Molecules as Automata

Representing Biochemical Systems as Collectives of Interacting Automata

Luca Cardelli

Microsoft Research

Open Lectures for PhD Students in Computer Science
Warsaw 2009-03-12..13

http://lucacardelli.name

Macro-Molecules as Interacting Automata

Cells Compute

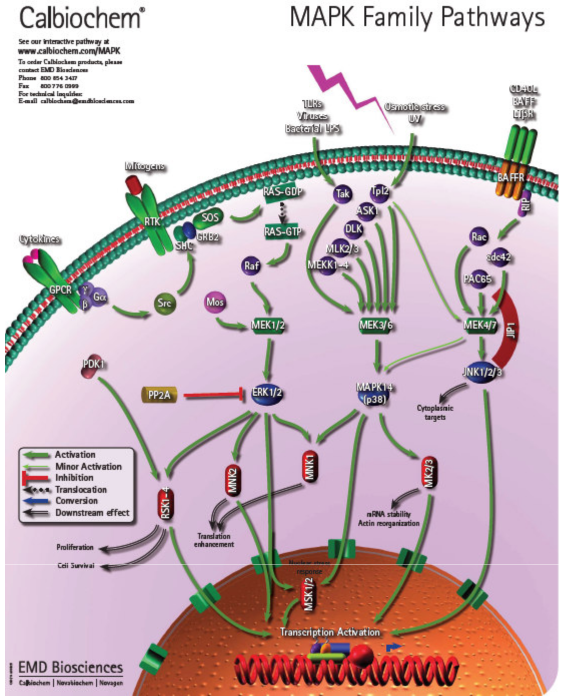
- No survival without computation!
  - Finding food
  - Avoiding predators

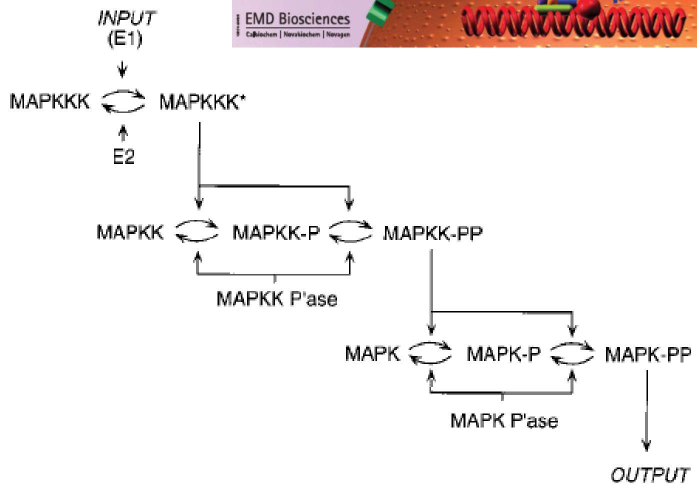
- *How* do they compute?
  - Unusual computational paradigms.
  - Proteins: do they work like electronic circuits?
  - Genes: what kind of software is that?

- 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.

- General models of biological computation
  - What are the appropriate ones?

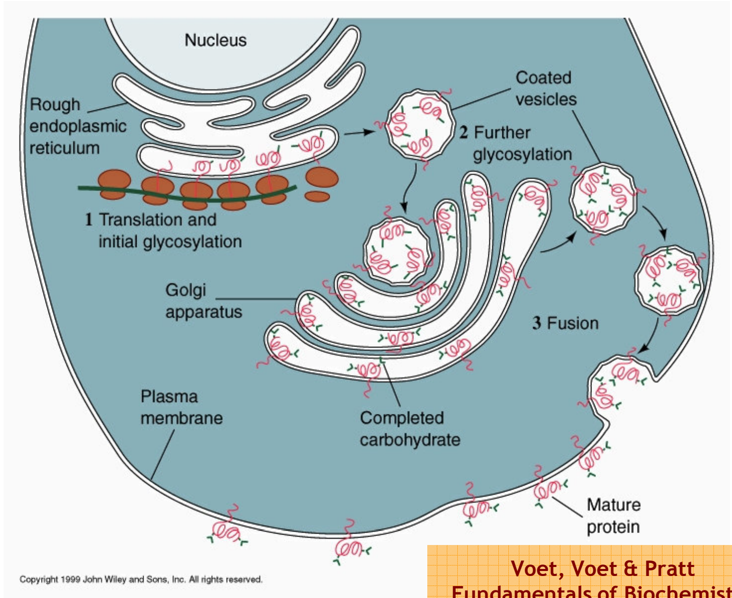




Ultrasensitivity in the mitogen-activated protein cascade, Chi-Ying F. Huang and James E. Ferrell, Jr., 1996, *Proc. Natl. Acad. Sci. USA*, 93, 10078-10083.

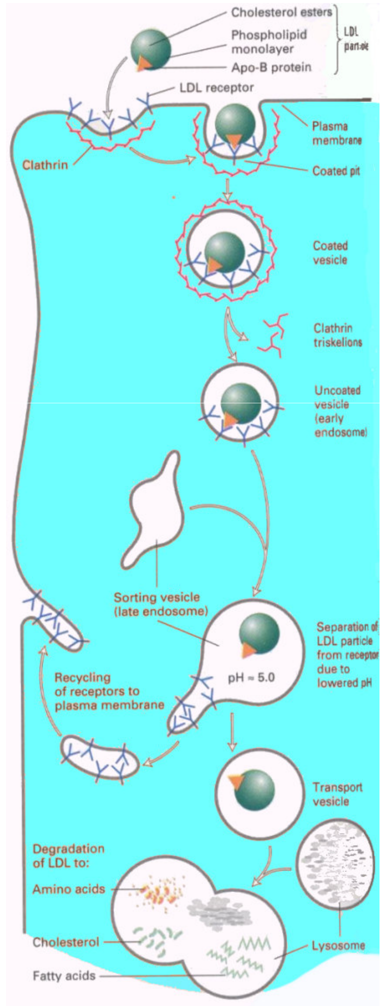
Biological “Algorithms”

Protein Production and Secretion

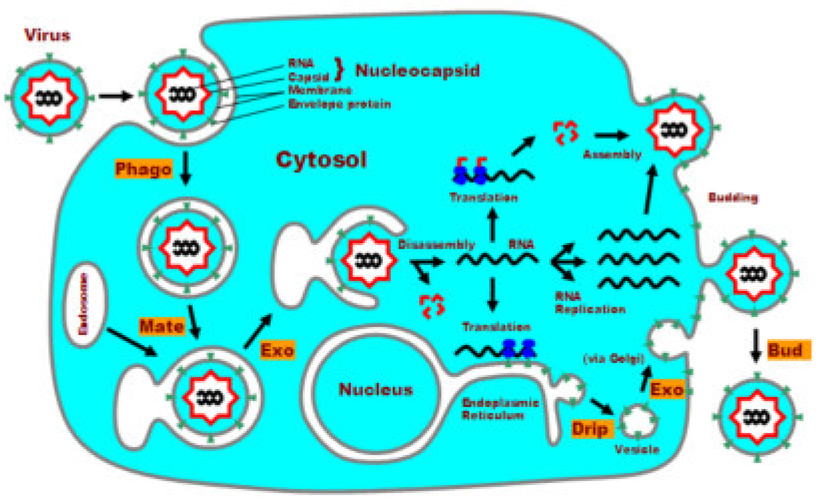


Voet, Voet & Pratt
Fundamentals of Biochemistry
Wiley 1999. Ch10 Fig 10-22.

LDL-Cholesterol Degradation



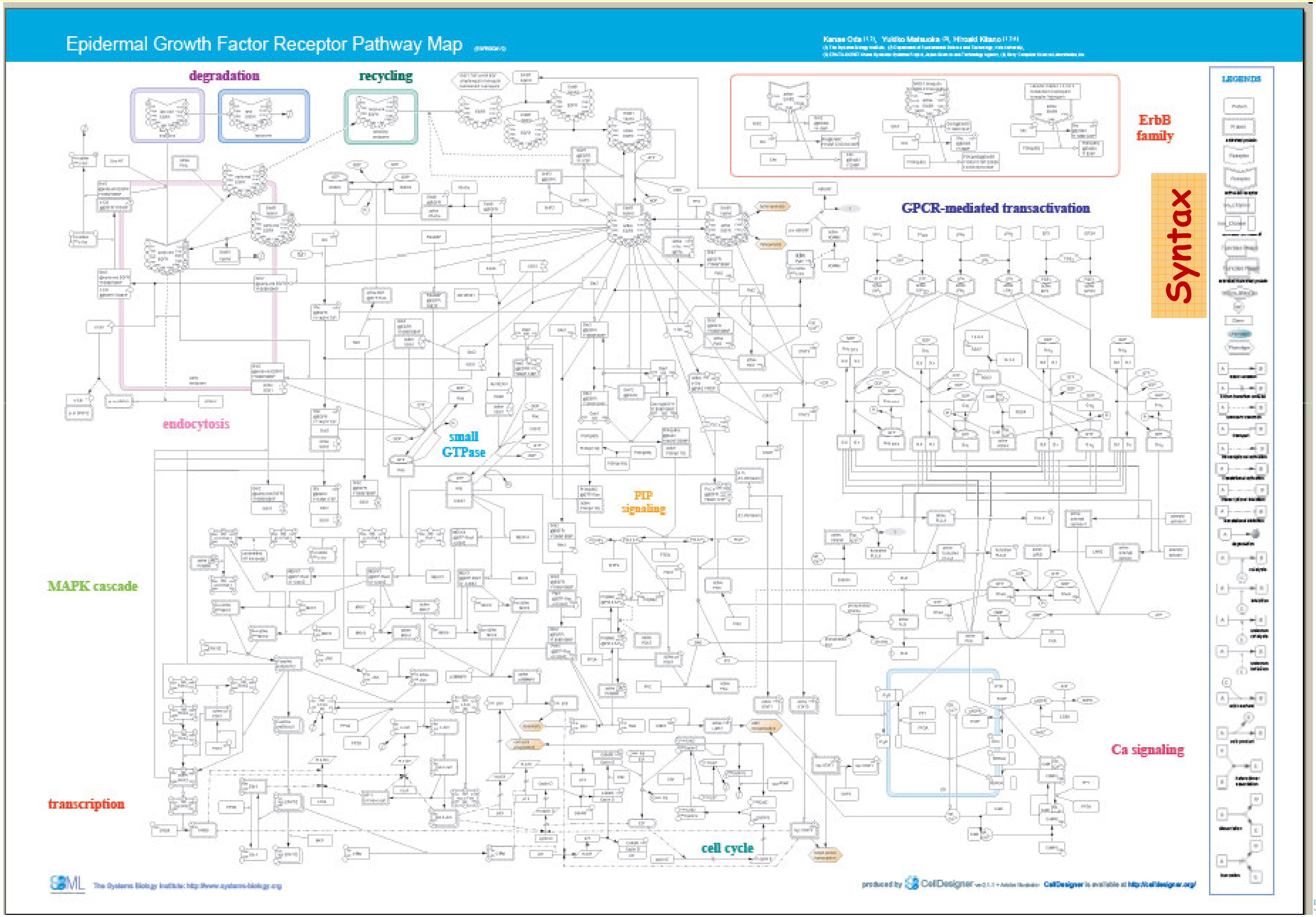
Viral Replication



Adapted from: B.Alberts et al.
Molecular Biology of the Cell
third edition p.279.

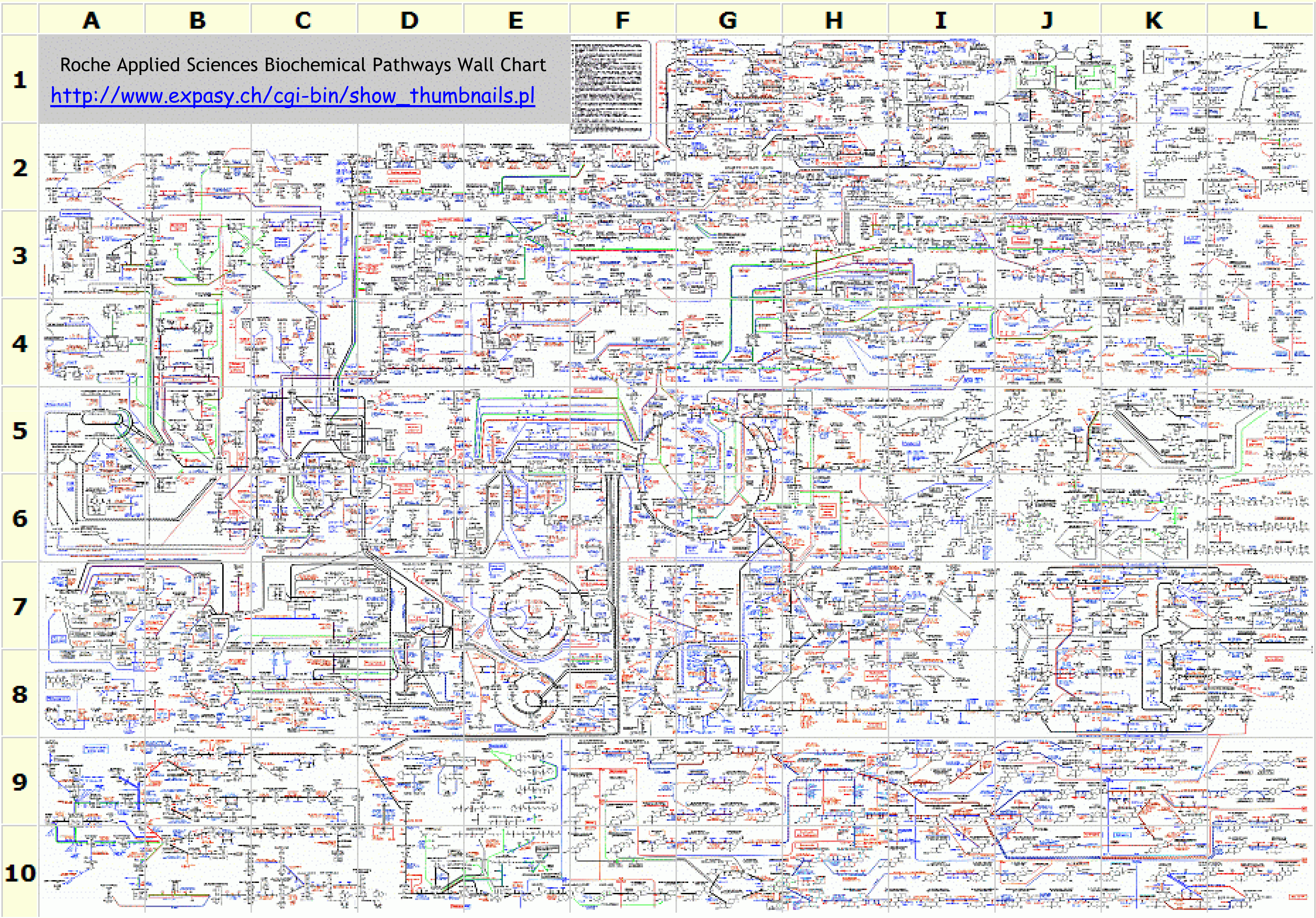
H.Lodish et al.
Molecular Cell Biology.
fourth Edition p.730.

Discrete State Transitions



Compositionality (NOT!)

Roche Applied Sciences Biochemical Pathways Wall Chart
http://www.expasy.ch/cgi-bin/show_thumbnails.pl



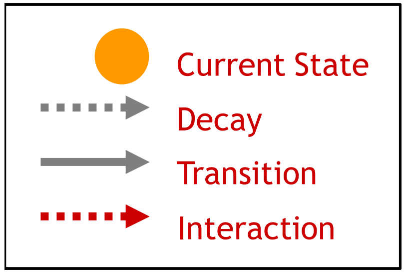
Process Algebra

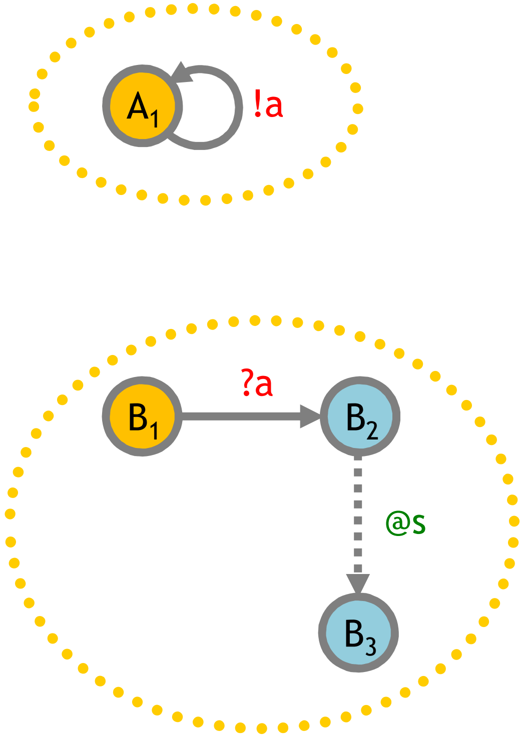
[Hoare, Milner, Pnueli, etc.]

- Reactive systems (living organisms, computer networks, operating systems, …)
  - Math is based on *entities that react/interact with their environment ("processes")*, not on functions from domains to codomains.
- Concurrent
  - Events (reactions/interactions) happen concurrently and asynchronously, not sequentially like in function composition.
- Stochastic
  - Or probabilistic, or nondeterministic, but is never about deterministic system evolution.
- Stateful
  - Each concurrent activity ("process") maintains its own local state, as opposed to stateless functions from inputs to outputs.
- Discrete
  - Evolution through discrete transitions between discrete states, not incremental changes of continuous quantities.
- Kinetics of interaction
  - An "interaction" is anything that moves a system from one state to another.

Interacting Automata

Legend

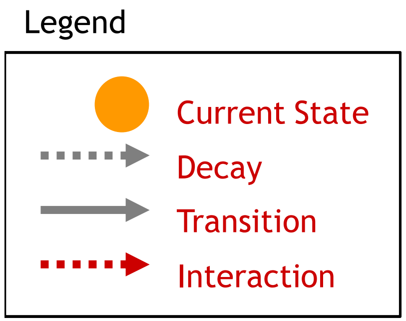


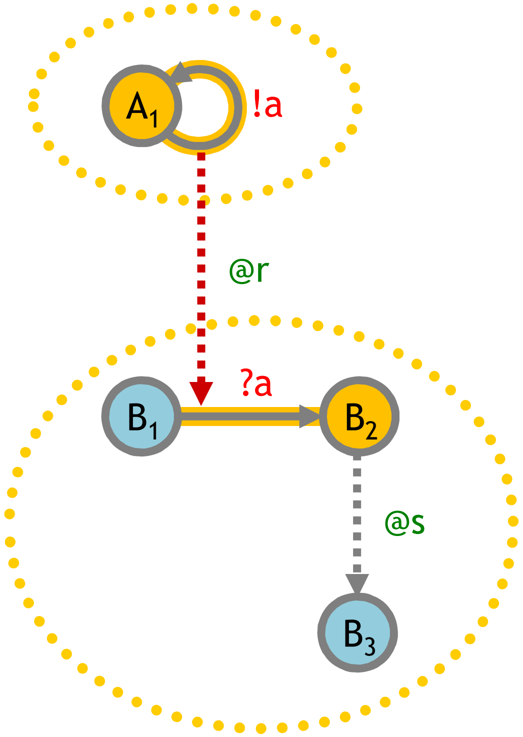


A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge)
?a, !a	indicate *complementary actions*,
@r, @s	are rates

Kinetic laws:

Interacting Automata



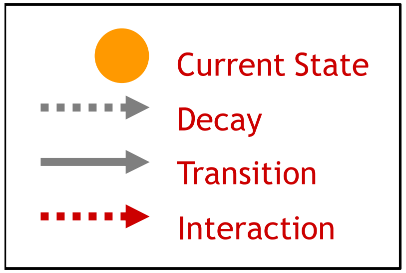


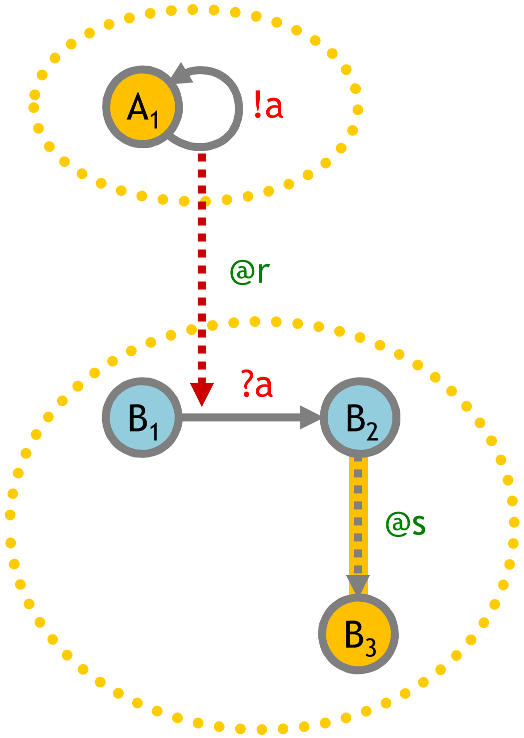
A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge, shape)
?a, !a	indicate *complementary actions*, joined by an interaction arrow
@r, @s	are rates

Kinetic laws: ***Two complementary actions may result in an interaction.***

Interacting Automata

Legend





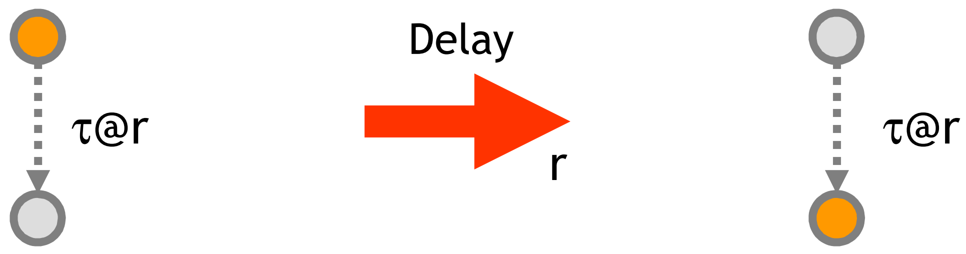
A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge)
?a, !a	indicate *complementary actions*, joined by an interaction arrow
@r, @s	are rates

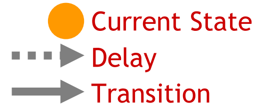
Kinetic laws:

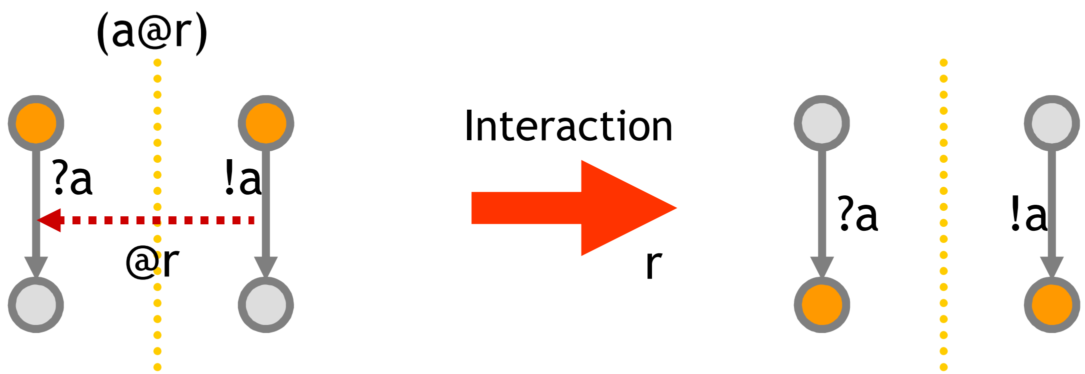
Two complementary actions may result in an interaction.

A decay may happen spontaneously.

Interacting Automata Transition Rules





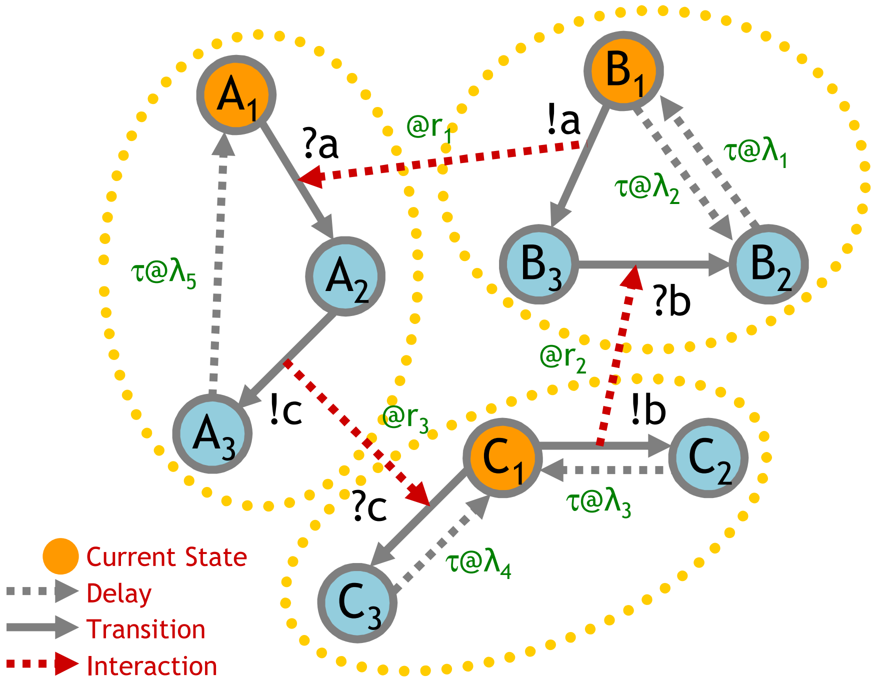


Interactions have rates. Actions DO NOT have rates.

Q: What kind of mass behavior can this produce?

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

Interacting Automata



Interactions have rates. Actions DO NOT have rates.

The equivalent process algebra model

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 = τ@λ_5; A_1

B_1 = τ@λ_2; B_2 + !a; B_3
B_2 = τ@λ_1; B_1
B_3 = ?b; B_2

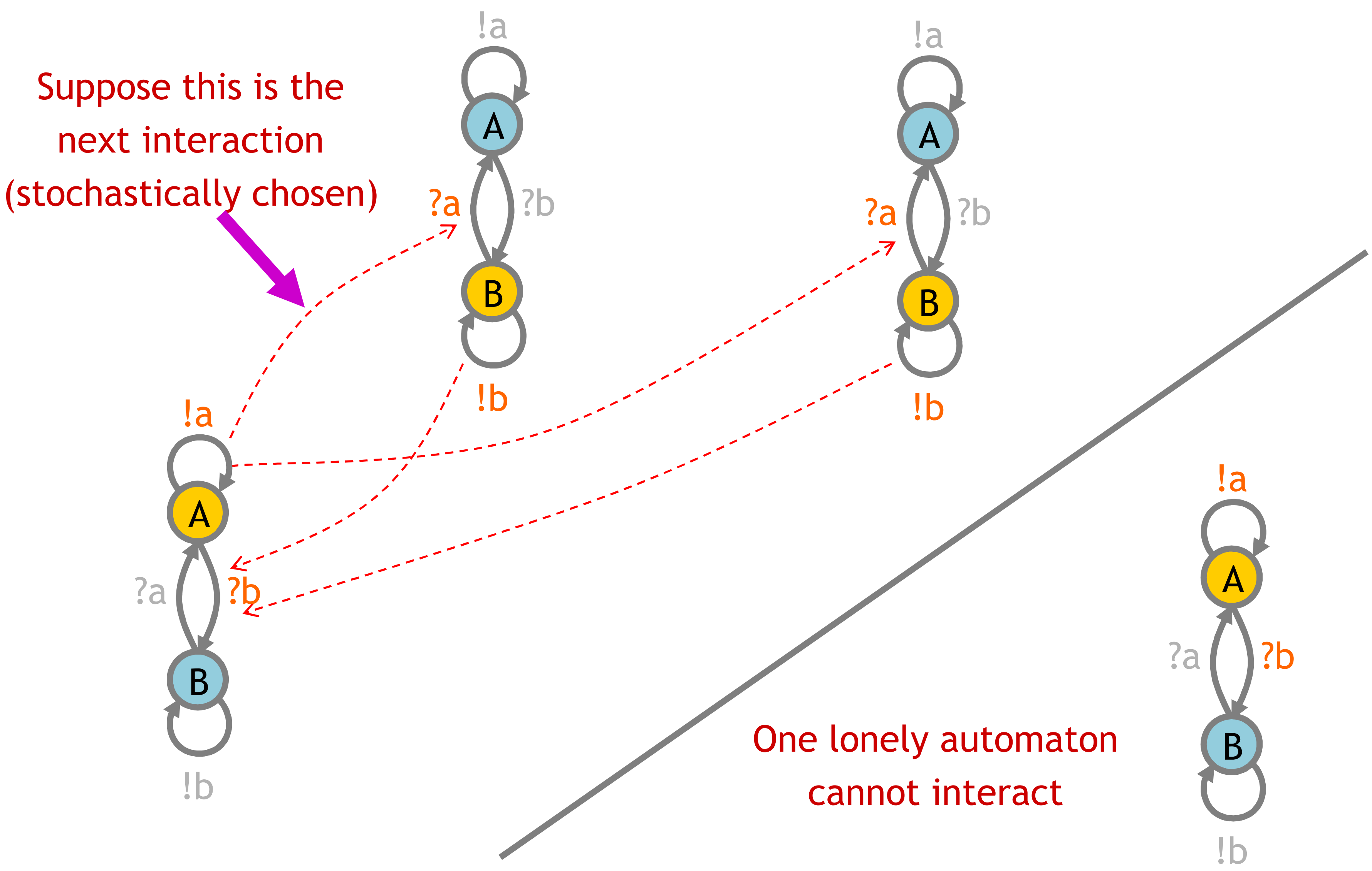
C_1 = !b; C_2 + ?c; C_3
C_2 = τ@λ_3; C_1
C_3 = τ@λ_4; C_2

Automata

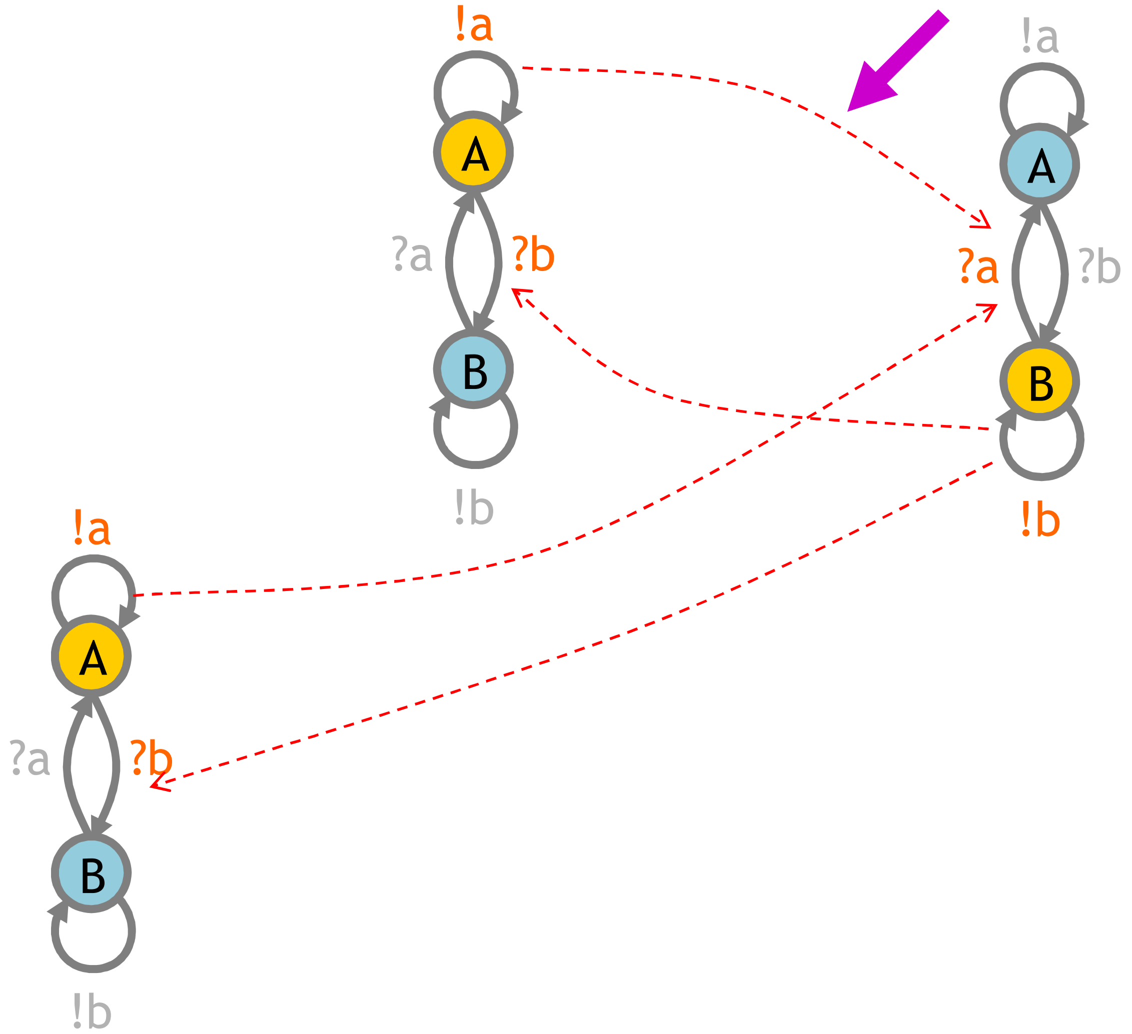
A_1 | B_1 | C_1

The system and initial state

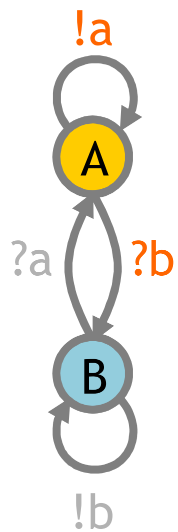
Interactions in a Population

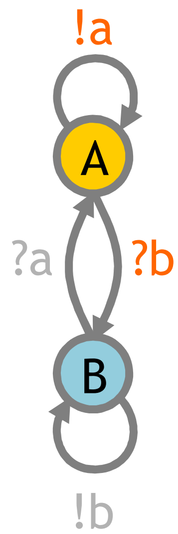


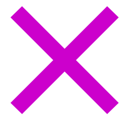
Interactions in a Population

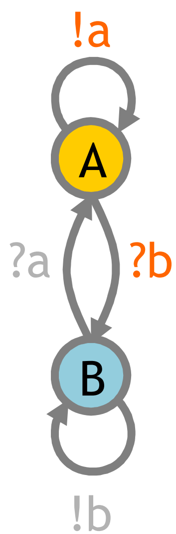


Interactions in a Population



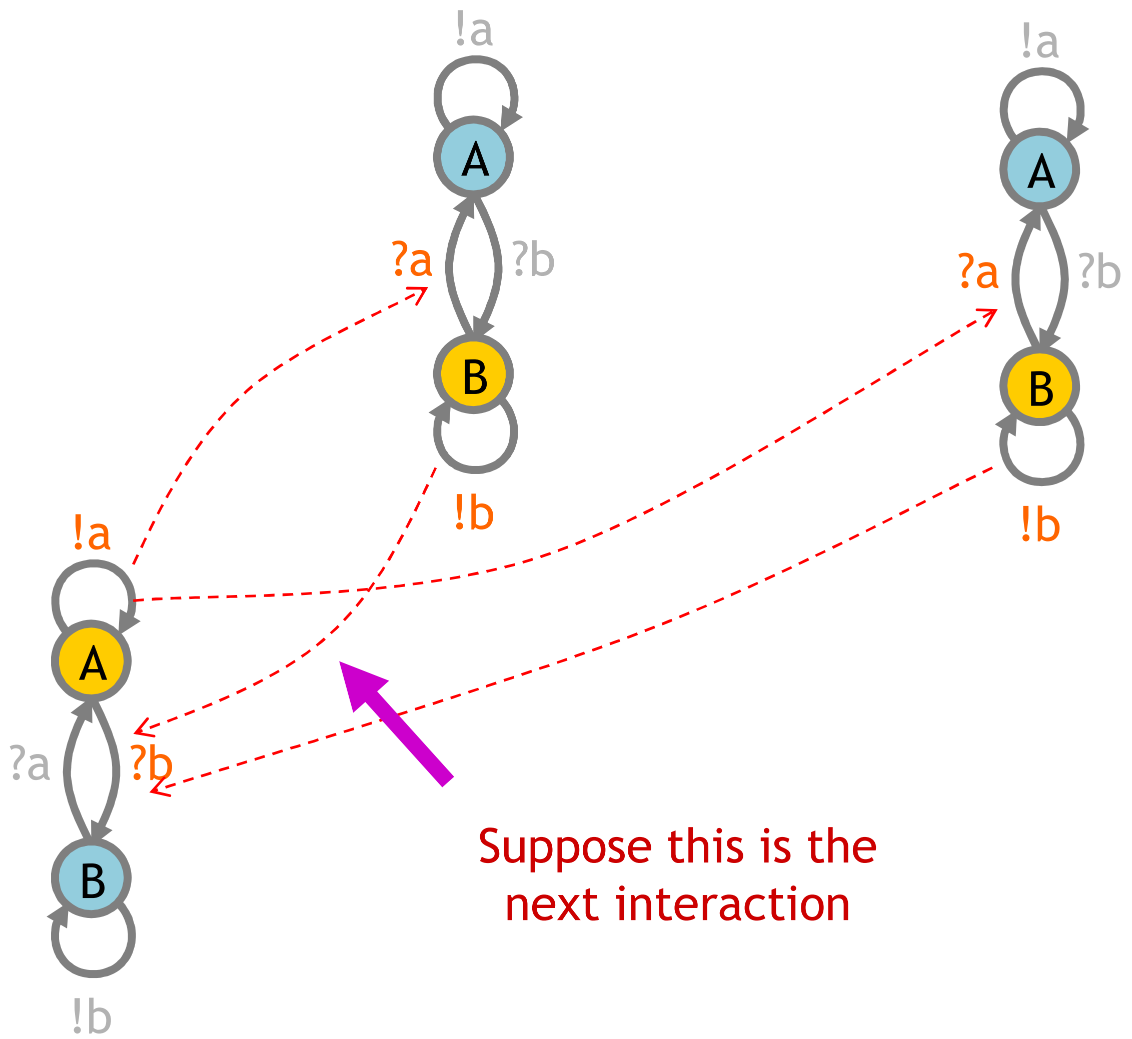






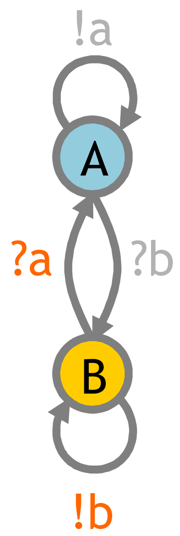
All-A stable
population

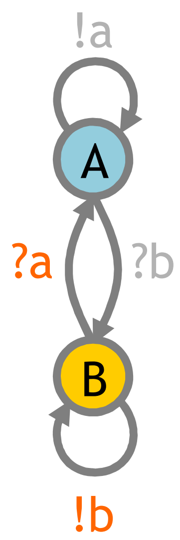
Interactions in a Population (2)

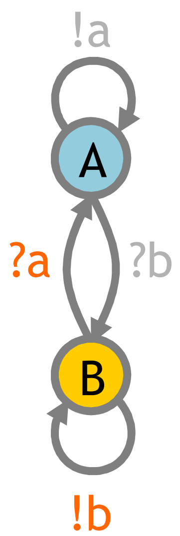


Suppose this is the next interaction

Interactions in a Population (2)



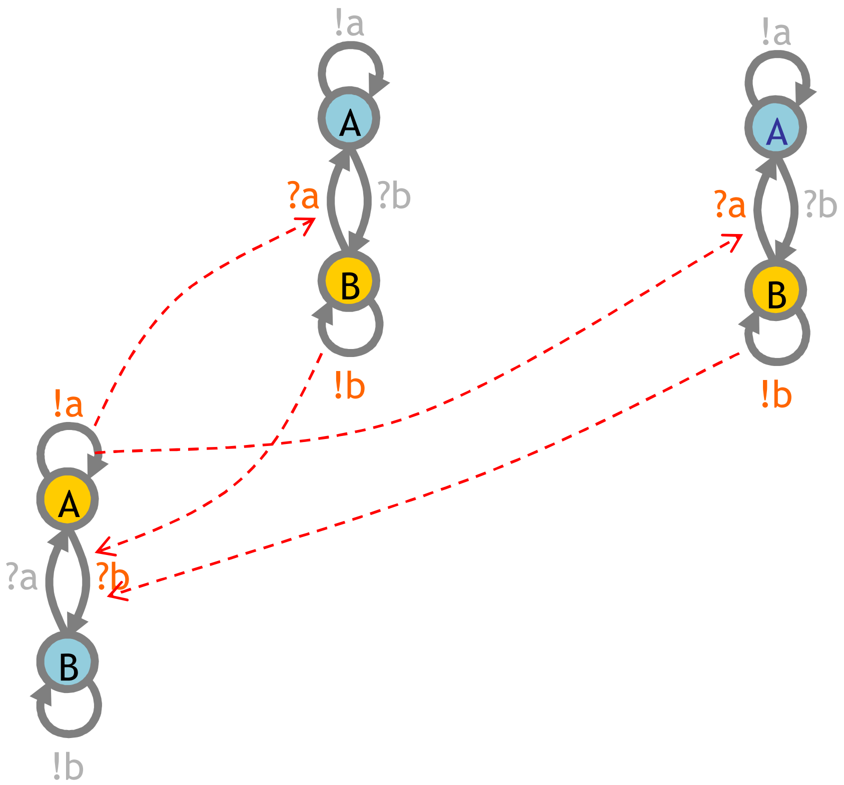


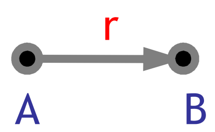


All-B stable population

Nondeterministic population behavior ("multistability")

CTMC Semantics



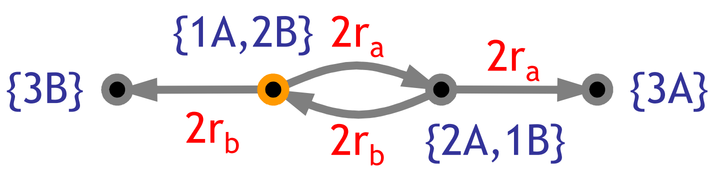


CTMC
(homogeneous) Continuous Time Markov Chain
- directed graph with no self loops
- nodes are system states
- arcs have transition rates

Probability of holding in state A:

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

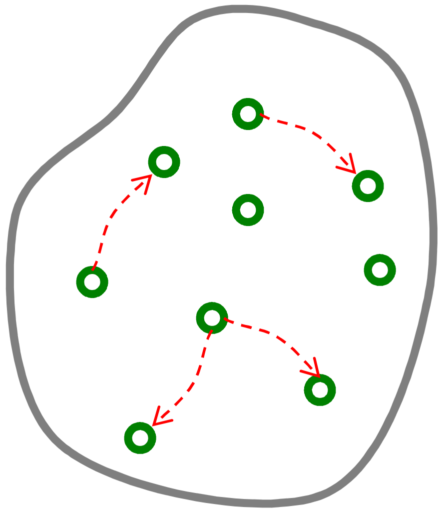
in general, $Pr(H_A>t) = e^{-Rt}$ where R is the sum of all the exit rates from A



CTMC

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].

Chemistry vs. Automata

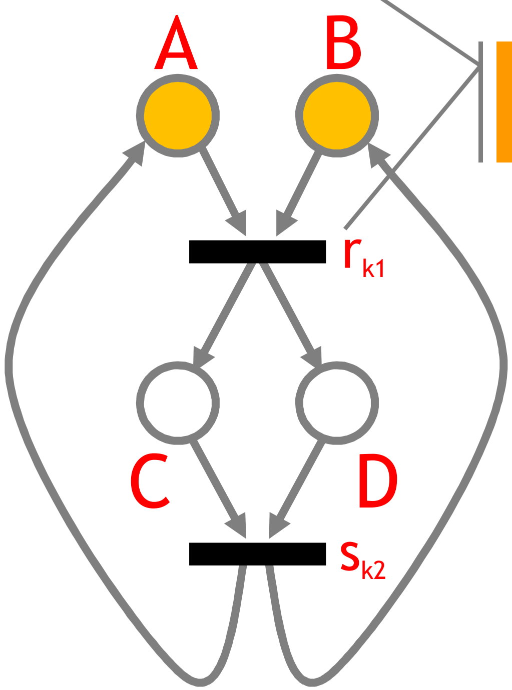
Says what "A" *does*.

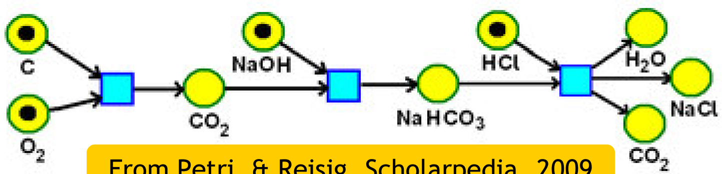
$$r: A + B \rightarrow_{k1} C + D$$
$$s: C + D \rightarrow_{k2} A + B$$

Does A become C or D?

Reaction oriented

1 line per reaction

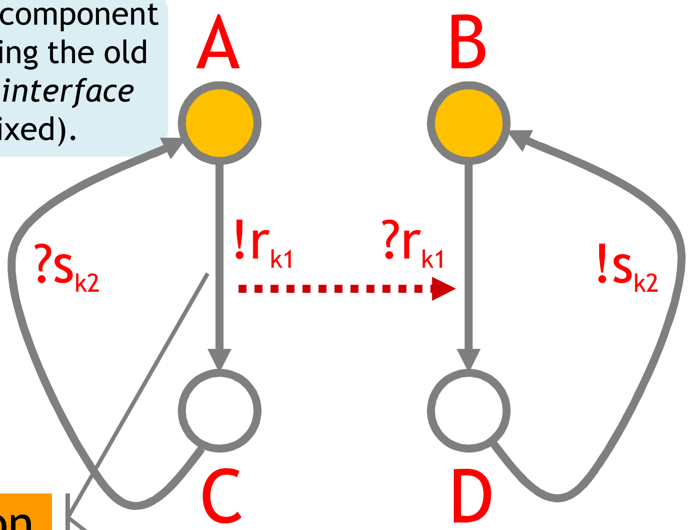




From Petri & Reisig, Scholarpedia, 2009

Says what "A" *is*.

Can add a new component without changing the old ones (if their *interface* remains fixed).



Interaction oriented

1 line per component

$$A = !r_{k1};\ C$$
$$C = ?s_{k2};\ A$$
$$B = ?r_{k1};\ D$$
$$D = !s_{k2};\ B$$

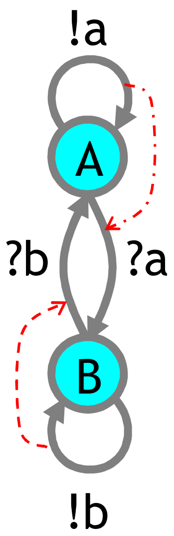
A becomes C not D!

The same "state space"

CTMC

Groupies and Celebrities

Groupies and Celebrities



Celebrity

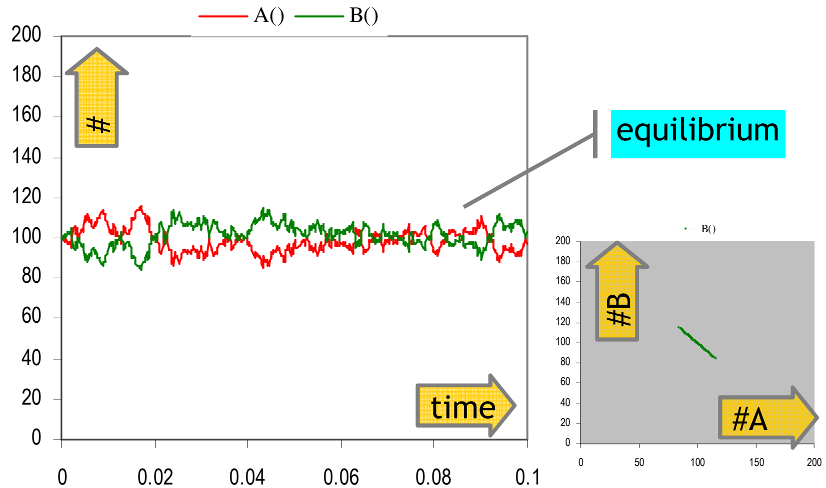
(does not want to be like somebody else)

```
directive sample 1.0 1000
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())
```

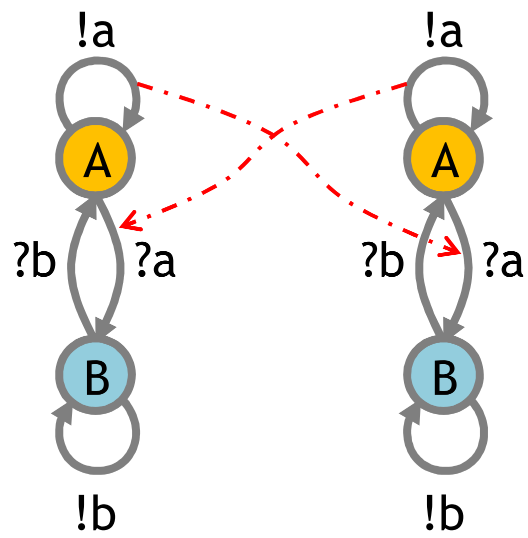
a@1.0

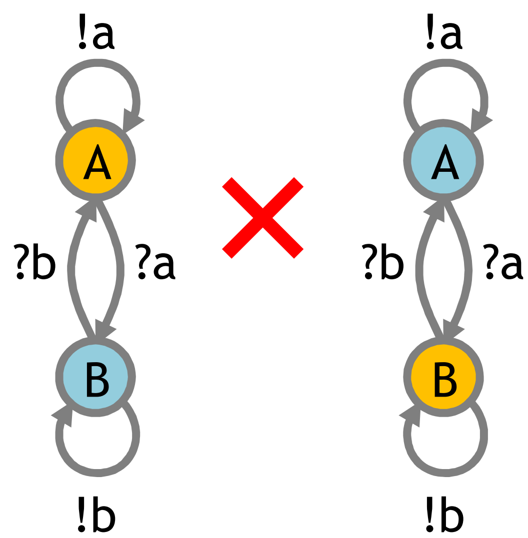
b@1.0

A stochastic collective of celebrities:

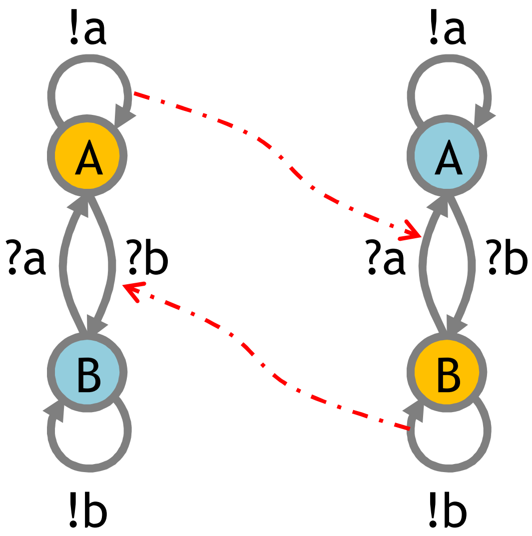


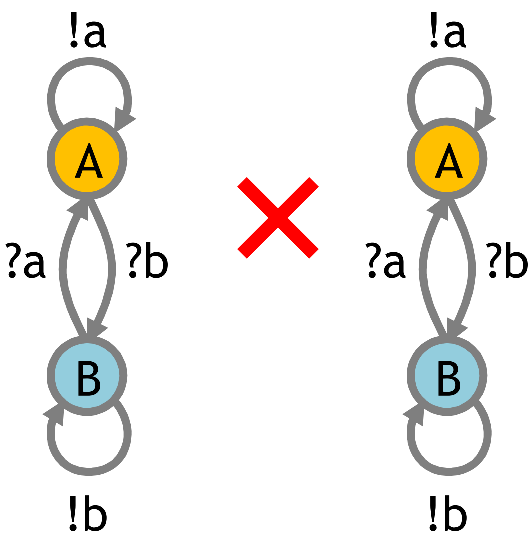
Stable because as soon as a A finds itself in the majority, it is more likely to find somebody in the same state, and hence change, so the majority is weakened.

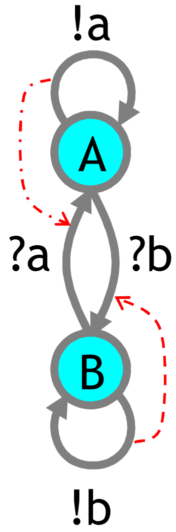




Groupies and Celebrities







Groupie

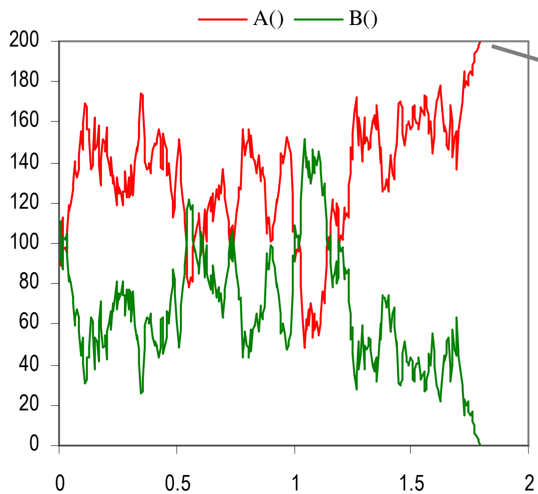
(wants to be like somebody different)

```
directive sample 1.0 1000
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())
```

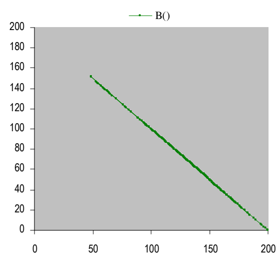
a@1.0

b@1.0

A stochastic collective of groupies:



always eventually deadlock

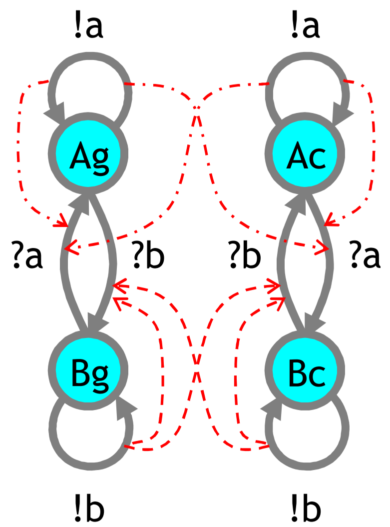


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

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

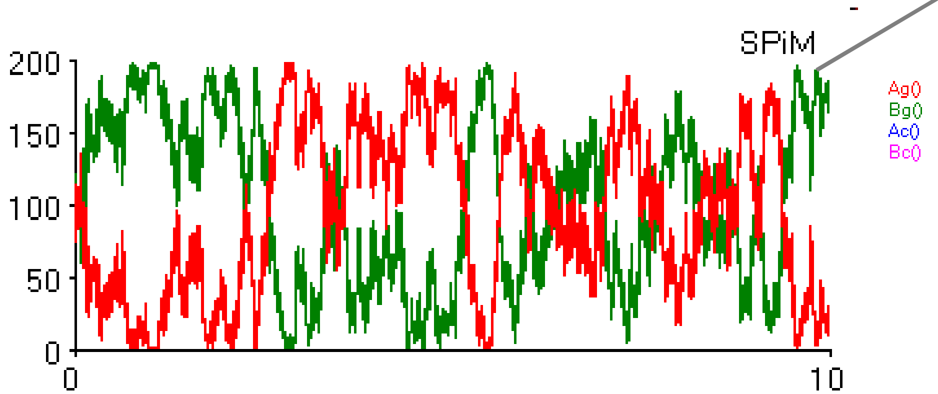
Many Groupies



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())
```

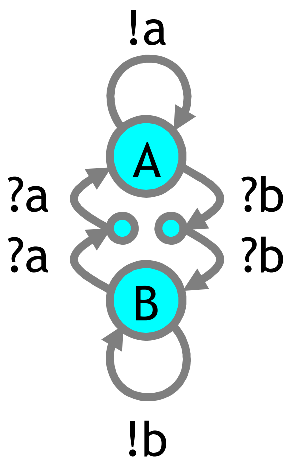
never deadlock

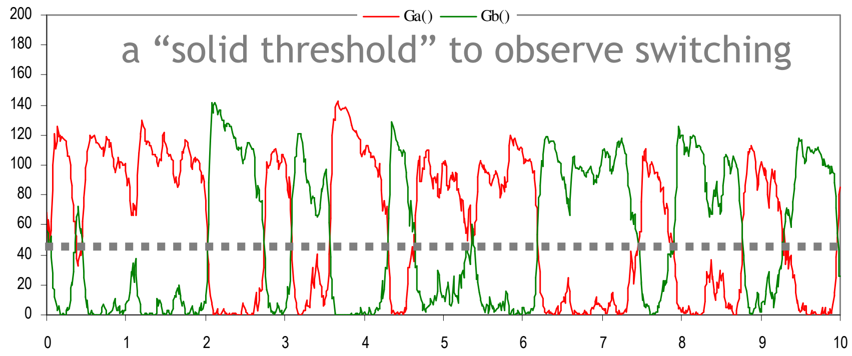


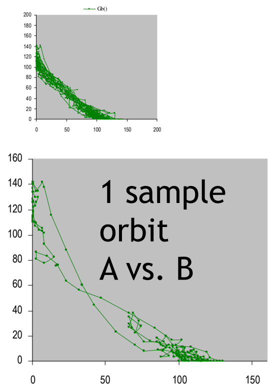
A tiny bit of "noise" can make a huge difference

Hysteric Groupies

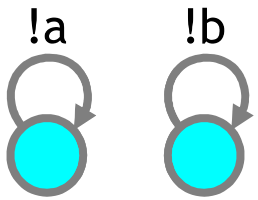
We can get more regular behavior from groupies if they “need more convincing”, or “hysteresis” (history-dependence), to switch states.







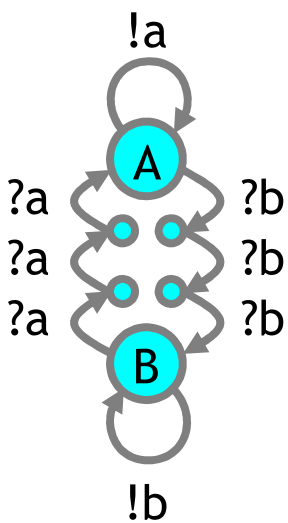

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

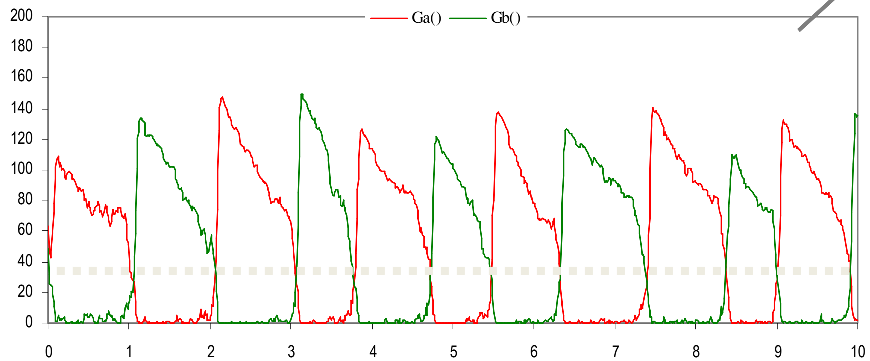


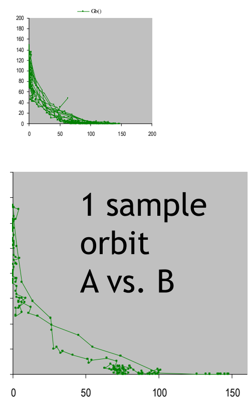
(With doping to break deadlocks)

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

“regular” oscillation





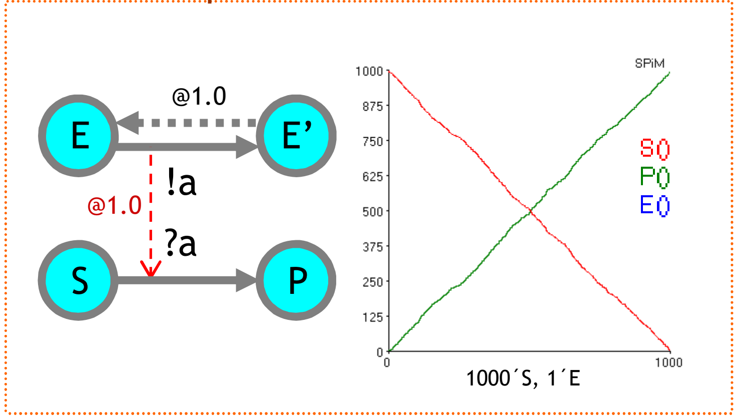



```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

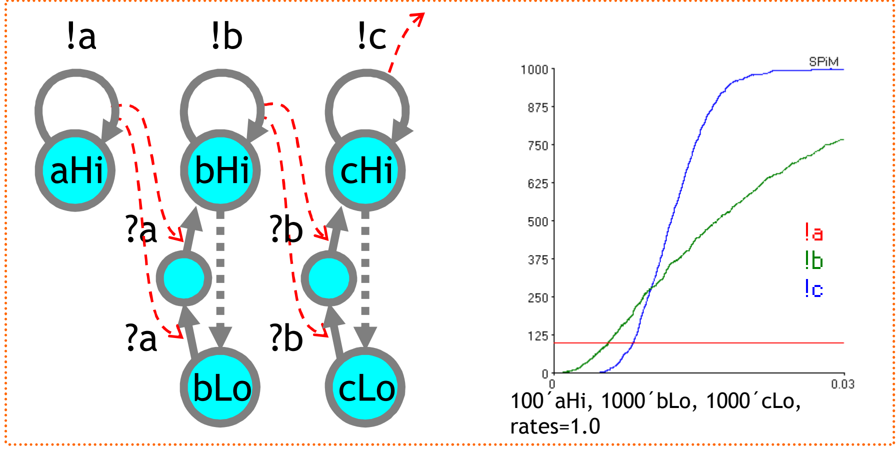
Devices

Some Devices

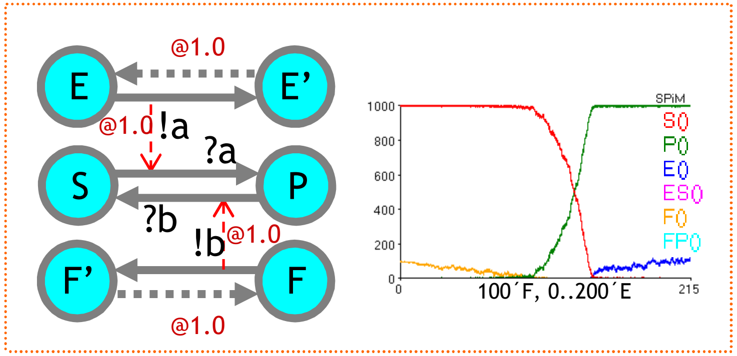
Linear Pump



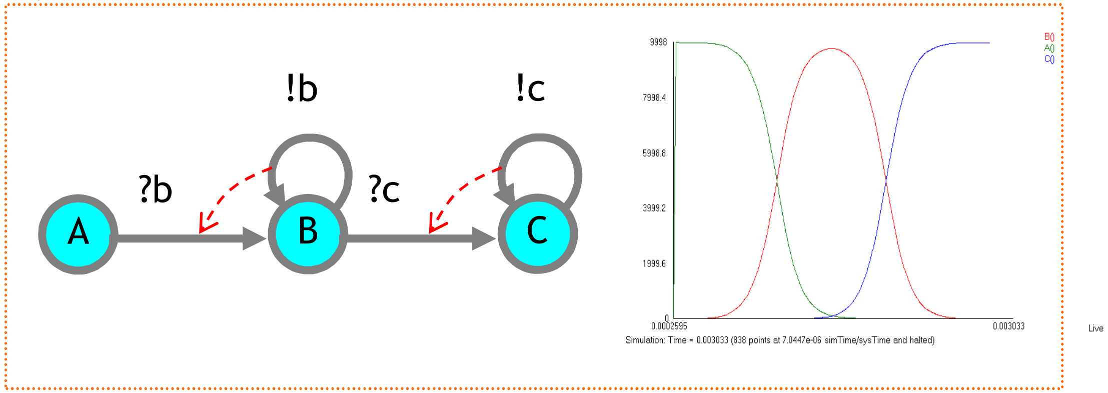
Cascade Amplifier



Ultrasensitive Switch

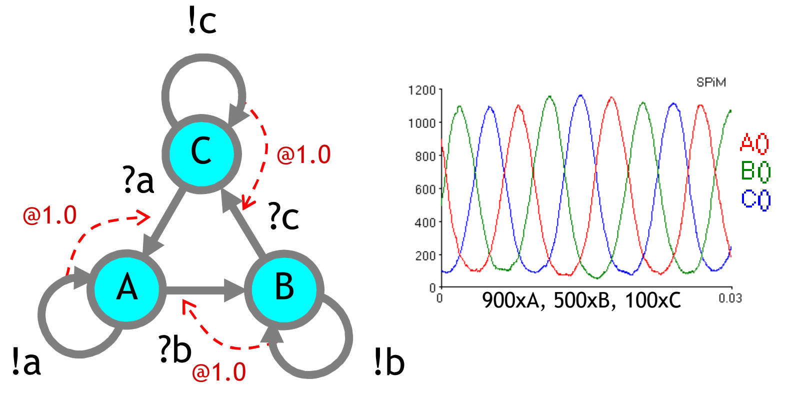


Symmetric Wave Generator

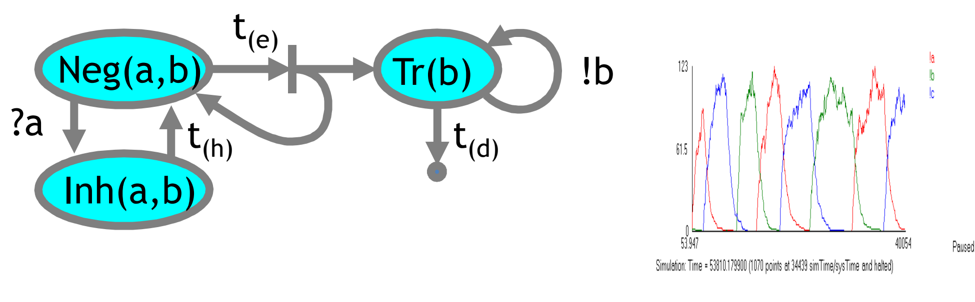


More Devices

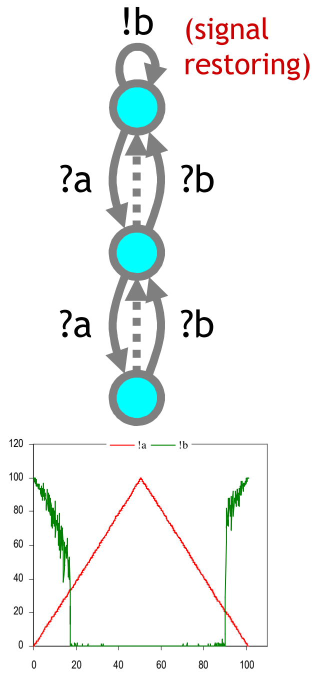
Oscillator



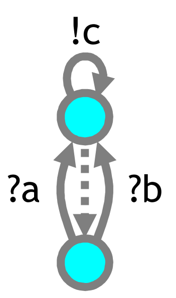
Repressilator (1 of 3 similar gates)



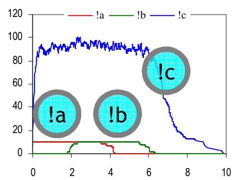
b = not a



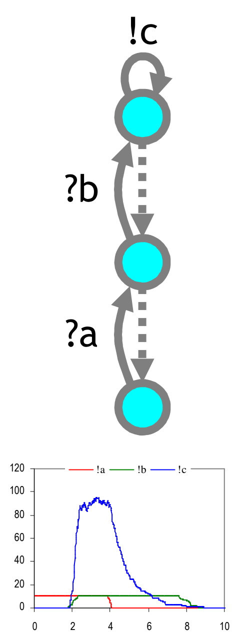
c = a or b



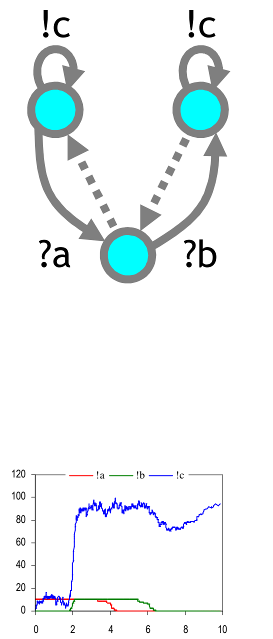
Inputs:
10 !a for 4t
2t; 10 !b for 4t



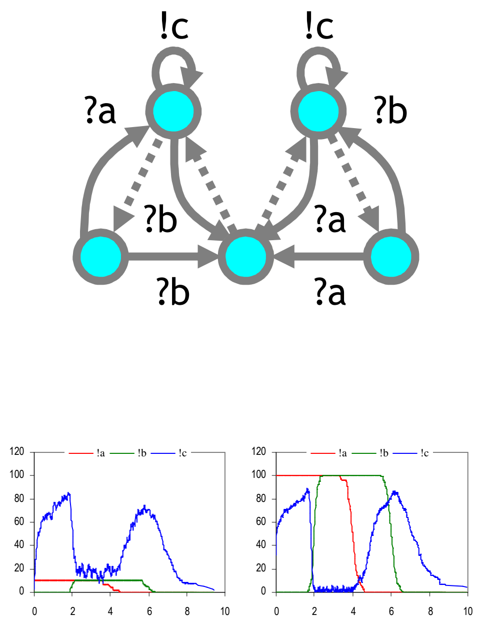
c = a and b



c = a imply b

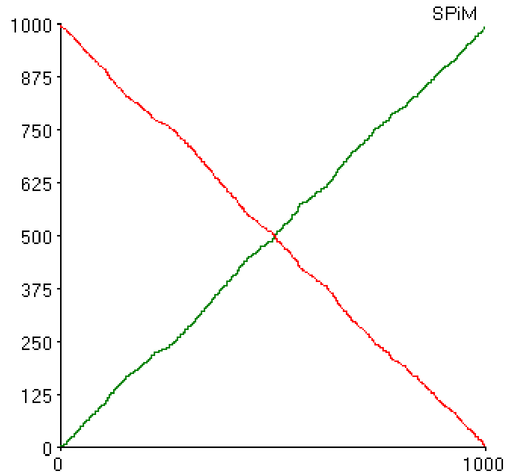


c = a xor b

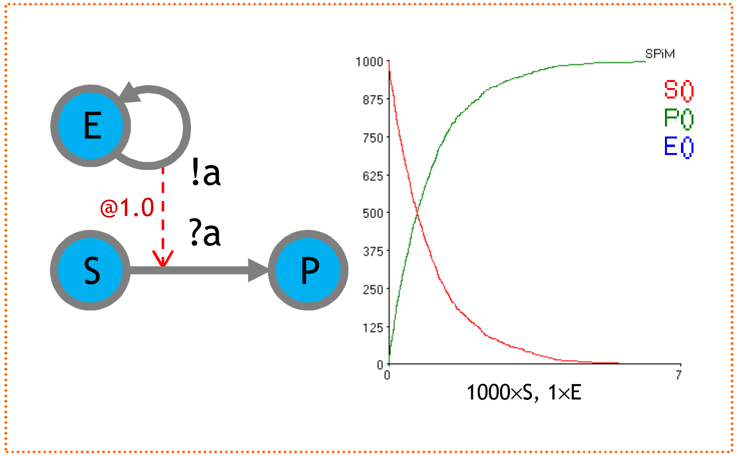


Design Exercise: Making Lines

Build me a population like this:



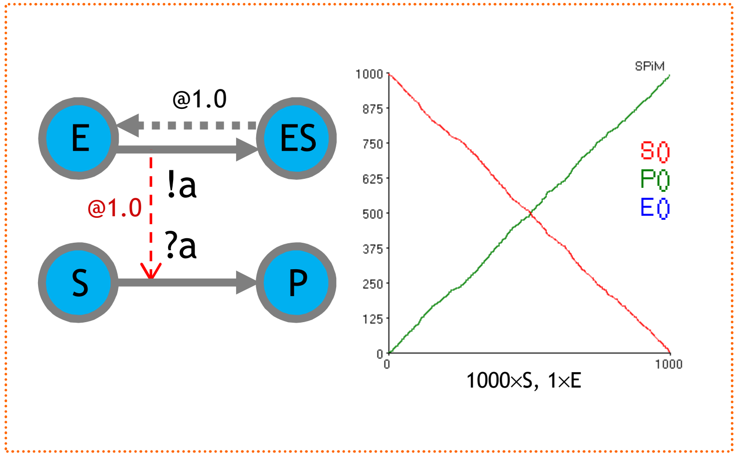
Second-order and Zero-order Regime



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

```
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())
```

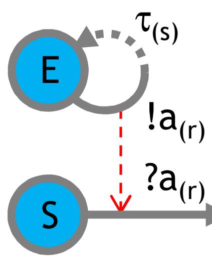
Second-Order Regime
$d[S]/dt = -r[E][S]$



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

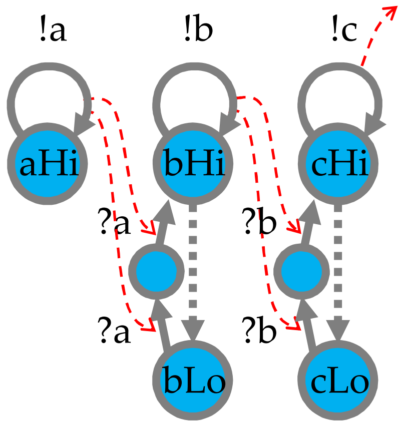
```
directive sample 1000.0
directive plot S(); P(); E()

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())
```

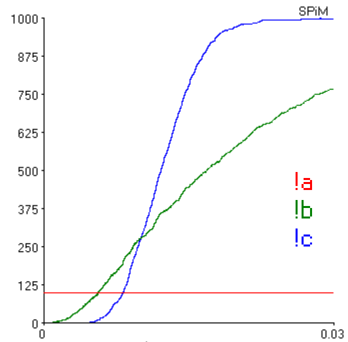
Zero-Order Regime
$d[S]/dt \cong -1$ (by assuming $d[ES]/dt = 0$)



Notation

Cascades



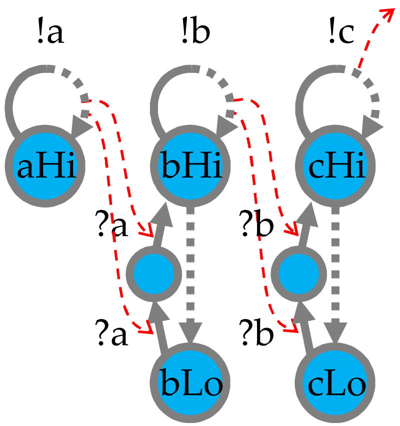


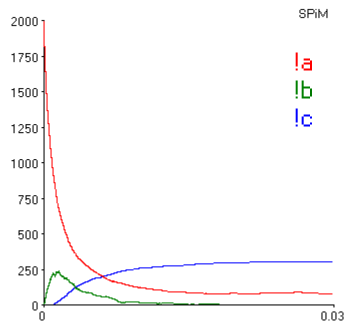
100×aHi, 1000×bLo, 1000×cLo, rates=1.0

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()
```





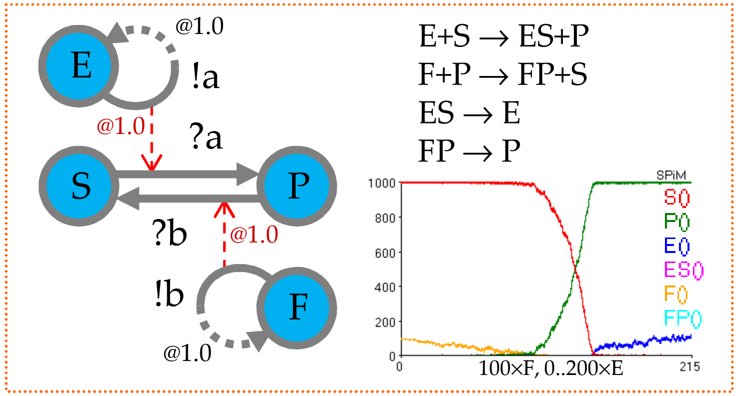
2000×aHi, 1000×bLo, 1000×cLo, rates=1.0

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

aHi = max ⇒ cHi = 1/3 max

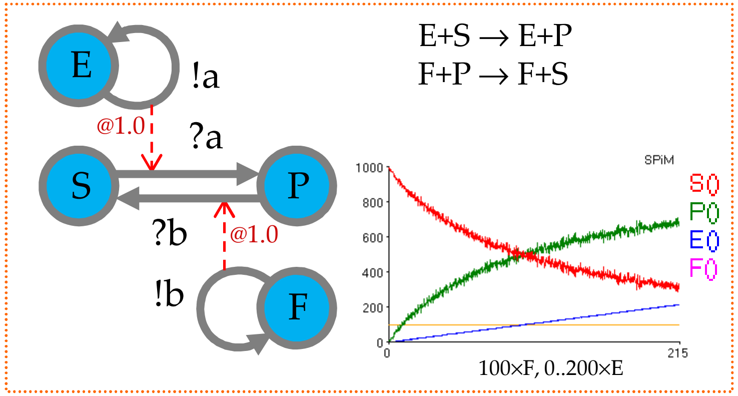
```
directive sample 0.03
directive plot !a; !b; !c

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




```
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.

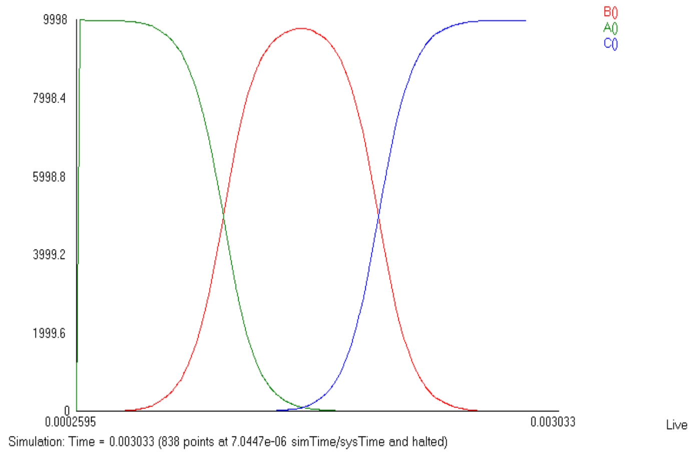



```
directive sample 215.0 1000
directive plot S(); P(); E(); F()

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)
```

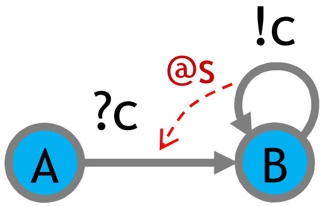
Second-Order Regime

Design Exercise: Making Waves

Build me a population like this:



Nonlinear Transition (NLT)

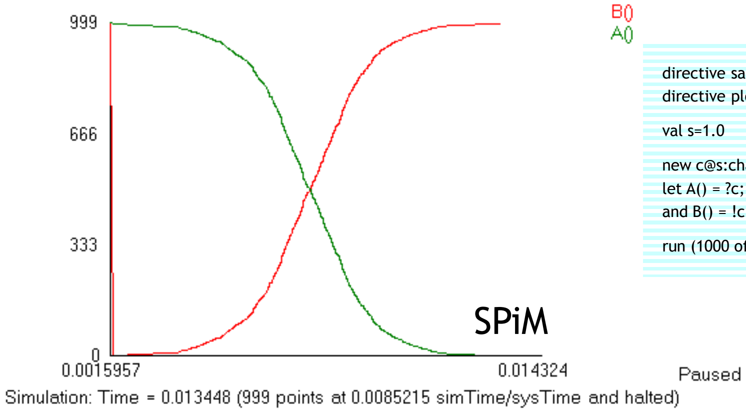


$A = ?c_{(s)};B$

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

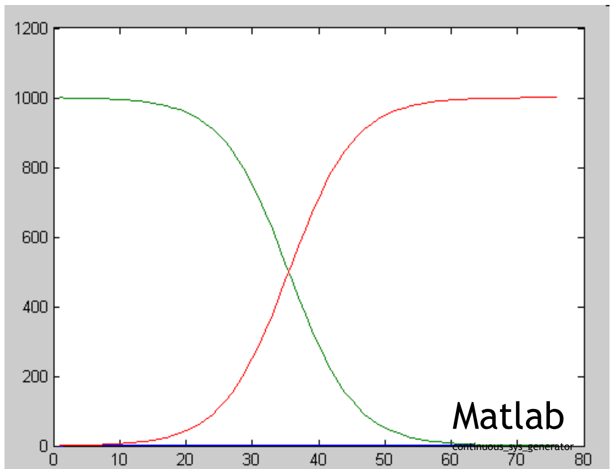
$A+B \longrightarrow^{s} B+B$

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



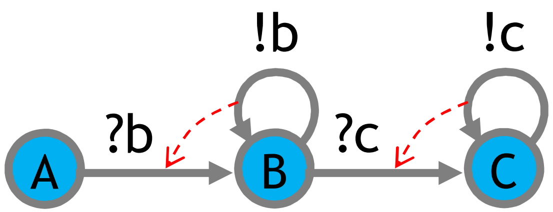

```
directive sample 0.02 1000
directive plot B(); A()

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



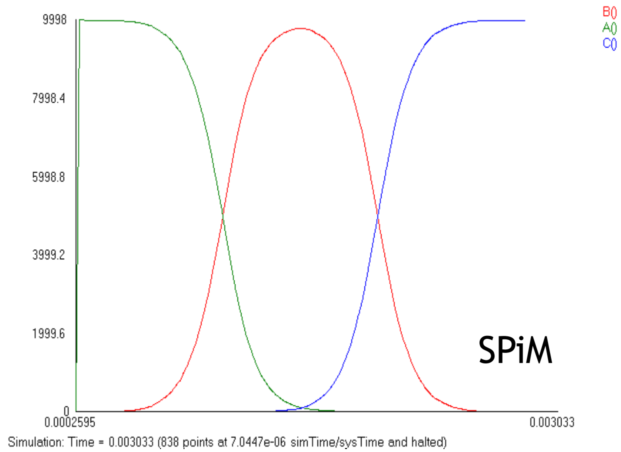
$$d[B]/dt = [B]([A]-[C])$$

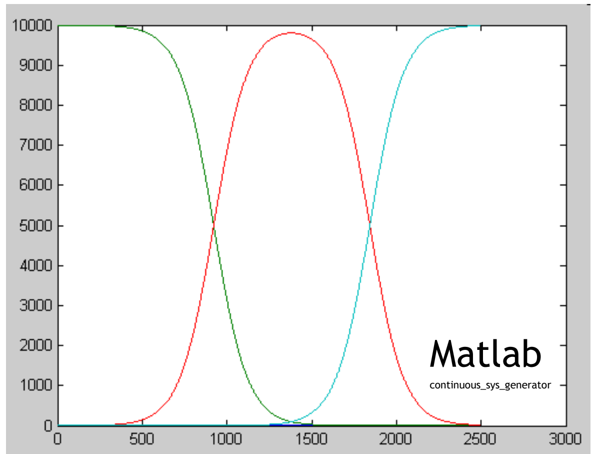
```
directive sample 0.0025 1000
directive plot B(); A(); C()

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())
```

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

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

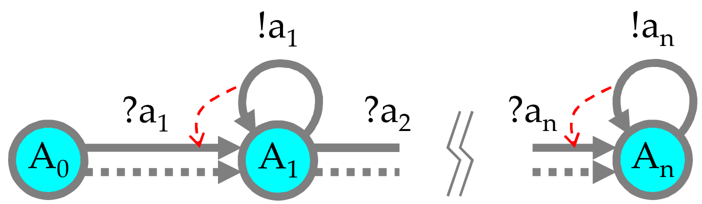
$$d[A]/dt = -[A][B]$$
$$d[B]/dt = [A][B]-[B][C]$$
$$d[C]/dt = [B][C]$$

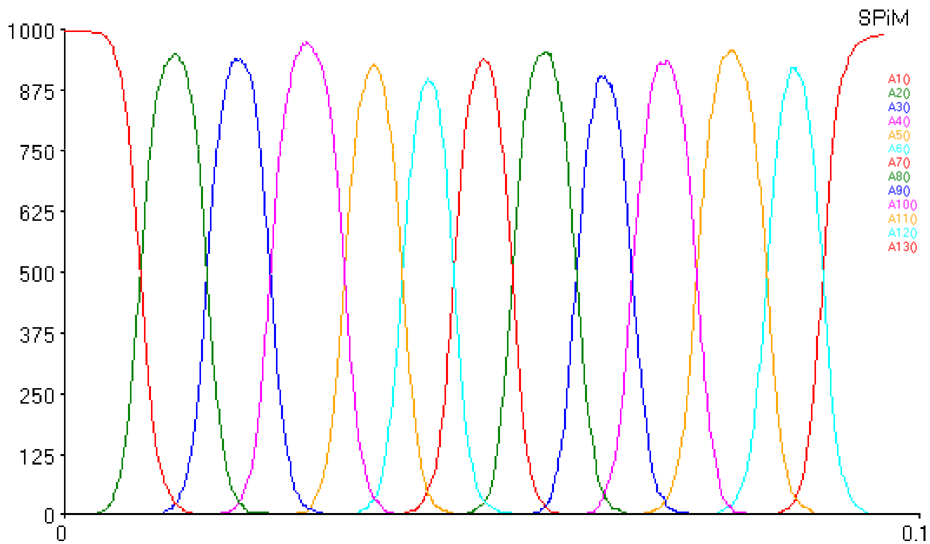





```
interval/step [0:0.000001:0.0025]
(A)     dx1/dt = -x1*x2           10000.0
(B)     dx2/dt = x1*x2 - x2*x3    1.0
(C)     dx3/dt = x2*x3            1.0
```

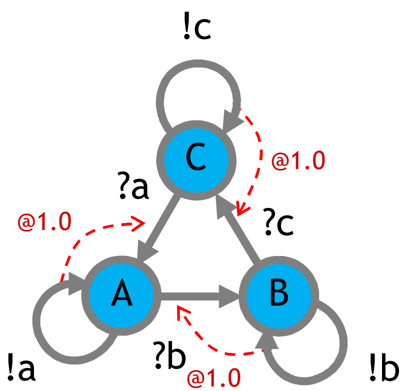
NLTs in Series: Soliton Propagation

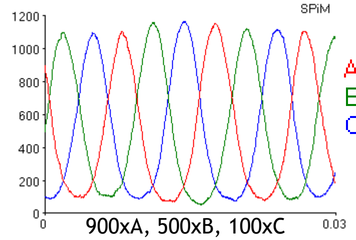





```
directive sample 0.1 1000
directive plot A1(); A2(); A3(); A4(); A5(); A6(); A7(); A8();
A9(); A10(); A11(); A12(); A13()

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 (unstable)






```
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())
```

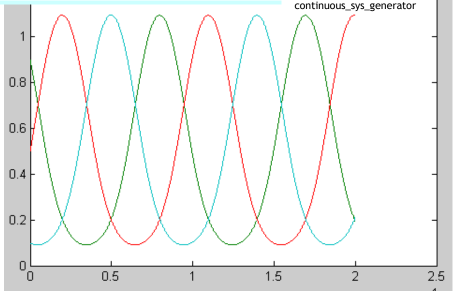
$$A = !a_{(s)};A \oplus ?b_{(s)};B$$
$$B = !b_{(s)};B \oplus ?c_{(s)};C$$
$$C = !c_{(s)};C \oplus ?a_{(s)};A$$

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

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

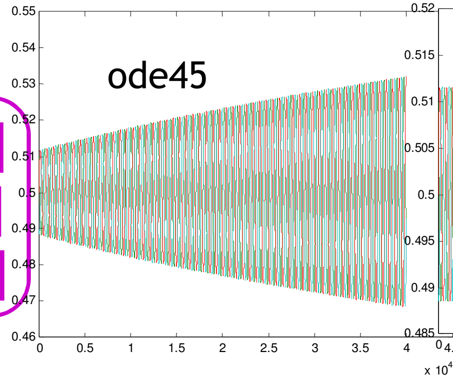
interval/step [0:0.001:20.0]

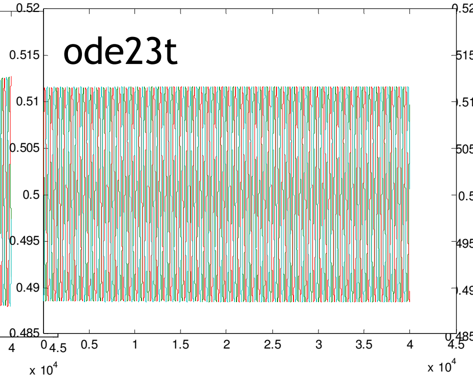
(A)	dx1/dt = - x1*x2 + x3*x1	0.9
(B)	dx2/dt = - x2*x3 + x1*x2	0.5
(C)	dx3/dt = - x3*x1 + x2*x3	0.1

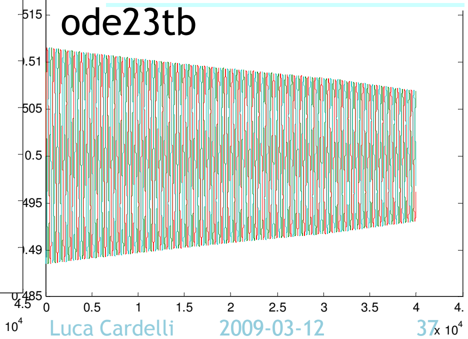


interval/step [0:0.01:400.0]

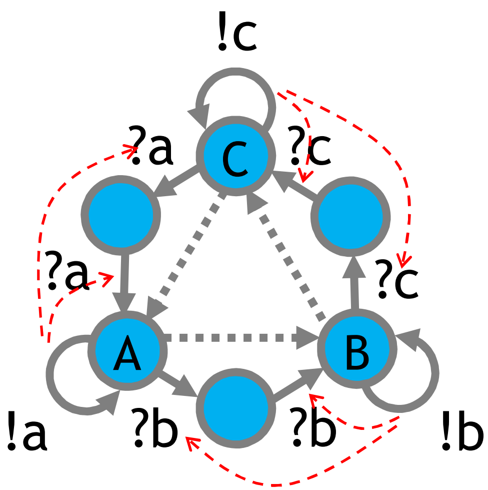
(A)	dx1/dt = - x1*x2 + x3*x1	0.51
(B)	dx2/dt = - x2*x3 + x1*x2	0.5
(C)	dx3/dt = - x3*x1 + x2*x3	0.49







Oscillator (stable)



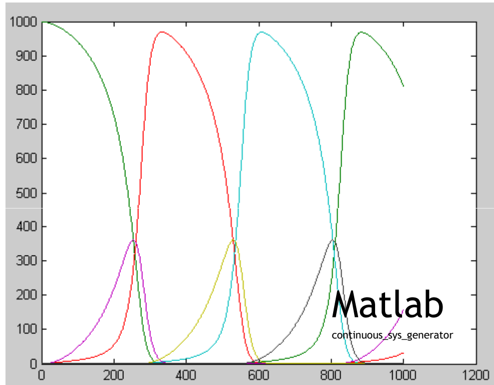

```
directive sample 0.1 1000
directive plot A1(); A2(); A3()

val r=1.0 val s=1.0

new a1@s:chan new a2@s:chan new a3@s:chan
let A1() = do !a1;A1() or delay@r;A2() or ?a2; ?a2; A2()
and A2() = do !a2;A2() or delay@r;A3() or ?a3; ?a3; A3()
and A3() = do !a3;A3() or delay@r;A1() or ?a1; ?a1; A1()

run 1000 of A1()
```

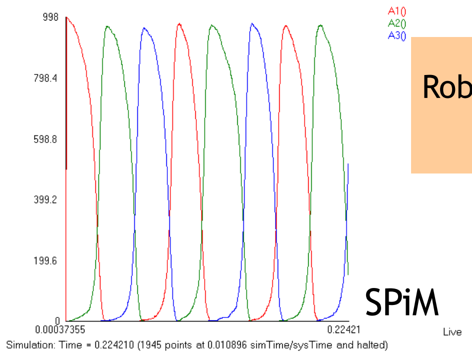
N.B. this does not deadlock!

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

Sustained Deterministic Oscillation

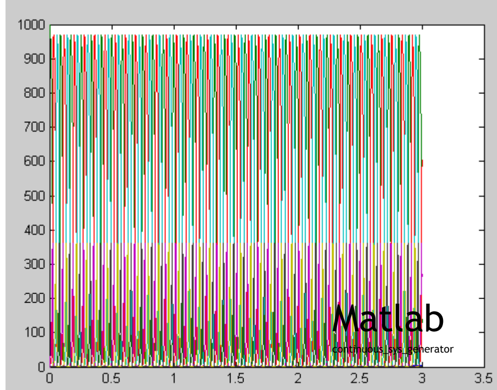
$A \xrightarrow{r} B$
$A+B \xrightarrow{s} A'+B$
$A'+B \xrightarrow{s} B+B$
$B \xrightarrow{r} C$
$B+C \xrightarrow{s} B'+C$
$B'+C \xrightarrow{s} C+C$
$C \xrightarrow{r} A$
$C+A \xrightarrow{s} C'+A$
$C'+A \xrightarrow{s} A+A$

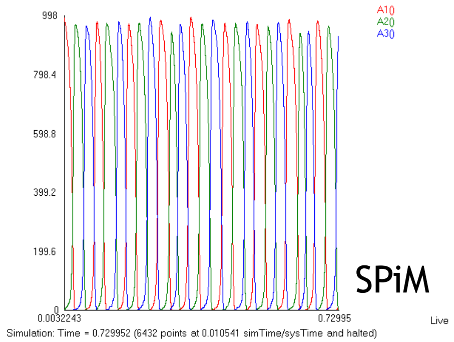




Robust Stochastic Oscillation

$d[A]/dt = -r[A]-s[A][B]+r[C]+s[C'][A]$
$d[B]/dt = -r[B]-s[B][C]+r[A]+s[A'][B]$
$d[C]/dt = -r[C]-s[C][A]+r[B]+s[B'][C]$
$d[A']/dt = -s[A'][B] + s[A][B]$
$d[B']/dt = -s[B'][C] + s[B][C]$
$d[C']/dt = -s[C'][A] + s[C][A]$






```
interval/step [0:0.0001:0.1]
(A)    dx1/dt = -x1 - x1*x2 + x3 + x6*x1    1000.0
(B)    dx2/dt = -x2 - x2*x3 + x1 + x4*x2    0.0
(C)    dx3/dt = -x3 - x3*x1 + x2 + x5*x3    0.0
(A')   dx4/dt = -x4*x2 + x1*x2              0.0
(B')   dx5/dt = -x5*x3 + x2*x3    0.0
(C')   dx6/dt = -x6*x1 + x3*x1              0.0
```

Semantics of Collective Behavior

"Micromodels": Continuous Time Markov Chains

- The underlying semantics of stochastic π-calculus (and stochastic interacting automata). Well established in many ways.
  - Automata with rates on transitions.
- "The" correct semantics for chemistry, executable.
  - Gillespie stochastic simulation algorithm
- Lots of advantages
  - Compositional, compact, mechanistic, etc.
- But do not give a good sense of "collective" properties.
  - Yes one can do simulation.
  - Yes one can do program analysis.
  - Yes one can perhaps do modelchecking.
  - But somewhat lacking in "analytical properties" and "predictive power".

“Macromodels”: Ordinary Differential Equations

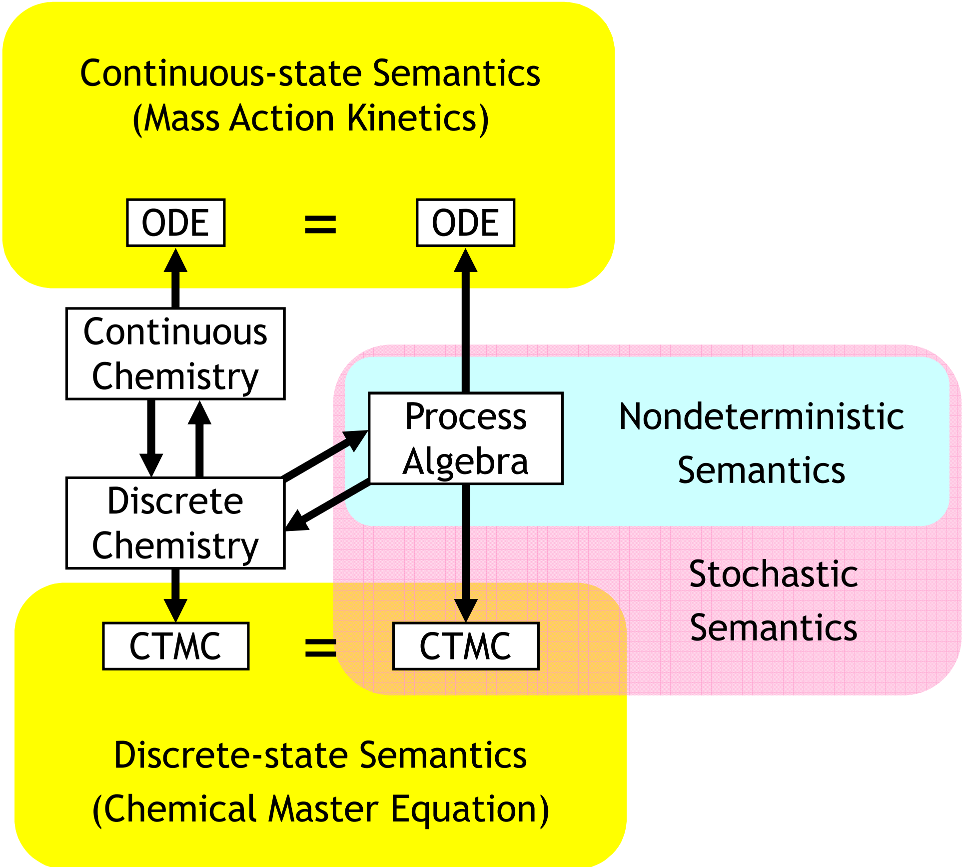
- The classical semantics of collective behavior.
  - E.g. kinetic theory of gasses.
  - They always ask: “How does you automata model relate to the 75 ODE models in the literature?”

- Going from processes/automata to ODEs <u>directly</u>:
  - *In principle*: just write down the Rate Equation:
    - Let [S] be the “number of processes in state S” as a function of time.
    - Define for each state S:

      d[S]/dt = (rate of change of the number of processes in state S)
      Cumulative rate of transitions from any state S’ to state S, times [S’],
      minus cumulative rate of transitions from S to any state S”, times [S].
  - Fairly intuitive (rate = inflow minus outflow)

- Going to ODEs <u>indirectly</u> through chemistry
  - If we first convert processes to chemical reactions, then we can convert to ODEs by standard means!

The Two Semantic Sides of Chemistry

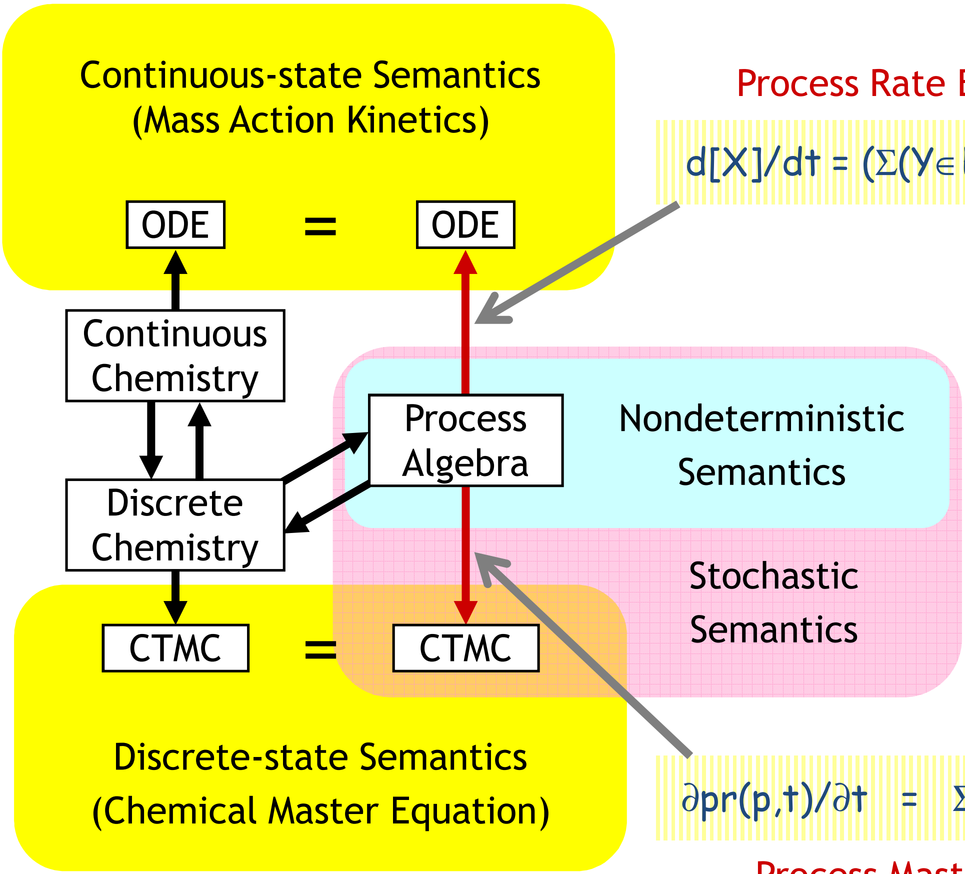


These diagrams commute via appropriate maps.

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

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

Quantitative Process Semantics



Process Rate Equation

$$d[X]/dt = (\Sigma(Y \in E)\ Accr_E(Y,X) \cdot [Y]) - Depl_E(X) \cdot [X] \qquad \text{for all } X \in E$$

Accretion

Depletion

Defined over the syntax of processes

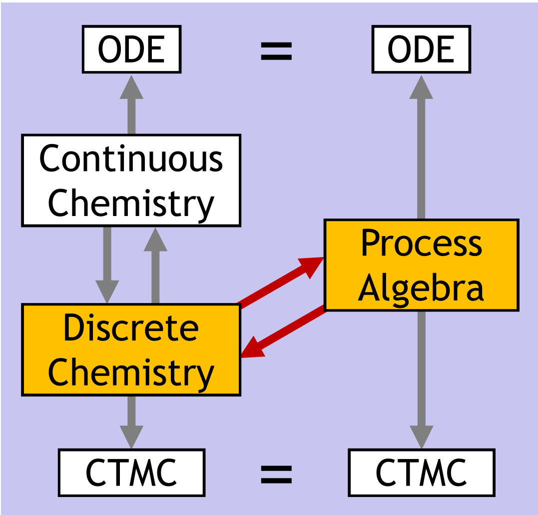
Interactions

Propensity

$$\partial pr(p,t)/\partial t \ = \ \Sigma_{i \in \mathfrak{I}}\ a_i(p-v_i) \cdot pr(p-v_i,t) - a_i(p) \cdot pr(p,t) \qquad \text{for all } p \in States(E)$$

Process Master Equation

Stochastic Processes & Discrete Chemistry



Chemical Reactions (FSRN)

$A \xrightarrow{r} B_1 + \ldots + B_n$ (n≥0)	Unary Reaction	$d[A]/dt = -r[A]$	Exponential Decay
$A_1 + A_2 \xrightarrow{r} B_1 + \ldots + B_n$ (n≥0)	Hetero Reaction	$d[A_i]/dt = -r[A_1][A_2]$	Mass Action Law
$A + A \xrightarrow{r} B_1 + \ldots + B_n$ (n≥0)	Homeo Reaction	$d[A]/dt = -2r[A]^2$	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!

Trimolecular reactions:

$A + B + C \xrightarrow{r} D$

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

$A + B \leftrightarrow AB$

$AB + C \rightarrow D$

Enzymatic reactions:

$S \xrightarrow{E\ r} P$

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

$E + S \leftrightarrow ES$

$ES \rightarrow P + E$

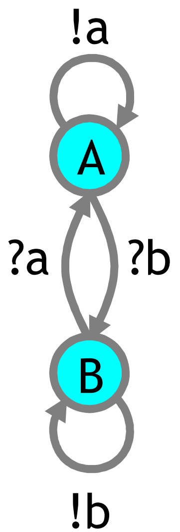
Chemical Ground Form (CGF)

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

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

(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):

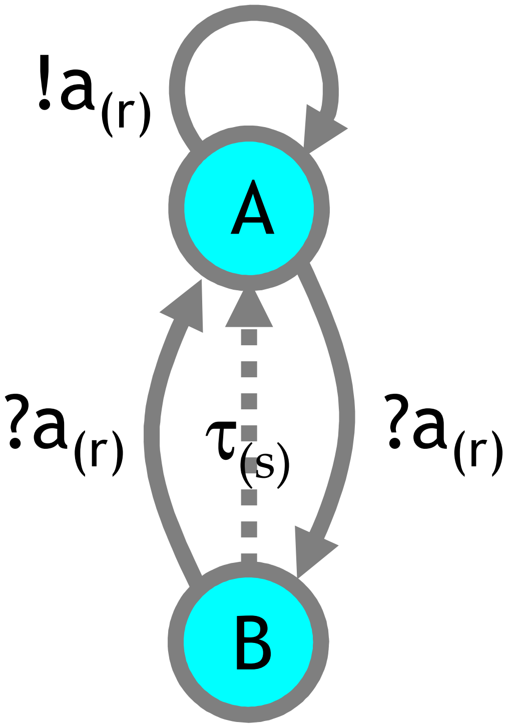
$A = !a;A \oplus ?b;B$ — Automaton in state A

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

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

From CGF to Chemistry

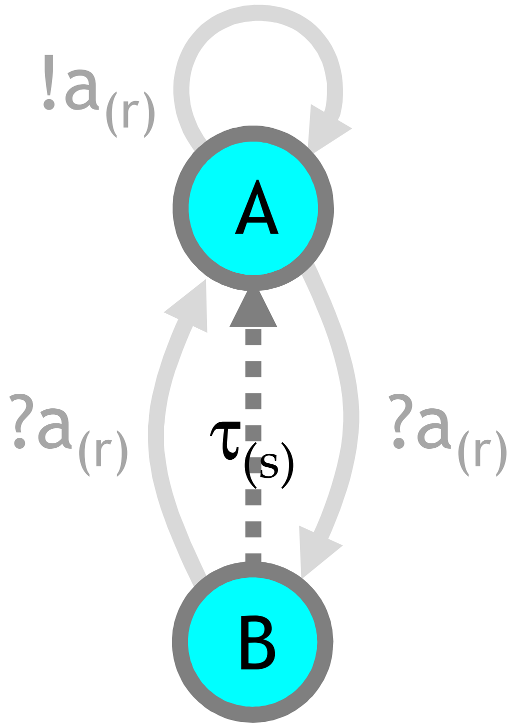
From CGF to Chemistry (by example)

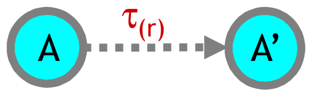


$A = !a_{(r)};A \oplus ?a_{(r)};B$

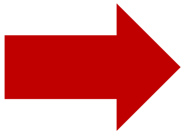
$B = ?a_{(r)};A \oplus \tau_{(s)};A$

From CGF to Chemistry (by example)





$A \rightarrow^{r} A'$

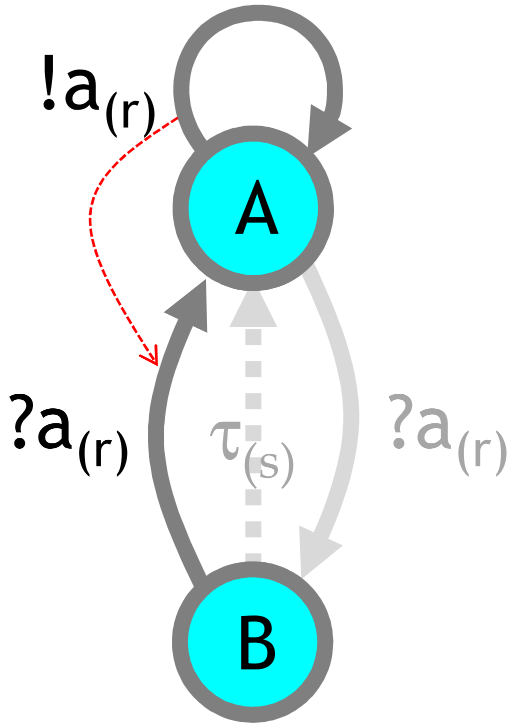


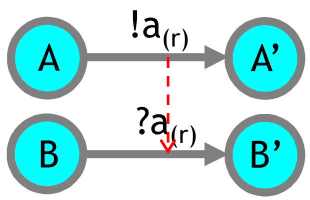
$B \rightarrow^{s} A$

$A = !a;A \oplus ?a;B$

$B = ?a;A \oplus \tau_{(s)};A$

From CGF to Chemistry (by example)





$A+B \rightarrow^{r} A'+B'$

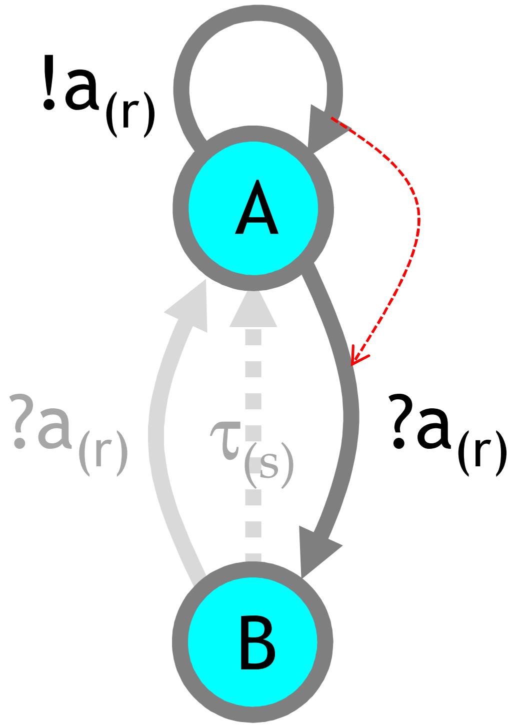
$B \rightarrow^{s} A$

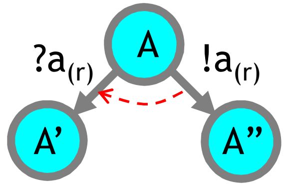
$A+B \rightarrow^{r} A+A$

$A = !a;A \oplus ?a;B$

$B = ?a;A \oplus \tau_{(s)};A$

From CGF to Chemistry (by example)





$A+A \rightarrow^{2r} A'+A''$

$B \rightarrow^{s} A$

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

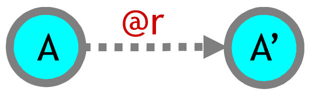
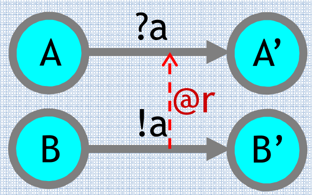
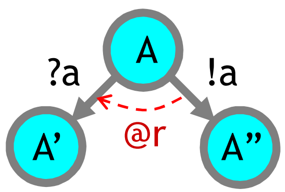
$A+A \rightarrow^{2r} A+B$

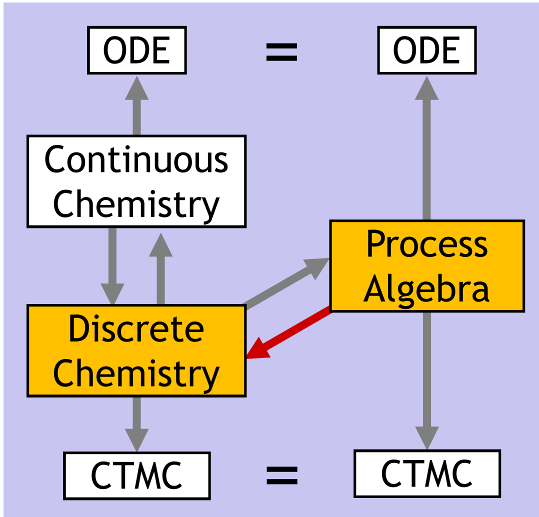
Double rate for homeo reactions

$A = !a;A \oplus ?a;B$

$B = ?a;A \oplus \tau_{(s)};A$

From CGF to Chemistry (by example)

Interacting Automata	Discrete Chemistry
initial states $A \mid A \mid \dots \mid A$	initial quantities $\#A_0$
	$A \rightarrow^{r} A'$
	$A+B \rightarrow^{r} A'+B'$
	$A+A \rightarrow^{2r} A'+A''$



From CGF to Chemistry: Ch(E)

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

E.X.i $\stackrel{\text{def}}{=}$ the i-th Å-summand of the molecule M associated with the X reagent of E

Chemical reactions for E,P: (N.B.: <...> are reaction tags to obtain multiplicity of reactions, and P is P with all the | changed to +)

$Ch(E) :=$

$\{(\langle X.i\rangle: X \to^{r} P)\ s.t.\ E.X.i = \tau_{(r)};P\} \cup$

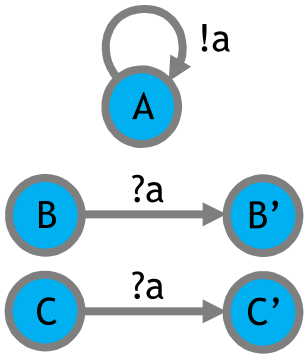
$\{(\langle X.i,Y.j\rangle: X + Y \to^{r} P + Q)\ s.t.\ X{\neq}Y,\ E.X.i = ?a_{(r)};P,\ E.Y.j = !a_{(r)};Q\} \cup$

$\{(\langle X.i,X.j\rangle: X + X \to^{2r} P + Q)\ s.t.\ E.X.i = ?a_{(r)};P,\ E.X.j = !a_{(r)};Q\}$

Initial conditions for P:

$Ch(P) := \quad P$

Entangled vs Detangled



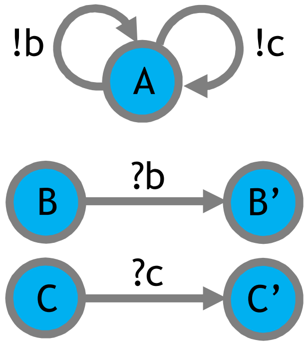
Entangled

$a\!: A+B \rightarrow^{r} A+B'$

$a\!: A+C \rightarrow^{r} A+C'$

(a@r)

A = !a;A
B = ?a;B’
C = ?a;C’
B’ = 0
C’ = 0



$b\!: A+B \rightarrow^{r} A+B'$

$c\!: A+C \rightarrow^{r} A+C'$

(b@r)
(c@r)

A = !b;A ⊕ !c;A
B = ?b;B’
C = ?c;C’
B’ = 0
C’ = 0

Entangled: Two reactions on one channel

Detangled: Two reactions on two separate channels

We need a semantics of automata that identifies automata that have the “same chemistry”.

No traditional process algebra equivalence is like this!

Entangled automata lead to more compact models than in chemistry.

Detangled automata are in simple correspondence with chemistry.

Some Syntactic Properties

- C and Ch(Pi(C)) have the same reactions
  - (and their reaction labels are in bijection)

- Def: E is detangled if each channel appears once as ?a and once as !a.

- If C is a system of chemical reactions then Pi(C) is detangled.
  - (hence chemical reactions embed into a subclass of CGFs)

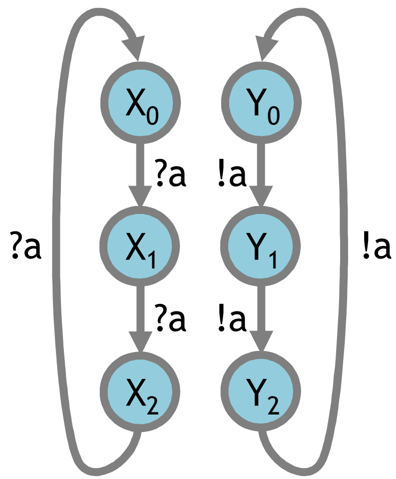
- Hence for any E, we have that Pi(Ch(E)) is detangled.
  - (E and Pi(Ch(E)) are "equivalent" CGFs, but that has to be shown later)

- Def: E,P is automata form if "|" occurs only (other than "|0") in P.

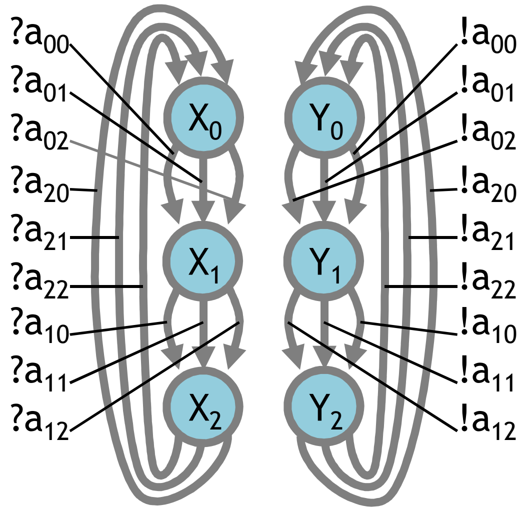
- Def: Detangle(E) is defined from Pi(Ch(E)) by replacing any occurrence pairs $?a_{(r)};(X|Y|0)$ and $!a_{(r)};0$ with $?a_{(r)};(X|0)$ and $!a_{(r)};(Y|0)$.

- If E is in automata form then Detangle(E) is (detangled and) in automata form
  - (but Pi(Ch(E)) may not be)

Entangled vs detangled



E_3



Detangle(E_3)

(closely related to
Pi(Ch(E_3)))

Chemical Parametric Form (CPF)

$E ::= 0 \;\vdots\; X(\mathbf{p})=M, E$	Reagents
$M ::= 0 \;\vdots\; \pi;P \oplus M$	Molecules
$P ::= 0 \;\vdots\; X(\mathbf{p}) \mid P$	Solutions
$\pi ::= \tau_{(r)} \;\vdots\; ?a_{(r)}(\mathbf{p}) \;\vdots\; !a_{(r)}(\mathbf{p})$	Actions
$CPF ::= E,P$	with initial conditions

$\oplus$ is stochastic choice (vs. + for chemical reactions)
0 is the null solution ($P|0 = 0|P = P$)
and null molecule ($M\oplus 0 = 0\oplus M = M$)
Each X in E is a distinct *species*
p are vectors of names
p are vectors of distinct names when in binding position
Each free name a in E is assigned a fixed rate r: $a_{(r)}$

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

Example:

$Neg(a,b) = ?a; Inh(a,b) \oplus \tau_e; (Tr(b) \mid Neg(a,b))$
$Inh(a,b) = \tau_h; Neg(a,b)$
$Tr(b) = !b; Tr(b) \oplus \tau_d; 0$
$Neg(x,x)$

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

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

CPF to CGF: Handling Parameters

Consider first the CPF subset with no communication (pure ?a, !a).

Grounding (replace parameters with constants)
where $X/\mathbf{p}$ is a name in bijection with $\langle X,\mathbf{p}\rangle$
(each $X/\mathbf{p}$ is seen as a separate *species*)

$$/(\pi_1;P_1 \oplus \dots \oplus \pi_n;P_n) =_{def} \pi_1;/(P_1) \oplus \dots \oplus \pi_n;/(P_n)$$
$$/(X_1(\mathbf{p}_1) \mid \dots \mid X_n(\mathbf{p}_n)) =_{def} X_1/\mathbf{p}_1 \mid \dots \mid X_n/\mathbf{p}_n$$

$$E ::= X_1(\mathbf{p}_1)=M_1, \dots, X_n(\mathbf{p}_n)=M_n$$
$$M ::= \pi_1;P_1 \oplus \dots \oplus \pi_n;P_n$$
$$P ::= X_1(\mathbf{p}_1) \mid \dots \mid X_n(\mathbf{p}_n)$$
$$\pi ::= \tau_r \quad ?a \quad !a$$

Let N be the set of free names occurring in E.

E_G is the **Parametric Explosion** of E (still a finite species system)
computed by replacing parameters with all combinations of free names in E

$$E_G := \{(X/\mathbf{q} = /(M\{\mathbf{p}\leftarrow\mathbf{q}\})) \text{ s.t. } (X(\mathbf{p}) = M) \in E \text{ and } \mathbf{q} \in N^{\#\mathbf{p}}\}$$
$$P_G := /P \qquad \text{(simply ground the given initial conditions once)}$$

E_G is a CGF! To obtain the chemical reactions $Ch_P(E)$, just compute $Ch_G(E_G)$

$$Ch_P(E) = Ch_G(E_G)$$

CPF to CGF: Handling Communication

Grounding (replace parameters with constants)

just one main change: now also convert each input parameter into a ground choice of all possible inputs

N is the set of free names in E,P

#**p** is the length of **p**

n/**p** is a name in bijection with <n,**p**>

X/**p** is a name in bijection with <X,**p**>

(each X/**p** is seen as a separate *species*)

$$E ::= X_1(\mathbf{p}_1)=M_1, \ldots, X_n(\mathbf{p}_n)=M_n$$
$$M ::= \pi_1;P_1 \oplus \ldots \oplus \pi_n;P_n$$
$$P ::= X_1(\mathbf{p}_1) \mid \ldots \mid X_n(\mathbf{p}_n)$$
$$\pi ::= \tau_r \quad ?a(\mathbf{p}) \quad !a(\mathbf{p})$$

$$/_N(\tau_r;P) = \tau_r;\ /_N(P)$$
$$/_N(!a_{(r)}(\mathbf{p});P) = !a/\mathbf{p}_{(r)};\ /_N(P)$$
$$/_N(?a_{(r)}(\mathbf{p});P) = \oplus(\mathbf{q}\in N^{\#\mathbf{p}}) \text{ of } ?a/\mathbf{q}_{(r)};\ /_N(P\{\mathbf{p}\leftarrow\mathbf{q}\})$$
$$/_N(\pi_1;P_1 \oplus \ldots \oplus \pi_n;P_n) = /_N(\pi_1;P_1) \oplus \ldots \oplus /_N(\pi_n;P_n)$$
$$/_N(X_1(\mathbf{p}_1) \mid \ldots \mid X_n(\mathbf{p}_n)) = X_1/\mathbf{p}_1 \mid \ldots \mid X_n/\mathbf{p}_n$$

E_G is again the **Parametric Explosion** of E

$$E_G := \{(X/\mathbf{q} = /_N(M\{\mathbf{p}\leftarrow\mathbf{q}\})) \text{ s.t. } (X(\mathbf{p}) = M) \in E \text{ and } \mathbf{q} \in N^{\#\mathbf{p}}\}$$
$$P_G := /_N(P)$$
(simply ground the given initial conditions once)

$$Ch(E) = Ch_G(E_G)$$

E_G is a again a CGF!

CPF to CGF Translation. Ex: Neg(x,x)

E =

$\boxed{\begin{array}{l} \mathrm{Neg}(a,b) = ?a;\ \mathrm{Inh}(a,b) \oplus \tau_e;\ (\mathrm{Tr}(b) \mid \mathrm{Neg}(a,b)) \\ \mathrm{Inh}(a,b) = \tau_h;\ \mathrm{Neg}(a,b) \\ \mathrm{Tr}(b) = !b;\ \mathrm{Tr}(b) \oplus \tau_d;\ 0 \\ \mathrm{Neg}(x,x) \end{array}}$

----- initialization -----

E_c:= {Neg/x,x = ?x; Inh/x,x $\oplus$ τ_e; (Tr/x | Neg/x,x)}

----- iteration 1 -----

C := {Neg/x,x $\to^e$ Tr/x + Neg/x,x }

E_c:= {Neg/x,x = ?x; Inh/x,x $\oplus$ τ_e; (Tr/x | Neg/x,x)

Tr/x = !x; Tr/x $\oplus$ τ_d; 0}

----- iteration 2 -----

C := {Neg/x,x $\to^e$ Tr/x + Neg/x,x

Tr/x $\to^d$ 0

Tr/x + Neg/x,x $\to^{\rho(x)}$ Tr/x + Inh/x,x }

E_c:= {Neg/x,x = ?x; Inh/x,x $\oplus$ τ_e; (Tr/x | Neg/x,x)

Tr/x = !x; Tr/x $\oplus$ τ_d; 0

Inh/x,x = τ_h; Neg/x,x}

----- iteration 3 -----

C := {Neg/x,x $\to^e$ Tr/x + Neg/x,x

Tr/x $\to^d$ 0

Tr/x + Neg/x,x $\to^{\rho(x)}$ Tr/x + Inh/x,x

Inh/x,x $\to^h$ Neg/x,x}

E_c:= no change

----- termination -----

$\boxed{\begin{array}{l} \mathrm{Neg/x,x} \to^e \mathrm{Tr/x} + \mathrm{Neg/x,x} \\ \mathrm{Tr/x} \to^d 0 \\ \mathrm{Tr/x} + \mathrm{Neg/x,x} \to^{\rho(x)} \mathrm{Tr/x} + \mathrm{Inh/x,x} \\ \mathrm{Inh/x,x} \to^h \mathrm{Neg/x,x} \\ \mathrm{Neg/x,x} \end{array}}$

From Chemistry to CGF

From Chemistry to CGF (by example)

x: $B \rightarrow^{s} A$

b: $A+B \rightarrow^{r} A+A$

c: $A+A \rightarrow^{2r} A+B$

Unique reaction names

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A			
B			

Reactions names

Half-rate for homeo reactions

Species

From Chemistry to CGF (by example)

x: $B \rightarrow^{s} A$

b: $A+B \rightarrow^{r} A+A$

c: $A+A \rightarrow^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A			
B	τ;A		

1: Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow k_i\ P_i$
add $\tau;P_i$ to $<X,v_{ij}>$.

From FSRN to CGF (by example)

x: $B \to^{s} A$

b: $A+B \to^{r} A+A$

c: $A+A \to^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A		?;A|A	
B	τ;A	!;0	

1: Fill the matrix by columns:

Degradation reaction v_i: $X \to k_i\ P_i$
add $\tau;P_i$ to <X,v_{ij}>.
Hetero reaction v_i: $X+Y \to k_i\ P_i$
add $?;P_i$ to <X,v_i> and !;0 to <Y,v_i>

From FSRN to CGF (by example)

x: $B \rightarrow^{s} A$

b: $A+B \rightarrow^{r} A+A$

c: $A+A \rightarrow^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A		?;A|A	?;A|B !;0
B	τ;A	!;0	

1: Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow_{k_i} P_i$

add $\tau;P_i$ to $\langle X,v_{ij}\rangle$.

Hetero reaction v_i: $X+Y \rightarrow_{k_i} P_i$

add $?;P_i$ to $\langle X,v_i\rangle$ and $!;0$ to $\langle Y,v_i\rangle$

Homeo reaction v_i: $X+X \rightarrow_{k_i} P_i$

add $?;P_i$ and $!;0$ to $\langle X,v_i\rangle$

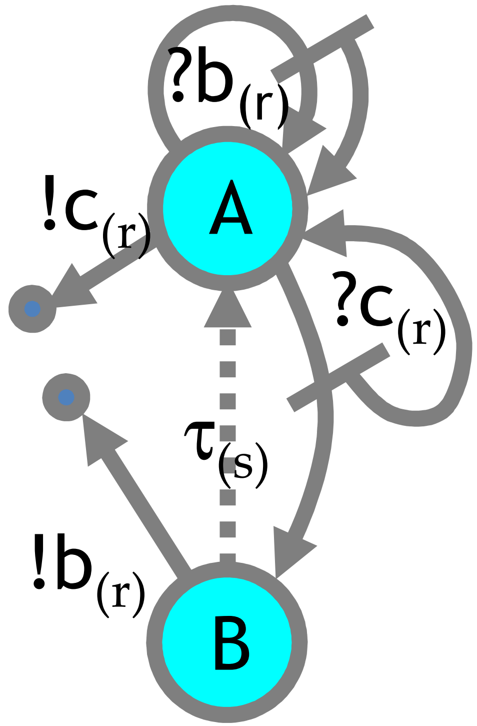
From FSRN to CGF (by example)

$x: \quad B \to^{s} A$

$b: \quad A+B \to^{r} A+A$

$c: \quad A+A \to^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A		?;A|A	?;A|B !;0
B	τ;A	!;0	



1: Fill the matrix by columns:

Degradation reaction v_i: $X \to k_i\ P_i$

add $\tau;P_i$ to $\langle X, v_{ij} \rangle$.

Hetero reaction v_i: $X+Y \to k_i\ P_i$

add $?;P_i$ to $\langle X, v_i \rangle$ and $!;0$ to $\langle Y, v_i \rangle$

Homeo reaction v_i: $X+X \to k_i\ P_i$

add $?;P_i$ and $!;0$ to $\langle X, v_i \rangle$

2: Read the result by rows:

$A = ?b_{(r)};(A|A) \ \oplus \ ?c_{(r)};(A|B) \ \oplus \ !c_{(r)};0$

$B = \tau_{(s)};A \ \oplus \ !b_{(r)};0$

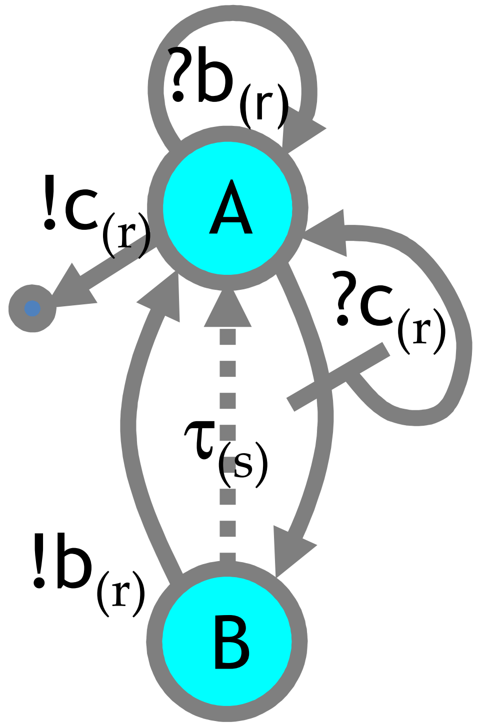
From FSRN to CGF (by example)

$x: \quad B \rightarrow^{s} A$

$b: \quad A+B \rightarrow^{r} A+A$

$c: \quad A+A \rightarrow^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A		$?;A$	$?;A\|B$ $!;0$
B	$\tau;A$	$!;A$	



1: Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow k_i\ P_i$

add $\tau;P_i$ to $<X,v_{ij}>$.

Hetero reaction v_i: $X+Y \rightarrow k_i\ P_i$

add $?;P_i$ to $<X,v_i>$ and $!;0$ to $<Y,v_i>$

Homeo reaction v_i: $X+X \rightarrow k_i\ P_i$

add $?;P_i$ and $!;0$ to $<X,v_i>$

2: Read the result by rows:

$A = ?b_{(r)};A \oplus ?c_{(r)};(A|B) \oplus !c_{(r)};0$

$B = \tau_{(s)};A \oplus !b_{(r)};A$

From FSRN to CGF (by example)

x: $B \to^{s} A$

b: $A+B \to^{r} A+A$

c: $A+A \to^{2r} A+B$

	$x_{(s)}$	$b_{(r)}$	$c_{(r)}$
A		?;A	?;B !;A
B	τ;A	!;A	

1: Fill the matrix by columns:

Degradation reaction v_i: $X \to^{k_i} P_i$

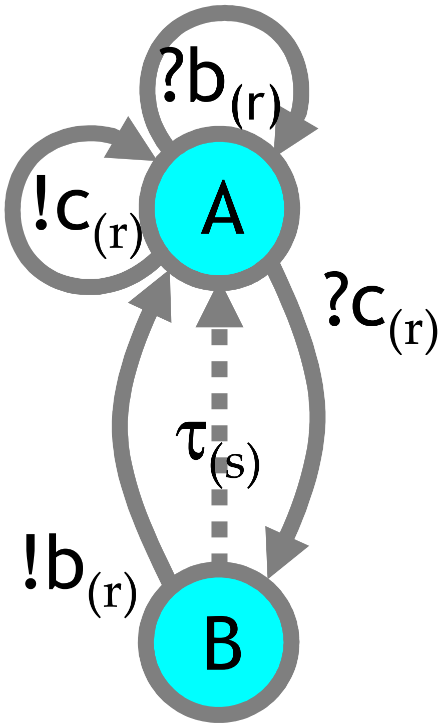
add τ;P_i to <X,v_{ij}>.

Hetero reaction v_i: $X+Y \to^{k_i} P_i$

add ?;P_i to <X,v_i> and !;0 to <Y,v_i>

Homeo reaction v_i: $X+X \to^{k_i} P_i$

add ?;P_i and !;0 to <X,v_i>



2: Read the result by rows:

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

$B = \tau_{(s)};A \oplus !b_{(r)};A$

From Chemistry to Automata (by example)

v_1: $A+B \rightarrow_{k_1} C+C$

v_2: $A+C \rightarrow_{k_2} D$

v_3: $C \rightarrow_{k_3} E+F$

v_4: $F+F \rightarrow_{k_4} B$

Interaction Matrix

channels and rates (1 per reaction)

Half-rate for homeo reactions

definitions (1 per species)

	$v_{1(k1)}$	$v_{2(k2)}$	$v_{3(k3)}$	$v_{4(k4/2)}$
A	?;(C|C)	?;D		
B	!;0			
C		!;0	τ;(E|F)	
D				
E				
F				?;B !;0

1: Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow k_i\ P_i$

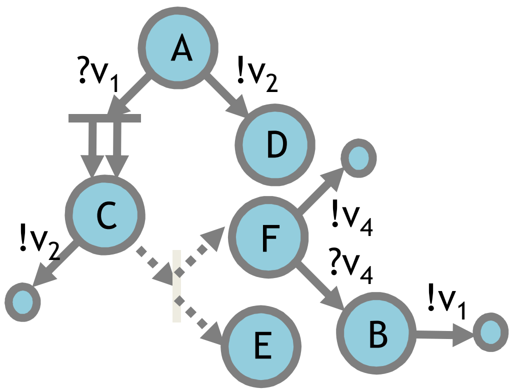
add $\tau;P_i$ to $\langle X, v_{ij} \rangle$.

Hetero reaction v_i: $X+Y \rightarrow k_i\ P_i$

add $?;P_i$ to $\langle X, v_i \rangle$ and $!;0$ to $\langle Y, v_i \rangle$

Homeo reaction v_i: $X+X \rightarrow k_i\ P_i$

add $?;P_i$ and $!;0$ to $\langle X, v_i \rangle$



2: Read the result by rows:

$A = ?v_{1(k1)};(C|C) \oplus ?v_{2(k2)};D$

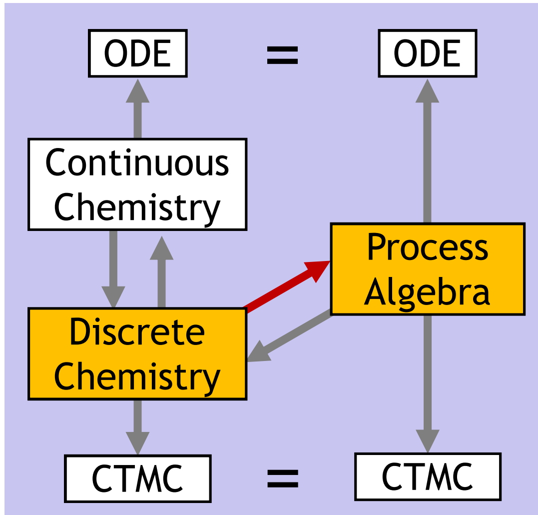
$B = !v_{1(k1)};0$

$C = !v_{2(k2)};0 \oplus \tau_{k3};(E|F)$

$D = 0$

$E = 0$

$F = ?v_{4(k4/2)};B \oplus !v_{4(k4/2)};0$

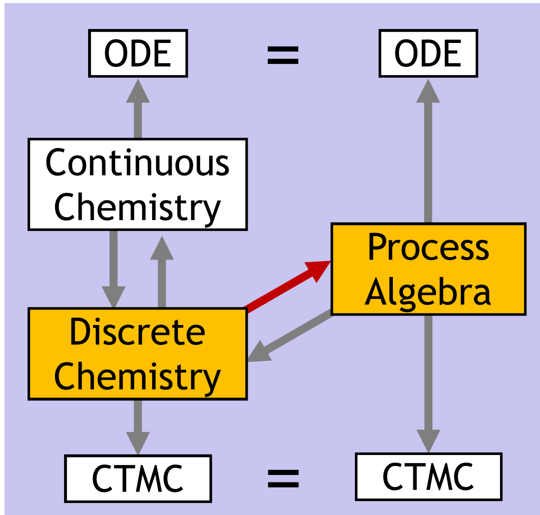


From Chemistry to CGF: Pi(C)

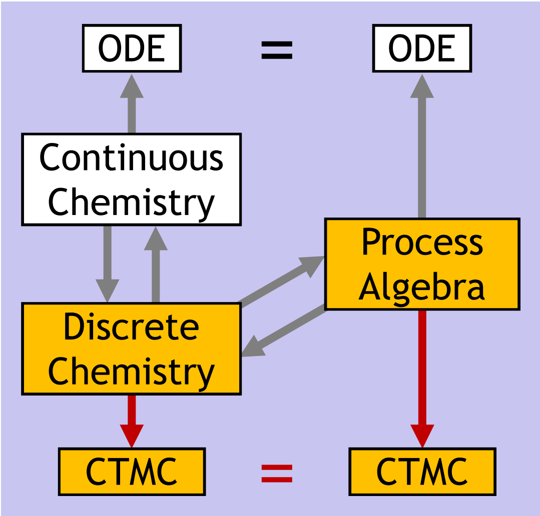
v:	$X \rightarrow^{r} Y_1 + \ldots + Y_n + 0$	Unary Reaction
v:	$X_1 + X_2 \rightarrow^{r} Y_1 + \ldots + Y_n + 0$	Binary Reaction

From uniquely-labeled (v:) chemical reactions C to a CGF Pi(C):

$$
\begin{aligned}
\mathrm{Pi}(C) \;=\; \{(X = \;& \oplus((v{:}\ X \rightarrow^{k} P) \in C) \textit{ of } (\tau_{(k)};P) && \oplus \\
& \oplus((v{:}\ X{+}Y \rightarrow^{k} P) \in C \text{ and } Y \neq X) \textit{ of } (?v_{(k)};P) && \oplus \\
& \oplus((v{:}\ Y{+}X \rightarrow^{k} P) \in C \text{ and } Y \neq X) \textit{ of } (!v_{(k)};0) && \oplus \\
& \oplus((v{:}\ X{+}X \rightarrow^{k} P) \in C) \textit{ of } (?v_{(k/2)};P \oplus !v_{(k/2)};0) &&) \\
& \textit{s.t. } X \text{ is a species in } C\}
\end{aligned}
$$



Discrete-State Semantics



Discrete Semantics of Reactions

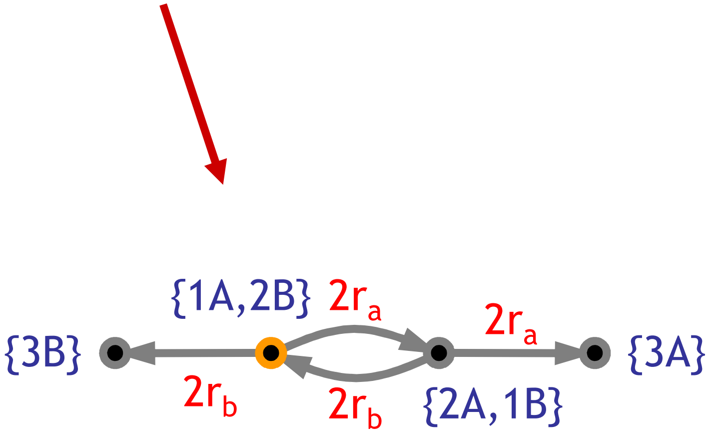
Syntax:

$A+B \to^{r} A+A$

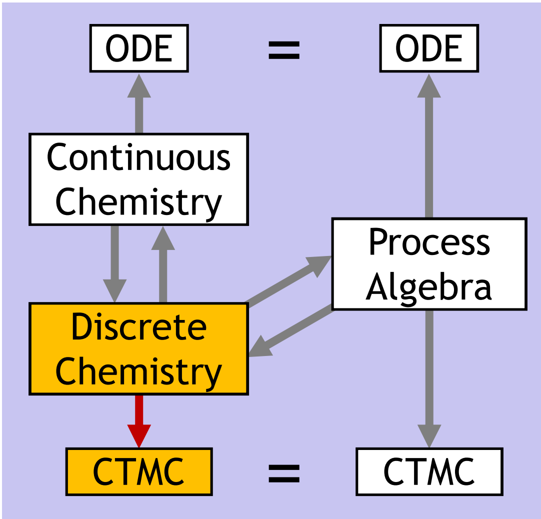
$A+B \to^{r} B+B$

$A+B+B$

Semantics:

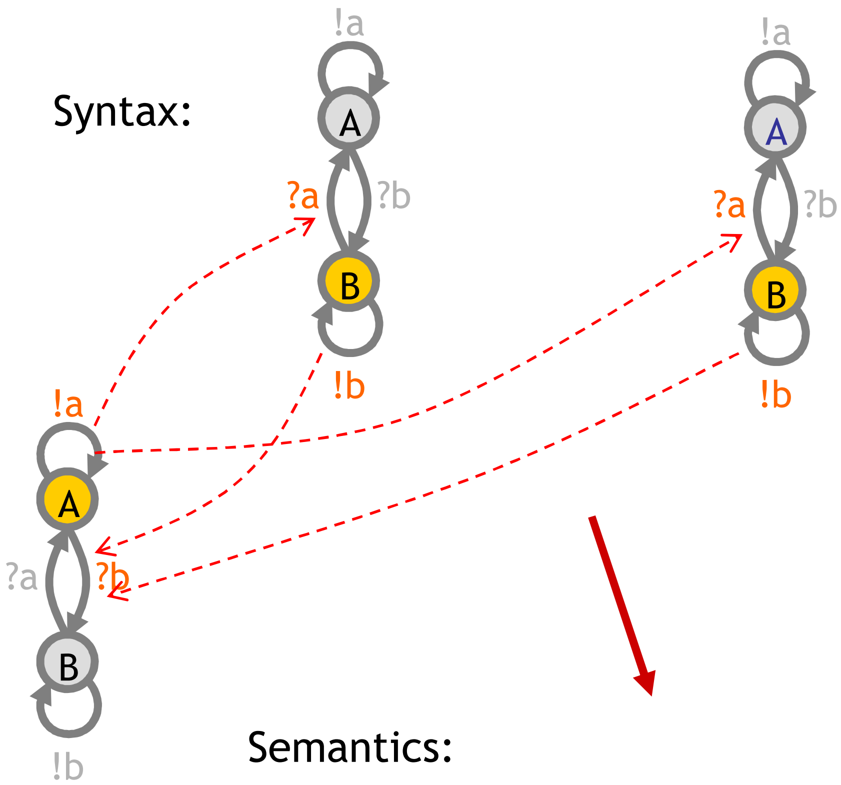


CTMC

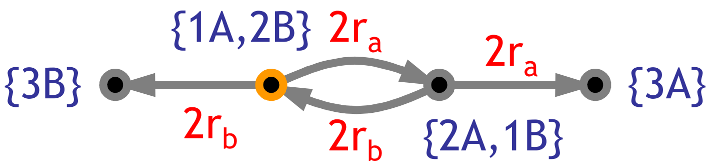


Discrete Semantics of Reagents

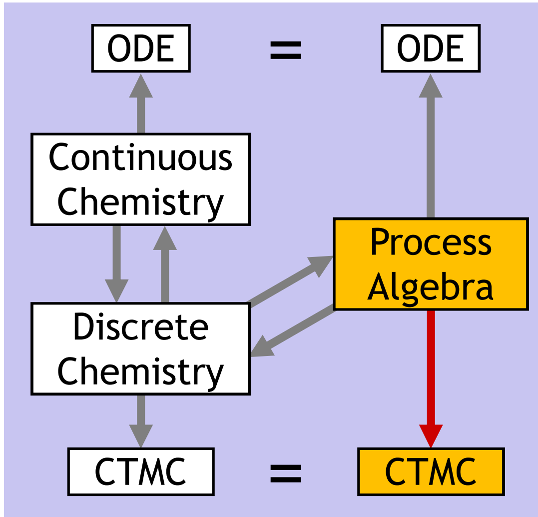
Syntax:



Semantics:

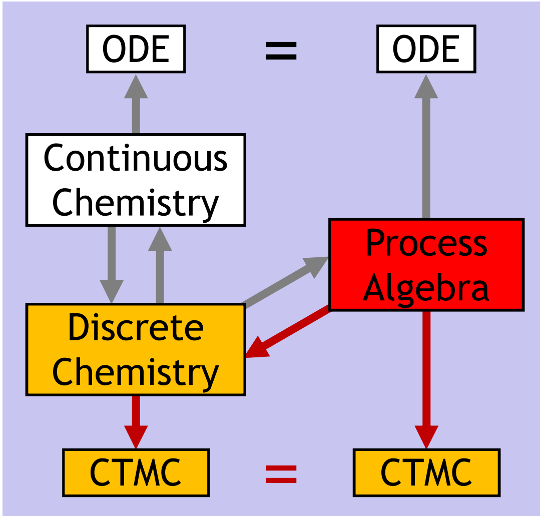


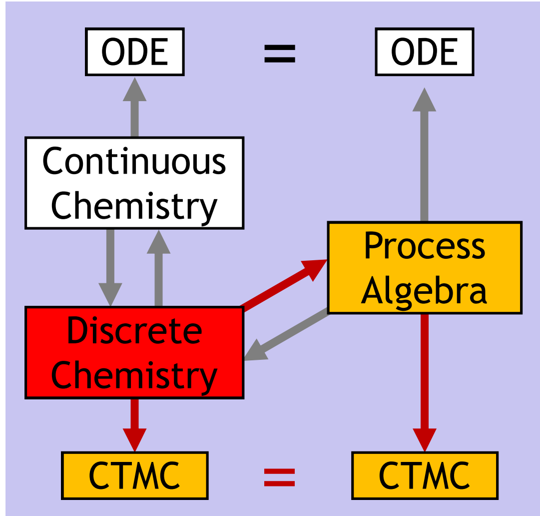
CTMC



Discrete State Equivalence

- Def: ≋ is equivalent CTMC's (isomorphic graphs with same rates).
- Thm: E ≋ Ch(E)
- Thm: C ≋ Pi(C)





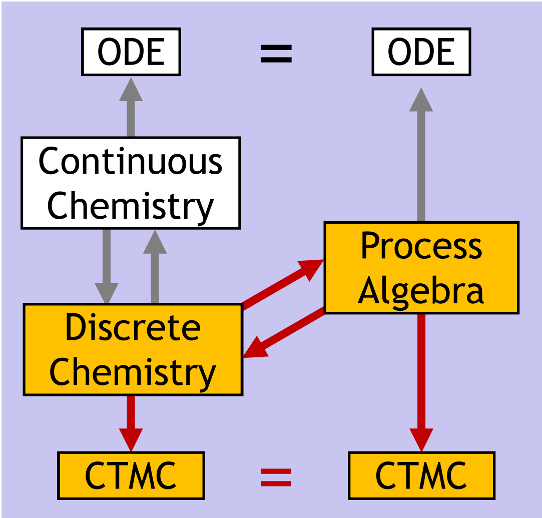
- For each E there is an E' ≋ E that is detangled (E' = Pi(Ch(E)))
- For each E in automata form there is an an E' ≋ E that is detangled and in automata form (E' = Detangle(E)).

Interacting Automata = Discrete Chemistry

This is enough to establish that the process algebra is really faithful to the chemistry.

But CTMC are not the "ultimate semantics" because there are still questions of when two different CTMCs are actually equivalent (e.g. "lumping").

The "ultimate semantics" of chemistry is the *Chemical Master Equation* (derivable from the Chapman-Kolmogorov equation of the CTMC).



Q?