Molecules as Automata

Representing Biochemical Systems as Collectives of Interacting Automata

Luca Cardelli

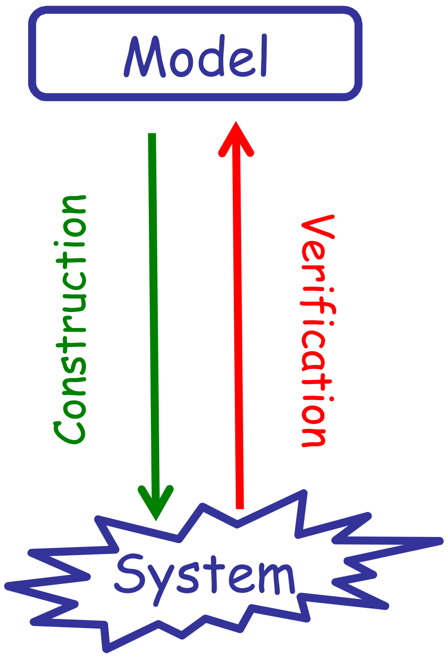
Microsoft Research

University of Western Ontario
London Ontario, 2008-08-20

http://LucaCardelli.name

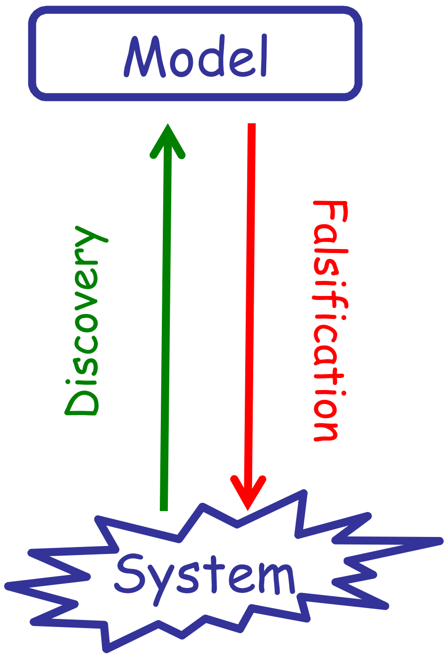
Scientific Method vs. Engineering Method

Engineering Method



Direct Engineering
(Synthetic Biology)

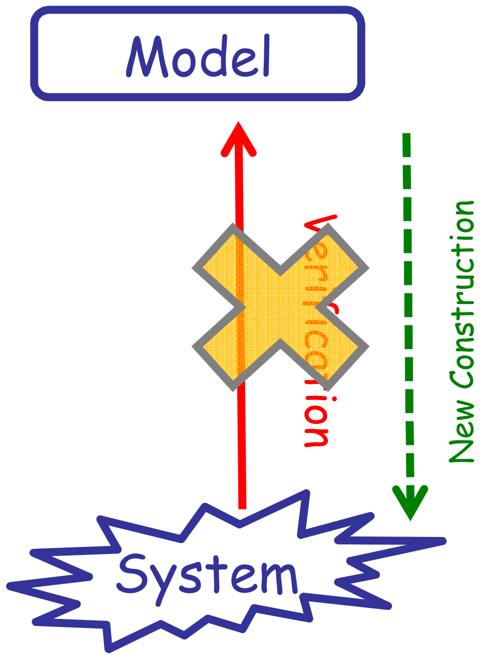
Scientific Method



Reverse Engineering
(Systems Biology)

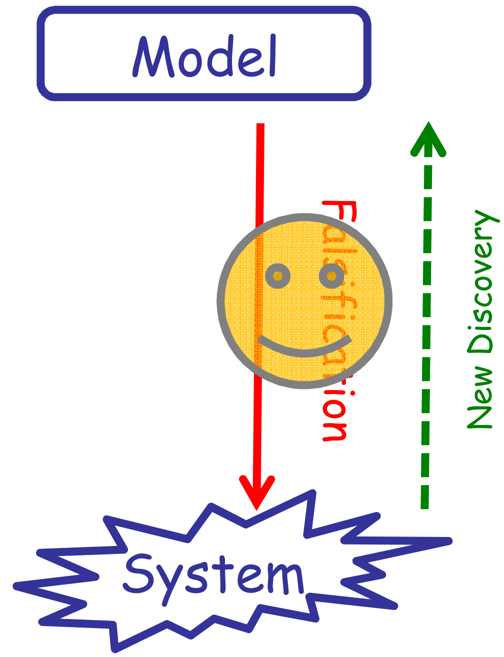
Scientific Method vs. Engineering Method

Engineering Method



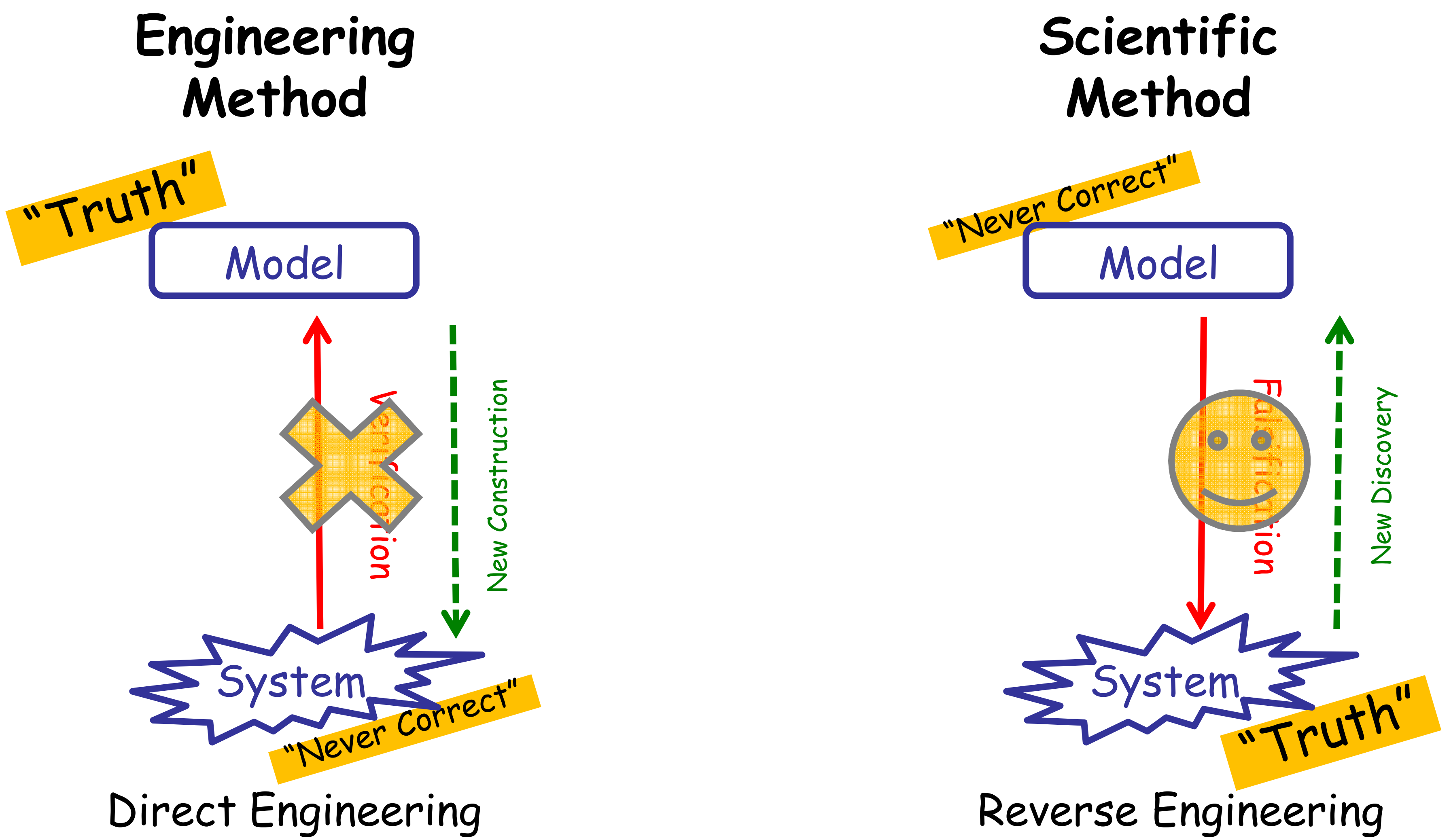
Direct Engineering

Scientific Method

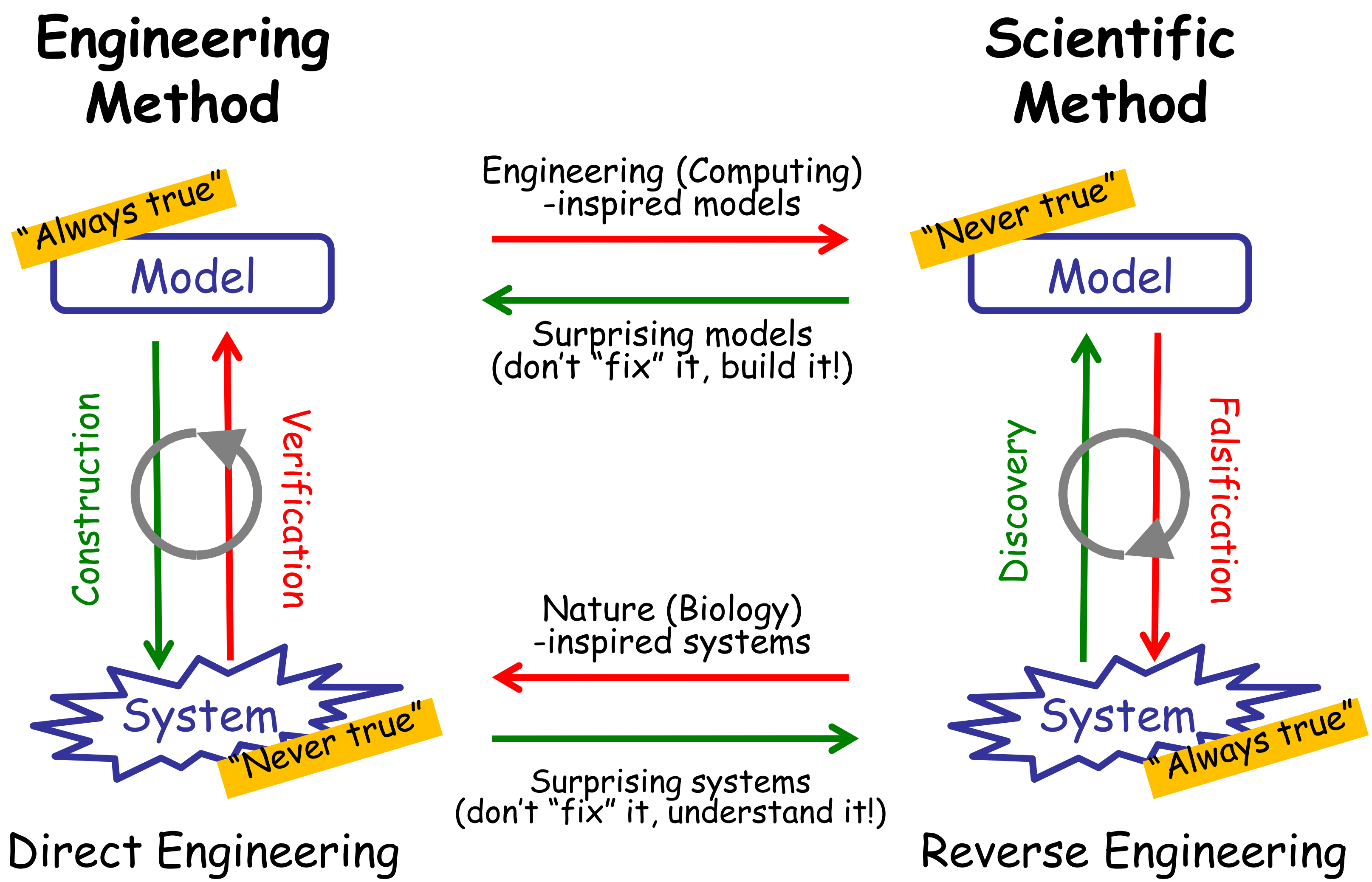


Reverse Engineering

Scientific Method vs. Engineering Method



Scientific Method vs. Engineering Method

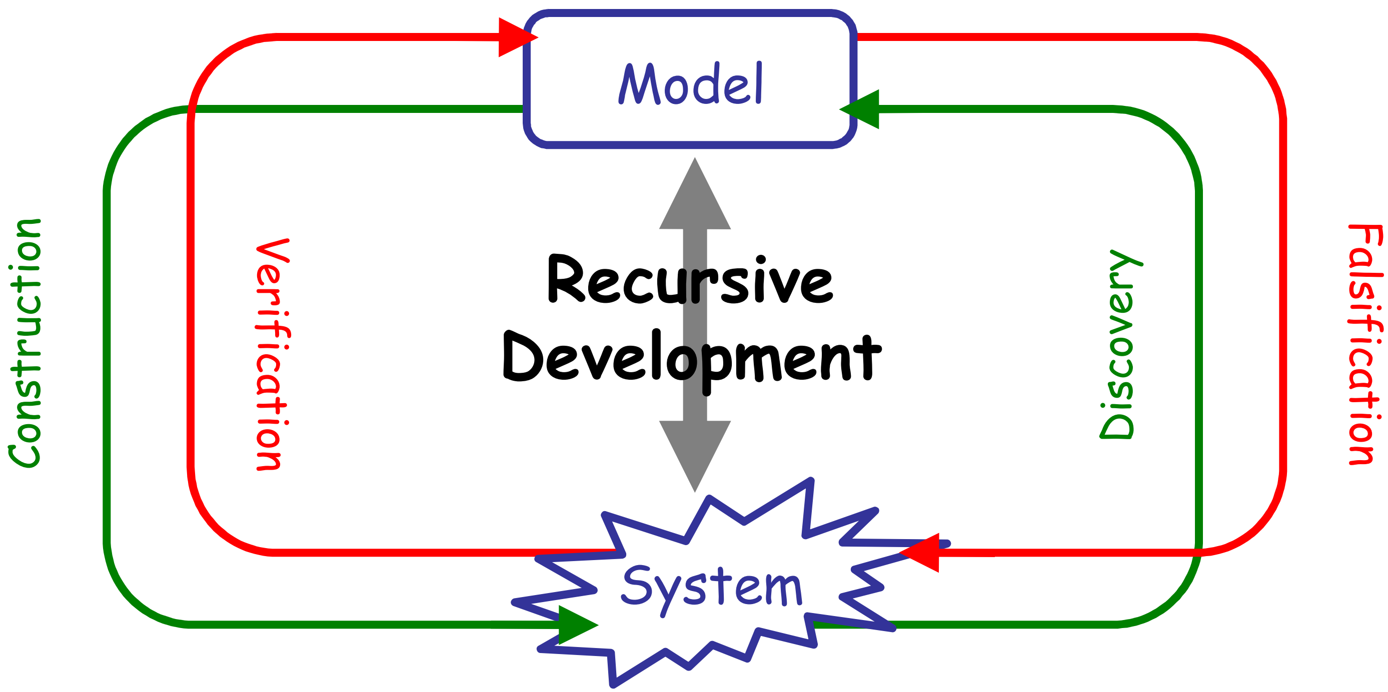


Scientific Method vs. Engineering Method

When the models and the systems are *both* too complex to *either* be the full Truth

Combined Method

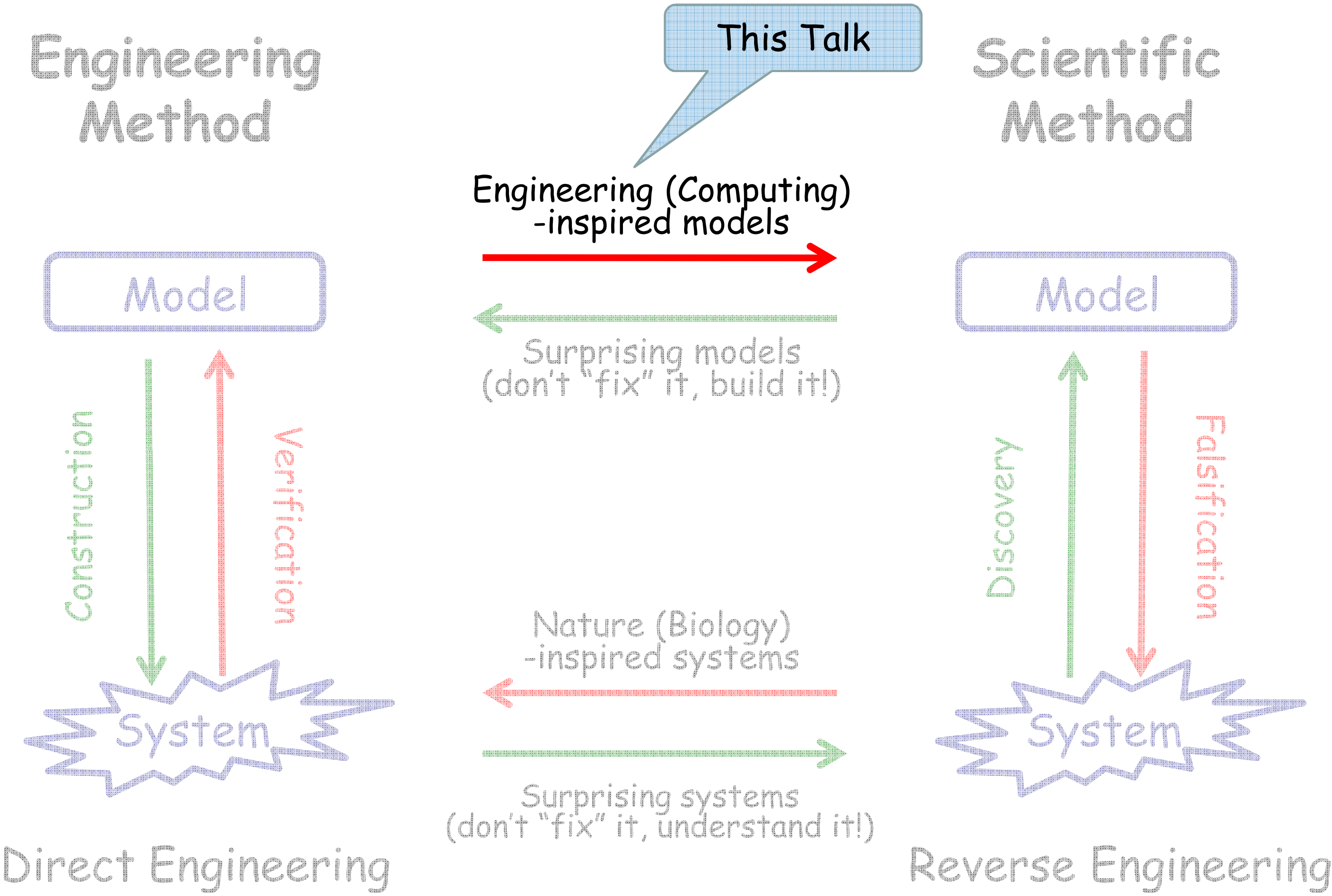
The models that we discover should be suitable for construction



The systems that we build should be suitable for discovery

2008-08-20

Scientific Method vs. Engineering Method



Motivation: Cells Compute

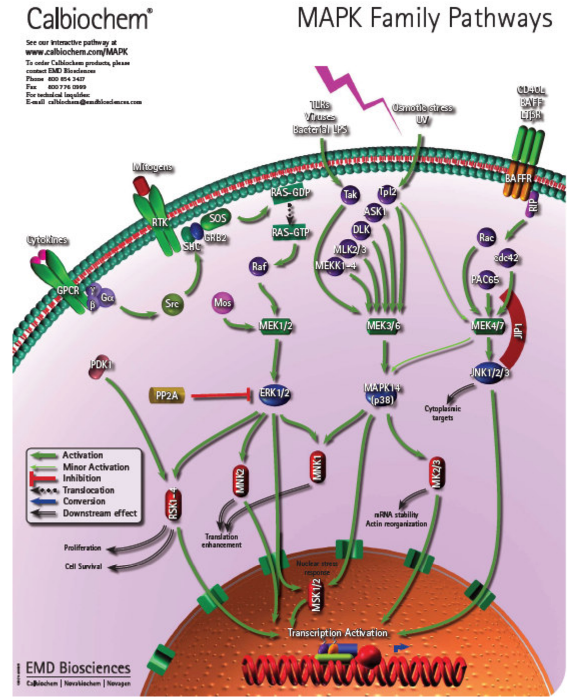
- No survival without computation!
  - Finding food
  - Avoiding predators

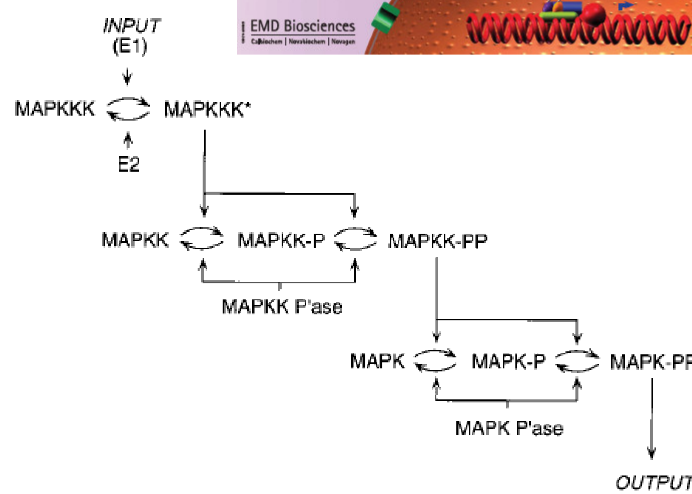
- *How* do they compute?
  - Unusual computational paradigms.
  - Proteins: do they work like electronic circuits?
  - Genes: what kind of software is that?

- 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.

(Macro-) Molecules as (Interacting) Automata

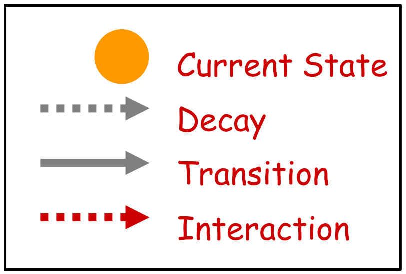
Process Algebra

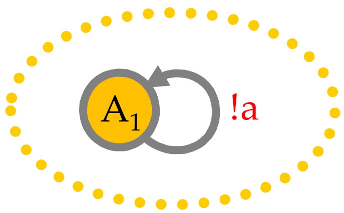
[Hoare, Milner, Pnueli, etc.]

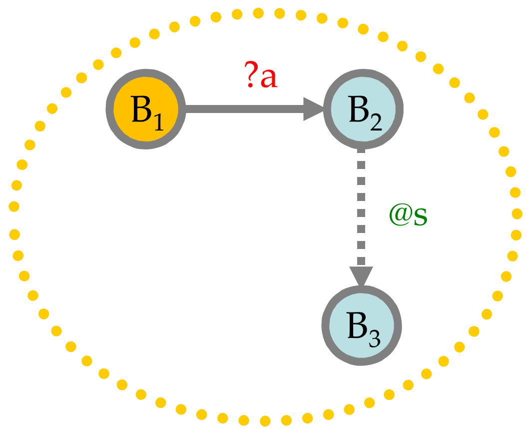
- Reactive systems (living organisms, computer networks, operating systems, …)
  - Math is based on *entities that react/interact with their environment ("processes")*, not on functions from domains to codomains.
- Concurrent
  - Events (reactions/interactions) happen concurrently and asynchronously, 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



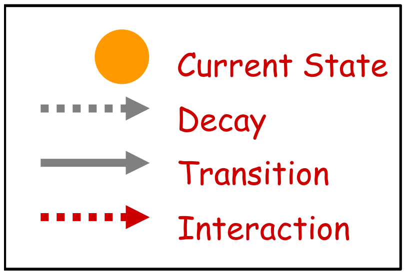


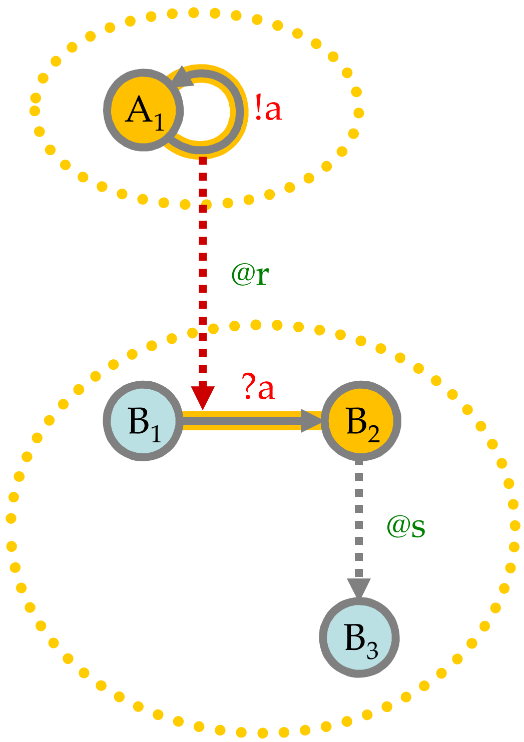


A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge)
?a, !a	indicate *complementary actions*,
@r, @s	are rates

Kinetic laws:

Interacting Automata

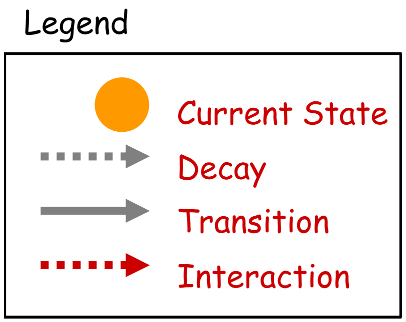


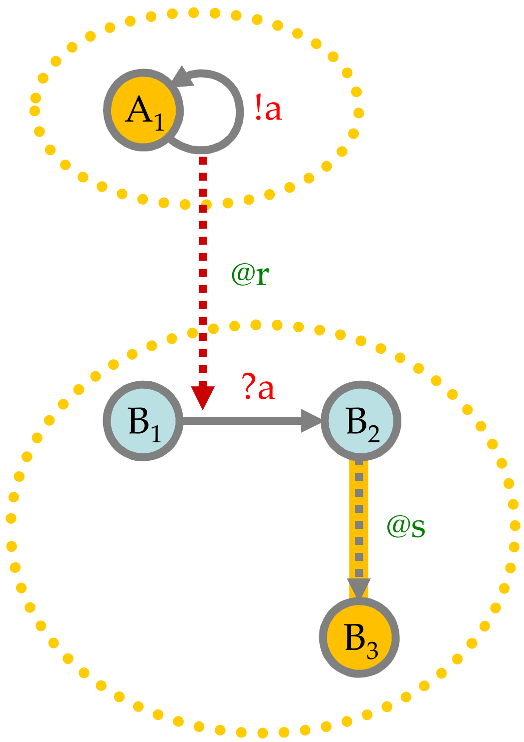


A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge, shape)
?a, !a	indicate *complementary actions*, joined by an interaction arrow
@r, @s	are rates

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

Interacting Automata

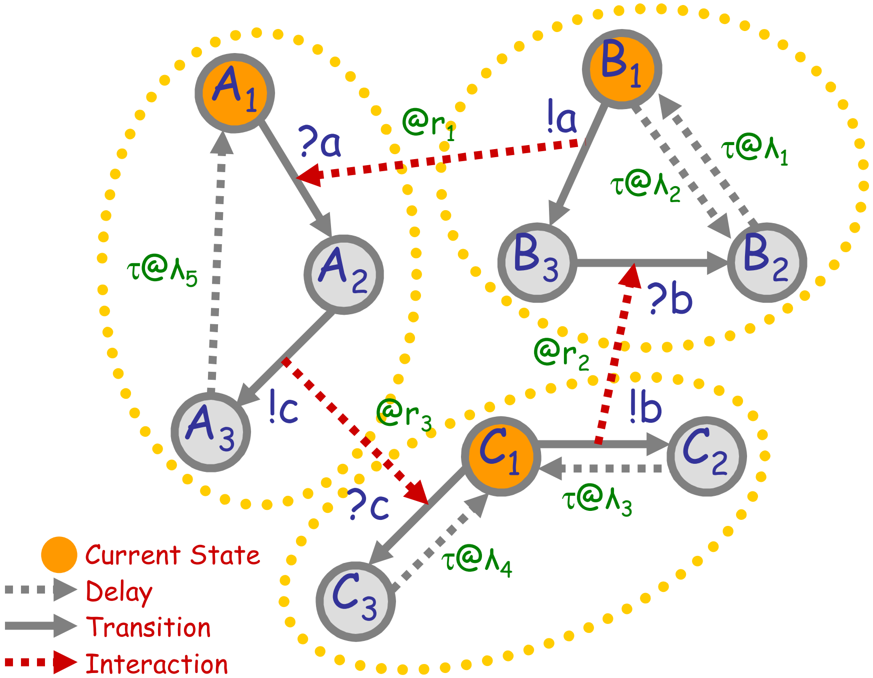




A_1	is a *state*
a	is a *channel* i.e. a named *interaction interface* (e.g. a surface patch)
?,!	indicate any *complementarity* of interaction (e.g. charge)
?a, !a	indicate *complementary actions*, joined by an interaction arrow
@r, @s	are rates

Kinetic laws: ***Two complementary actions may result in an interaction.*** ***A decay may happen spontaneously.***

Interacting Automata



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

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

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

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

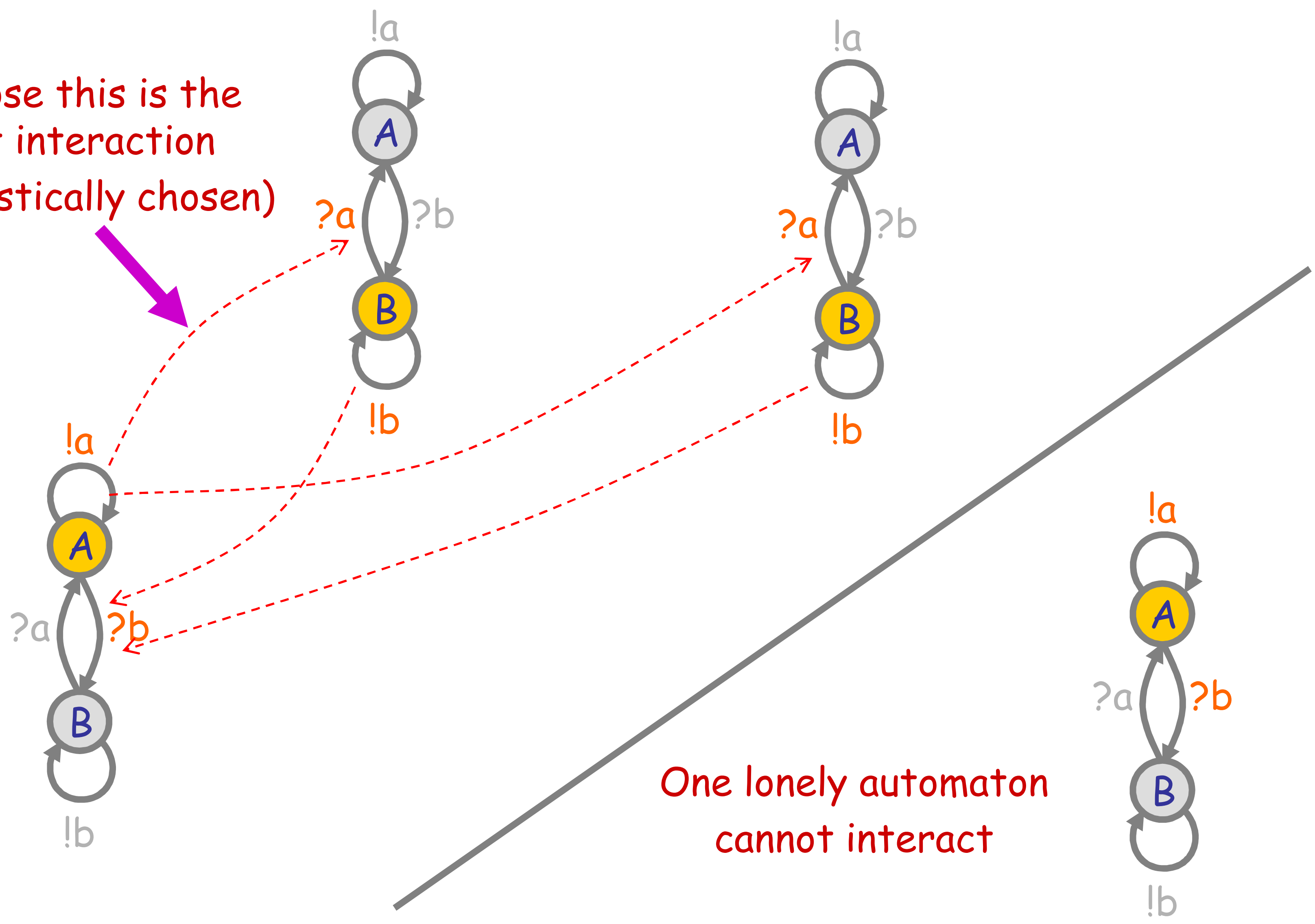
Automata

$A_1 \mid B_1 \mid C_1$

The system and initial state

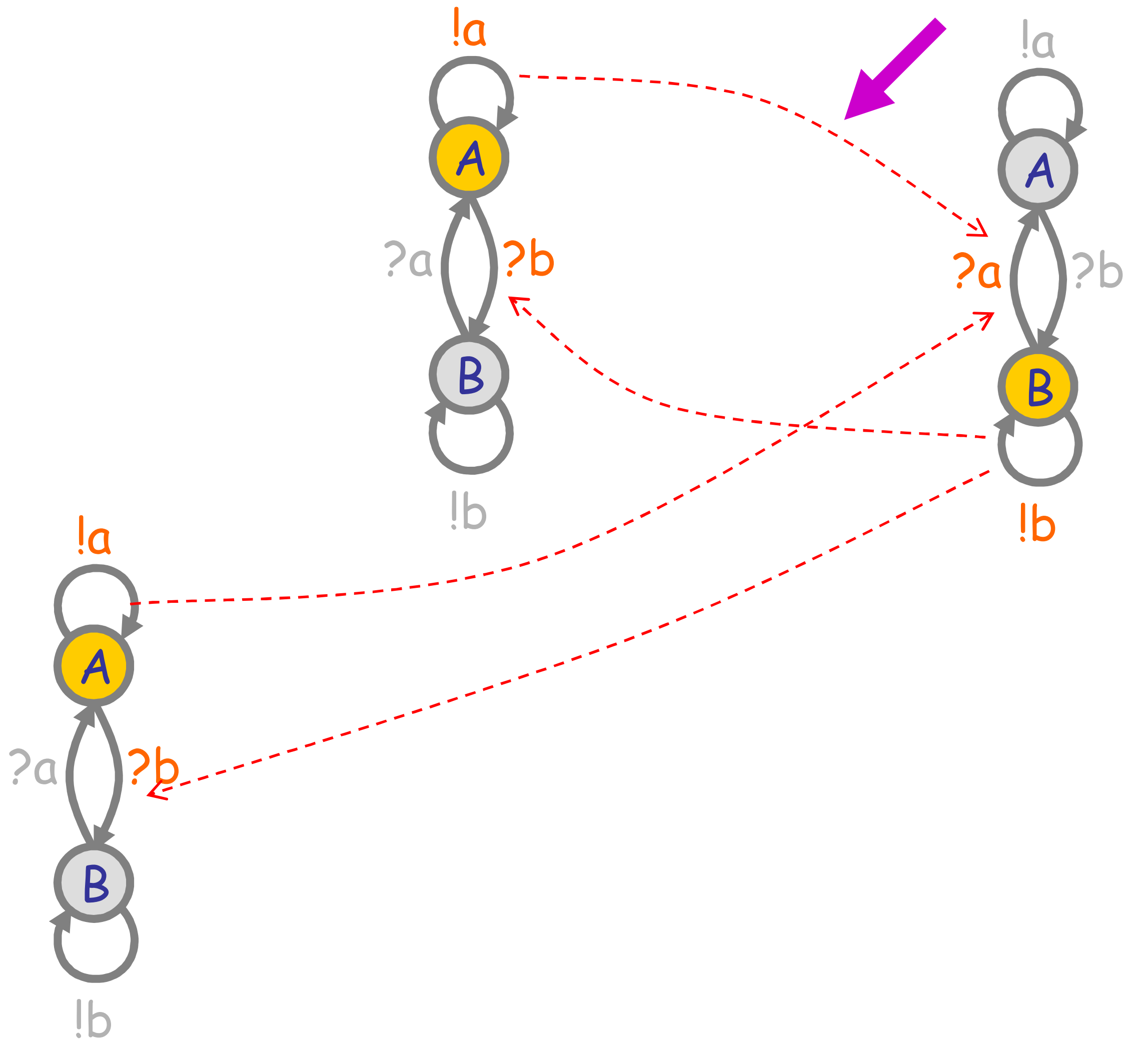
Interactions in a Population

Suppose this is the next interaction (stochastically chosen)

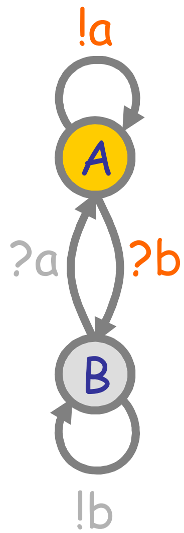


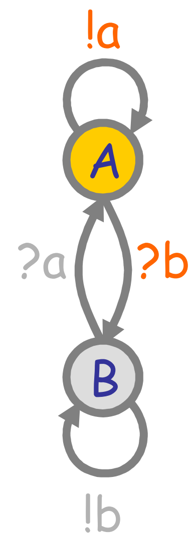
One lonely automaton cannot interact

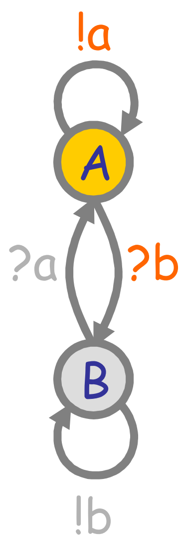
Interactions in a Population



Interactions in a Population

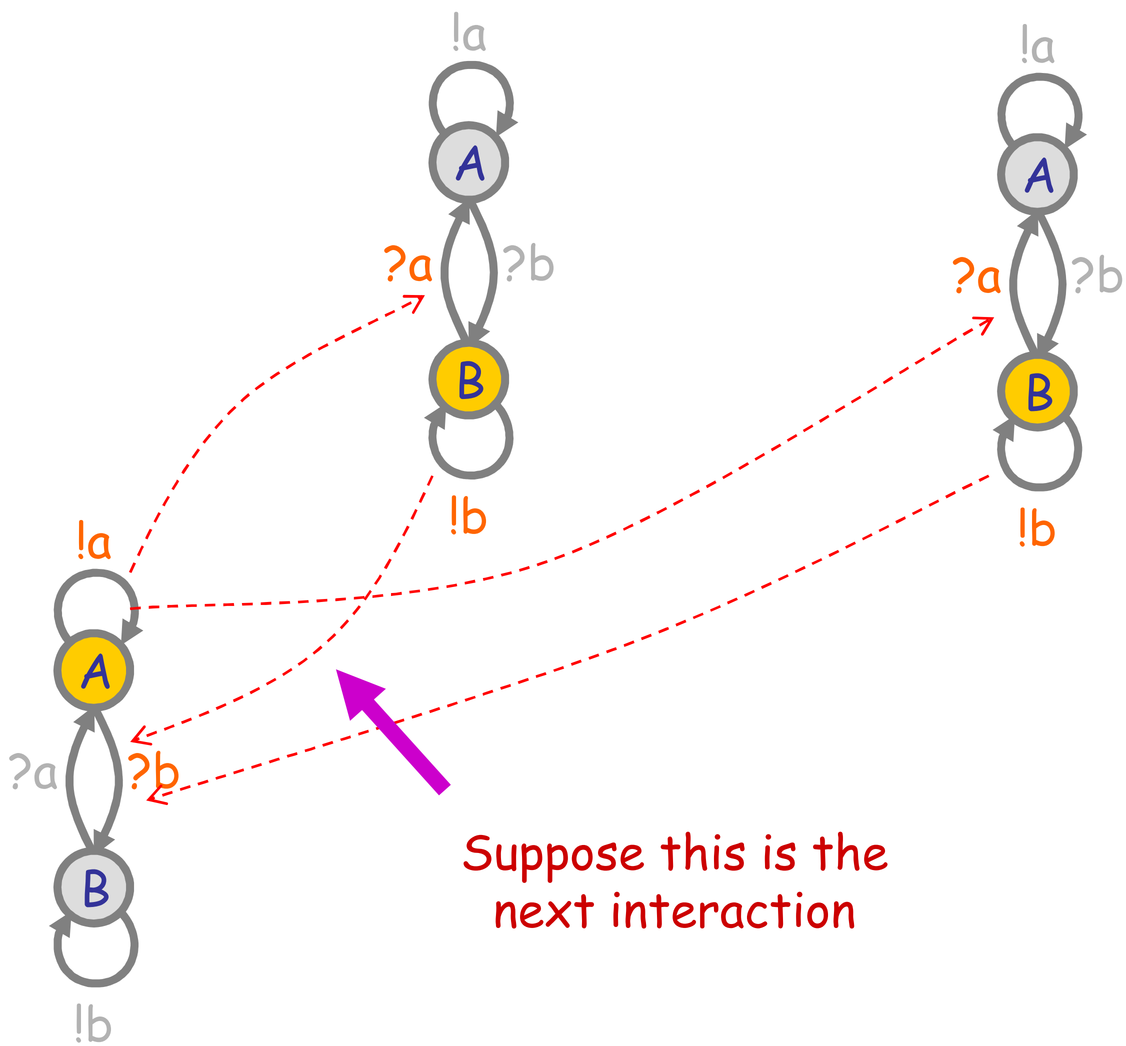






All-A stable population

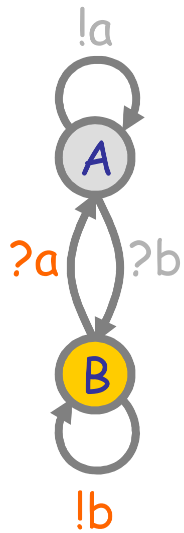
Interactions in a Population (2)

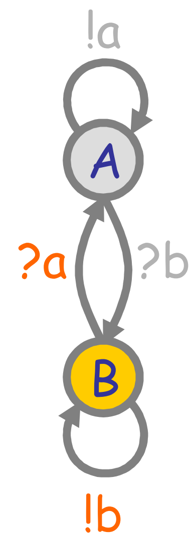


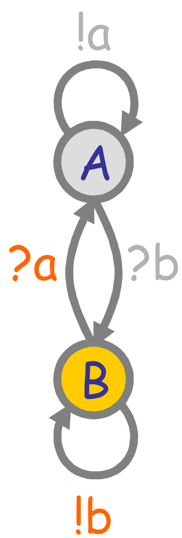
Suppose this is the next interaction

Luca Cardelli

Interactions in a Population (2)



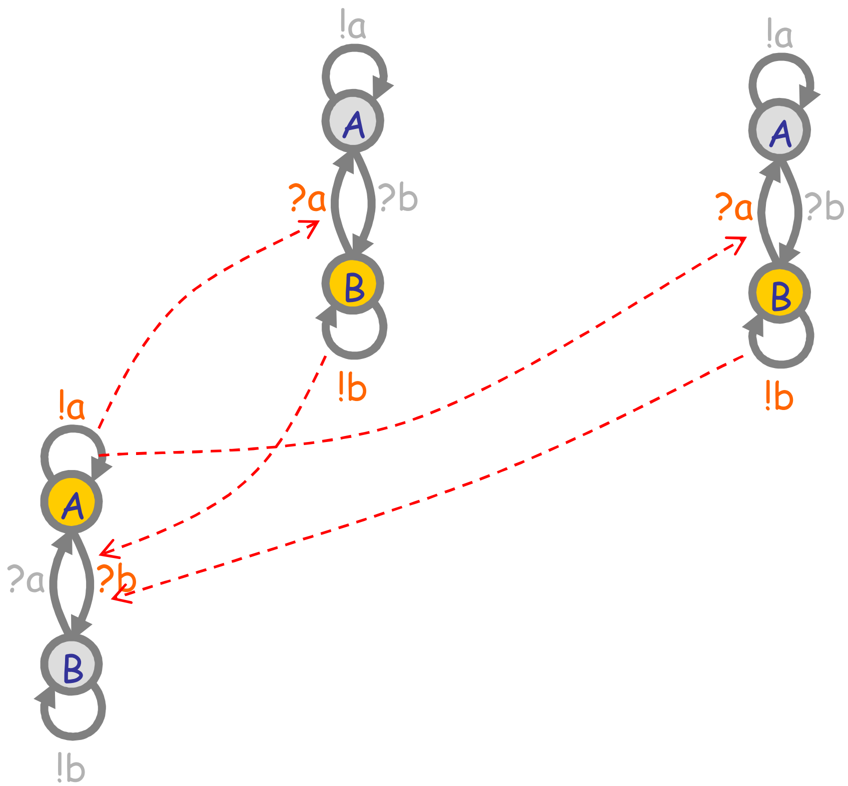


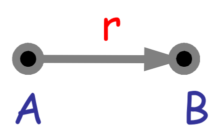


All-B stable population

Nondeterministic population behavior ("multistability")

CTMC Semantics



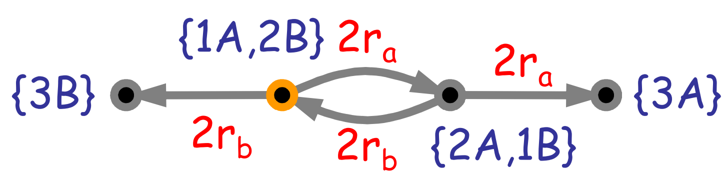


CTMC
(homogeneous) Continuous Time Markov Chain
- directed graph with no self loops
- nodes are system states
- arcs have transition rates

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

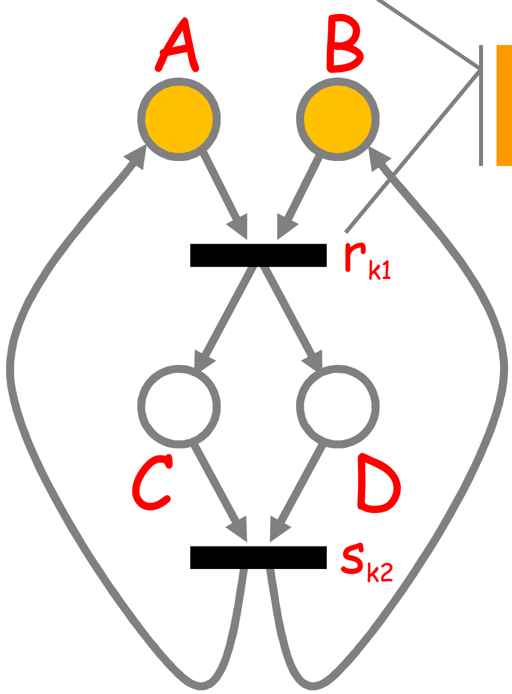
Reactions vs. Components

Says what "A" *does*.

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

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

Does A become C or D?

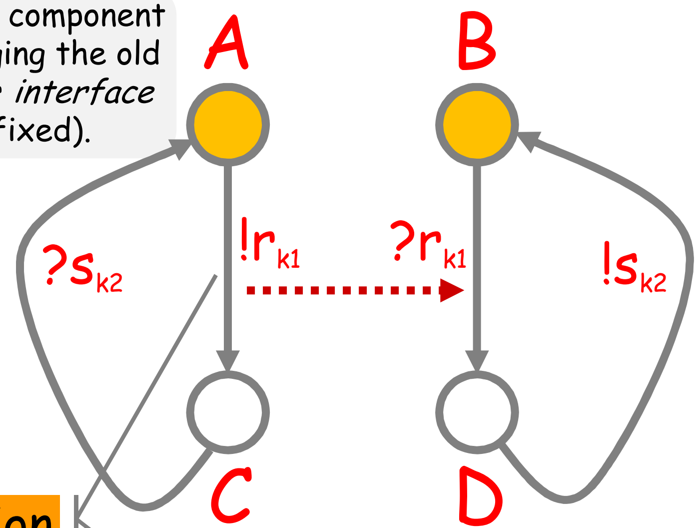


Reaction oriented

1 line per reaction

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

$A = !r_{k1}; C$

$C = ?s_{k2}; A$

$B = ?r_{k1}; D$

$D = !s_{k2}; B$

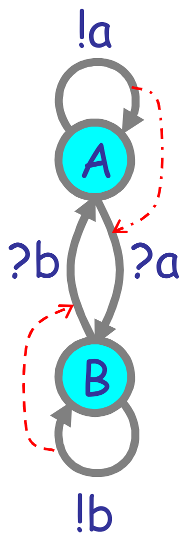
A becomes C not D!

The same "state space"

CTMC

Groupies and Celebrities

Groupies and Celebrities



Celebrity

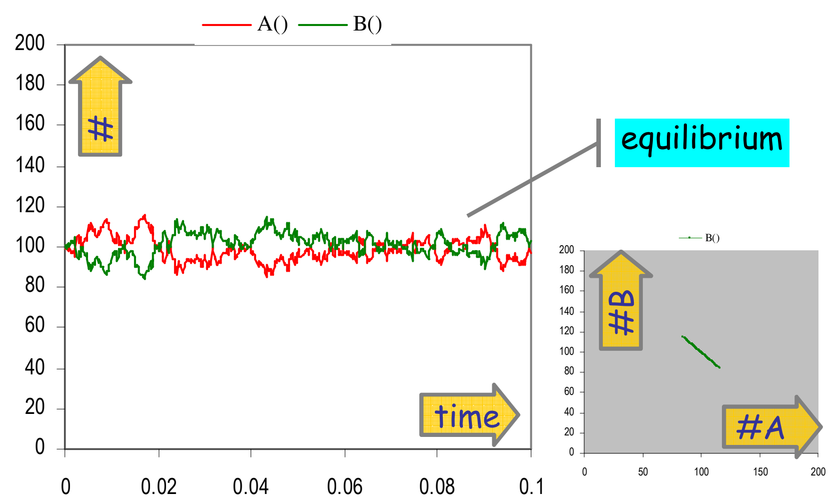
(does not want to be like somebody else)

```
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())
```

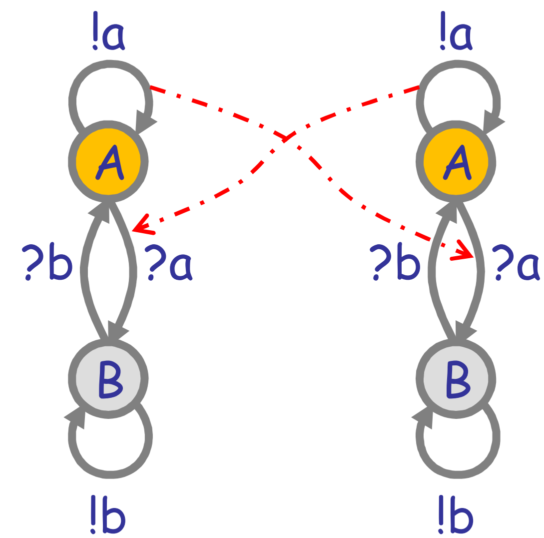
a@1.0

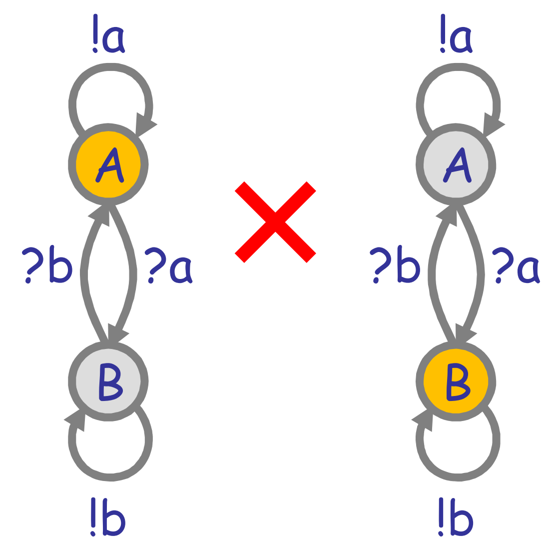
b@1.0

A stochastic collective of celebrities:

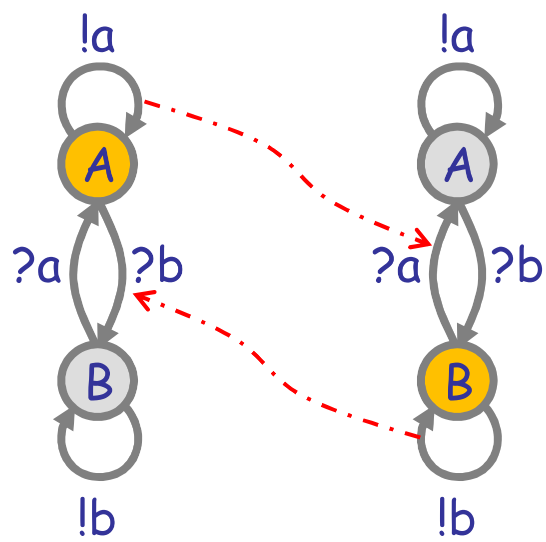


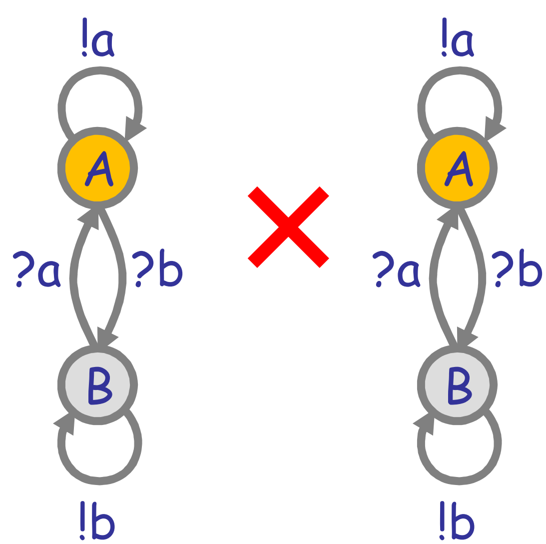
Stable because as soon as a A finds itself in the majority, it is more likely to find somebody in the same state, and hence change, so the majority is weakened.

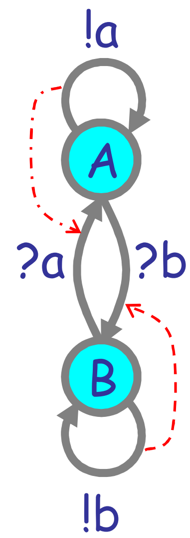




Groupies and Celebrities







Groupie

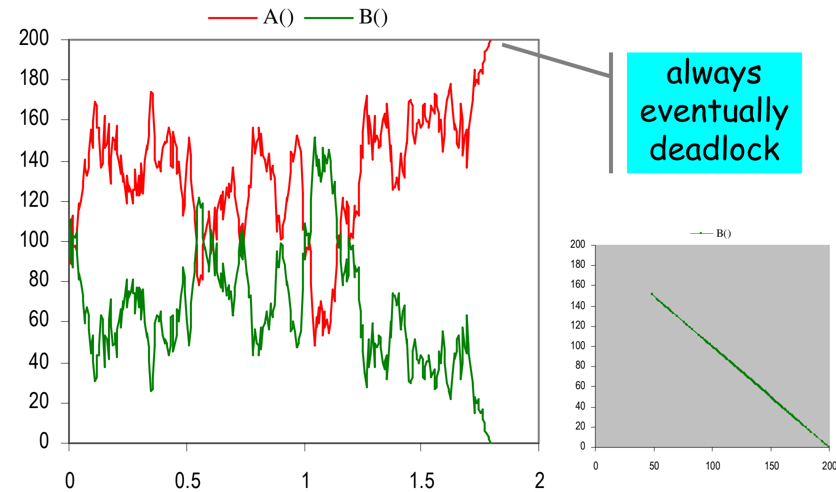
(wants to be like somebody different)

```
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:

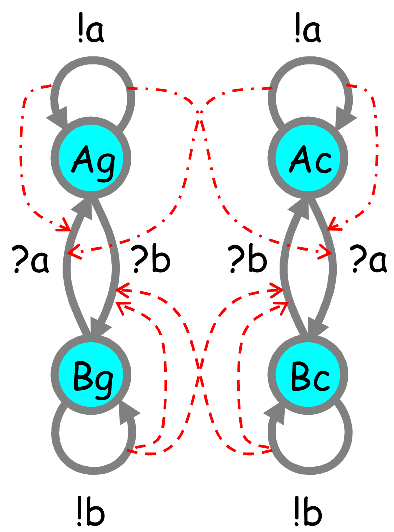


Unstable because within an A majority, an A has difficulty finding a B to emulate, but the few B's have plenty of A's to emulate, so the majority may switch to B. Leads to deadlock when everybody is in the same state and there is nobody different to emulate.

Both Together

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

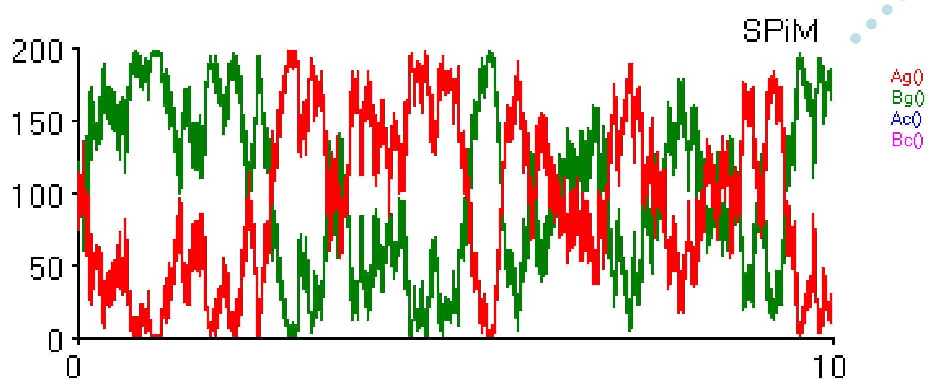
Many Groupies



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())
```

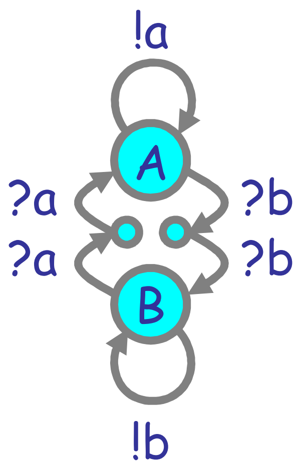
never deadlock

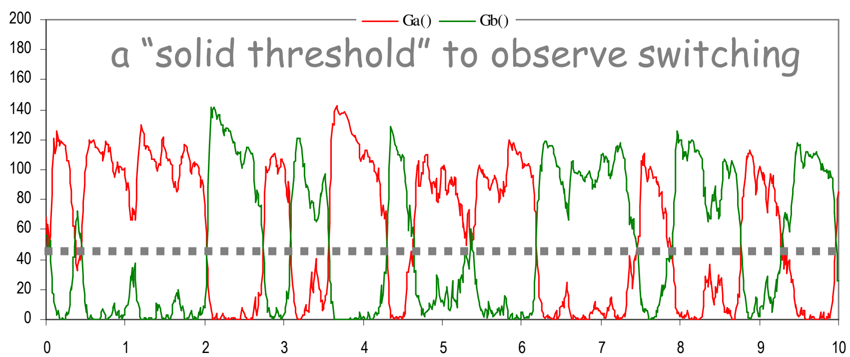


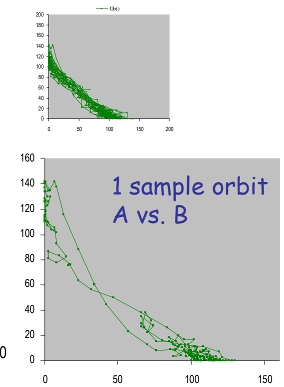
A tiny bit of "noise" can make a huge difference

Hysteric Groupies

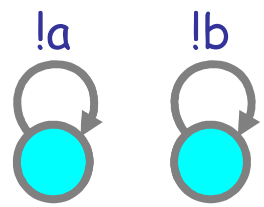
We can get more regular behavior from groupies if they "need more convincing", or "hysteresis" (history-dependence), to switch states.







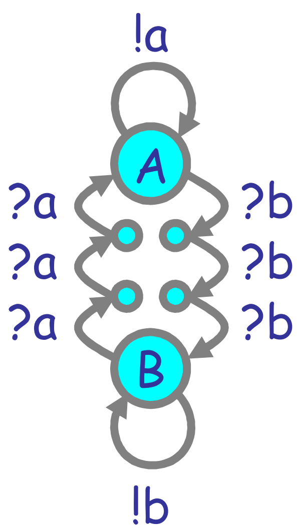

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

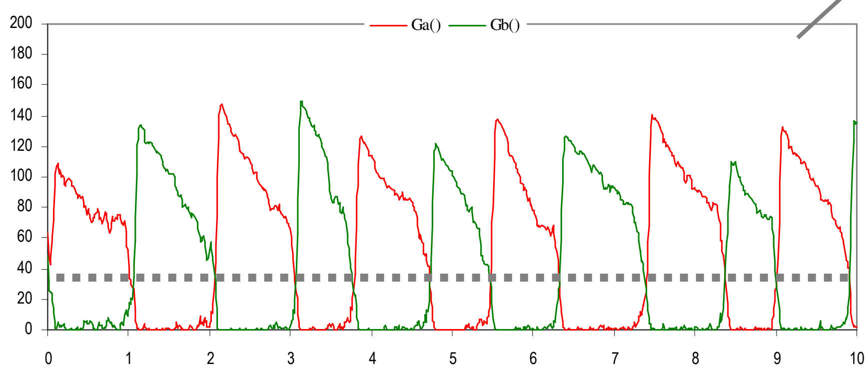


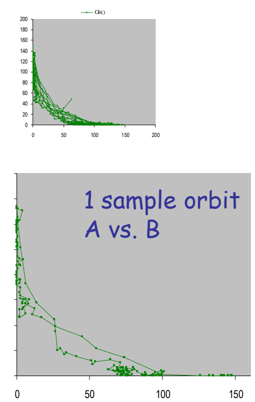
(With doping to break deadlocks)

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

"regular" oscillation



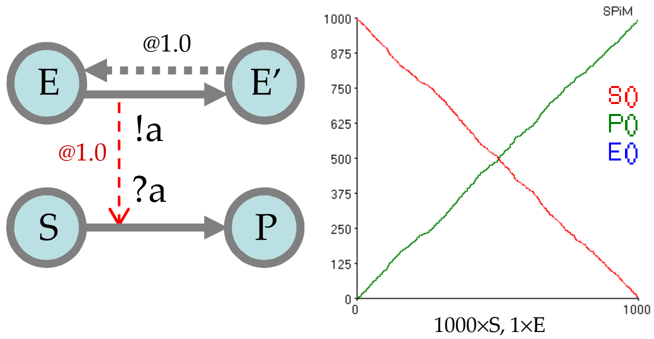




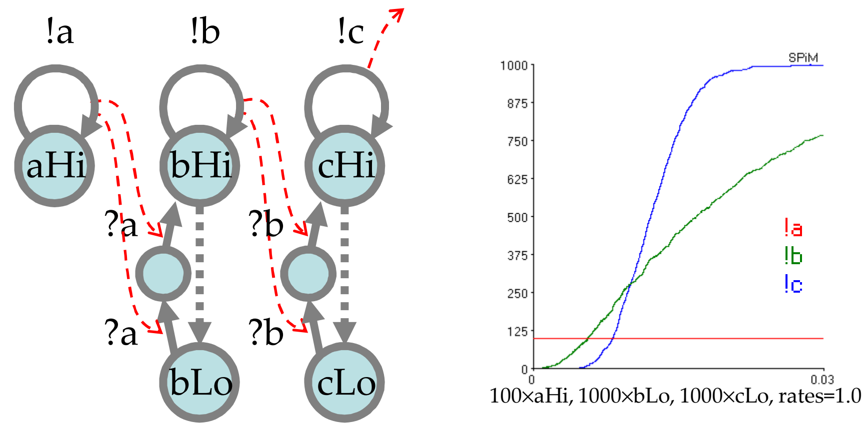

```
directive sample 10.0 1000
directive plot Ga(); Gb()

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())
```

Some Devices

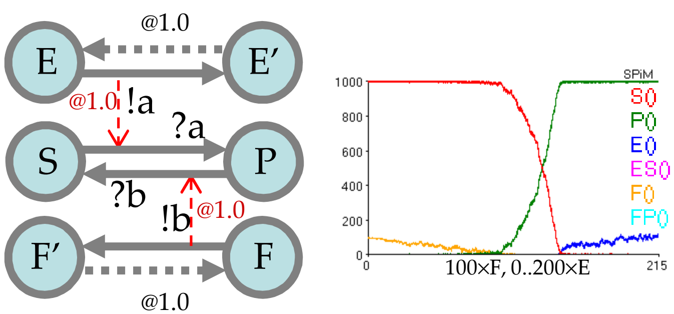
Linear Pump



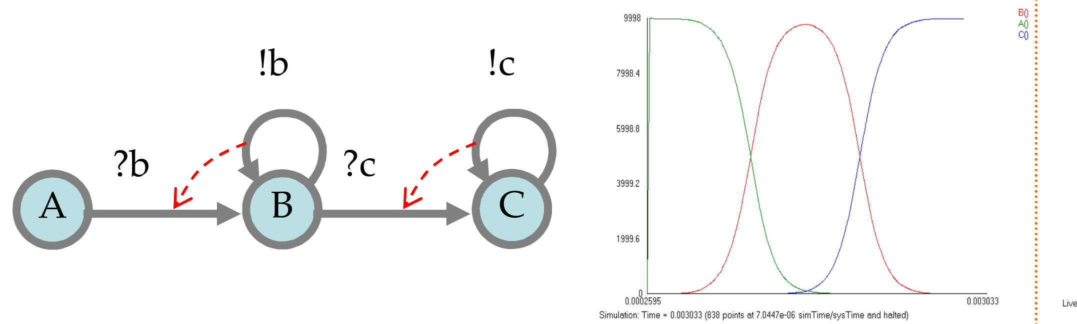
Cascade Amplifier



Ultrasensitive Switch

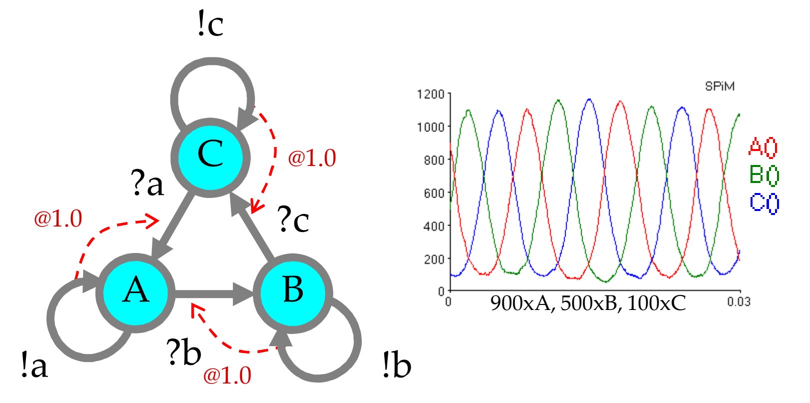


Symmetric Wave Generator

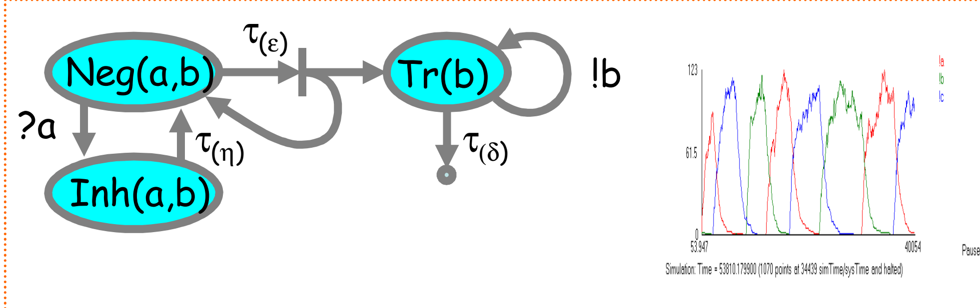


Some Devices

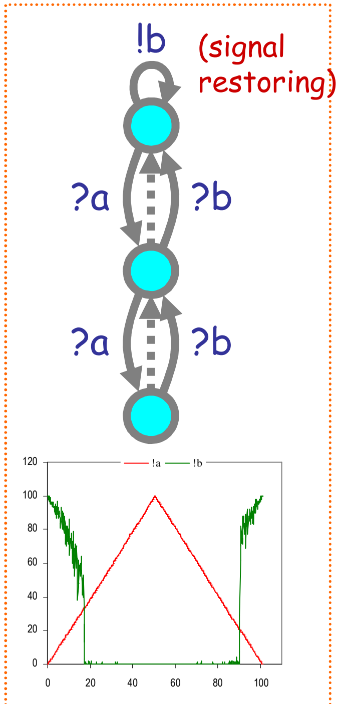
Oscillator



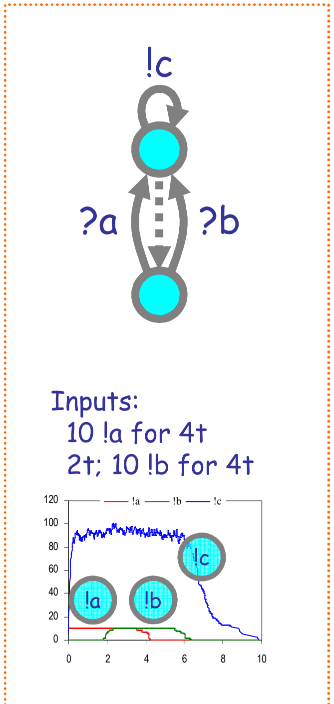
Repressilator (1 of 3 similar gates)



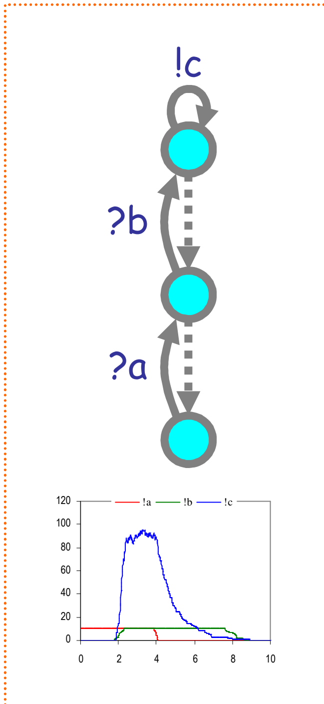
b = not a



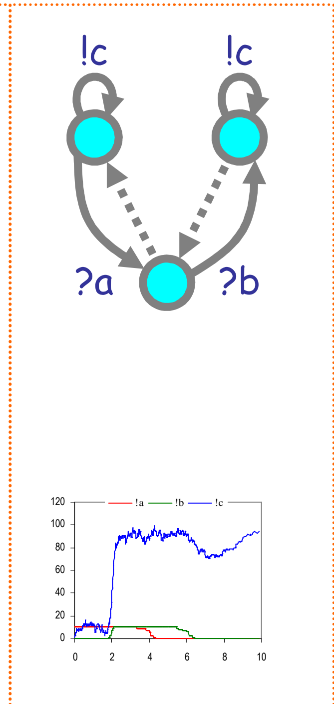
c = a or b



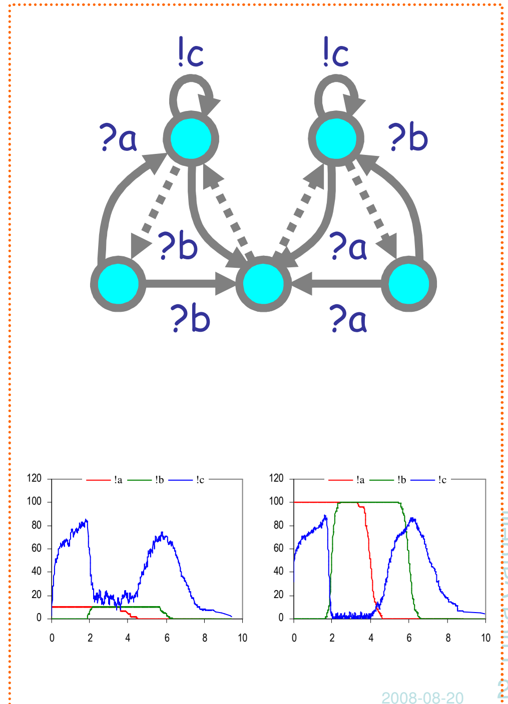
c = a and b



c = a imply b

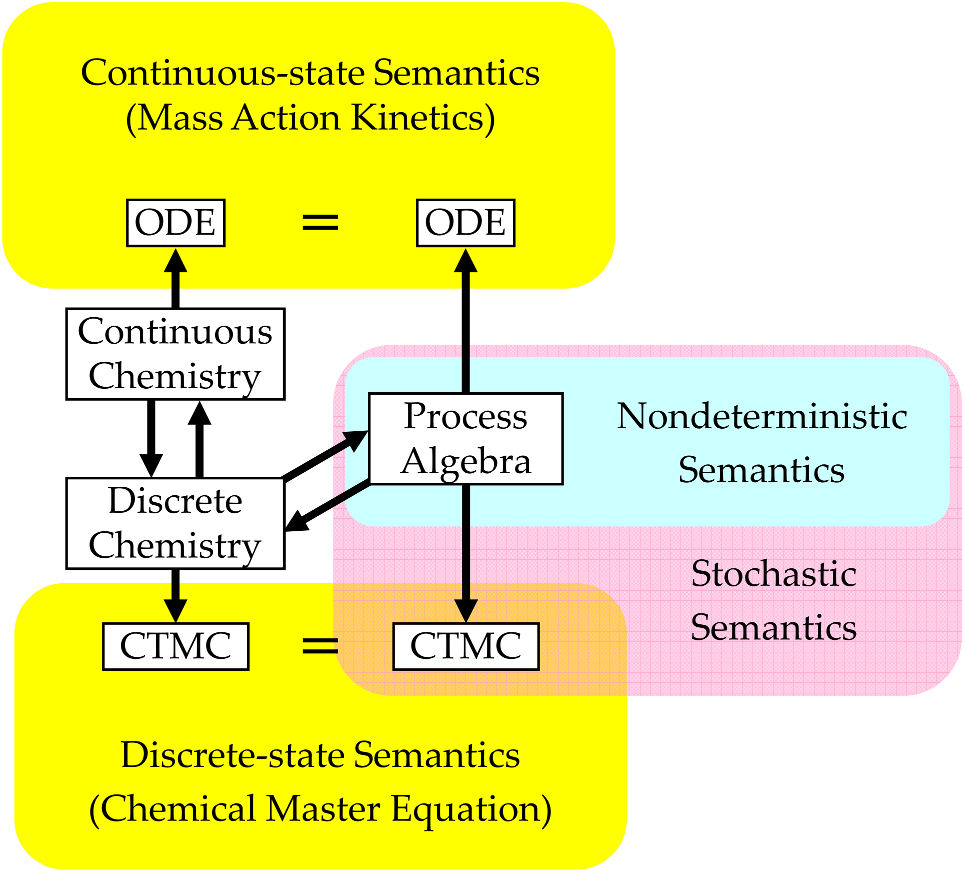


c = a xor b



Semantics of Collective Behavior

The Two Semantic Sides of Chemistry

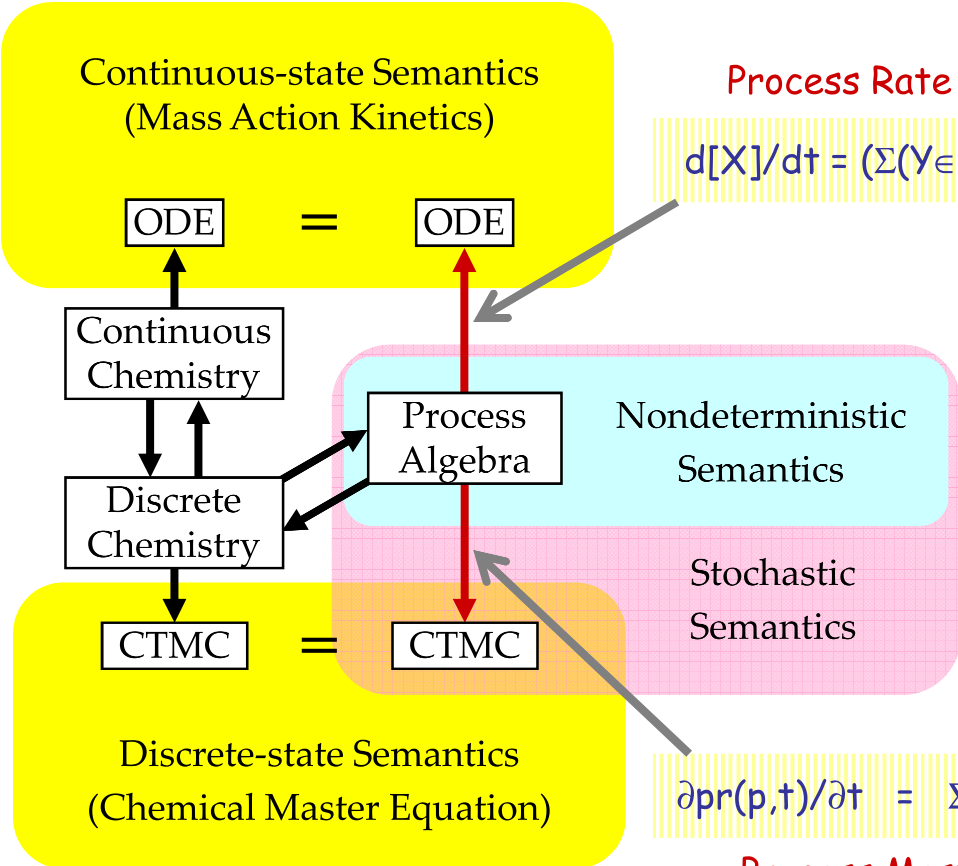


These diagrams commute via appropriate maps.

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

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

Quantitative Process Semantics



Process Rate Equation

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

Accretion — Depletion

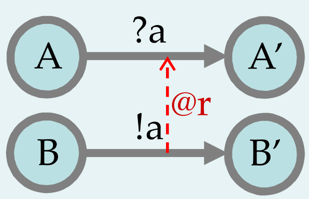
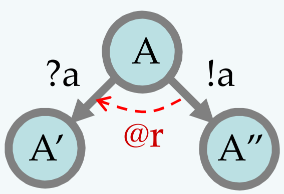
Defined over the syntax of processes

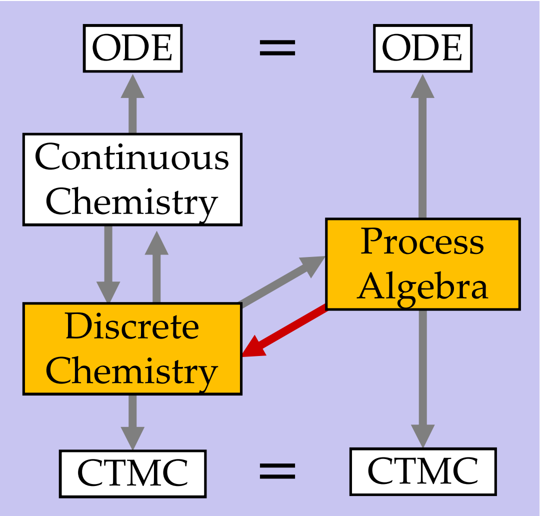
Interactions — Propensity

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

Process Master Equation

From Automata to Reactions (by example)

Interacting Automata →	Discrete Chemistry
initial states A | A | ... | A	initial quantities $\#A_0$
	$A \xrightarrow{r} A'$
	$A+B \xrightarrow{r} A'+B'$
	$A+A \xrightarrow{2r} A'+A''$



From Reactions 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)

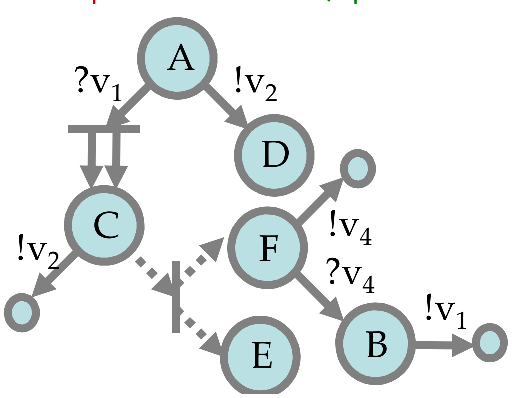
	$v_{1(k1)}$	$v_{2(k2)}$	$v_{3(k3)}$	$v_{4(k4/2)}$
A	?;(C|C)	?;D		
B	!;0			
C		!;0	τ;(E|F)	
D				
E				
F				?;B !;0

1: Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow_{k_i} P_i$
  add $\tau;P_i$ to $\langle X,v_i \rangle$.

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:

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

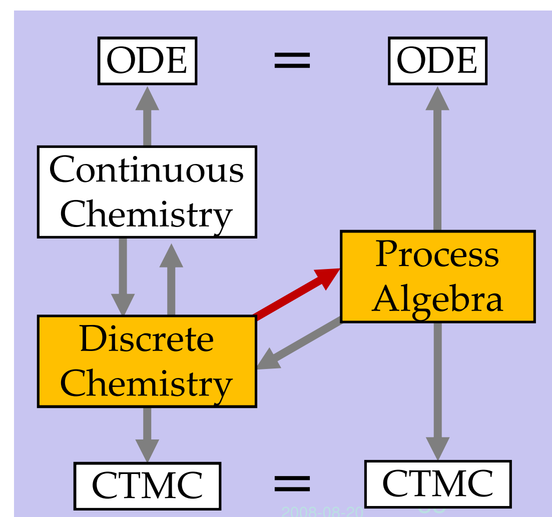
$B = !v_{1(k1)};0$

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

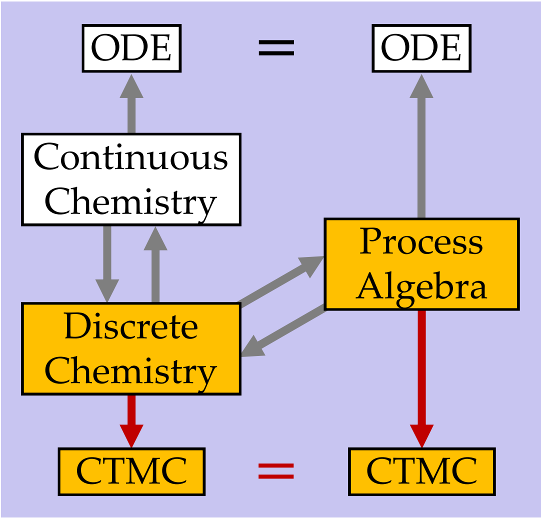
$D = 0$

$E = 0$

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



Discrete-State Semantics



Discrete Semantics of Reactions

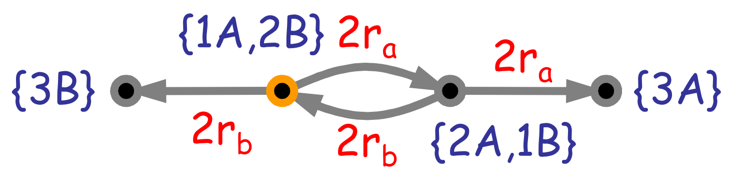
Syntax:

A+B →ʳ A+A

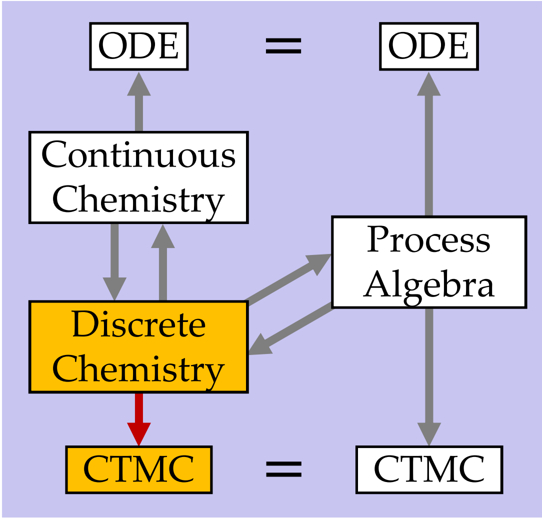
A+B →ʳ B+B

A+B+B

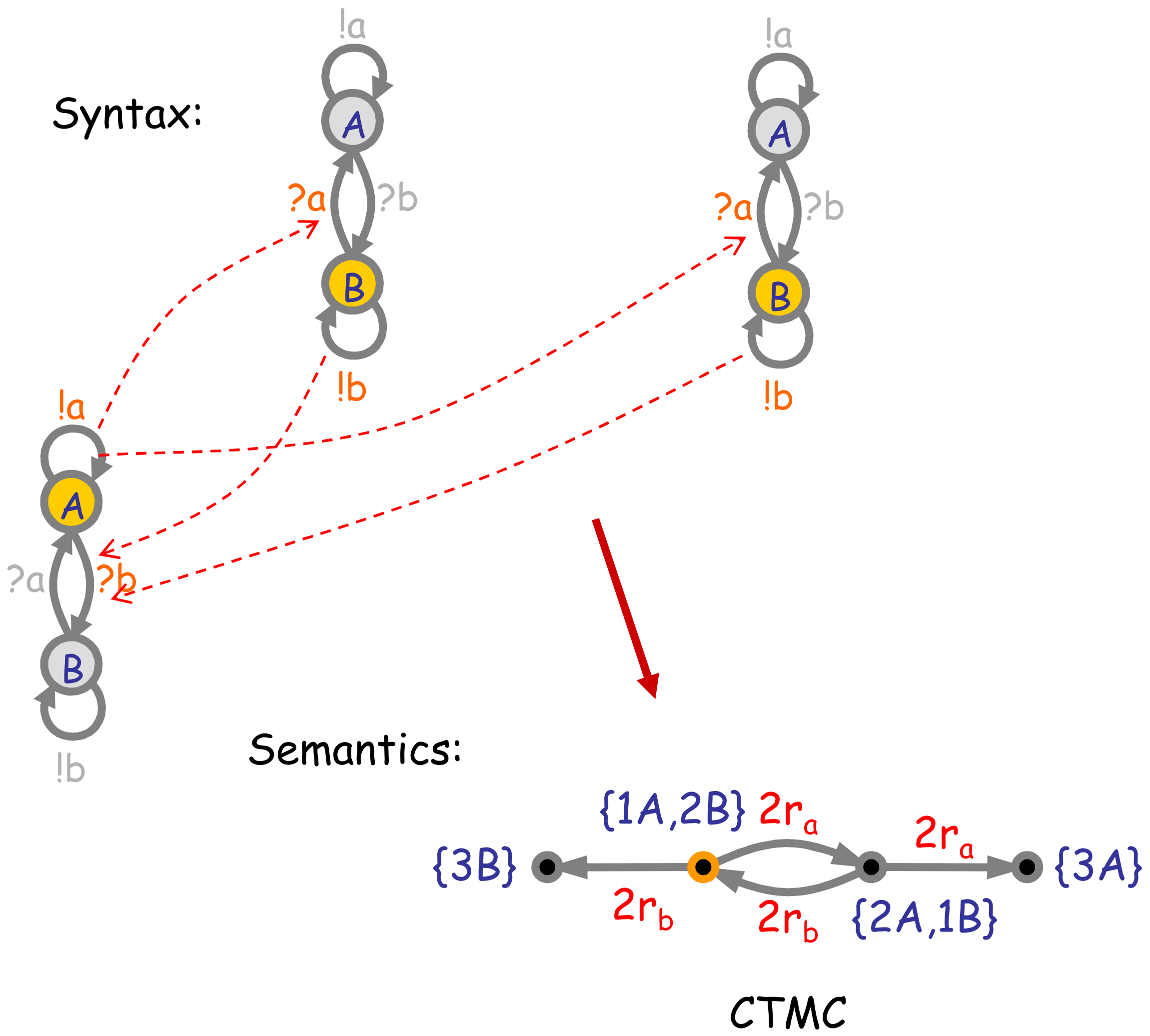
Semantics:

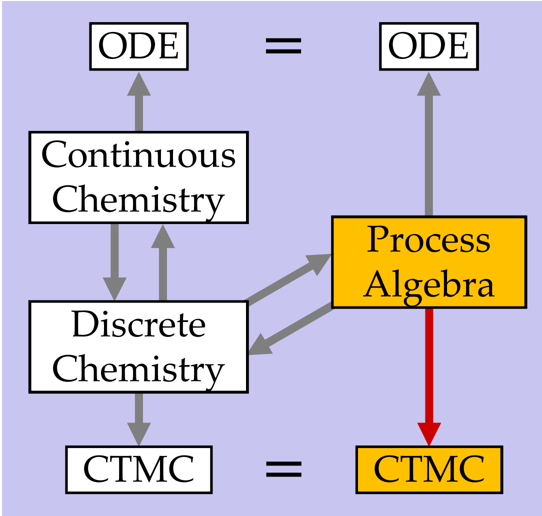


CTMC



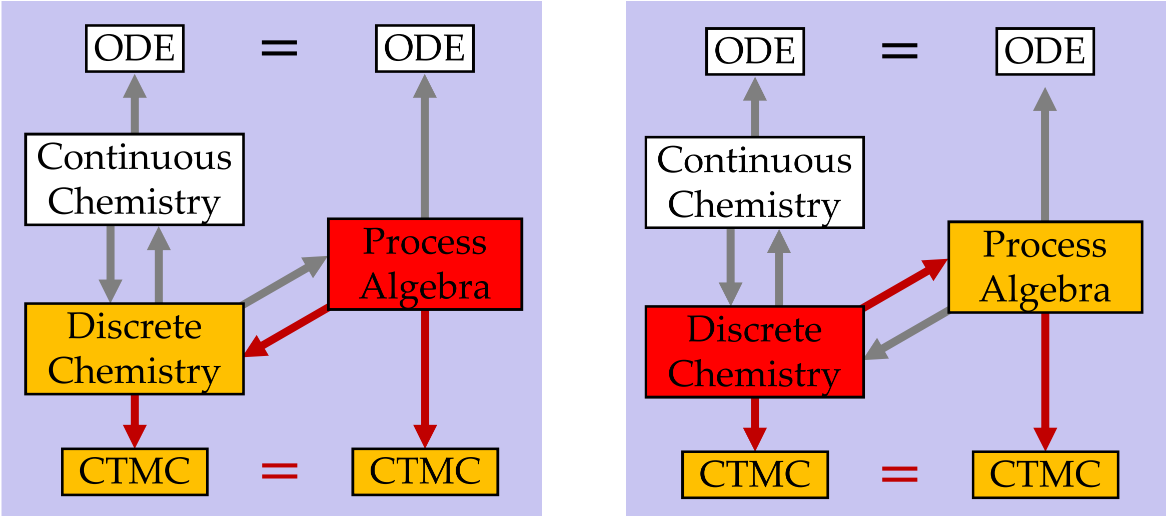
Discrete Semantics of Reagents





Discrete State Equivalence

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



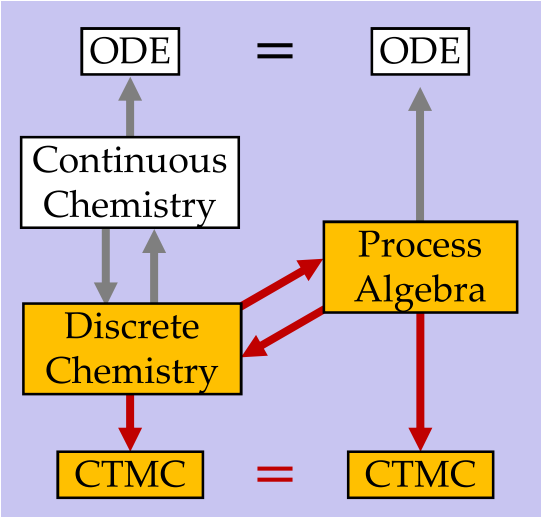
- For each E there is an E' ≈ E that is detangled (E' = Pi(Ch(E)))
- For each E in automata form there is an an E' ≈ E that is detangled and in automata form (E' = Detangle(E)).

Interacting Automata = Discrete Chemistry

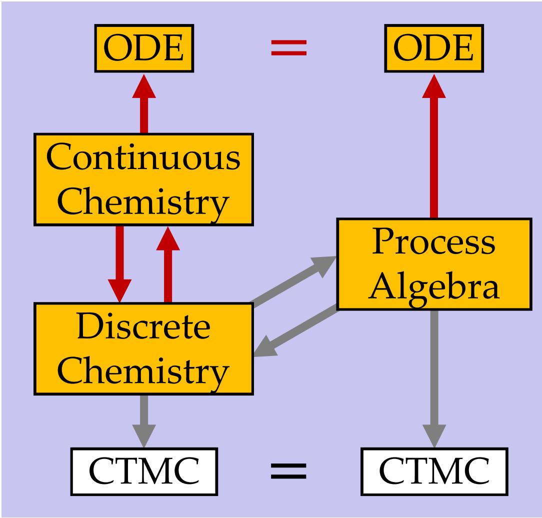
This is enough to establish that the process algebra is really faithful to the chemistry.

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).

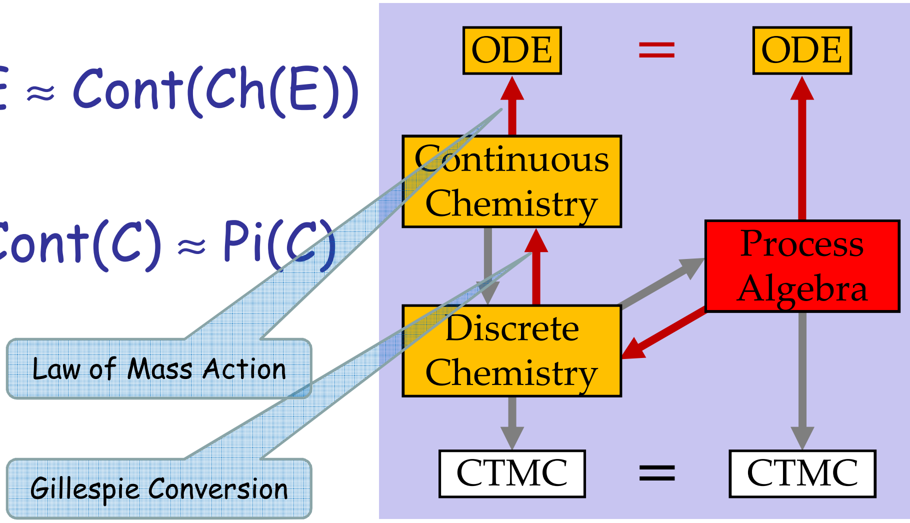


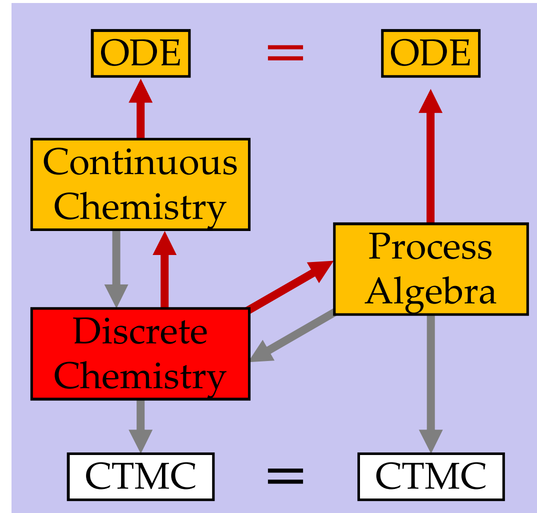
Continuous-State Semantics
(summary)



Continuous State Equivalence

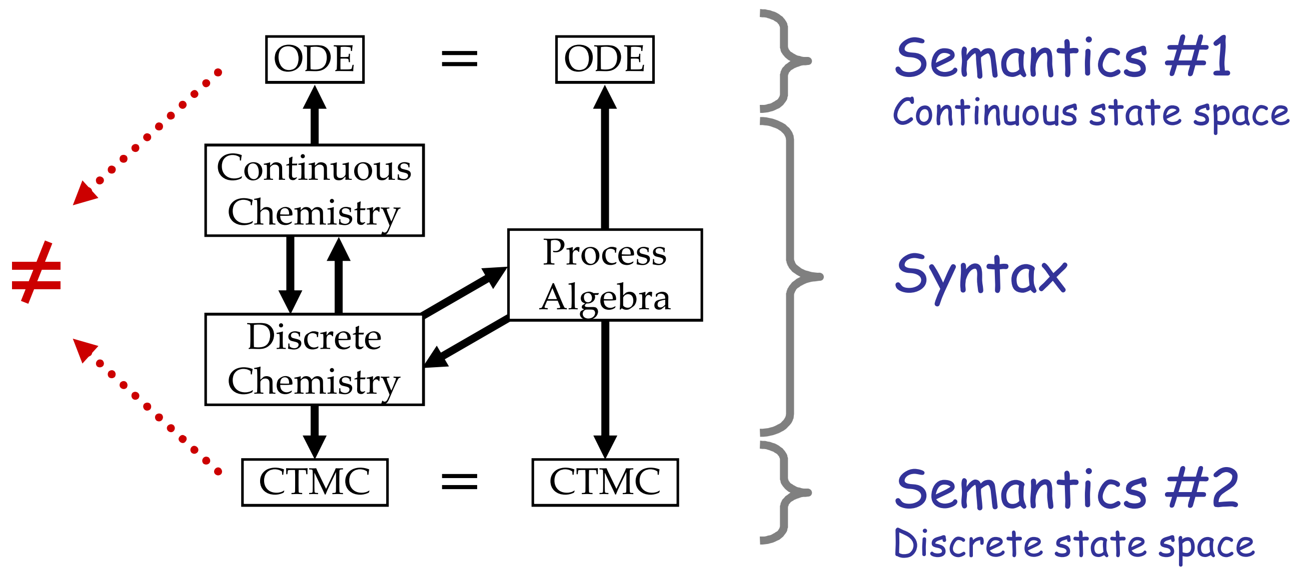
- Def: ≈ is equivalence of polynomials over the field of reals.
- Thm: E ≈ Cont(Ch(E))
- Thm: Cont(C) ≈ Pi(C)





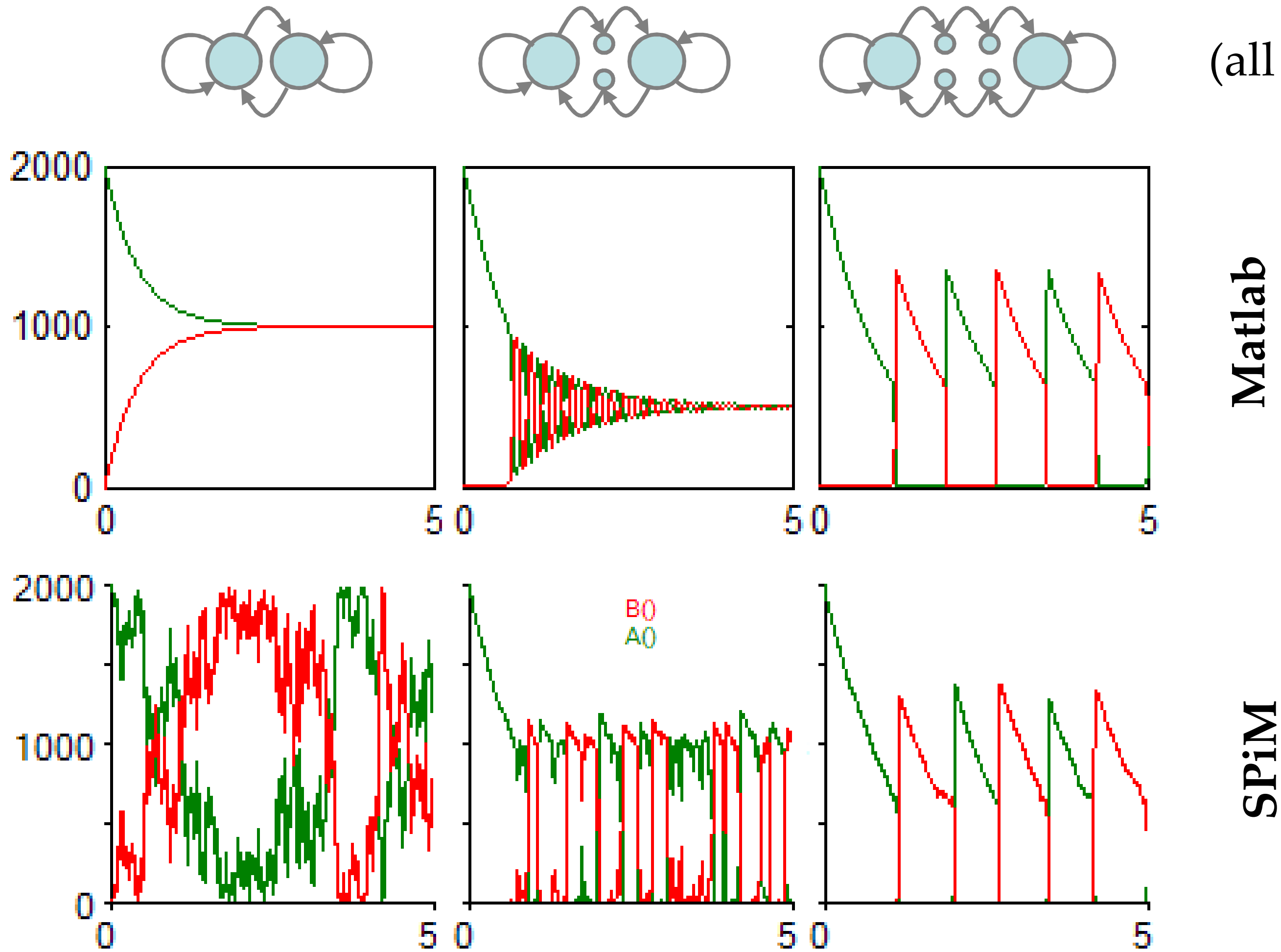
- For each E there is an E′ ≈ E that is detangled (E′ = Pi(Ch(E)))
- For each E in automata form there is an an E′ ≈ E that is detangled and in automata form (E′ = Detangle(E)).

GMA ≠ CME


ODE
=
ODE
Continuous Chemistry
Process Algebra
Discrete Chemistry
CTMC
=
CTMC
≠
Semantics #1
Continuous state space
Syntax
Semantics #2
Discrete state space

Continuous vs. Discrete Groupies

(all with doping)



$2000{\times}A\ ,\ 0{\times}B\ ,\ 1{\times}A_d\ ,\ 1{\times}B_d\ ,\ r = 1.0$

```
directive sample 5.0 1000
directive plot B(); A()

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()

let Ad() = !a; Ad()
and Bd() = !b; Bd()

run 2000 of A()
run 1 of (Ad() | Bd())
```

```
directive sample 5.0 1000
directive plot B(); A()

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

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

let Ad() = !a; Ad()
and Bd() = !b; Bd()

run 2000 of A()
run 1 of (Ad() | Bd())
```

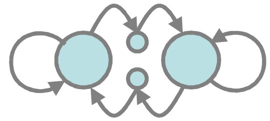
```
directive sample 5.0 1000
directive plot B(); A()

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

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

let Ad() = !a; Ad()
and Bd() = !b; Bd()

run 2000 of A()
run 1 of (Ad() | Bd())
```

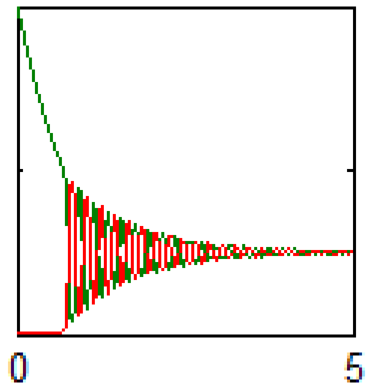
```
Groupe ODEs - Groupies.mat

[0:0.001:5.0] r=1.0 k=1.0
A dx1/dt = -(x1-x2),   2000.0
B dx2/dt = (x1-x2),   0.0
```

```
Groupe ODEs - Groupies Hysteric 1.mat

[0:0.001:5.0] r=1.0 k=1.0
A dx1/dt=x1*x4-x3*x1-x1+x4, 2000.0
A' dx2/dt=x3*x1-x3*x2+x1-x2, 0.0
B dx3/dt=x3*x2-x1*x3-x3+x2, 0.0
B' dx4/dt=x1*x3-x1*x4+x3-x4, 0.0
```

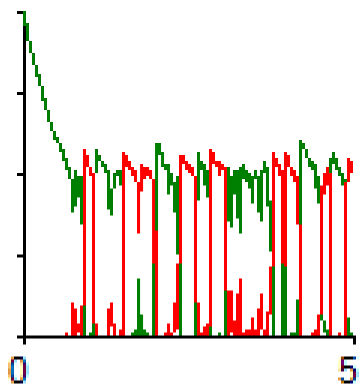
```
Groupe ODEs - Groupies Hysteric 2.mat

[0:0.001:5.0] r=1.0 k=1.0
A  dx1/dt=x1*x6-x3*x1-x1+x6, 2000.0
A'  dx2/dt=x3*x1-x3*x2+x1-x2, 0.0
A" dx5/dt=x3*x2-x3*x5+x2-x5, 0.0
B  dx3/dt=x3*x5-x1*x3-x3+x5, 0.0
B'  dx4/dt=x1*x3-x1*x4+x3-x4, 0.0
B"  dx6/dt=x1*x4-x1*x6+x4-x6, 0.0
```

Scientific Predictions





After a while, all 4 states are almost equally occupied.

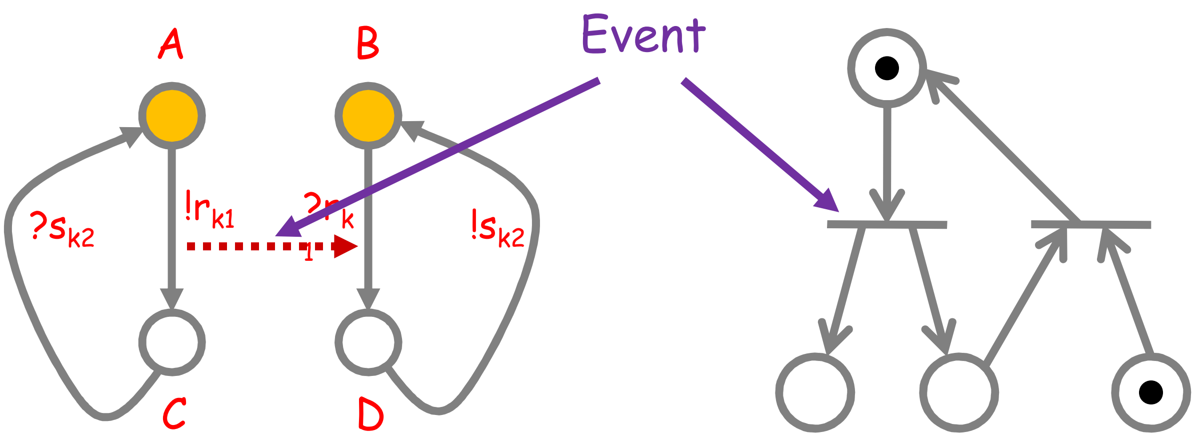


The 4 states are almost never equally occupied.

Discrete Analysis Techniques

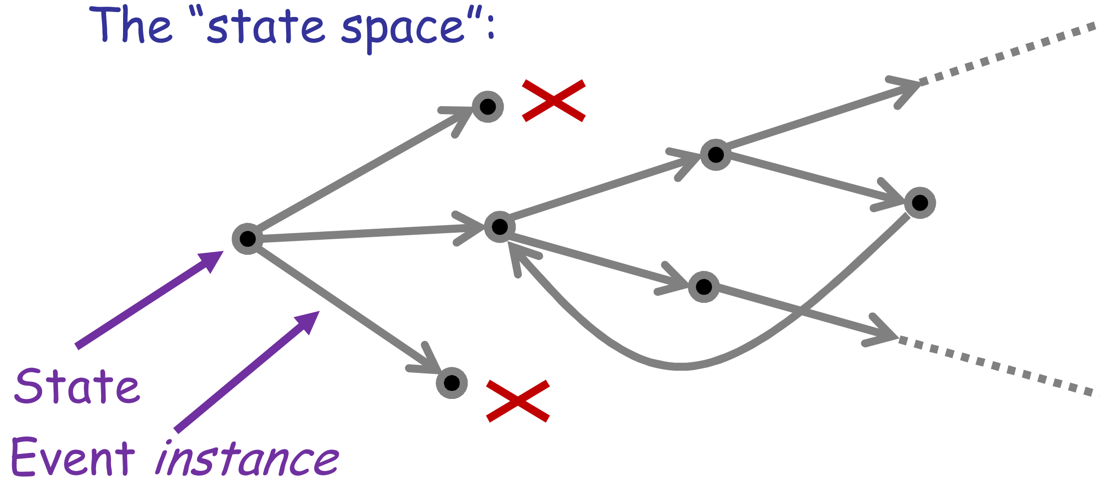
The Program vs. the State Space

The "program":



Finite

The "state space":



Potentially infinite

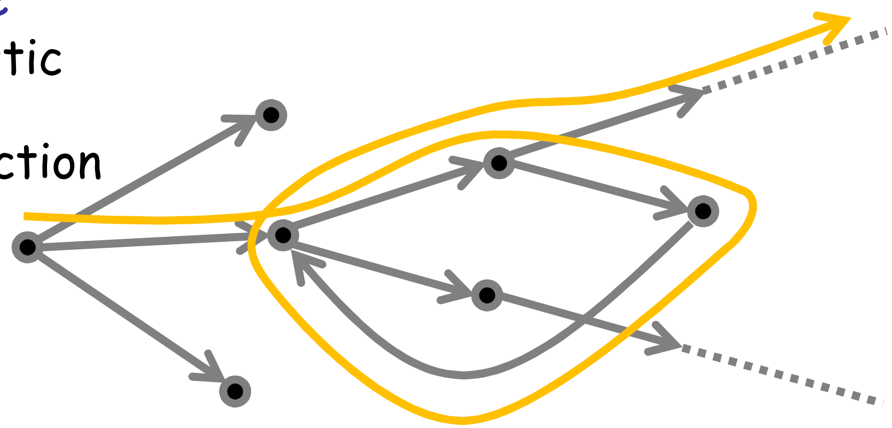
Simulation

- Run "the program" through a walk in states space.

- Basic stochastic algorithm: Gillespie
  - Exact (i.e. based on physics) stochastic simulation of chemical kinetics.
  - Can compute concentrations and reaction times for biochemical networks.

- Stochastic Process Algebras
  - Now many [BioSPi, SPiM, BioPEPA, BetaBinders, …]

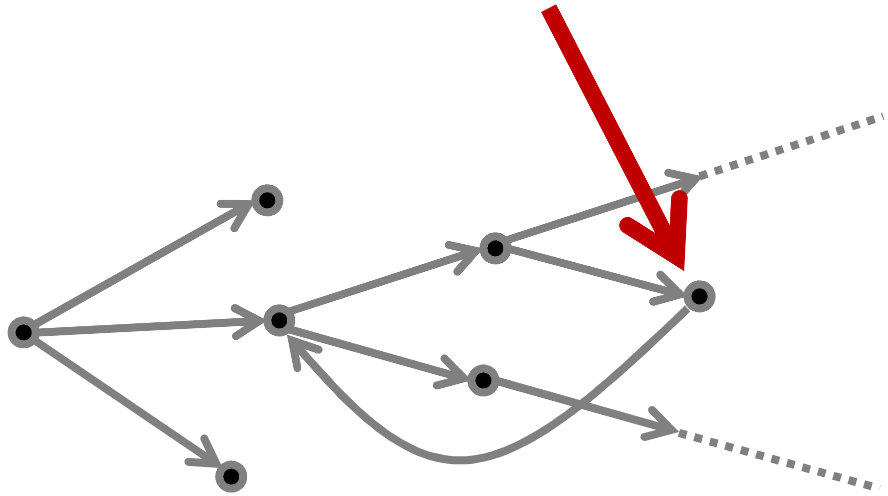
- Hybrid approaches
  - Continuous + discrete/stochastic switching



Control Flow Analysis

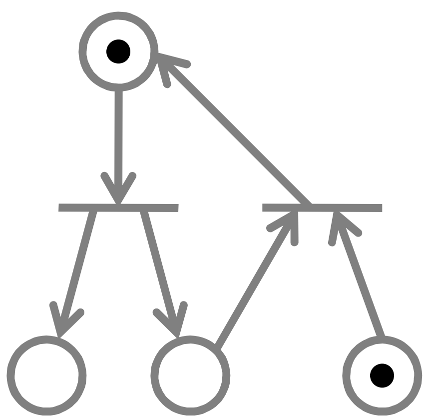
- Who may call who?
  - Overapproximation of behavior used to answer questions about what “cannot happen”.

What event may (or may not) have been involved in reaching this state?

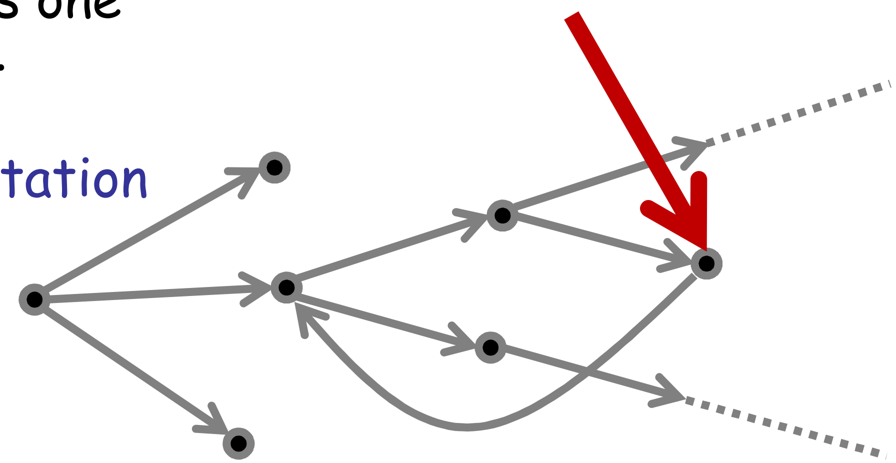


Causality Analysis

- What event caused what other event or state to happen?
  - E.g.: if in all possible executions one event always precedes another.
- Need a different level of representation (the "event space")
  - Petri Nets
  - Event Structures

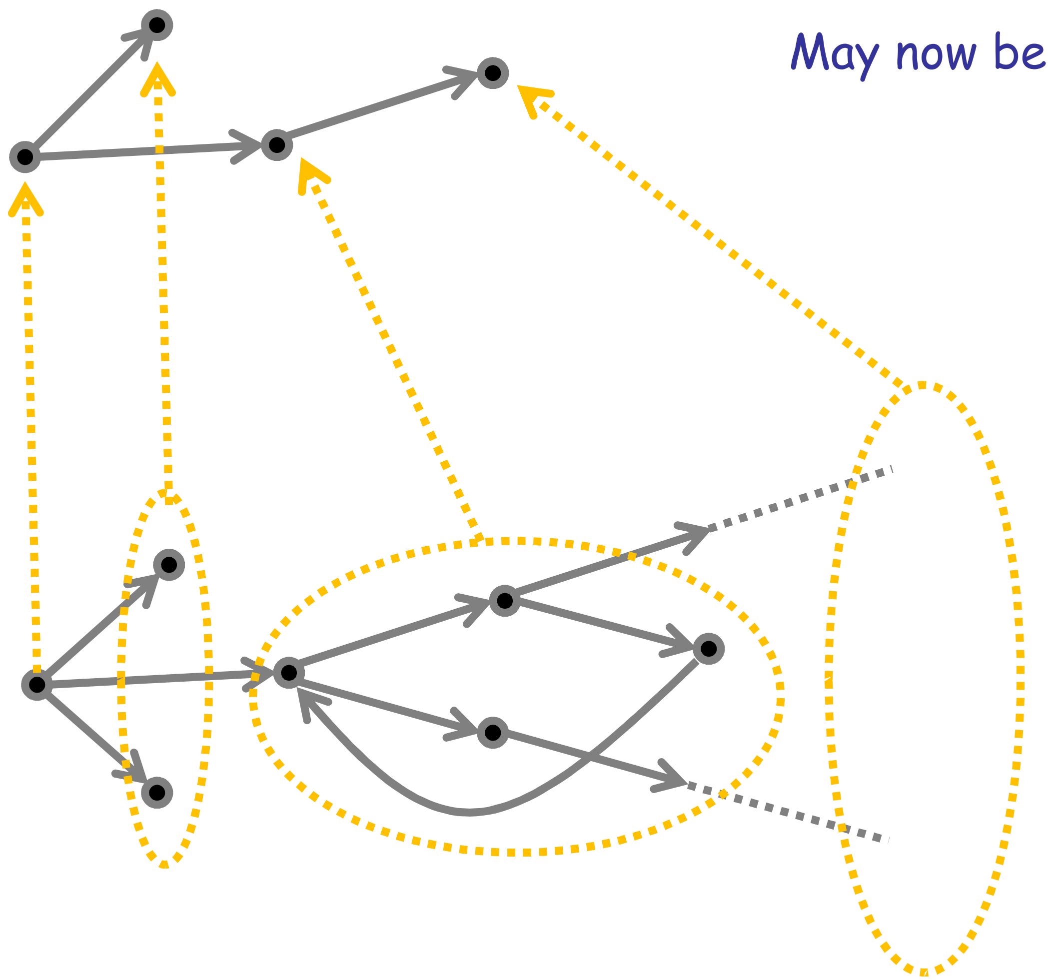


What event "caused" this state?



Abstract Interpretation

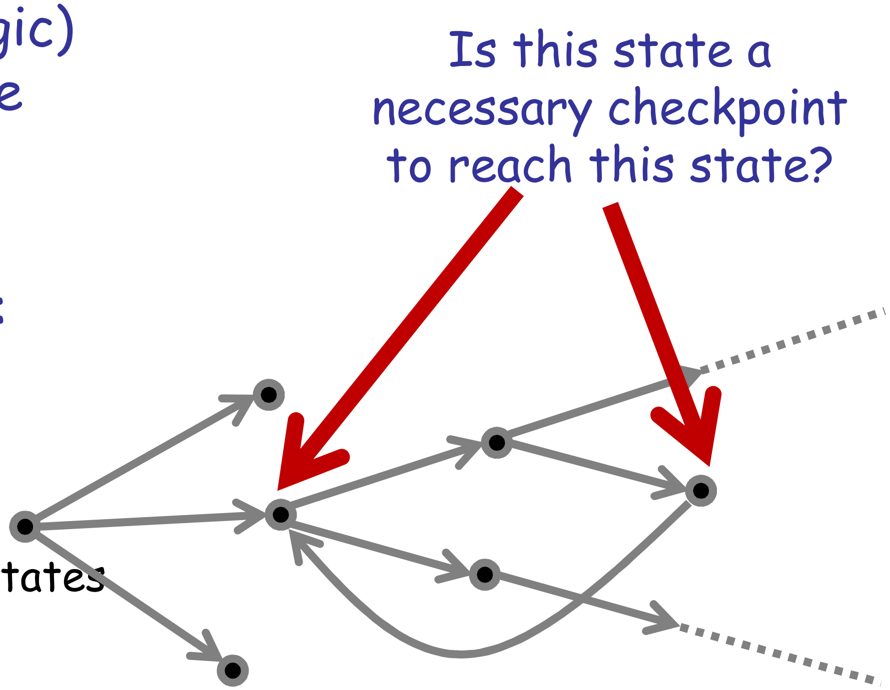
- Precisely relating abstract views to more concrete views of the system



Modelchecking

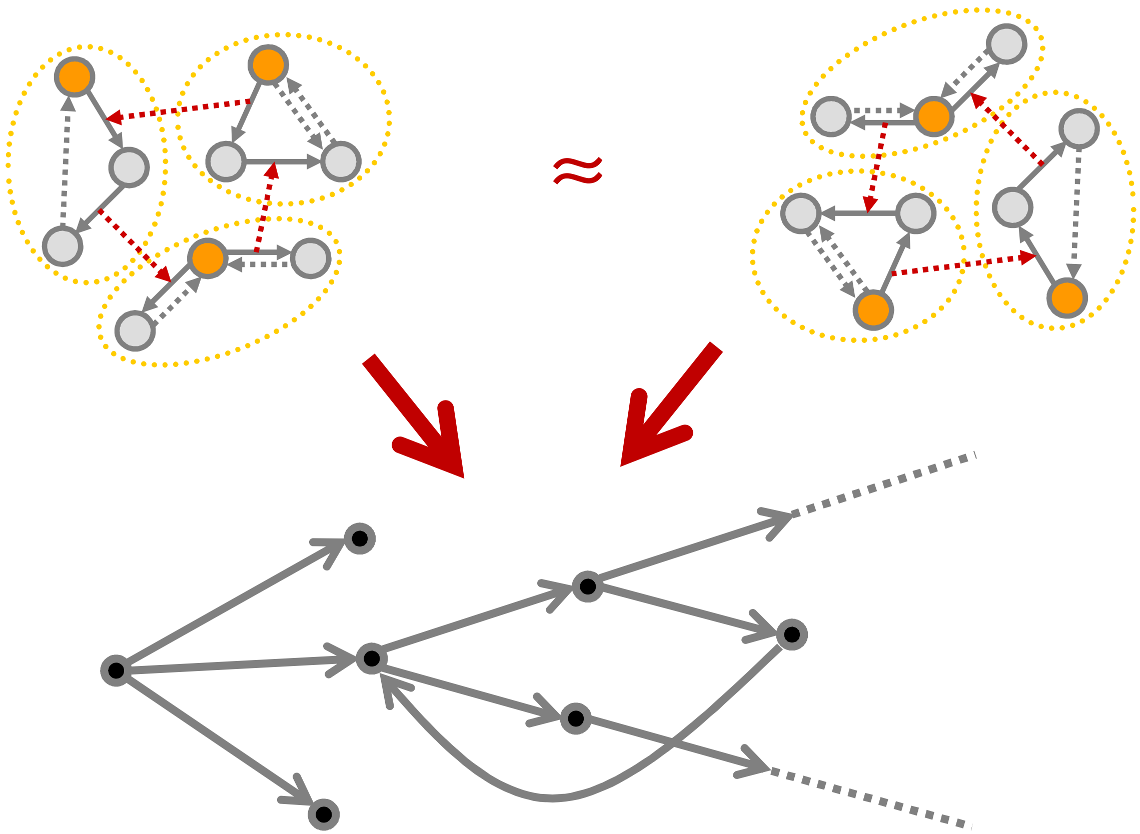
- Asking questions (in Temporal Logic) about structure of a (finite) state space.

- Various flavors of modelchecking:
  - Temporal
    - About paths through state space
  - Quantitative
    - About quantitative measures of states
  - Probabilistyc/Stochastic
    - About probabilities of reaching states.



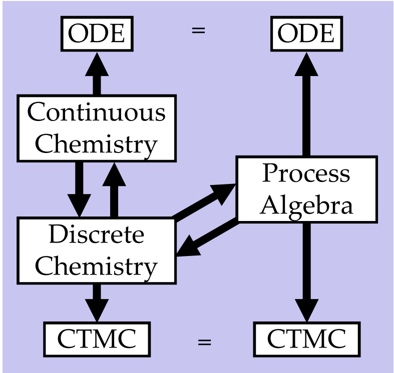
Bisimulation

- Are two programs generating the same state space?
  - E.g.: Is a compact description of a system equivalent to a more detailed one in all possible environments?



Conclusions

Conclusions



- Process Algebra
  - An extension of automata theory to populations of interacting automata
  - Modeling the behavior of individuals in an arbitrary environment
  - Compositionality (combining models by juxtaposition)
- Connections between modeling approaches
  - Connecting the discrete/concurrent/stochastic/molecular approach
  - to the continuous/sequential/deterministic/population approach
- Connecting syntax with semantics
  - Syntax = model presentation (equations/programs/diagrams/blobs etc.)
  - Semantics = state space (generated by the syntax)
- Ultimately, connections between analysis techniques
  - We need (and sometimes have) good semantic techniques to analyze state spaces (e.g. calculus, but also increasingly modelchecking)
  - But we need equally good syntactic techniques to structure complex models (e.g. compositionality) and analyze them (e.g. process algebra)