Theory is closer to practice
in theory than in practice.

Case Studies

Luca Cardelli

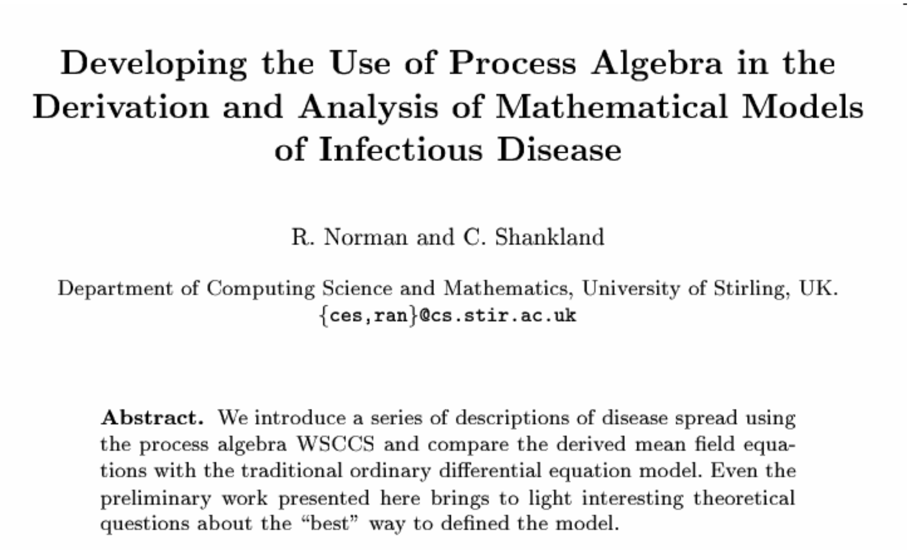
Microsoft Research

The Microsoft Research - University of Trento
Centre for Computational and Systems Biology

Trento, 2006-05-22..26

www.luca.demon.co.uk/ArtificialBiochemistry.htm

Epidemics



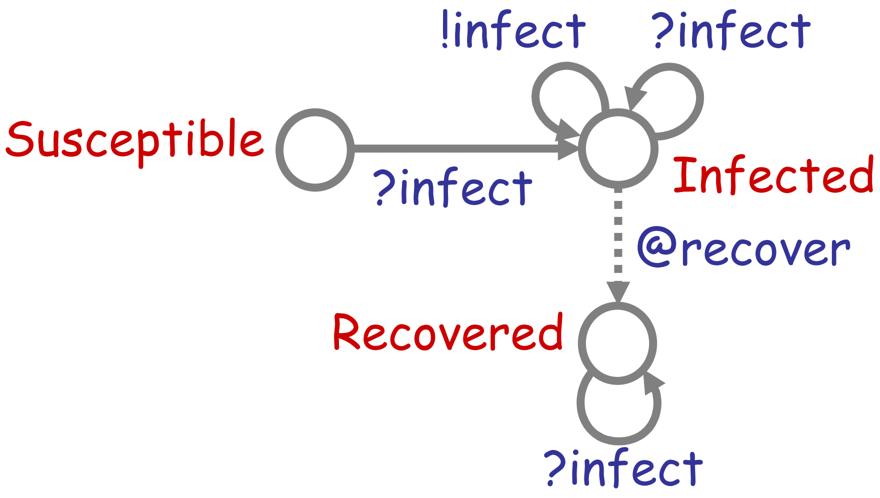
Developing the Use of Process Algebra in the Derivation and Analysis of Mathematical Models of Infectious Disease

R. Norman and C. Shankland

Department of Computing Science and Mathematics, University of Stirling, UK.
{ces,ran}@cs.stir.ac.uk

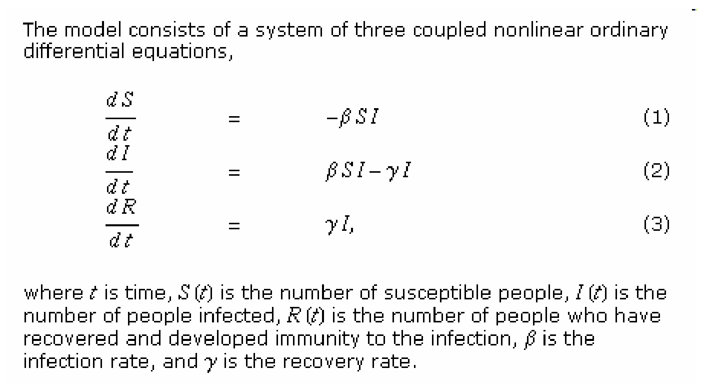
Abstract. We introduce a series of descriptions of disease spread using the process algebra WSCCS and compare the derived mean field equations with the traditional ordinary differential equation model. Even the preliminary work presented here brings to light interesting theoretical questions about the "best" way to defined the model.

Epidemics



```
directive sample 1000.0 1000
directive plot Recovered(); Susceptible(); Infected()

val recover_rate = 0.01
val infect_rate = 0.0001

new infect @infect_rate:chan()

let Recovered() =
  ?infect; Recovered()

and Susceptible() =
  ?infect; Infected()

and Infected() =
  do !infect; Infected()
  or ?infect; Infected()
  or delay@recover_rate; Recovered()

run (500 of Susceptible() | 1 of Infected())
```

As opposed to the way it is normally done.
http://mathworld.wolfram.com/Kermack-McKendrickModel.html



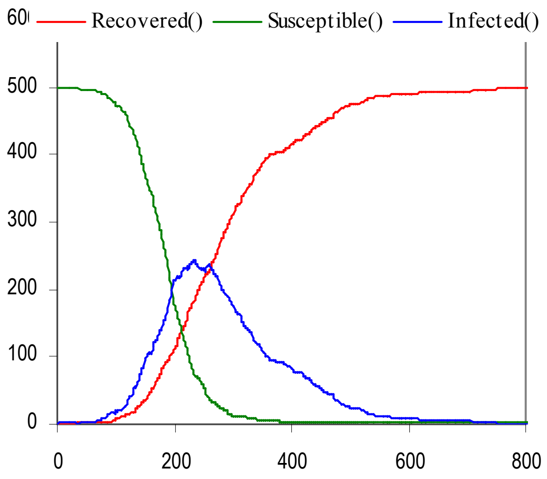
The model consists of a system of three coupled nonlinear ordinary differential equations,

$$\frac{dS}{dt} = -\beta S I \qquad (1)$$

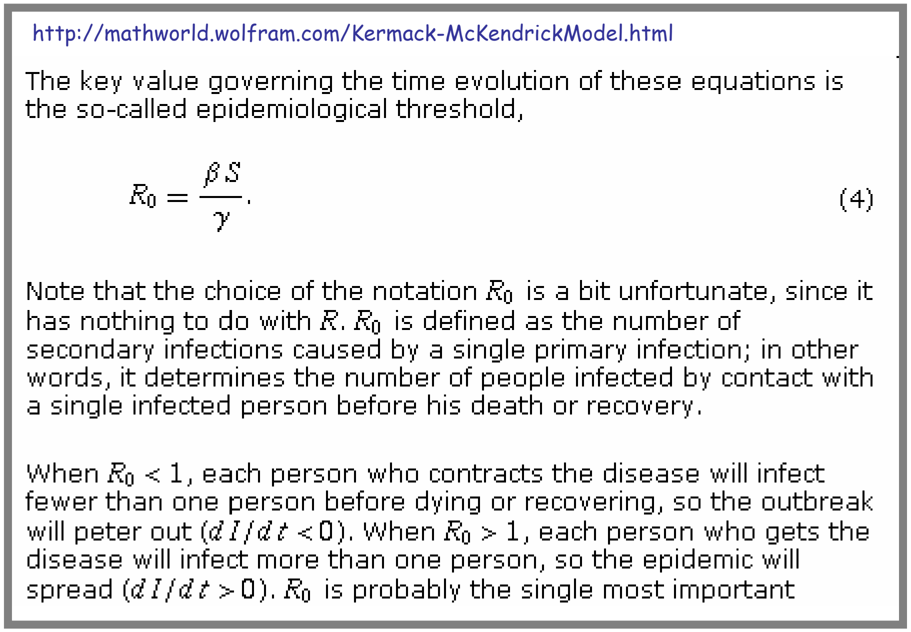
$$\frac{dI}{dt} = \beta S I - \gamma I \qquad (2)$$

$$\frac{dR}{dt} = \gamma I, \qquad (3)$$

where t is time, $S(t)$ is the number of susceptible people, $I(t)$ is the number of people infected, $R(t)$ is the number of people who have recovered and developed immunity to the infection, β is the infection rate, and γ is the recovery rate.



Exercise: Epidemic Simulations



http://mathworld.wolfram.com/Kermack-McKendrickModel.html

The key value governing the time evolution of these equations is the so-called epidemiological threshold,

$$R_0 = \frac{\beta S}{\gamma}. \qquad (4)$$

Note that the choice of the notation R_0 is a bit unfortunate, since it has nothing to do with R. R_0 is defined as the number of secondary infections caused by a single primary infection; in other words, it determines the number of people infected by contact with a single infected person before his death or recovery.

When $R_0 < 1$, each person who contracts the disease will infect fewer than one person before dying or recovering, so the outbreak will peter out ($dI/dt < 0$). When $R_0 > 1$, each person who gets the disease will infect more than one person, so the epidemic will spread ($dI/dt > 0$). R_0 is probably the single most important

Knowing that

β = infect_rate

γ = recover_rate

try various values to see how the infection progresses.

In the previous example, $R_0 = 5$ (everybody gets infected).

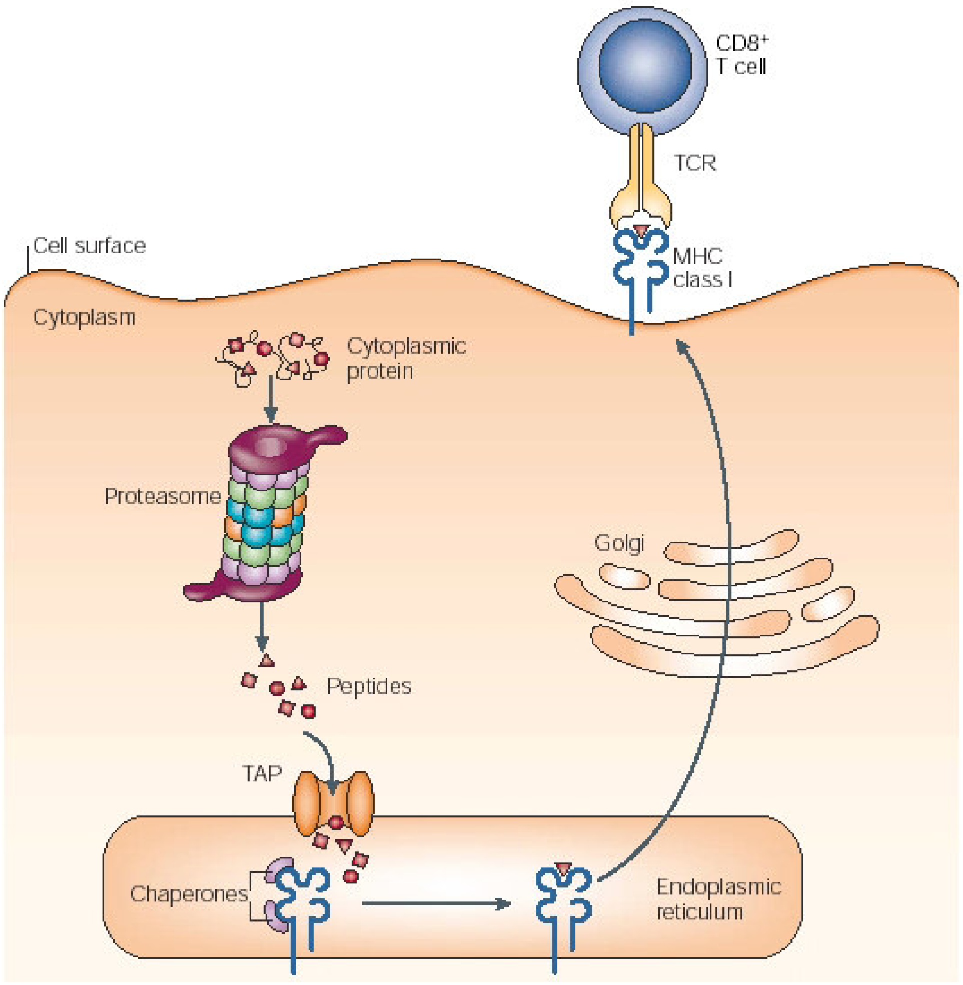
You can get $R_0 = 1$ (infection dies out) by reducing the S population to 100.

But stochastic effects (initial infected population = 1!) play a major role between $R_0 = 1$ and $R_0 = 5$.

MHC Class I Flytrap

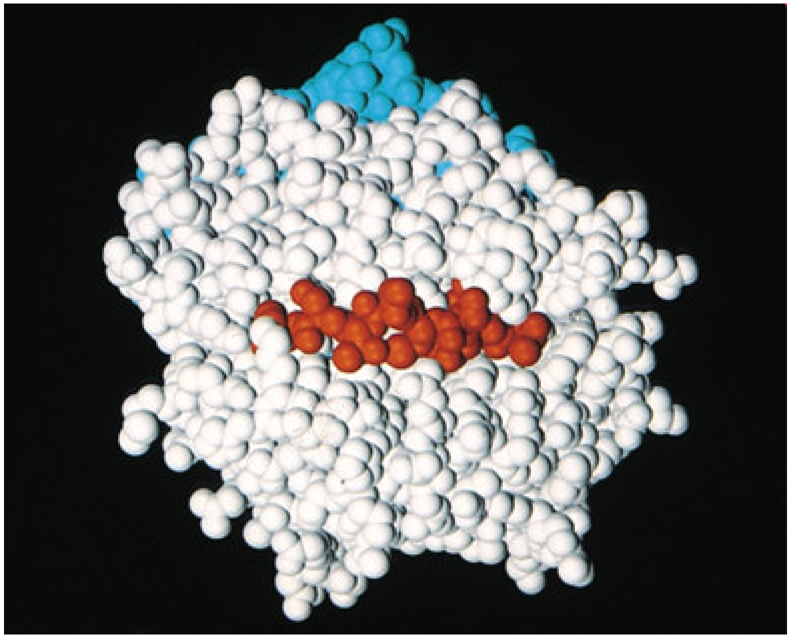
MHC Class I Antigen Presentation

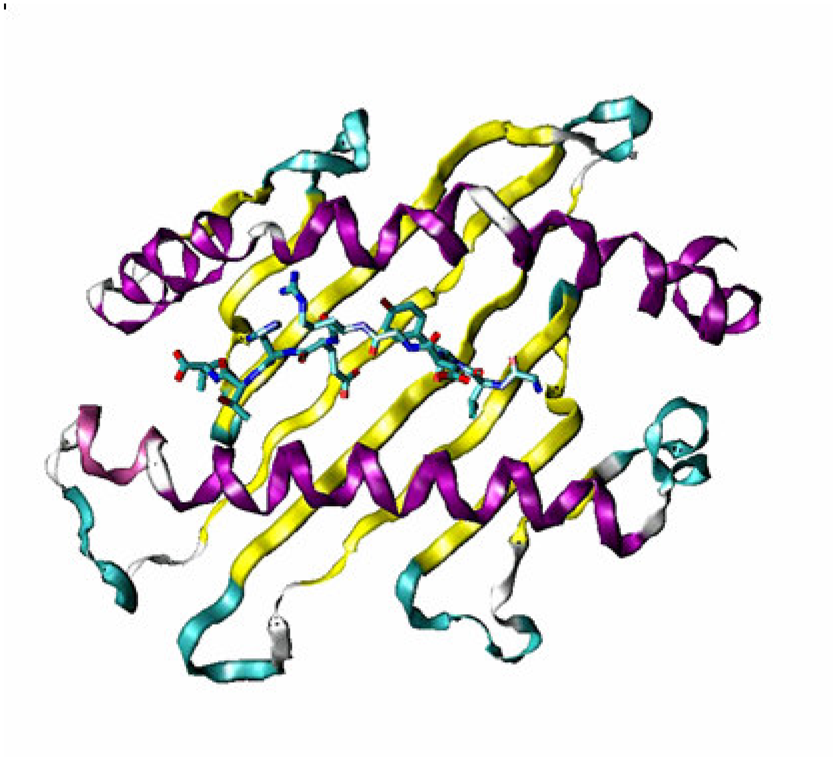
- part of the cellular immune response
- MHC class I complexes present self and foreign peptide at the cell surface
- recognized by T lymphocytes and natural killer cells
- also required for development of self tolerant T cells in thymus



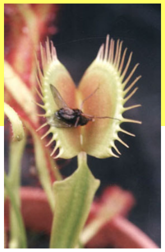
Source: Jonathan W. Yewdell, Eric Reits, and Jacques Neefjes. Making sense of mass destruction: quantitating MHC class I antigen presentation. *Nature Reviews Immunology*, 3(12):952–961, 2003.

MHC Class I Peptide Binding





T.J.Elliott



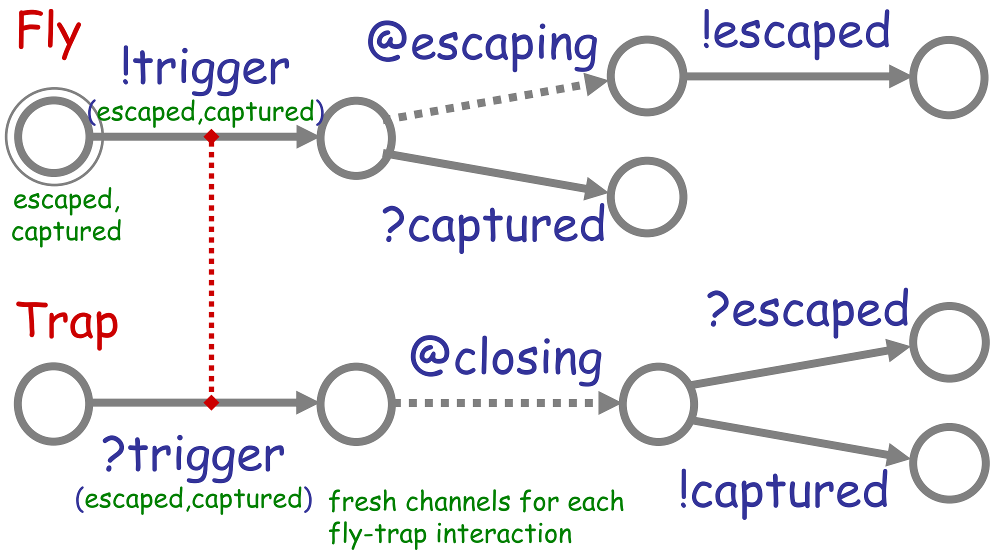
Flytrap

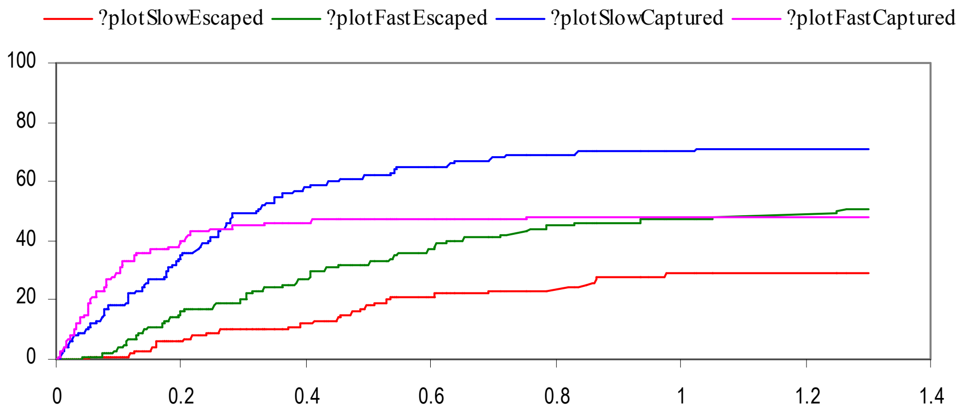
C.f.: L.Cardelli, T.J.Elliott, L.Goldstein, A.Phillips. J.M.Werner.

We want to model the situation where the trap is shutting at a constant rate, but different kinds of flies are escaping at different "dissociation" rates.

Hence we cannot model this simply as a channel of given rate where trap and flies synchronize.

We need to model a race between two delays in two independent processes But in the end, both the trap and the fly must agree on whether the fly was captured or not. (With no deadlock.)



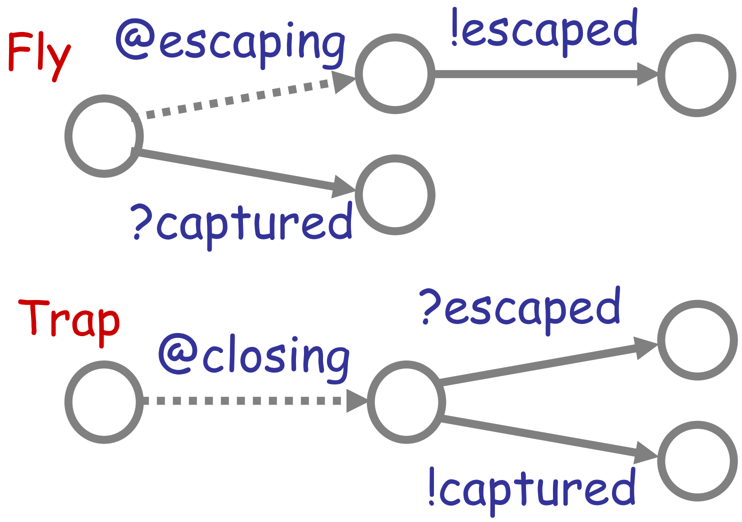


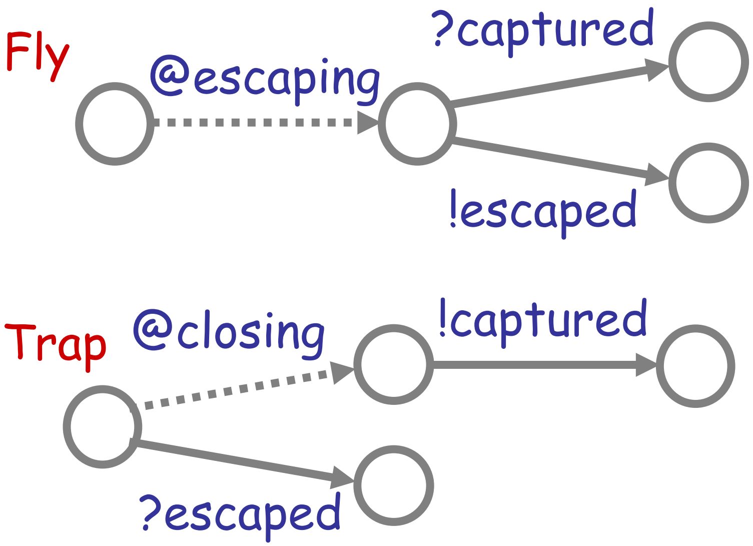
```
directive sample 4.0 10000
directive plot ?plotSlowEscaped; ?plotFastEscaped;
  !plotSlowCaptured; ?plotFastCaptured
new plotSlowEscaped@1.0:chan()
new plotFastEscaped@1.0:chan()
new plotSlowCaptured@1.0:chan()
new plotFastCaptured@1.0:chan()

new trigger@1000.0:chan(chan(),chan())
val closing = 3.0
val slowEscaping = 1.0
val fastEscaping = 5.0

let Fly(escaping:float, plotEscaped:chan(), plotCaptured:chan()) =
  (new captured@1000.0:chan()
   new escaped@1000.0:chan()
   !trigger(captured,escaped);
      do delay@escaping; !escaped; ?plotEscaped
      or ?captured; ?plotCaptured
  )

let Trap() =
  ?trigger(captured,escaped);
  delay@closing;
  do ?escaped or !captured

run (
100 of Fly(slowEscaping,plotSlowEscaped,plotSlowCaptured) |
100 of Fly(fastEscaping,plotFastEscaped,plotFastCaptured) |
200 of Trap())
```

Different Flytraps?

The two definitions are alpha-convertible!
Fly ⇔ Trap
escaping ⇔ closing
escaped ⇔ captured
Who is the fly and who is the trap?

It's a race, first, between @closing and @escaping.

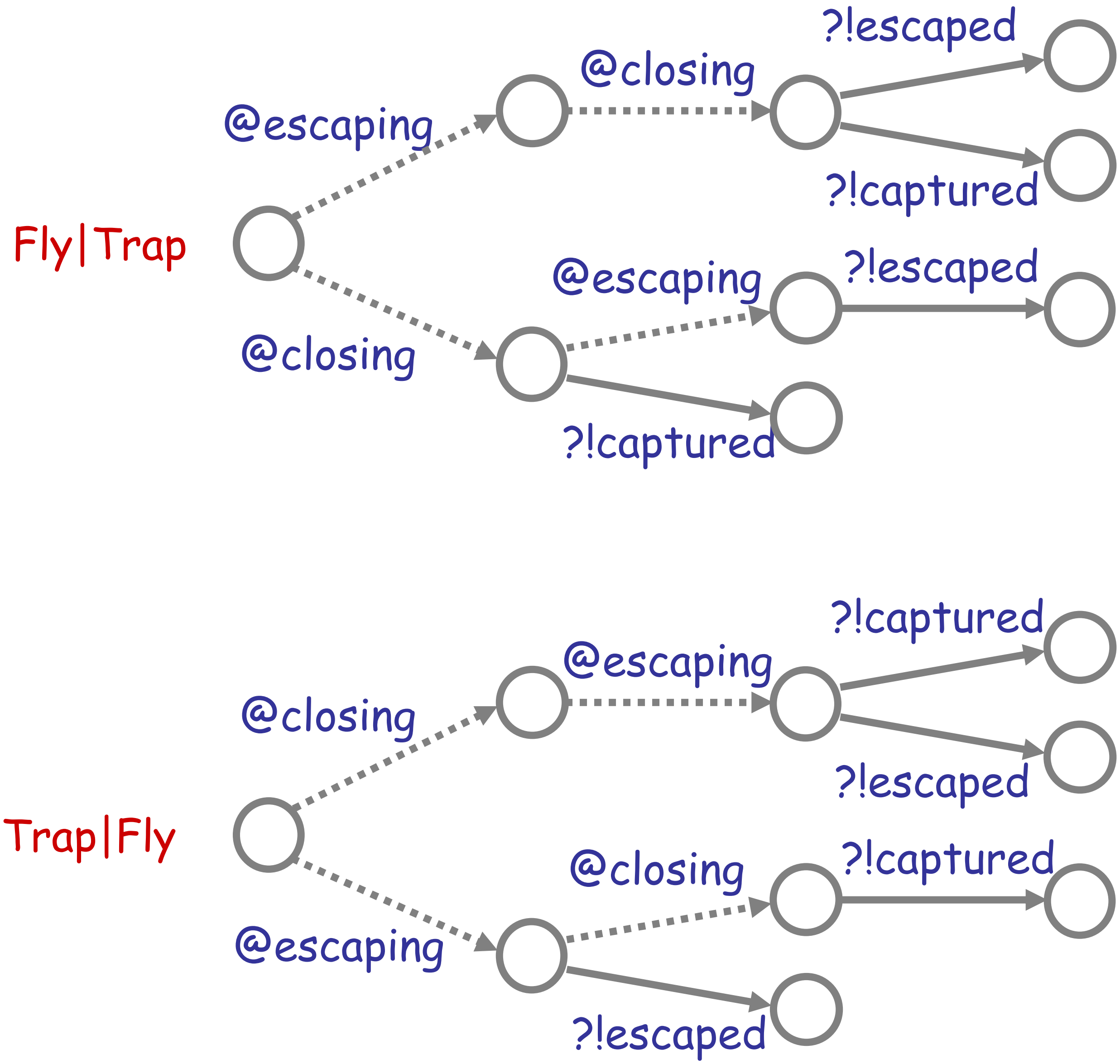




→ Finite Rates	If @escaping wins the race, the fly has escaped. If @closing wins the race, there is a second race between @escaping and ?!captured	If @closing wins the race, the fly is captured. If @escaping wins the race, there is a second race between @closing and ?!escaped
→ Infinite Rates	If @escaping wins the race, the fly has escaped. If @closing wins the race, the fly is captured. (Race between finite @escaping and infinite ?!captured)	If @closing wins the race, the fly is captured. If @escaping wins the race, the fly has escaped. (Race between finite @closing and infinite ?!escaped)

Equivalent flytraps if → transitions have infinite rate?

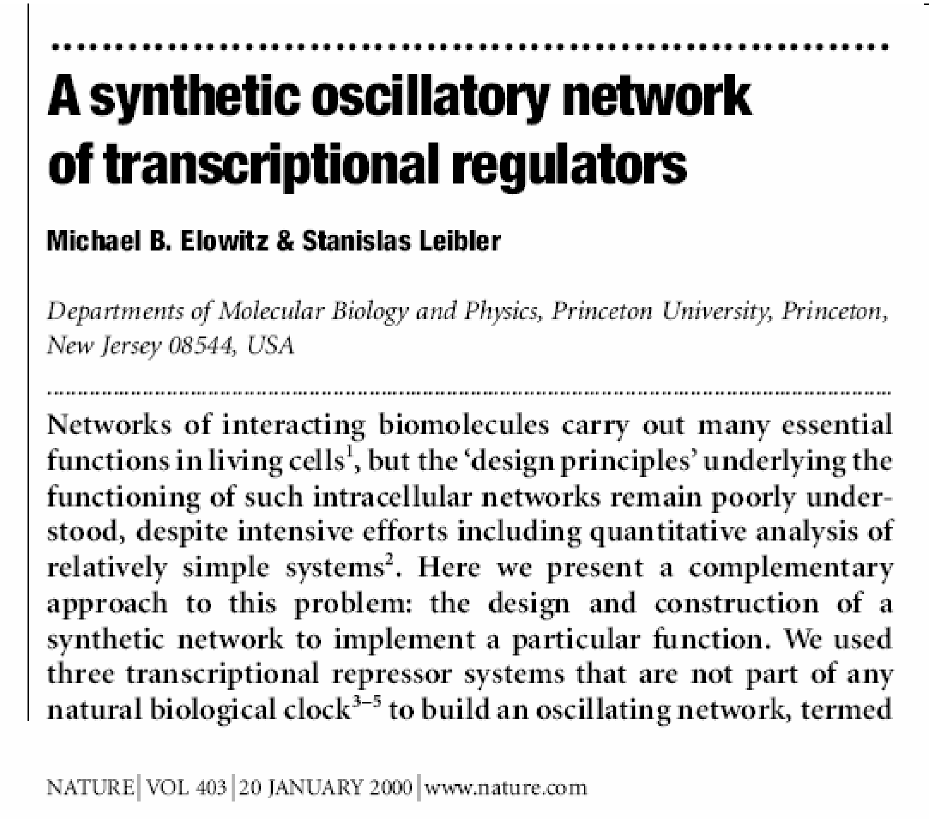
Flytrap Product Automata



Exercise (Open)

- Prove or disprove that the two flytraps are equivalent
  - Not necessarily for all intermediate states or quantities but, e.g.,
    - Do Escaped-Flies have the same distribution in both version?
  - What about the infinite-rate version?

Repressilator



A synthetic oscillatory network of transcriptional regulators

Michael B. Elowitz & Stanislas Leibler

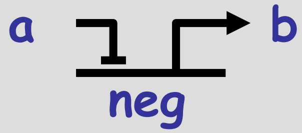
Departments of Molecular Biology and Physics, Princeton University, Princeton, New Jersey 08544, USA

Networks of interacting biomolecules carry out many essential functions in living cells[1], but the 'design principles' underlying the functioning of such intracellular networks remain poorly understood, despite intensive efforts including quantitative analysis of relatively simple systems[2]. Here we present a complementary approach to this problem: the design and construction of a synthetic network to implement a particular function. We used three transcriptional repressor systems that are not part of any natural biological clock[3–5] to build an oscillating network, termed

NATURE | VOL 403 | 20 JANUARY 2000 | www.nature.com

Gene Gates and Circuits

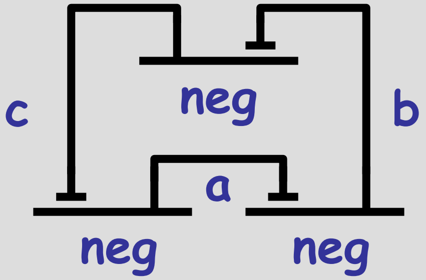
A gene gate



$$neg(a,b) \triangleq ?a_r;\ \tau_\eta;\ neg(a,b) + \tau_\varepsilon;\ (tr(b) \mid neg(a,b))$$

$$tr(p) \triangleq (!p_r;\ tr(p)) + \tau_\delta$$

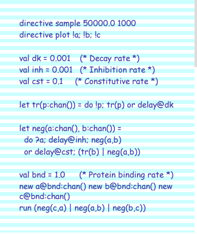
A genetic circuit (engineered in E.Coli)



$$neg(a,b) \mid neg(b,c) \mid neg(c,a)$$

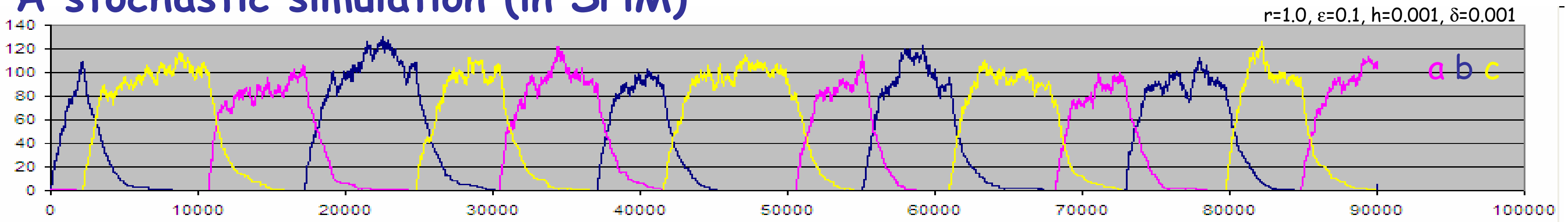
The stochastic-π program

```
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))

(* The circuit *)
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))
```



```
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))
```

A stochastic simulation (in SPiM)



Scaling Reactions

Scaling Moles and Rates

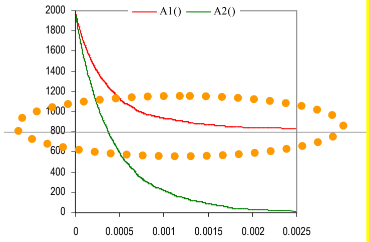
- Say the original reaction specifies a *quantity* of 1mol for a given species A, with reaction rates expressed in mol/Ls (rate of change in *concentration* mol/L of reactants). (Or maybe it specifies a *concentration* of 1μM = one millionth of a mole *per liter*, with rates expressed in μM/s; it does not matter here.) How many processes should we use for the simulation? Well, we can't use quantity 1; that's too few!
- So, let's multiply all mol quantities by say, 10, and use that many processes. What effect does that have on the reactions? Unary (decay) reactions now operate on an initial quantity 10*bigger. However those are exponential decays, which means that the half-life (and the general shape) of the reaction is *independent of the initial quantity*. So the time it takes for such reactions to operate *does not change*, and we do not have to scale the time axis (although our vertical axis is now off by a factor of 10).
- Binary reactions, however, now operate on 10*bigger quantities and by the mass action law run 100 times faster (–k(10*[A] 10*[B])), which means they are 10 times too fast with respect to the degradations. We can compensate by dividing the rates of *those* reactions by 10, the scaling factor.
- That way, in the end, our plots have the same curves as for any other scaling factor, and an accurate timeline, but the vertical axis numbers must be divided by 10 to compare with the original 1mol quantity.

Scaling Quantities and Rates

For example:

$A1 + A2 \rightarrow^{c} 0$

$A2 \rightarrow^{deg} 0$

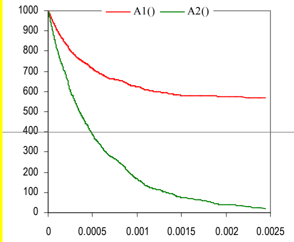


Original reaction
2000 molecules each
c=1.0 deg=1000.0

```
directive sample 0.0025 1000
directive plot A1(); A2()

new c@1.0:chan
val deg = 1000.0

let A1() = ?c;()
and A2() = do !c;() or delay@deg;()

run 2000 of (A1() | A2())
```

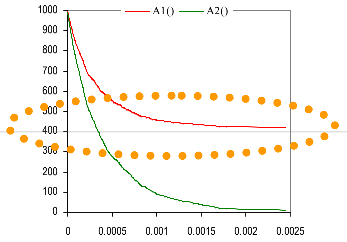
Scaling down the molecules by a factor of 2

Molecules halved
1000 molecules each
c=1.0 deg=1000.0
y*2 (plot rescaled)

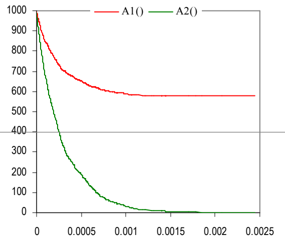


does not match

Binary rate doubled
1000 molecules
c=2.0 deg=1000.0
y*2

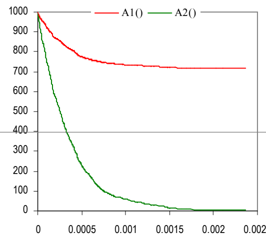


Both rates doubled
1000 molecules
c@2.0 deg=2000.0
y*2



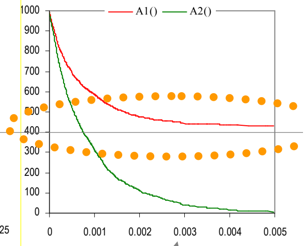
does not match

Degradation rate doubled
1000 molecules each
c=1.0 deg=1000.0
y*2



does not match

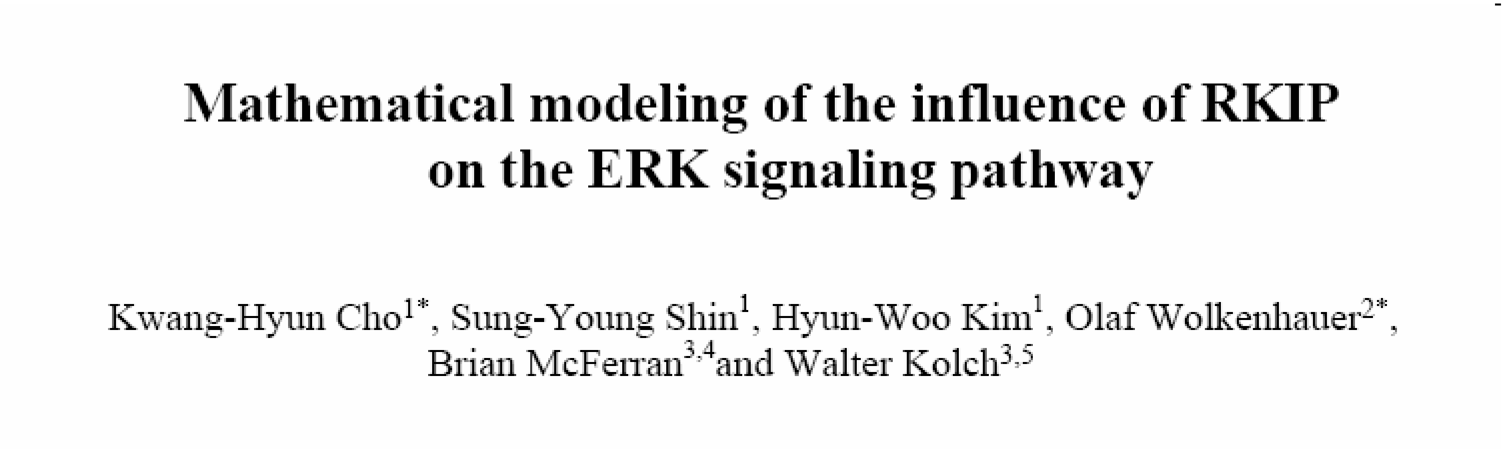
Degradation rate halved
1000 molecules each
c=1.0 deg=500.0
y*2, x/2



To get the same curves (up to rescaling of the y axis) we need to scale up the rate of binary reactions (only) by the same factor.

Scaling down the degradation rates by the same factor works too, but then we have to rescale the x axis as well.

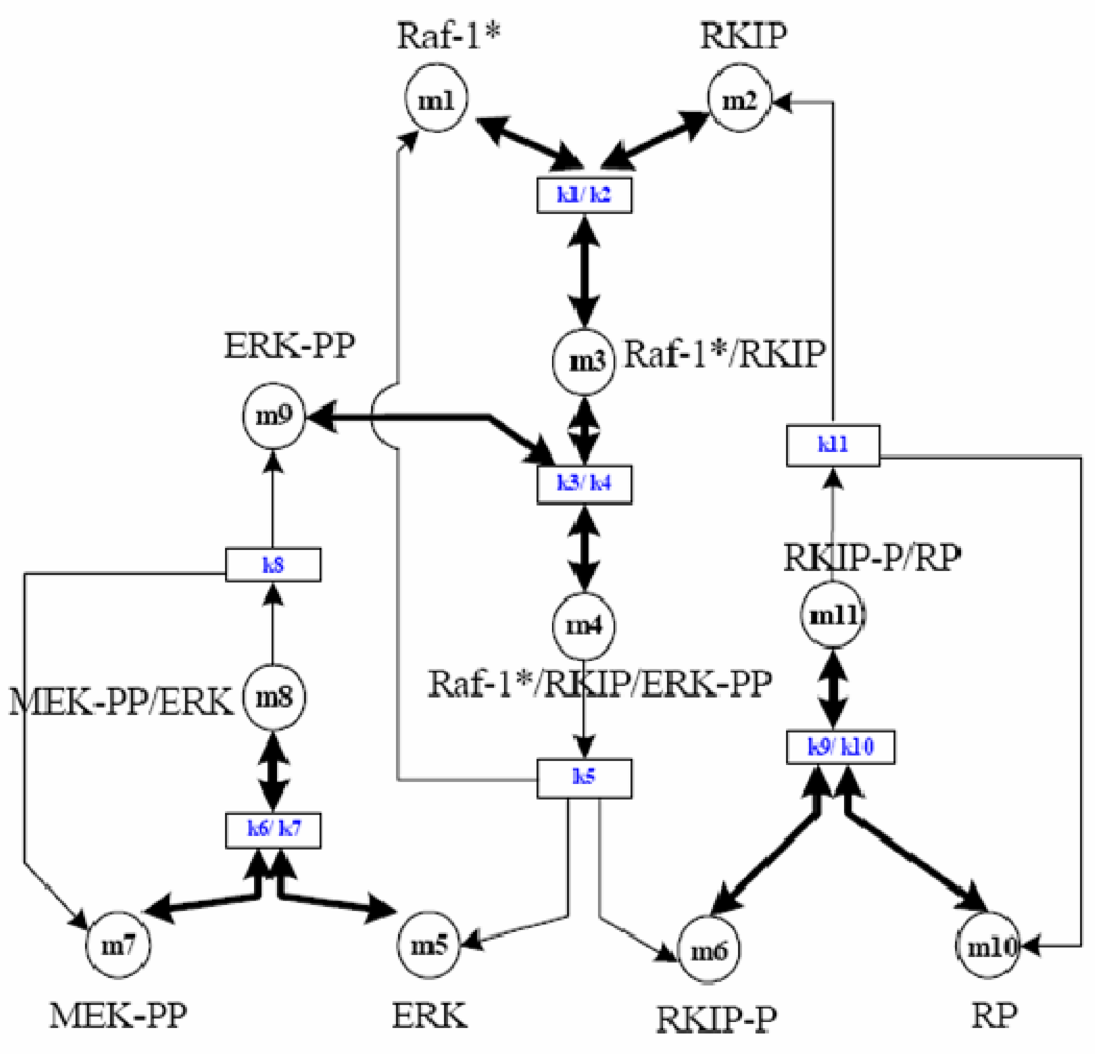
ERK Pathway



Mathematical modeling of the influence of RKIP on the ERK signaling pathway

Kwang-Hyun Cho[1*], Sung-Young Shin[1], Hyun-Woo Kim[1], Olaf Wolkenhauer[2*],
Brian McFerran[3,4]and Walter Kolch[3,5]

ERK Pathway



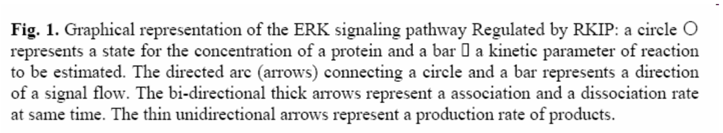
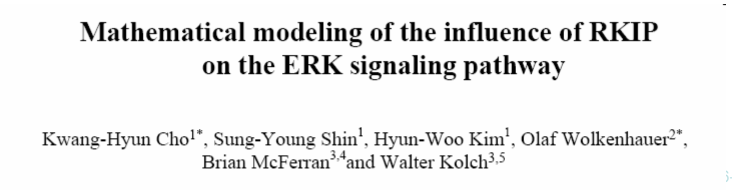


Fig. 1. Graphical representation of the ERK signaling pathway Regulated by RKIP: a circle ○ represents a state for the concentration of a protein and a bar ▯ a kinetic parameter of reaction to be estimated. The directed arc (arrows) connecting a circle and a bar represents a direction of a signal flow. The bi-directional thick arrows represent a association and a dissociation rate at same time. The thin unidirectional arrows represent a production rate of products.


Mathematical modeling of the influence of RKIP on the ERK signaling pathway

Kwang-Hyun Cho[1*], Sung-Young Shin[1], Hyun-Woo Kim[1], Olaf Wolkenhauer[2*], Brian McFerran[3,4]and Walter Kolch[3,5]

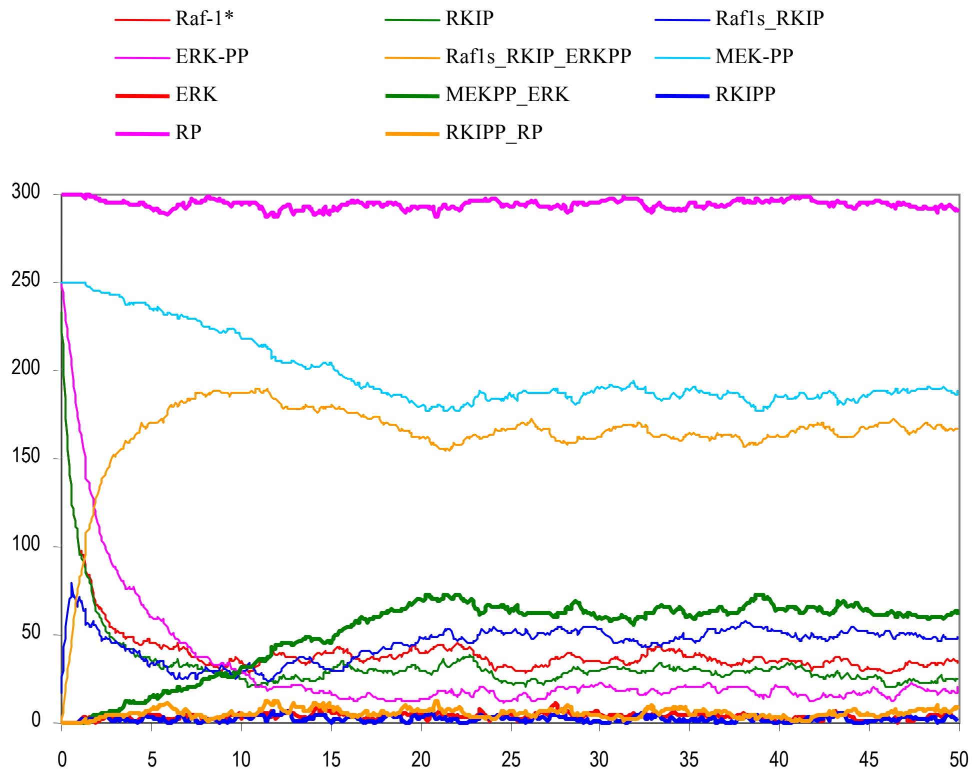
ERK Pathway in SPiM

```
(* ERK Signalling, Cho et al. *)
directive sample 50.0 1000
directive plot
(* Plot all *)
!k1 as "Raf-1*"; ?k1 as "RKIP"; !k3 as "Raf1s_RKIP"; ?k3 as "ERK-PP"; ?b1 as "Raf1s_RKIP_ERKPP";
!k6 as "MEK-PP"; ?k6 as "ERK"; ?b2 as "MEKPP_ERK"; !k9 as "RKIPP"; ?k9 as "RP"; ?b3 as "RKIPP_RP"

(* Plot Fig 5 top left
  !k1 as "Raf-1*"; ?k1 as "RKIP"; !k3 as "Raf1s_RKIP" *)
(* Plot Fig 5 top right
  ?k6 as "ERK"; !k6 as "MEK-PP"; ?b2 as "MEKPP_ERK"; ?k3 as "ERK-PP" *)
(* Plot Fig 5 bottom left
  !k3 as "Raf1s_RKIP" ; ?k3 as "ERK-PP"; ?b1 as "Raf1s_RKIP_ERKPP"  *)
(* Plot Fig 5 bottom right
  !k9 as "RKIPP"; ?k9 as "RP"; ?b3 as "RKIPP_RP"; ?k1 as "RKIP" *)
(* Plot MEK-PP
  !k6 as "MEK-PP" *)

new b1@1.0:chan()  (* dummy barbs for plotting *)
new b2@1.0:chan()
new b3@1.0:chan()

(* ----------------------------------- *)
val quantity = 100.0
val concentration = 1.0

(* Binary reactions *)
new k1  @ 0.53/quantity :chan()
new k3  @ 0.625/quantity :chan()
new k6  @ 0.8/quantity :chan()
new k9  @ 0.92/quantity :chan()

(* Decay reactions *)
val k2  = 0.0072
val k4  = 0.00245
val k5  = 0.0315
val k7  = 0.0075
val k8  = 0.071
val k10 = 0.00122
val k11 = 0.87
```

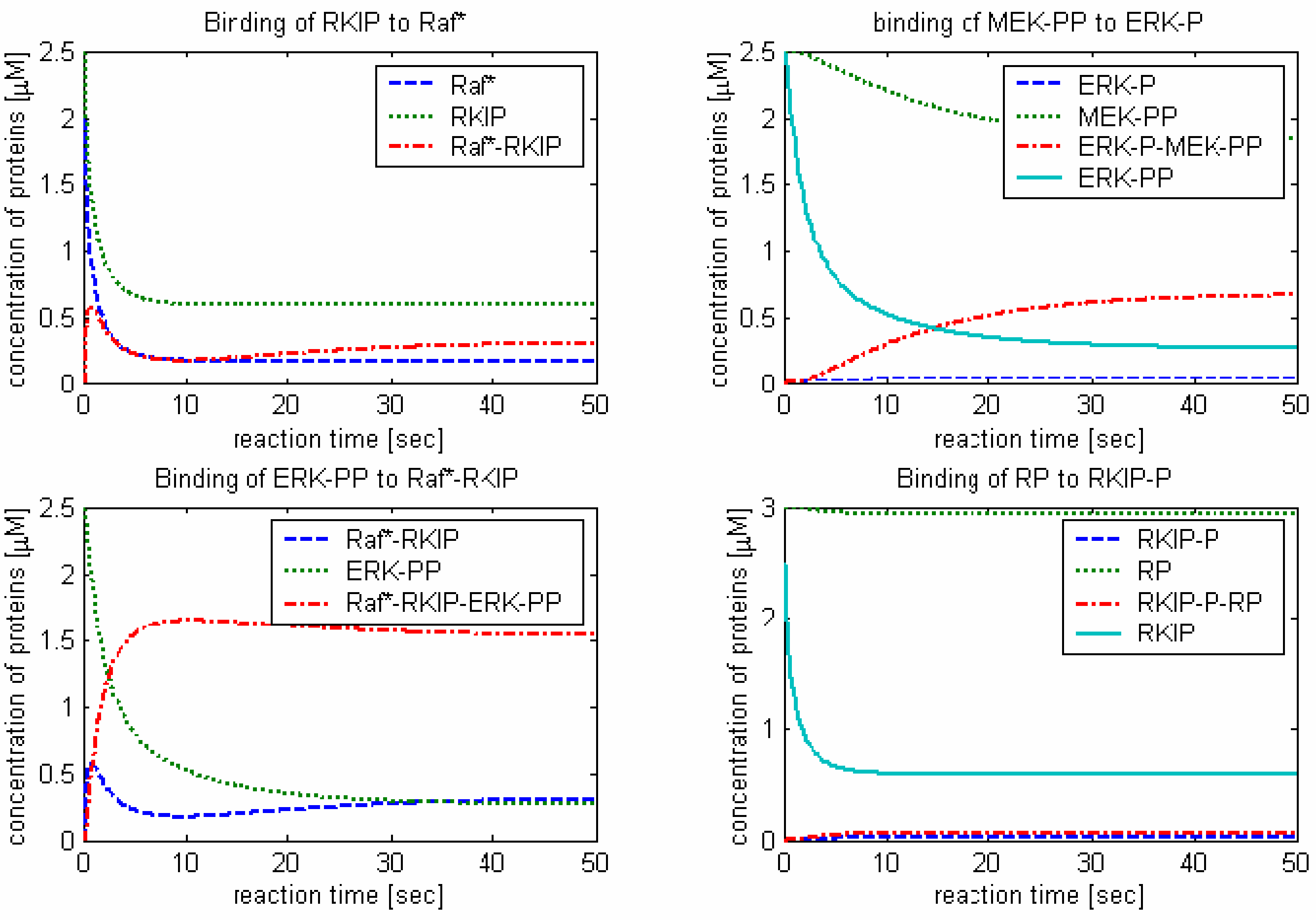
```
(* -------------------
Initial concentrations
m1=2.5, m2=2.5, m3=0, m4=0, m5=0, m6=0, m7=2.5, m8=0,
m9=2.5, m10=3, m11=0
Raf1s=2.5, RKIP=2.5, Raf1s_RKIP=0,
Raf1s_RKIP_ERKPP=0,
ERK=0, RKIPP=0, MEKPP=2.5, MEKPP_ERK=0, ERKPP=2.5,
RP=3, RKIPP_RP=0

Reactions
[01] Raf-1* + RKIP -->k1  Raf-1*_RKIP
[02] Raf-1*_RKIP -->k2 Raf-1* + RKIP
[03] Raf-1*_RKIP + ERK-PP -->k3 Raf-1*_RKIP_ERK-PP
[04] Raf-1*_RKIP_ERK-PP -->k4 Raf-1*_RKIP + ERK-PP
[05] Raf-1*_RKIP_ERK-PP -->k5 Raf-1* + RKIP-P + ERK
[06] MEK-PP + ERK -->k6 MEK-PP_ERK
[07] MEK-PP_ERK -->k7 MEK-PP + ERK
[08] MEK-PP_ERK -->k8 MEK-PP + ERK-PP
[09] RKIP-P + RP -->k9 RKIP-P_RP
[10] RKIP-P_RP -->k10 RKIP-P + RP
[11] RKIP-P_RP -->k11 RKIP + RP
*)
```

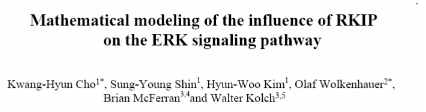
```
let Raf1s() =
  !k1; Raf1s_RKIP()              (* ![01] *)

and RKIP() =
  ?k1; ()                                  (* ?[01] *)

and Raf1s_RKIP() =
   do delay@k2; (Raf1s() | RKIP())            (*  [02] *)
   or !k3; Raf1s_RKIP_ERKPP()          (* ![03] *)

and ERKPP() =
  ?k3; ()               (* ?[03] *)

and Raf1s_RKIP_ERKPP() =
  do delay@k4; (Raf1s_RKIP() | ERKPP())   (*  [04] *)
  or delay@k5; (Raf1s() | RKIPP() | ERK()) (*  [05] *)
  or ?b1

and MEKPP() =
  !k6; MEKPP_ERK()            (* ![06] *)

and ERK() =
  ?k6; ()               (* ?[06] *)

and MEKPP_ERK() =
  do delay@k7; (MEKPP() | ERK())             (*  [07] *)
  or delay@k8; (MEKPP() | ERKPP())           (*  [08] *)
  or ?b2

and RKIPP() =
  !k9; RKIPP_RP()                (* ![09] *)

and RP() =
  ?k9; ()                    (* ?[09] *)

and RKIPP_RP() =
  do delay@k10; (RKIPP() | RP())        (*  [10] *)
  or delay@k11; (RKIP() | RP())           (*  [11] *)
  or ?b3

let many(n:float, p:proc()) =  if n<=0.0 then () else p(); many(n-1.0, p)

run many(2.5*quantity*concentration, Raf1s)
run many(2.5*quantity*concentration, RKIP)
run many(2.5*quantity*concentration, ERKPP)
run many(2.5*quantity*concentration, MEKPP)
run many(3.0*quantity*concentration, RP)
```

SPiM Simulation



Original Simulation





Mathematical modeling of the influence of RKIP on the ERK signaling pathway

Kwang-Hyun Cho[1*], Sung-Young Shin[1], Hyun-Woo Kim[1], Olaf Wolkenhauer[2*], Brian McFerran[3,4]and Walter Kolch[3,5]

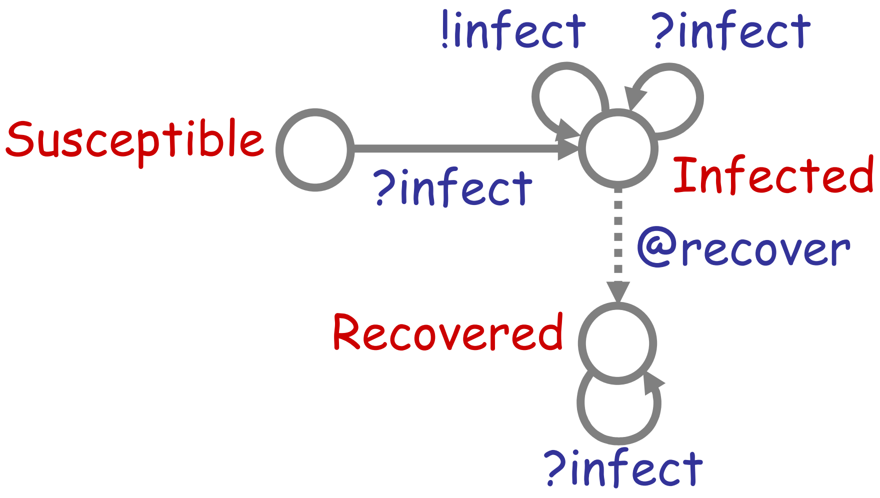
Fig. 5. Simulation results of the mathematical modeling for fixed initial condition: the upper left shows the dynamics for Raf-1*, RKIP, and their complex Raf-1*/RKIP, the upper right shows the activity of MEK-PP which phosphorylates and activates ERK, the lower left shows the activity of ERK-PP, and the lower right shows the activity of RP.

Epidemics ODE

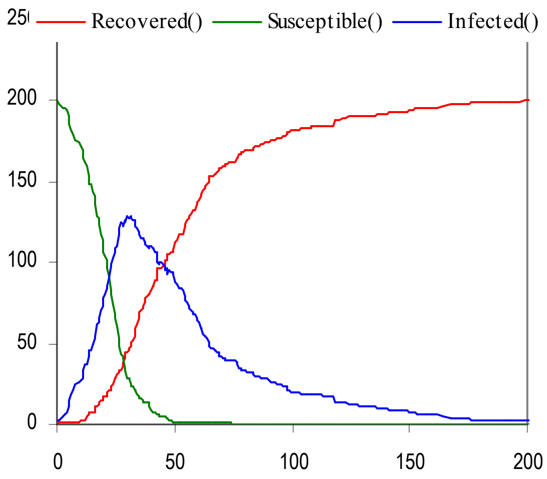
Kermack, W. O. and McKendrick, A. G. "**A Contribution to the Mathematical Theory of Epidemics.**" *Proc. Roy. Soc. Lond. A* **115**, 700-721, 1927.

http://mathworld.wolfram.com/Kermack-McKendrickModel.html

Epidemics



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

new infect @0.001:chan()
val recover = 0.03

let Recovered() =
  ?infect; Recovered()

and Susceptible() =
  ?infect; Infected()

and Infected() =
  do !infect; Infected()
  or ?infect; Infected()
  or delay@recover; Recovered()

run (200 of Susceptible() | 2 of Infected())
```



ODE

$S = ?i_{(t)};I$

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

$R = ?i_{(t)};R$

$S + I \rightarrow^{t} I + I$

$I + I \rightarrow^{t} I + I$

$I \rightarrow^{r} R$

$R + I \rightarrow^{t} R + I$

"useless" reactions

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

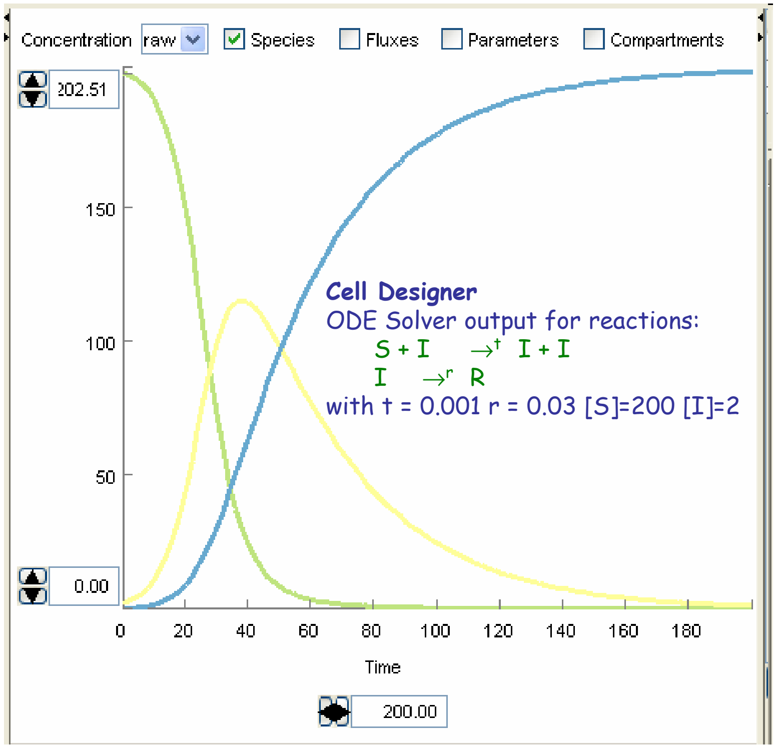
$[I]^\bullet = t[S][I]-r[I]$

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

Automata match the standard ODE model!

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

(the Kermack-McKendrick, or SIR model)



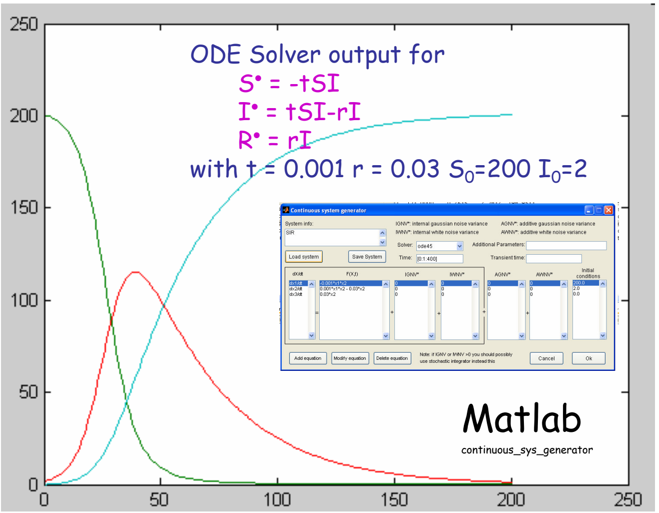
Cell Designer

ODE Solver output for reactions:

$S + I \rightarrow^{t} I + I$

$I \rightarrow^{r} R$

with t = 0.001 r = 0.03 [S]=200 [I]=2



ODE Solver output for

$S^\bullet = -tSI$

$I^\bullet = tSI-rI$

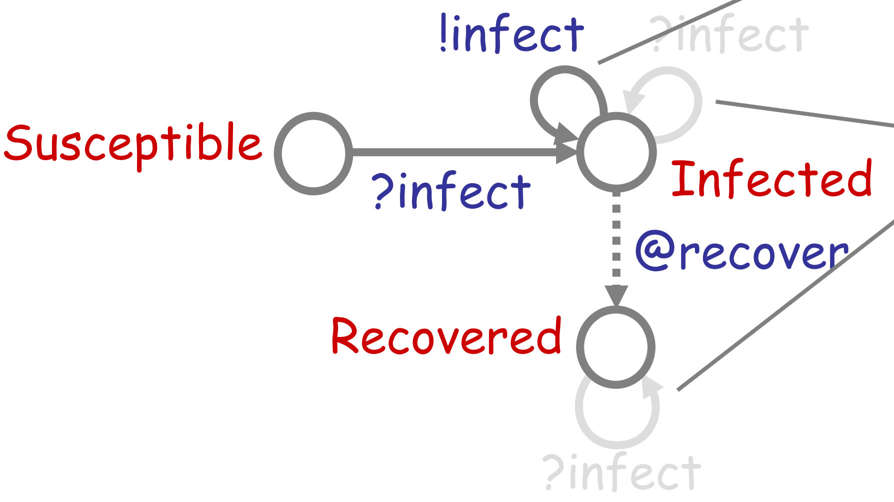
$R^\bullet = rI$

with t = 0.001 r = 0.03 S_0=200 I_0=2

Matlab

continuous_sys_generator

Simplified Model



not useless!

useless

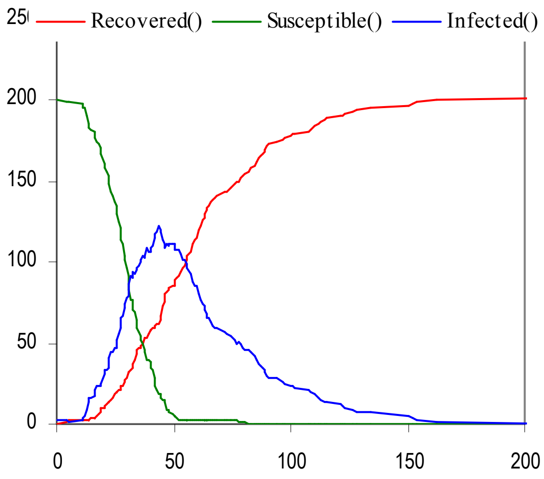
Not totally obvious that one *could* have simplified the automata model.

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

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

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

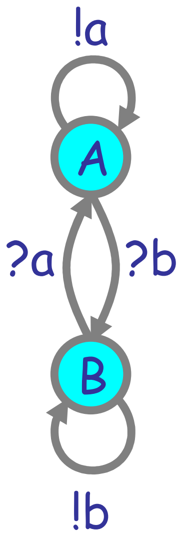
```
directive sample 500.0 1000
directive plot Recovered(); Susceptible(); Infected()

new infect @0.001:chan()
val recover = 0.03

let Recovered() =
  ()

and Susceptible() =
  ?infect; Infected()

and Infected() =
  do !infect; Infected()
  or delay@recover; Recovered()

run (200 of Susceptible() | 2 of Infected())
```



Same ODE, hence equivalent automata models.

Groupies ODEs

Groupies ODE

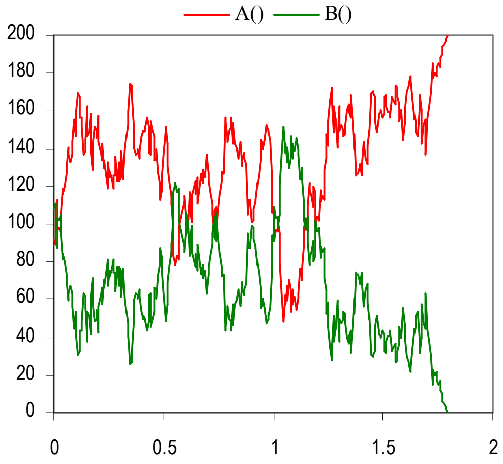


$$A = !a_{(r)};A \oplus ?b_{(r)};B$$
$$B = !b_{(r)};B \oplus ?a_{(r)};A$$

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

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

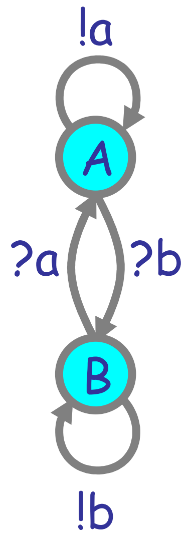
$$[A]^\bullet = 0$$
$$[B]^\bullet = 0$$



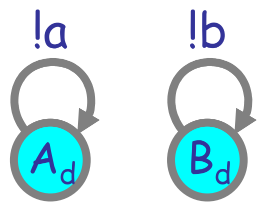
Wrong Answer?

ODE predicts stability $[A]^\bullet=0$ for any value of [A], while the stochastic system is stable only when [A] is either 0 or Max.

Doped Groupies ODE



Doping



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

$A_d = !a_{(r)};A_d$
$B_d = !b_{(r)};B_d$

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

$A+B_d \rightarrow^r B+B_d$
$B+A_d \rightarrow^r A+A_d$

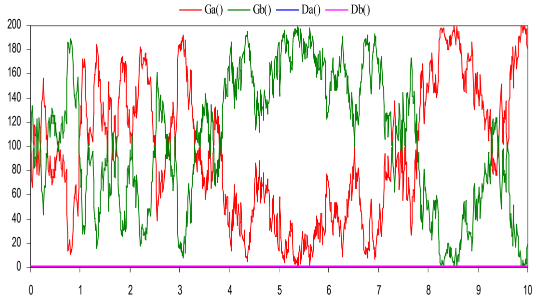
$[A]^\bullet = \cancel{r[A][B]-r[B][A]}-r[A][B_d]+r[B][A_d]$
$[B]^\bullet = \cancel{r[B][A]-r[A][B]}-r[B][A_d]+r[A][B_d]$

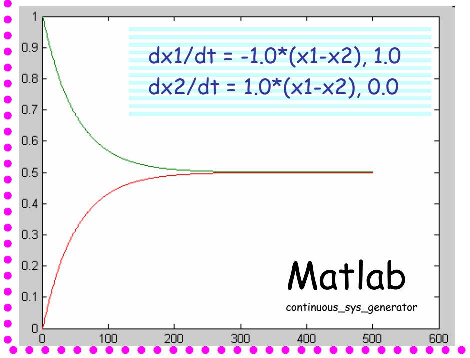
$[A_d]^\bullet = 0$
$[B_d]^\bullet = 0$

$[A_d],[B_d]$ are constant;
assume them both = k

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

At $[B]=0$: $[A]^\bullet=-rk[A]$, $[B]^\bullet=rk[A]$
At $[A]\approx[B]$: $[A]^\bullet=[B]^\bullet\approx 0$
At $[A]=[B]$: $[A]^\bullet=[B]^\bullet=0$



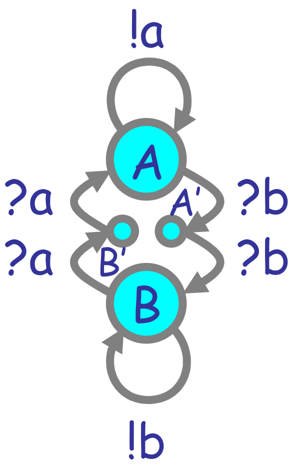

dx1/dt = -1.0*(x1-x2), 1.0
dx2/dt = 1.0*(x1-x2), 0.0

Wrong Answer?

ODE predicts converging stable equilibrium at [A]=[B] instead of the total chaos observed in the stochastic system!

For k=0 (no dope), predicts deadlock $[A]^\bullet=[B]^\bullet=0$ but at any value of [A], which is definitely not true in the stochastic system.

Hysteric Groupies ODE



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

$B = !b_{(r)};B \oplus ?a;B' \quad B' = ?a;A$

$A_d = !a_{(r)};A_d$

$B_d = !b_{(r)};B_d$

$A+B \rightarrow^r A+B' \quad A+B' \rightarrow^r A+A$

$B+A \rightarrow^r B+A' \quad B+A' \rightarrow^r B+B$

$A+B_d \rightarrow^r A'+B_d \quad A'+B_d \rightarrow^r B+B_d$

$B+A_d \rightarrow^r B'+A_d \quad B'+A_d \rightarrow^r A+B_d$

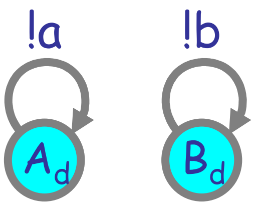
$[A]^\bullet = r[A][B']-r[B][A]-r[A][B_d]+r[B'][A_d]$
$[A']^\bullet = r[B][A]-r[B][A']+r[A][B_d]-r[A'][B_d]$
$[B]^\bullet = r[B][A']-r[A][B]-r[B][A_d]+r[A'][B_d]$
$[B']^\bullet = r[A][B]-r[A][B']+r[B][A_d]-r[B'][A_d]$

$[A_d]^\bullet = 0$
$[B_d]^\bullet = 0$

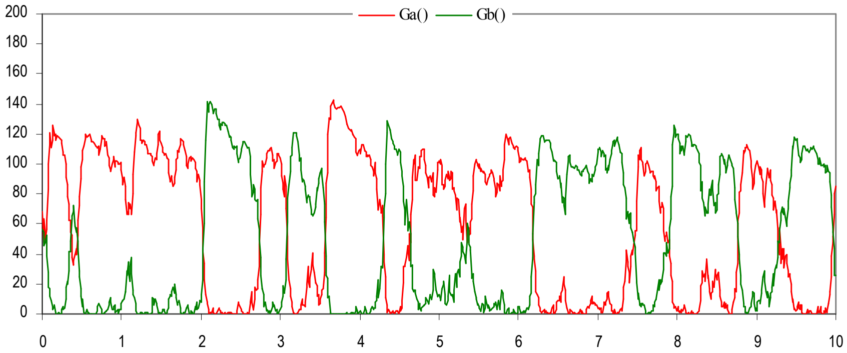
$[A]^\bullet = r[A][B']-r[B][A]-rk[A]+rk[B']$
$[A']^\bullet = r[B][A]-r[B][A']+rk[A]-rk[A']$
$[B]^\bullet = r[B][A']-r[A][B]-rk[B]+rk[A']$
$[B']^\bullet = r[A][B]-r[A][B']+rk[B]-rk[B']$

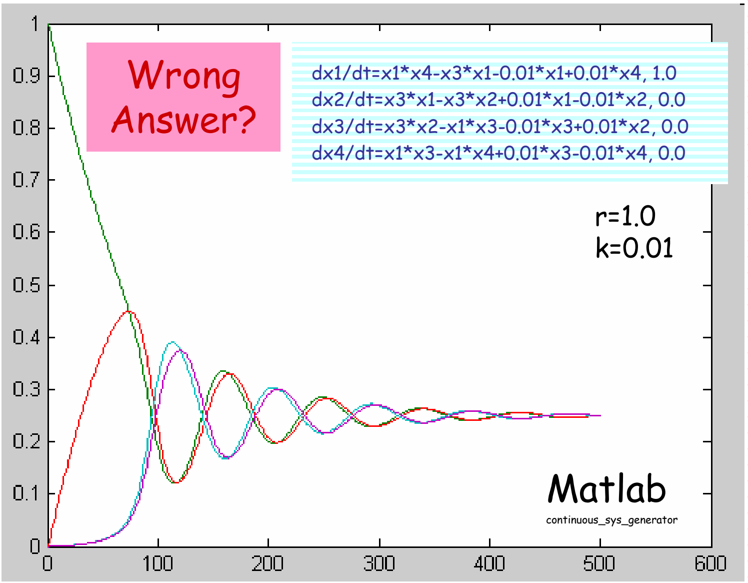
$[A_d],[B_d]$ are constant; assume them both = k

Doping



ODE predicts dampened oscillation, while the stochasic system keeps oscillating at max level.



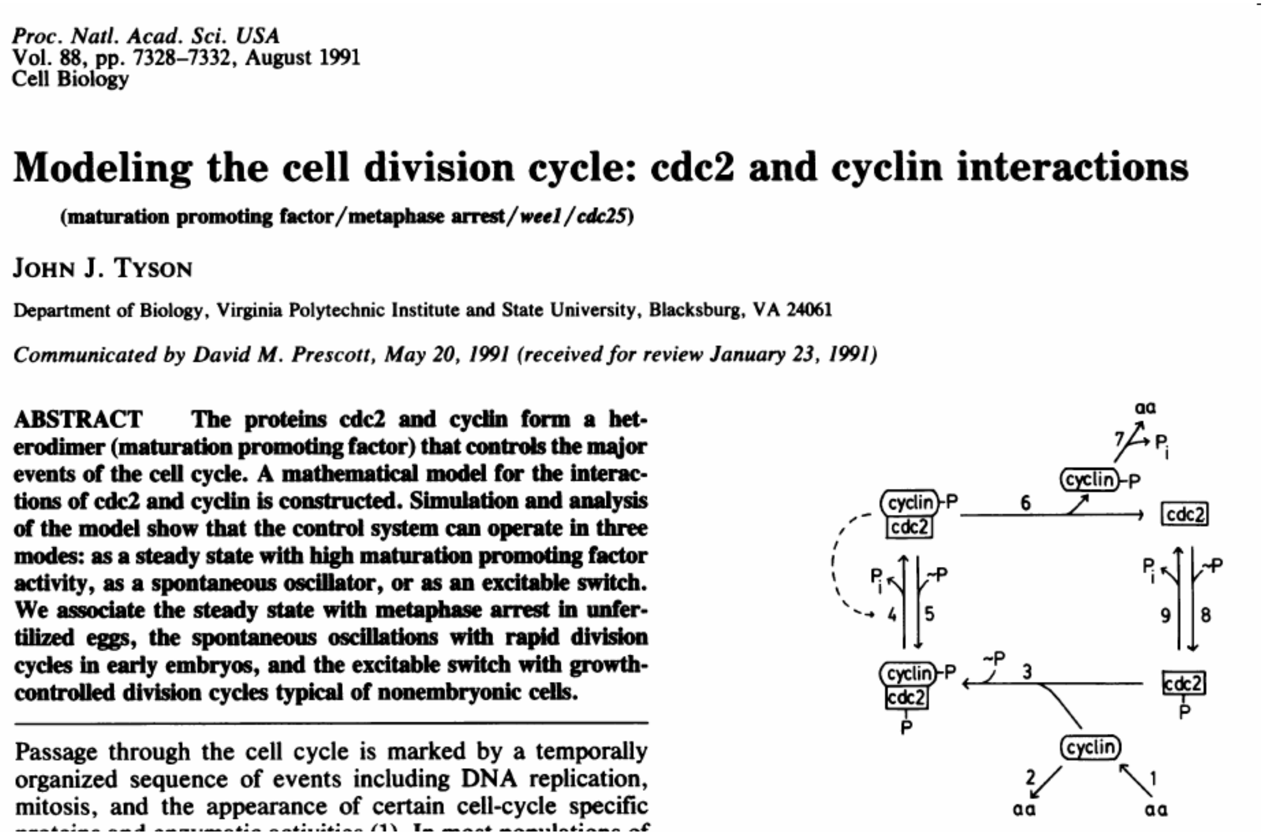


Wrong Answer?

dx1/dt=x1*x4-x3*x1-0.01*x1+0.01*x4, 1.0
dx2/dt=x3*x1-x3*x2+0.01*x1-0.01*x2, 0.0
dx3/dt=x3*x2-x1*x3-0.01*x3+0.01*x2, 0.0
dx4/dt=x1*x3-x1*x4+0.01*x3-0.01*x4, 0.0

r=1.0
k=0.01

Tyson Cell Cycle



Proc. Natl. Acad. Sci. USA
Vol. 88, pp. 7328–7332, August 1991
Cell Biology

Modeling the cell division cycle: cdc2 and cyclin interactions

(maturation promoting factor/metaphase arrest/*wee1*/*cdc25*)

JOHN J. TYSON

Department of Biology, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061

Communicated by David M. Prescott, May 20, 1991 (received for review January 23, 1991)

ABSTRACT **The proteins cdc2 and cyclin form a heterodimer (maturation promoting factor) that controls the major events of the cell cycle. A mathematical model for the interactions of cdc2 and cyclin is constructed. Simulation and analysis of the model show that the control system can operate in three modes: as a steady state with high maturation promoting factor activity, as a spontaneous oscillator, or as an excitable switch. We associate the steady state with metaphase arrest in unfertilized eggs, the spontaneous oscillations with rapid division cycles in early embryos, and the excitable switch with growth-controlled division cycles typical of nonembryonic cells.**

Passage through the cell cycle is marked by a temporally organized sequence of events including DNA replication, mitosis, and the appearance of certain cell-cycle specific proteins and enzymatic activities (1). In most populations of

Tyson Cell Cycle

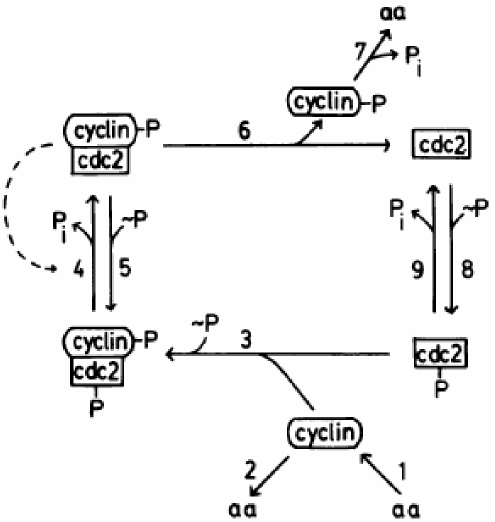


FIG. 1. The relationship between cyclin and cdc2 in the cell cycle. In step 1, cyclin is synthesized *de novo*. Newly synthesized cyclin may be unstable (step 2). Cyclin combines with cdc2-*P* (step 3) to form "preMPF." At some point after heterodimer formation, the cyclin subunit is phosphorylated. (Assuming phosphorylation is faster than dimerization, I write the two-step process as a single step, rate-limited by dimerization.) The cdc2 subunit is then dephosphorylated (step 4) to form "active MPF." In principle, the activation of MPF may be opposed by a protein kinase (step 5). Assuming that active MPF enhances the catalytic activity of the phosphatase (as indicated by the dashed arrow), I arrange that MPF activation is switched on in an autocatalytic fashion. Nuclear division is triggered when a sufficient quantity of MPF has been activated, but concurrently active MPF is destroyed by step 6. Breakdown of the MPF complex releases phosphorylated cyclin, which is subject to rapid proteolysis (step 7). Finally, the cdc2 subunit is phosphorylated (step 8, possibly reversed by step 9), and the cycle repeats itself. aa, amino acids; ~P, ATP; P_i, inorganic phosphate.

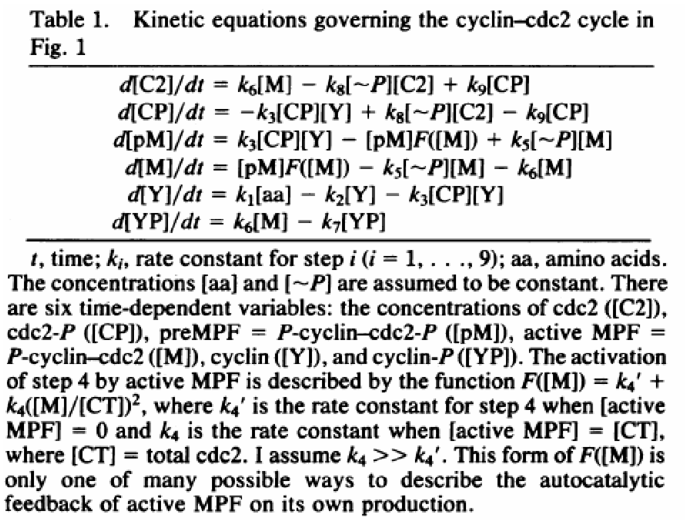


Table 1. Kinetic equations governing the cyclin–cdc2 cycle in Fig. 1

$$
\begin{aligned}
d[\mathrm{C2}]/dt &= k_6[\mathrm{M}] - k_8[\sim P][\mathrm{C2}] + k_9[\mathrm{CP}] \\
d[\mathrm{CP}]/dt &= -k_3[\mathrm{CP}][\mathrm{Y}] + k_8[\sim P][\mathrm{C2}] - k_9[\mathrm{CP}] \\
d[\mathrm{pM}]/dt &= k_3[\mathrm{CP}][\mathrm{Y}] - [\mathrm{pM}]F([\mathrm{M}]) + k_5[\sim P][\mathrm{M}] \\
d[\mathrm{M}]/dt &= [\mathrm{pM}]F([\mathrm{M}]) - k_5[\sim P][\mathrm{M}] - k_6[\mathrm{M}] \\
d[\mathrm{Y}]/dt &= k_1[\mathrm{aa}] - k_2[\mathrm{Y}] - k_3[\mathrm{CP}][\mathrm{Y}] \\
d[\mathrm{YP}]/dt &= k_6[\mathrm{M}] - k_7[\mathrm{YP}]
\end{aligned}
$$

t, time; k_i, rate constant for step i ($i = 1, \ldots, 9$); aa, amino acids. The concentrations [aa] and [$\sim P$] are assumed to be constant. There are six time-dependent variables: the concentrations of cdc2 ([C2]), cdc2-*P* ([CP]), preMPF = *P*-cyclin–cdc2-*P* ([pM]), active MPF = *P*-cyclin–cdc2 ([M]), cyclin ([Y]), and cyclin-*P* ([YP]). The activation of step 4 by active MPF is described by the function $F([\mathrm{M}]) = k_4' + k_4([\mathrm{M}]/[\mathrm{CT}])^2$, where k_4' is the rate constant for step 4 when [active MPF] = 0 and k_4 is the rate constant when [active MPF] = [CT], where [CT] = total cdc2. I assume $k_4 >> k_4'$. This form of $F([\mathrm{M}])$ is only one of many possible ways to describe the autocatalytic feedback of active MPF on its own production.

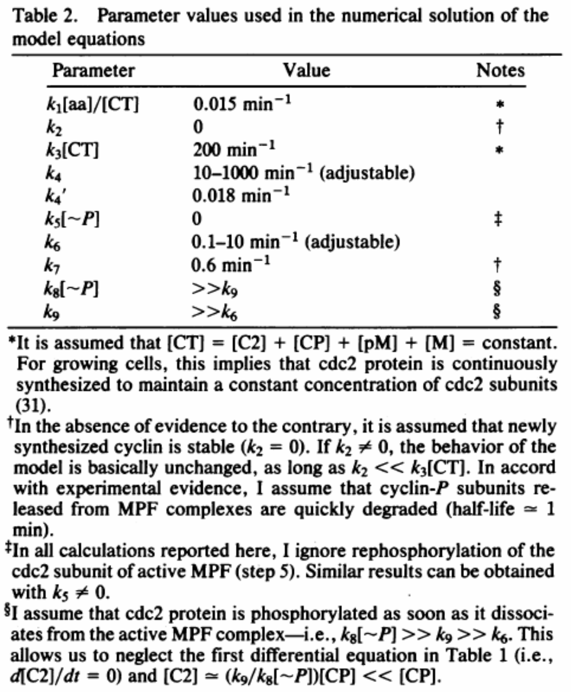


Table 2. Parameter values used in the numerical solution of the model equations

Parameter	Value	Notes
$k_1[\mathrm{aa}]/[\mathrm{CT}]$	0.015 min^{-1}	*
k_2	0	†
$k_3[\mathrm{CT}]$	200 min^{-1}	*
k_4	10–1000 min^{-1} (adjustable)	
k_4'	0.018 min^{-1}	
$k_5[\sim P]$	0	‡
k_6	0.1–10 min^{-1} (adjustable)	
k_7	0.6 min^{-1}	†
$k_8[\sim P]$	$>>k_9$	§
k_9	$>>k_6$	§

*It is assumed that [CT] = [C2] + [CP] + [pM] + [M] = constant. For growing cells, this implies that cdc2 protein is continuously synthesized to maintain a constant concentration of cdc2 subunits (31).

†In the absence of evidence to the contrary, it is assumed that newly synthesized cyclin is stable ($k_2 = 0$). If $k_2 \neq 0$, the behavior of the model is basically unchanged, as long as $k_2 << k_3[\mathrm{CT}]$. In accord with experimental evidence, I assume that cyclin-*P* subunits released from MPF complexes are quickly degraded (half-life ≃ 1 min).

‡In all calculations reported here, I ignore rephosphorylation of the cdc2 subunit of active MPF (step 5). Similar results can be obtained with $k_5 \neq 0$.

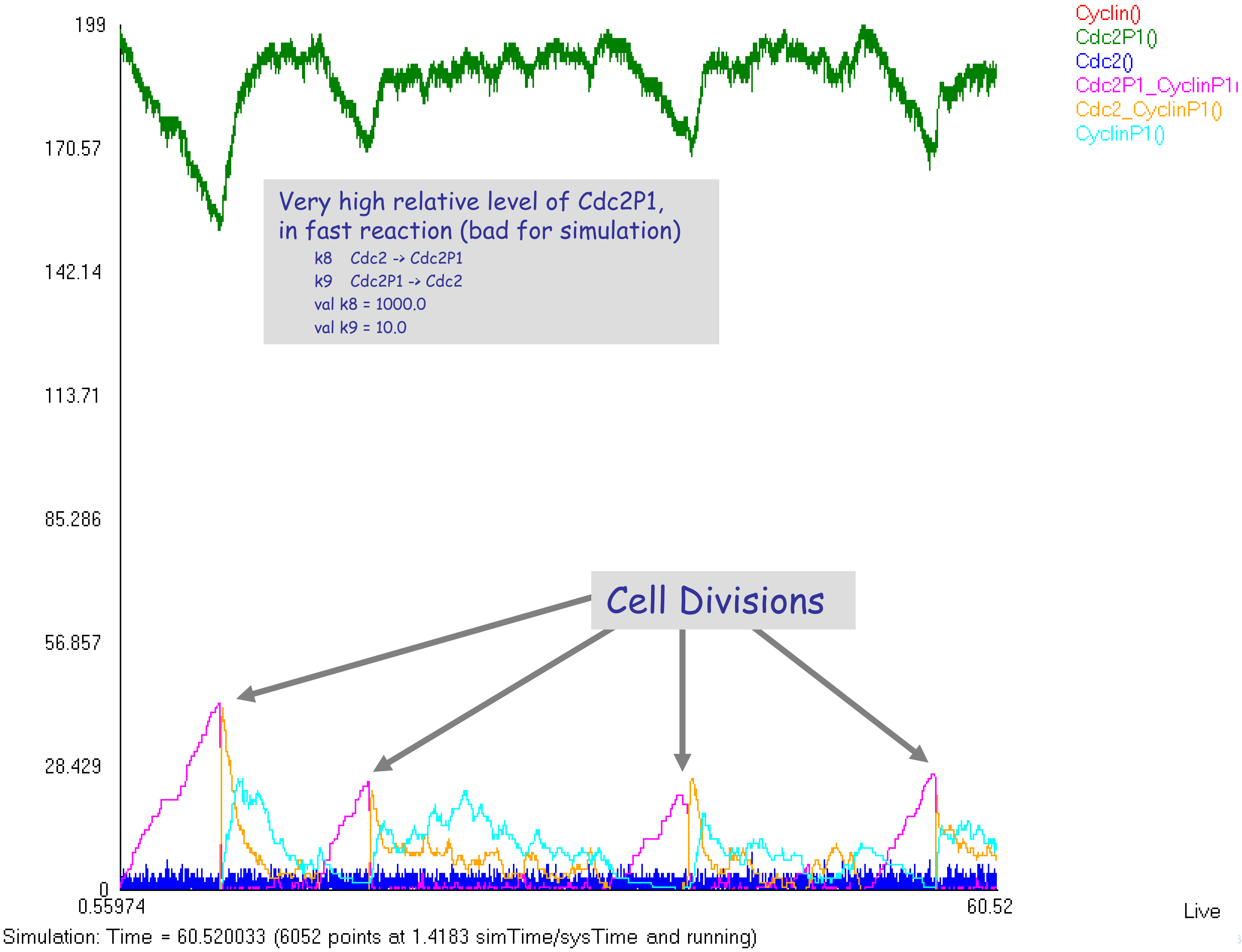
§I assume that cdc2 protein is phosphorylated as soon as it dissociates from the active MPF complex—i.e., $k_8[\sim P] >> k_9 >> k_6$. This allows us to neglect the first differential equation in Table 1 (i.e., $d[\mathrm{C2}]/dt = 0$) and $[\mathrm{C2}] \simeq (k_9/k_8[\sim P])[\mathrm{CP}] << [\mathrm{CP}]$.

The Tyson Cell Cycle in SPiM

```
directive sample 10.0 1000
directive plot Cyclin(); Cdc2P1(); Cdc2();
  Cdc2P1_CyclinP1(); Cdc2_CyclinP1(); CyclinP1()

val factor = 200.0  (* Scaling Factor *)

val k1 = 5.0 (* 0.015 cyclin production cranked up *)
val k2 = 0.0
val k3 = 200.0/factor
val k4p = 0.018
val k4 = 200.0/factor
val k5 = 0.0
val k6 = 1.0
val k7 = 0.6
val k8 = 1000.0
val k9 = 10.0

(*    THE REACTIONS
k1    0 -> Cyclin
k2    Cyclin -> 0
k3    Cyclin + Cdc2P1 -> Cdc2P1_CyclinP1

k4p  Cdc2P1_CyclinP1 -> Cdc2_CyclinP1
k4    Cdc2P1_CyclinP1 + Cdc2_CyclinP1 -> 2* Cdc2_CyclinP1
k5    Cdc2_CyclinP1 -> Cdc2P1_CyclinP1

k6    Cdc2_CyclinP1 -> CyclinP1 + Cdc2
k7    CyclinP1 -> 0

k8    Cdc2 -> Cdc2P1
k9    Cdc2P1 -> Cdc2
*)
```

```
new c3@k3:chan
new c4@k4:chan

let genCyclin() = delay@k1; (Cyclin() | genCyclin())

and Cyclin() =
  do delay@k2; ()
  or ?c3; Cdc2P1_CyclinP1()

and Cdc2P1() =
  do !c3; ()
  or delay@k9; Cdc2()

and Cdc2() =
  delay@k8; Cdc2P1()

and Cdc2P1_CyclinP1() =
  do delay@k4p; Cdc2_CyclinP1()
  or ?c4; Cdc2_CyclinP1()

and Cdc2_CyclinP1() =
  do !c4; Cdc2_CyclinP1()
  or delay@k5; Cdc2P1_CyclinP1()
  or delay@k6; (CyclinP1() | Cdc2())

and CyclinP1() =
  delay@k7; ()

run genCyclin()
run 200 of Cdc2P1()
```

SPiM Simulation



Very high relative level of Cdc2P1, in fast reaction (bad for simulation)

```
k8    Cdc2 -> Cdc2P1
k9    Cdc2P1 -> Cdc2
val k8 = 1000.0
val k9 = 10.0
```

Cell Divisions

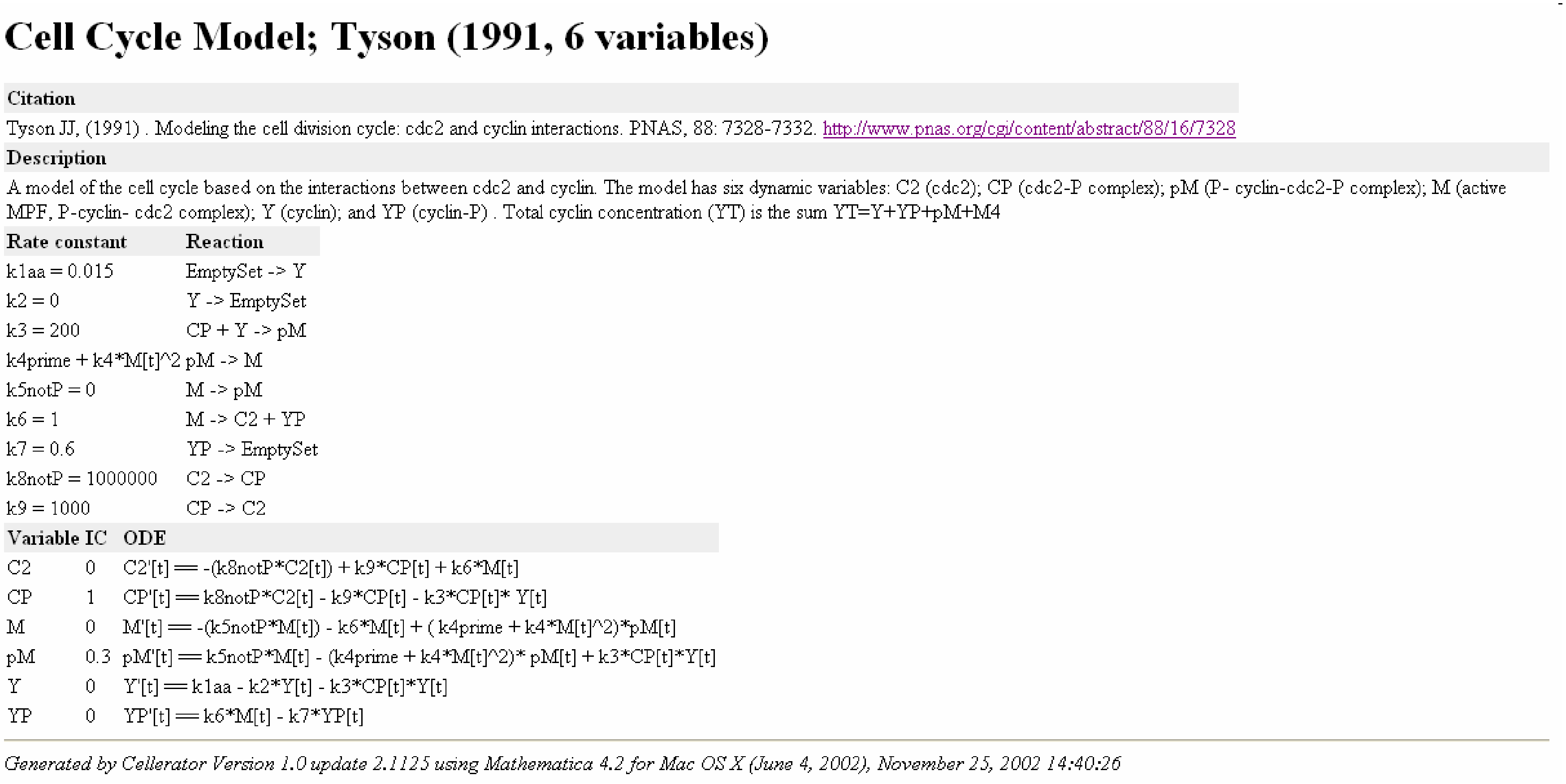
```
Simulation: Time = 60.520033 (6052 points at 1.4183 simTime/sysTime and running)
```

The Tyson Cell Cycle in BIOCHAM

```
%Description
%A model of the cell cycle based on the interactions between cdc2 and cyclin.

%present(Cdc2,0.39).
%present(Cdc2~{p1},0.0001).
%present(Cyclin,0.0001).
%present(Cdc2~{p1}-Cyclin~{p1},0.0001).
%present(Cdc2-Cyclin~{p1},0.0001).
%present(Cyclin~{p1},0.0001).

present(Cdc2,1).
absent(Cdc2~{p1}).
absent(Cyclin).
absent(Cdc2-Cyclin~{p1}).
absent(Cdc2~{p1}-Cyclin~{p1}).
absent(Cyclin~{p1}).

k1                 for _=>Cyclin.
k2*[Cyclin]             for Cyclin=>_.

k3*[Cyclin]*[Cdc2~{p1}]for Cyclin+Cdc2~{p1} => Cdc2~{p1}-Cyclin~{p1}.

k4p*[Cdc2~{p1}-Cyclin~{p1}] for Cdc2~{p1}-Cyclin~{p1} => Cdc2-Cyclin~{p1}.
k4*([Cdc2-Cyclin~{p1}])^2*[Cdc2~{p1}-Cyclin~{p1}]
            for Cdc2~{p1}-Cyclin~{p1} =[Cdc2-Cyclin~{p1}]=> Cdc2-Cyclin~{p1}.
k5*[Cdc2-Cyclin~{p1}]   for Cdc2-Cyclin~{p1} => Cdc2~{p1}-Cyclin~{p1}.

k6*[Cdc2-Cyclin~{p1}]   for Cdc2-Cyclin~{p1} => Cyclin~{p1}+Cdc2.
k7*[Cyclin~{p1}]        for Cyclin~{p1} =>_.

k8*[Cdc2]               for Cdc2 => Cdc2~{p1}.
k9*[Cdc2~{p1}]          for Cdc2~{p1} => Cdc2.

%Cdc2-Cyclin~{p1}=>Cdc2~{p1}.

macro(YT,[Cyclin]+[Cyclin~{p1}]+[Cdc2~{p1}-Cyclin~{p1}]+[Cdc2-Cyclin~{p1}]).
macro(CT,[Cdc2]+[Cdc2~{p1}]+[Cdc2~{p1}-Cyclin~{p1}]+[Cdc2-Cyclin~{p1}]).
macro(ratio,YT/CT).
```

```
parameter(k1,0.015).
parameter(k2,0.015).
parameter(k3,200).
parameter(k4p,0.018).
parameter(k4,180).
parameter(k5,0).
parameter(k6,1).
parameter(k7,0.6).
parameter(k8,100).
parameter(k9,100).
```

The Tyson Cell Cycle in Cellerator



Cell Cycle Model; Tyson (1991, 6 variables)

Citation

Tyson JJ, (1991) . Modeling the cell division cycle: cdc2 and cyclin interactions. PNAS, 88: 7328-7332. http://www.pnas.org/cgi/content/abstract/88/16/7328

Description

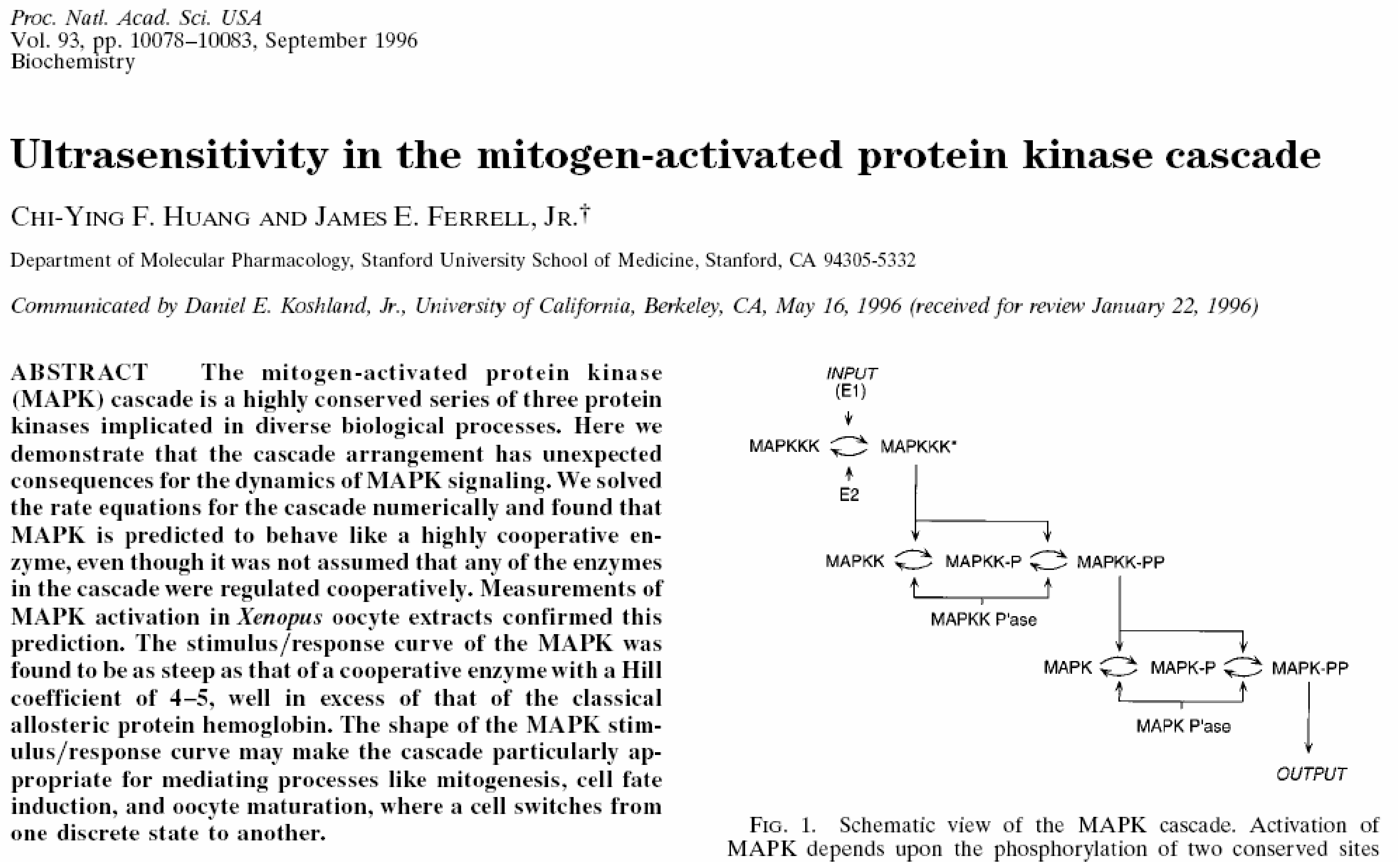
A model of the cell cycle based on the interactions between cdc2 and cyclin. The model has six dynamic variables: C2 (cdc2); CP (cdc2-P complex); pM (P- cyclin-cdc2-P complex); M (active MPF, P-cyclin- cdc2 complex); Y (cyclin); and YP (cyclin-P) . Total cyclin concentration (YT) is the sum YT=Y+YP+pM+M4

Rate constant	Reaction
k1aa = 0.015	EmptySet -> Y
k2 = 0	Y -> EmptySet
k3 = 200	CP + Y -> pM
k4prime + k4*M[t]^2	pM -> M
k5notP = 0	M -> pM
k6 = 1	M -> C2 + YP
k7 = 0.6	YP -> EmptySet
k8notP = 1000000	C2 -> CP
k9 = 1000	CP -> C2

Variable	IC	ODE
C2	0	C2'[t] == -(k8notP*C2[t]) + k9*CP[t] + k6*M[t]
CP	1	CP'[t] == k8notP*C2[t] - k9*CP[t] - k3*CP[t]* Y[t]
M	0	M'[t] == -(k5notP*M[t]) - k6*M[t] + (k4prime + k4*M[t]^2)*pM[t]
pM	0.3	pM'[t] == k5notP*M[t] - (k4prime + k4*M[t]^2)* pM[t] + k3*CP[t]*Y[t]
Y	0	Y'[t] == k1aa - k2*Y[t] - k3*CP[t]*Y[t]
YP	0	YP'[t] == k6*M[t] - k7*YP[t]

Generated by Cellerator Version 1.0 update 2.1125 using Mathematica 4.2 for Mac OS X (June 4, 2002), November 25, 2002 14:40:26

MAPK Cascade



Proc. Natl. Acad. Sci. USA
Vol. 93, pp. 10078–10083, September 1996
Biochemistry

Ultrasensitivity in the mitogen-activated protein kinase cascade

CHI-YING F. HUANG AND JAMES E. FERRELL, JR.[†]

Department of Molecular Pharmacology, Stanford University School of Medicine, Stanford, CA 94305-5332

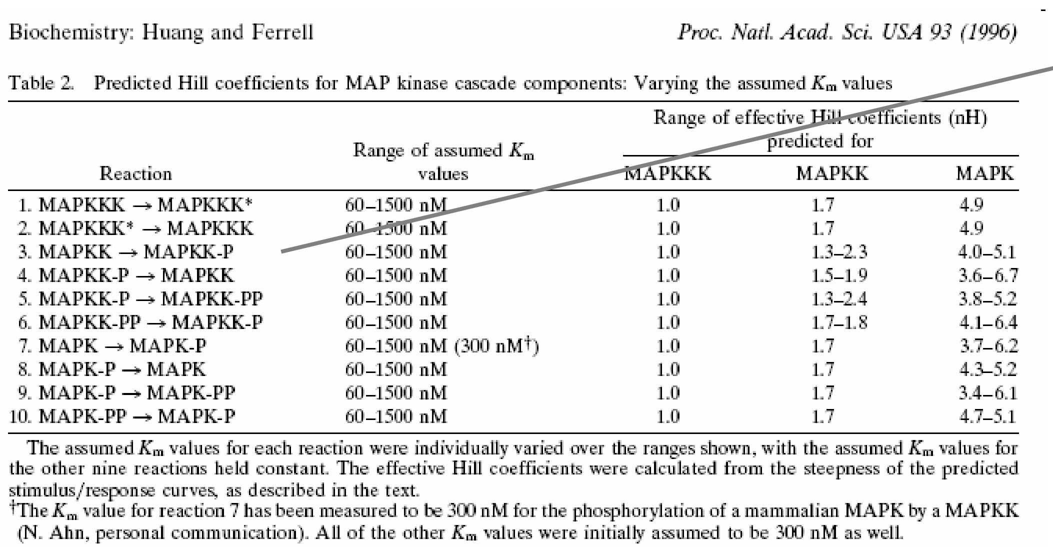
Communicated by Daniel E. Koshland, Jr., University of California, Berkeley, CA, May 16, 1996 (received for review January 22, 1996)

ABSTRACT The mitogen-activated protein kinase (MAPK) cascade is a highly conserved series of three protein kinases implicated in diverse biological processes. Here we demonstrate that the cascade arrangement has unexpected consequences for the dynamics of MAPK signaling. We solved the rate equations for the cascade numerically and found that MAPK is predicted to behave like a highly cooperative enzyme, even though it was not assumed that any of the enzymes in the cascade were regulated cooperatively. Measurements of MAPK activation in *Xenopus* oocyte extracts confirmed this prediction. The stimulus/response curve of the MAPK was found to be as steep as that of a cooperative enzyme with a Hill coefficient of 4–5, well in excess of that of the classical allosteric protein hemoglobin. The shape of the MAPK stimulus/response curve may make the cascade particularly appropriate for mediating processes like mitogenesis, cell fate induction, and oocyte maturation, where a cell switches from one discrete state to another.

FIG. 1. Schematic view of the MAPK cascade. Activation of MAPK depends upon the phosphorylation of two conserved sites

MAPK Cascade

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.



Biochemistry: Huang and Ferrell *Proc. Natl. Acad. Sci. USA 93 (1996)*

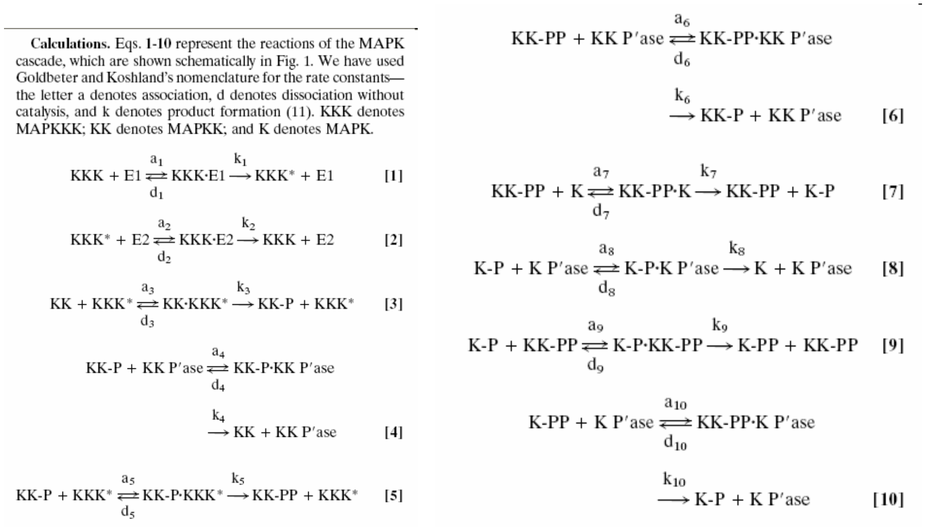
Table 2. Predicted Hill coefficients for MAP kinase cascade components: Varying the assumed K_m values

Reaction	Range of assumed K_m values	Range of effective Hill coefficients (nH) predicted for		
		MAPKKK	MAPKK	MAPK
1. MAPKKK → MAPKKK*	60–1500 nM	1.0	1.7	4.9
2. MAPKKK* → MAPKKK	60–1500 nM	1.0	1.7	4.9
3. MAPKK → MAPKK-P	60–1500 nM	1.0	1.3–2.3	4.0–5.1
4. MAPKK-P → MAPKK	60–1500 nM	1.0	1.5–1.9	3.6–6.7
5. MAPKK-P → MAPKK-PP	60–1500 nM	1.0	1.3–2.4	3.8–5.2
6. MAPKK-PP → MAPKK-P	60–1500 nM	1.0	1.7–1.8	4.1–6.4
7. MAPK → MAPK-P	60–1500 nM (300 nM†)	1.0	1.7	3.7–6.2
8. MAPK-P → MAPK	60–1500 nM	1.0	1.7	4.3–5.2
9. MAPK-P → MAPK-PP	60–1500 nM	1.0	1.7	3.4–6.1
10. MAPK-PP → MAPK-P	60–1500 nM	1.0	1.7	4.7–5.1

The assumed K_m values for each reaction were individually varied over the ranges shown, with the assumed K_m values for the other nine reactions held constant. The effective Hill coefficients were calculated from the steepness of the predicted stimulus/response curves, as described in the text.

†The K_m value for reaction 7 has been measured to be 300 nM for the phosphorylation of a mammalian MAPK by a MAPKK (N. Ahn, personal communication). All of the other K_m values were initially assumed to be 300 nM as well.

10 chemical reactions



Calculations. Eqs. **1-10** represent the reactions of the MAPK cascade, which are shown schematically in Fig. 1. We have used Goldbeter and Koshland's nomenclature for the rate constants—the letter a denotes association, d denotes dissociation without catalysis, and k denotes product formation (11). KKK denotes MAPKKK; KK denotes MAPKK; and K denotes MAPK.

$$\mathrm{KKK + E1} \underset{d_1}{\overset{a_1}{\rightleftharpoons}} \mathrm{KKK{\cdot}E1} \xrightarrow{k_1} \mathrm{KKK^* + E1} \qquad [1]$$

$$\mathrm{KKK^* + E2} \underset{d_2}{\overset{a_2}{\rightleftharpoons}} \mathrm{KKK{\cdot}E2} \xrightarrow{k_2} \mathrm{KKK + E2} \qquad [2]$$

$$\mathrm{KK + KKK^*} \underset{d_3}{\overset{a_3}{\rightleftharpoons}} \mathrm{KK{\cdot}KKK^*} \xrightarrow{k_3} \mathrm{KK\text{-}P + KKK^*} \qquad [3]$$

$$\mathrm{KK\text{-}P + KK\ P'ase} \underset{d_4}{\overset{a_4}{\rightleftharpoons}} \mathrm{KK\text{-}P{\cdot}KK\ P'ase} \xrightarrow{k_4} \mathrm{KK + KK\ P'ase} \qquad [4]$$

$$\mathrm{KK\text{-}P + KKK^*} \underset{d_5}{\overset{a_5}{\rightleftharpoons}} \mathrm{KK\text{-}P{\cdot}KKK^*} \xrightarrow{k_5} \mathrm{KK\text{-}PP + KKK^*} \qquad [5]$$

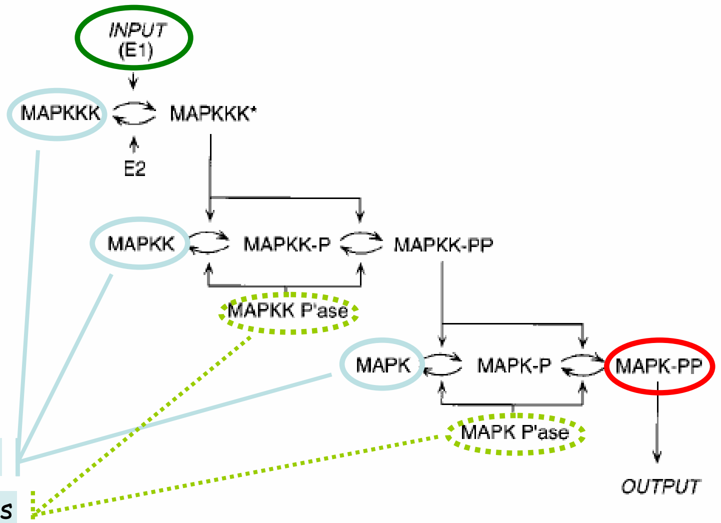
$$\mathrm{KK\text{-}PP + KK\ P'ase} \underset{d_6}{\overset{a_6}{\rightleftharpoons}} \mathrm{KK\text{-}PP{\cdot}KK\ P'ase} \xrightarrow{k_6} \mathrm{KK\text{-}P + KK\ P'ase} \qquad [6]$$

$$\mathrm{KK\text{-}PP + K} \underset{d_7}{\overset{a_7}{\rightleftharpoons}} \mathrm{KK\text{-}PP{\cdot}K} \xrightarrow{k_7} \mathrm{KK\text{-}PP + K\text{-}P} \qquad [7]$$

$$\mathrm{K\text{-}P + K\ P'ase} \underset{d_8}{\overset{a_8}{\rightleftharpoons}} \mathrm{K\text{-}P{\cdot}K\ P'ase} \xrightarrow{k_8} \mathrm{K + K\ P'ase} \qquad [8]$$

$$\mathrm{K\text{-}P + KK\text{-}PP} \underset{d_9}{\overset{a_9}{\rightleftharpoons}} \mathrm{K\text{-}P{\cdot}KK\text{-}PP} \xrightarrow{k_9} \mathrm{K\text{-}PP + KK\text{-}PP} \qquad [9]$$

$$\mathrm{K\text{-}PP + K\ P'ase} \underset{d_{10}}{\overset{a_{10}}{\rightleftharpoons}} \mathrm{KK\text{-}PP{\cdot}K\ P'ase} \xrightarrow{k_{10}} \mathrm{K\text{-}P + K\ P'ase} \qquad [10]$$

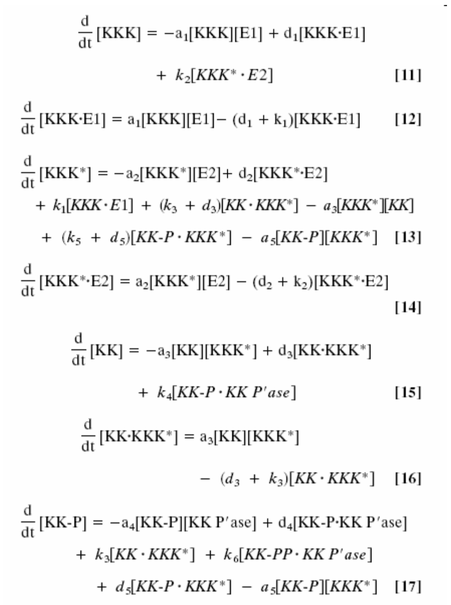


Reservoirs

Back Enzymes

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

As 18 Ordinary Differential Equations
Plus 7 conservation equations



$$\frac{d}{dt}[\text{KKK}] = -a_1[\text{KKK}][\text{E1}] + d_1[\text{KKK·E1}] + k_2[KKK^* \cdot E2] \qquad [11]$$

$$\frac{d}{dt}[\text{KKK·E1}] = a_1[\text{KKK}][\text{E1}] - (d_1 + k_1)[\text{KKK·E1}] \qquad [12]$$

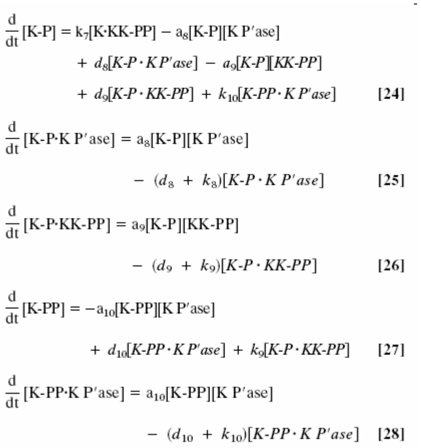
$$\frac{d}{dt}[\text{KKK}^*] = -a_2[\text{KKK}^*][\text{E2}] + d_2[\text{KKK}^*\text{·E2}] + k_1[KKK \cdot E1] + (k_3 + d_3)[KK \cdot KKK^*] - a_3[KKK^*][KK] + (k_5 + d_5)[KK\text{-}P \cdot KKK^*] - a_5[KK\text{-}P][KKK^*] \qquad [13]$$

$$\frac{d}{dt}[\text{KKK}^*\text{·E2}] = a_2[\text{KKK}^*][\text{E2}] - (d_2 + k_2)[\text{KKK}^*\text{·E2}] \qquad [14]$$

$$\frac{d}{dt}[\text{KK}] = -a_3[\text{KK}][\text{KKK}^*] + d_3[\text{KK·KKK}^*] + k_4[KK\text{-}P \cdot KK\ P'ase] \qquad [15]$$

$$\frac{d}{dt}[\text{KK·KKK}^*] = a_3[\text{KK}][\text{KKK}^*] - (d_3 + k_3)[KK \cdot KKK^*] \qquad [16]$$

$$\frac{d}{dt}[\text{KK-P}] = -a_4[\text{KK-P}][\text{KK P}'\text{ase}] + d_4[\text{KK-P·KK P}'\text{ase}] + k_3[KK \cdot KKK^*] + k_6[KK\text{-}PP \cdot KK\ P'ase] + d_5[KK\text{-}P \cdot KKK^*] - a_5[KK\text{-}P][KKK^*] \qquad [17]$$



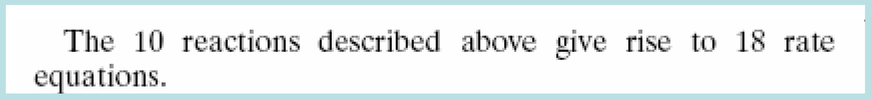
$$\frac{d}{dt}[\text{K-P}] = k_7[\text{K·KK-PP}] - a_8[\text{K-P}][\text{K P}'\text{ase}] + d_8[K\text{-}P \cdot K\,P'ase] - a_9[K\text{-}P][KK\text{-}PP] + d_9[K\text{-}P \cdot KK\text{-}PP] + k_{10}[K\text{-}PP \cdot K\,P'ase] \qquad [24]$$

$$\frac{d}{dt}[\text{K-P·K P}'\text{ase}] = a_8[\text{K-P}][\text{K P}'\text{ase}] - (d_8 + k_8)[K\text{-}P \cdot K\ P'ase] \qquad [25]$$

$$\frac{d}{dt}[\text{K-P·KK-PP}] = a_9[\text{K-P}][\text{KK-PP}] - (d_9 + k_9)[K\text{-}P \cdot KK\text{-}PP] \qquad [26]$$

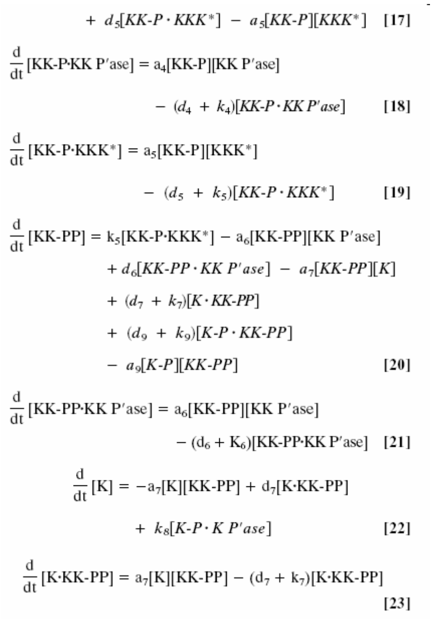
$$\frac{d}{dt}[\text{K-PP}] = -a_{10}[\text{K-PP}][\text{K P}'\text{ase}] + d_{10}[K\text{-}PP \cdot K\,P'ase] + k_9[K\text{-}P \cdot KK\text{-}PP] \qquad [27]$$

$$\frac{d}{dt}[\text{K-PP·K P}'\text{ase}] = a_{10}[\text{K-PP}][\text{K P}'\text{ase}] - (d_{10} + k_{10})[K\text{-}PP \cdot K\ P'ase] \qquad [28]$$



The 10 reactions described above give rise to 18 rate equations.

One equation for each species (8) and complex (10), but not for constant concentration enzymes (4)



$$+ d_5[KK\text{-}P \cdot KKK^*] - a_5[KK\text{-}P][KKK^*] \qquad [17]$$

$$\frac{d}{dt}[\text{KK-P·KK P}'\text{ase}] = a_4[\text{KK-P}][\text{KK P}'\text{ase}] - (d_4 + k_4)[KK\text{-}P \cdot KK\,P'ase] \qquad [18]$$

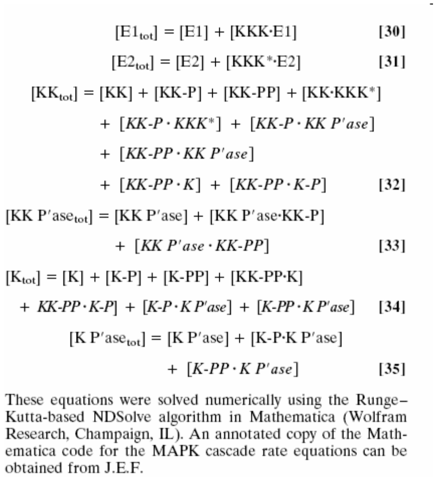
$$\frac{d}{dt}[\text{KK-P·KKK}^*] = a_5[\text{KK-P}][\text{KKK}^*] - (d_5 + k_5)[KK\text{-}P \cdot KKK^*] \qquad [19]$$

$$\frac{d}{dt}[\text{KK-PP}] = k_5[\text{KK-P·KKK}^*] - a_6[\text{KK-PP}][\text{KK P}'\text{ase}] + d_6[KK\text{-}PP \cdot KK\ P'ase] - a_7[KK\text{-}PP][K] + (d_7 + k_7)[K \cdot KK\text{-}PP] + (d_9 + k_9)[K\text{-}P \cdot KK\text{-}PP] - a_9[K\text{-}P][KK\text{-}PP] \qquad [20]$$

$$\frac{d}{dt}[\text{KK-PP·KK P}'\text{ase}] = a_6[\text{KK-PP}][\text{KK P}'\text{ase}] - (d_6 + K_6)[\text{KK-PP·KK P}'\text{ase}] \qquad [21]$$

$$\frac{d}{dt}[\text{K}] = -a_7[\text{K}][\text{KK-PP}] + d_7[\text{K·KK-PP}] + k_8[K\text{-}P \cdot K\ P'ase] \qquad [22]$$

$$\frac{d}{dt}[\text{K·KK-PP}] = a_7[\text{K}][\text{KK-PP}] - (d_7 + k_7)[\text{K·KK-PP}] \qquad [23]$$



$$[\text{E1}_{\text{tot}}] = [\text{E1}] + [\text{KKK·E1}] \qquad [30]$$

$$[\text{E2}_{\text{tot}}] = [\text{E2}] + [\text{KKK}^*\text{·E2}] \qquad [31]$$

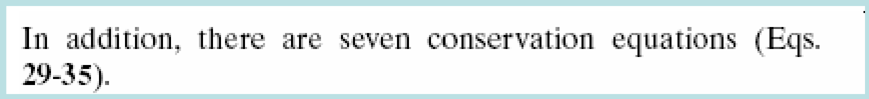
$$[\text{KK}_{\text{tot}}] = [\text{KK}] + [\text{KK-P}] + [\text{KK-PP}] + [\text{KK·KKK}^*] + [KK\text{-}P \cdot KKK^*] + [KK\text{-}P \cdot KK\ P'ase] + [KK\text{-}PP \cdot KK\ P'ase] + [KK\text{-}PP \cdot K] + [KK\text{-}PP \cdot K\text{-}P] \qquad [32]$$

$$[\text{KK P}'\text{ase}_{\text{tot}}] = [\text{KK P}'\text{ase}] + [\text{KK P}'\text{ase·KK-P}] + [KK\ P'ase \cdot KK\text{-}PP] \qquad [33]$$

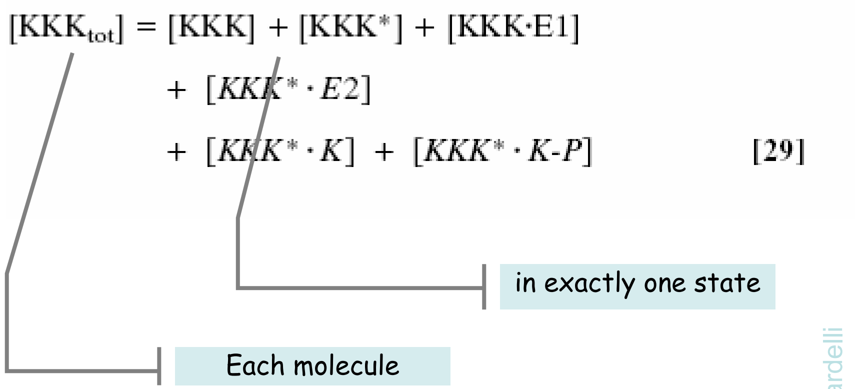
$$[\text{K}_{\text{tot}}] = [\text{K}] + [\text{K-P}] + [\text{K-PP}] + [\text{KK-PP·K}] + KK\text{-}PP \cdot K\text{-}P] + [K\text{-}P \cdot K\,P'ase] + [K\text{-}PP \cdot K\,P'ase] \qquad [34]$$

$$[\text{K P}'\text{ase}_{\text{tot}}] = [\text{K P}'\text{ase}] + [\text{K-P·K P}'\text{ase}] + [K\text{-}PP \cdot K\ P'ase] \qquad [35]$$

These equations were solved numerically using the Runge–Kutta-based NDSolve algorithm in Mathematica (Wolfram Research, Champaign, IL). An annotated copy of the Mathematica code for the MAPK cascade rate equations can be obtained from J.E.F.



In addition, there are seven conservation equations (Eqs. 29-35).

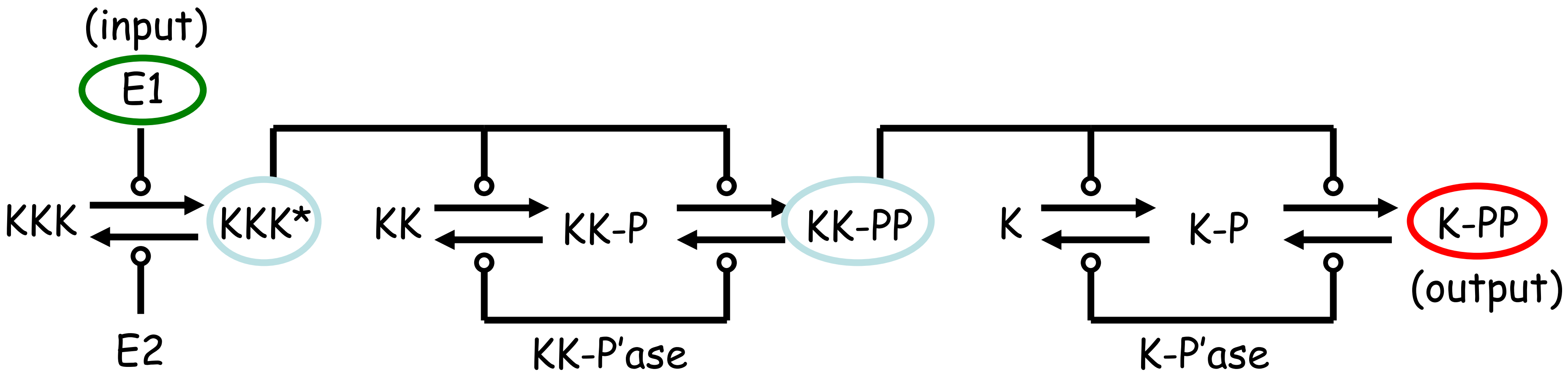


$$[\text{KKK}_{\text{tot}}] = [\text{KKK}] + [\text{KKK}^*] + [\text{KKK·E1}] + [KKK^* \cdot E2] + [KKK^* \cdot K] + [KKK^* \cdot K\text{-}P] \qquad [29]$$

in exactly one state

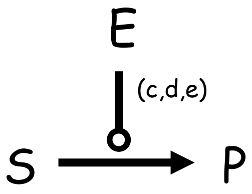
Each molecule

The Circuit



Enzymatic Reactions

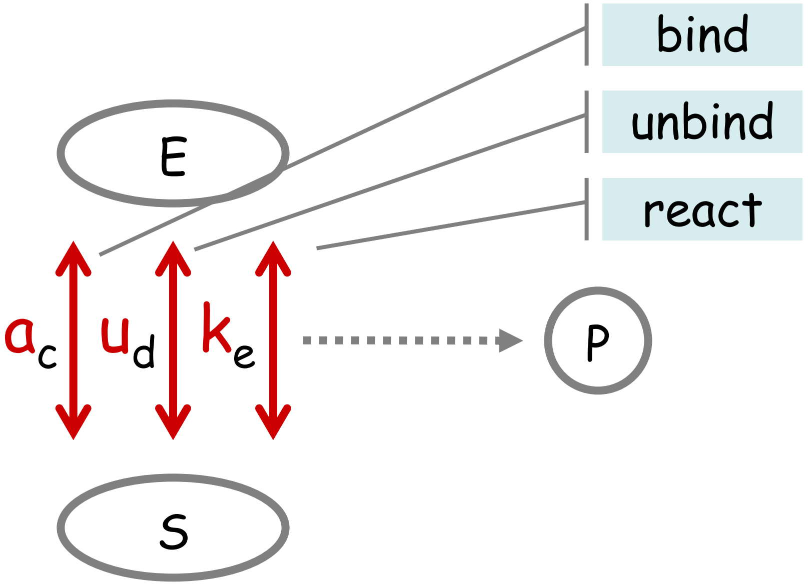
Reaction View



$$\equiv \qquad E+S \underset{d}{\overset{c}{\rightleftharpoons}} ES \xrightarrow{e} P+E$$

intermediate complex

Interaction View



bind

unbind

react

private bindings between one S and one E molecule

S() ≜ new u@d new k@e
$!a_c(u,k); (!u_d; S() + !k_e; P())$

bind unbind react

E() ≜ $?a_c(u,k); (?u_d; E() + ?k_e; E())$

P() ≜ …

As 12 processes (in SPiM)

```
let KKK() =
   (new u1@d1:Release new k1@r1:React
    !a1(u1,k1); (do !u1;KKK() or !k1;KKKst()))          [1]substrate
```

KKK:E1 complex

```
and KKKst() =
   (new u2@d2:Release new k2@r2:React
    do !a2(u2,k2); (do !u2;KKKst() or !k2;KKK())        [2]substrate
    or ?a3(u3,k3); (do ?u3;KKKst() or ?k3;KKKst())      [3]kinase
    or ?a5(u5,k5); (do ?u5;KKKst() or ?k5;KKKst()))     [5]kinase

let E1() =
   ?a1(u1,k1); (do ?u1;E1() or ?k1;E1())                [1]enzyme
```

E1:KKK complex

```
let E2() =
   ?a2(u2,k2); (do ?u2;E2() or ?k2;E2())                [2]enzyme

let KK() =
   (new u3@d3:Release new k3@r3:React
    !a3(u3,k3); (do !u3;KK() or !k3;KK_P()))            [3]substrate

and KK_P() =
   (new u4@d4:Release new k4@r4:React
    new u5@d5:Release new k5@r5:React
    do !a4(u4,k4); (do !u4;KK_P() or !k4;KK())          [4]substrate
    or !a5(u5,k5); (do !u5;KK_P() or !k5;KK_PP()))      [5]substrate
```

```
and KK_PP() =
   (new u6@d6:Release new k6@r6:React
    do !a6(u6,k6); (do !u6;KK_PP() or !k6;KK_P())       [6]substrate
    or ?a7(u7,k7); (do ?u7;KK_PP() or ?k7;KK_PP())      [7]kinase
    or ?a9(u9,k9);                                  ))  [9]kinase
```

One process for each component (12) including enzymes, but not for complexes.

```
and KKPse() =
   do ?a4(u4,k4); (do ?u4;KKPse() or ?k4;KKPse())       [4]phtase
   or ?a6(u6,k6); (do ?u6;KKPse() or ?k6;KKPse())       [6]phtase

let K() =
   (new u7@
    !a7(u7,k7)                                          [7]substrate
```

No need for conservation equations: implicit in “choice” operator in the calculus.

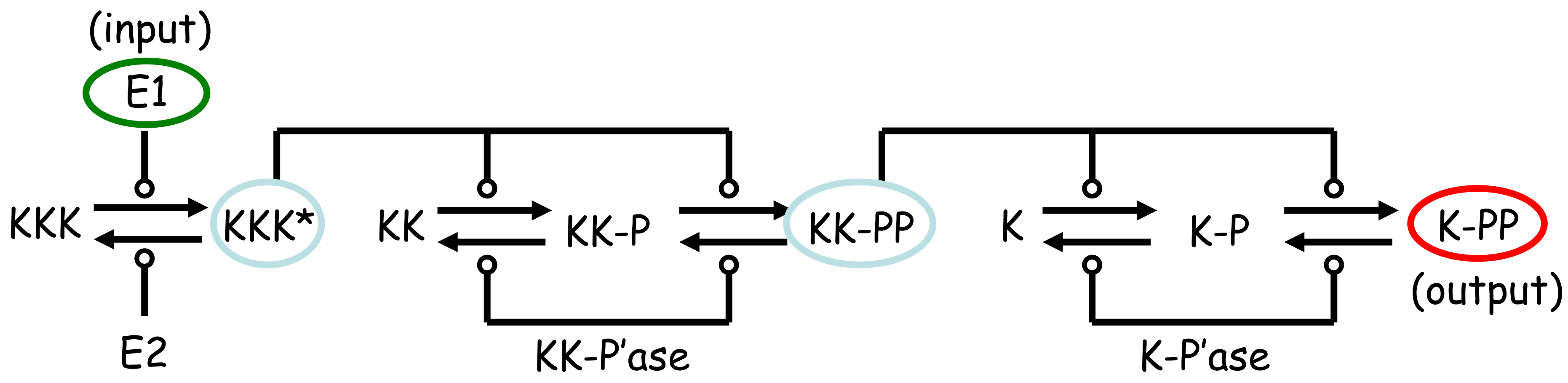
```
and K_P() =
   (new u8@d8:Release new k8@r8:React
    new u9@d9:Release new k9@r9:React
    do !a8(u8,k8); (do !u8;K_P() or !k8;K())            [8]substrate
    or !a9(u9,k9); (do !u9;K_P() or !k9;K_PP()))        [9]substrate

and K_PP() =
   (new u10@d10:Release new k10@r10:React
    !a10(u10,k10); (do !u10;K_PP() or !k10;K_P()))      [10]substrate

and KPse() =
   do ?a8(u8,k8); (do ?u8;KPse() or ?k8;KPse())         [8]phtase
   or ?a10(u10,k10); (do ?u10;KPse() or ?k10;KPse())    [10]phtase
```

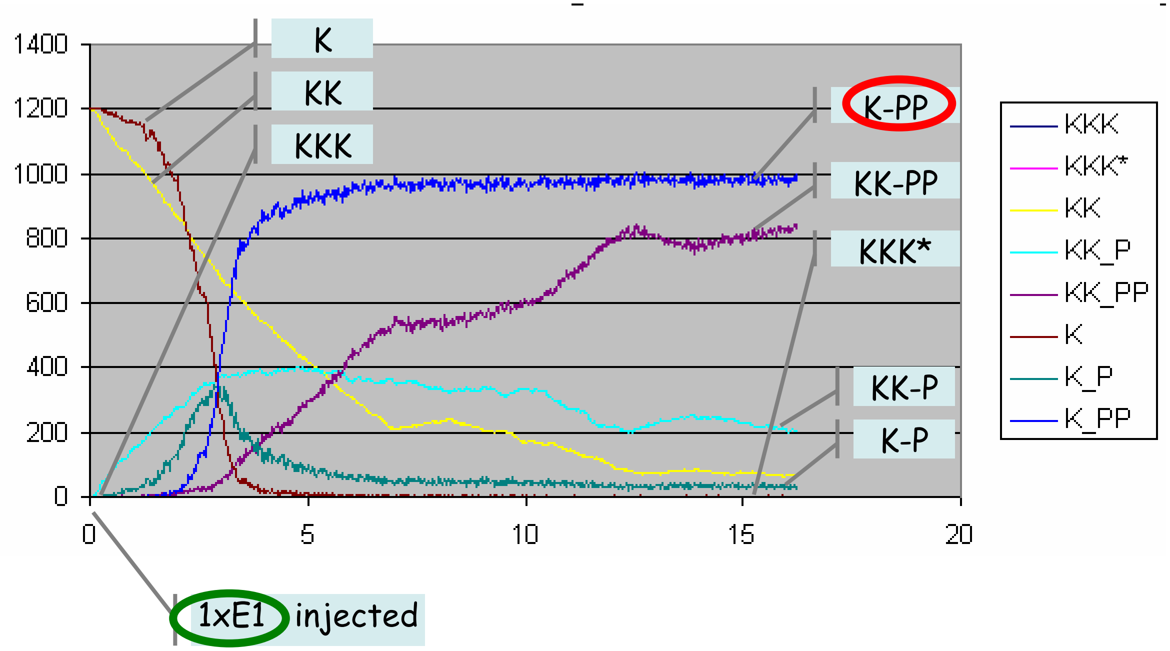
… and 30 Interaction Channels

```
type Release = chan()
type React = chan()
type Bond = chan(Release,React)

new a1@1.0:Bond val d1=1.0 val r1=1.0
new a2@1.0:Bond val d2=1.0 val r2=1.0
new a3@1.0:Bond val d3=1.0 val r3=1.0
new a4@1.0:Bond val d4=1.0 val r4=1.0
new a5@1.0:Bond val d5=1.0 val r5=1.0
new a6@1.0:Bond val d6=1.0 val r6=1.0
new a7@1.0:Bond val d7=1.0 val r7=1.0
new a8@1.0:Bond val d8=1.0 val r8=1.0
new a9@1.0:Bond val d9=1.0 val r9=1.0
new a10@1.0:Bond val d10=1.0 val r10=1.0

…

run 100 of KKK()  run 100 of KK()   run 100 of K()
run 1 of E2()  run 1 of KKPse()  run 1 of KPse()
run 1 of E1()
```

$a_i(u_i,k_i)$: release (u_i@d_i) and react (k_i@r_i) channels passed over bond (a_i) channel.
(No behavior attached to channels except interaction rate.)

MAPK Cascade Simulation in SPiM





1st stage:
KKK* barely rises

2nd stage:
KK-PP rises, but is not stable

3rd stage:
K-PP flips up to max
even anticipating 2nd stage

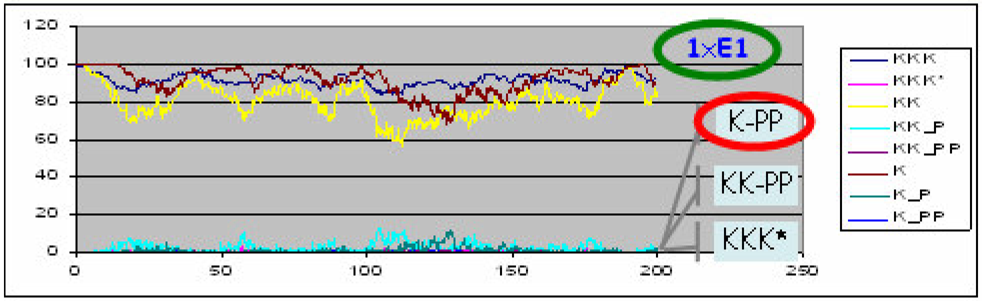
Rates and concentrations from paper:

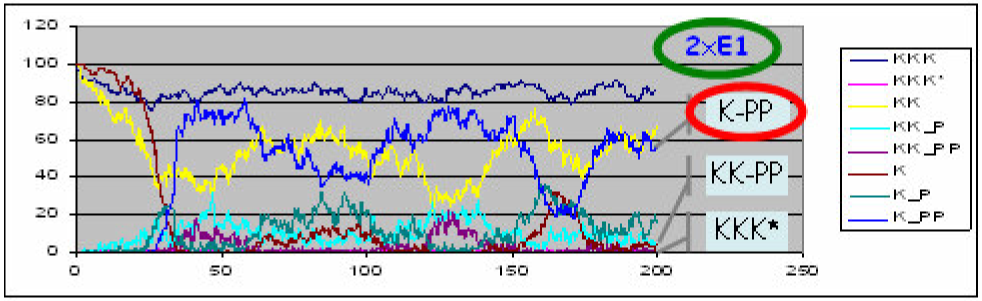
1xE2 (0.3 nM)
1xKKPase (0.3 nM)
120xKPase (120 nM)
3xKKK (3 nM)
1200xKK (1.2 uM)
1200xK (1.2 uM)

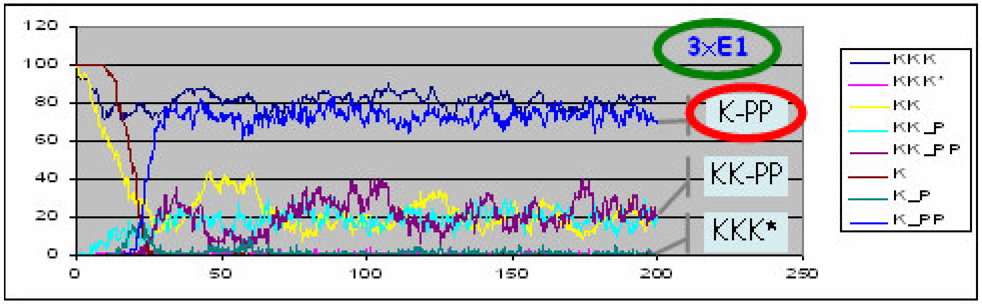
dx = rx = 150, ax = 1
(Kmx = (dx + rx) / ax, Km = 300 nM)

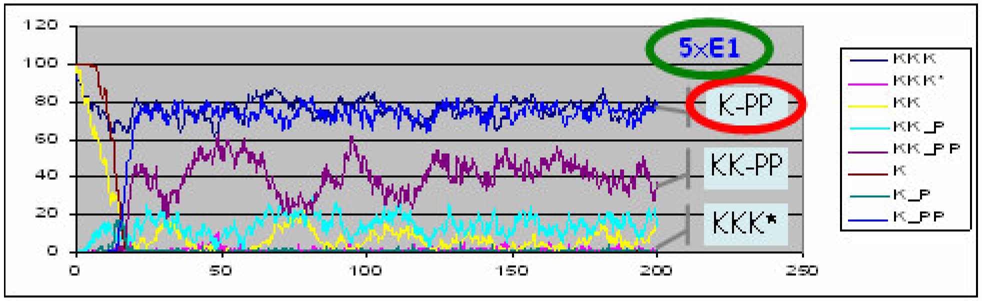
1xE1

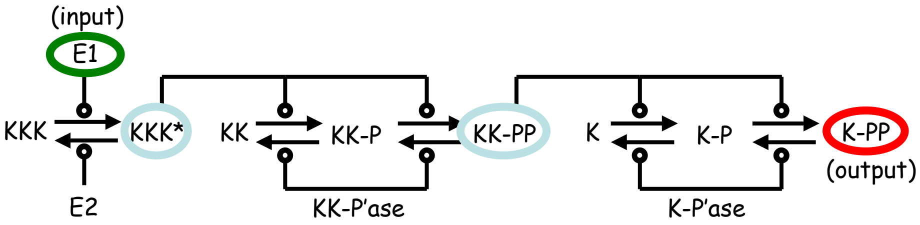
MAPK Cascade Simulation in SPiM



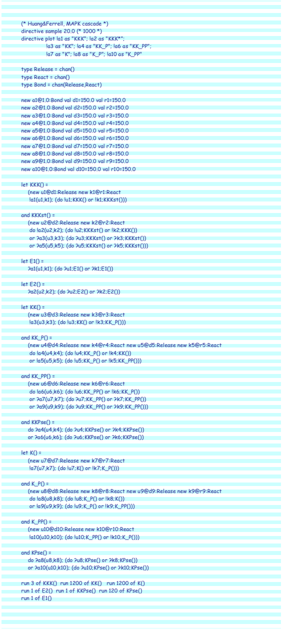






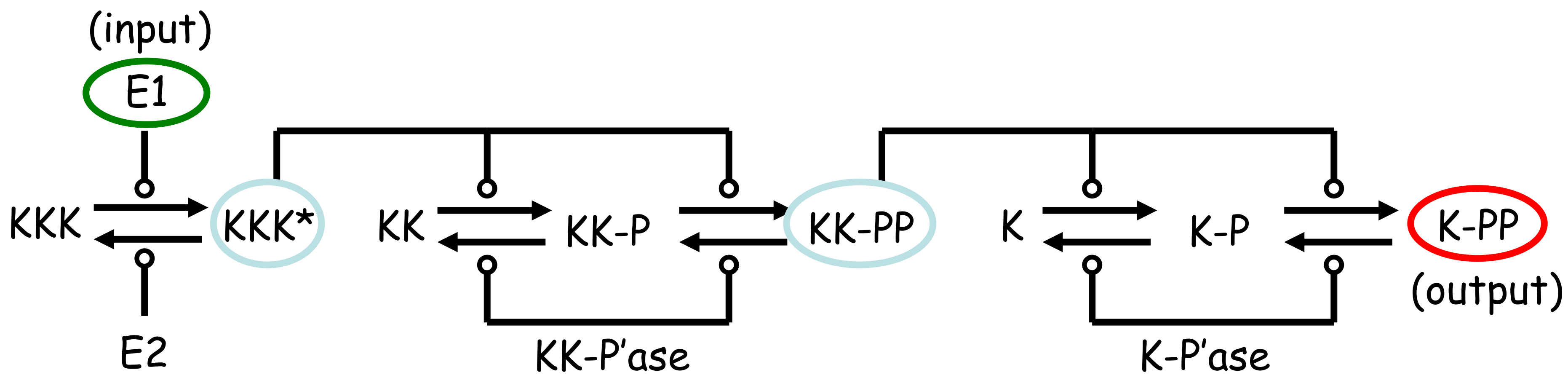


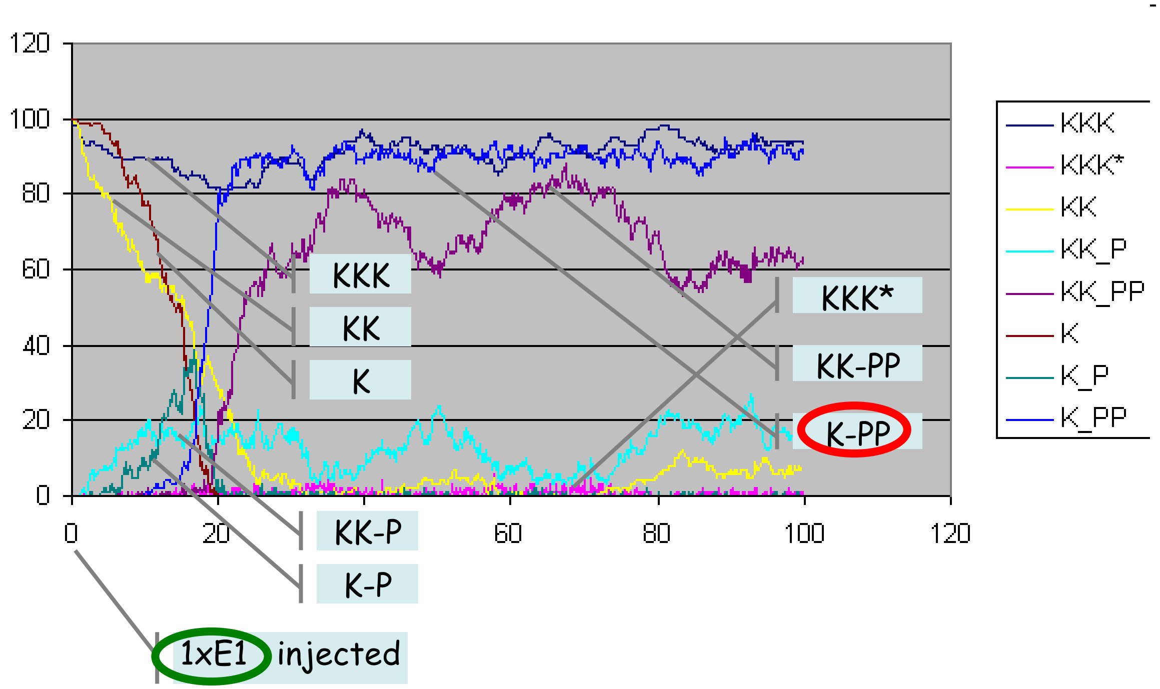
All coefficients 1.0 !!!
100xKKK, 100xKK, 100xK,
13xE2, 13xKKPse, 13xKPse.
nxE1 as indicated
(1xE1 is not sufficient to produce an output)



```
(* Huang&Ferrell, MAPK cascade *)
directive sample 20.0 (* 1000 *)
directive plot !a1 as "KKK"; !a2 as "KKK*";
        !a3 as "KK"; !a4 as "KK_P"; !a6 as "KK_PP";
        !a7 as "K"; !a8 as "K_P"; !a10 as "K_PP"

type Release = chan()
type React = chan()
type Bond = chan(Release,React)

new a1@1.0:Bond val d1=150.0 val r1=150.0
new a2@1.0:Bond val d2=150.0 val r2=150.0
new a3@1.0:Bond val d3=150.0 val r3=150.0
new a4@1.0:Bond val d4=150.0 val r4=150.0
new a5@1.0:Bond val d5=150.0 val r5=150.0
new a6@1.0:Bond val d6=150.0 val r6=150.0
new a7@1.0:Bond val d7=150.0 val r7=150.0
new a8@1.0:Bond val d8=150.0 val r8=150.0
new a9@1.0:Bond val d9=150.0 val r9=150.0
new a10@1.0:Bond val d10=150.0 val r10=150.0

let KKK() =
  (new u1@d1:Release new k1@r1:React
   !a1(u1,k1); (do !u1;KKK() or !k1;KKKst()))

and KKKst() =
  (new u2@d2:Release new k2@r2:React
   do !a2(u2,k2); (do !u2;KKKst() or !k2;KKK())
   or ?a3(u3,k3); (do ?u3;KKKst() or ?k3;KKKst())
   or ?a5(u5,k5); (do ?u5;KKKst() or ?k5;KKKst()))

let E1() =
  ?a1(u1,k1); (do ?u1;E1() or ?k1;E1())

let E2() =
  ?a2(u2,k2); (do ?u2;E2() or ?k2;E2())

let KK() =
  (new u3@d3:Release new k3@r3:React
   !a3(u3,k3); (do !u3;KK() or !k3;KK_P()))

and KK_P() =
  (new u4@d4:Release new k4@r4:React new u5@d5:Release new k5@r5:React
   do !a4(u4,k4); (do !u4;KK_P() or !k4;KK())
   or !a5(u5,k5); (do !u5;KK_P() or !k5;KK_PP()))

and KK_PP() =
  (new u6@d6:Release new k6@r6:React
   do !a6(u6,k6); (do !u6;KK_PP() or !k6;KK_P())
   or ?a7(u7,k7); (do ?u7;KK_PP() or ?k7;KK_PP())
   or ?a9(u9,k9); (do ?u9;KK_PP() or ?k9;KK_PP()))

and KKPse() =
  do ?a4(u4,k4); (do ?u4;KKPse() or ?k4;KKPse())
  or ?a6(u6,k6); (do ?u6;KKPse() or ?k6;KKPse())

let K() =
  (new u7@d7:Release new k7@r7:React
   !a7(u7,k7); (do !u7;K() or !k7;K_P()))

and K_P() =
  (new u8@d8:Release new k8@r8:React new u9@d9:Release new k9@r9:React
   do !a8(u8,k8); (do !u8;K_P() or !k8;K())
   or !a9(u9,k9); (do !u9;K_P() or !k9;K_PP()))

and K_PP() =
  (new u10@d10:Release new k10@r10:React
   !a10(u10,k10); (do !u10;K_PP() or !k10;K_P()))

and KPse() =
  do ?a8(u8,k8); (do ?u8;KPse() or ?k8;KPse())
  or ?a10(u10,k10); (do ?u10;KPse() or ?k10;KPse())

run 3 of KKK()  run 1200 of KK()   run 1200 of K()
run 1 of E2()  run 1 of KKPse()  run 120 of KPse()
run 1 of E1()
```

MAPK Cascade Simulation in SPiM





1st stage:

KKK* barely rises

2nd stage:

KK-PP rises, but is not stable

3rd stage:

K-PP flips up to max

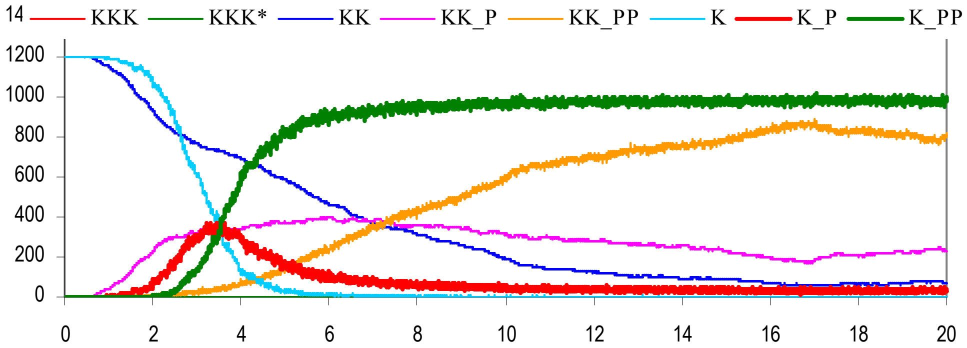
even anticipating 2nd stage

All coefficients 1.0 !!!
100xKKK, 100xKK, 100xK,
5xE2, 5xKKPse, 5xKPse.

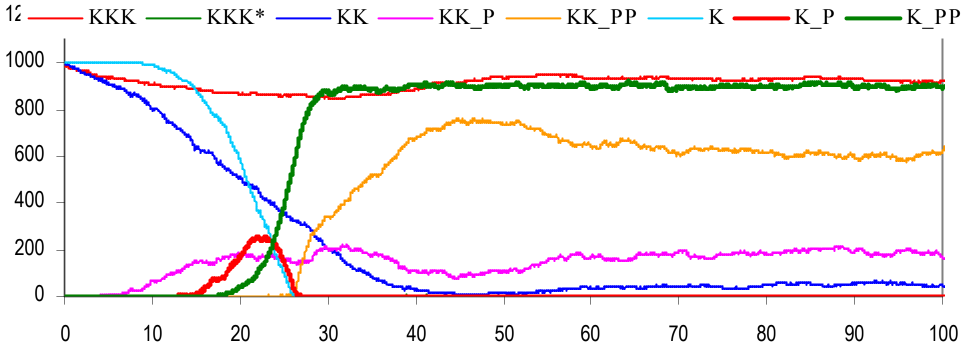
Input is 1x**E1**.
Output is 90x**K-PP** (ultrasensitivity).

MAPK Cascade Simulation in SPiM

Parameters from paper
(wide rate range: 1-150, wide concentration range: 3nm – 1200nm)

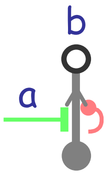


Artificial parameters
(all rates 1.0, all concentrations 1000)



Inverter ODE

Inverter ODE



$Not_{hi}(a,b) = !b;Not_{hi}(a,b) \oplus ?a;Not_{lo}(a,b)$

$Not_{lo}(a,b) = \tau_{del};Not_{hi}(a,b)$

$A(a) = !a;(A(a)|A(a))$ <u>linearly increasing input</u>

$A(x_{(r)})$ | n of $Not_{hi}(x_{(r)},y_{(s)})$

$Not_{hi}/x,y + A/x \rightarrow^{r} Not_{lo}/x,y + A/x + A/x$

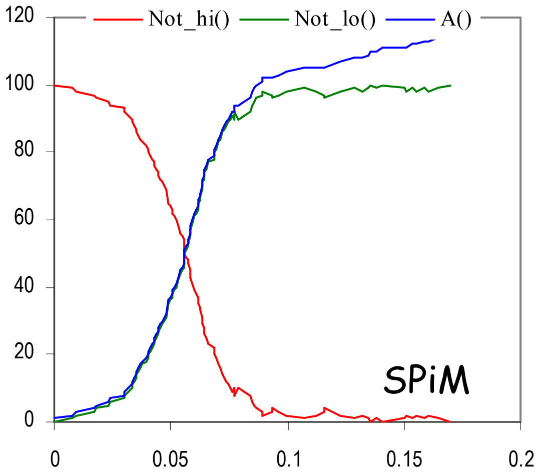
$Not_{lo}/x,y \rightarrow^{del} Not_{hi}/x,y$

A/x + n of $Not_{hi}/x,y$

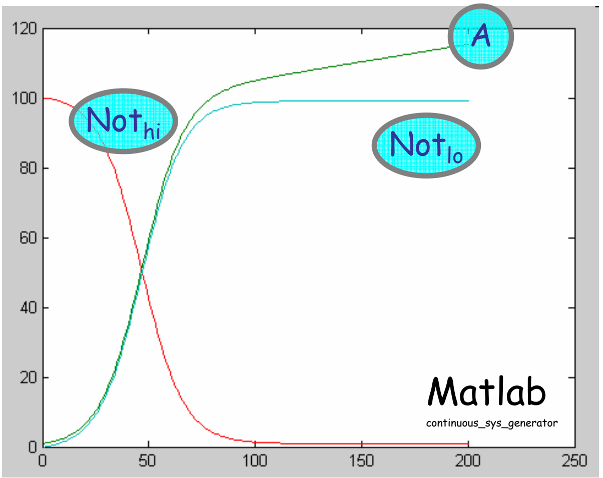
$[A/x]^{\bullet} = r[Not_{hi}/x,y][A/x]$

$[Not_{hi}/x,y]^{\bullet} = -r[Not_{hi}/x,y][A/x]+del[Not_{lo}/x,y]$

$[Not_{lo}/x,y]^{\bullet} = r[Not_{hi}/x,y][A/x]-del[Not_{lo}/x,y]$

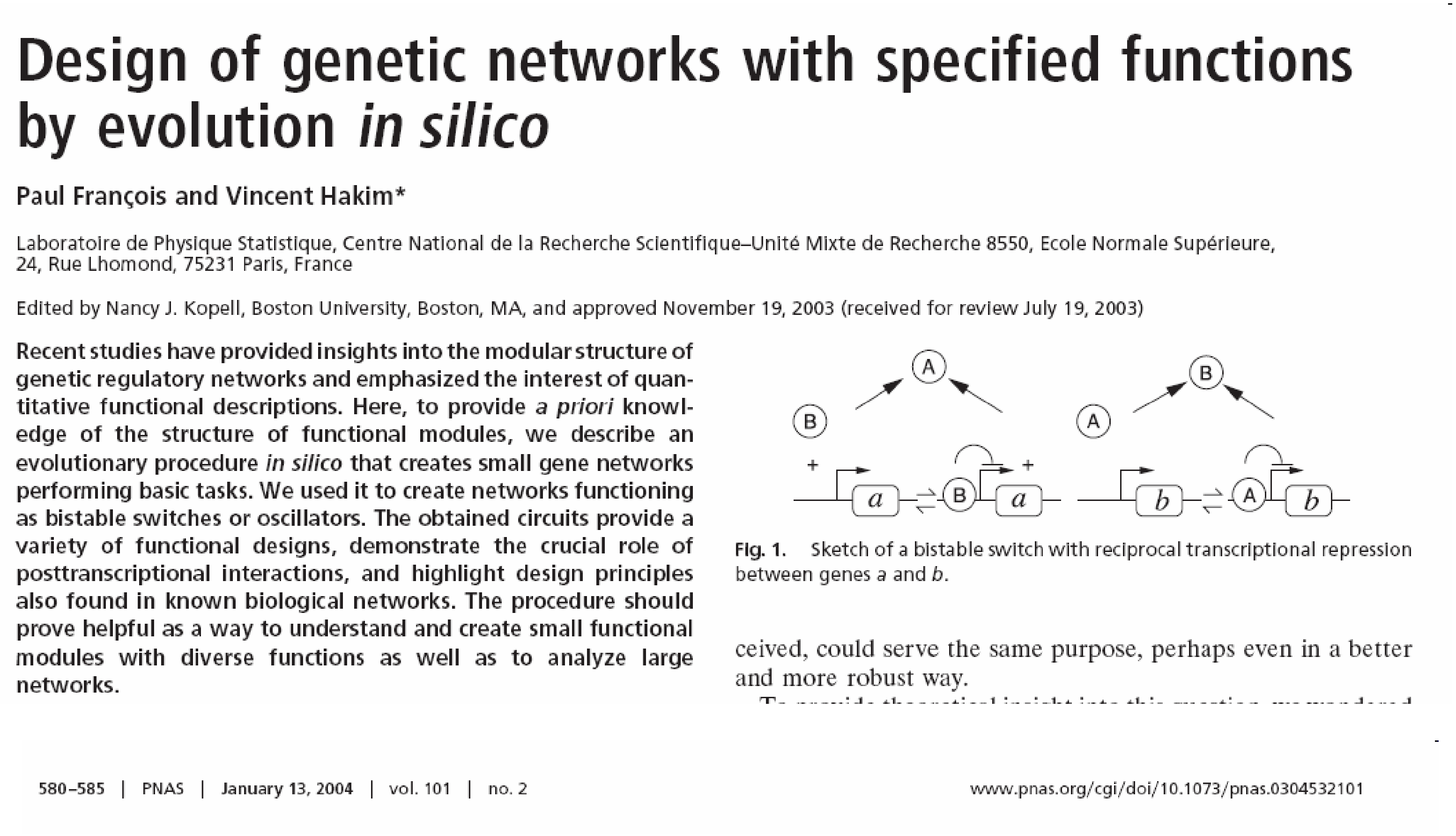


```
directive sample 0.2 1000
directive plot Not_hi(); Not_lo(); A()
new a@1.0:chan new b@1.0:chan
val del = 1.0

let Not_hi(a:chan, b:chan) =
  do !b; Not_hi(a,b) or ?a;
Not_lo(a,b)
and Not_lo(a:chan, b:chan) =
  delay@del;Not_hi(a,b)

let A(a:chan) = !a;(A(a)|A(a))

run (A(a) | 100 of Not_hi(a,b))
```



```
interval/step [0:0.001:0.2]      n=100, r=1, del=1
(A)       dx1/dt = x2*x1                   1
(Not_hi)  dx2/dt = -x2*x1+x3             100
(Not_lo)  dx3/dt = x2*x1-x3                0
```

François and Hakim



Design of genetic networks with specified functions by evolution *in silico*

Paul François and Vincent Hakim*

Laboratoire de Physique Statistique, Centre National de la Recherche Scientifique–Unité Mixte de Recherche 8550, Ecole Normale Supérieure, 24, Rue Lhomond, 75231 Paris, France

Edited by Nancy J. Kopell, Boston University, Boston, MA, and approved November 19, 2003 (received for review July 19, 2003)

Recent studies have provided insights into the modular structure of genetic regulatory networks and emphasized the interest of quantitative functional descriptions. Here, to provide *a priori* knowledge of the structure of functional modules, we describe an evolutionary procedure *in silico* that creates small gene networks performing basic tasks. We used it to create networks functioning as bistable switches or oscillators. The obtained circuits provide a variety of functional designs, demonstrate the crucial role of posttranscriptional interactions, and highlight design principles also found in known biological networks. The procedure should prove helpful as a way to understand and create small functional modules with diverse functions as well as to analyze large networks.

Fig. 1. Sketch of a bistable switch with reciprocal transcriptional repression between genes *a* and *b*.

ceived, could serve the same purpose, perhaps even in a better and more robust way.

To provide theoretical insight into this question, we wondered

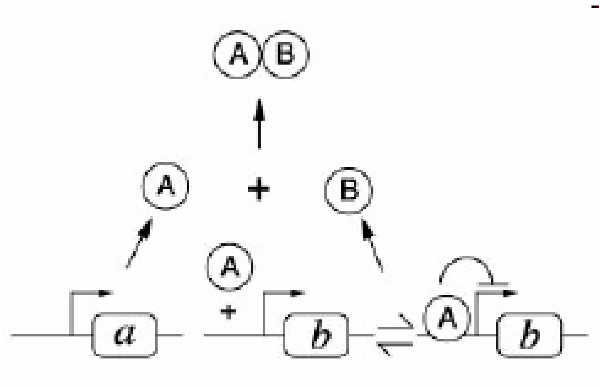
580–585 | PNAS | January 13, 2004 | vol. 101 | no. 2

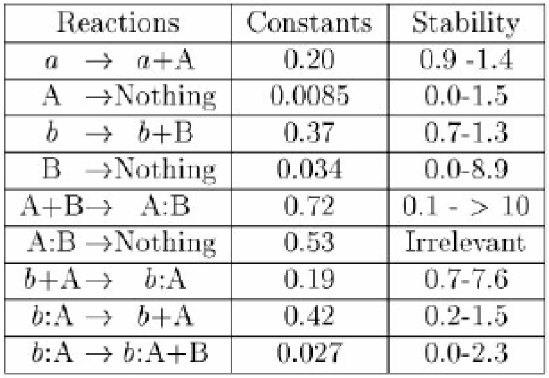
www.pnas.org/cgi/doi/10.1073/pnas.0304532101

François & Hakim Fig3A

PNAS (101)2, 580-585, 2004

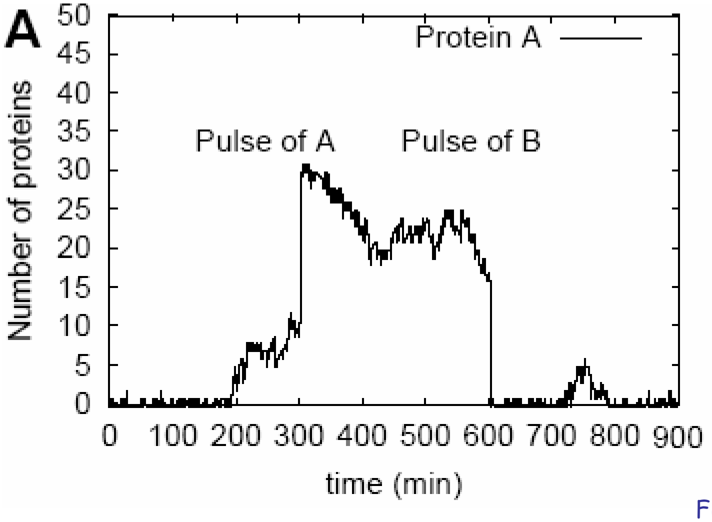
Design of genetic networks with specified functions by evolution in silico

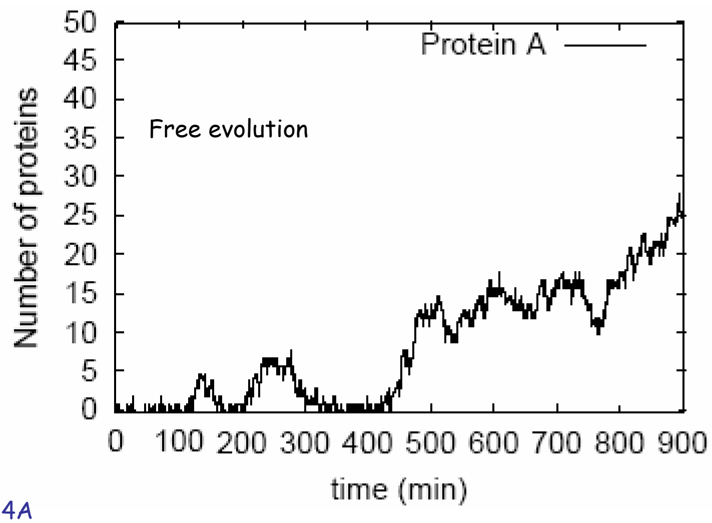




Reactions	Constants	Stability
a → a+A	0.20	0.9 -1.4
A →Nothing	0.0085	0.0-1.5
b → b+B	0.37	0.7-1.3
B →Nothing	0.034	0.0-8.9
A+B→ A:B	0.72	0.1 - > 10
A:B →Nothing	0.53	Irrelevant
b+A→ b:A	0.19	0.7-7.6
b:A → b+A	0.42	0.2-1.5
b:A → b:A+B	0.027	0.0-2.3

Fig 3A



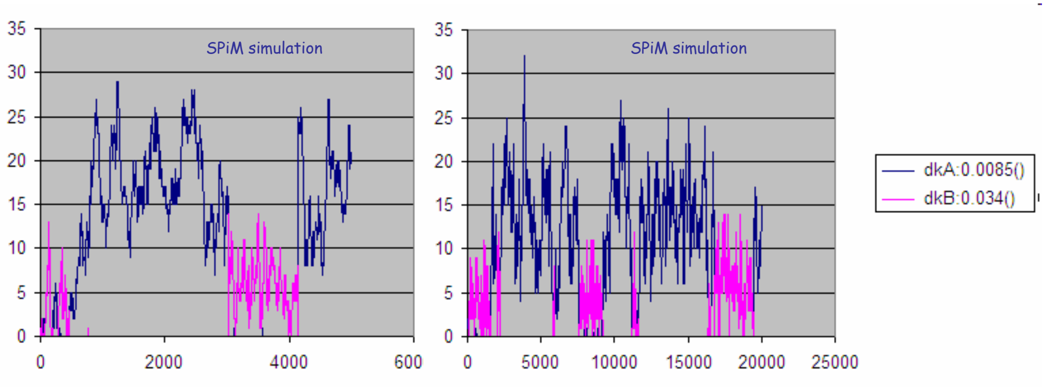


Free evolution

Fig 14A

François & Hakim Fig3A, SPiM simulation

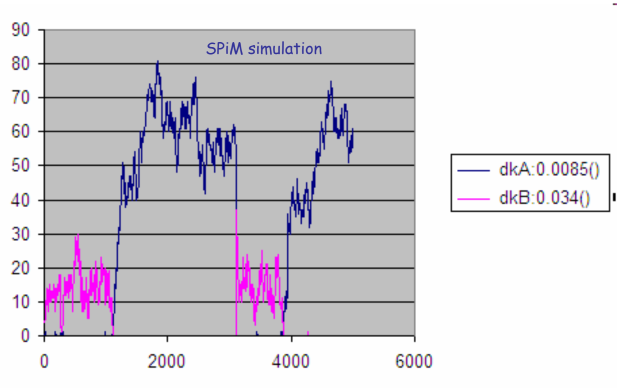
Parameters as in paper



Spontaneous switch at ~500
(as discussed in Supporting Text)
30xB injected at ~3000
30xA injected at ~4000

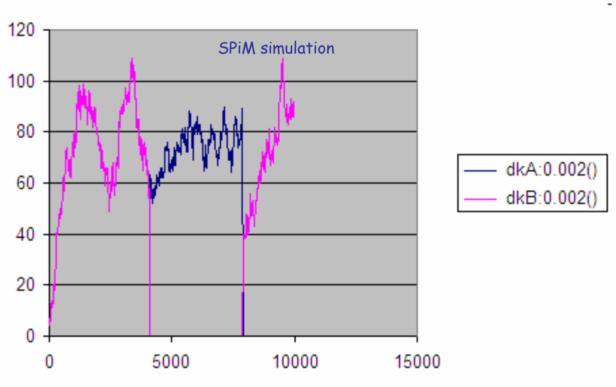
Free evolution

3 copies of each gene.

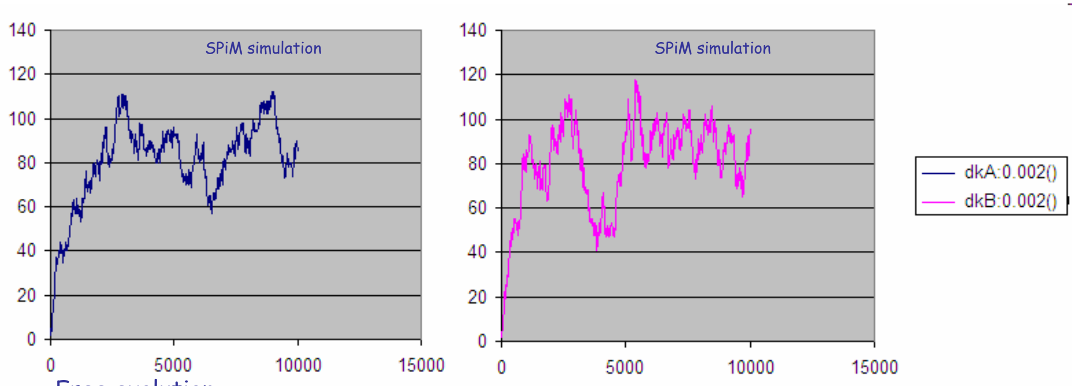


Spontaneous switch at ~1100
100xB injected at ~3000
30xA injected at ~4000

Modified for stability: dkA = 0.02, dkB = 0.02



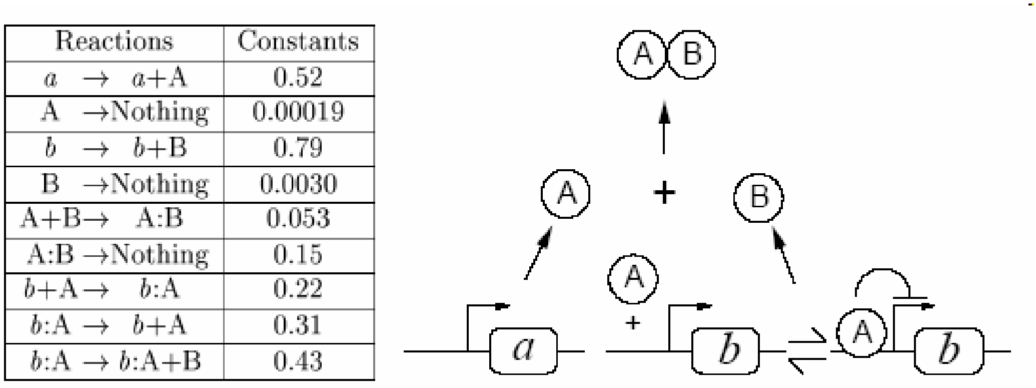
120xA injected at ~4000
120xB injected at ~8000



Free evolution

François & Hakim Fig3Ast8

Circuit of Fig 3A with parameters from SupportingText Fig 8, plotted in Fig 13A



Reactions	Constants
$a \rightarrow a+A$	0.52
$A \rightarrow$ Nothing	0.00019
$b \rightarrow b+B$	0.79
$B \rightarrow$ Nothing	0.0030
$A+B \rightarrow A{:}B$	0.053
$A{:}B \rightarrow$ Nothing	0.15
$b+A \rightarrow b{:}A$	0.22
$b{:}A \rightarrow b+A$	0.31
$b{:}A \rightarrow b{:}A+B$	0.43

Fig 8

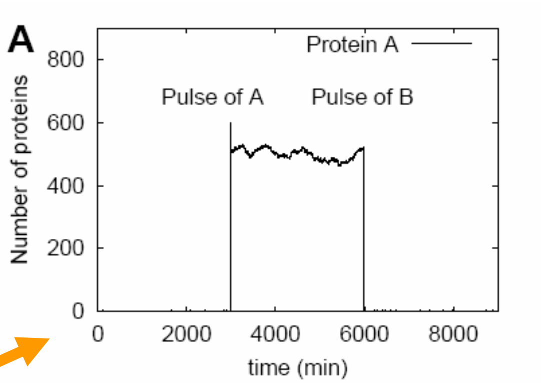
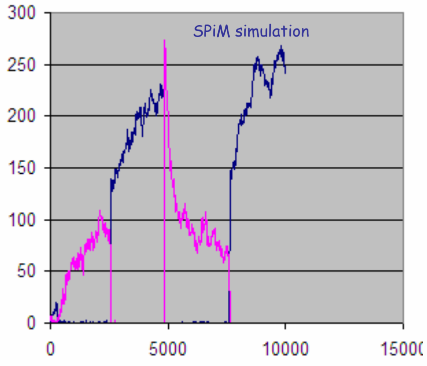
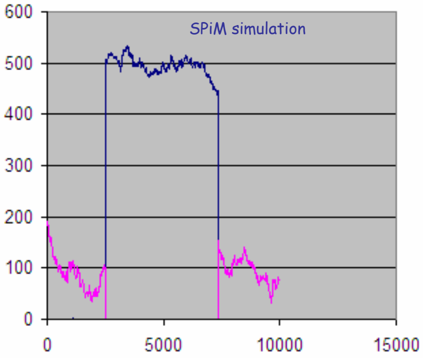


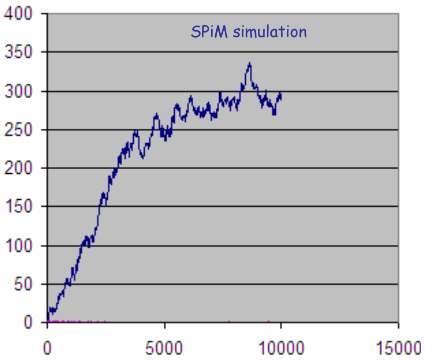
Fig 13A

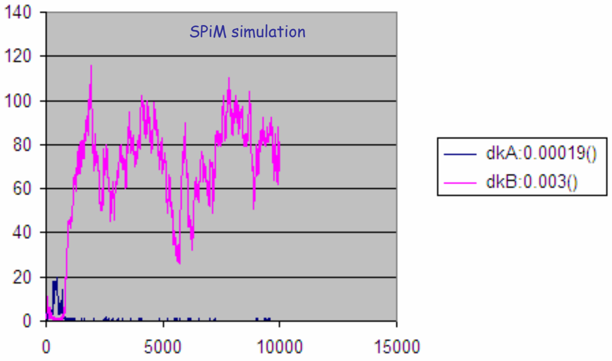


200xA injected at ~2500
500xB injected at ~5000
200xA injected at ~7500



200xB injected at 0
600xA injected at ~2500
600xB injected at ~7500





Free evolution

François & Hakim 3A in SPiM

```
(* Francois and Hakim circuit 3A *)

val pntAunb = 0.42
val geneACst = 0.20
val geneBCst = 0.37
val geneBInh = 0.027
val bA = 0.19
val AB = 0.72
val dkA = 0.0085
val dkB = 0.034
val dkAB = 0.53
```

```
let ptnA() =
    (new unb@pntAunb
     do delay@dkA or !AB or !bA(unb);(?unb; ptnA()))

let ptnB() =
    do delay@dkB or ?AB;cpxAB()

let cpxAB() = delay@dkAB

let geneA() =
    delay@geneACst; (ptnA() | geneA())

let geneBfree() =
    do delay@geneBCst; (ptnB() | geneBfree())
    or ?bA(unb); geneBbound(unb)

and geneBbound(unb:ch()) =
    do delay@geneBInh; (ptnB() | geneBbound(unb))
    or !unb; geneBfree()

run (geneA() | geneBfree())
```