Biochemical Systems as Reactive Systems

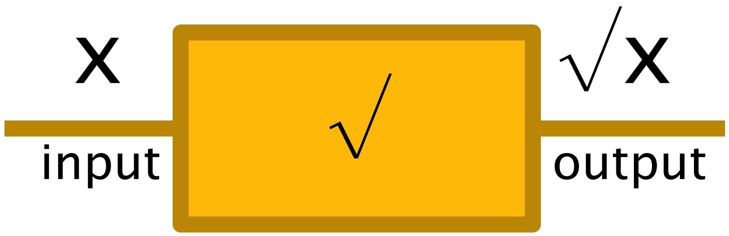
Luca Cardelli
Microsoft Research

Cambridge 2011-02-22
http://lucacardelli.name

Processes and Functions

Functions

$$f(x) = \sqrt{x}$$



x

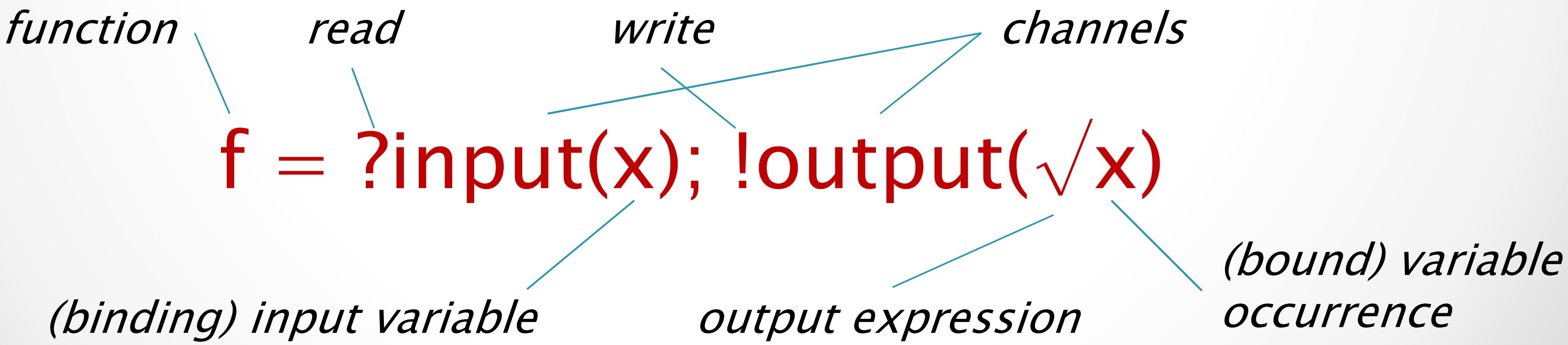
$\sqrt{}$

$\sqrt{x}$

input

output

“read input into x; then write √x to output”



function

read

write

channels

$$f = ?\mathrm{input}(x);\ !\mathrm{output}(\sqrt{x})$$

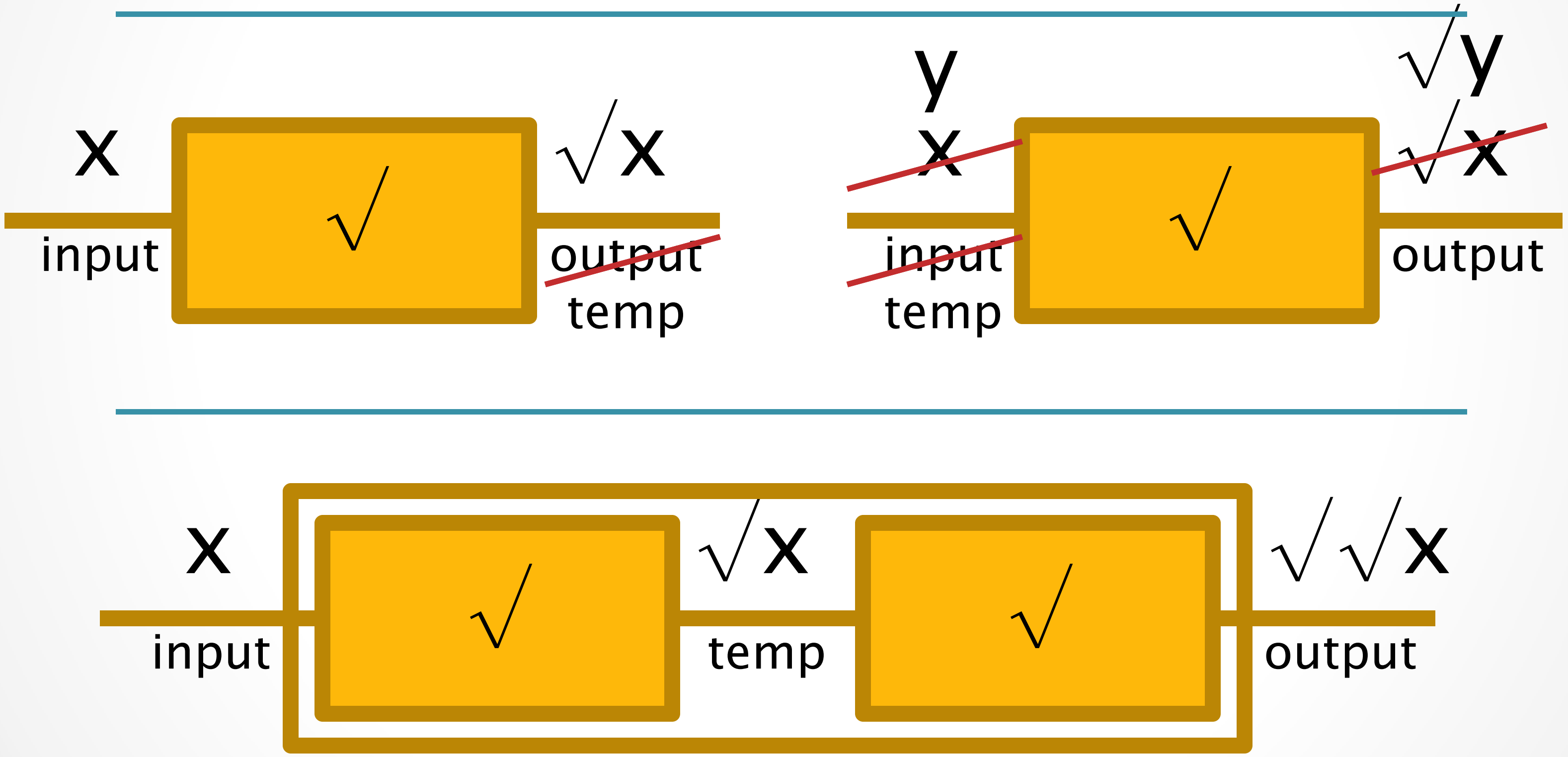
(binding) input variable

output expression

(bound) variable occurrence

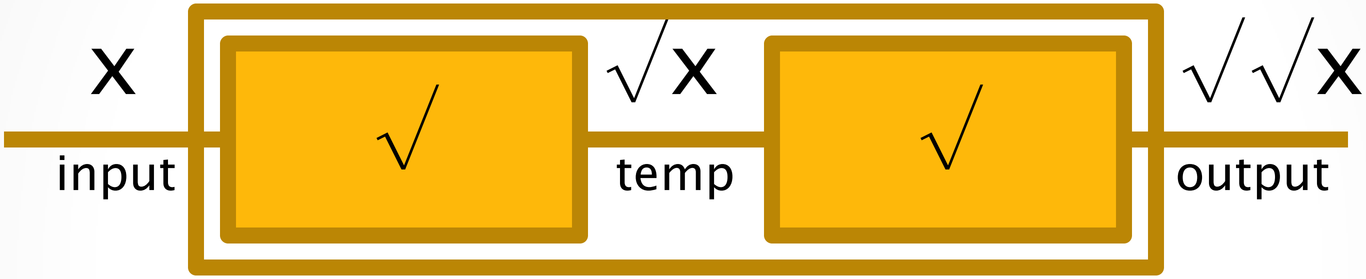
Composing Functions

$$g(x) = (f \circ f)(x) \quad (= f(f(x)))$$



Composing Functions

$$g(x) = (f \circ f)(x)$$



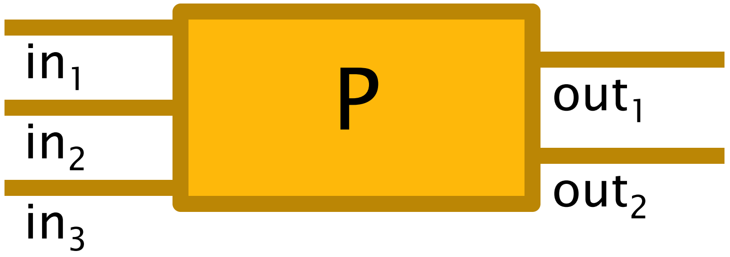
“create a *new* channel and use it to compose two copies of f”

channel creation (restriction/hiding/boxing)

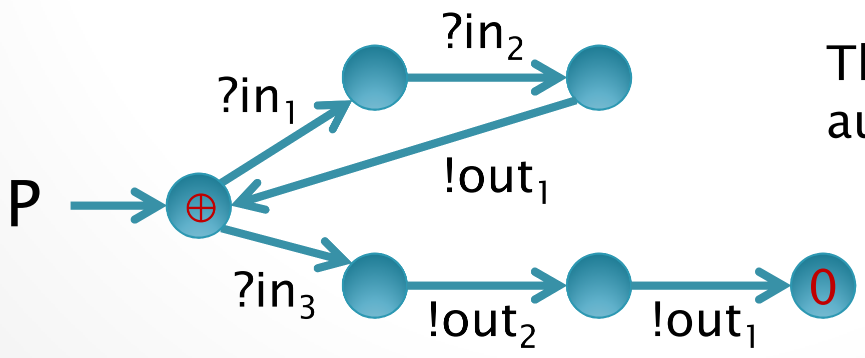
(parallel/process) composition

g = (ν temp)
?input(x); !temp(√x) |
?temp(y); !output(√y)

Many inputs and outputs



$$\begin{aligned} P = \quad & ?in_1(x);\ ?in_2(y);\ !out_1(x+y);\ P \\ \oplus\ & ?in_3(z);\ !out_2(\surd z);\ !out_1(2z);\ 0 \end{aligned}$$



The ‘skeleton’ automaton

That’s π–calculus

- To compose processes P we need:

Composition:	P | P	(with identity elem. 0)
Channel cration:	(ν x) P	(with x bound in P)
Recursion:	*P	(equal to P | *P)

- To execute actions we need:

Channel reading:	?c(x); P	(with x bound in P)
Channel writing:	!c(M); P	(with message M)
Choice:	P ⊕ P	(with identity elem. 0)

- … and channels can be sent as messages!

Generalizing Functions and Automata

- Unlike functions…
  - Processes have multiple, explicitly named, input and output channels.
  - Processes can run in *parallel* , can *deadlock* on their inputs, and can be *nondeterministic* in their outputs.

- Unlike automata (FSA)…
  - Processes can transmit data (not just change state).
  - While automata ‘talk’ to input strings, processes ‘talk’ to other processes: processes are communicating automata.
  - Processes are not “finite state”; they can express unbounded computation in time (divergence) and space (proliferation).

Algebraic Properties

- Functions have one binder and one rule:
  - Function application:

    If $f(x) =_{def} M\{x\}$ then $f(a) = M\{a/x\}$

- Processes have two binders and two rules:
  - Communication (input '?' binder)

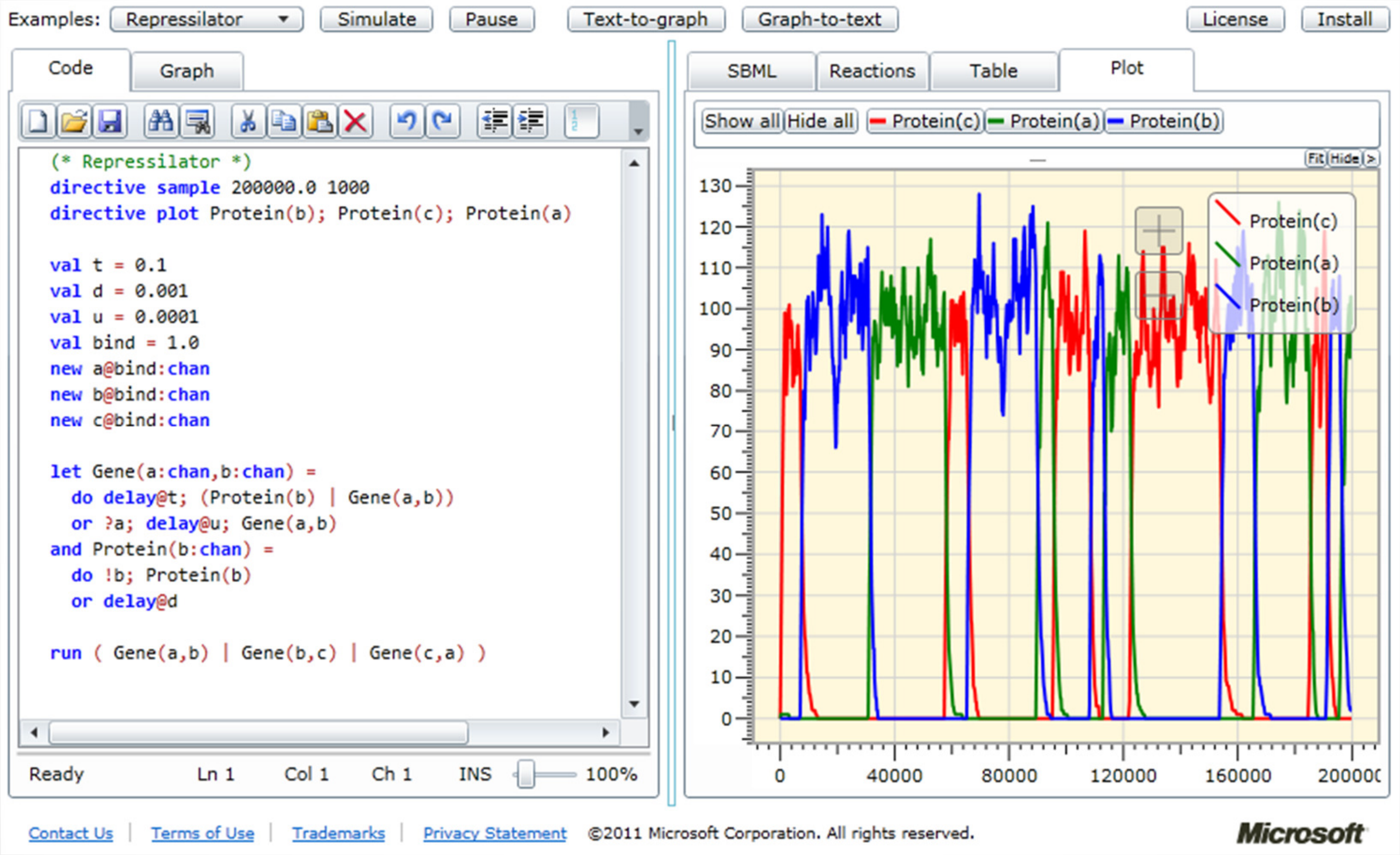
    $(?c(x);P\{x\}) \oplus P' \mid (!c(a);Q) \oplus Q' \quad = \quad P\{a/x\} \mid Q$

  - Scope extrusion (new 'ν' binder)

    If x not occurring in Q then $((\nu\ x)P) \mid Q = (\nu\ x)(P|Q)$

Implementations

- SPiM (Stochastic Pi Machine)
  - http://lepton.research.microsoft.com/VisualSPiM/
  - Runs in a browser with Silverlight.



```
(* Repressilator *)
directive sample 200000.0 1000
directive plot Protein(b); Protein(c); Protein(a)

val t = 0.1
val d = 0.001
val u = 0.0001
val bind = 1.0
new a@bind:chan
new b@bind:chan
new c@bind:chan

let Gene(a:chan,b:chan) =
  do delay@t; (Protein(b) | Gene(a,b))
  or ?a; delay@u; Gene(a,b)
and Protein(b:chan) =
  do !b; Protein(b)
  or delay@d

run ( Gene(a,b) | Gene(b,c) | Gene(c,a) )
```

©2011 Microsoft Corporation. All rights reserved.

Processes and Chemistry

Continuous Chemical Systems

Reactions:

$A \to^{r} B_1 + \ldots + B_n$	Degradation
$A_1 + A_2 \to^{r} B_1 + \ldots + B_n$	Asymmetric Collision
$A + A \to^{r} B_1 + \ldots + B_n$	Symmetric Collision

Continuous reaction kinetics, respectively:

$[A]^\bullet = -r[A]$	Exponential Decay
$[A_i]^\bullet = -r[A_1][A_2]$	Mass Action Law
$[A]^\bullet = -2r[A]^2$	Mass Action Law

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

π–calculus for Chemistry

- To compose *soups* P we need:
  - Stochastic channels: $(\nu\ x_r)\ P$ r is the rate of an exponential distribution: the rate of communication on that channel
  - Composition: $P \mid P$ (with identity elem. 0)
  - Recursion: $*P$ (equal to $P \mid *P$)

- To execute *species* we need:
  - Collision: $?x_r;\ P$ (with no input variables)
  - Co-collision: $!x_r;\ P$ (with no output messages)
  - Delay: $\tau_r;\ P$ (= $(\nu\ x_r)\ ?x_r;P|!x_r;0$ for any x not in P)
  - Choice: $P \oplus P$ (with identity elem. 0)

Discrete Chemical Systems (1)

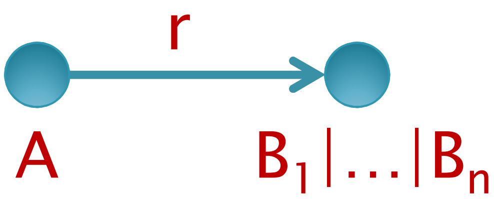
Reaction:

$$A \;\rightarrow^{r}\; B_1 + \ldots + B_n$$

Discrete reaction kinetics:

$$A = \tau_r;\, (B_1|\ldots|B_n)$$

The mathematical meaning of that is a Continuous Time Markov Chain (for a specific set of initial conditions, e.g. a single A molecule), here represented as a transition graph:



Hence the π–calculus description abstracts from initial conditions (like ODEs). For each set of initial conditions, a CTMC can be systematically extracted from the stochastic π–calculus models.

Discrete Chemical Systems (2)

(Uniquely named) reaction:

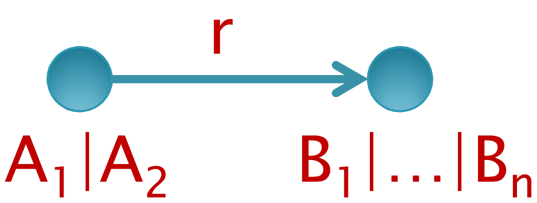
$c{:}\ \ A_1 + A_2 \ \ \to^r \ \ B_1 + \ldots + B_n$

Discrete reaction kinetics:

$A_1 = ?c_r;\ (B_1|\ldots|B_i)$ (the name of the reaction becomes the channel)

$A_2 = !c_r;\ (B_i|\ldots|B_n)$ (splitting results is arbitrary: $1 \leq i \leq n$)

With initial conditions $A_1|A_2$ (single molecules), the CTMC is:



Discrete Chemical Systems (3)

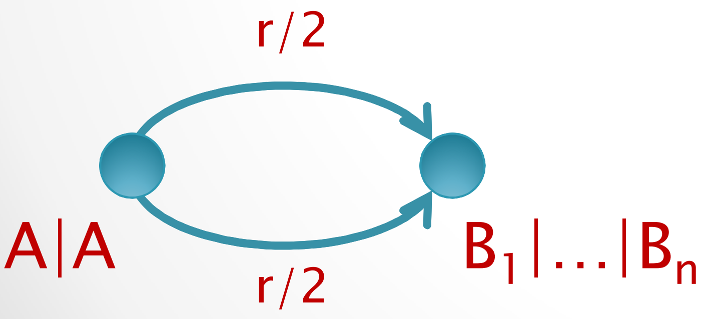
(Uniquely named) reaction:

$$c:\ A + A \ \to^{r}\ B_1 + \ldots + B_n$$

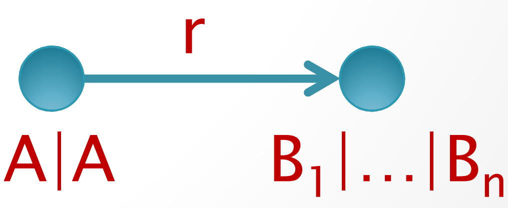
Discrete reaction kinetics:

$$A = ?c_{r/2};\ (B_1|\ldots|B_i) \oplus\ !c_{r/2};\ (B_i|\ldots|B_n) \qquad 1 \leq i \leq n$$

With initial conditions $A|A$ (two molecules), the CTMC is as follows; note that each copy of A can do an input or an output, so there are two possible paths to the outcome:



That is:



From Reactions to Processes

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
(1 per reaction)

Half-rate for symmetric reactions

processes
(1 per species)

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

Fill the matrix by columns:

Degradation reaction v_i: $X \rightarrow^{ki} P_i$

add $\tau;P_i$ to $\langle X,v_i \rangle$.

Asymmetric reaction v_i: $X+Y \rightarrow^{ki} P_i$

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

Symmetric reaction v_i: $X+X \rightarrow^{ki} P_i$

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

Read out the processes by rows:

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

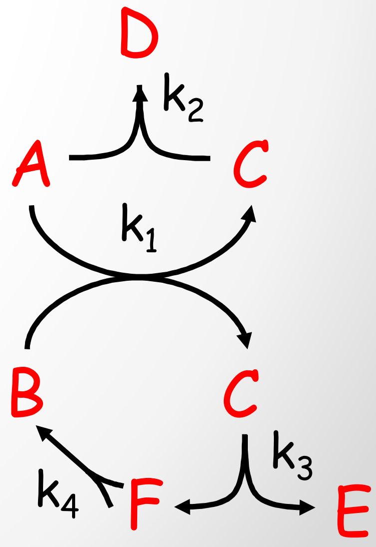
$B = !v_{1(k1)};0$

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

$D = 0$

$E = 0$

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



That Chemical System in SPiM

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

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

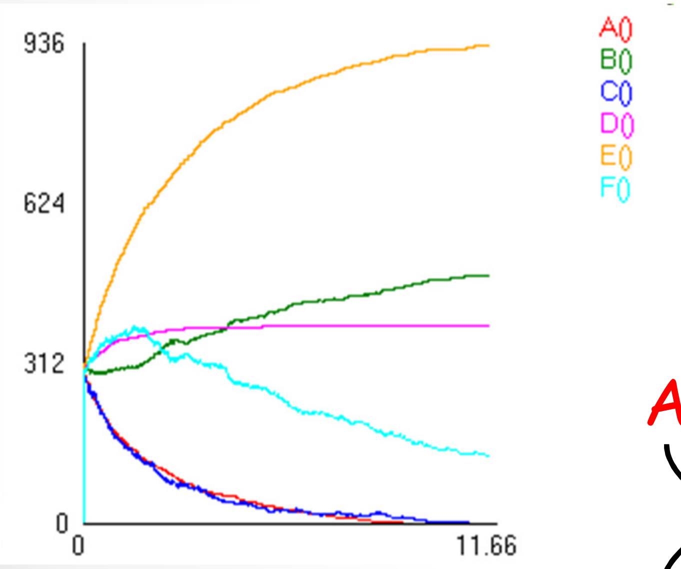
$C = \;!v_{2(k2)};0 \;\oplus\; \tau_{k3};(E|F)$

$D = 0$

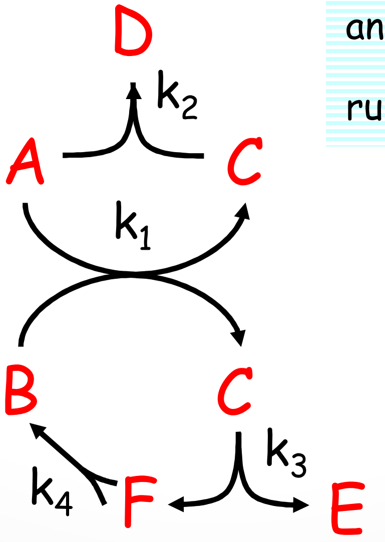
$E = 0$

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

```
directive sample 10.0
directive plot A(); B(); C(); D(); E(); F()

val k1 = 0.001  new v1@k1:chan
val k2 = 0.001  new v2@k1:chan
val k3 = 1.0
val k4 = 0.001  new v4@k4/2.0:chan

let A() = do ?v1;(C()|C()) or ?v2;D()
and B() = !v1
and C() = do !v2 or delay@k3;(E()|F())
and D() = ()
and E() = ()
and F() = do ?v4;B() or !v4

run 300 of (A()|B()|C()|D()|E()|F())
```



Gillespie-style
stochastic simulation



Modeling Techinques

- That is a *systematic* way to translate reactions to processes.
- But there can be *better* ways to do it.
- That is, ways that produce *more compact and/or modular models*, but with the *same kinetics*.

Ex: Catalysis

- Two reactions, same catalyst C
  - According to the general scheme the catalyst uses one channel for each reaction it catalyzes

a: $A + C \rightarrow^{r} C + B$ $\quad$ $C = !a_r; C \oplus !b_r; C$

b: $D + C \rightarrow^{r} C + E$ $\quad$ $A = ?a_r; B$

$D = ?b_r; E$

  - Modularizing: the catalyst has its own catalysis channel c, used for all the reactions it catalyzes:

$A + C \rightarrow^{r} C + B$ $\quad$ $C = !c_r; C$

$D + C \rightarrow^{r} C + E$ $\quad$ $A = ?c_r; B$

$D = ?c_r; E$

Processes and Biochemistry

π–calculus for Biochemistry

- Biochemistry here means: direct modeling of complexation and polymerization.
- We now go back to the full (and stochastic) π–calculus: we need to pass channels as messages!

Complexation

$$A + B \quad {}^{s}\!\leftrightarrow^{r} \quad A{:}B$$

There is no good notation for this reaction in chemistry: A:B is considered as a separate species (which leads to combinatorial explosion of models).

But there is a way to write this precisely in π–calculus. There is a single public *association* channel a_r at rate r, and many private *dissociations* channels d_s at rate s, one for each complexation event (created by ν):

$$\begin{aligned} A_{free} &= (\nu\ d_s)\ !a_r(d_s);\ A_{bound}(d_s) \\ A_{bound}(d_s) &= !d_s;\ A_{free} \end{aligned}$$

$$\begin{aligned} B_{free} &= ?a_r(d_s);\ B_{bound}(d_s) \\ B_{bound}(d_s) &= ?d_s;\ B_{free} \end{aligned}$$

Note that we are describing A *independently* of B: as in the catalysis example, A could form complexes with many different species over the a_r channel.

More compactly:

$$\begin{aligned} A &= (\nu\ d_s)\ !a_r(d_s);\ !d_s;\ A \\ B &= ?a_r(d_s);\ ?d_s;\ B \end{aligned}$$

Polymerization

- Polymerization is iterated complexation
    - It can be represente in π-calculus *finitely*, with one process (definition) for each monomer.
    - Note that polymerization cannot be described *finitely* in chemistry (or ODEs) because there it needs one reaction for each *length* of polymer.
    - The reason it works in π-calculus is because of the ν operator. It enables the finite representation of systems of potentially unbounded complexity.
    - Like real biochemistry, where the structure of each monomer is coded in a finite piece of DNA, and yet unbounded-length polymers happen.

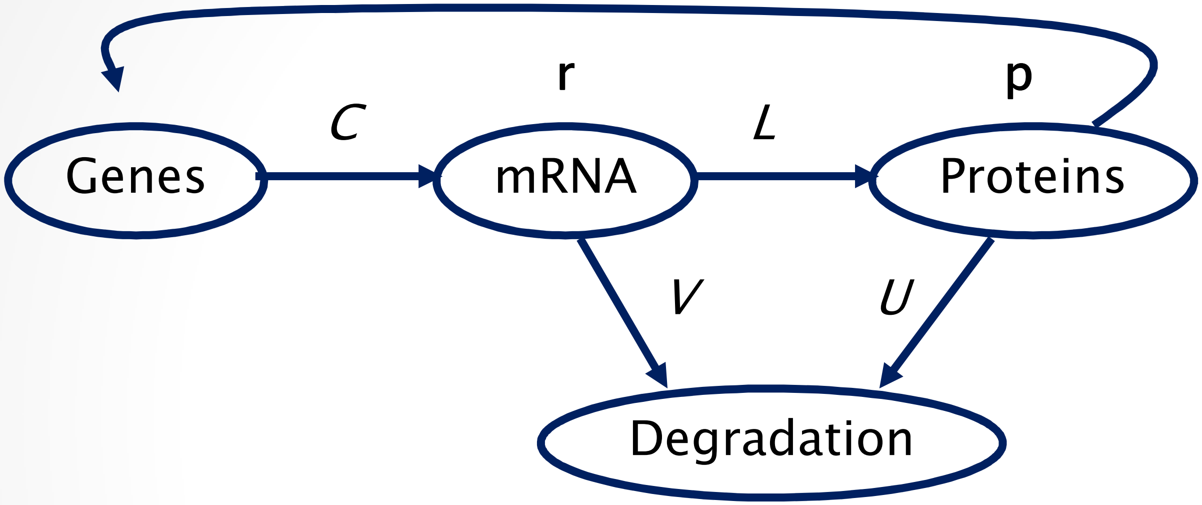
Processes and Genes

Higher–level Modeling

- The modeling level is up to you
  - We have seen how to use π–calculus to model chemical and biochemical events.
  - But it can be used to model any kind of ‘events’:
    - A process can represent: a molecule, a gene, a cell, an organism, etc.
    - A communication on a channel can represent: a molecular collision, a transcripion activation, a communication between cells, an interaction between organism (epidemics), etc.
  - Let’s see how to model gene networks
    - No longer single–molecule interactions, but rather interactions between genes and transcription factors.

The Classical ODE Approach

[Chen, He, Church]



$$\frac{d\mathbf{r}}{dt} = f(\mathbf{p}) - V\mathbf{r}$$

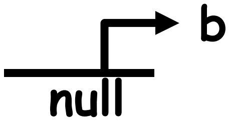
$$\frac{d\mathbf{p}}{dt} = L\mathbf{r} - U\mathbf{r}$$

n: number of genes
r mRNA concentrations (n-dim vector)
p protein concentrations (n-dim vector)

f(**p**) transcription functions:
(n-dim vector polynomials on **p**)

Nullary Gate

no input



spontaneous ("constitutive") output

interaction site of output protein

$$\mathrm{null}(b) = \tau_e;\ (\mathrm{tr}(b)\ |\ \mathrm{null}(b))$$

(recursive, parametric) process definition

stochastic delay (t) with rate e of constitutive transcription

and repeat

output protein (transcription factor), spawn out

A stochastic rate r is always associated with each channel a_r (at channel creation time) and delay t_r, but is often omitted when unambiguous.

Production and Degradation

$$tr(b) = (!b_r;\ tr(p)) \oplus \tau_d$$

transcription factor

interaction site of transcription factor

stochastic choice (race between r and d)

and repeat

degradation

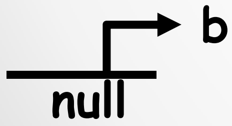
degradation rate d

(output, !) interaction with rate r (input, ?, is on the target gene)

A transcription factor is a *process* (not a message or a channel): it has behavior such as interaction on p and degradation.

combined effect of production and degradation (without any interaction on b)

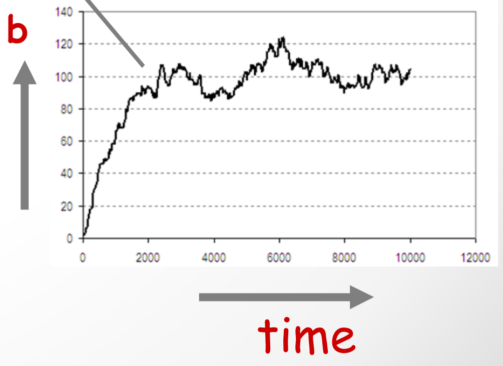
product



$$null(b) = \tau_e;\ (tr(b)\ |\ null(b))$$

null(b)

e=0.1, d=0.001

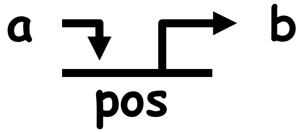


interaction offers on b

(= number of tr processes)

Unary Pos Gate

input (excitatory)



output (stimulated or constitutive)

$$\text{pos}(a,b) = \; ?a_r;\ \tau_h;\ (\text{tr}(b) \mid \text{pos}(a,b)) \oplus \tau_e;\ (\text{tr}(b) \mid \text{pos}(a,b))$$

(input, ?) interaction with rate r

transcription delay with rate h

or constitutive transcription to always get things started

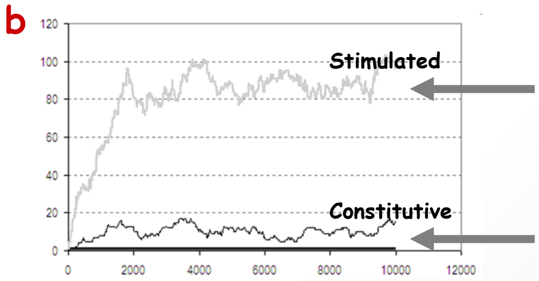
race between r and e

output protein

parallel, not sequence, to handle self-loops without deadlock

unlimited amount of

r=1.0, e=0.01, h=0.1, d=0.001

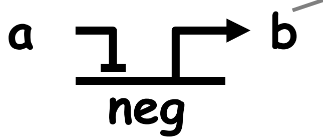


$*\text{tr}(a_r) \mid \text{pos}(a_r,b)$

$\text{pos}(a,b)$

Unary Neg Gate

input (inhibitory)



output (constitutive when not inhibited)

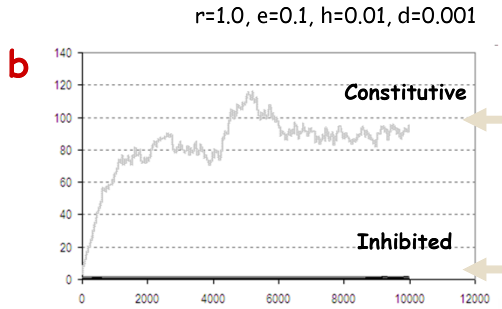
(input, ?) interaction with rate r

or constitutive transcription to always get things started

$$neg(a,b) = \; ?a_r;\ \tau_h;\ neg(a,b) \oplus \tau_e;\ (tr(b) \mid neg(a,b))$$

inhibition delay with rate h

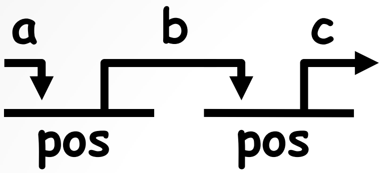
race between r and e



$neg(a_r,b)$

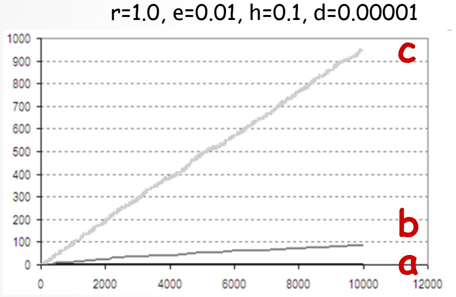
$*tr(a_r) \mid neg(a_r,b)$

Signal Amplification

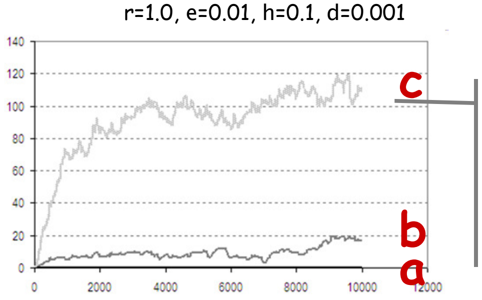


pos(a,b) | pos(b,c)

With little degradation



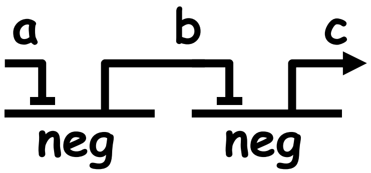
pos(a,b) | pos(b,c)



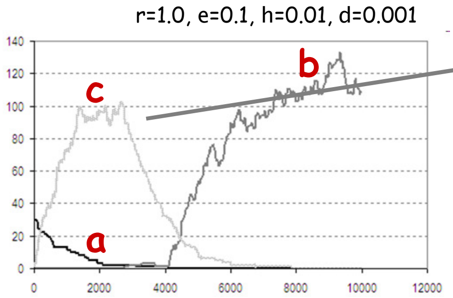
even with no a input, constitutive production of b gets amplified to a high c signal

pos(a,b) | pos(b,c)

Signal Normalization



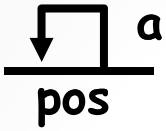
neg(a,b) | neg(b,c)



^{30}*tr(a) | neg(a,b) | neg(b,c)

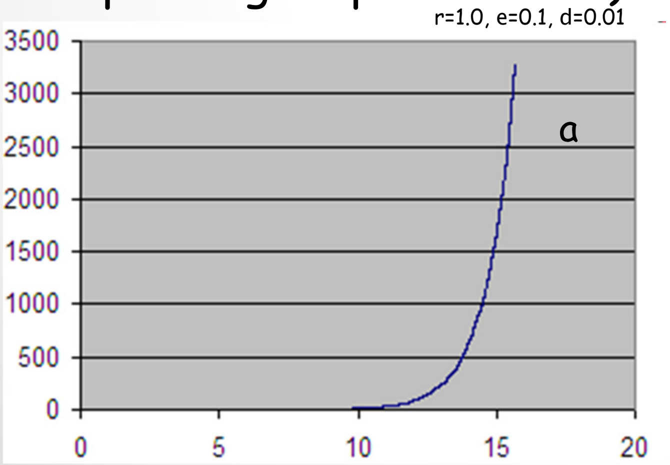
a non-zero input level, a, whether weak or strong, is renormalized to a standard level, c.

Self Feedback Circuits

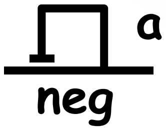


pos(a,a)

(Can overwhelm degradation, depending on parameters)

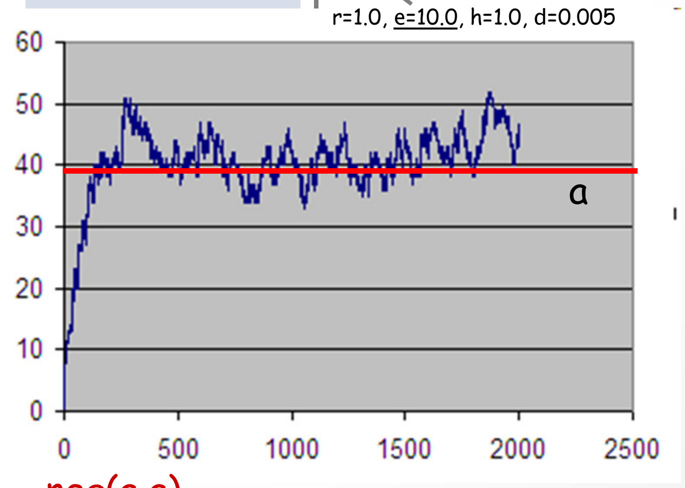


pos(a,a)



neg(a,a)

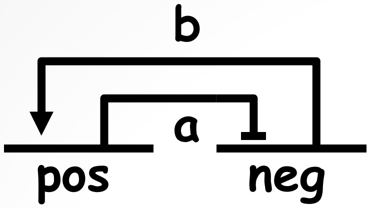
high, to raise the signal



neg(a,a)

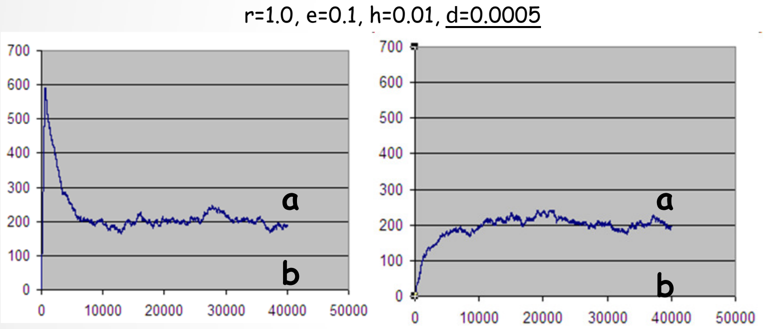
Two-gate Feedback Circuits

pos(b,a) | neg(a,b)



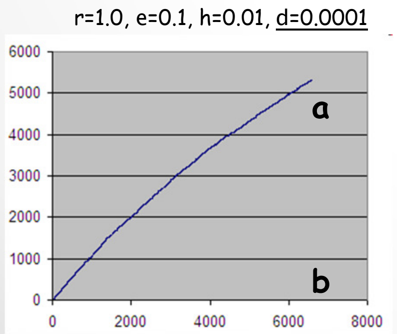
Monostable:

For some degradation rates is quite stable:



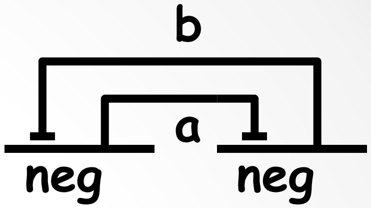
pos(b,a) | neg(a,b)

But with a small change in degradation, it goes wild:

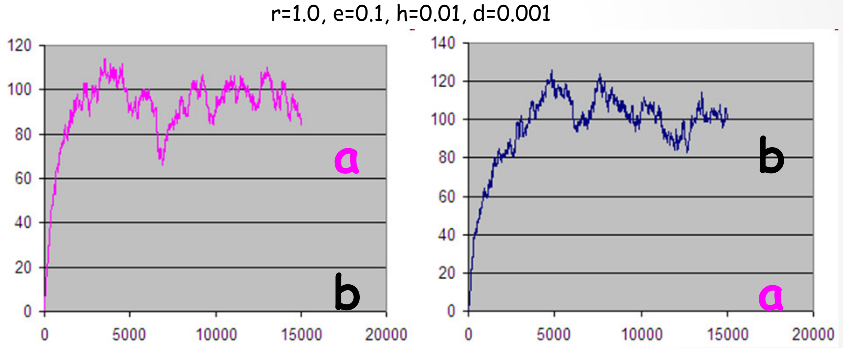


pos(b,a) | neg(a,b)

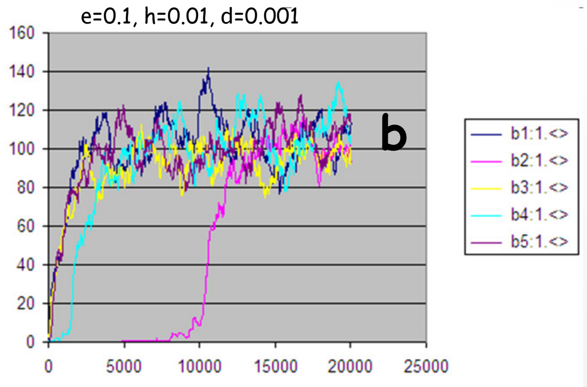
neg(b,a) | neg(a,b)



Bistable:



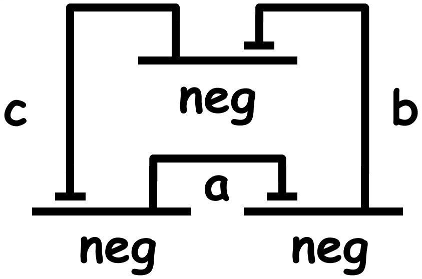
neg(b,a) | neg(a,b)



5 runs with r(a)=0.1, r(b)=1.0 shows that circuit is now biased towards expressing b

Repressilator

neg(a,b) |
neg(b,c) |
neg(c,a)



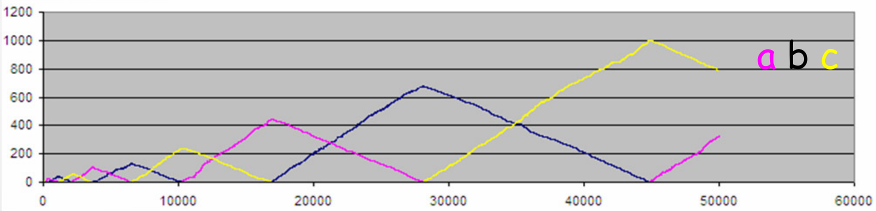
```
directive sample 50000.0  1000
directive plot !a; !b; !c

val dk = 0.001    (* Decay rate *)
val inh = 0.001   (* Inhibition rate *)
val cst = 0.1     (* Constitutive rate *)

let tr(p:chan()) = do !p; tr(p) or delay@dk

let neg(a:chan(), b:chan()) =
  do ?a; delay@inh; neg(a,b)
  or delay@cst; (tr(b) | neg(a,b))

val bnd = 1.0     (* Protein binding rate *)
new a@bnd:chan() new b@bnd:chan() new
c@bnd:chan()
run (neg(c,a) | neg(a,b) | neg(b,c))
```

Same circuit, three different degradation models by changing the tr component:

$\mathrm{tr}(p) = !p_r$

interact once and die
otherwise stick around

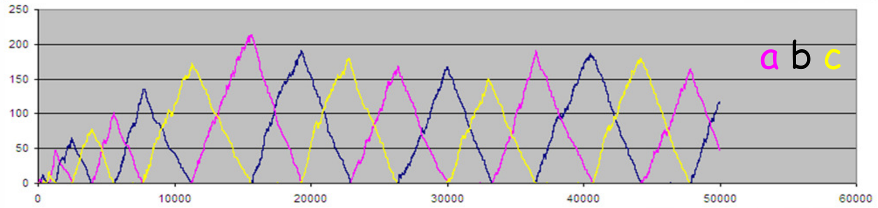
r=1.0, e=0.1, h=0.04



$\mathrm{tr}(p) = !p_r + \tau_d$

interact once and die
otherwise decay

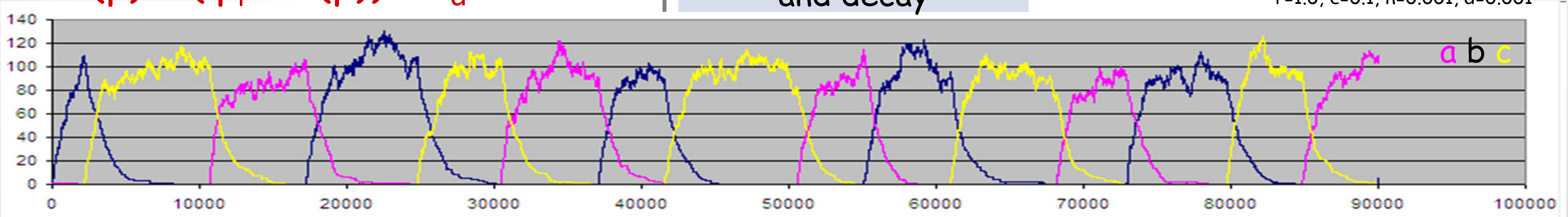
r=1.0, e=0.1, h=0.04, d=0.0001



$\mathrm{tr}(p) = (!p_r;\ \mathrm{tr}(p)) + \tau_d$

interact many times
and decay

r=1.0, e=0.1, h=0.001, d=0.001



Conclusions

Conclusions

- π–calculus
  - A mathematical notation for reactive systems
  - In stochastic form, suitable for representing discrete chemistry, biochemistry, etc.
  - Some unique properties: ability to finitely express systems of unbounded complexity, like networks of complexing proteins.

- Further Reading
  - R. Milner: **Communicating and Mobile Systems: The Pi Calculus**
  - A. Regev, E. Shapiro. **Cellular Abstractions: Cells as Computation.** NATURE vol 419, 2002-09-26, 343.
  - L. Cardelli: **From Processes to ODEs by Chemistry.** TCS 2008, DOI: http://dx.doi.org/10.1007/978-0-387-09680-3_18
  - A. Phillips,L. Cardelli, **A Correct Abstract Machine for the Stochastic Pi-calculus,** in Concurrent Models in Molecular Biology, 2004.