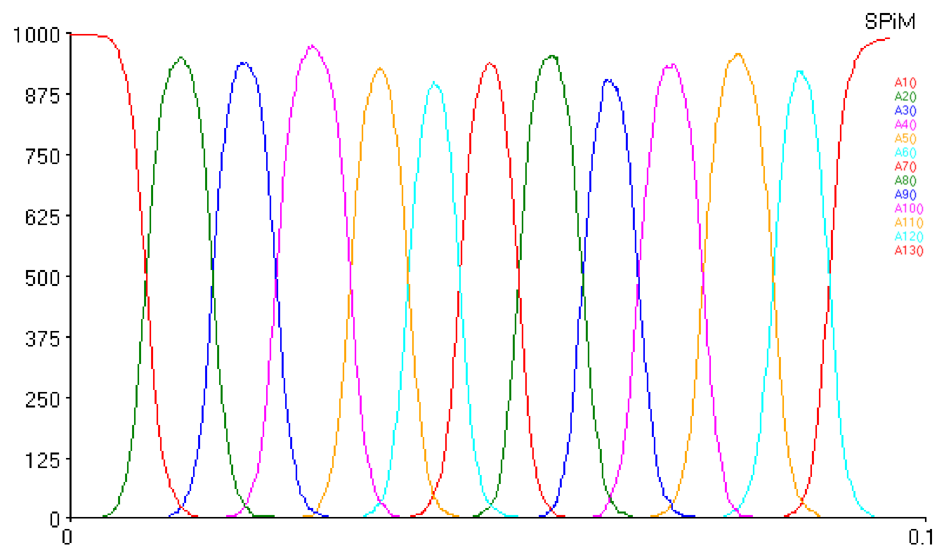
$[C]^{\bullet} = -s[C][A]+s[B][C]$

```
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
```



NLTs in Series: Soliton Propagation

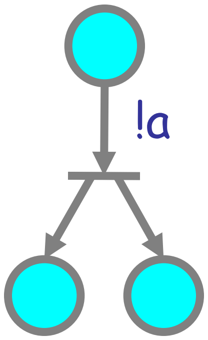




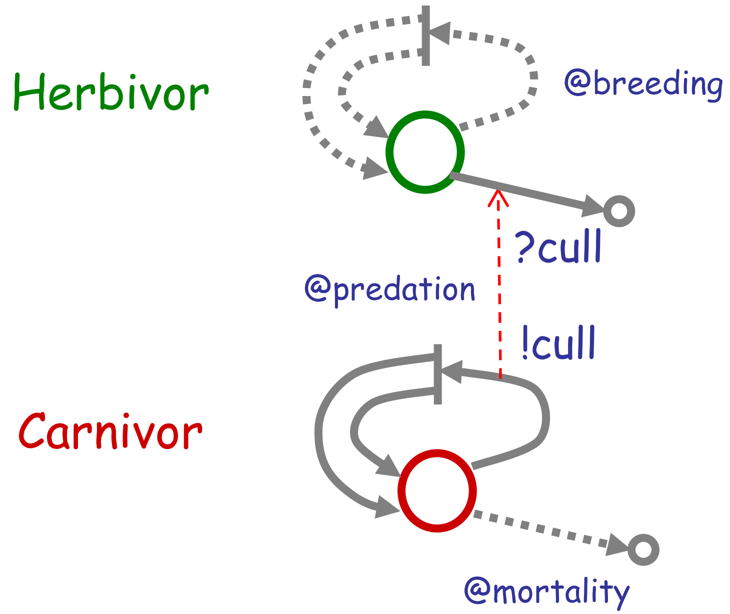

```
directive sample 0.1 1000
directive plot A1(); A2(); A3(); A4(); A5(); A6(); A7(); A8();
A9(); A10(); A11(); A12(); A13()

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()
```

Lotka-Volterra

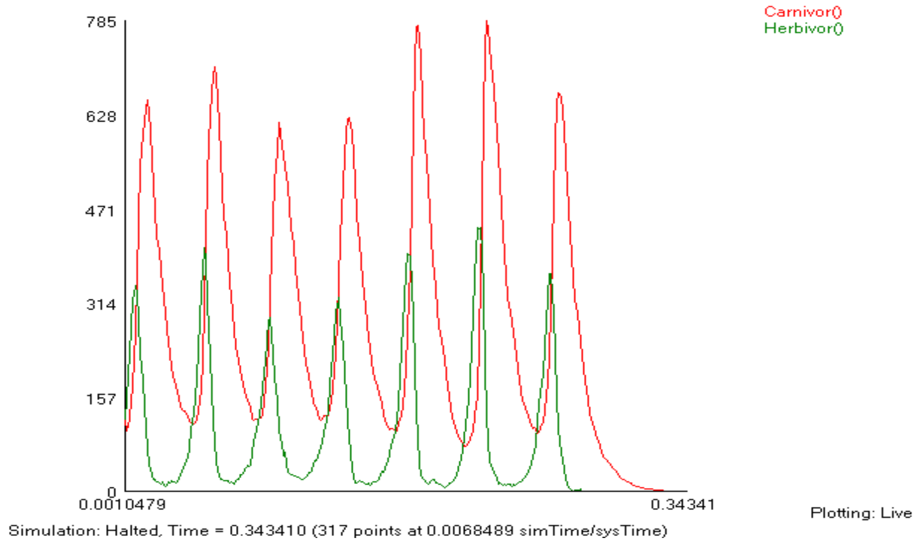
Or: beyond automata



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

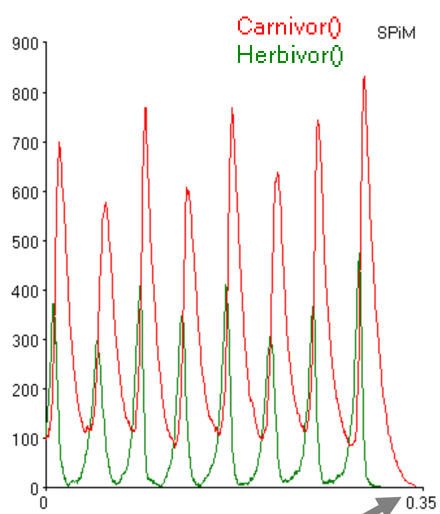
$H = \tau_b; (H|H) \oplus ?c_{(p)};0$
$C = \tau_m;0 \oplus !c_{(p)};(C|C)$
$\#H_0, \#C_0$

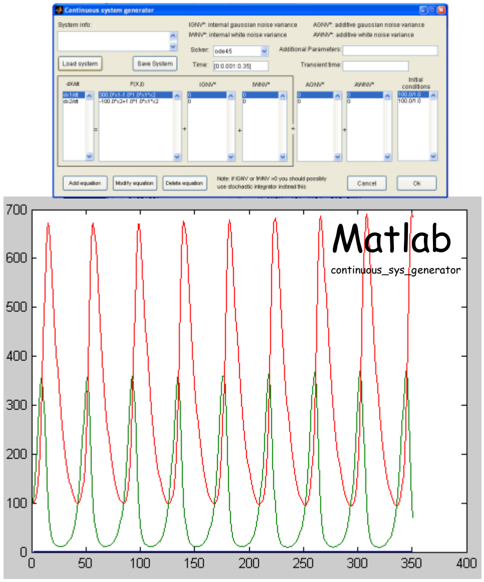
$H \rightarrow^b H + H$
$C \rightarrow^m 0$
$H + C \rightarrow^{p\gamma} C + C$
$[H]_0 = \#H_0/\gamma$
$[C]_0 = \#C_0/\gamma$

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

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

```
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()
```





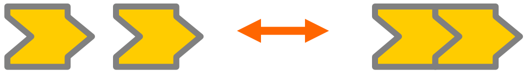
Extinction

No extinction

Which one is the “right prediction”?

Biochemistry

Or: Interaction + Complexation



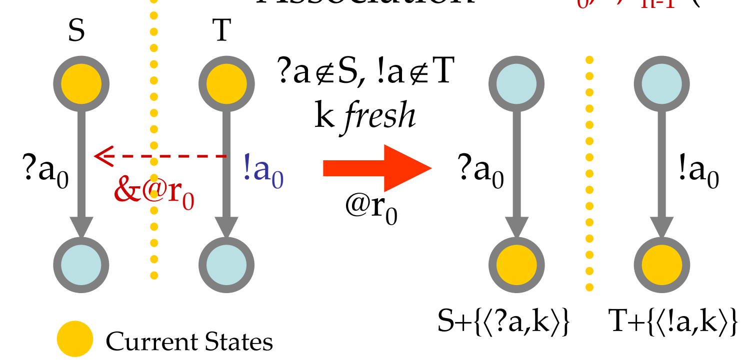
Without complexation, many "finite" combinatorial systems can only be expressed by an infinite number of elementary chemical reactions.

Polyautomata

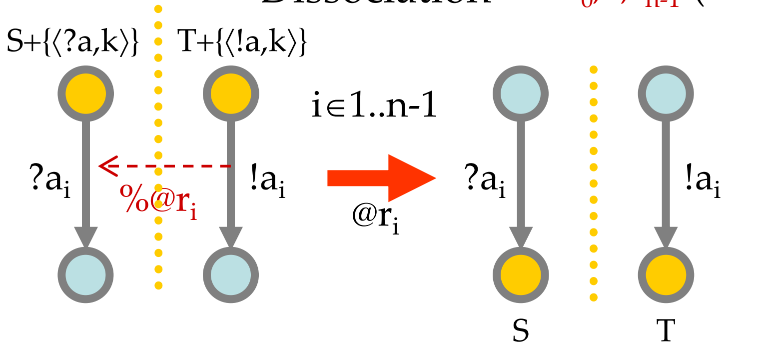
Two new operations

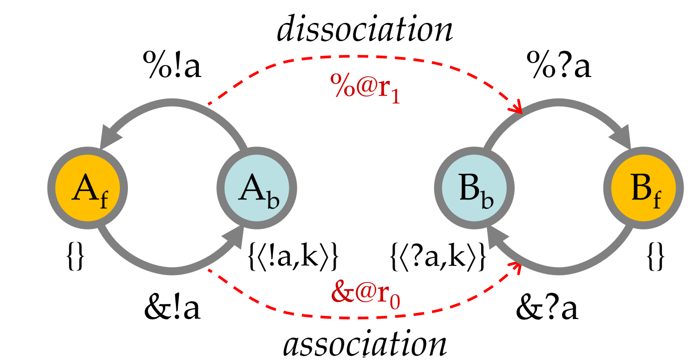
the current states S,T carry an "associaton history"

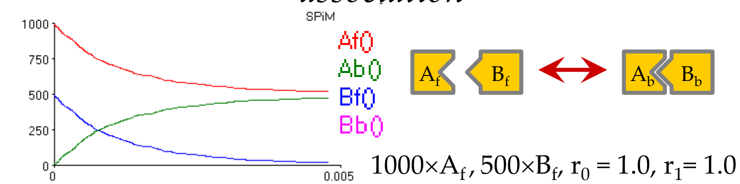
Association $a@r_0,..,r_{n-1}$ $(n\geq 2)$



Dissociation $a@r_0,..,r_{n-1}$ $(n\geq 2)$



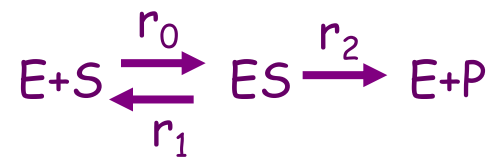


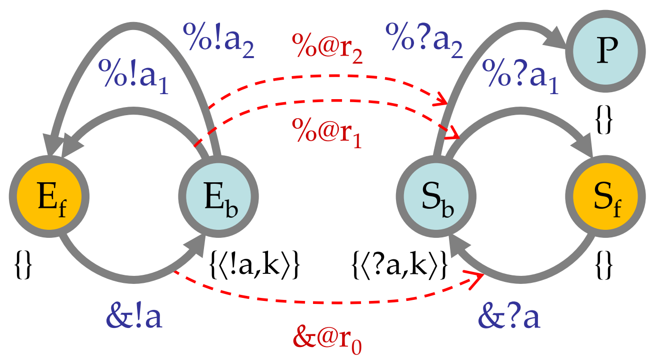


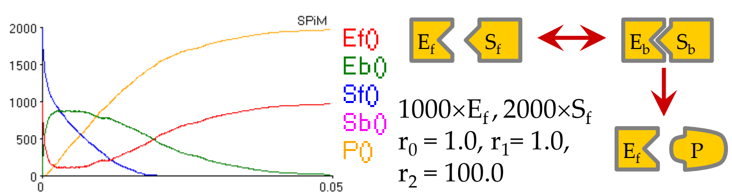
Can be encoded in π-calculus (and SPiM) by bound-output/bound-input.

```
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())
```

(Compositional) Enzyme Kinetics







$a@r_0,r_1,r_2$

```
directive sample 0.05 1000
directive plot Ef(); Eb(); Sf(); Sb(); P()

val k1 = 1.0    val km1 = 1.0    val k2 = 100.0
new a@k1:chan(chan,chan)

let P() = ()

let Ef() =
   (new n@km1:chan new m@k2:chan
    run !a(n,m); Eb(n,m))
and Eb(n:chan,m:chan) =
  do !n; Ef() or !m; Ef()

let Sf() = ?a(n,m); Sb(n,m)
and Sb(n:chan,m:chan) =
  do ?n; Sf() or ?m; P()

run (1000 of Ef() | 2000 of Sf())
```

Bidirectional

Polymerization

Polymerization is iterated complexation.

```
new c@μ  new stop@1.0
```

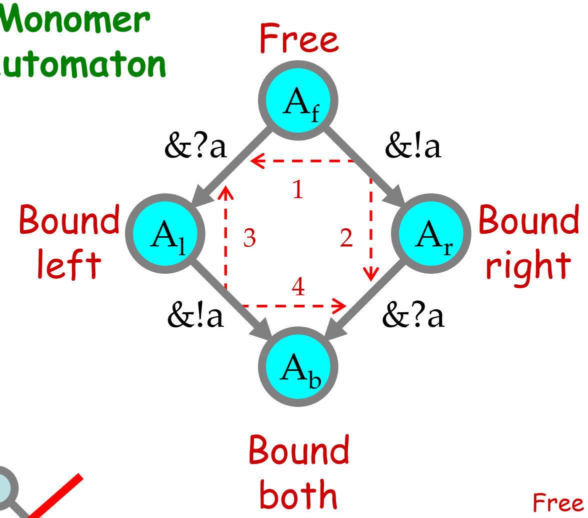
A_{free} =
  $!c(^{\nu}rht_{\lambda})$; $A_{brht}(rht)$) +
  $?c(lft)$; $A_{blft}(lft)$)

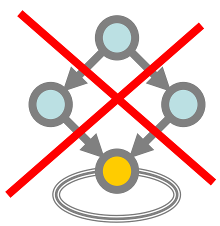
$A_{blft}(lft)$ =
  $!c(^{\nu}rht_{\lambda})$; $A_{bound}(lft,rht)$)

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

$A_{bound}(lft,rht)$ = ?stop

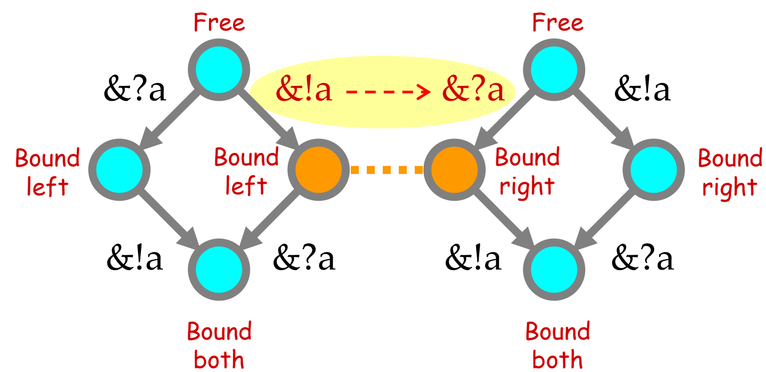
Monomer Automaton

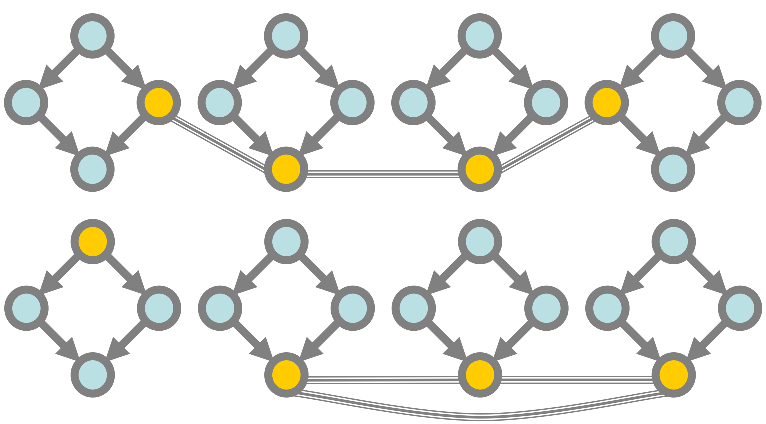




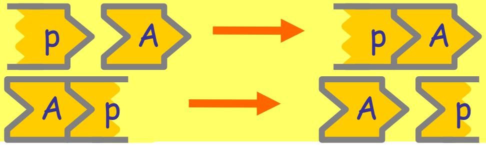
Polyautomata

Bound output $!c(^{\nu}r)$ 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())
```



Actin-like
Poly/Depolymerization

new c@μ

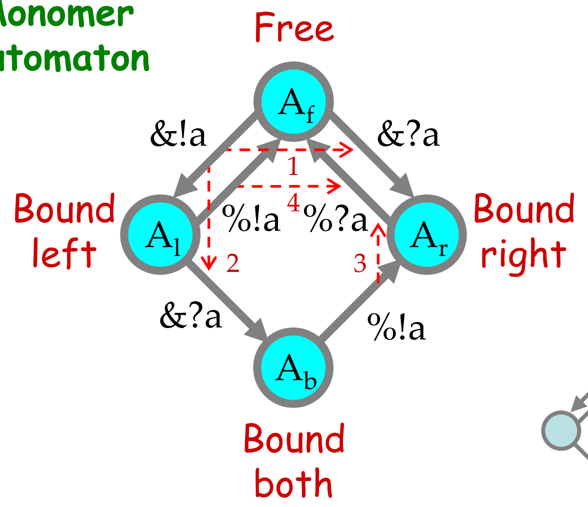
A_{free} =
  !c(${}^{\nu}lft_{\lambda}$); A_{blft}(lft)) +
  ?c(rht); A_{brht}(rht)

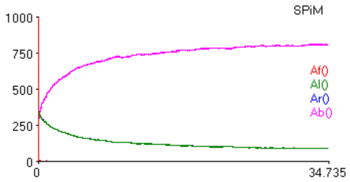
A_{blft}(lft) =
  !!lft; A_{free} +
  ?c(rht); A_{bound}(lft,rht)

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

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

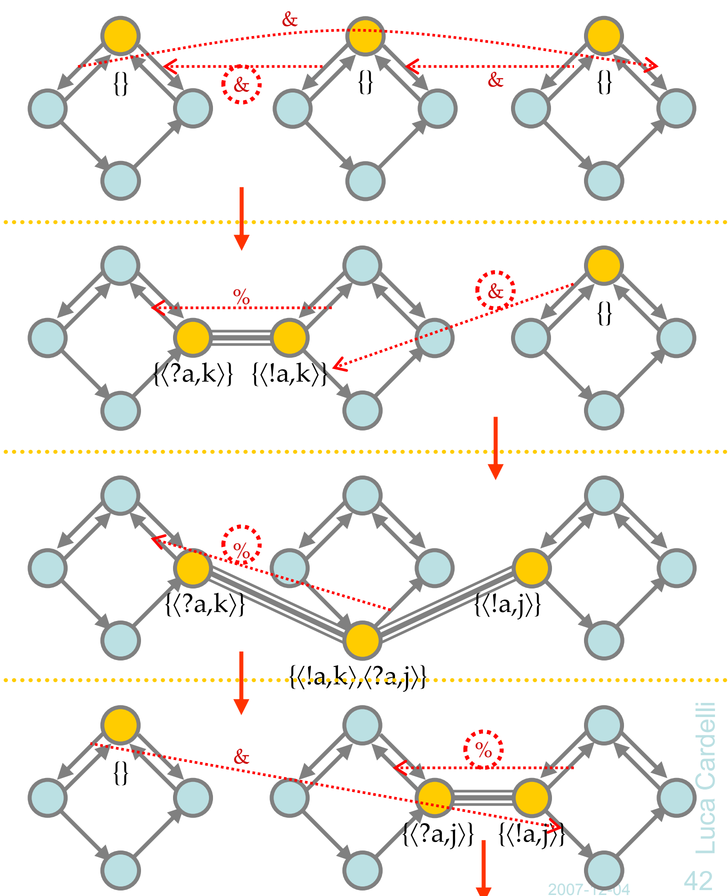
Monomer Automaton





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

Compactness of Representation

- E_n has 2n variables (nodes) and 2n terms (arcs).
- $Ch(E_n)$ has 2n species and n^2 reactions.
- The stoichiometric matrix has size $2n \cdot n^2 = 2n^3$.
- The ODEs have 2n variables and $2n(n+n) = 4n^2$ terms
  (number of variables times number of accretions plus depletions when sums are distributed)

E_3

$X_0 = ?a_{(r)};X_1$
$X_1 = ?a_{(r)};X_2$
$X_2 = ?a_{(r)};X_0$
$Y_0 = !a_{(r)};Y_1$
$Y_1 = !a_{(r)};Y_2$
$Y_2 = !a_{(r)};Y_0$

$Ch(E_3)$

$a_{00}: X_0+Y_0 \to^r X_1+Y_1$
$a_{01}: X_0+Y_1 \to^r X_1+Y_2$
$a_{02}: X_0+Y_2 \to^r X_1+Y_0$
$a_{10}: X_1+Y_0 \to^r X_2+Y_1$
$a_{11}: X_1+Y_1 \to^r X_2+Y_2$
$a_{12}: X_1+Y_2 \to^r X_2+Y_0$
$a_{20}: X_2+Y_0 \to^r X_0+Y_1$
$a_{21}: X_2+Y_1 \to^r X_0+Y_2$
$a_{22}: X_2+Y_2 \to^r X_0+Y_0$

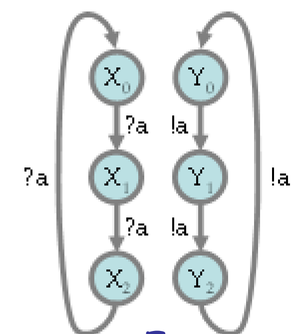
StoichiometricMatrix($Ch(E_3)$)

	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}	a_{20}	a_{21}	a_{22}
X_0	-1	-1	-1				+1	+1	+1
X_1	+1	+1	+1	-1	-1	-1			
X_2				+1	+1	+1	-1	-1	-1
Y_0	-1		+1	-1		+1	-1		+1
Y_1	+1	-1		+1	-1		+1	-1	
Y_2		+1	-1		+1	-1		+1	-1

ODE(E_3)

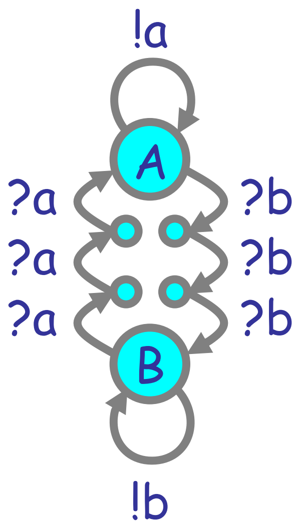
$d[X_0]/dt = -r[X_0][Y_0] - r[X_0][Y_1] - r[X_0][Y_2] + r[X_2][Y_0] + r[X_2][Y_1] + r[X_2][Y_2]$
$d[X_1]/dt = -r[X_1][Y_0] - r[X_1][Y_1] - r[X_1][Y_2] + r[X_0][Y_0] + r[X_0][Y_1] + r[X_0][Y_2]$
$d[X_2]/dt = -r[X_2][Y_0] - r[X_2][Y_1] - r[X_2][Y_2] + r[X_1][Y_0] + r[X_1][Y_1] + r[X_1][Y_2]$
$d[Y_0]/dt = -r[X_0][Y_0] - r[X_1][Y_0] - r[X_2][Y_0] + r[X_0][Y_2] + r[X_1][Y_2] + r[X_2][Y_2]$
$d[Y_1]/dt = -r[X_0][Y_1] - r[X_1][Y_1] - r[X_2][Y_1] + r[X_0][Y_0] + r[X_1][Y_0] + r[X_2][Y_0]$
$d[Y_2]/dt = -r[X_0][Y_2] - r[X_1][Y_2] - r[X_2][Y_2] + r[X_0][Y_1] + r[X_1][Y_1] + r[X_2][Y_1]$

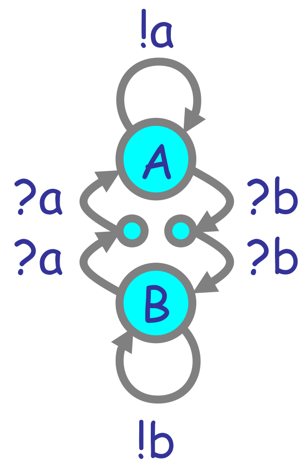
=

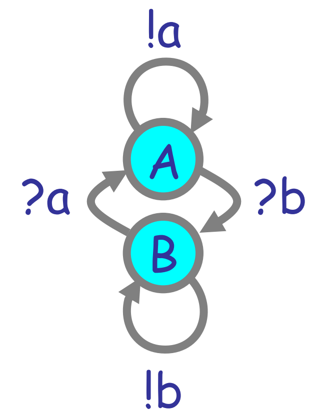


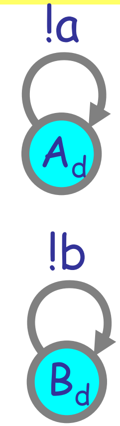
E_3

Continuous vs. Discrete Kinetics

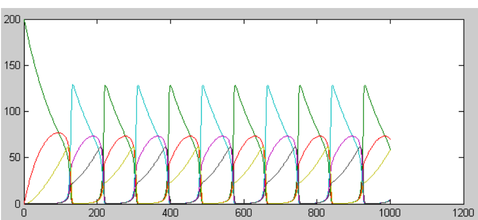


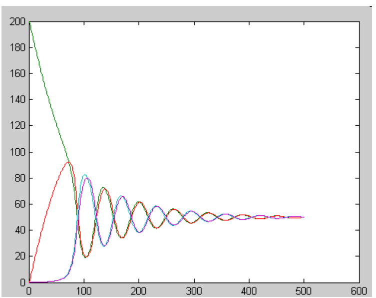


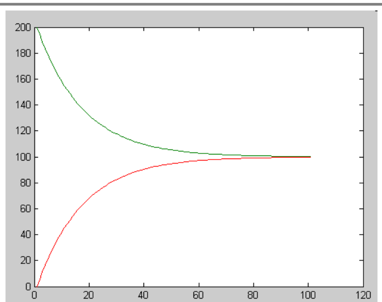




All with 1x Doping

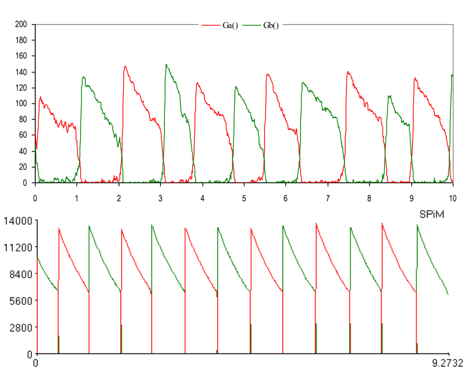


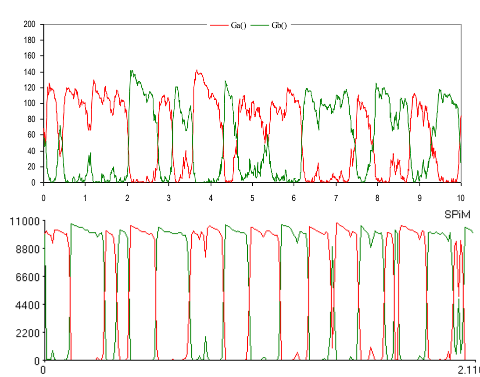


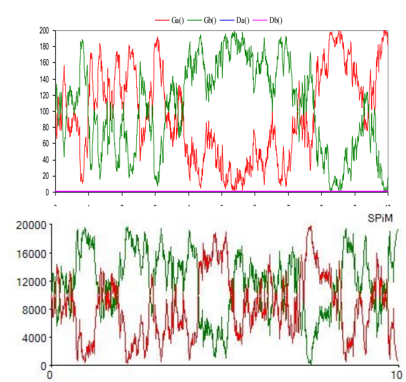


Matlab

continuous_sys_generator







SPiM

x200

x20000

Conclusions

- Compositional models
  - Accurate (at the "appropriate" abstraction level).
  - Manageable (so we can scale them up by composition).
  - Executable (stochastic simulation).

- Analysis techniques
  - Mathematical techniques: Markov theory, Chemical Master Equation, and Rate Equation
  - Computing techniques: Abstraction and Refinement, Model Checking, Causality Analysis.

- Many lines of extensions
  - Parametric processes for model factorization
  - Ultimately, rich process-algebra based modeling languages.

- Quantitative techniques
  - Important in the "real sciences".