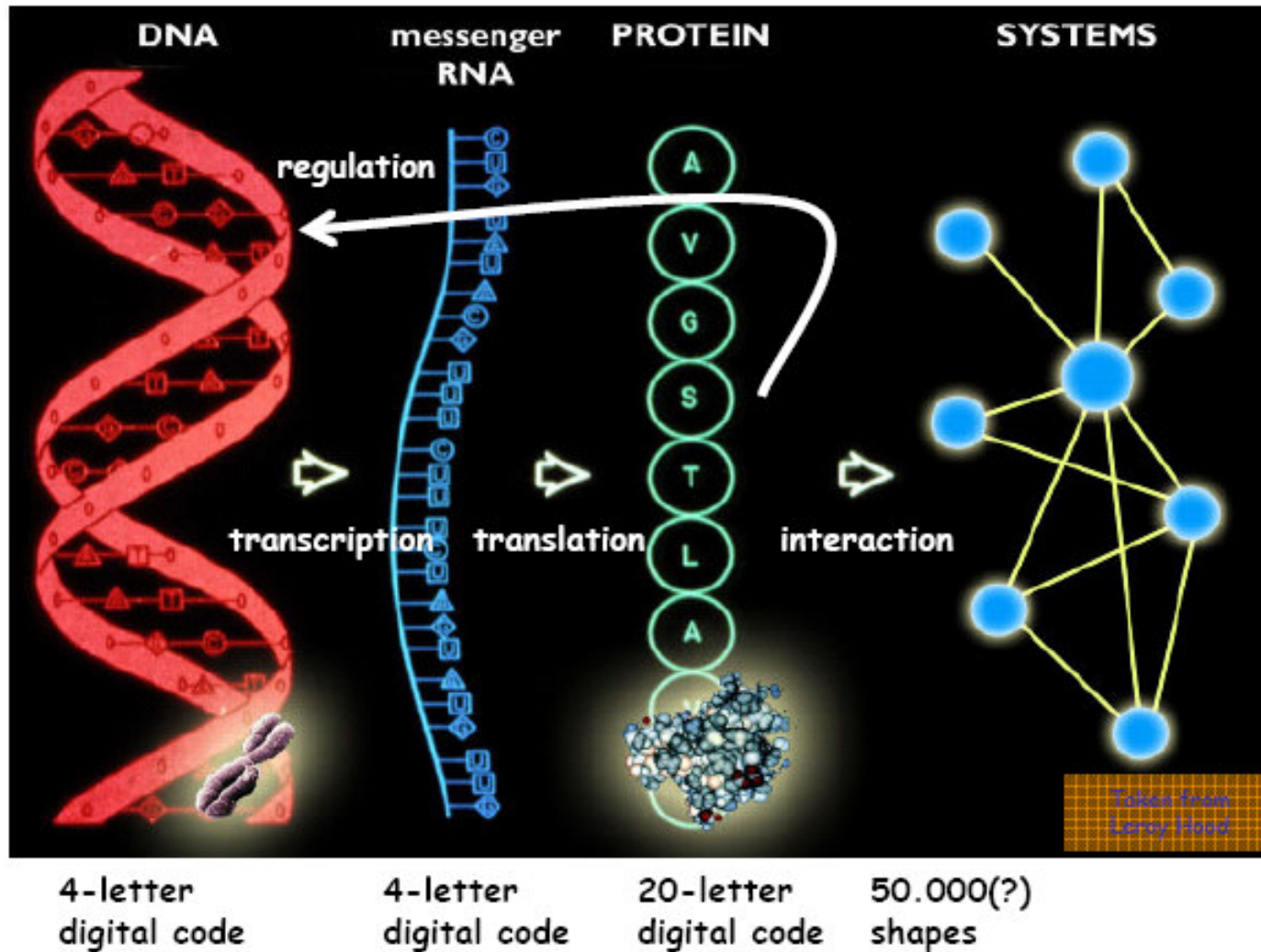

Simulating Biological Systems in the Stochastic π -calculus

Andrew Phillips

Microsoft Research
Cambridge UK

andrew.phillips@microsoft.com

Summary of Molecular Biology



Understanding Biological Systems

- The Human Genome project generated vast quantities of data
- Challenge:
 - Understand and precisely describe the behaviour of biological systems
- Two complementary approaches:
 - Look at experimental results and “infer” general system properties
 - Build detailed models of systems and test these in the lab
- Traditional modelling tools do not scale:
 - If a small part of the model changes, need to update the rest
 - e.g. Differential Equations, Chemical Notations...
- Need a more scalable approach.

Process Calculi for Biology

- Ongoing Experiment:
 - Use *process calculi* to model, simulate and analyse biological systems
- Modelling:
 - Model very large systems incrementally
 - By composing simpler models of subsystems in an intuitive way
- Simulation:
 - Perform *in silico* experiments on biological system models
 - Use these to formulate testable hypotheses for experimentation *in vivo*
- Analysis:
 - A range of established techniques (types, equivalences, model checking)
 - Could provide insight into fundamental properties of biological systems
- π -calculus: one of the simplest and most well-studied calculi

Related Work

- Stochastic π -calculus proposed by [Priami, 1995]
- Used to model and simulate a range of biological systems:
 - RTK MAPK pathway [Regev et al., 2001]
 - Gene Regulation by positive Feedback [Priami et al., 2001]
 - Cell Cycle Control in Eukaryotes [Lecca and Priami, 2003]
- First simulator for stochastic π -calculus [BioSPI]
 - A subset of π -calculus with limited choice
 - Compiles a calculus process to an FCP procedure
 - Executed by the FCP Logix platform [Silverman et al., 1987]

Outline: simulating biological systems in the stochastic π -calculus

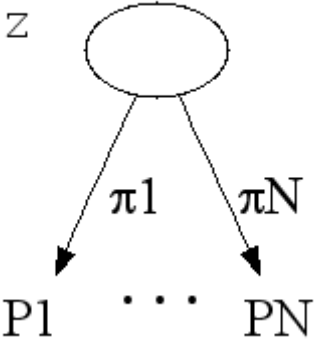
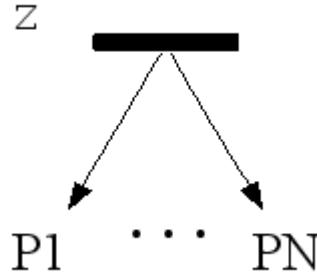
- The stochastic π -calculus (theory)
 - Brief description
 - Graphical representation
- The SPiM language and simulator (tutorial)
 - Simulation algorithm
 - Simple chemical examples
- Biological examples (systems)
 - Gene networks engineered in vivo
 - Gene networks designed in silico
 - Immune system pathway (in progress)

The Stochastic π -calculus

Calculus Syntax

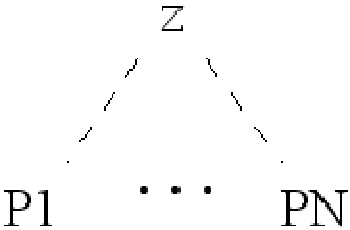
$\pi ::=$	$?x(m)$	Input value m on channel x at $rate(x) s^{-1}$
	$!x(n)$	Output value n on channel x at $rate(x) s^{-1}$
	τ_r	Delay at $r s^{-1}$
$P ::=$	$\pi_1.P_1 + \dots + \pi_N.P_N$	Choice between actions
	$P_1 \mid \dots \mid P_M$	Parallel composition of processes
	$X(n)$	Instance of X with arguments n
	$\text{new } z P$	Restriction of names z to a process
$E ::=$	$X(m) = P$	Definition of X , where $fn(P) \subseteq m$
	E_1, \dots, E_N	Union of definitions

Graphical Syntax [Phillips and Cardelli, 2005]

	Choice	Parallel	Instance
P	$\text{new } z (\pi_1.P_1 + \dots + \pi_N.P_N)$	$\text{new } z (P_1 \dots P_N)$	$X(m) = P$ $\text{new } z X(n)$
P			

	Definition	Union
E	$X(m) = P$	E_1, \dots, E_N
E	P	$E_1 \quad \dots \quad E_N$

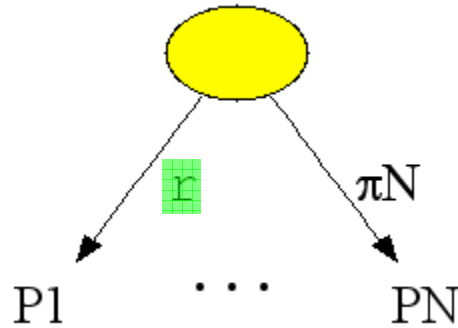
Graphical Syntax: Execution

	Null	Parallel	Instance
P	$\text{new } z \ \mathbf{0}$	$\text{new } z \ (P_1 \mid \dots \mid P_N)$	$X(m) = P$ $\text{new } z \ X(n)$
P	\emptyset		$z \ \text{---} \ \mathbf{X} \{n/m\}$

During execution:

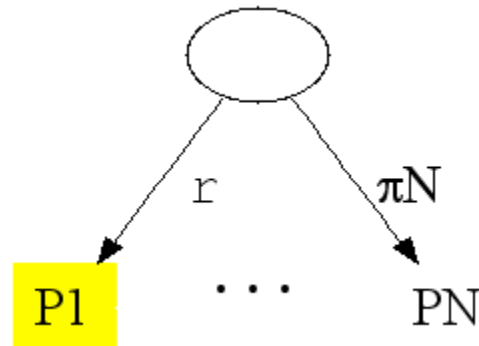
- Highlight nodes in the graph to represent current state
- Draw links between nodes to represent complexes

Execution: Stochastic Delay



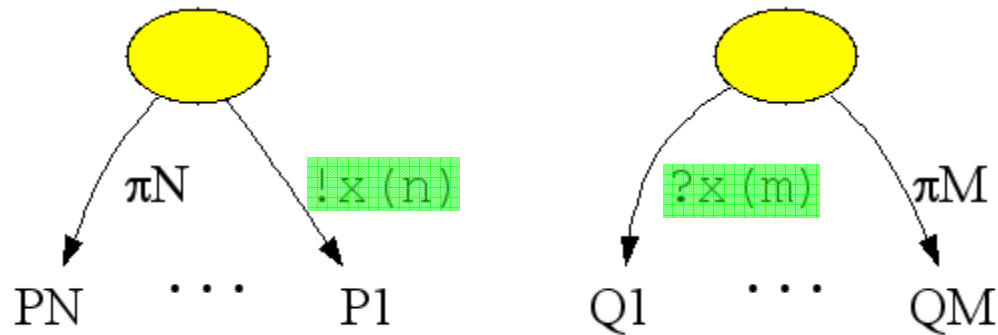
$$\tau_r.P_1 + \dots + \pi_N.P_N$$

Execution: Stochastic Delay



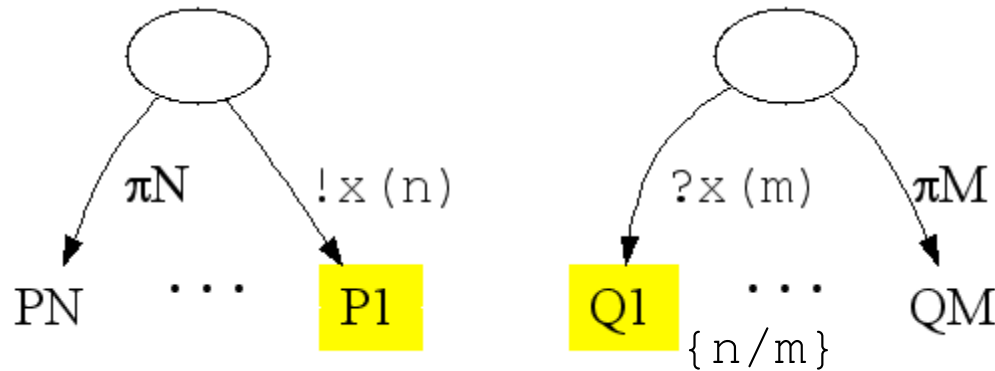
$$\tau_r.P_1 + \dots + \pi_N.P_N \longrightarrow P_1$$

Execution: Interaction



$$!x(n).P_1 + \dots + \pi_N.P_N \quad | \quad ?x(m).Q_1 + \dots + \pi_M.Q_M$$

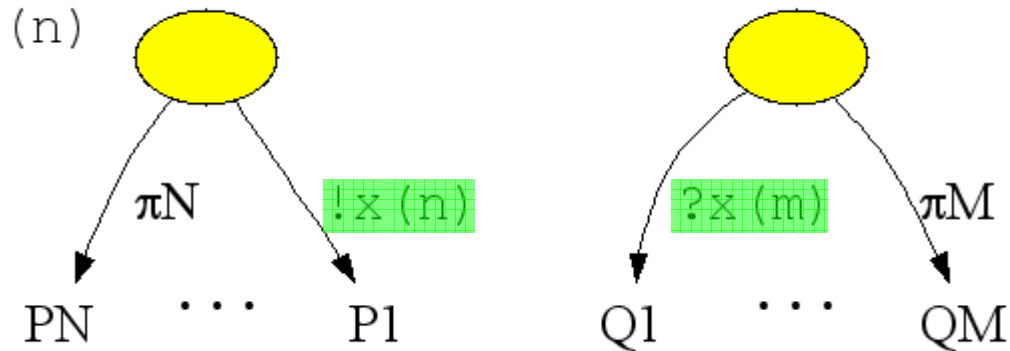
Execution: Interaction



$$\begin{aligned} & !x(n).P_1 + \dots + \pi_N.P_N \\ \longrightarrow & P_1 \end{aligned}$$

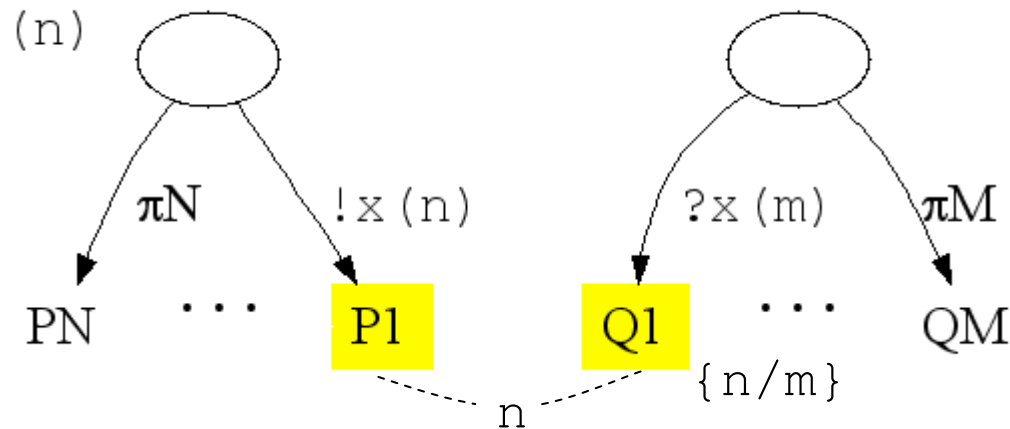
$$\begin{aligned} & ?x(m).Q_1 + \dots + \pi_M.Q_M \\ & Q_{1\{n/m\}} \end{aligned}$$

Execution: Binding



$$\text{new } n (!x(n).P_1 + \dots + \pi_N.P_N) \quad | \quad ?x(m).Q_1 + \dots + \pi_M.Q_M$$

Execution: Binding



$$\begin{array}{l}
 \text{new } n (!x(n).P_1 + \dots + \pi_N.P_N) \quad | \quad ?x(m).Q_1 + \dots + \pi_M.Q_M \\
 \longrightarrow \quad \text{new } n (P_1 \quad | \quad Q_{1\{n/m\}})
 \end{array}$$

The SPiM Language and Simulator

with Luca Cardelli (Microsoft Research)

Abstract Machine [Phillips and Cardelli, 2004]

- Defined an abstract machine for the stochastic π -calculus.
- Based on standard kinetic theory [Gillespie 1977]
- Probability of a reaction determined by:
 - Reaction rate
 - Number of reactants
- Proved correct with respect to the calculus.
- Direct mapping to functional code (OCaml / F#).
- Used as the basis for implementing the SPiM simulator.

The SPiM Simulator [Phillips, 2005]

■ Language Definition (v0.04)

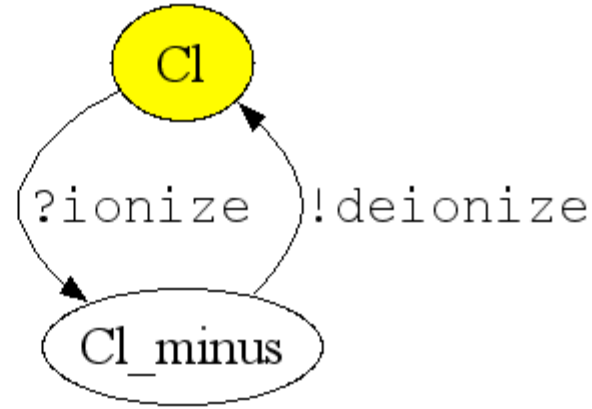
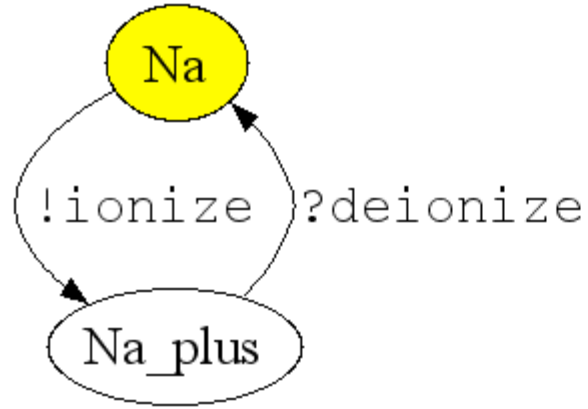
Dec	$::=$	new $x\{@r\} : t$ type $n = t$ val $m = v$ run P let D_1 and ... and D_N	Channel Declaration Type Declaration Value Declaration Process Declaration Definitions, $N \geq 1$
D	$::=$	$X(m_1, \dots, m_N) = P$	Definition, $N \geq 0$
P	$::=$	$()$ $(P_1 \mid \dots \mid P_M)$ $X(v_1, \dots, v_N)$ $\pi\{; P\}$ do $\pi_1\{; P_1\}$ or ... or $\pi_M\{; P_M\}$ $(Dec_1 \dots Dec_N P)$	Null Process Parallel, $M \geq 2$ Instantiation, $N \geq 0$ Action Choice, $M \geq 2$ Declarations, $N \geq 0$
π	$::=$	$!x\{(v_1, \dots, v_N)\}$ $?x\{(m_1, \dots, m_N)\}$ $delay@r$	Output, $N \geq 0$ Input, $N \geq 0$ Delay

Ionization:



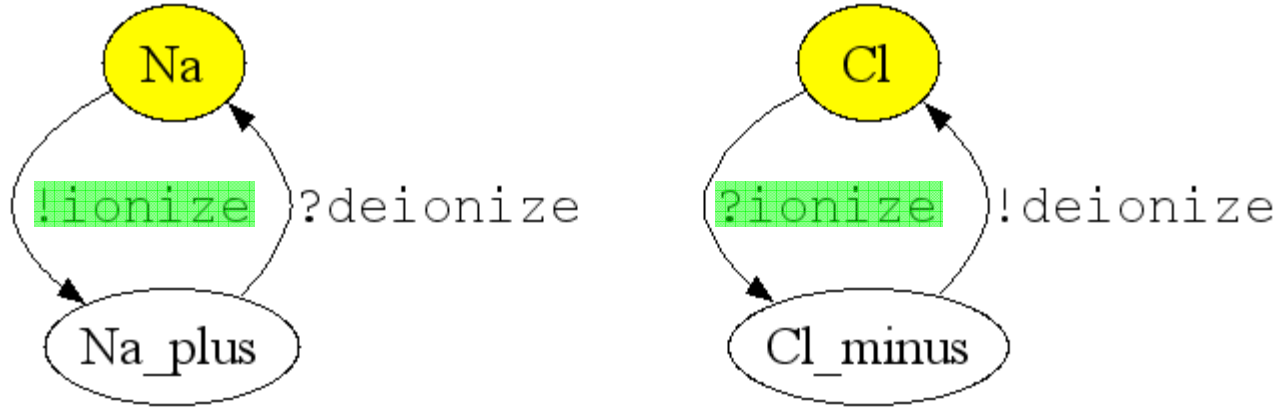
let Na() = !ionize; Na_plus()
and Na_plus() = ?deionize; Na()

let Cl() = ?ionize; Cl_minus()
and Cl_minus() = !deionize; Cl()



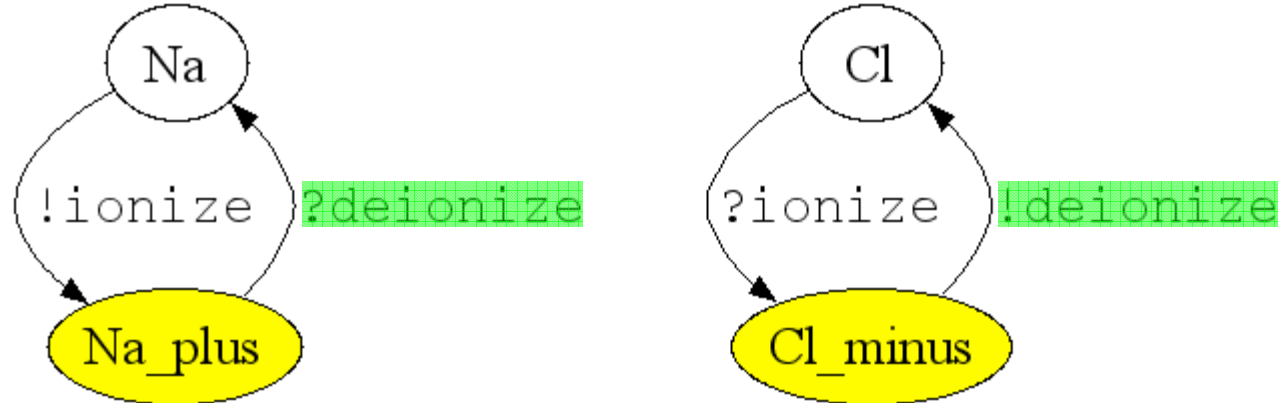
- *Na* can ionize *Cl* at $rate(ionize) = 100s^{-1}$
- *Cl*⁻ can deionize *Na*⁺ at $rate(deionize) = 10s^{-1}$

Ionization:



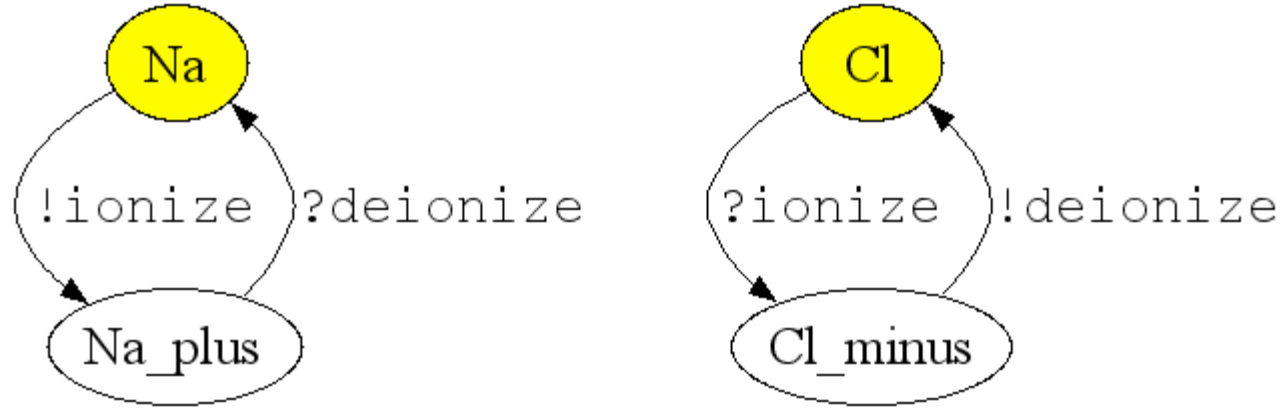
- *Na* can ionize *Cl* by an output on the *ionize* channel

Ionization:



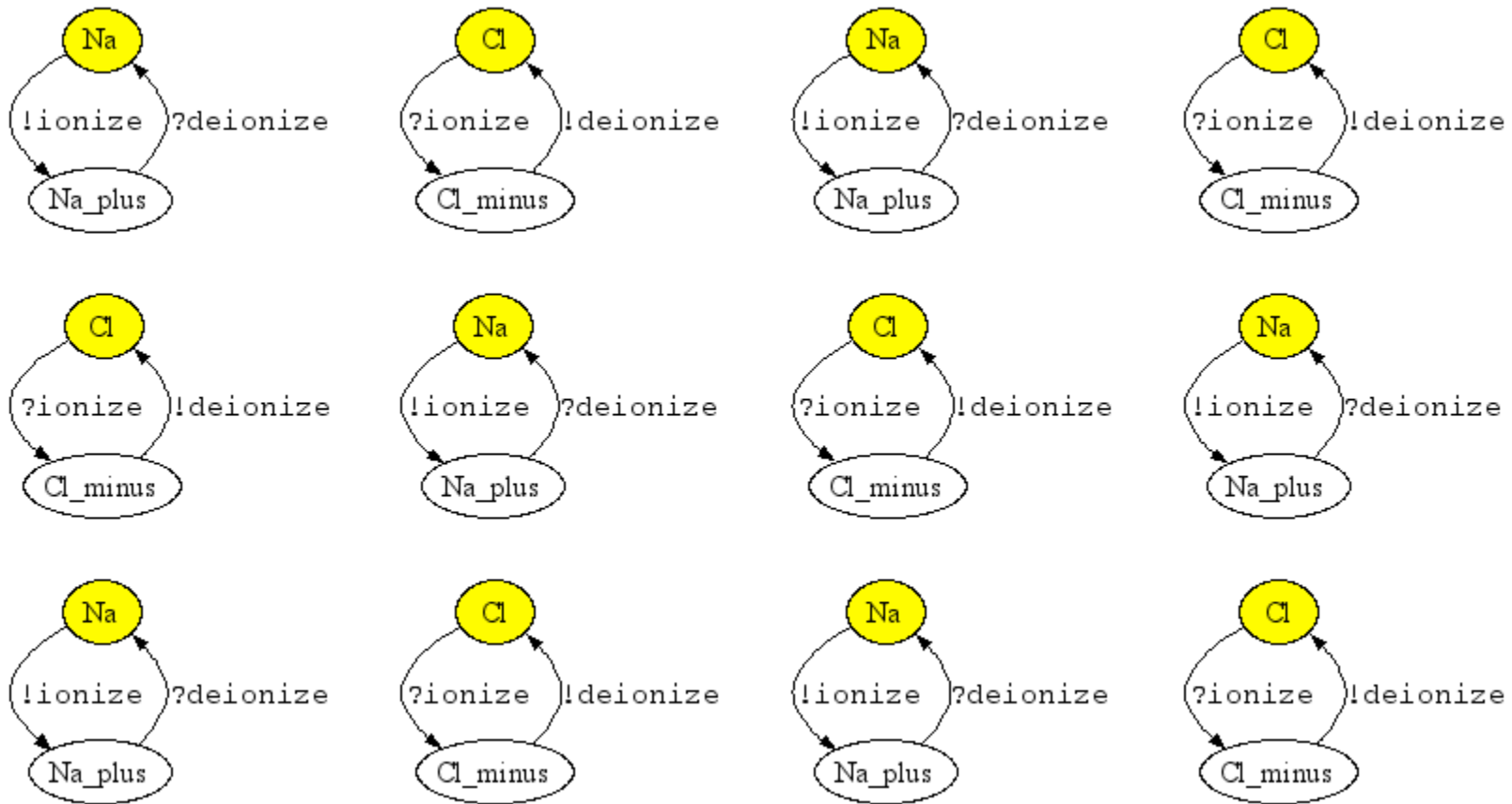
- Cl^- can deionize Na^+ by an output on the *deionize* channel

Ionization:



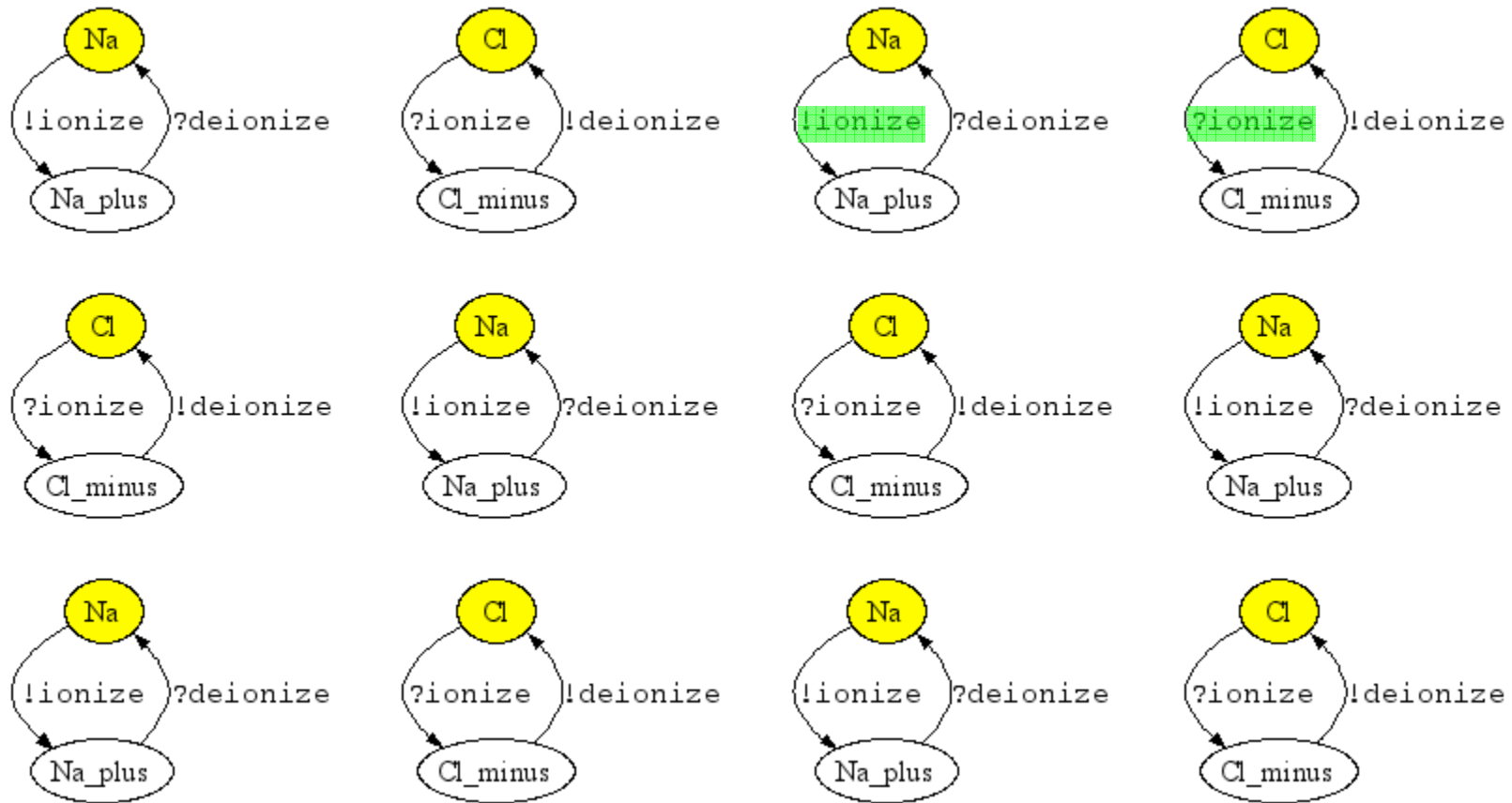
- *Na* and *Cl* are no longer charged

Ionization:



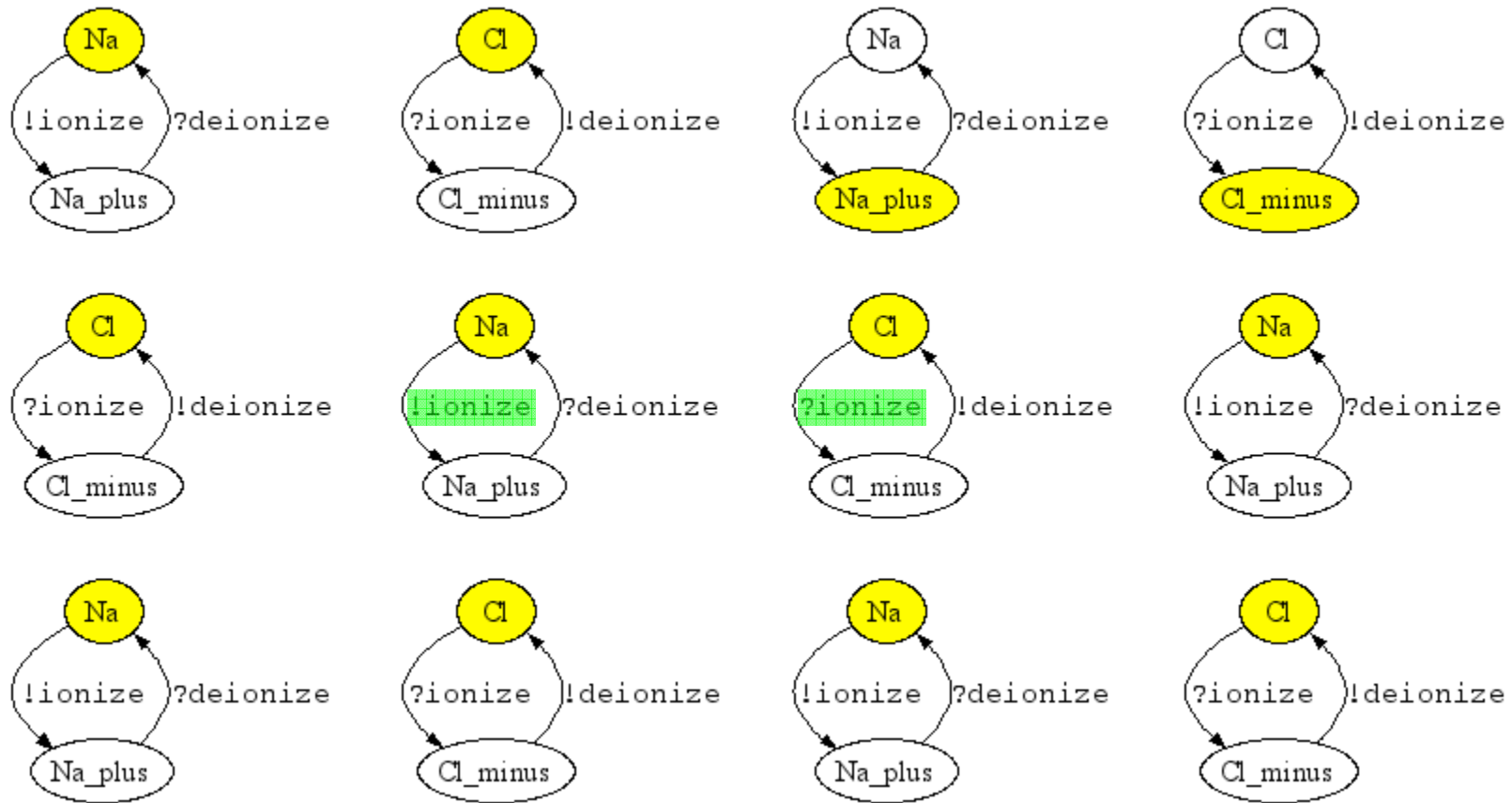
- A number of Na and Cl atoms can be composed in parallel.

Ionization:



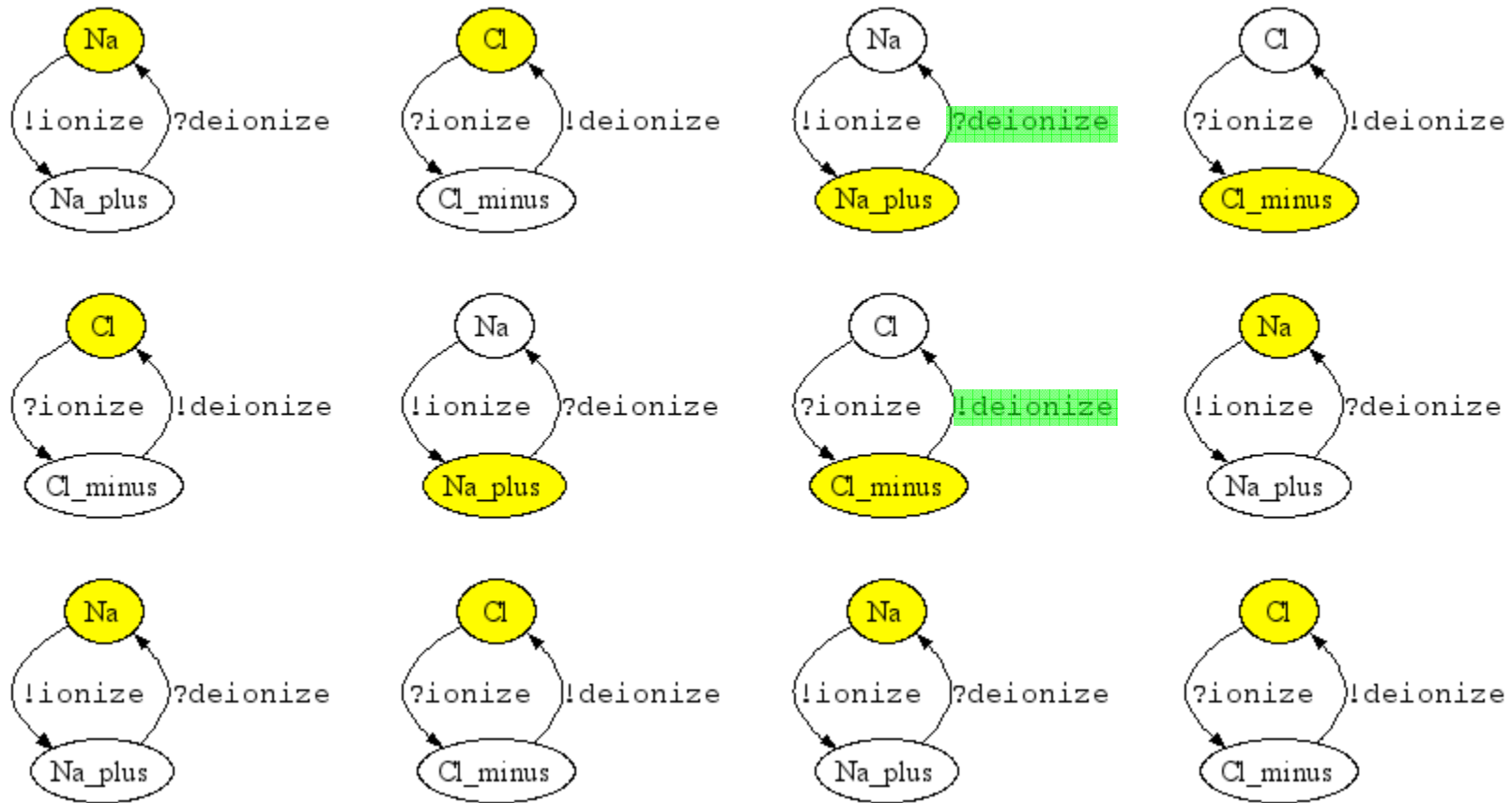
- One of the *Na* atoms can ionize one of the *Cl* atoms.

Ionization:



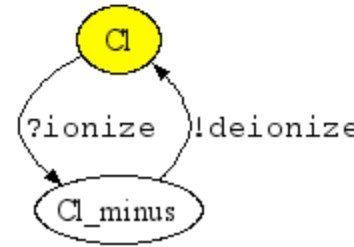
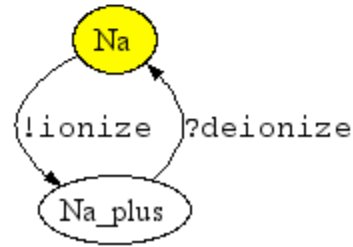
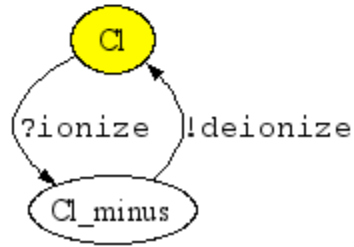
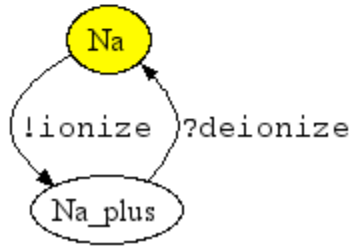
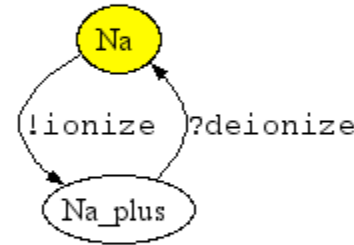
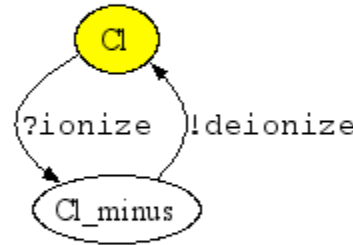
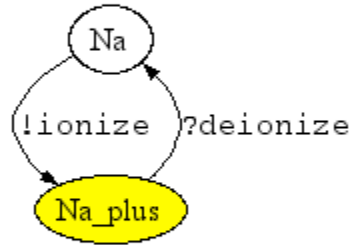
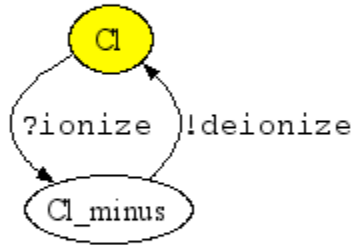
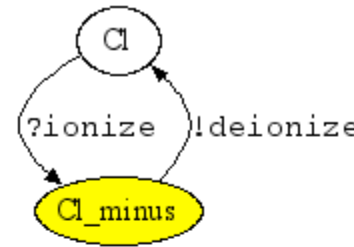
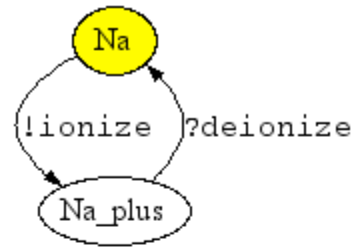
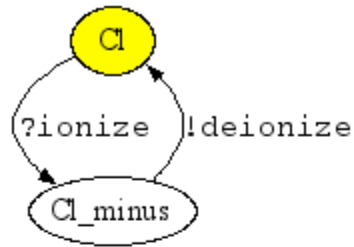
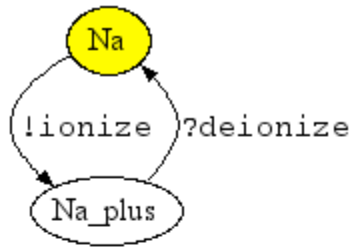
- Additional *Na* and *Cl* atoms can interact in parallel.

Ionization:



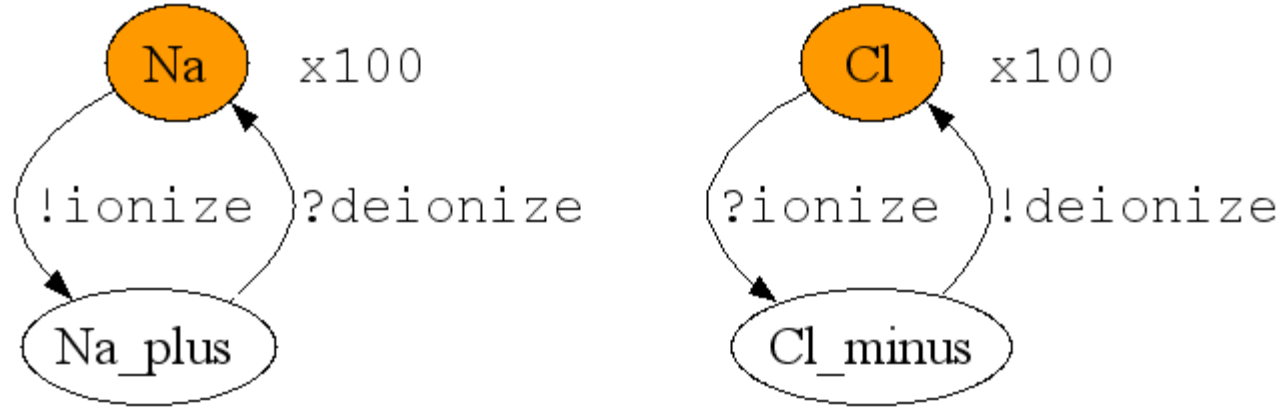
- A Cl^- ion can deionize any of the Na^+ ions.


Ionization:



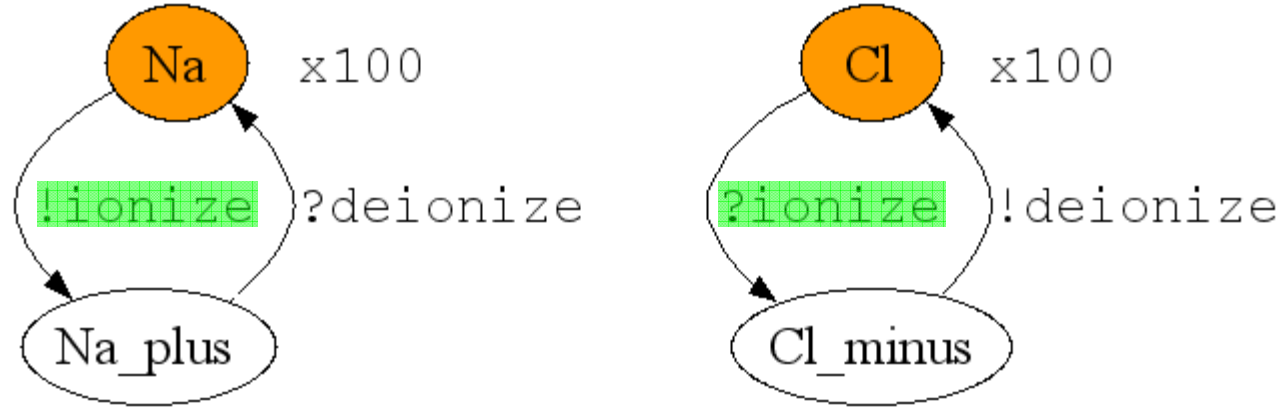
- These reactions can continue indefinitely...

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



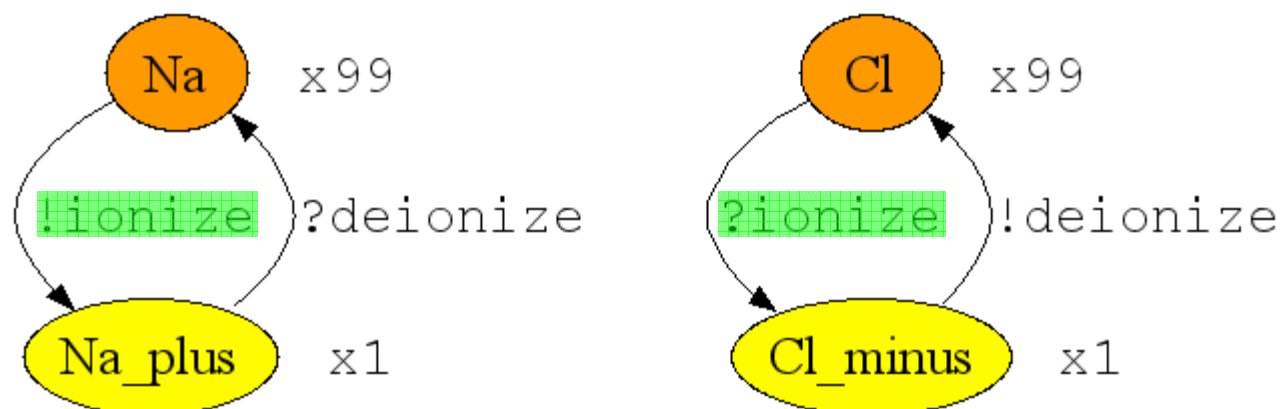
- What happens if we mix $100 \times Na$ and $100 \times Cl$?
- Use a more compact representation.
- The colour is proportional to the number of atoms: 

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



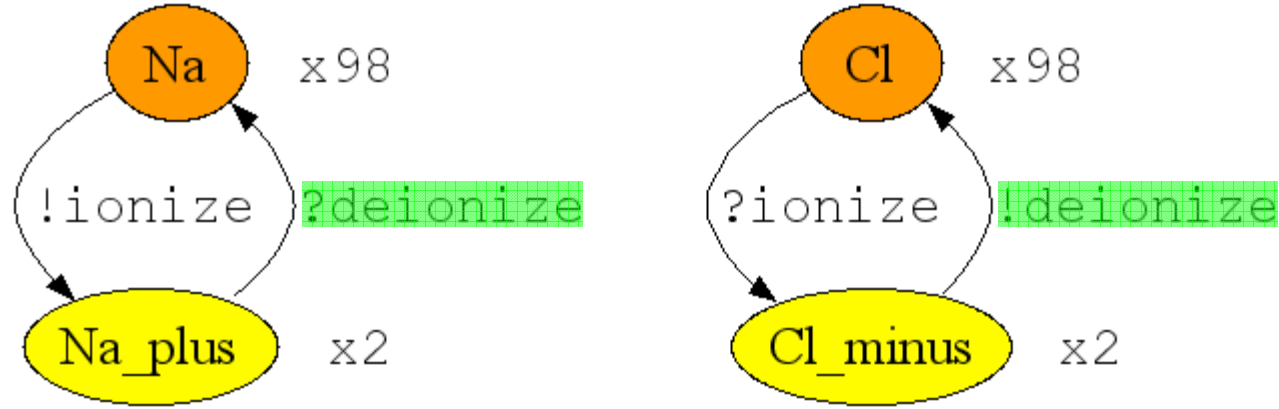
- One of the *Na* atoms can ionize one of the *Cl* atoms.

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



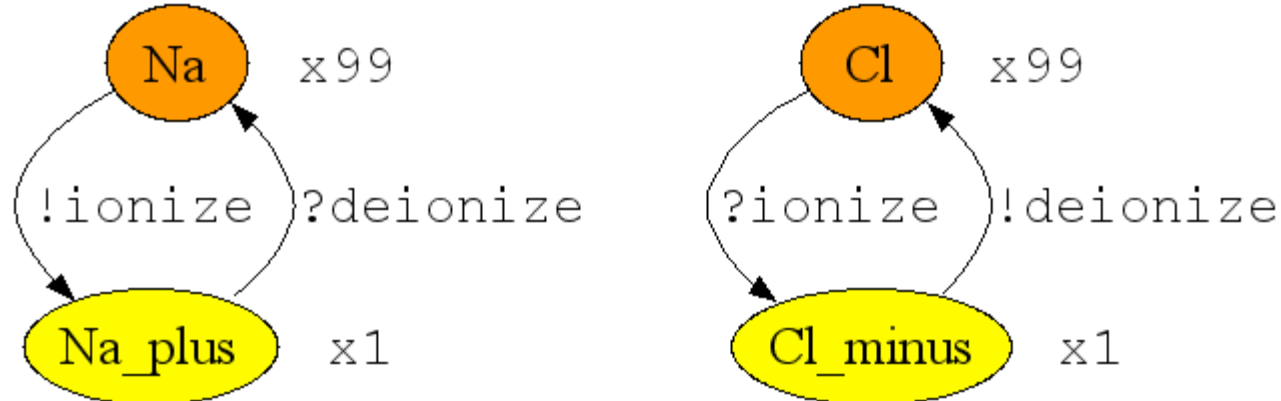
- Additional *Na* and *Cl* atoms can interact in parallel.

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



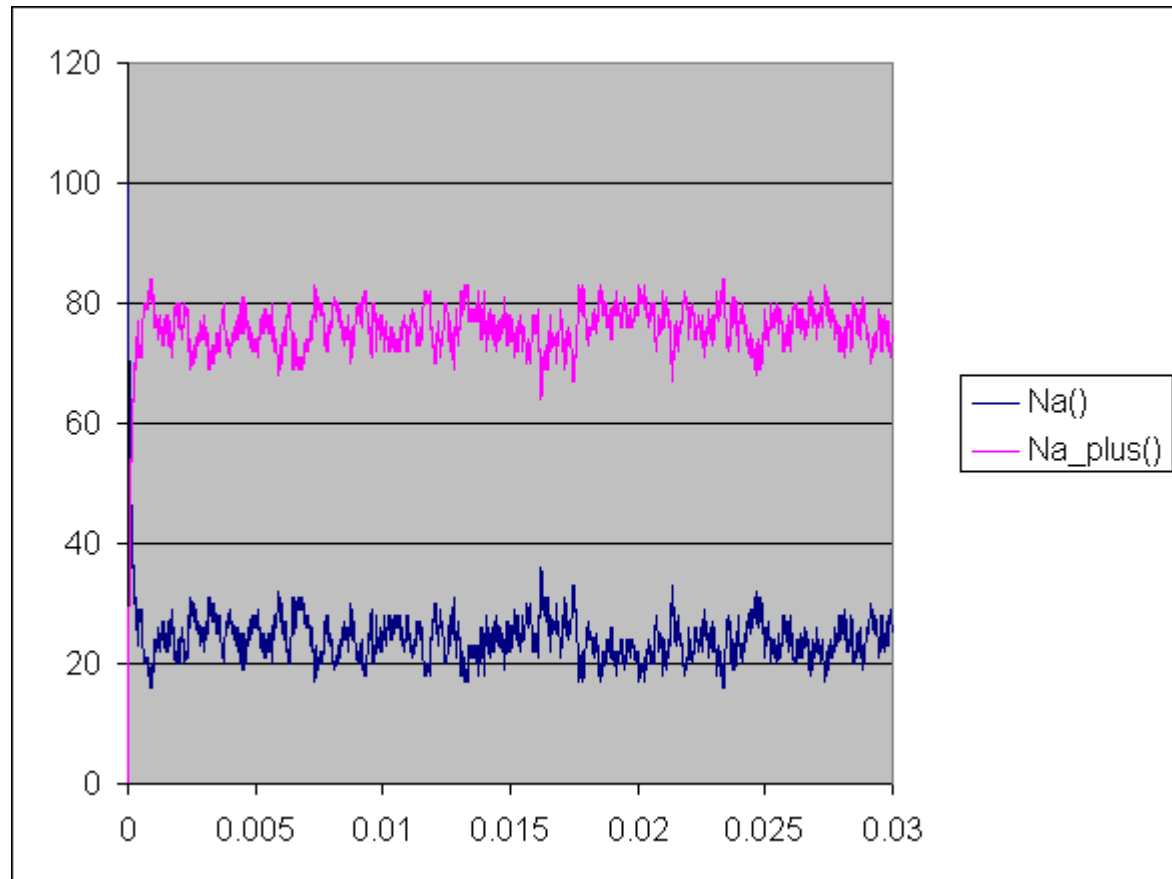
- A Cl^- ion can deionize any of the Na^+ ions.

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



- Eventually an Equilibrium is reached...

Virtual Experiment: $Na + Cl \leftrightarrow Na^+ + Cl^-$



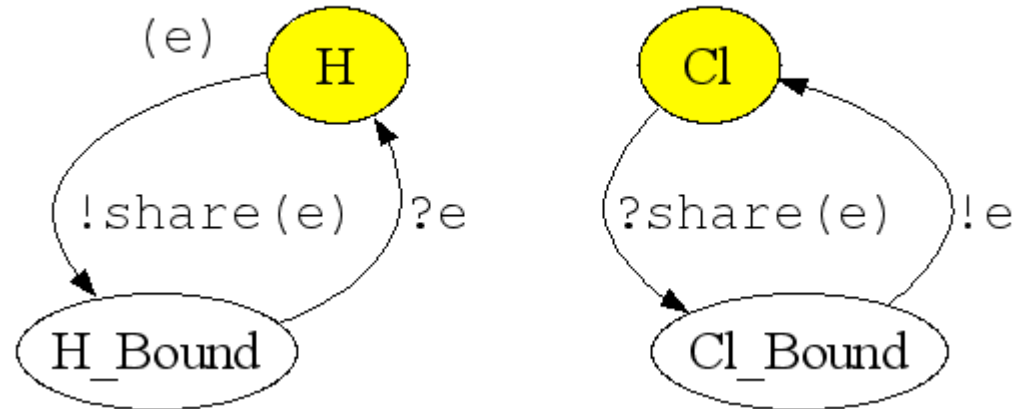
- At equilibrium: $100 \times [Na][Cl] = 10 \times [Na^+][Cl^-]$
- Approximately $76 \times Na$ and $24 \times Na^+$

Binding:



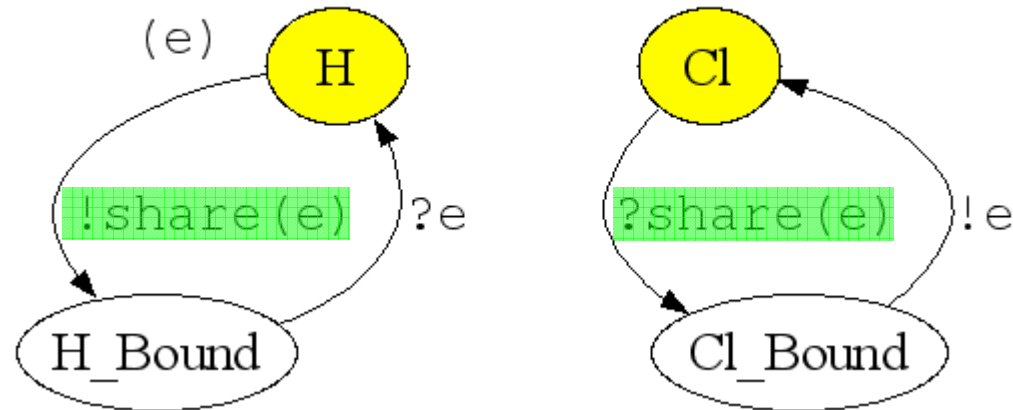
let $H() = \text{new } e@10.0$
 $(!share(e); H_Bound(e))$
and $H_Bound(e) = !e; H()$

let $Cl() = ?share(e); Cl_Bound(e)$
and $Cl_Bound(e) = ?e; Cl()$



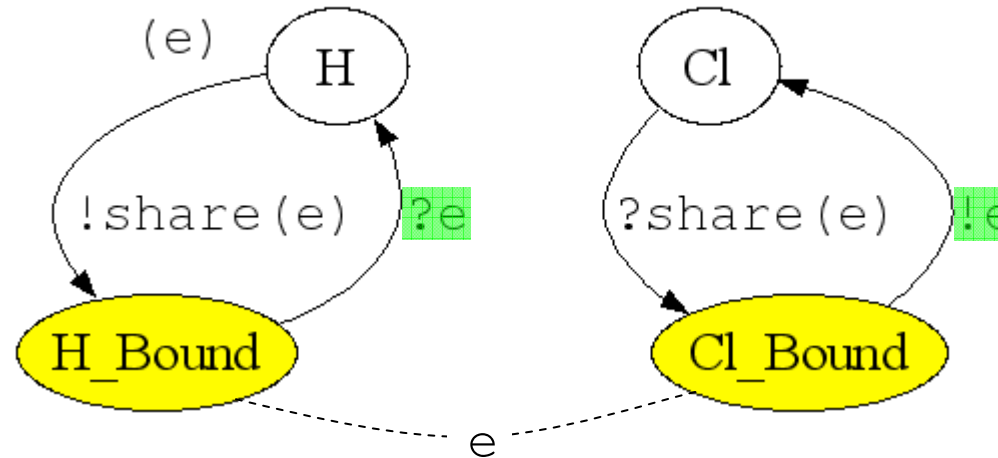
- H has a *private* electron e .
- H can share its electron with Cl at $rate(share) = 100s^{-1}$
- HCl can break its private bond at $rate(e) = 10s^{-1}$

Binding:



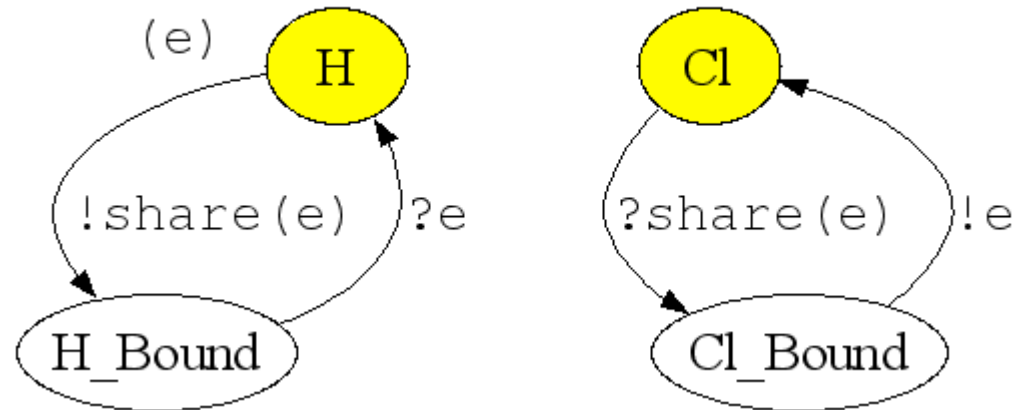
- H can share its electron with Cl on the *share* channel.

Binding:



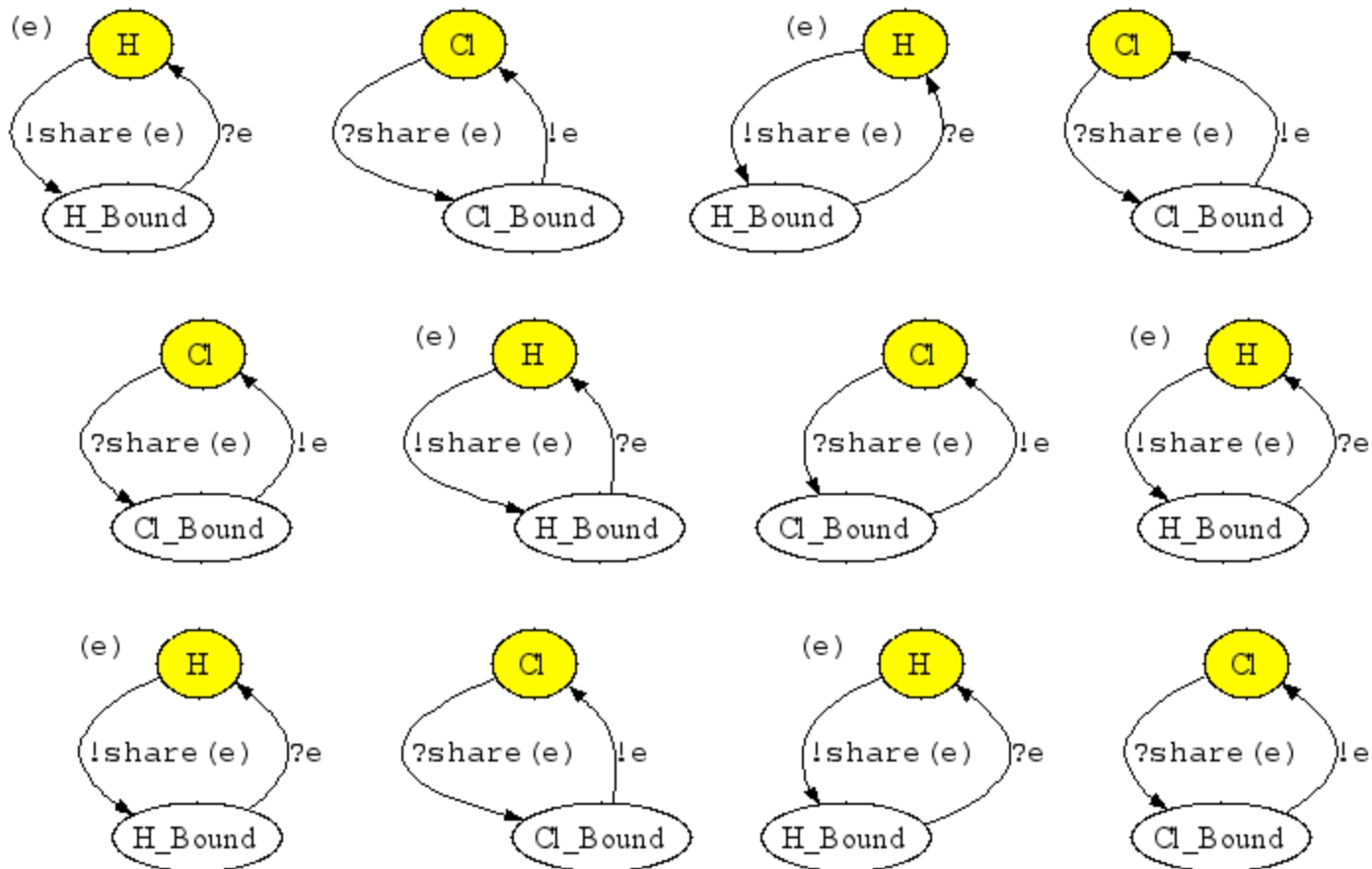
- *HCl* can break its private bond by synchronising on *e*.

Binding:



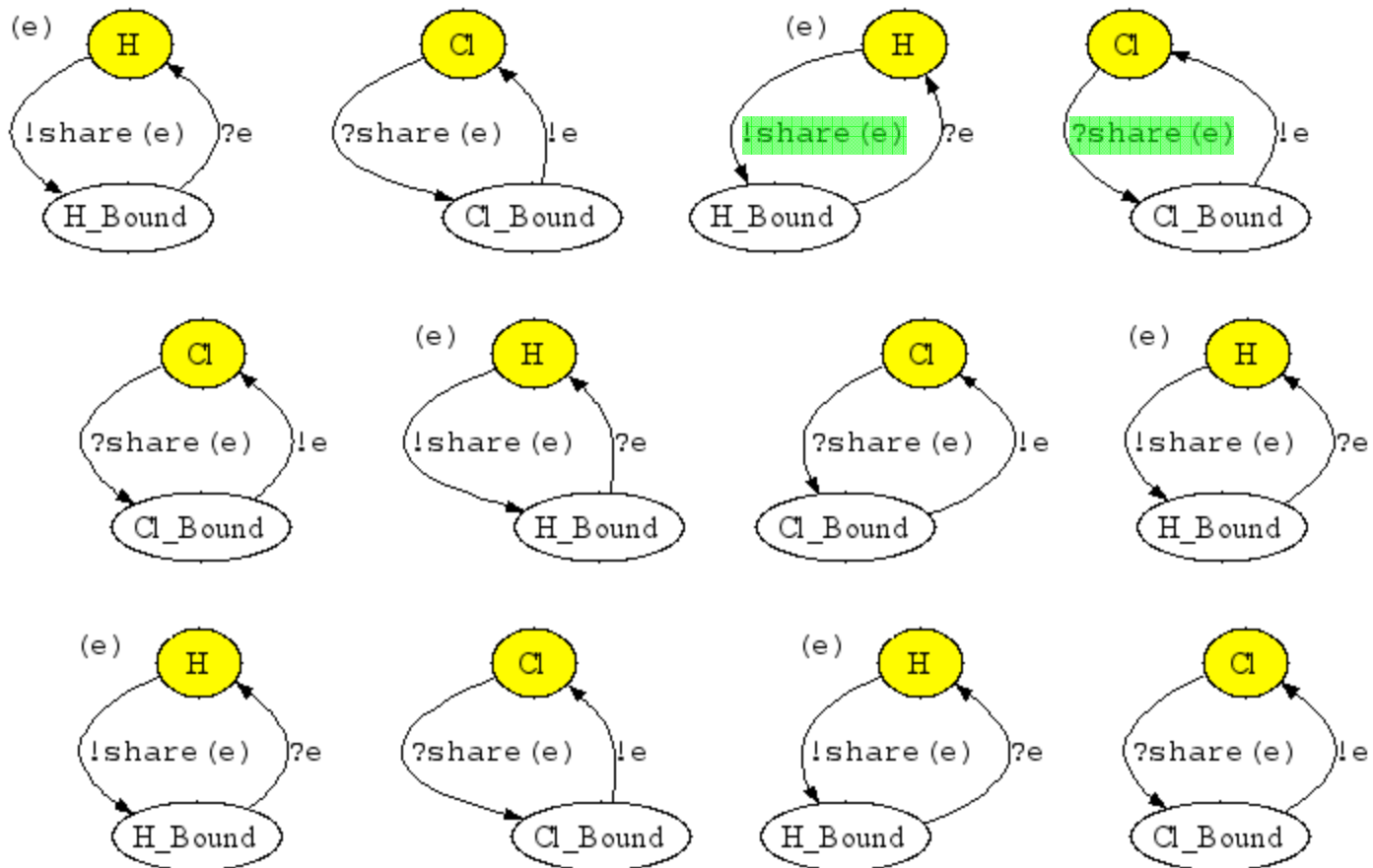
- *H* and *Cl* are no longer bound

Binding:



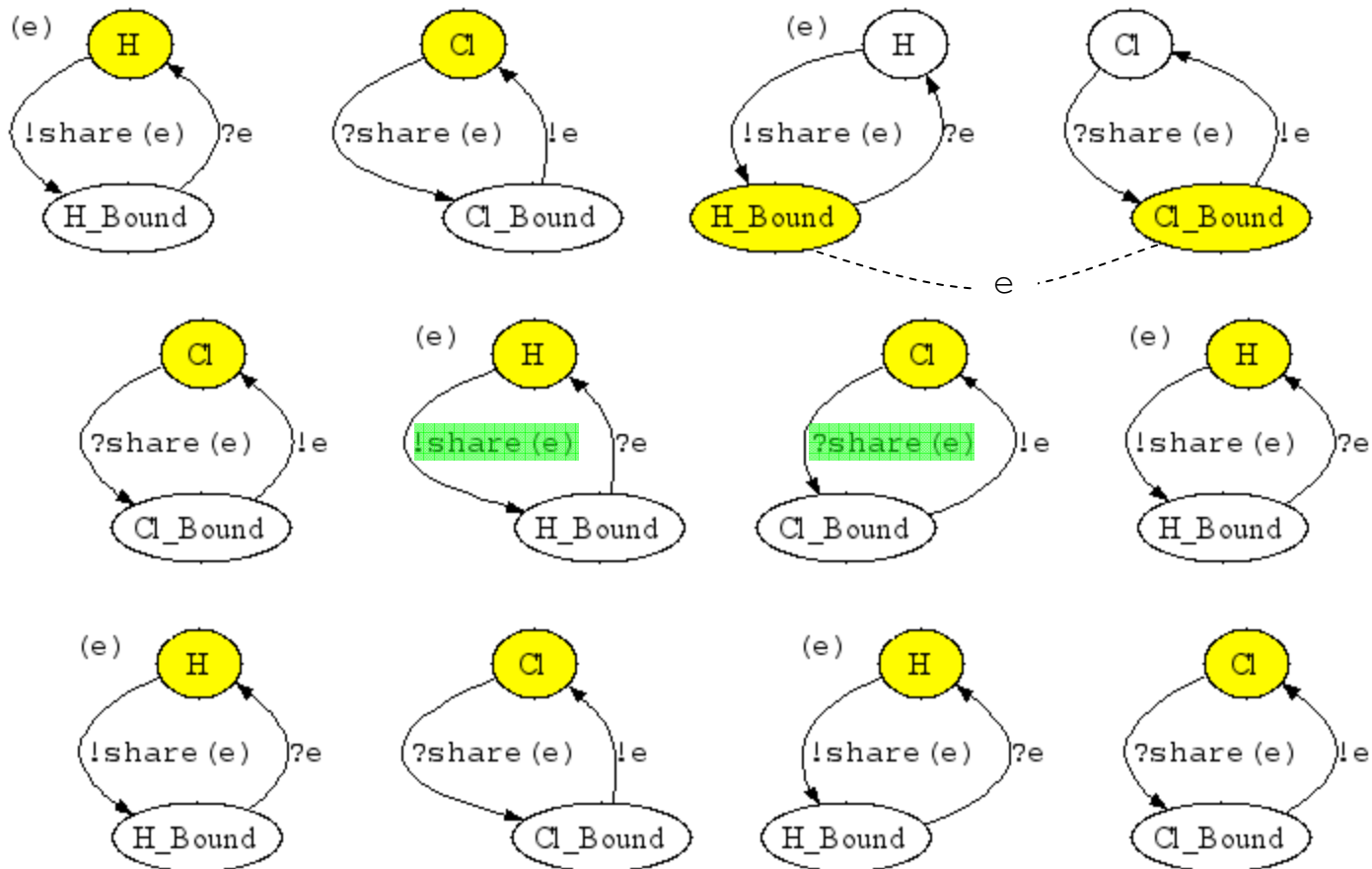
- A number of *H* and *Cl* atoms can be composed in parallel.

Binding:



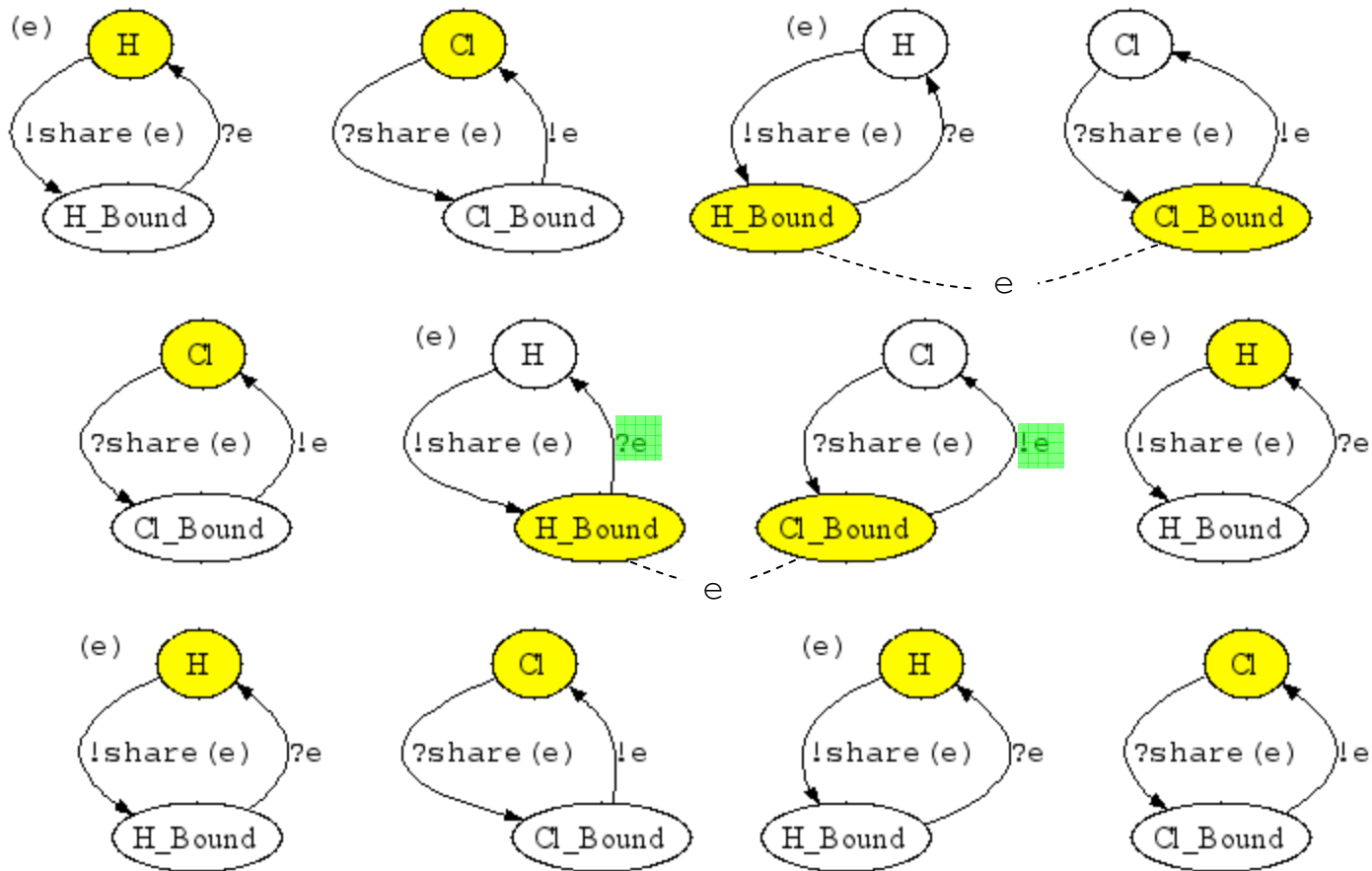
- One of the *H* atoms can bind with one of the *Cl* atoms

Binding:



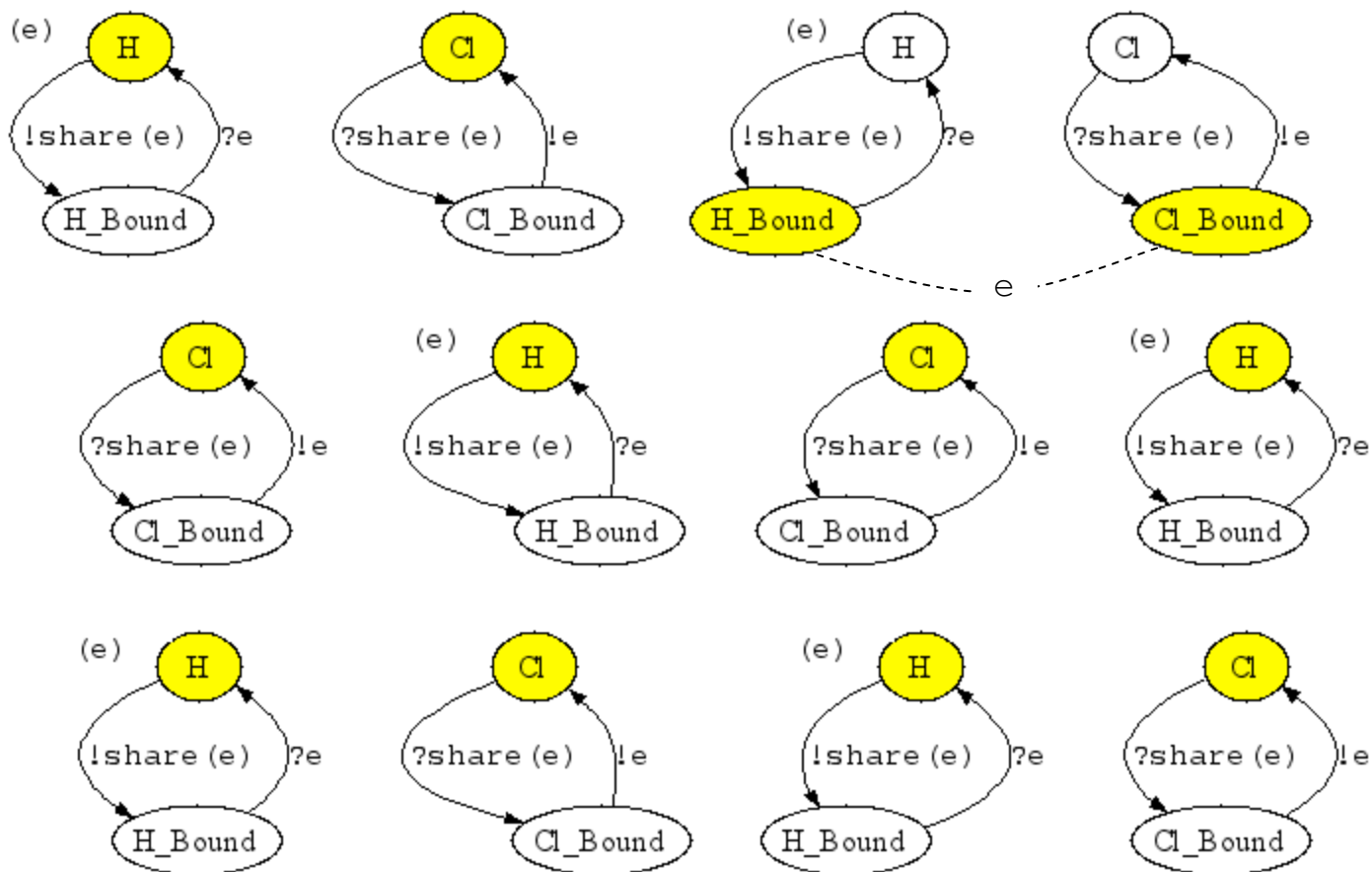
- Additional *H* and *Cl* atoms can bind in parallel.

Binding:



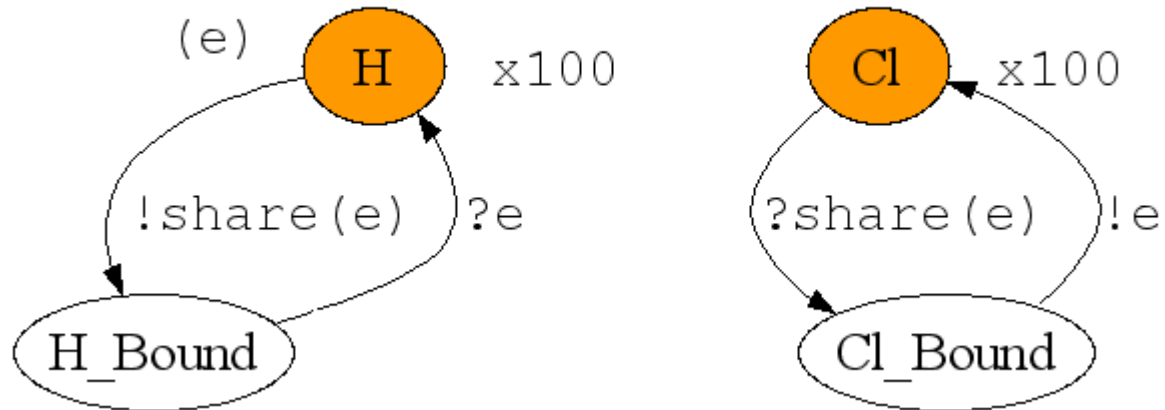
- A single *HCl* molecule can split into *H* and *Cl* atoms.

Binding:



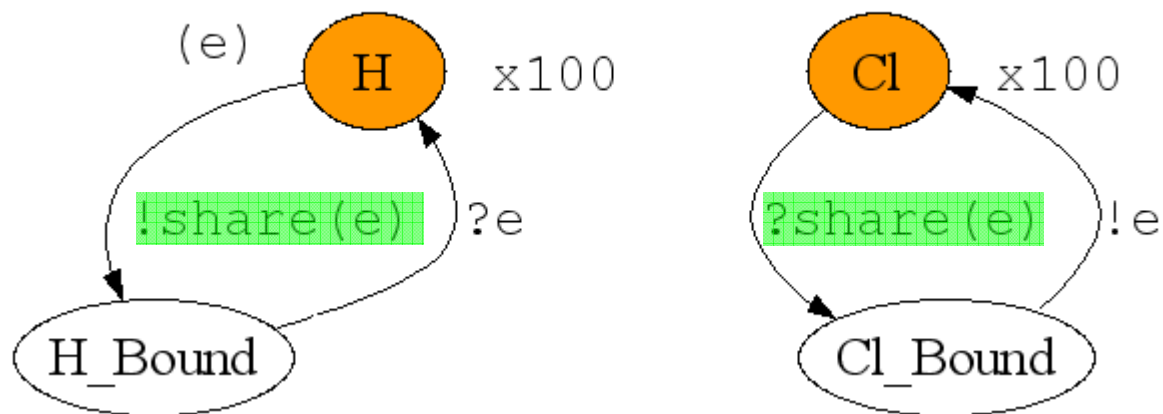
- These reactions can continue indefinitely...

Virtual Experiment: $H + Cl \leftrightarrow HCl$



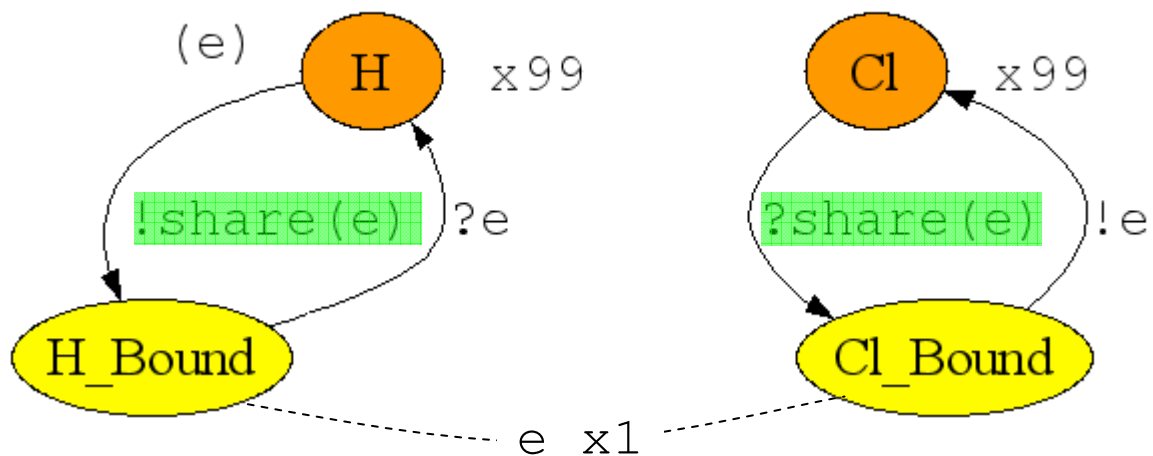
- As with the previous reaction, we mix $100 \times H$ and $100 \times Cl$

Virtual Experiment: $H + Cl \leftrightarrow HCl$



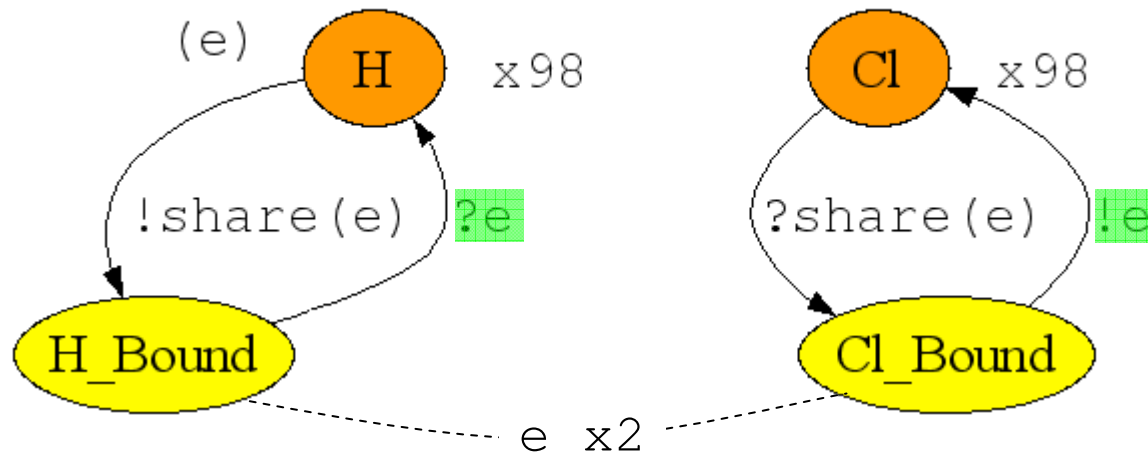
- One of the H atoms can bind with one of the Cl atoms

Virtual Experiment: $H + Cl \leftrightarrow HCl$



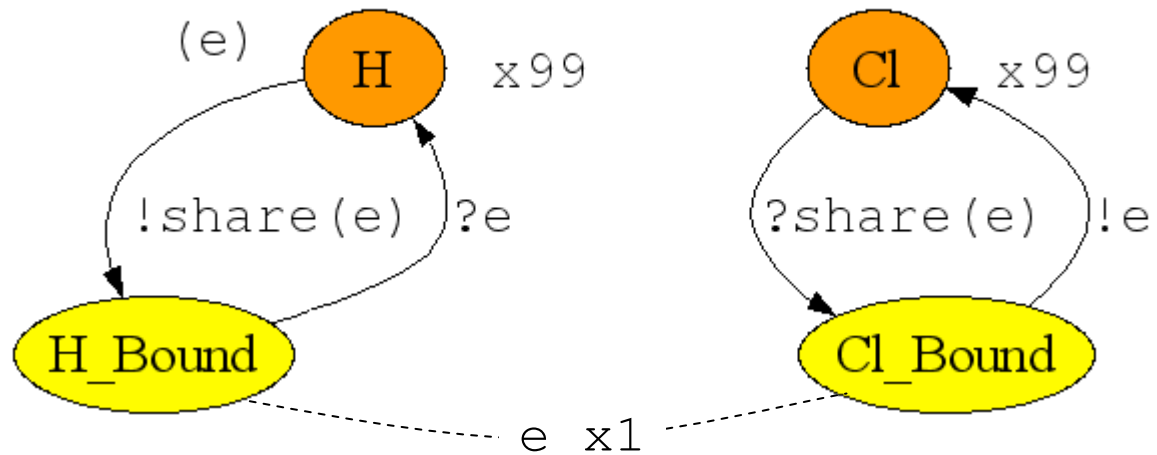
- Additional H and Cl atoms can bind in parallel.

Virtual Experiment: $H + Cl \leftrightarrow HCl$



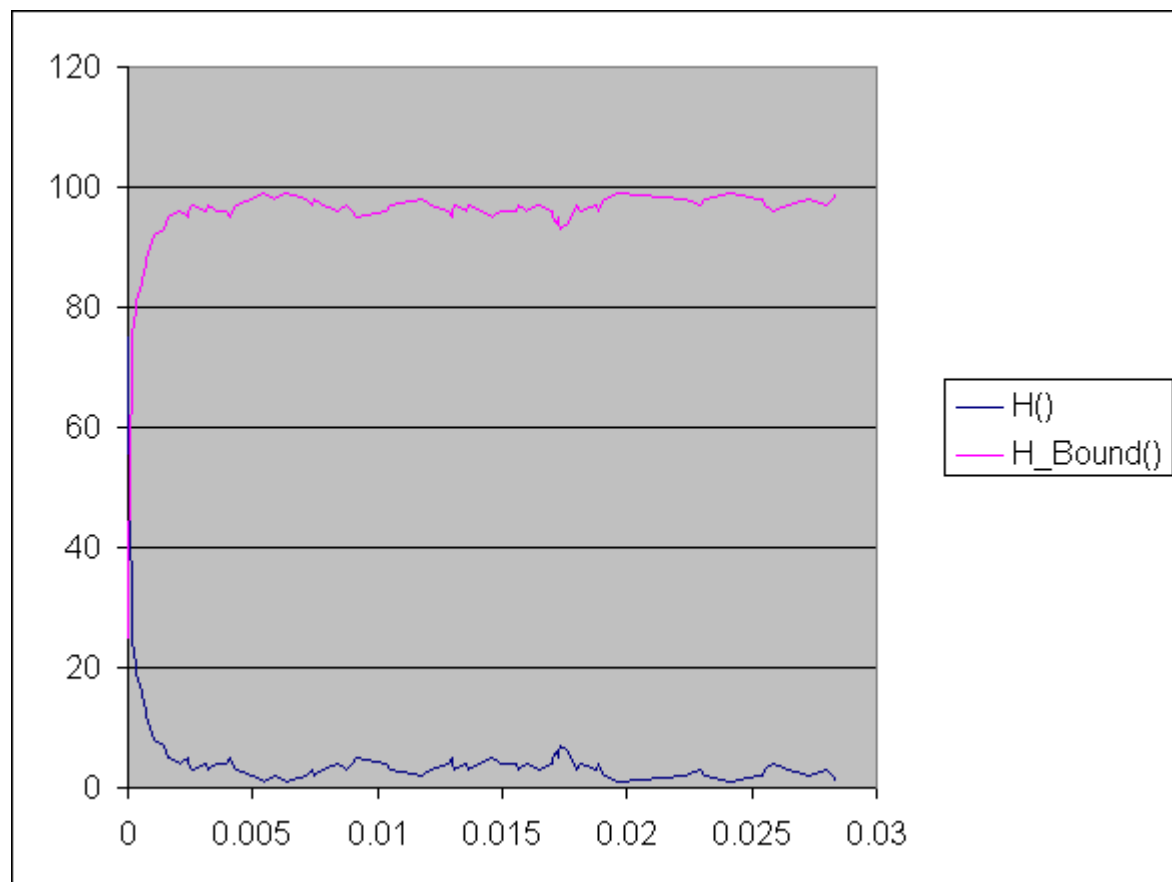
- A single HCl molecule can split into H and Cl atoms.

Virtual Experiment: $H + Cl \leftrightarrow HCl$



- Eventually an Equilibrium is reached...

Virtual Experiment: $H + Cl \leftrightarrow HCl$

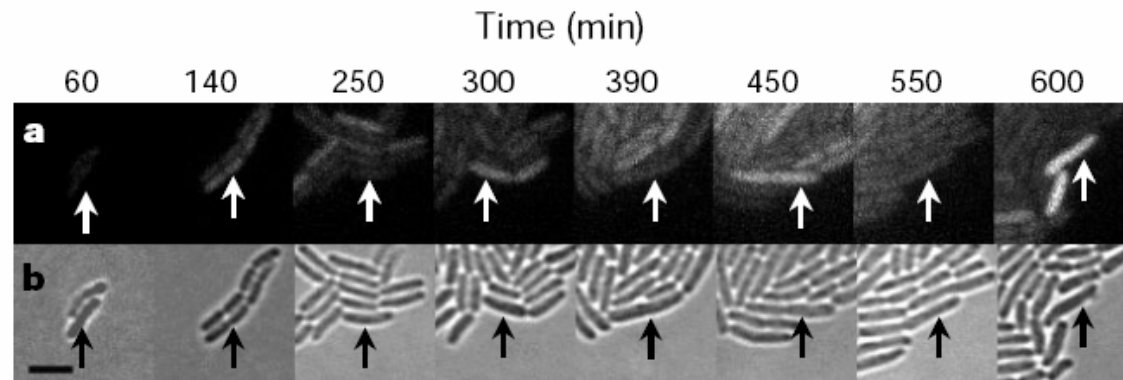
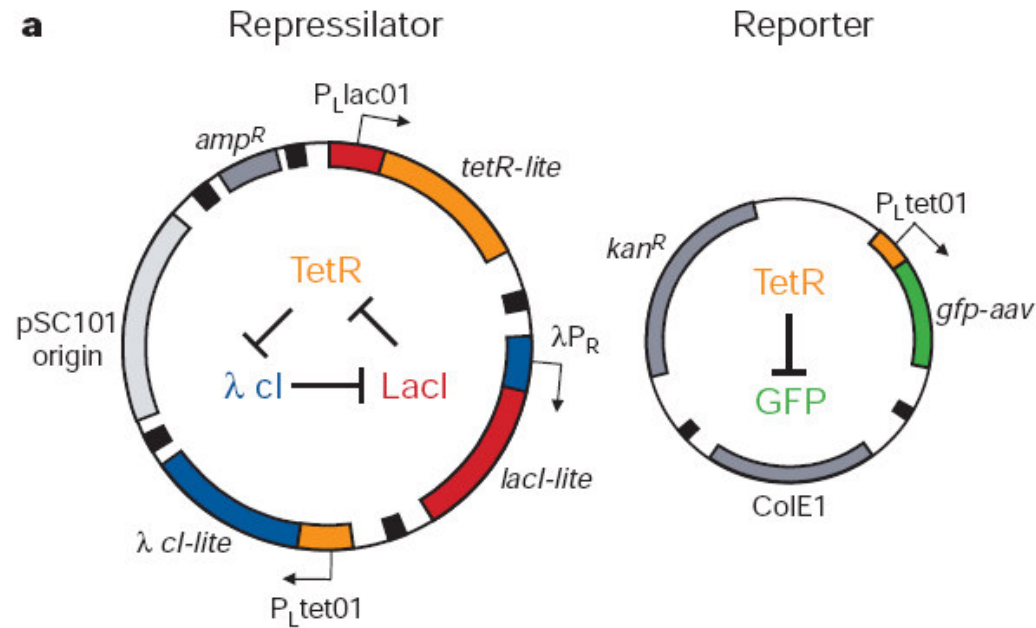


- At equilibrium: $100 \times [H][Cl] = 10 \times [HCl]$
- Approximately $3 \times H$ and $97 \times HCl$

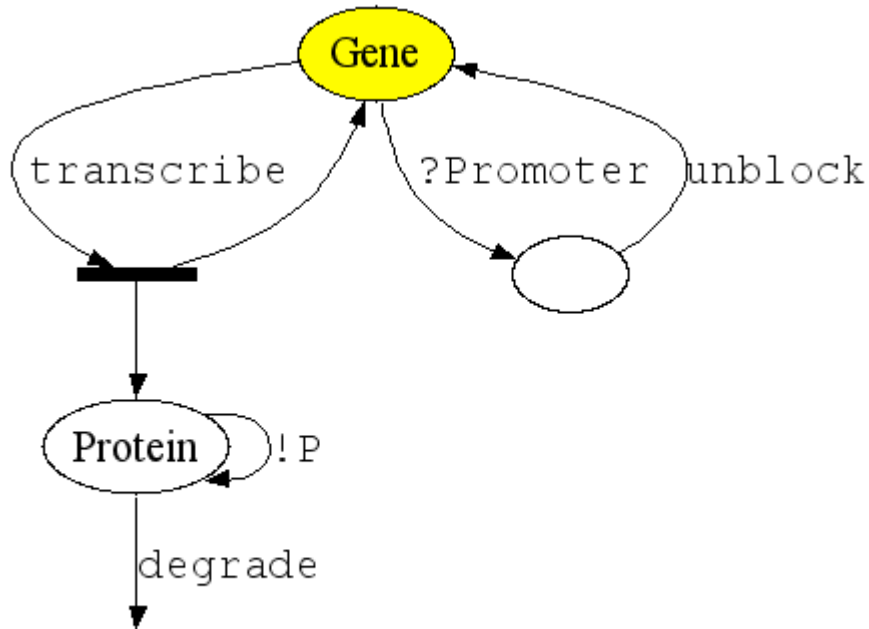
Gene Networks Engineered in Vivo

with Luca Cardelli (Microsoft Research)
and Ralf Blossey (IRI Lille)

Repressilator [Elowitz and Leibler, 2000]



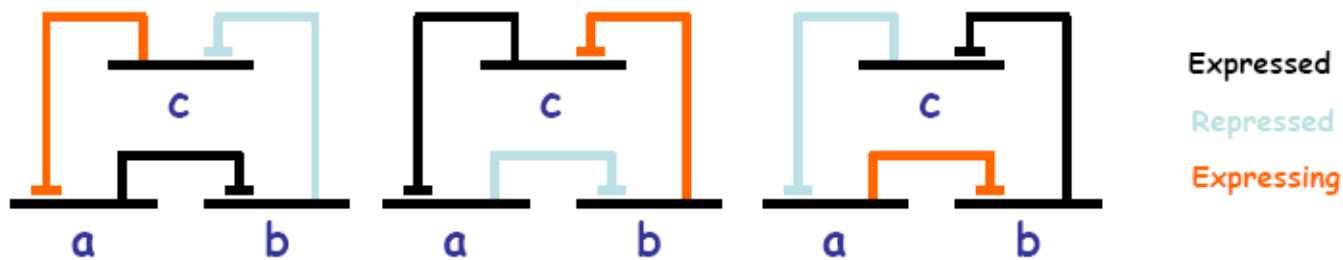
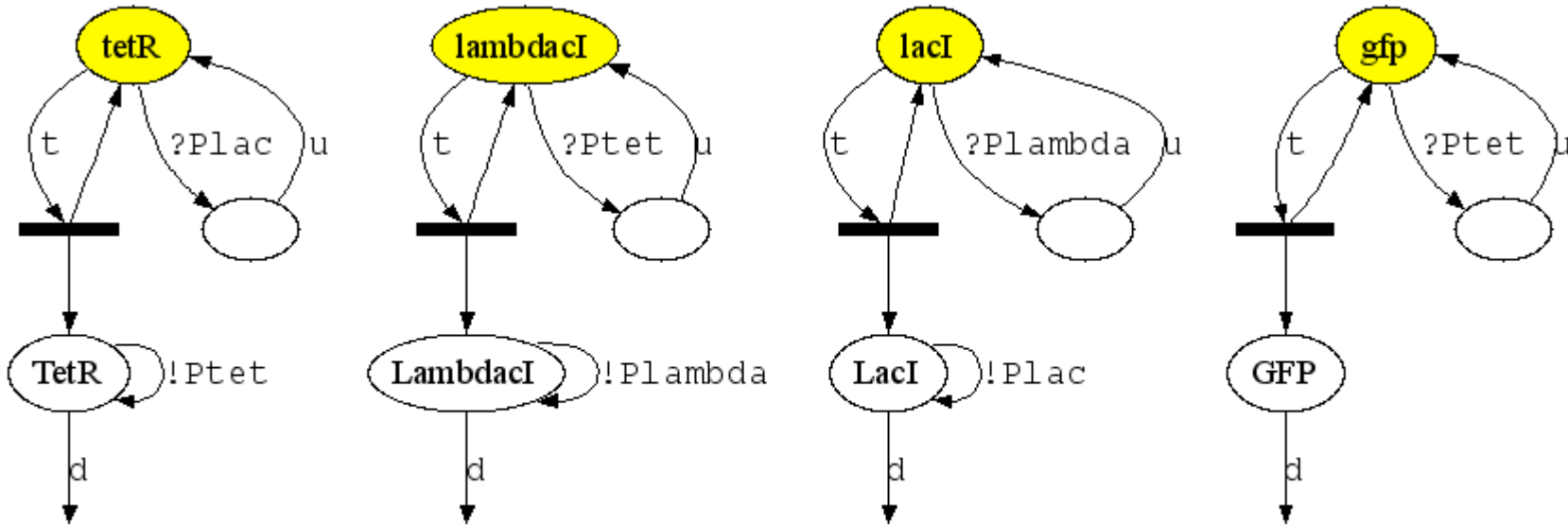
Repressilator: *Genes*



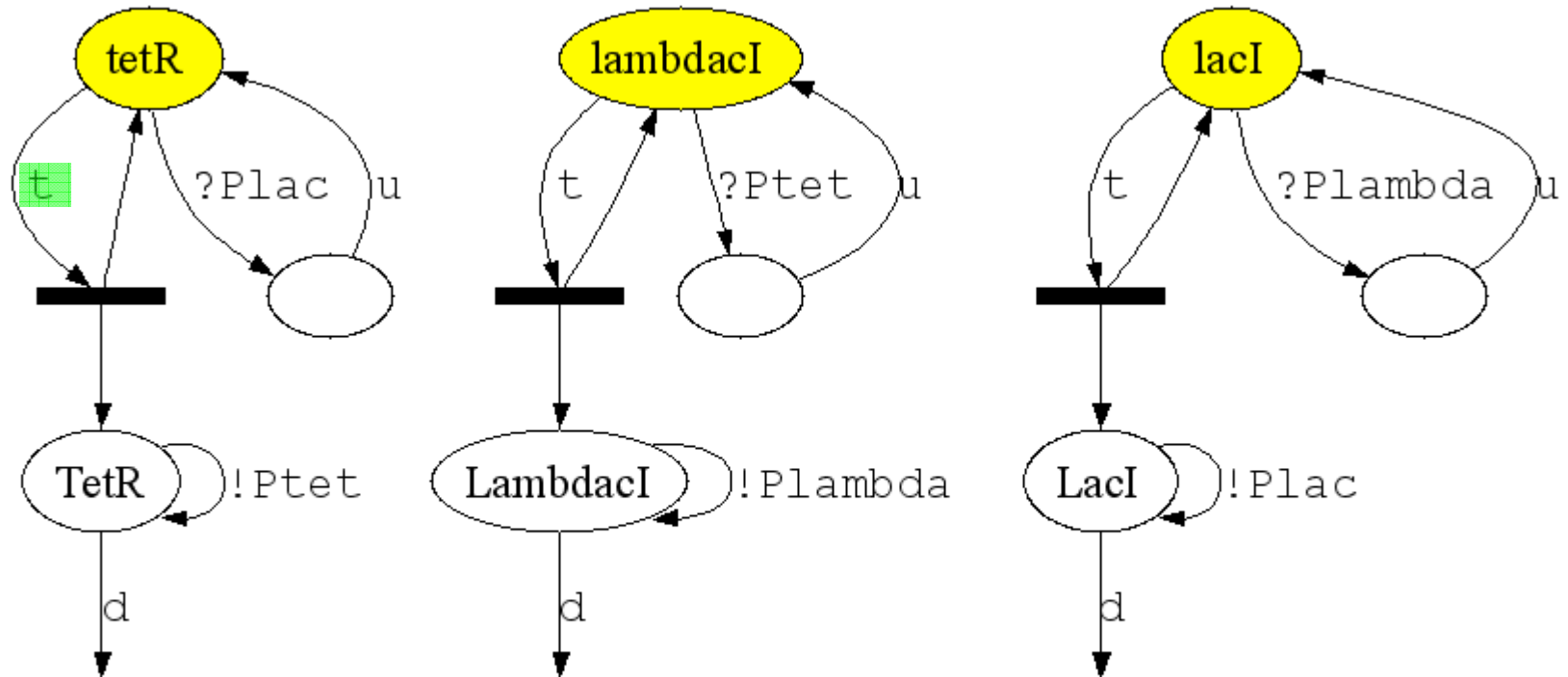
```
let Gene() =  
  do delay@transcribe;  
    (Protein() | Gene())  
  or ?Promoter;  
    delay@unblock; Gene()  
and Protein() =  
  do !P; Protein()  
  or delay@degrade
```

```
val transcribe = 0.1  
val degrade = 0.001  
val unblock = 0.0001  
val bind = 1.0
```

Repressilator: Network

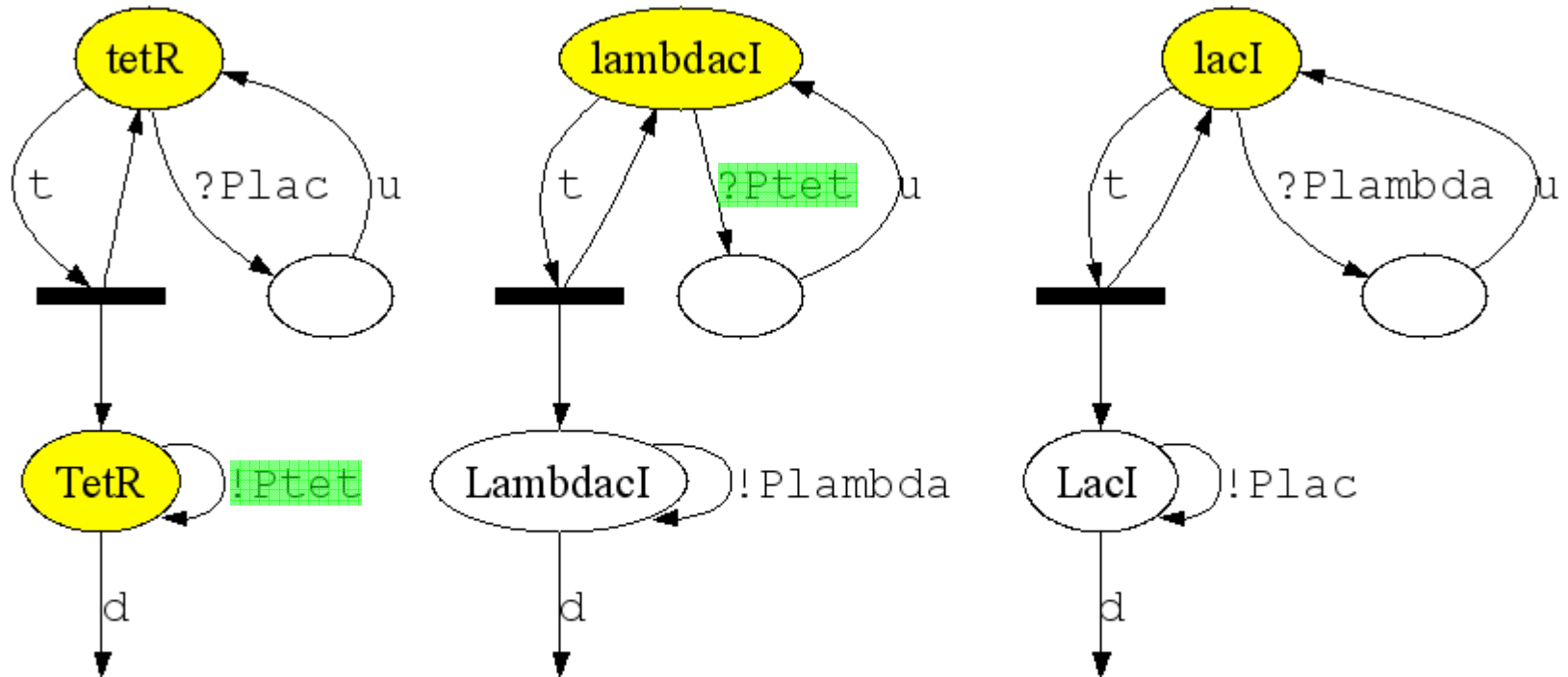


Repressilator: Oscillations



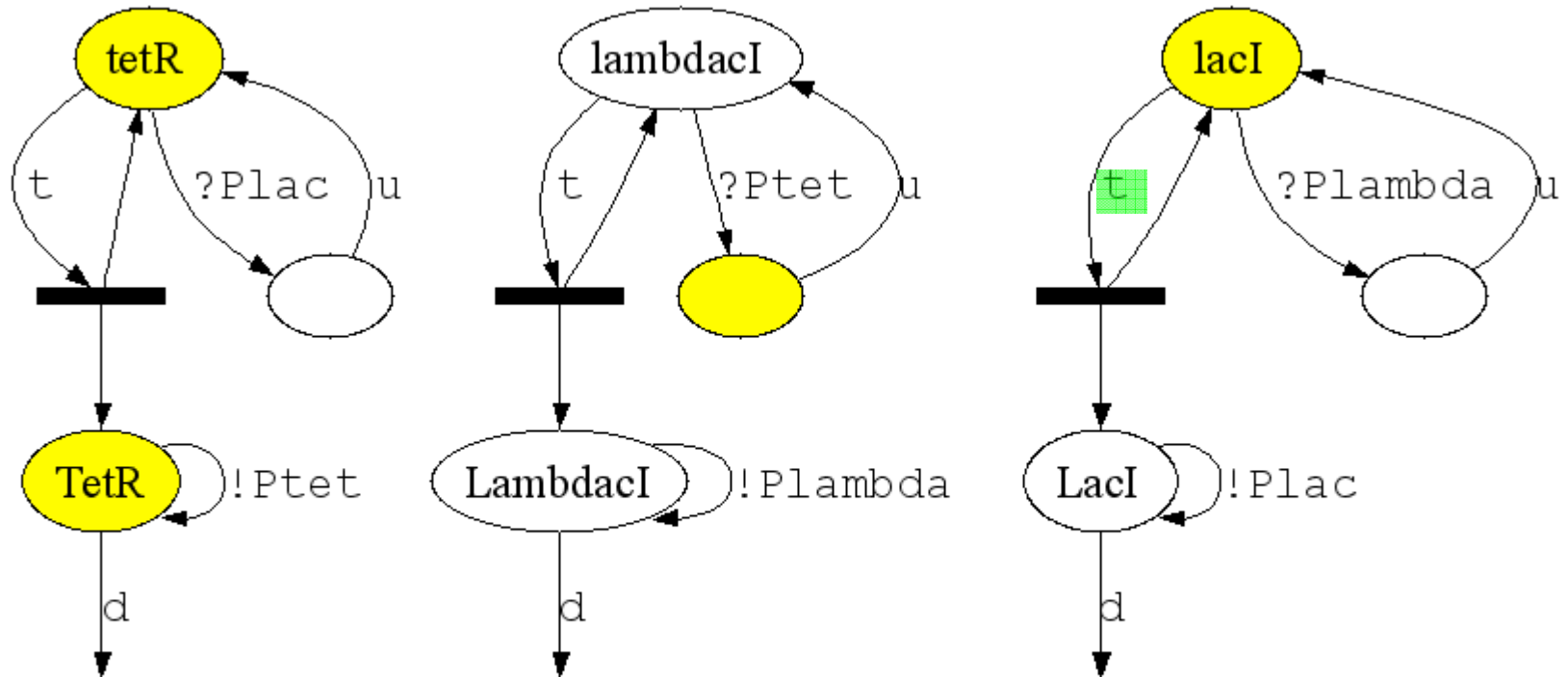
Any of the genes can transcribe. Supposing tet starts first.

Repressilator: Oscillations



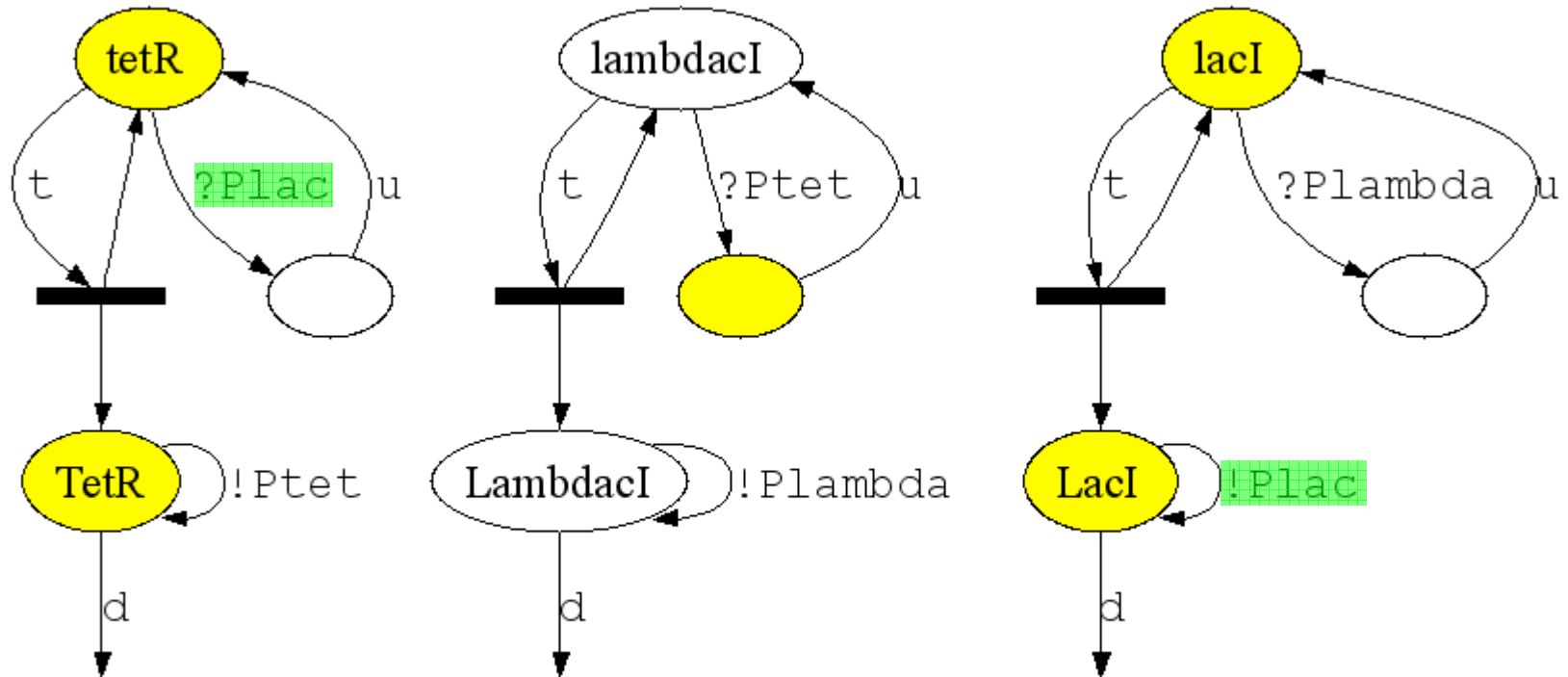
The Tet protein can inhibit the lambda gene

Repressilator: Oscillations



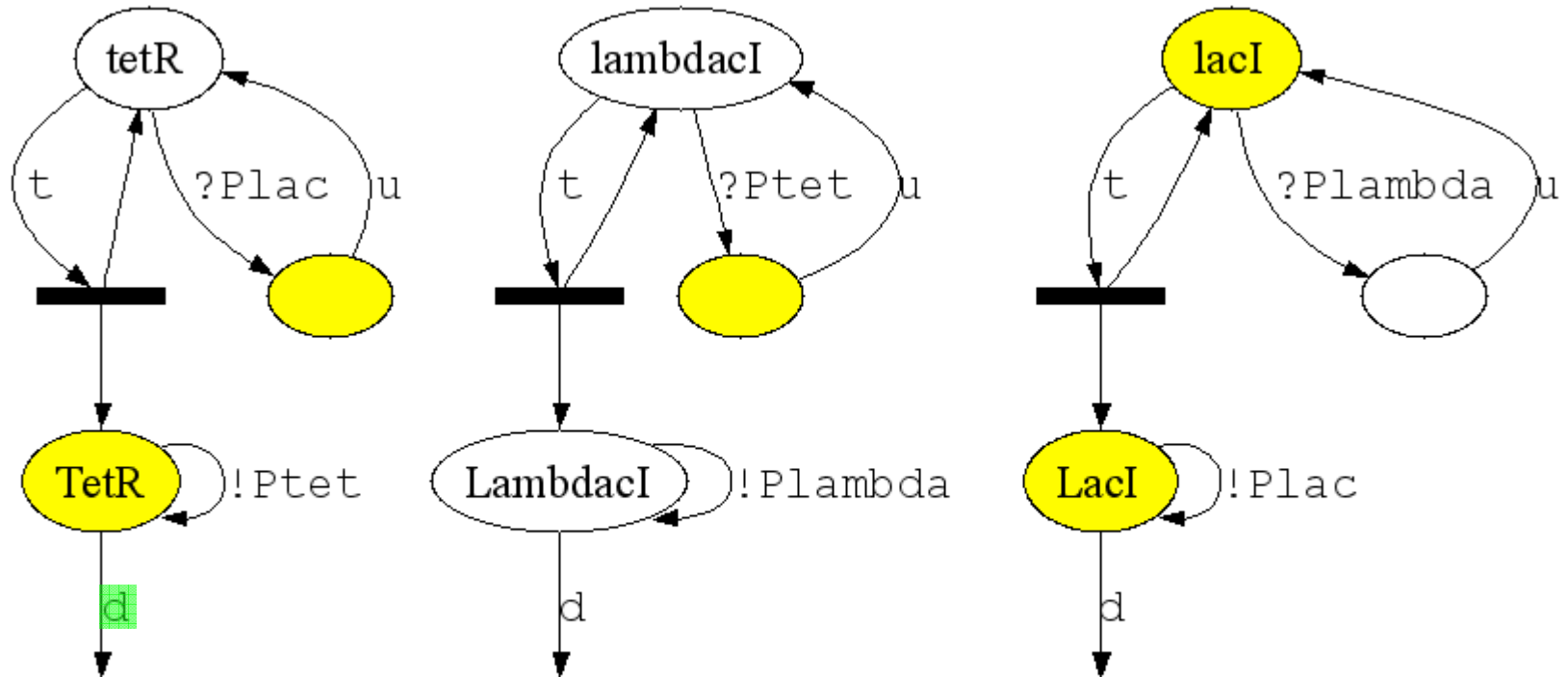
Lambda is now blocked, but lac can still transcribe.

Repressilator: Oscillations



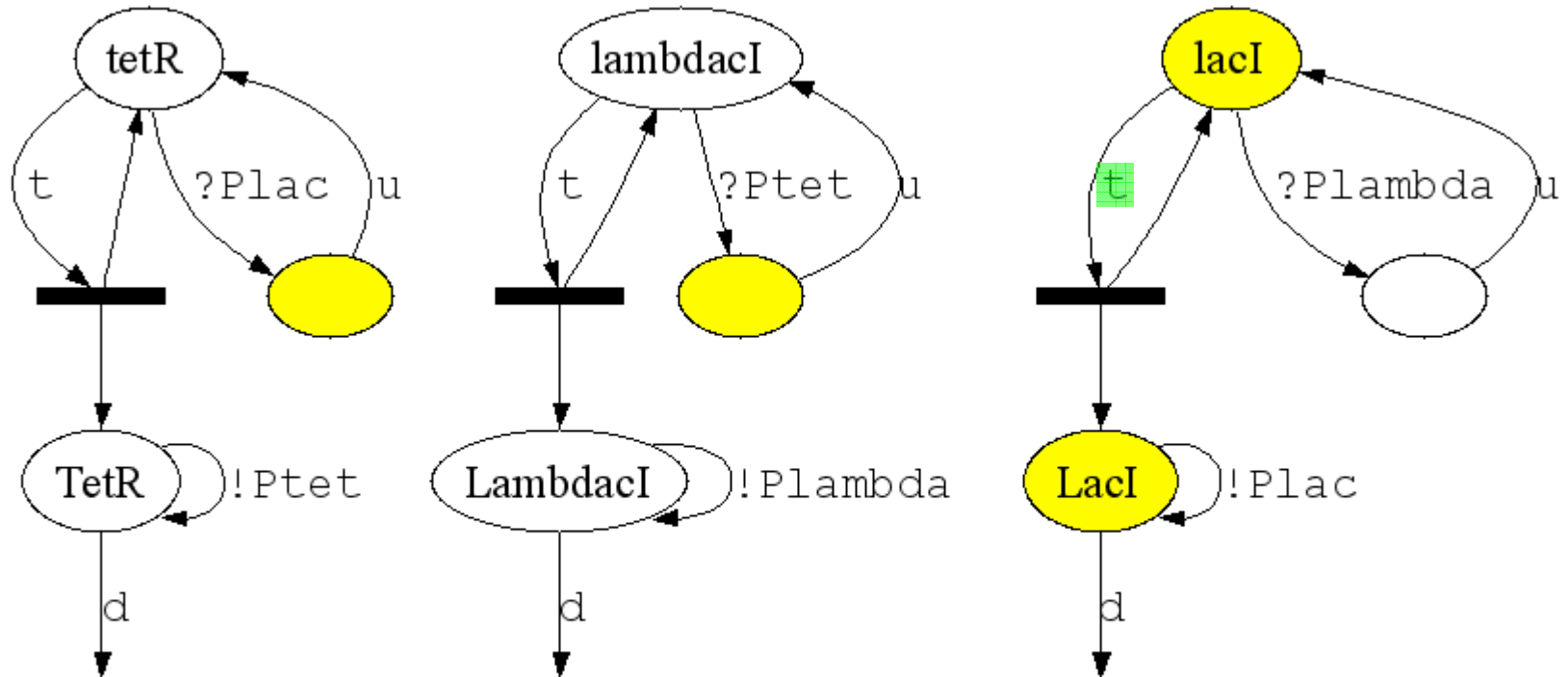
The Lac protein can inhibit the tet gene.

Repressilator: Oscillations



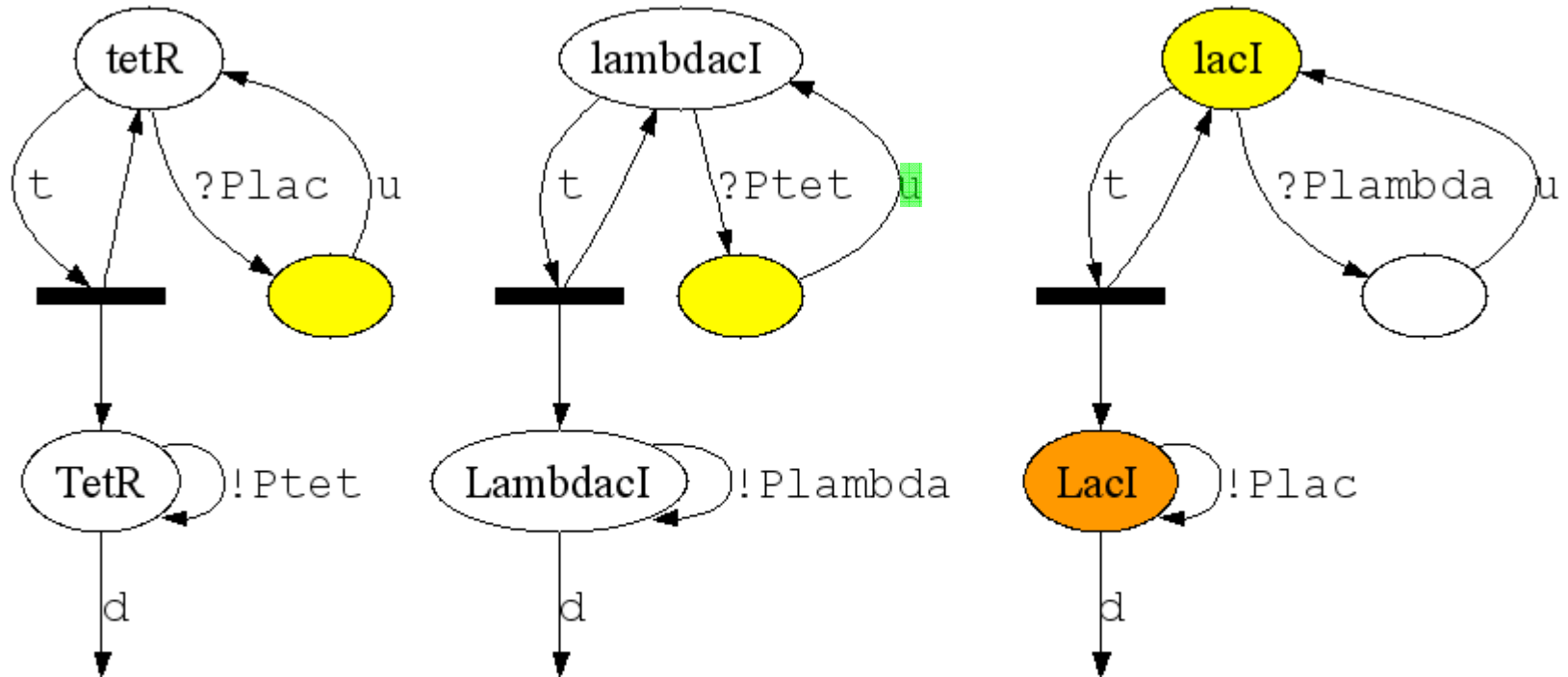
Both the tet and lambda genes are blocked. Tet can degrade.

Repressilator: Oscillations



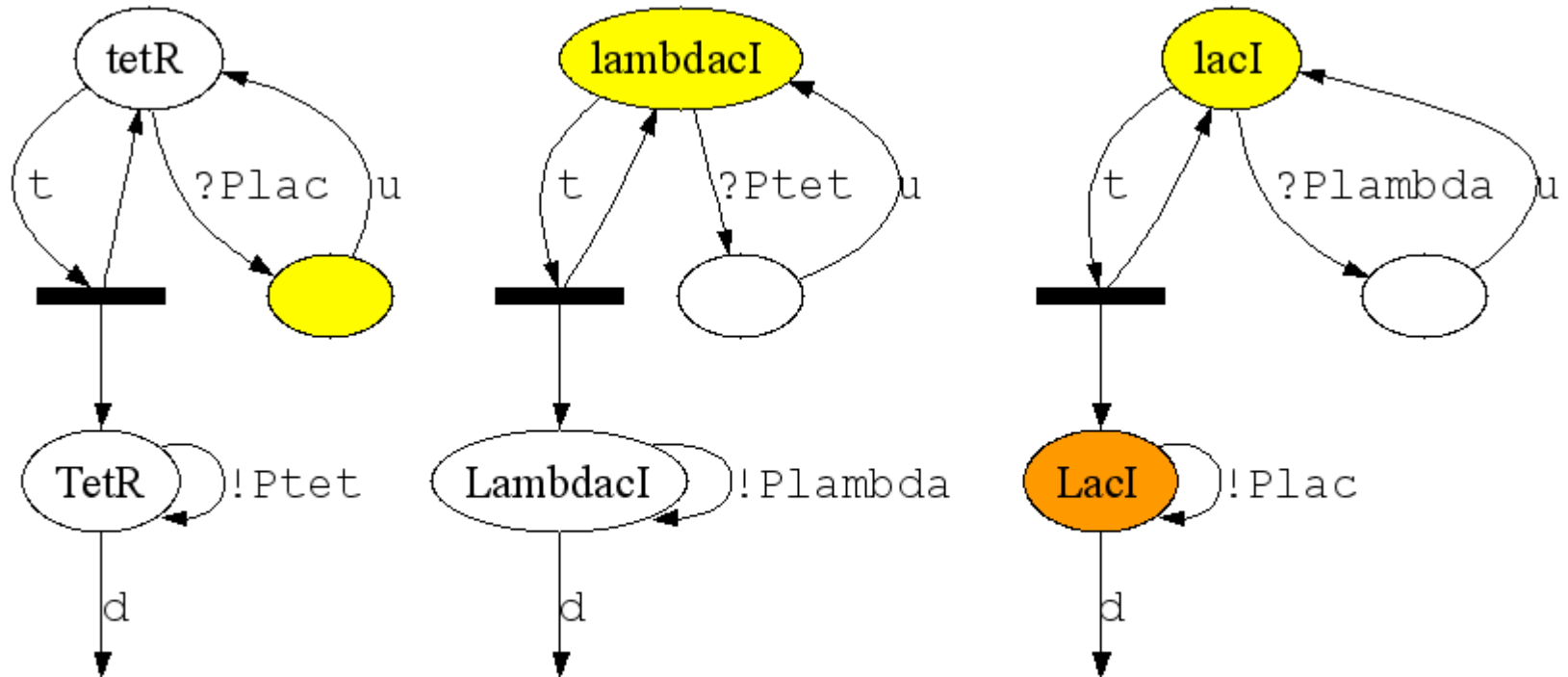
Now only the lac gene can transcribe

Repressilator: Oscillations



A large amount of Lac protein is produced, but lambda unblocks.

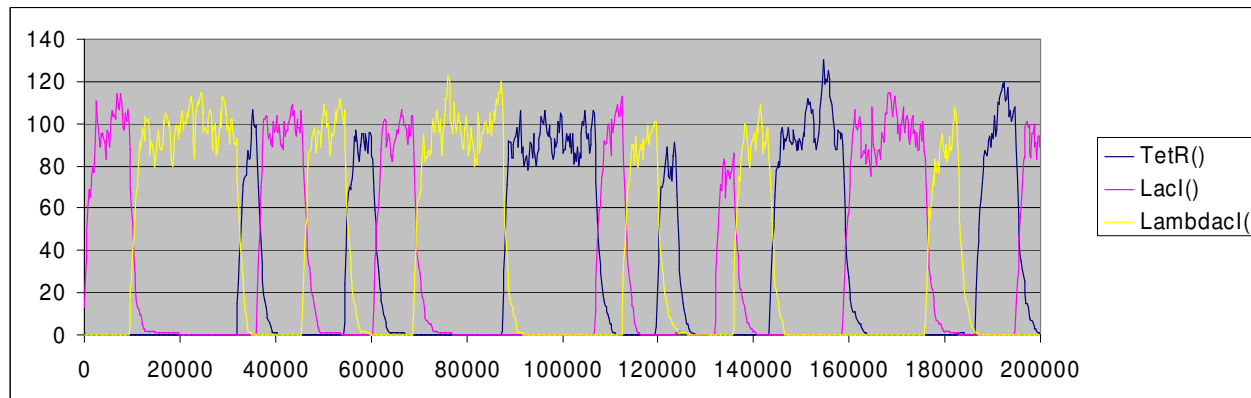
Repressilator: Oscillations



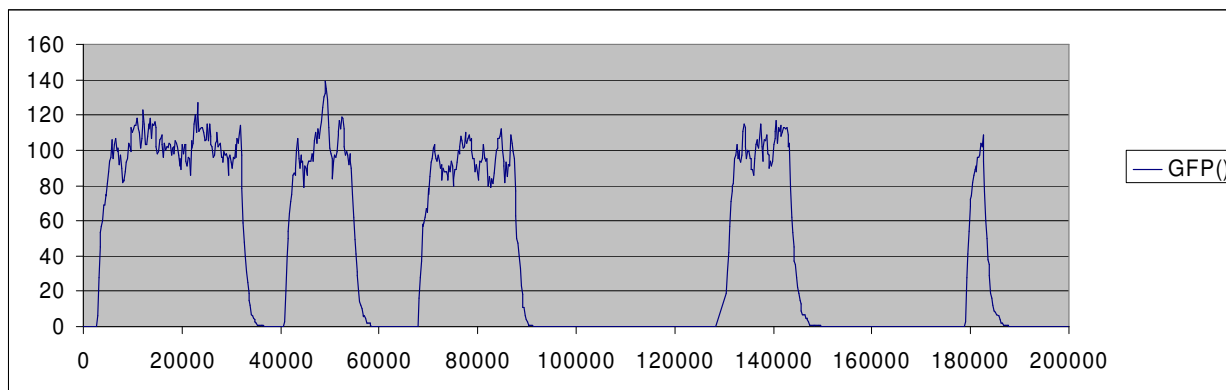
Eventually, the Lambda protein will overtake Lac. Oscillations...

Repressilator: Results

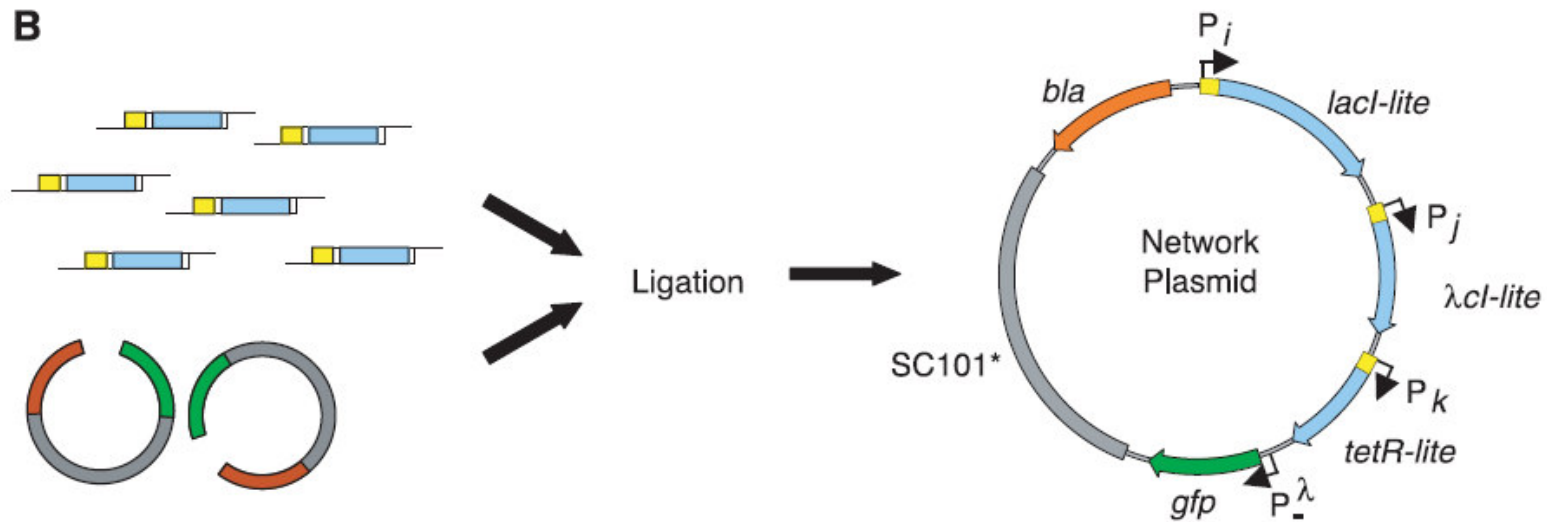
- Alternate Oscillation of Proteins:



- Variable Oscillation of GFP:

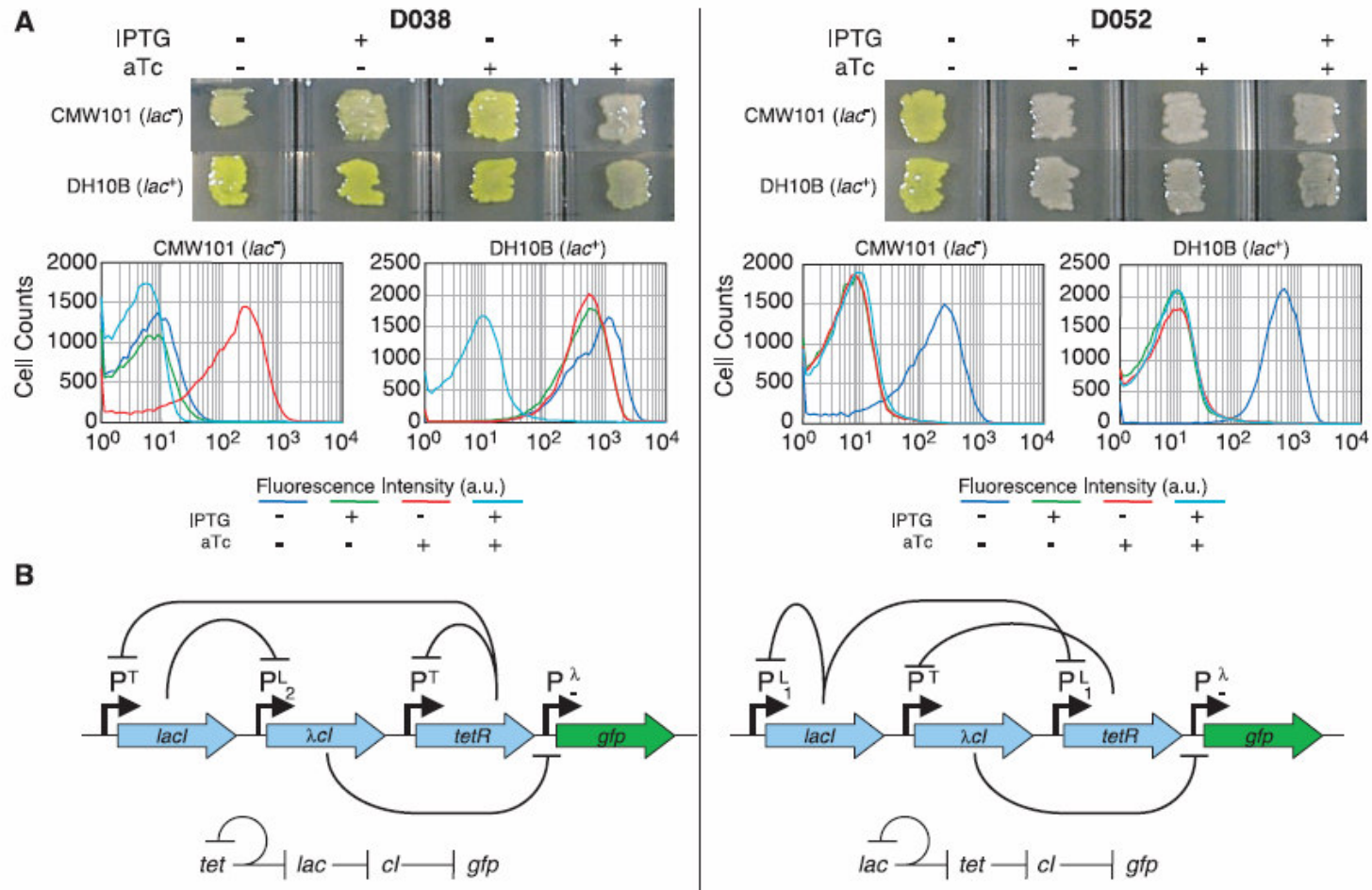


Combinatorial Synthesis [Guet et al., 2002]



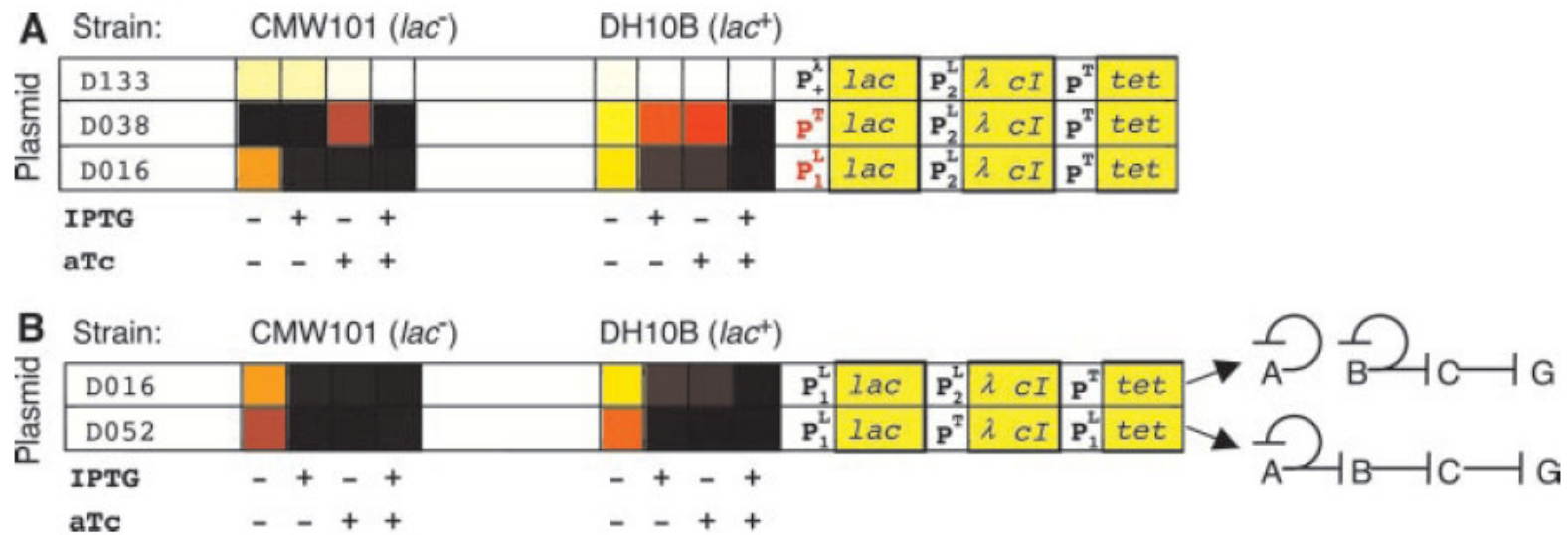
- 3 genes: *tetR*, *lacI*, λcI
- 5 promoters: P_{L1} , P_{L2} , P_T , $P_{\lambda-}$, $P_{\lambda+}$
- 2 inputs: IPTG (represses Tet), aTc (represses Lac)
- 1 output: GFP (linked to $P_{\lambda-}$)
- 125 possible networks consisting of 3 promoter-gene units

In Vivo Logic Gates [Guet et al., 2002]



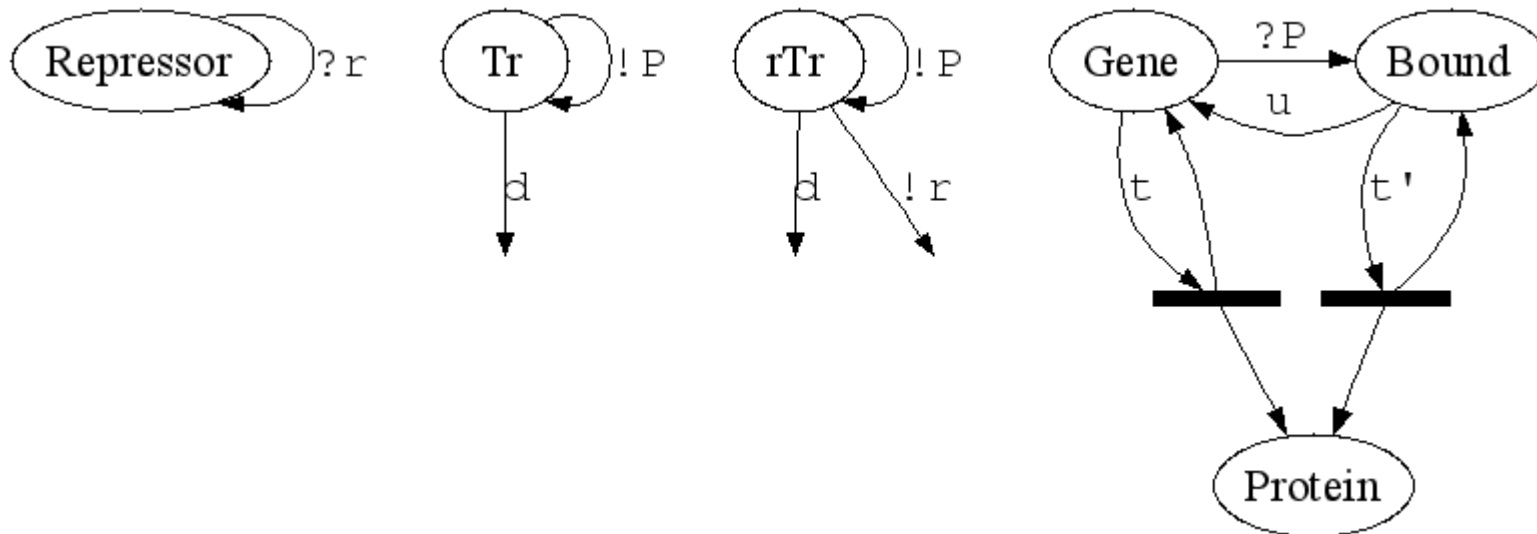
- IPTG and aTc as boolean inputs, GFP as boolean output

Unexpected Results [Guet et al., 2002]



Library of Genes [Blossey et al., 2005]

- Constructed a library of genes:



- The circuits are built by simple combinations of the library e.g:
 - $D038() = (\text{tet}(PT) \mid \text{lac}(PT) \mid \text{cI}(PL2) \mid \text{gfp}(Plm))$
 - $D016() = (\text{tet}(PT) \mid \text{lac}(PL1) \mid \text{cI}(PL2) \mid \text{gfp}(Plm))$
- Stochastic simulation of resulting networks

Library of Genes [Blossey et al., 2005]

```
new PTetR @1.0: chan
new PLacI @1.0: chan
new PLambdacI @1.0: chan
new PGFP @1.0: chan
new RaTc @100.0: chan
new RIPTG @100.0: chan
val PL1 = (PLacI, 0.01, 0.0)
val PL2 = (PLacI, 0.01, 0.0)
val PT = (PTetR, 0.01, 0.0)
val Plm = (PLambdacI, 0.01, 0.0)
val Plp = (PLambdacI, 0.01, 1.0)

let TetR() = rTr(PTetR,RaTc)
let LacI() = rTr(PLacI,RIPTG)
let LambdacI() = Tr(PLambdacI)
let GFP() = Tr(PGFP)
let IPTG() = Repressor(RIPTG)
let aTc() = Repressor(RaTc)

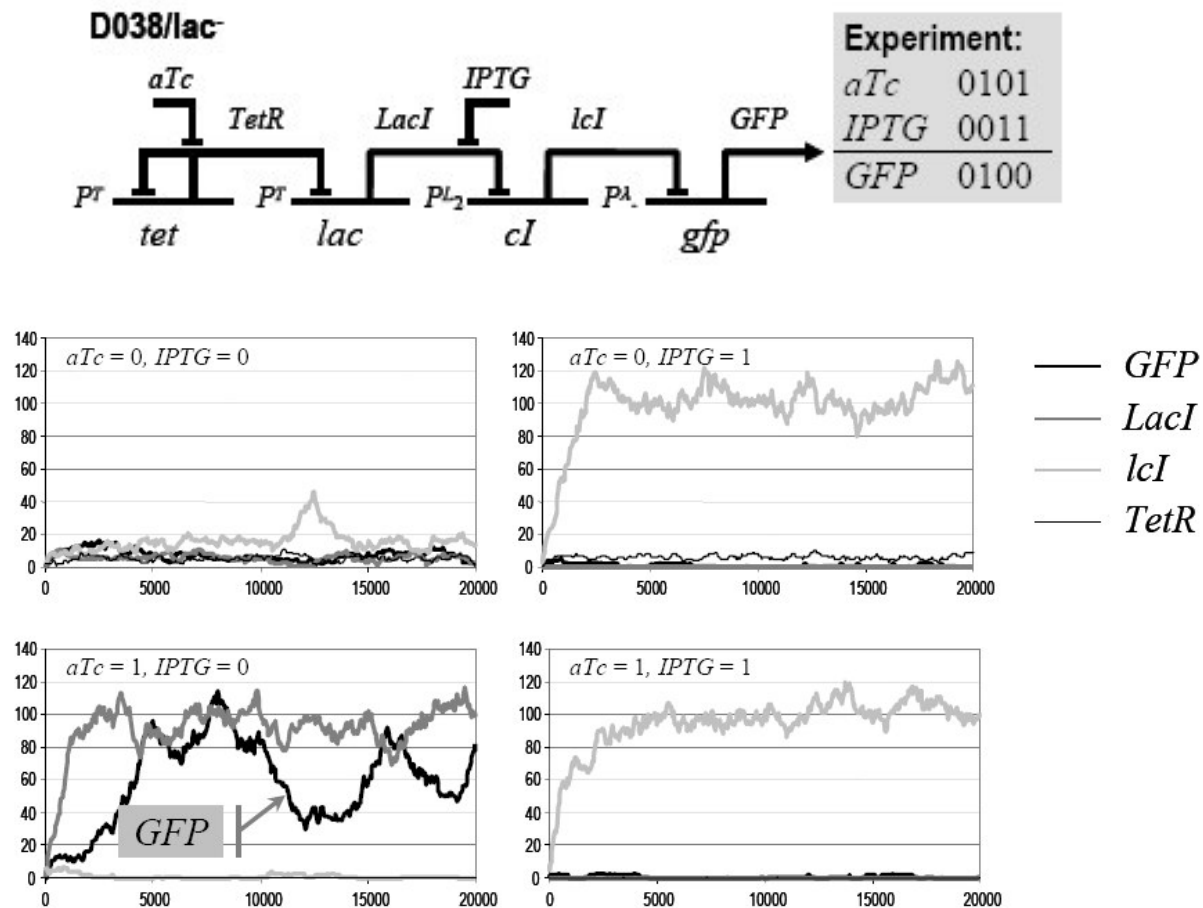
let tet(pr:promoter) = Gene(pr, TetR)
let lac(pr:promoter) = Gene(pr, LacI)
let cI(pr:promoter) = Gene(pr,LambdacI)
let gfp(pr:promoter) = Gene(pr, GFP)
```

D038() = (tet(PT) | lac(PT) | cI(PL2) | gfp(Plm))

D016() = (tet(PT) | lac(PL1) | cI(PL2) | gfp(Plm))

Simulation Results: D038

- $D038() = (\text{tet}(PT) \mid \text{lac}(PT) \mid \text{cI}(PL2) \mid \text{gfp}(Plm))$

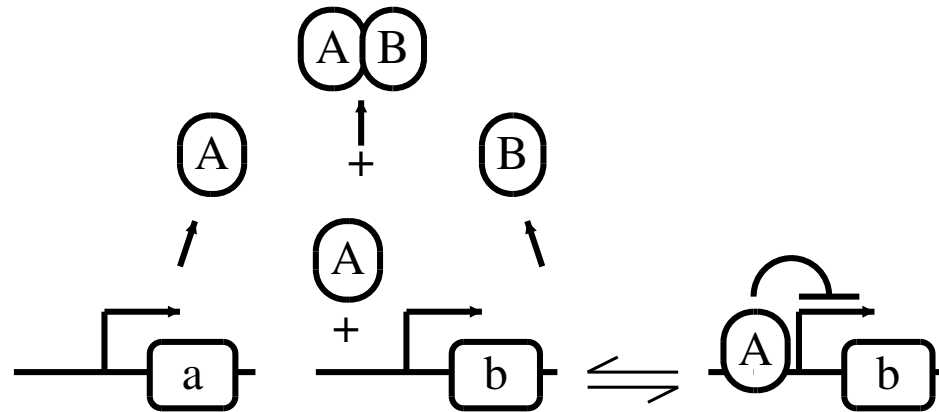


Gene Networks Designed in Silico

with Luca Cardelli (Microsoft Research)

Evolution in Silico [Francois and Hakim, 2004]

- Gene networks can be evolved in silico to perform specific functions, e.g. a bistable switch:



- Genes *a* and *b* can produce proteins *A* and *B* respectively:
 - *A* and *B* can bind irreversibly to produce *AB*, which degrades.
 - *A* can also bind reversibly to gene *b*, to inhibit transcription of *B*

Stochastic π -calculus Model

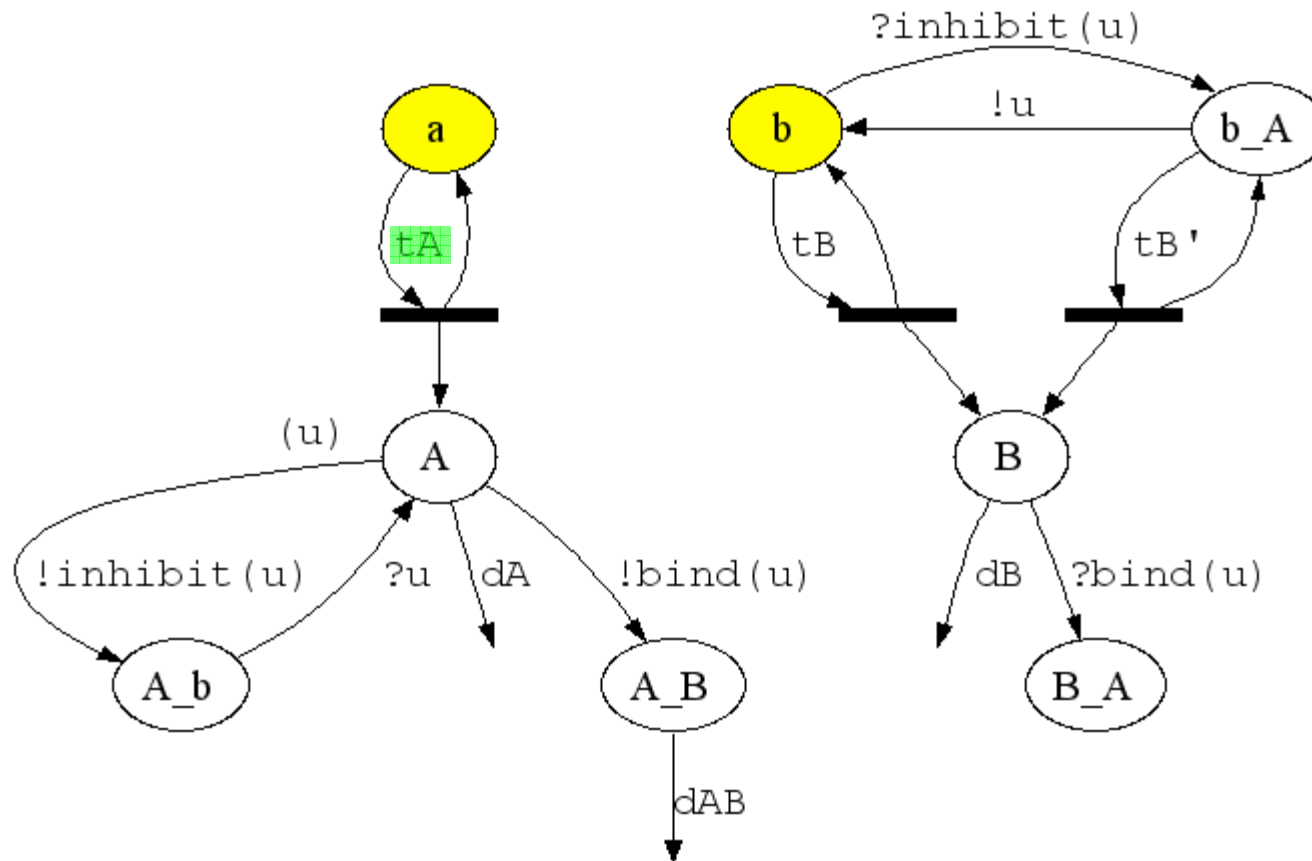
```
val tA = 0.20
val dA = 0.002   val dAB = 0.53
new bind@0.72   new inhibit@0.19
```

```
let a() = delay@tA; ( A() | a() )
and A() = (
  new u@unbind:chan
  do delay@dA
  or !bind(u); A_B(u)
  or !inhibit(u); A_b(u)
)
and A_b(u:chan) = ?u; A()
and A_B(u:chan) = delay@dAB
run (a() | b())
```

```
val tB = 0.37   val tB' = 0.027
val dB = 0.002   val unbind = 0.42
```

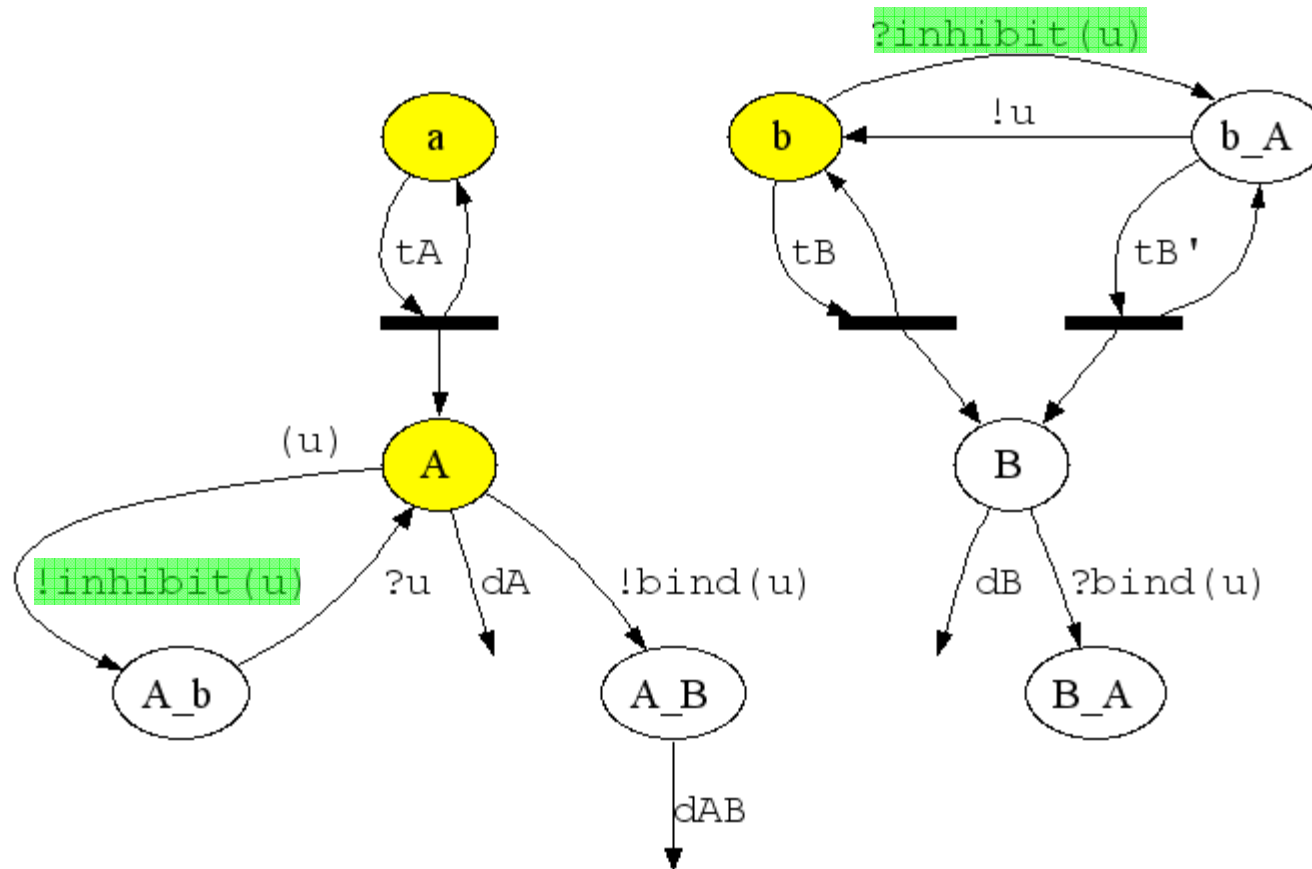
```
let b() =
  do delay@tB; ( B() | b() )
  or ?inhibit(u); b_A(u)
and b_A(u:chan) =
  do !u; b()
  or delay@tB'; ( B() | b_A(u) )
and B() =
  do delay@dB
  or ?bind(u); B_A(u)
and B_A(u:chan) = ()
```

Bistable Network: Protein A



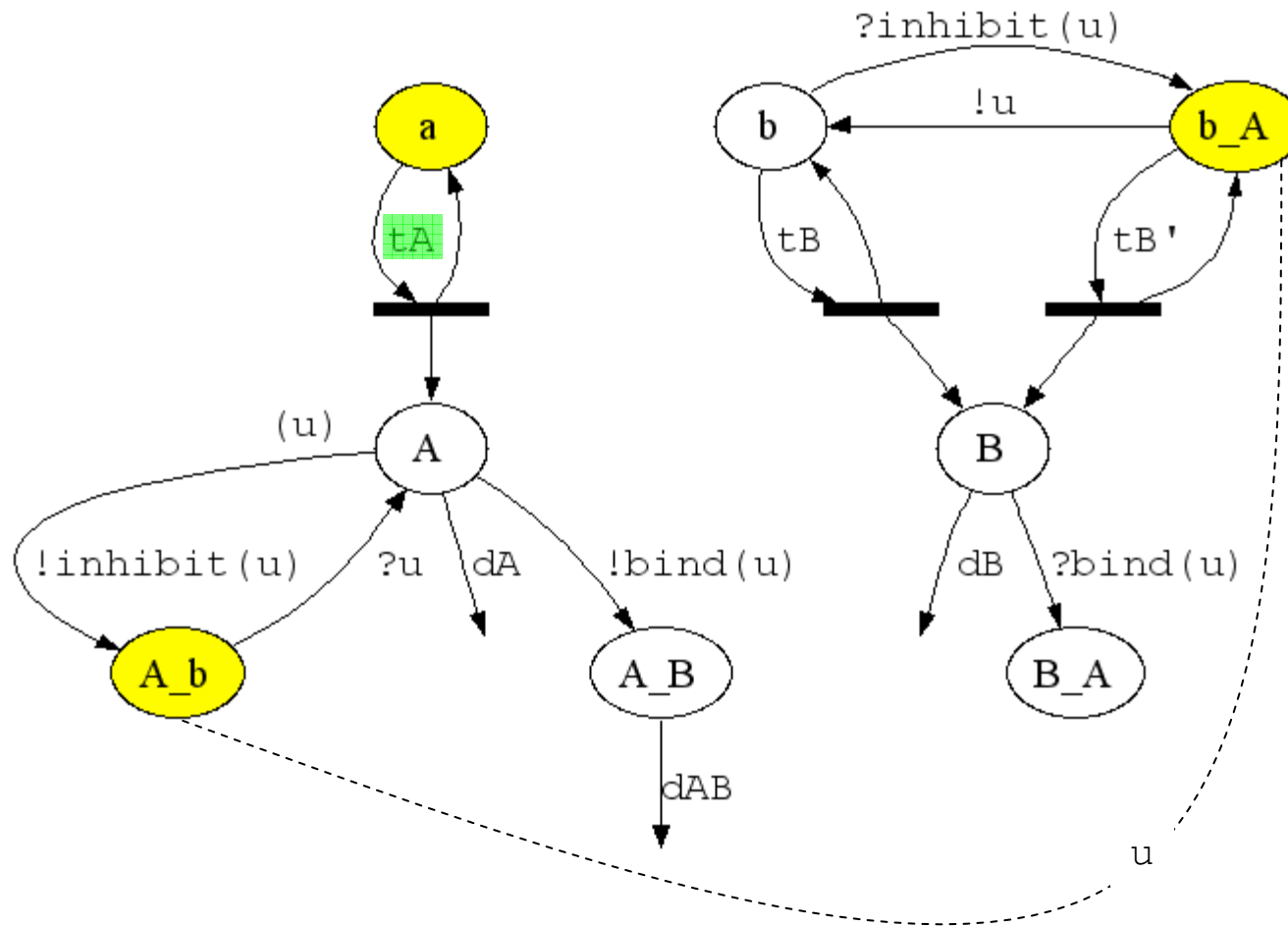
- Gene *a* can transcribe a new protein A at rate tA

Bistable Network: Protein A



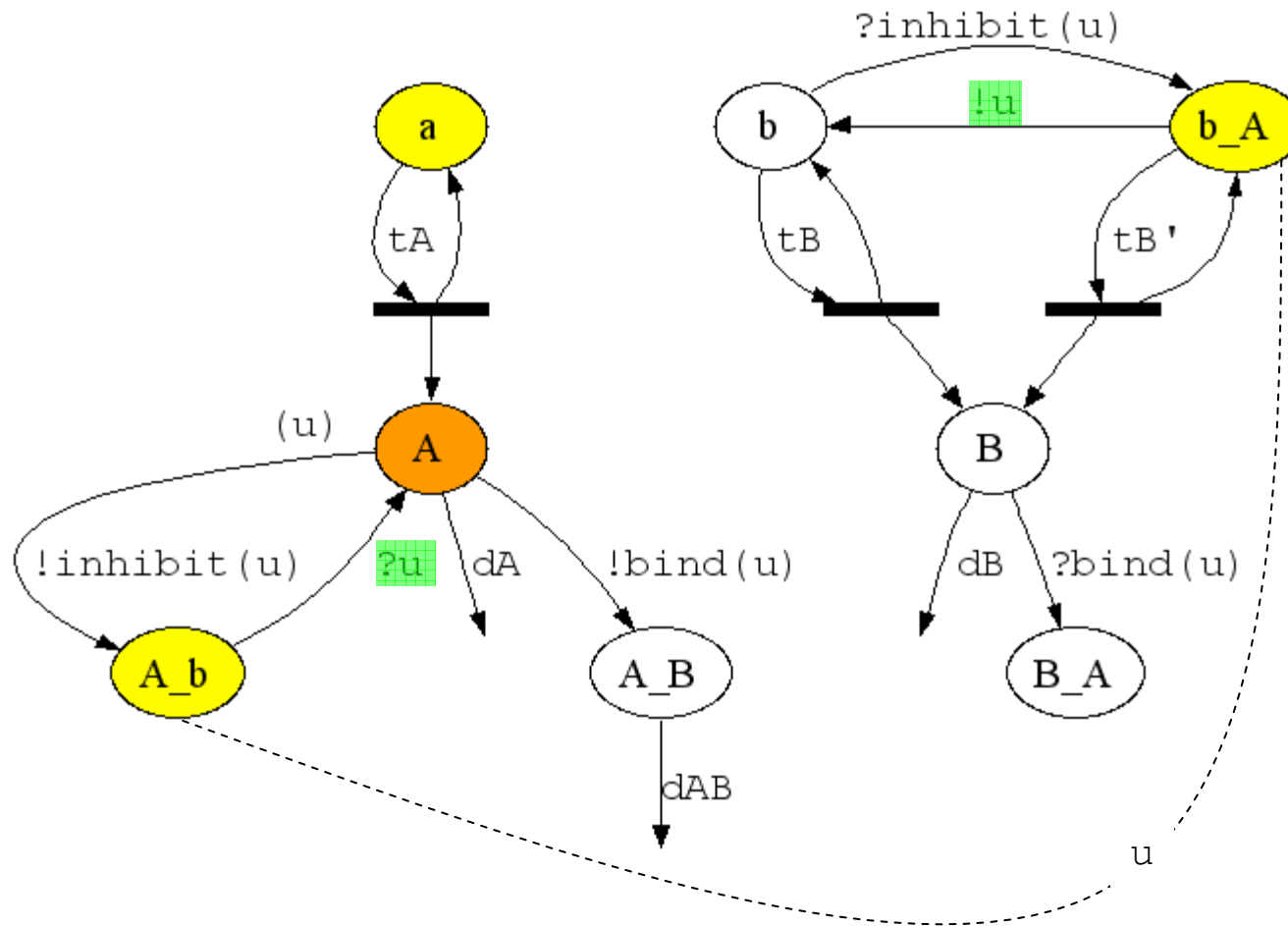
- Protein A can bind to gene *b* to inhibit production of protein *B*

Bistable Network: Protein A



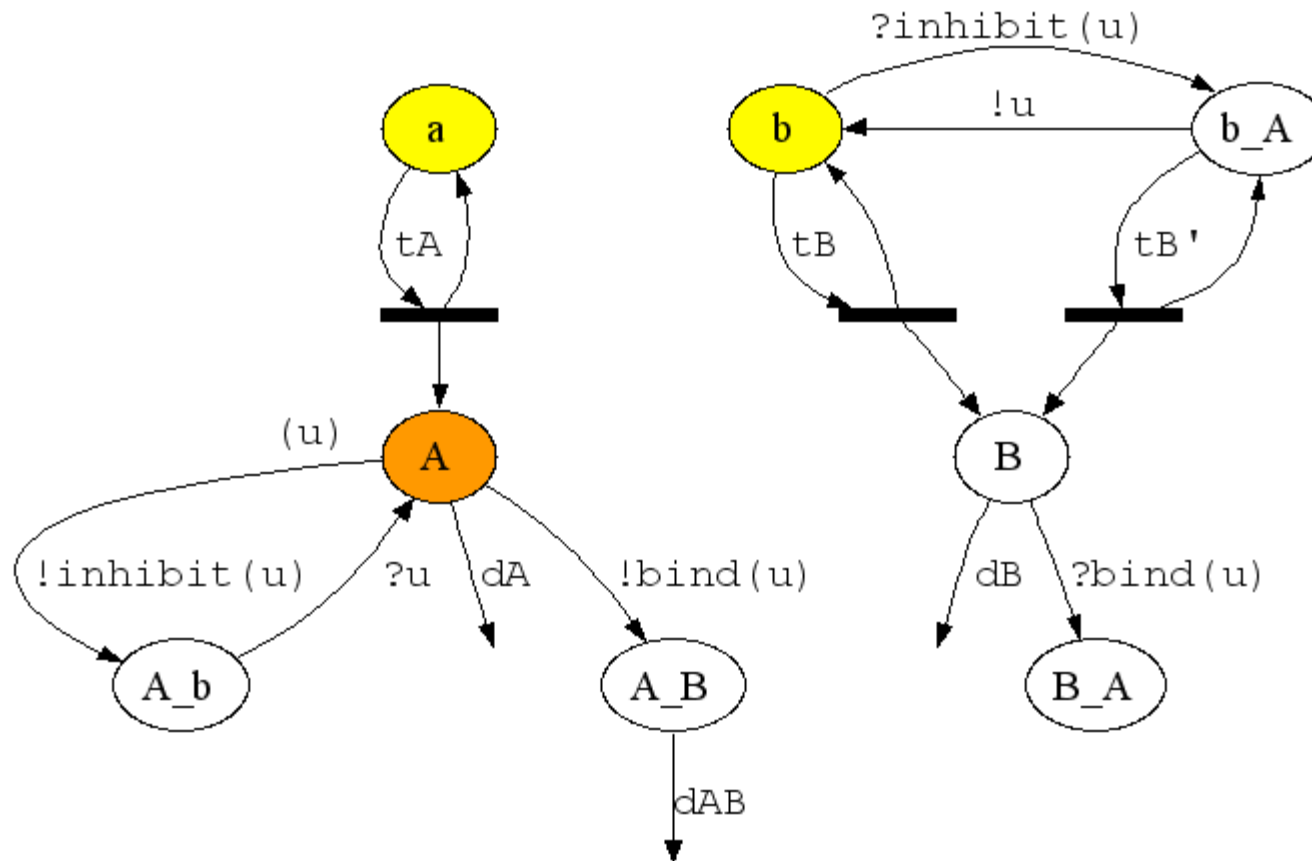
- More protein A is transcribed

Bistable Network: Protein A



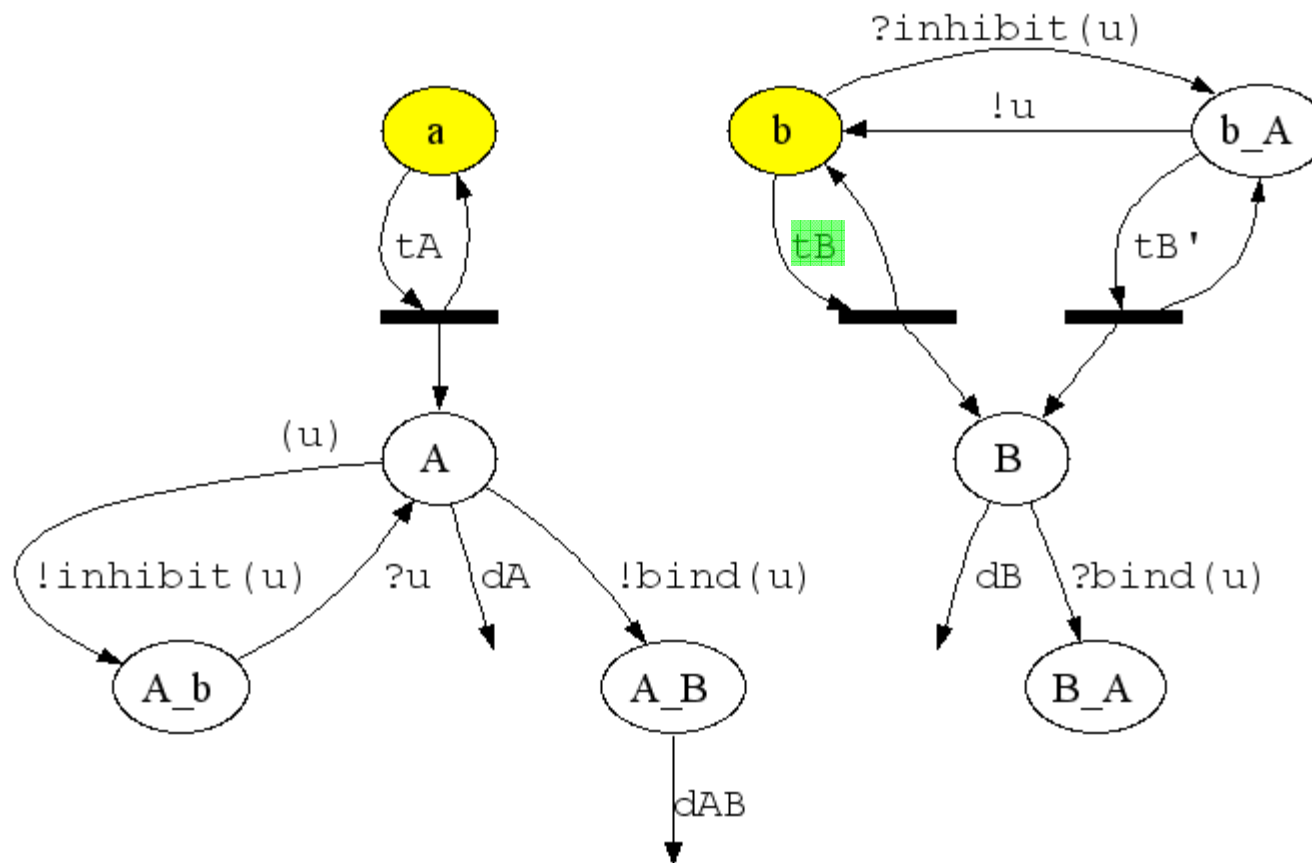
- Protein *A* can unbind from gene *b* using private channel *u*

Bistable Network: Protein A



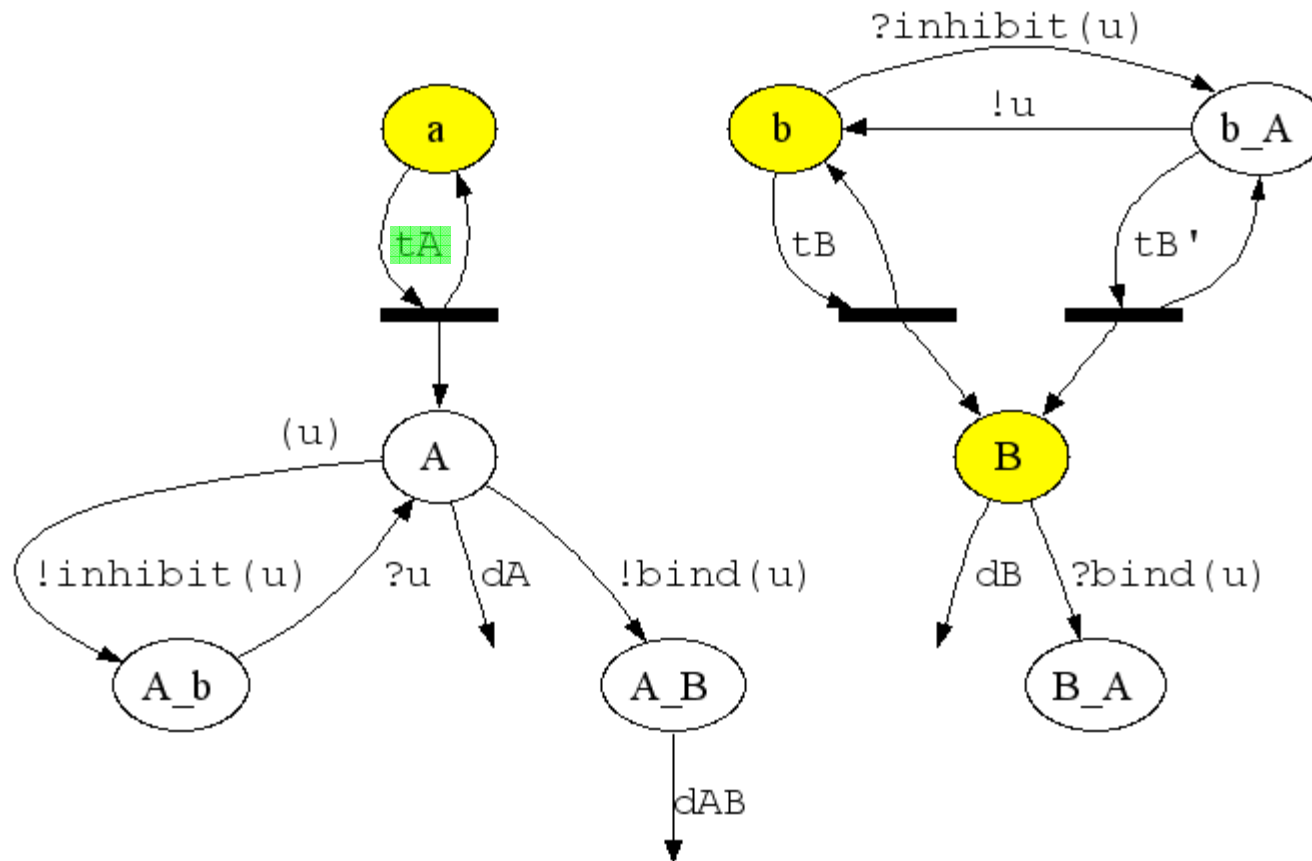
- High proportion of protein A

Bistable Network: Protein B



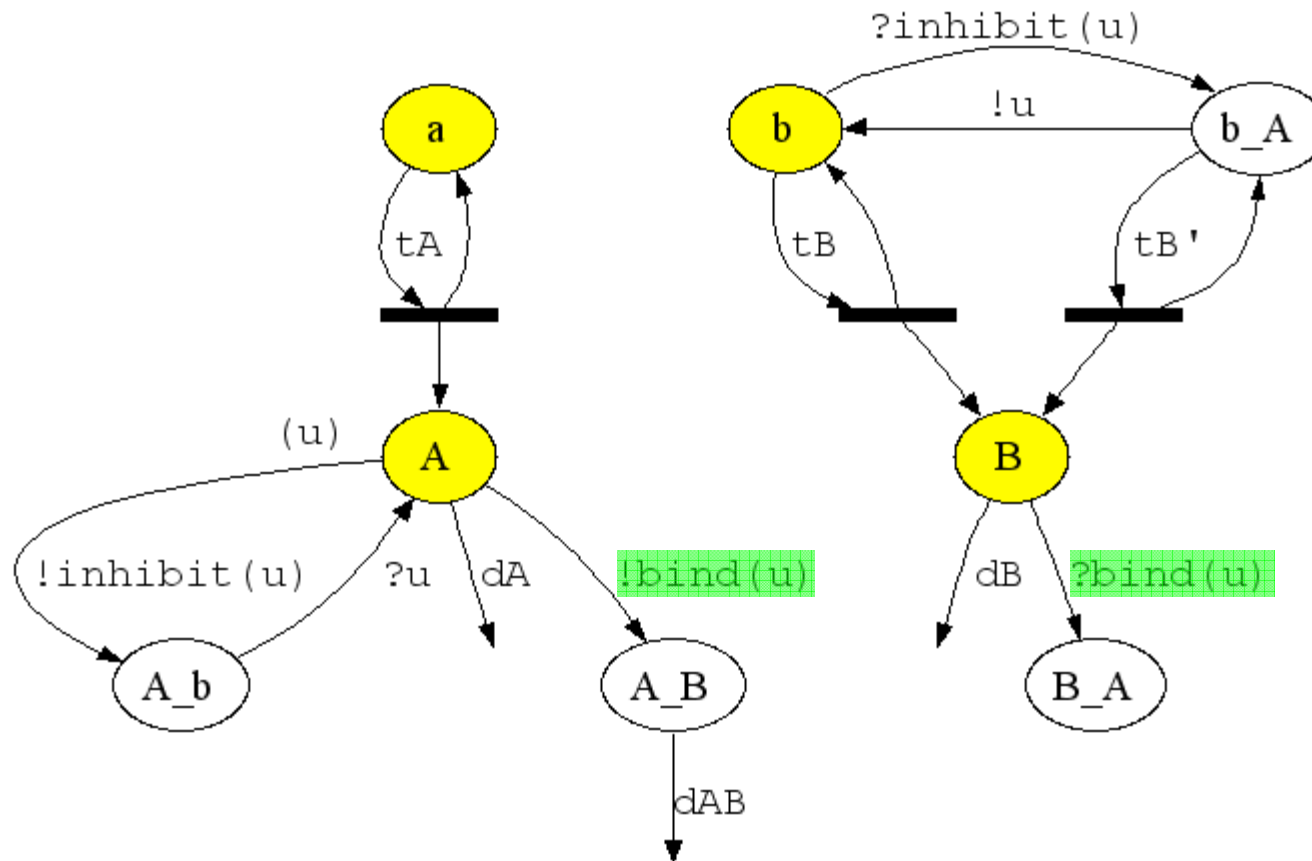
- Gene *b* can transcribe a new protein *B* with rate tB

Bistable Network: Protein B



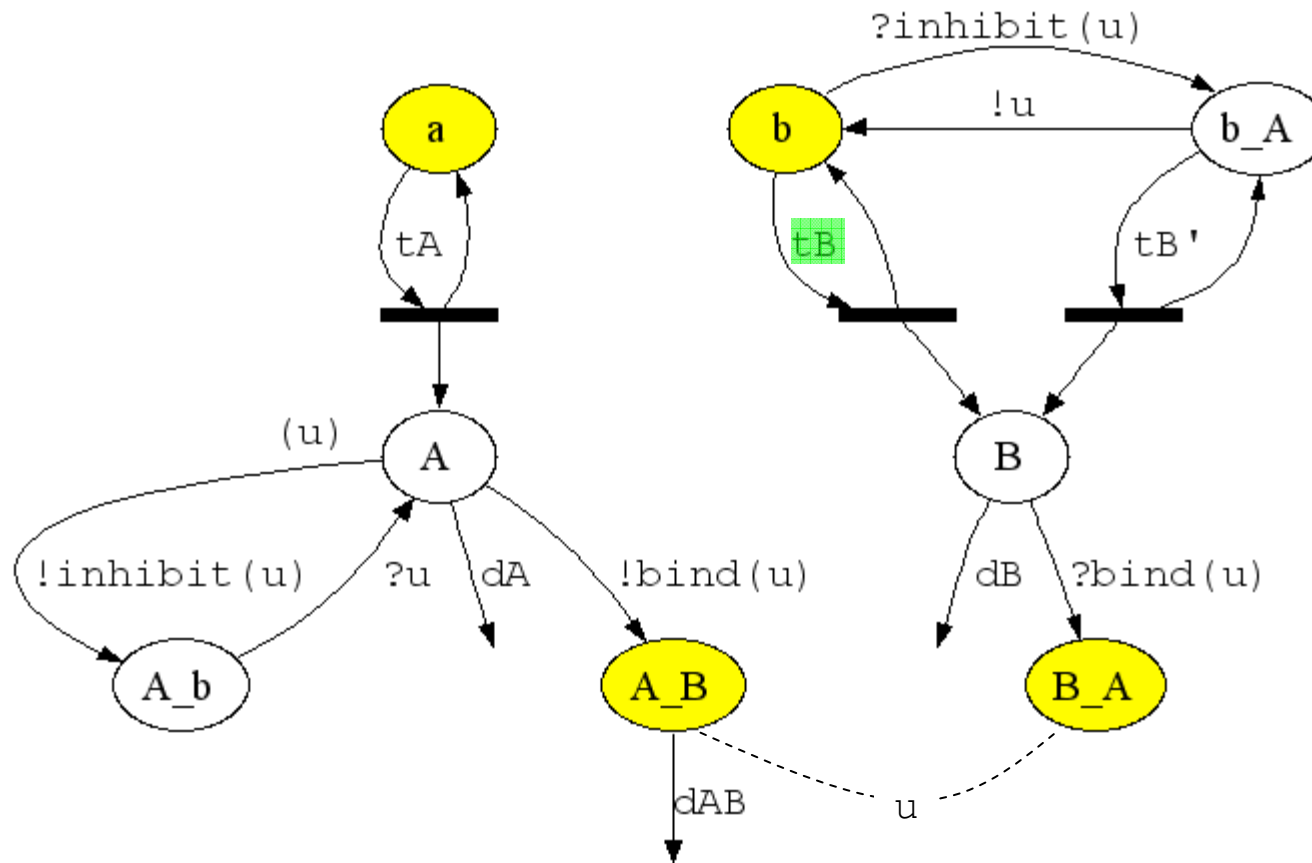
- Gene *a* can transcribe a new protein A at rate tA

Bistable Network: Protein B



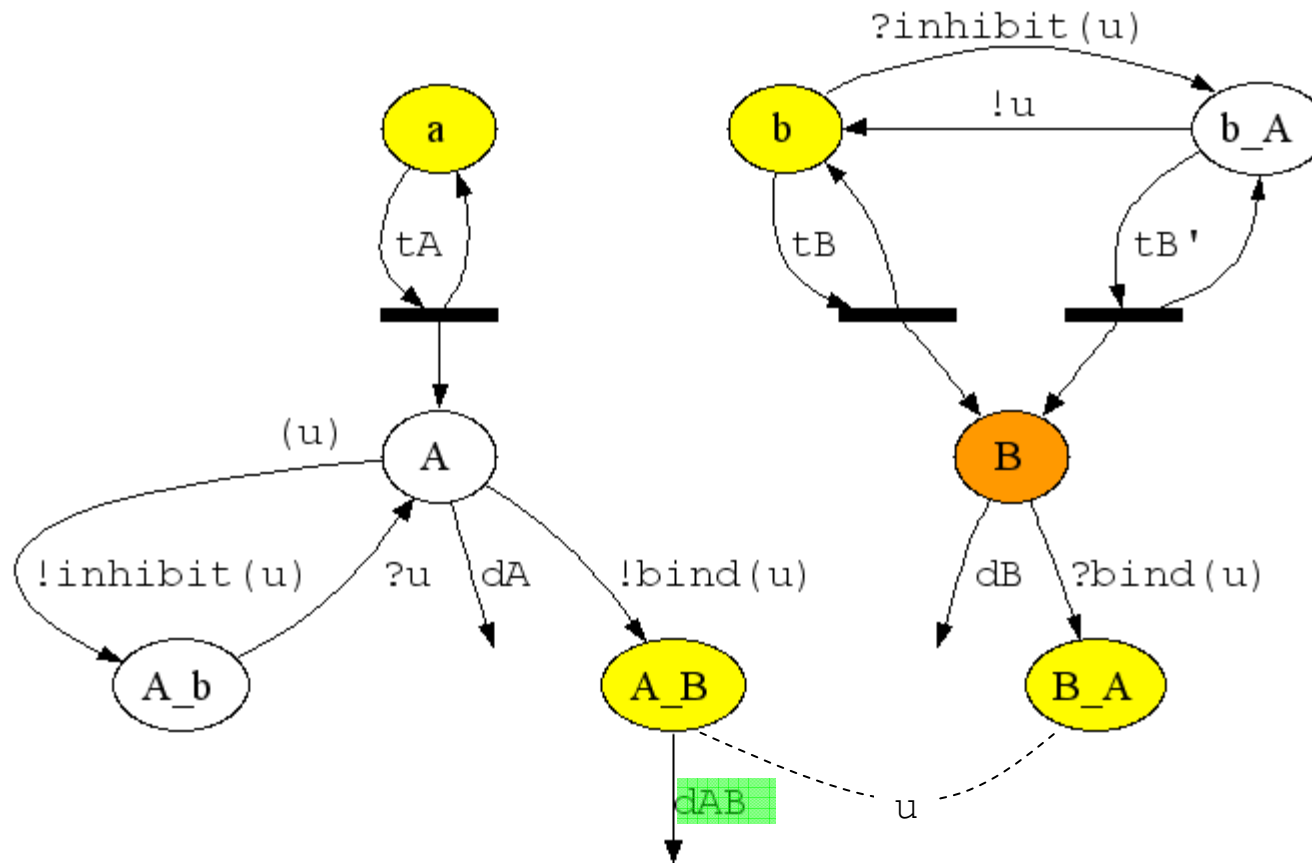
- Protein A can bind with protein B

Bistable Network: Protein B



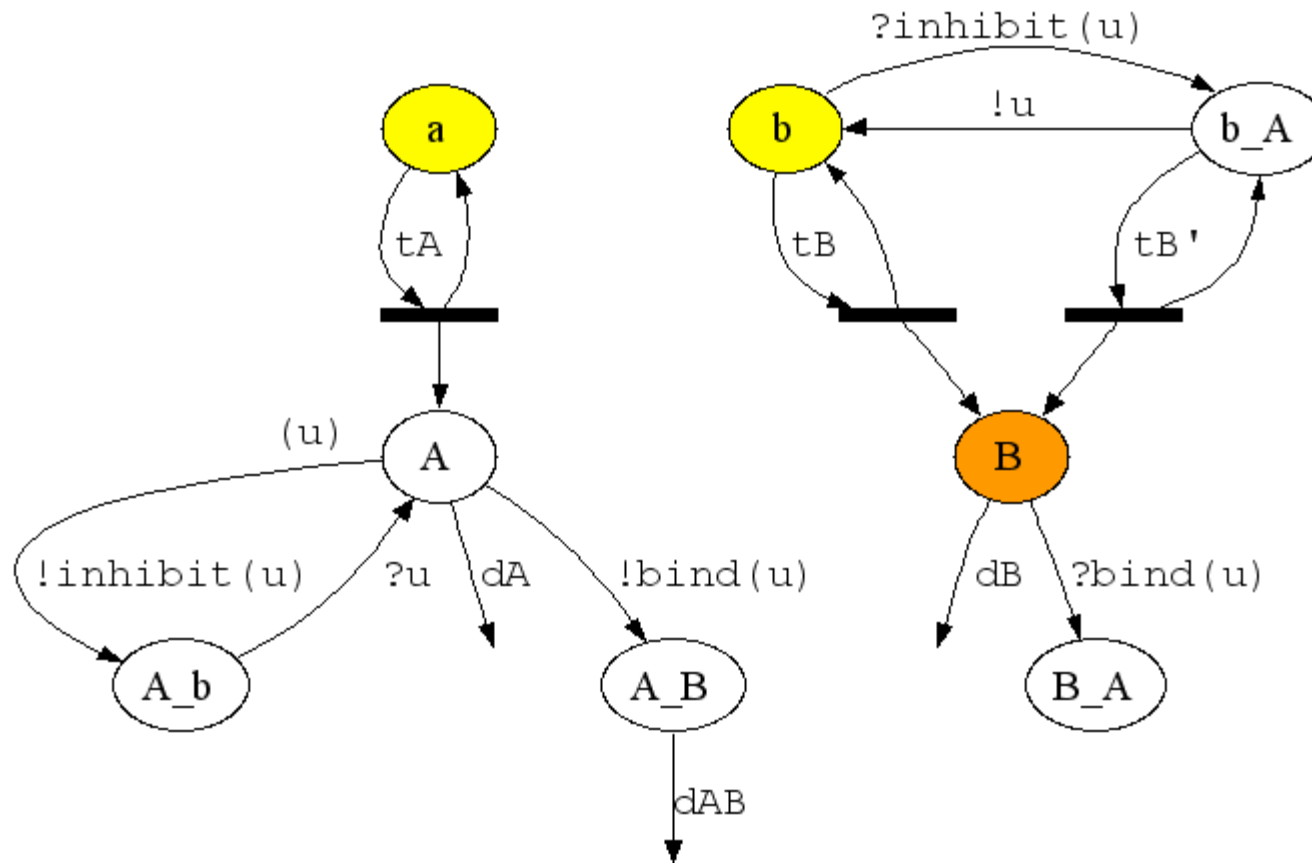
- More protein *B* is transcribed

Bistable Network: Protein B



- Complex AB can degrade with rate d_{AB}

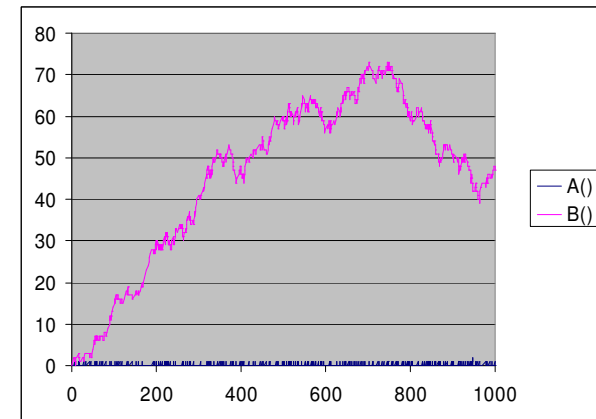
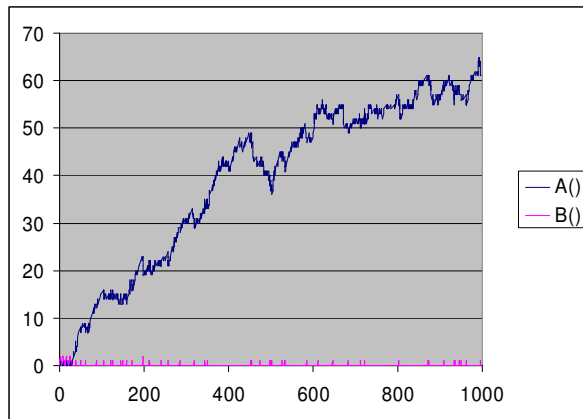
Bistable Network: Protein B



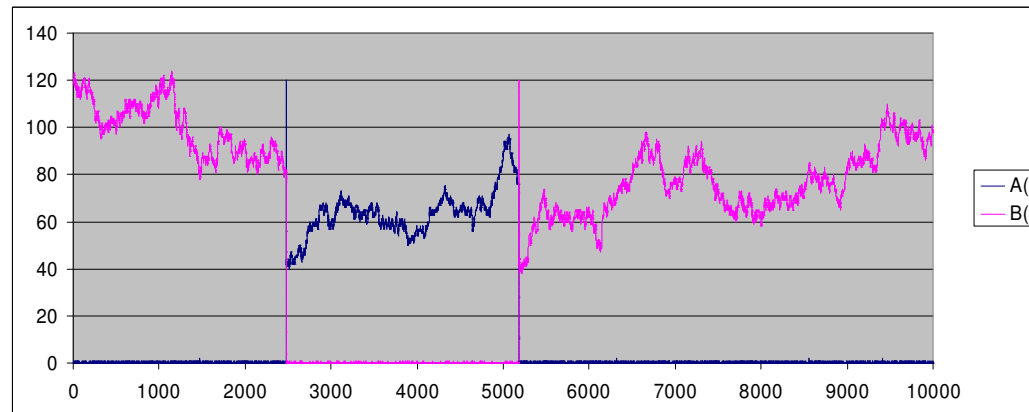
- High proportion of protein *B*

Bistable Network: Results

- Random Initialisation:



- Stable Switching:



Immune System Pathway (in progress)

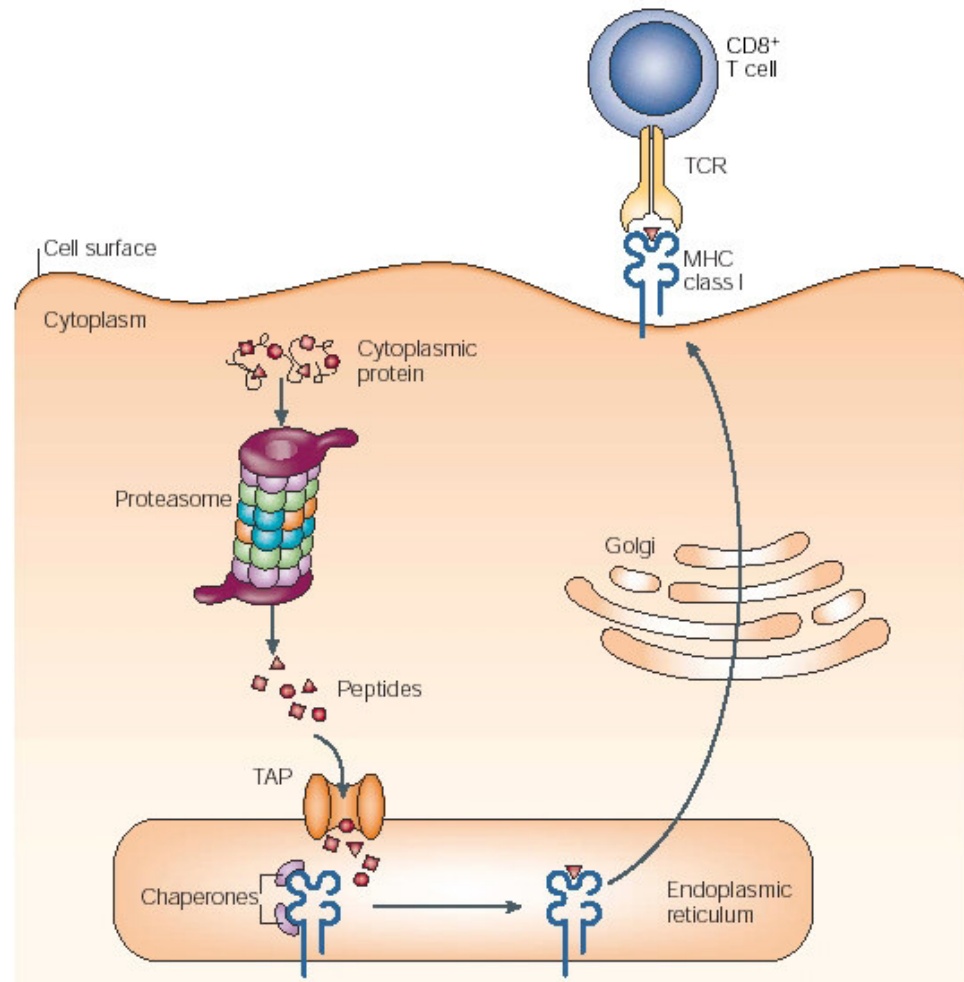
with Luca Cardelli (Microsoft Research)

Leonard Goldstein (Cambridge University)

Tim Elliott and Joern Werner (Southampton University)

MHC 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

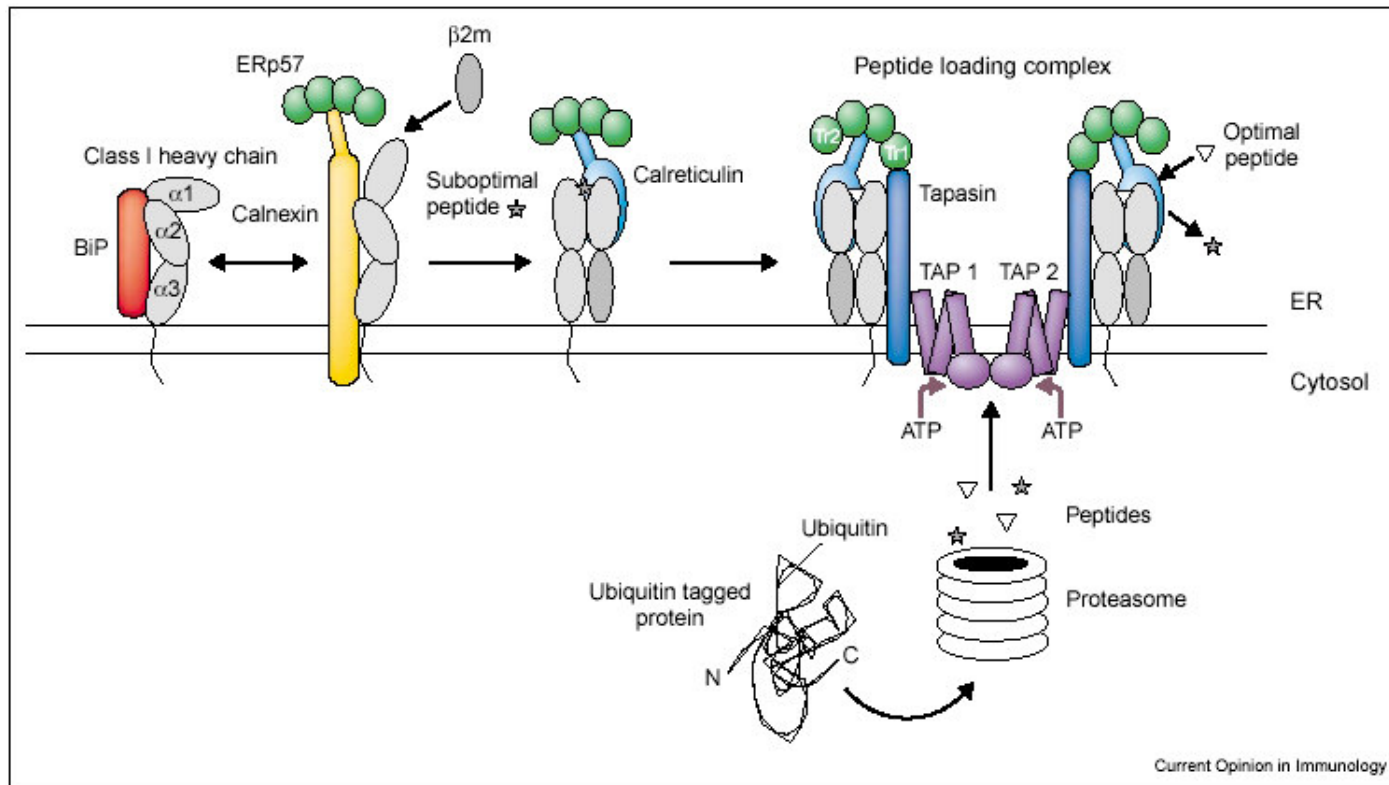


Source: Jonathan W. Yewdell, Eric Reits, and Jacques Neefjes. Making sense of mass destruction: quantitating MHC class I antigen presentation. *Nature Reviews Immunology*, 3(12):952-961, 2003.

MHC I Antigen Presentation

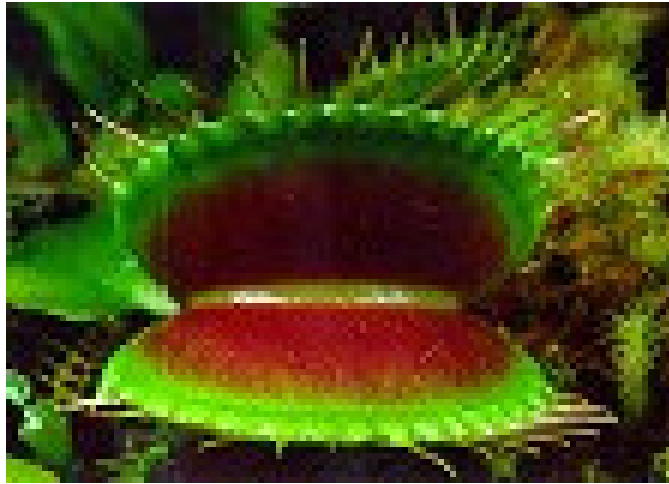
- MHC Class I Antigen Presentation [Immunobiology 6th Ed.]

MHC I Assembly

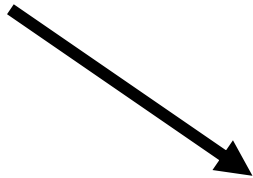
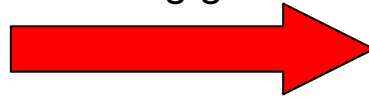


Source: Antony N Antoniou, Simon J Powis, and Tim Elliott. Assembly and export of MHC class I peptide ligands. *Current Opinion in Immunology*, 15:75-81, 2003.

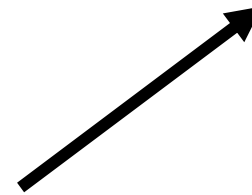
Peptide Capture: Flytrap Model



Peptide contacts
Binding groove

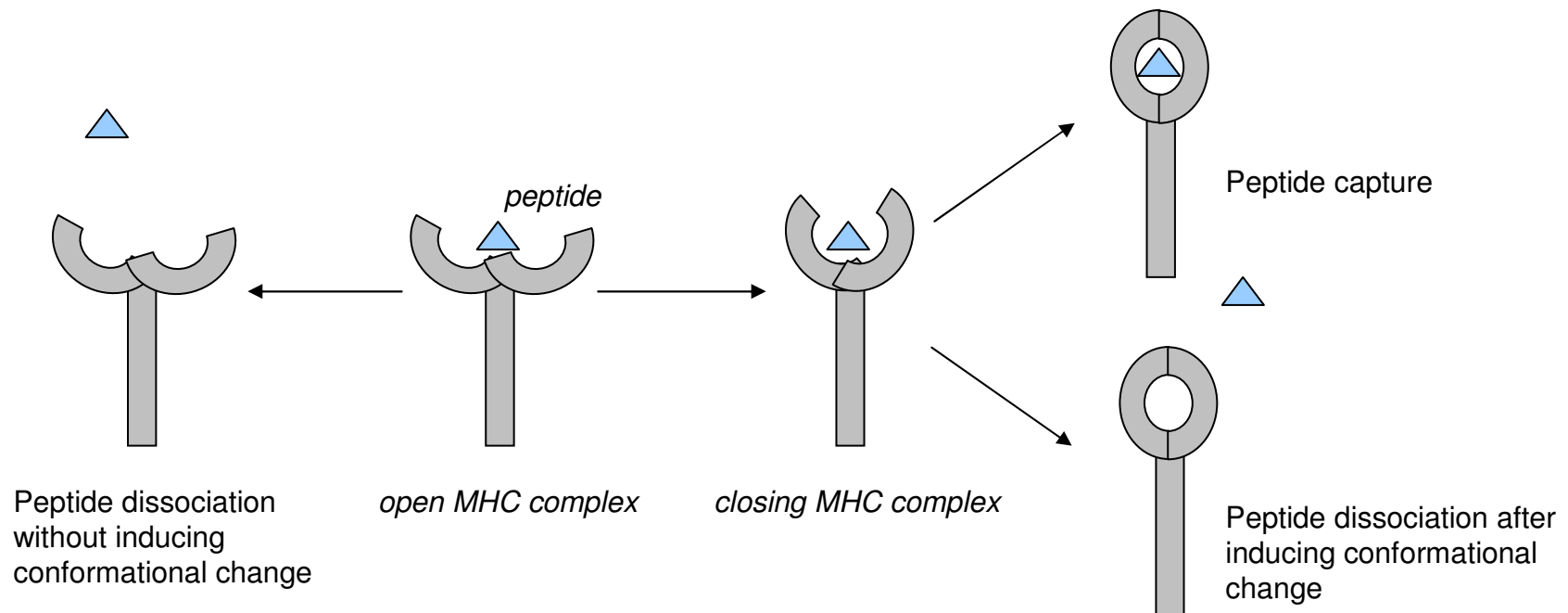


Tapasin stabilises open
Conformation and gives
Weak binders a chance to
escape

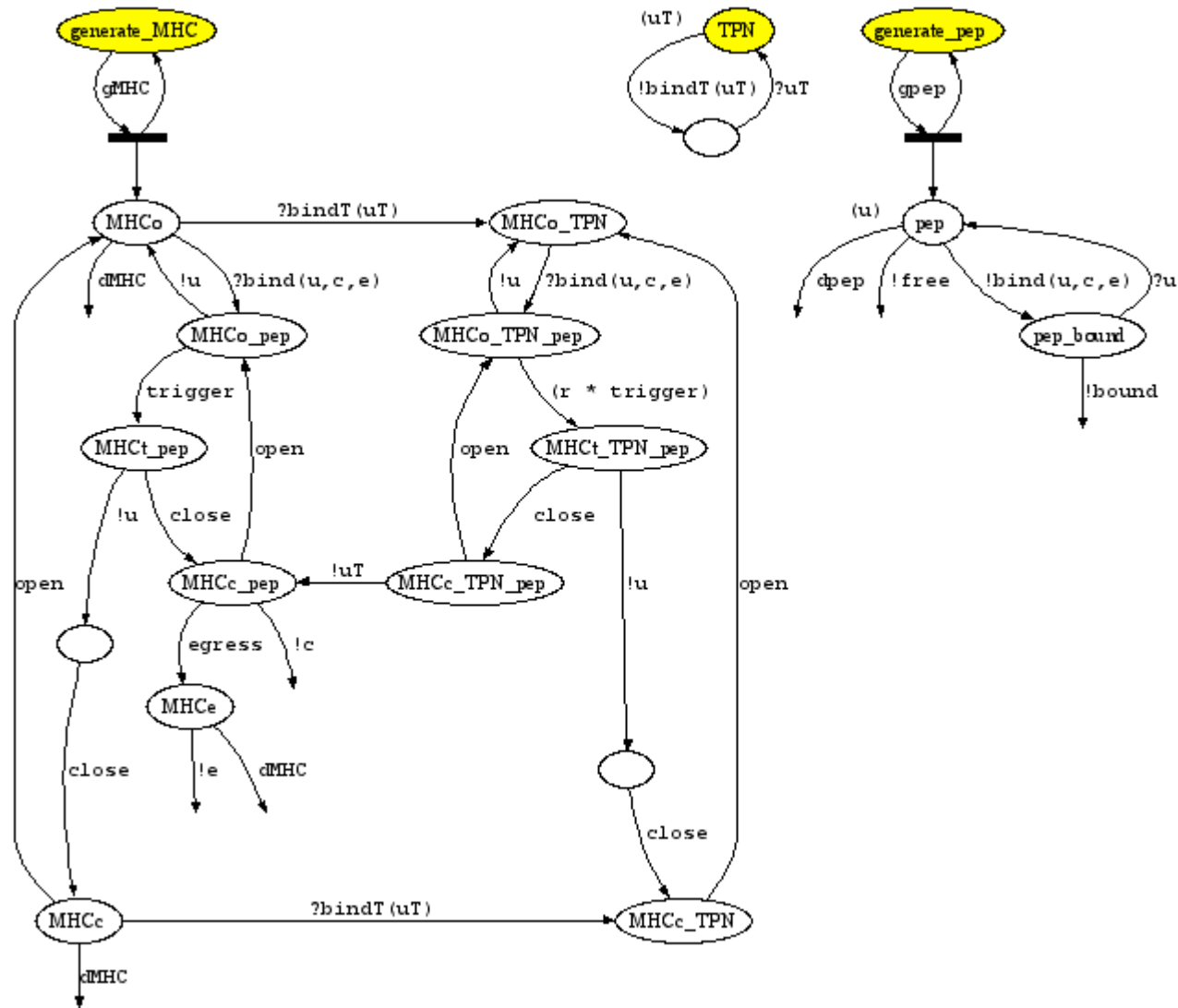


Flytrap Model: Overview

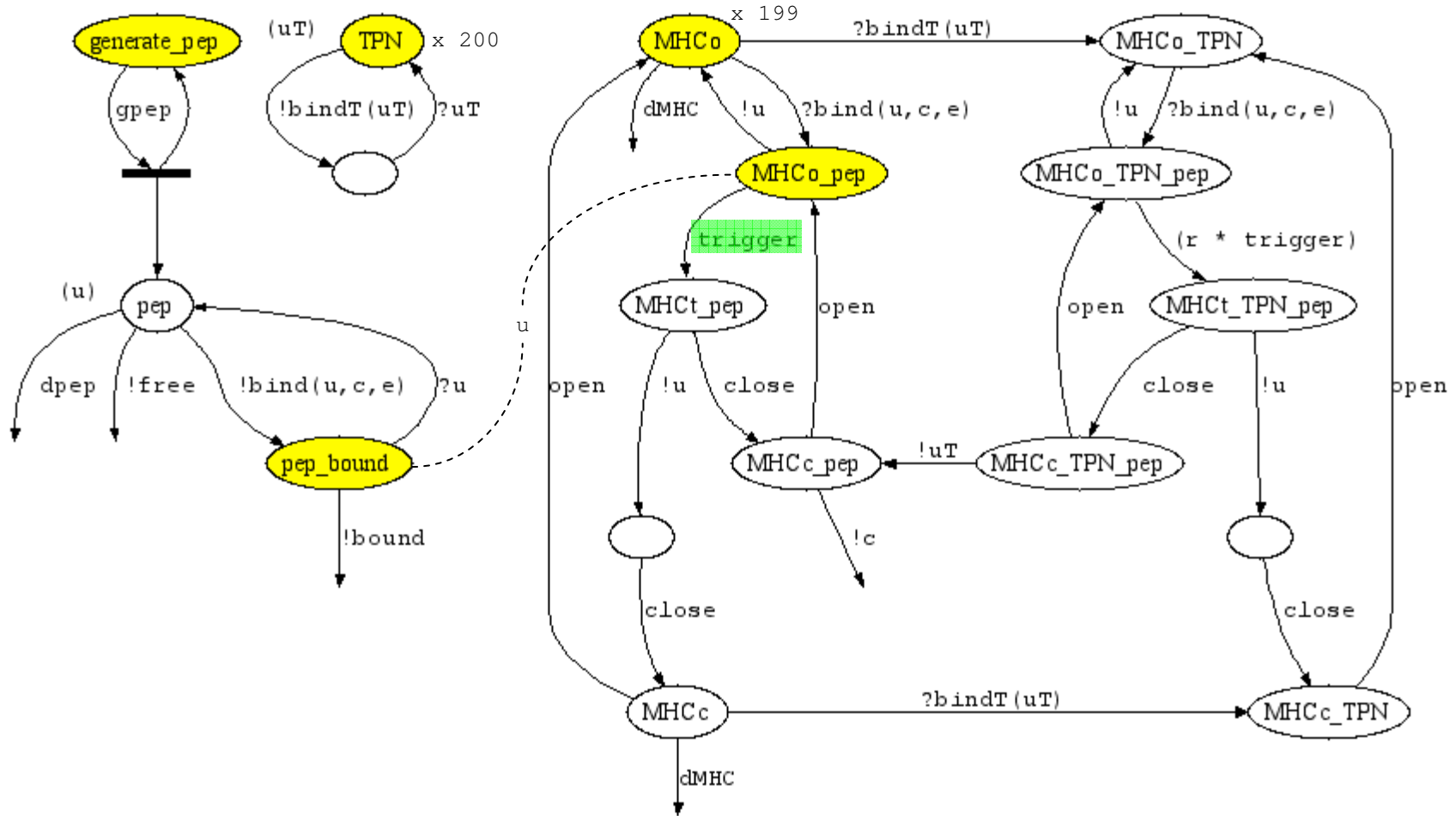
- Experimental results lead to the proposal of a “venus fly trap” model [A. Williams, C. Au Peh, and T. Elliott. *Tissue Antigens*, 59:3–17, 2002.]
- Initial encounter between peptide and MHC can induce conformational change



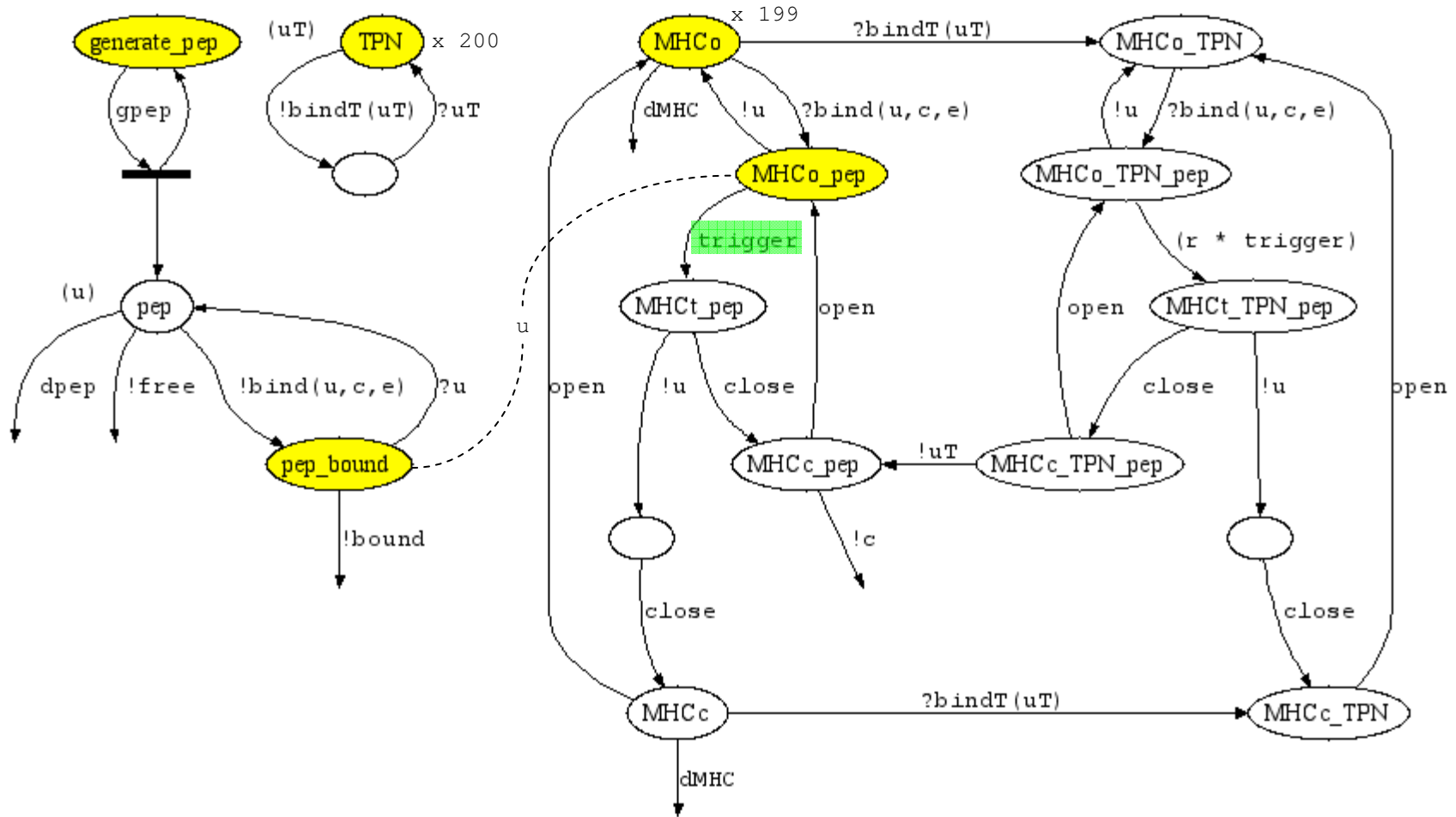
Flytrap Model: π -calculus



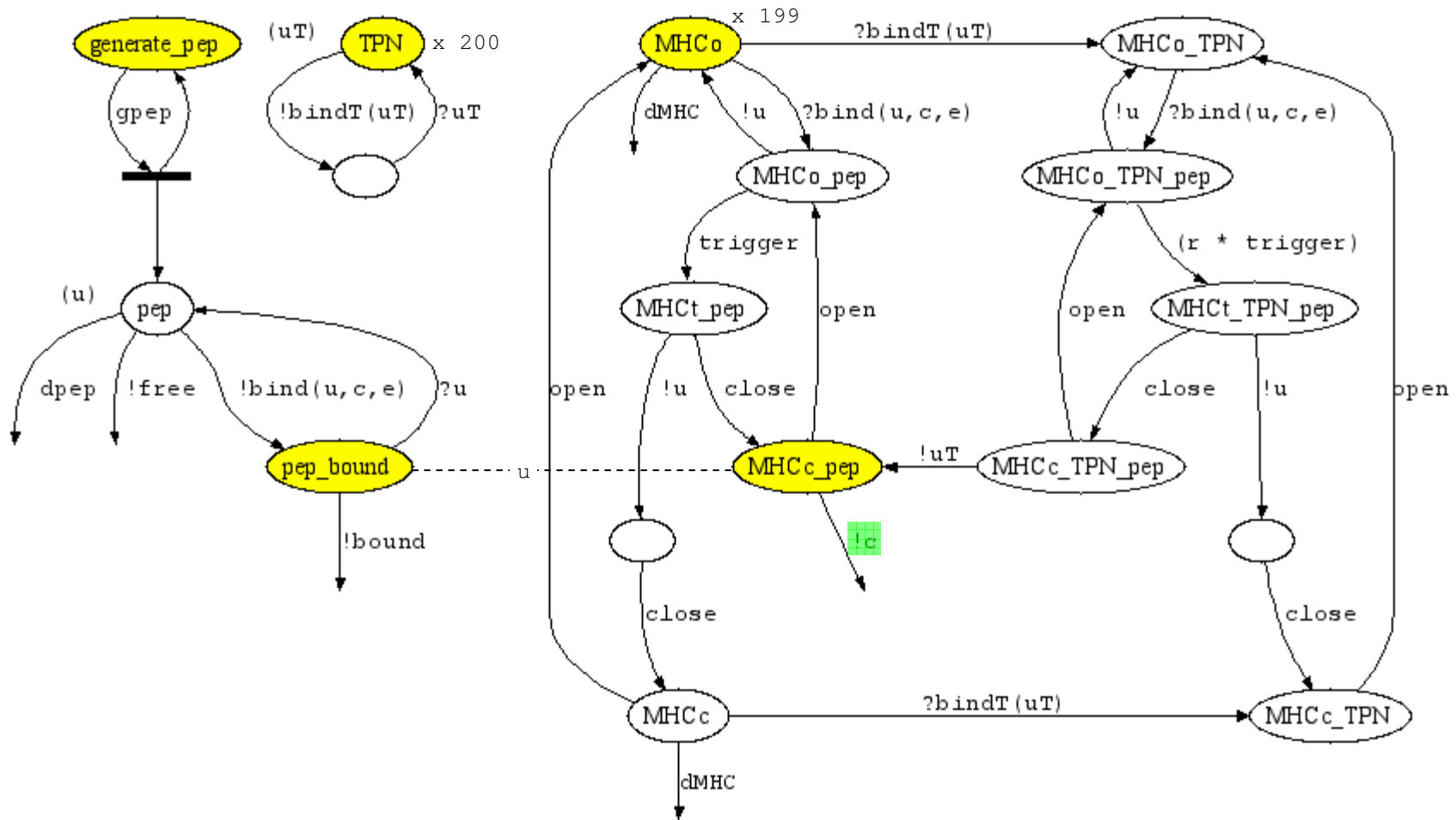
Flytrap Model: π -calculus



Flytrap Model: π -calculus



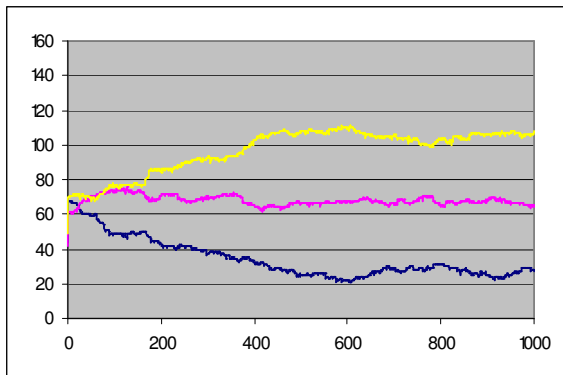
Flytrap Model: π -calculus



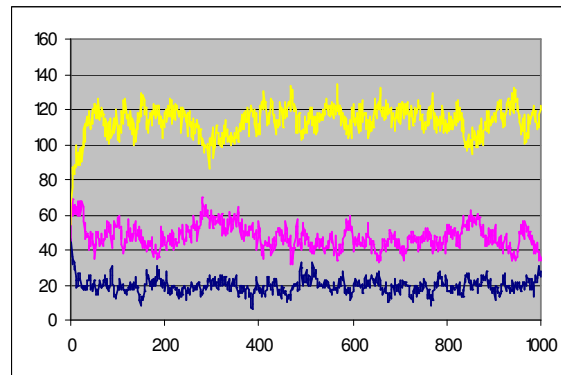
Simulation Results

- Partitioning of loading through both pathways allows:
 - Enhanced optimisation rate
 - Enhanced quantities of loaded peptides

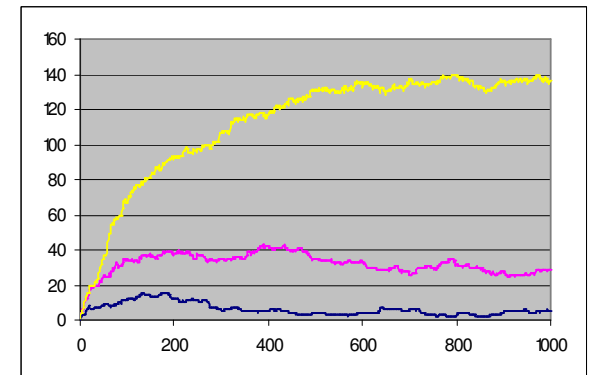
No Tapasin



Always with Tapasin



Transient binding to Tapasin



- Next:
 - Investigate flytrap hypothesis in the lab.
 - Incrementally build a more complete model (compositionality).

Acknowledgements

- Luca Cardelli (Microsoft Research)
- Ralf Blossey (IRI Lille)
- Leonard Goldstein (University of Cambridge)
- Tim Elliott and Joern Werner (University of Southampton)

Thanks.

Questions?

Flytrap Model: Equivalence B

