Ctrl & Comp in Bio Sys - Westlake Univ., Fall 2025

Lecture 10: Computation Biomachine

2025.1113

Lecturer: Fangzhou Xiao *Scribe: Kaijun WANG, Xinyu WANG, Xiaowen ZHANG*

Contents

Again, like what we discussed in adaptation biomachine, when we consider biological systems, we often consider they have distinct functions. e.g. computation. To achieve such functions, we have built engineered machines. e.g. computers. Since biological systems also need to achieve the same tasks, the structures and design principles of engineered machines could be used to better understand why biological systems are built in this way. Although the functions of engineered machines and biomachines are the same (hopefully), their implementations can be different.

Biological systems need to process information to respond to environments. e.g. activate a gene expression to take up new nutrient, start a stringent response when harsh conditions hit because a new cell type when growth factors say "differentials"...

Figure 1

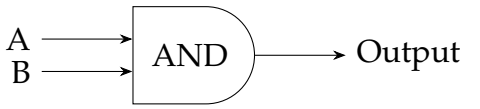
How are the computations done? How complex a computation the cell achieve in this way?

In engineered machines, we build computational units, then link them up for larger and larger computations. In the following sections, we will search for biological computational units.

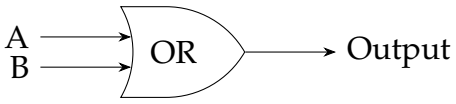
1 Logic gate computation in cells

1.1 Basic logic gate

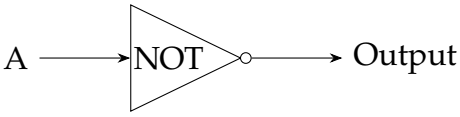
We built computers consisting of binary logic gates to perform computation. Both the inputs and the outputs of logic gates are binary, either true(1) or false(0). There are three basic logic gates: AND, OR, NOT.



A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1



A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



A	−A
0	1
1	0

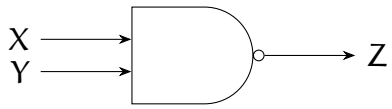
Figure 2 Diagrams of AND(top), OR(middle and NOT(down) gates and their corresponding truth tables

1.2 Logic gate implementation with catalysis reactions

Maybe cells can do the same thing. Let's assume that cells encode $0/1$ as low/high concentration, X_0, X_1. And cells may implement logic gates via catalysis reactions. The following equations show a series of catalysis reactions that achieves a NAND gate. We should note that NAND gate itself is enough to build universal Turing machines.

1.3 Comparison of complexity with engineered machines

But let us slow down here. If we assume that three substrates and four reactions are needed to implement a single NAND gate, then a cell—with roughly 10^3 types of enzymes—could



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

$$X_0 + Y_0 + Z_0 \rightarrow X_0 + Y_0 + Z_1$$
$$X_0 + Y_1 + Z_0 \rightarrow X_0 + Y_1 + Z_1$$
$$X_1 + Y_0 + Z_0 \rightarrow X_1 + Y_0 + Z_1$$
$$X_1 + Y_1 + Z_1 \rightarrow X_1 + Y_1 + Z_0$$

Figure 3 Diagrams of NAND gate, truth table and catalysis reactions implementation of NAND

effectively generate only about (3×10^2) logic gates, even if all enzymes were devoted to computation. When we compare this level of complexity with engineered machines, early electronics such as a 1972 pocket calculator (about 3×10^3 transistors) and the Apollo Guidance Computer used for the moon landing (around 5×10^3 logic gates) exceeded cellular computational complexity only modestly. As technology progressed, microcontrollers with limited functionality reached about 10^6 gates, and modern laptop processors now contain on the order of 10^9 gates. At the very least, we should recognize that cells operate in a far more complex and dynamic environment than a microcontroller does—yet there remains a vast gap in raw computational complexity between a cell and a modern microcontroller. Thus, we conclude that if cells only use logic gates for computation, the complexity of logic gates is not enough. Maybe cells do computations in smarter ways to achieve more efficient computing than logic gates.

Machine	**Complexity**
cell (all the enzymes process information with gates)	3×10^2 gates
pocket calculator in 1972	3×10^3 transistors
moonlanding in 1970s	5×10^3 gates
microcontroller	10^6 gates
intel laptop chip	10^9 gates

Table 1 Complexity gap between cell and real engineered machines when assuming cells process information with logic gates only.

2 Neural networks

Recent years (2010 onwards). Artificial neural networks (ANN) have taken over in complexity of certain computational tasks, e.g. computer vision, language models/translation, image recognition, robotics... ANNs are more powerful by "depth". Instead of just linking computational units as cascades,

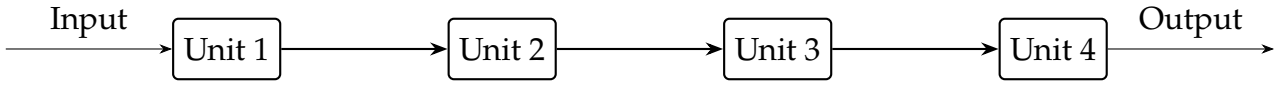


Figure 4 Diagrams linking computational units as cascades

put them in layers, so they have exponential–in–layer power of representational complexity. Idea: use layers to build a very complex input–output map that represents the solution of a computational task. Then learn/represent this map by ANN. (Perceptron.) Computational units of ANN are linear threshold units (LTU).

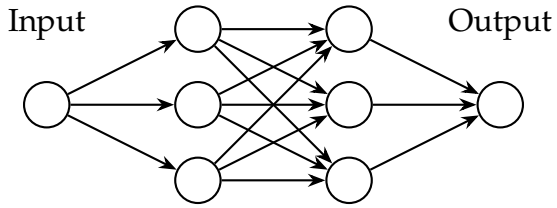


Figure 5 Diagrams linking computational units in layers

$$w^\top x = \sum_i w_i x_i > \theta \ : \ y = 1$$

$$w^\top x = \sum_i w_i x_i < \theta \ : \ y = 0$$

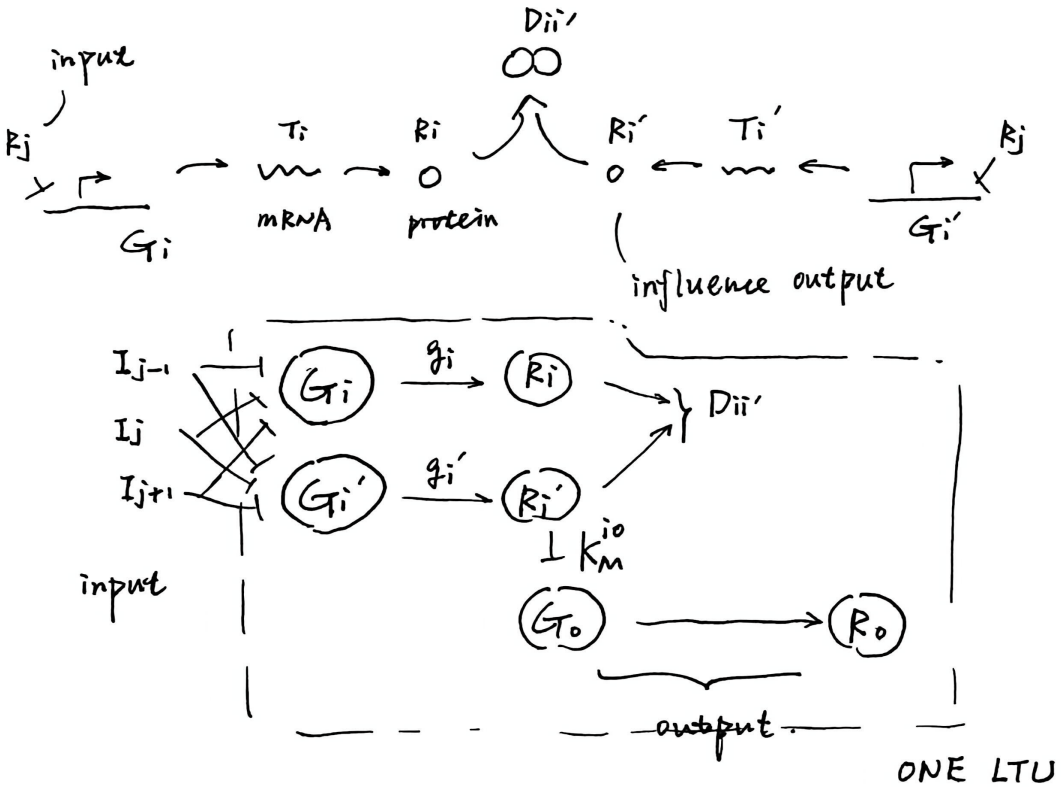


Figure 6 LTU implementation with gene regulatory network

$G_i \xrightarrow{k_{tx}} G_i + T_i$	Transcription
$T_i \xrightarrow{k_{tl}} T_i + R_i$	Translation
$T_i \xrightarrow{k_{rd}} \emptyset$	Transcript degradation
$R_i \xrightarrow{k_{pd}} \emptyset$	Regulatory species degradation
$R_j + G_i \underset{k_{ji}^-}{\overset{k_{ji}^+}{\rightleftharpoons}} G_{ij}$	Regulation of transcription
$R_i + R_j \underset{k_{ij}^-}{\overset{k_{ij}^+}{\rightleftharpoons}} D_{ij}$	Regulation among transcription factors
$D_{ij} + G_k \underset{k_k^-}{\overset{k_k^+}{\rightleftharpoons}} G_{ijk}$	Binding of TF complex to gene
$G_{ij} \xrightarrow{k_{ji}^{tx}} G_{ij} + T_i$	Result of regulation (Rate change)
$G_{ijk} \xrightarrow{k_{ijk}^{tx}} G_{ijk} + T_k$	Result of regulation (Rate change)

2.1 LTU: Titration GRN Implementation

After given the definitions of the GRN, in this section we try to make it clear that why it represents a LTU(or a neuron element) for further construction of an Artificial Neural Network.

We begin by analyzing the two important species: a specific gene (G_i) and a specific regulator (R_i). (In this case we assume the regulator is an repressor of gene). To make the intuition clear, we denote the input repressor as I (which is the resulting output of the previous layer), and the regulators in current layer would be the output variable, which we denote R.

Single Gene G_i We denote the total amount of gene as $G_{tot,i}$, consisting the free state G_i and regulated state G_{ji}. We can first solve the regulated state quantity using the equilibrium of the binding reaction:

$$I_i + G_j \to G_{ij}, \qquad K_M^{ij} := \frac{I_i G_j}{G_{ij}}.$$

Thus the regulated (bound) state is

$$G_{ij} = \frac{I_i \, G_j}{K_M^{ij}}.$$

So the total amount of gene i would be:

$$\begin{aligned} G_{tot,i} &= G_i + \sum_j G_{ij} \\ &= G_i + \sum_j \frac{I_j G_i}{K_M^{ij}} \\ &= G_i \left(1 + \sum_j \frac{1}{K_M^{ij}} I_j \right), \end{aligned} \tag{}$$

hence

$$G_i = \frac{G_{tot,i}}{1 + \sum_j \frac{1}{K_M^{ij}} I_j}.$$

Notice that the mathematical form of this sum $\sum_j \frac{1}{K_M^{ij}} I_j$ is analogous to the ANN, with the form of a weighted linear combination. (i is the current node, and the j sums up all the inputs). This is the exact result of design that we use the multiple regulator inputs to assemble the input signals of a single LTU.

Single Regulator R_i Similarly the regulator would have free state (R_i) and the muted dimer ($D_{ii'}$, formed by its complementary regulator R_i'). At steady state, the total regulator produced from gene G_i can be written as

$$R_{tot,i} = g_i G_i \frac{k_{+l} k_{+x}}{k_{pd} k_{rd}}.$$

Substituting the expression of G_i above,

$$R_{tot,i} = g_i \frac{k_{+l} k_{+x}}{k_{pd} k_{rd}} \cdot \frac{G_{tot,i}}{1 + \sum_j \frac{1}{K_M^{ij}} I_j}.$$

Note that in the last equation we assumed

$$g_i \frac{k_{+l} k_{+x}}{k_{pd} k_{rd}} \approx \frac{1}{G_{tot}},$$

it's more like a choice of simplification and this whole term represents certain "gain of protein R_i from gene G_i copy number."

This is the output of our GRN version LTU. We anticipate this to be capable of performing the "decision" function of an LTU. Recall that we would have a firing output of the neuron if the linear combination of input signals is higher than a threshold, and on the contrary the neuron would be silent if the linear combination of input signals can't achieve the threshold. This gives the graphical intuition of a "decision boundary."

In our molecular design, this exact decision boundary would be implemented by the result of a "cruel competition" between R_i and R'_i: since the binding of two regulators is so tight, we can almost say that R_i occurs only if $R_{tot,i} > R_{tot,i'}$, since otherwise it would all show up in the binding form $D_{ii'}$.

In the previous paragraphs we already analyzed the computation essence (G_i and R_i) of our GRN implementation, and we can now connect the whole body of this one LTU, starting from the decision goal $R_{tot,i} > R_{tot,i'}$:

$$
\begin{aligned}
R_{tot,i} &> R_{tot,i'} \\
\frac{\alpha_i}{1+\sum_j \frac{1}{K_M^{ij}} I_j} &> \frac{\alpha_{i'}}{1+\sum_j \frac{1}{K_M^{i'j}} I_j} \qquad \left(\alpha_i := g_i \frac{k_{+l} k_{+x}}{k_{pd} k_{rd}} G_{tot,i}\right) \\
\alpha_i \left(1+\sum_j \frac{1}{K_M^{i'j}} I_j\right) &> \alpha_{i'} \left(1+\sum_j \frac{1}{K_M^{ij}} I_j\right) \\
\sum_j \left(\frac{\alpha_i}{K_M^{i'j}} - \frac{\alpha_{i'}}{K_M^{ij}}\right) I_j &> \alpha_{i'} - \alpha_i. \qquad ()
\end{aligned}
$$

This has the LTU form "weighted sum $>$ threshold."

By comparing the last equation with the LTU definition, we can see that we successfully built a LTU element with 3 genes. So theoretically, if we want, we can now build up an ANN in biology.

2.2 Practical Considerations

The computational complexity (or specifically, the functional expression capability) of ANN scales with the number of weights, or the number of connections between neuron units. For one GRN-version LTU that we just created, this number of connection is roughly the number of transcription regulatory binding to the genes (like G_i and G'_i). So the number of connection in this network would be the typical number of gene transcription binding times the total number of genes involved.

Biological knowledge would give us a rough taste of this number of connections:

	E. coli	Yeast	Human
# TFs	300 (7%)	200 (3%)	1800
# TFs on a gene (ChIP-seq etc.)	$1 \sim 2$ (0 to 10)	5–12 (0 to 20)	10–12 (0 to 100)
# weights	10^2	10^3	10^4

The number of connections of network is at most the extend of 10^4. We can also take a look at the history of ANN, the success of deep learning and machine learning areas enables amazing ai products and changed the human society. How many connection do we need there?

Era	Representative models/tasks	# weights (order)
1990s	LeNet (LeCun); MLP on MNIST; etc.	$\sim 10^5$
2010s	Deep nets: AlexNet (2012), VGG-16 (2014), ResNet (2015) on ImageNet	10^7–10^8
2020s	Transformers: GPT-1, GPT-2, GPT-3, GPT-4	$10^8,\ 10^9,\ 10^{11},\ 10^{12}$

So with our reasoning and estimation, even human cells achieves 1/10 of LeNet, which is just for simple tasks like hand-written digit recognitions. . . . . This is somewhat unsatisfactory and disappointed: we believe that cells are definitely more powerful and efficient, and we are eager to achieve better computation with biology-implemented circuits. But I think it's no suprise that we face this challenge. We just made a hard, superficial analogy to make a molecular network exactly the same as ANN to implement the computation, and this is fairly impossible that cell happens to choose this very strategy. The success of human world ANN, what problems we meet, what we can do, how ANN address computation tasks. . . would be quite different with the situations for a cell, a biological entity.

3 Side Note: Binding is much more efficient!

When we attempt to build bio-computation machines, binding reaction network may be more powerful than the gene regulation network(GRN).

We first show that a simple tight binding would implement a ReLU function. If two species A and B bind very tightly to form complex C, then in the tight-binding limit the free amount of A is approximately

$$\mathrm{A} \approx \max(0, \mathrm{A}_{\mathrm{tot}} - \mathrm{B}_{\mathrm{tot}}) = \mathrm{ReLU}(\mathrm{A}_{\mathrm{tot}} - \mathrm{B}_{\mathrm{tot}}).$$

The number of binding reactions in cells would be much larger:

Type of binding reaction	*E. coli*	Yeast	Mammalian
protein–protein	10^4–10^5	10^5–10^6	10^6–10^7
enzyme–substrate / cofactor	10^5	—	—
protein–DNA	—	—	—
protein–RNA	—	10^4–10^5	—
protein–lipid / ion . . .	—	—	—
Total	10^5–10^6	10^6–10^7	10^7–10^8

But it still don't scales to contribute to the computation complexity. What's more, not all bindings are tight. And we should not assume all of these binding reactions serve for just "computation." After all, cells need to do a whole bunch of other tasks.

4 Computational Tasks in Biology vs Engineered Systems

4.1 Logic Gates, LTUs, and Cellular Computation

In engineered systems such as digital computers, the fundamental computational elements are **logic gates** (e.g. AND, OR, NOT, NAND), which implement Boolean operations and logical deduction. In deep artificial neural networks (ANNs), the basic units are linear threshold units (LTUs); large networks of LTUs can approximate arbitrary complex input–output maps.

A natural question is: **what are the analogous computational tasks in biological systems?** Do cells need to perform logical deduction, or fit arbitrary input–output functions the way engineered ANNs do?

One ubiquitous task is the following: in response to different combinations of environmental signals (nutrients, antibiotics, hormones, morphogens, and so on), a cell switches into different internal states, often realized as distinct gene-expression programs. Thus the cell must implement at least one nontrivial map.

$$\text{environmental inputs} \quad \longrightarrow \quad \text{cellular states / behaviors.}$$

A convenient abstraction is shown in Fig7.

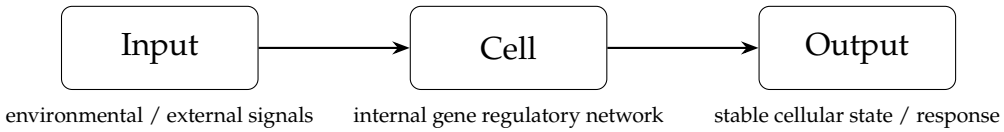


Figure 7 Abstract view of cellular computation: environmental inputs are processed by the cell into distinct output states.

4.2 Real Example: Multi-fate Differentiation

A concrete biological example is multi-fate differentiation (e.g. Zhu Ronghui, Elowitz lab, *Science* 2022). The computational task is to generate on the order of 10^2–10^3 distinct cell types using on the order of 10 transcription factors (TFs), rather than 10^2–10^3 distinct TFs.

This is achieved by *combinational encoding*. For example, if three genes $\mathrm{A}, \mathrm{B}, \mathrm{C}$ can each be high or low, there are $2^3 = 8$ possible expression patterns. More generally, N genes provide 2^{N} combinatorial patterns.

The implementation is through *multistability* in gene-expression space: each stable cell type corresponds to a distinct stable fixed point (attractor) of the underlying dynamical system. Figure 8 shows a simple schematic with three stable fixed points in the plane spanned by gene A and gene B.

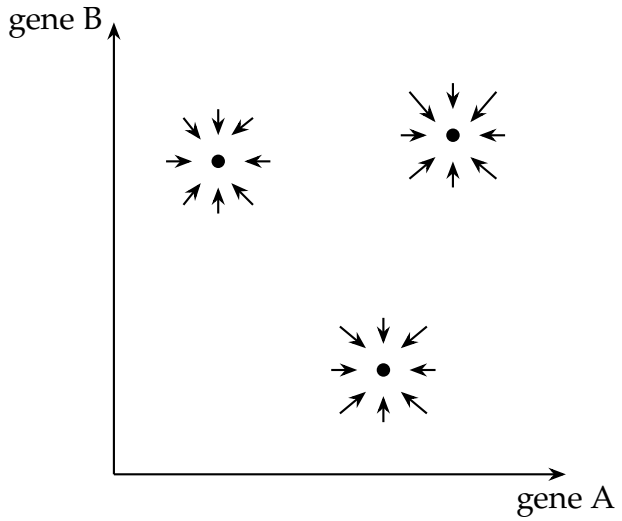


Figure 8 Multistability in the gene A–gene B plane. Filled dots are stable fixed points; surrounding arrows indicate flow of cell states toward these attractors.

Another Example of Multistability: Toggle Switch

A canonical synthetic example is the toggle switch with mutual repression (Gardner & Collins 2000). Two genes A and B repress each other's expression via their protein products.

At the gene level, the proteins bind the opposite promoter and inhibit transcription; at the protein level, A and B form a mutually repressive pair. This is summarized schematically in Fig. 9.

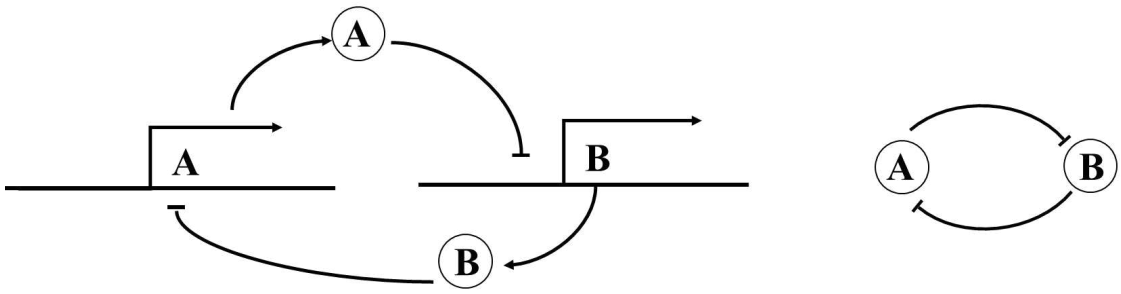


Figure 9 Mutual repression between two genes A and B and between their protein products.

Mutual Repression: Dynamics and Nullclines

We write the binding reactions

$$G_A + B \rightleftharpoons G_A^B, \qquad G_B + A \rightleftharpoons G_B^A,$$

and the dynamics for the total protein concentrations A_{tot} and B_{tot} as

$$\frac{dA_{tot}}{dt} = \nu + \beta \frac{G_A}{G_{tot}} - \delta A_{tot}, \quad ()$$

$$\frac{dB_{tot}}{dt} = \nu + \beta \frac{G_B}{G_{tot}} - \delta B_{tot}. \quad ()$$

Assuming $G_{tot} \ll K_A, K_B$,

$$G_A \approx \frac{G_{tot}}{1 + B_{tot}/K_A}, \quad ()$$

$$G_B \approx \frac{G_{tot}}{1 + A_{tot}/K_B}. \quad ()$$

From $dB_{tot}/dt = 0$ we obtain the nullcline

$$B_{tot} = \frac{1}{\delta}\left(\nu + \beta \frac{1}{1 + A_{tot}/K_A}\right).$$

A schematic of the A_{tot} and B_{tot} nullclines is shown in Fig. 10; there is only one intersection, so the system is monostable in this simple parameter regime.

To increase nonlinearity, we can let repressors be dimers (or multimers):

$$G_A + 2B \Rightarrow G_A^B, \quad G_A \approx \frac{G_{tot}}{1 + (B_{tot}/K_A)^2}.$$

However, even then we may obtain only two stable states, with a middle unstable fixed point.

4.3 Developmental Motifs and SAMI

Motifs from developmental biology often combine *self-activation* with *mutual inhibition* (sometimes referred to as SAMI). It is easy in such motifs to make one gene high and the other low, yielding two alternative stable states.

However, competitive binding for inhibition and higher-order binding such as dimerization can further sharpen response curves and introduce ultrasensitivity. This raises the question: can two genes produce more than two stable states (e.g. three stable states)?

Binding Reactions and Nullcline Analysis

We enrich the model by introducing explicit binding between A and B:

$$G_A + A \rightleftharpoons G_A^*, \quad ()$$

$$G_B + B \rightleftharpoons G_B^*, \quad ()$$

$$A + B \rightleftharpoons C_{AB}. \quad ()$$

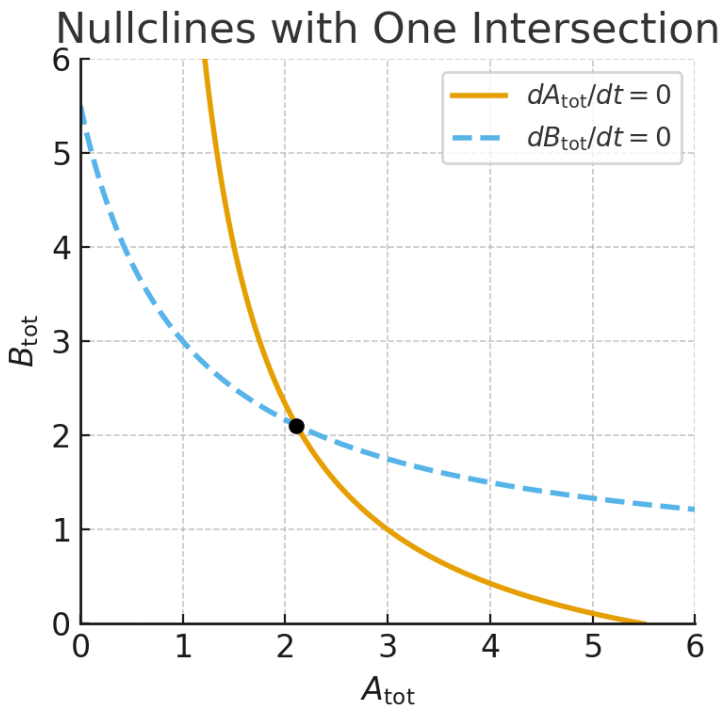


Figure 10 Nullclines for the mutual repression toggle in a simple parameter regime with a single intersection. More curvature is needed to obtain multiple fixed points.

The dynamics become

$$\frac{dA_{tot}}{dt} = \alpha + \beta \frac{G_A^*}{G_{tot}} - \delta A_{tot}, \qquad ()$$

$$\frac{dB_{tot}}{dt} = \alpha + \beta \frac{G_B^*}{G_{tot}} - \delta B_{tot}. \qquad ()$$

Using a quasi–steady-state approximation,

$$G_A^* \approx \frac{G_{tot} A}{A + K_A}, \qquad ()$$

$$G_B^* \approx \frac{G_{tot} B}{B + K_B}, \qquad ()$$

and assuming $C_{AB} \ll K_A, K_B$ so that

$$A_{tot} \approx A + C_{AB}, \qquad B_{tot} \approx B + C_{AB}.$$

For the A_{tot} nullcline,

$$\alpha + \beta \frac{A}{A + K_A} = \delta A_{tot}.$$

We consider different regimes:

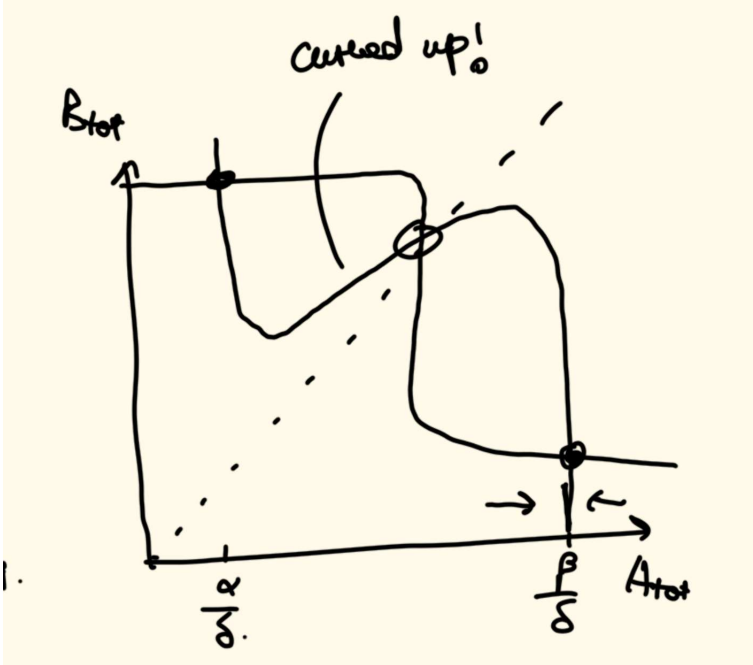


Figure 11 With stronger nonlinearity, the nullclines can intersect three times; typically the middle fixed point is unstable and the outer ones are stable, giving bistability.

1. A_{tot} very low:
$$\alpha \gg \beta \frac{A}{A + K_A} \quad \Rightarrow \quad A_{\text{tot}} \approx \frac{\alpha}{\delta}.$$

2. A_{tot} very large:
$$\alpha \ll \beta \frac{A}{A + K_A} \quad \Rightarrow \quad \beta \frac{A}{A + K_A} = \delta A_{\text{tot}}.$$

   (a) If B_{tot} is small, then $A \approx A_{\text{tot}}$ and
   $$A_{\text{tot}} \sim \frac{\beta}{\delta}.$$

   (b) If B_{tot} is comparable with A_{tot}, then
   $$A \approx A_{\text{tot}} - B_{\text{tot}} \ll K_A,$$
   giving
   $$\frac{\beta}{K_A}(A_{\text{tot}} - B_{\text{tot}}) = \delta A_{\text{tot}},$$
   hence
   $$B_{\text{tot}} = \left(1 - \frac{\delta K_A}{\beta}\right) A_{\text{tot}}.$$

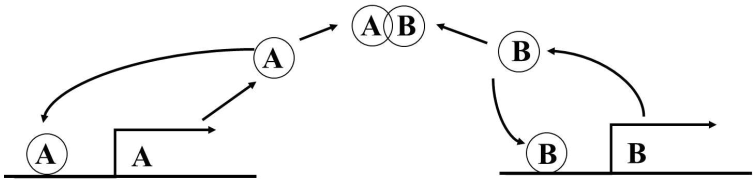


Figure 12 Schematic nullcline of A_{tot} in the presence of A–B binding; in the relevant region the slope can be tuned to be less than 1.

Despite the additional binding, the system may still be only bistable unless we introduce stronger ultrasensitivity in the activation step.

Homodimer Activation, Heterodimer Inhibition and Scaling

We now consider homodimer activation together with heterodimer inhibition. Dimers A_2 and B_2 activate their own promoters, whereas the heterodimer C_{AB} is inactive:

$$A + A \rightleftharpoons A_2, \tag{}$$
$$B + B \rightleftharpoons B_2, \tag{}$$
$$A + B \rightleftharpoons C_{AB}, \tag{}$$
$$A_2 + G_A \rightleftharpoons G_A^*, \tag{}$$
$$B_2 + G_B \rightleftharpoons G_B^*. \tag{}$$

$$A_{tot} = A + 2A_2 + C_{AB}, \qquad B_{tot} = B + 2B_2 + C_{AB}.$$

Assuming $G_{tot} \ll K_{GA}, K_{GB}$ and that A_2, B_2 never dominate, one can show that in the regime where B_{tot} is comparable to A_{tot},

$$A \approx A_{tot} - B_{tot}, \qquad A_2 \approx \frac{(A_{tot} - B_{tot})^2}{K_A},$$

and setting $dA_{tot}/dt = 0$ yields

$$\beta \frac{(A_{tot} - B_{tot})^2}{K_A K_{GA}} = \delta A_{tot}.$$

This corresponds to an effective A^2-type activation that can bend the nullcline upward and produce three intersections (three stable states) for two genes.

Ultrasensitivity from binding and dimerization thus adds extra stable states. Roughly, if N transcription factors participate in such combinatorial binding networks, one can achieve

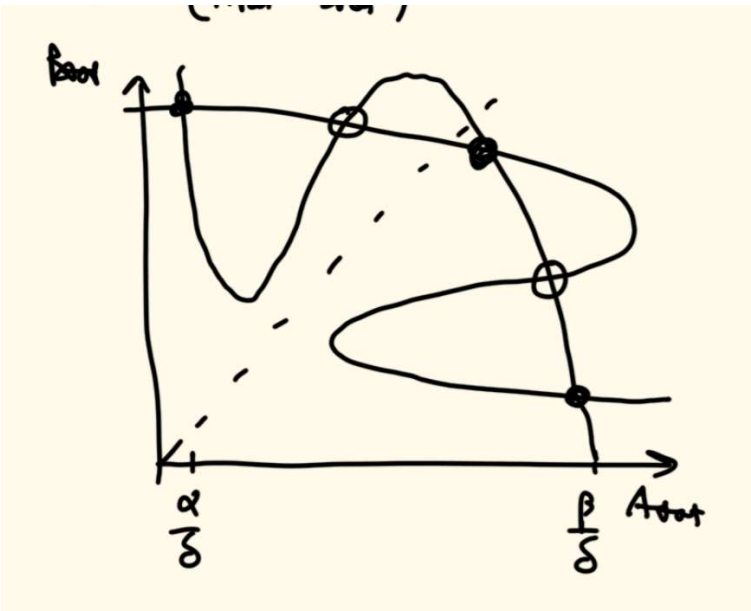


Figure 13 A^2-like activation bends the nullcline upward, allowing three intersections and thus three stable states for two genes.

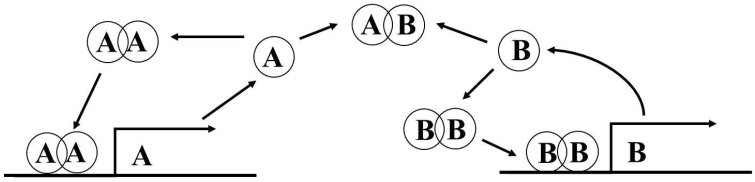


Figure 14 Homodimer activation (A_2, B_2) and heterodimer inhibition via C_{AB}.

on the order of 2^N stable expression patterns. For $N \approx 10$, this yields 10^2 to 10^3 stable cell types—comparable to the diversity of cell types in many multicellular organisms.

Cells therefore perform efficient and complex computation primarily through binding networks: many binding reactions and comparatively few catalytic reactions, a very different "hardware" from digital electronics but capable of similarly rich computational behavior.