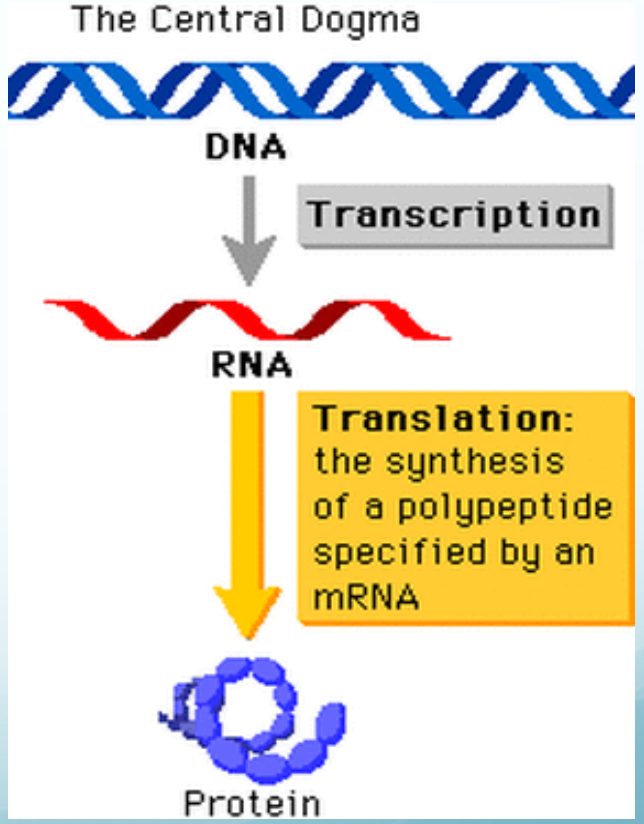
# Modeling of the *lac* operon in *E. coli*

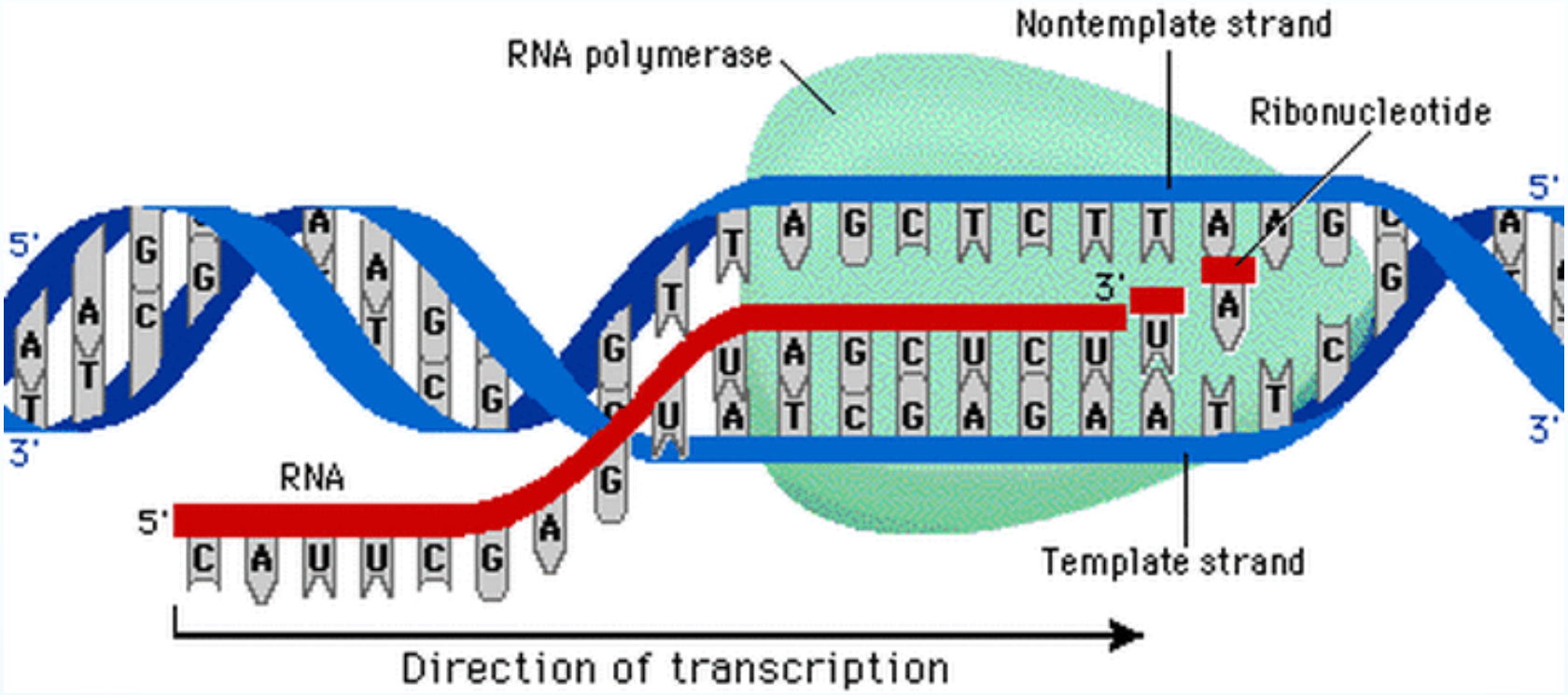
Matthew Macauley
Clemson University

# Gene expression

- Gene expression is a process that takes gene info and creates a functional gene product (e.g., a protein).
- Gene Expression is a 2-step process:
  1) transcription of genes (messenger RNA synthesis)
  2) translation of genes (protein synthesis)
- DNA consists of bases A, C, G, T.
- RNA consists of bases A, C, G, U.
- Proteins are long chains of amino acids.
- Gene expression is used by all known life forms.

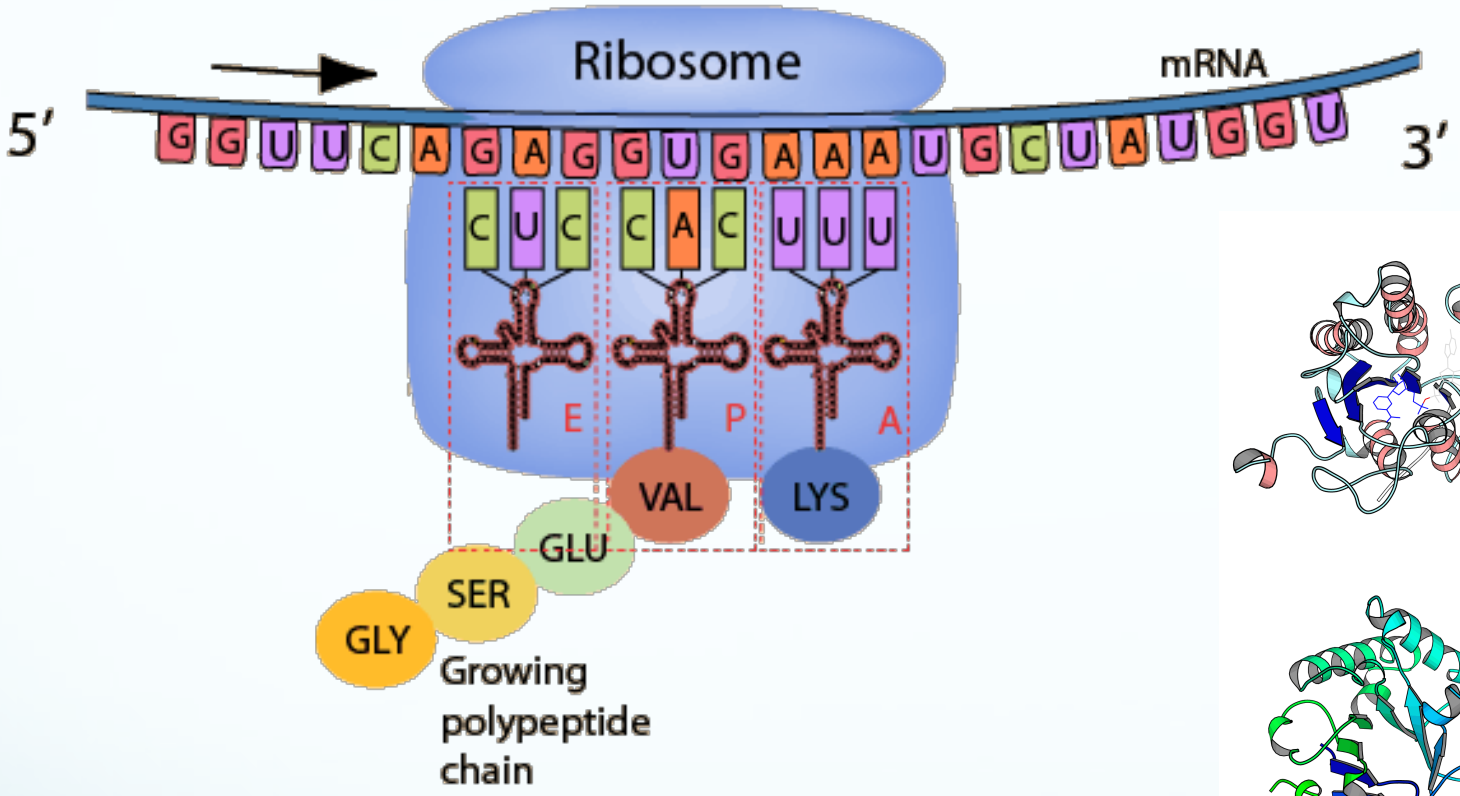


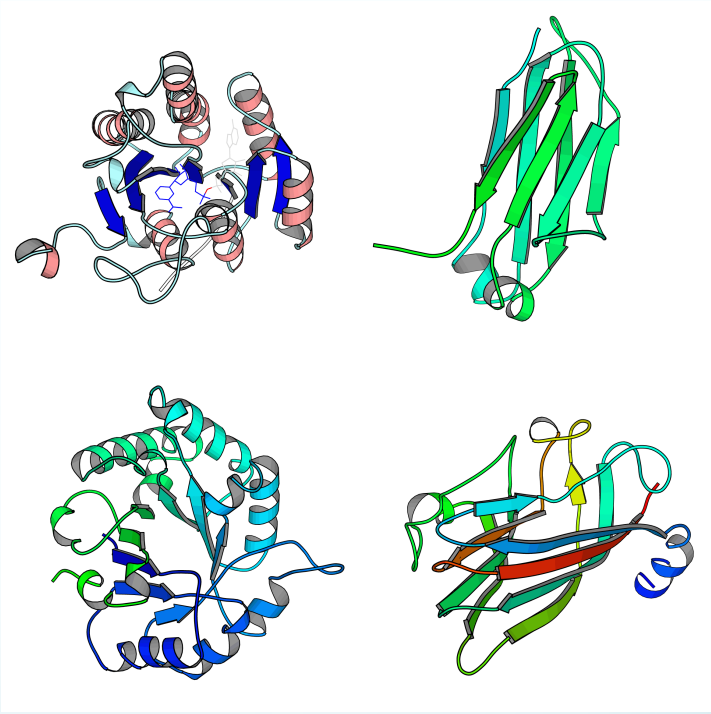
# Transcription




- Transcription occurs inside the cell nucleus.
- A helicase enzyme binds to and “unzips” DNA to read it.
- DNA is copied into mRNA.
- Segments of RNA not needed for protein coding are removed.

# Translation



- During translation, the mRNA is read by ribosomes.
- Each triple of RNA bases codes for an amino acid.
- The result is a protein: a long chain of amino acids.
- Proteins fold into a 3-D shape which determine their function

# Gene expression

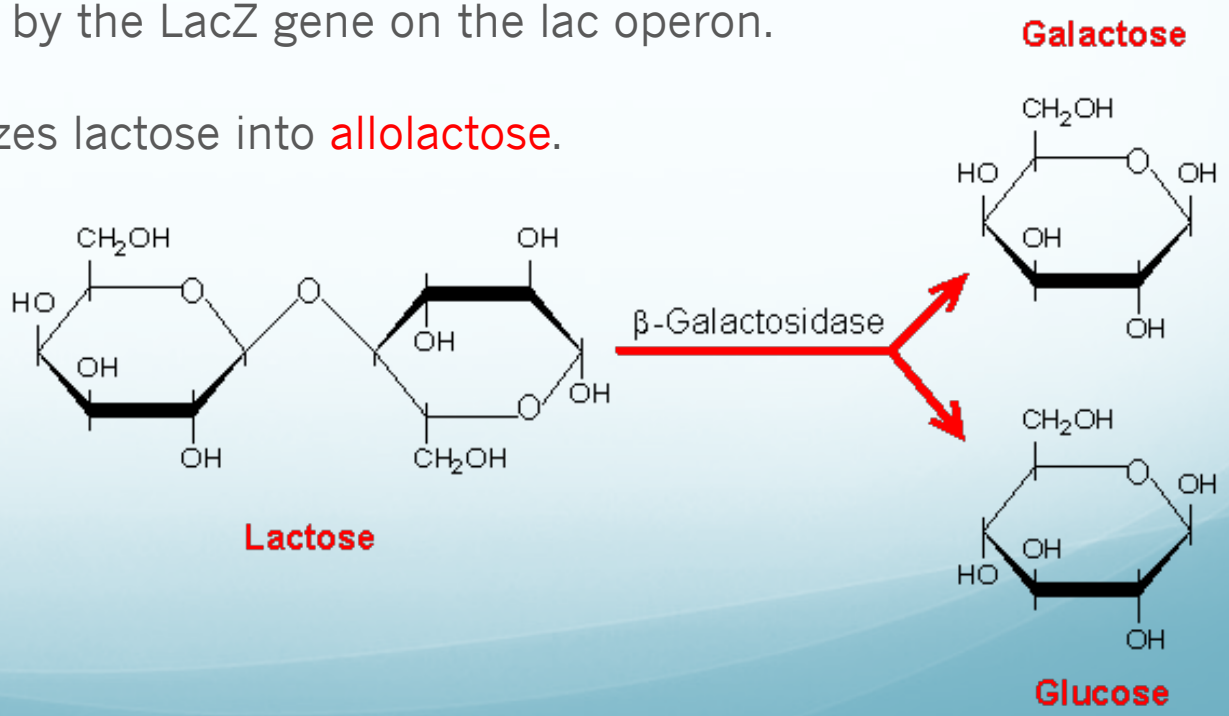
- The expression level is the rate at which a gene is being expressed.
- Housekeeping genes are continuously expressed, as they are essential for basic life processes.
- Regulated genes are expressed only under certain outside factors (environmental, physiological, etc.). Expression is controlled by the cell.
- It is easiest to control gene regulation by affecting transcription.
- One way to block transcription is for repressor proteins bind to the DNA or RNA.
- **Goal**: Understand the complex cell behaviors of gene regulation, which is the process of turning on/off certain genes depending on the requirements of the organism.

# The *lac* operon in *E. coli*

- An operon is a region of DNA that contains a cluster of genes that are transcribed together.
- *Escherichia coli* is a bacterium in the gut of mammals and birds. Its genome has been sequenced and its physiology is well-understood.
- The lactose (*lac*) operon controls the transport and metabolism of lactose in *E. coli*.
- The *lac* operon was discovered by Francois Jacob and Jacques Monod in 1961, which earned them the Nobel Prize.
- The *lac* operon was the first operon discovered and is the most widely studied mechanism of gene regulation.
- The *lac* operon is used as a "test system" for models of gene regulation.
- DNA replication and gene expression were all studied in *E. coli* before they were studied in eukaryotic cells.

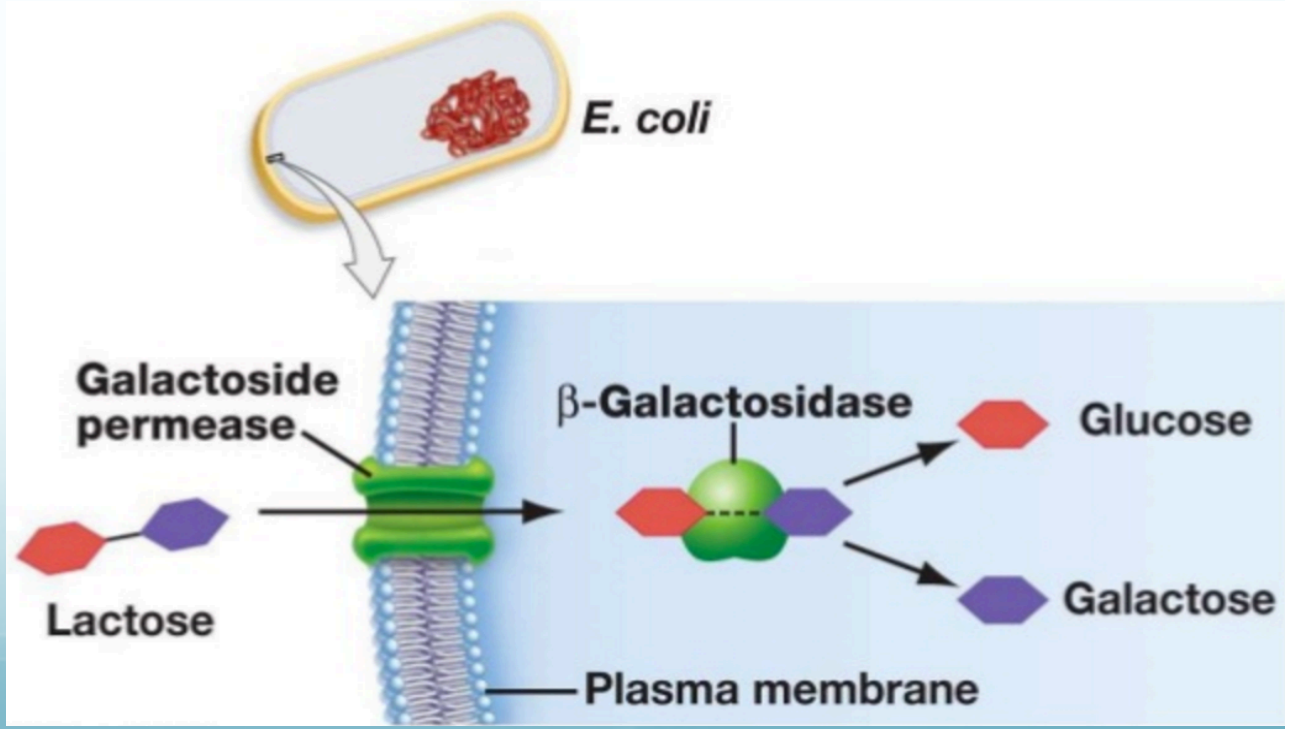
# Lactose and β–galactosidase

- When a host consumes milk, *E. coli* is exposed to lactose (milk sugar).
- Lactose consists of one glucose sugar linked to one galactose sugar.
- If both glucose and lactose are available, then glucose is the preferred energy source.
- Before lactose can be used as energy, the β–galactosidase enzyme is needed to break it down.
- β–galactosidase is encoded by the LacZ gene on the lac operon.
- β–galactosidase also catalyzes lactose into allolactose.

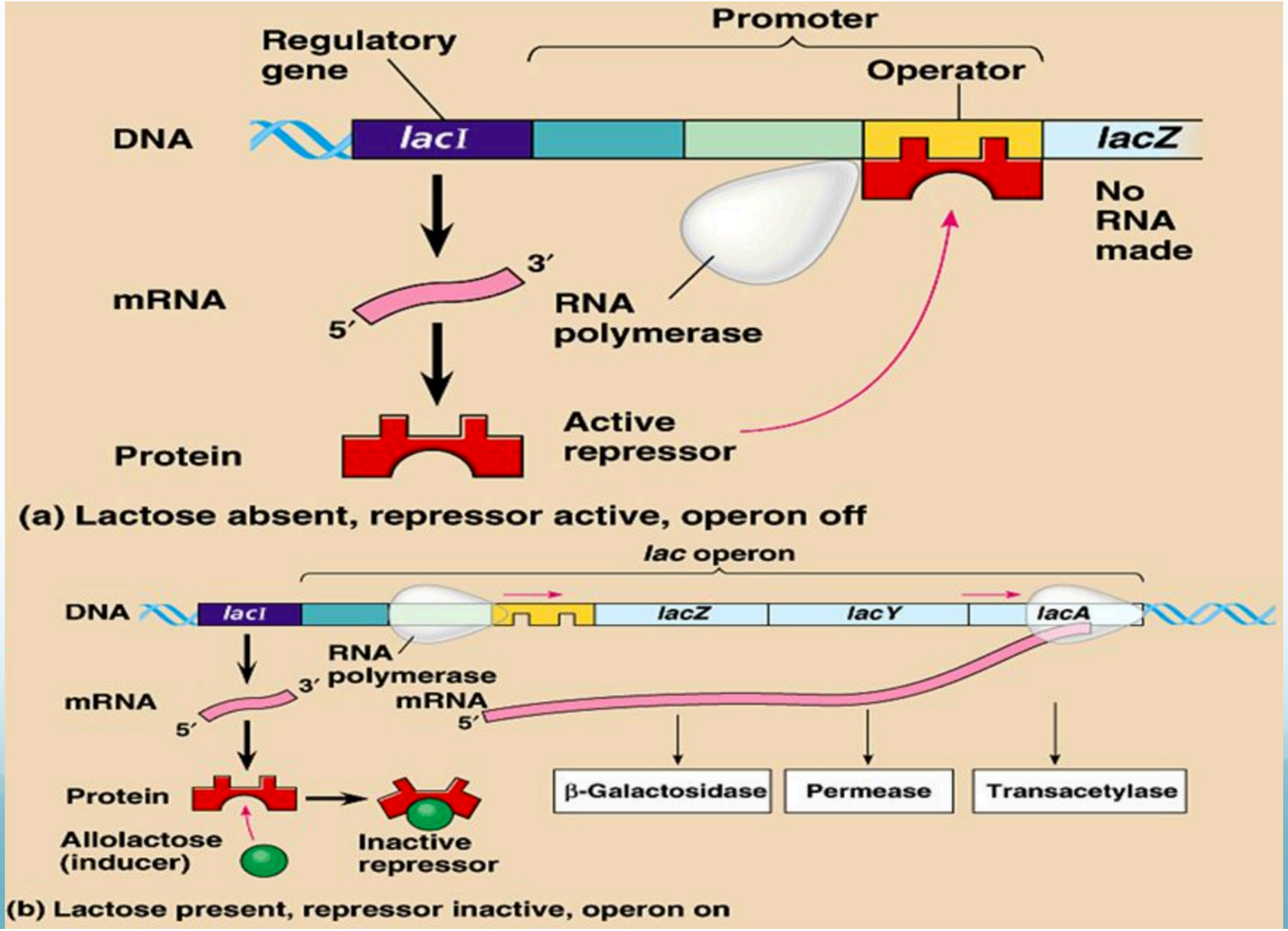


# Transporter protein

- To bring lactose into the cell, a transport protein, called *lac* permease, is required.
- This protein is encoded by the LacY gene on the *lac* operon.
- If lactose is not present, then neither of the following are produced:
  1) β–galactosidase (LacZ gene)
  2) *lac* permease (LacY gene)
- In this case, the *lac* operon is OFF.



# The *lac* operon

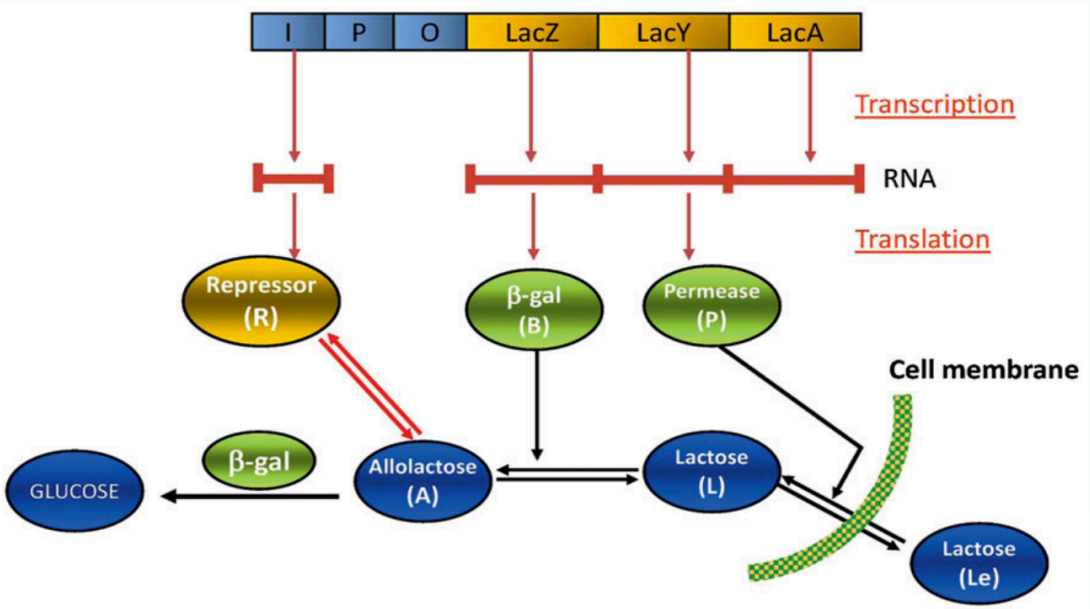


# with lactose and no glucose

- Lactose is brought into the cell by the *lac* permease transporter protein
- β–galactosidase breaks up lactose into glucose and galactose..
- β–galactosidase also converts lactose into allolactose.
- Allolactose binds to the *lac* repressor protein, preventing it from binding to the operator region of the genome.
- Transcription begins: mRNA encoding the *lac* genes is produced.
- Lac proteins are produced, and more lactose is brought into the cell. (The operon is ON.)
- Eventually, all lactose is used up, so there will be no more allolactose.
- The *lac* repressor can now bind to the operator, so mRNA transcription stops. (The operon has turned itself OFF.)

# *An* ODE *lac* operon model

- M: mRNA
- B: β–galactosidase
- A: allolactose
- P: transporter protein
- L: lactose



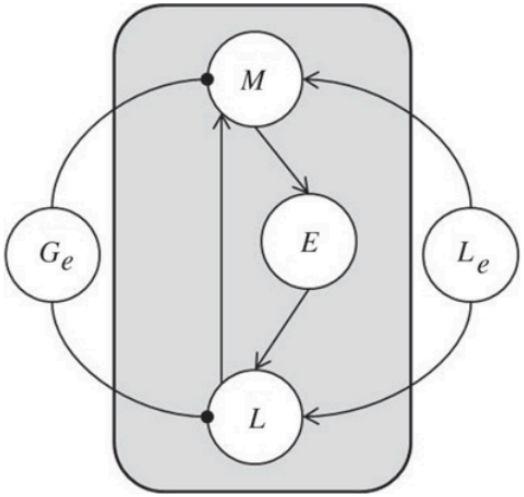

$$
\begin{aligned}
\frac{dM}{dt} &= \alpha_M \frac{1 + K_1 \left(e^{-\mu\tau_M} A_{\tau_M}\right)^n}{K + K_1 \left(e^{-\mu\tau_M} A_{\tau_M}\right)^n} + \Gamma_0 - \widetilde{\gamma_M} M \\
\frac{dB}{dt} &= \alpha_B e^{-\mu\tau_B} M_{\tau_B} - \widetilde{\gamma_B} B \\
\frac{dA}{dt} &= \alpha_A B \frac{L}{K_L + L} - \beta_A B \frac{A}{K_A + A} - \widetilde{\gamma_A} A \\
\frac{dP}{dt} &= \alpha_P e^{-\mu(\tau_B + \tau_P)} M_{\tau_B + \tau_P} - \widetilde{\gamma_P} P \\
\frac{dL}{dt} &= \alpha_L P \frac{L_e}{K_{L_e} + L_e} - \beta_{L_1} P \frac{L}{K_{L_1} + L} - \alpha_A B \frac{L}{K_L + L} - \widetilde{\gamma_L} L
\end{aligned}
$$

# Downsides of an ODE model

- Very mathematically technical.
- Too hard to solve explicitly. Numerical methods are needed.
- MANY experimentally determined "rate constants" (I count 22...)
- Often, these rate constants aren't known even up to orders of magnitude.

$$
\begin{aligned}
\frac{dM}{dt} &= \alpha_M \frac{1 + K_1 \left(e^{-\mu\tau_M} A_{\tau_M}\right)^n}{K + K_1 \left(e^{-\mu\tau_M} A_{\tau_M}\right)^n} + \Gamma_0 - \widetilde{\gamma_M} M \\
\frac{dB}{dt} &= \alpha_B e^{-\mu\tau_B} M_{\tau_B} - \widetilde{\gamma_B} B \\
\frac{dA}{dt} &= \alpha_A B \frac{L}{K_L + L} - \beta_A B \frac{A}{K_A + A} - \widetilde{\gamma_A} A \\
\frac{dP}{dt} &= \alpha_P e^{-\mu(\tau_B + \tau_P)} M_{\tau_B + \tau_P} - \widetilde{\gamma_P} P \\
\frac{dL}{dt} &= \alpha_L P \frac{L_e}{K_{L_e} + L_e} - \beta_{L_1} P \frac{L}{K_{L_1} + L} - \alpha_A B \frac{L}{K_L + L} - \widetilde{\gamma_L} L
\end{aligned}
$$

# A Boolean approach




- Let's assume everything is "Boolean" (0 or 1):
  - Gene products are either present or absent
  - Enzyme concentrations are either high or low.
  - The operon is either ON or OFF.

- mRNA is transcribed (M=1) if there is no external glucose (G=0), *and* either internal lactose (L=1) or external lactose ($L_e$=1) are present.

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

- The LacY and LacZ gene products (E=1) will be produced if mRNA is available (M=1).

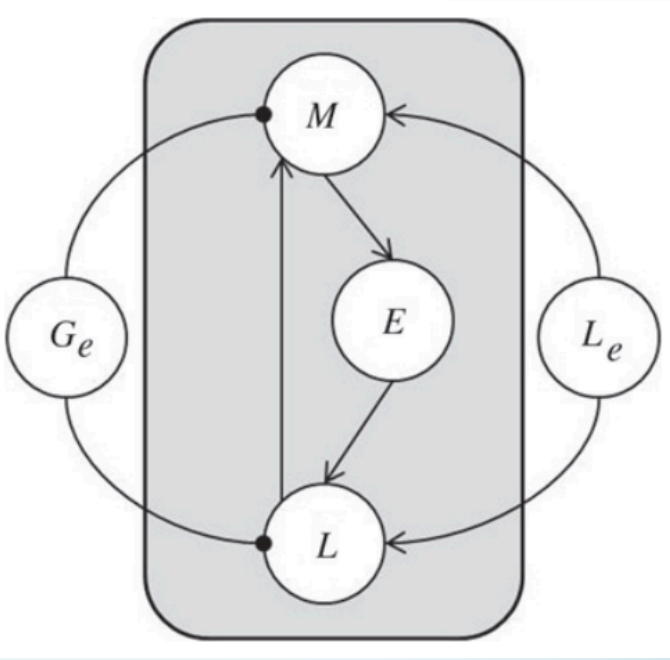
$$x_E(t+1) = f_E(t+1) = M(t)$$

- Lactose will be present in the cell if there is no external glucose ($G_e$=0), and either of the following holds:
  - ✓ External lactose is present ($L_e$=1) and *lac* permease (E=1) is available.
  - ✓ Internal lactose is present (L=1), but β–galactosidase is absent (E=0).

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

# Comments on the Boolean model

- We have two "types" of Boolean quantities:
  - mRNA (M), lac gene products (E), and internal lactose (L) are variables.
  - External glucose ($G_e$) and lactose ($L_e$) are parameters (constants).
- Variables and parameters are drawn as nodes.
- Interactions can be drawn as signed edges.
- A signed graph called the wiring diagram describes the dependencies of the variables.
- Time is discrete: t = 0, 1, 2, ....




$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

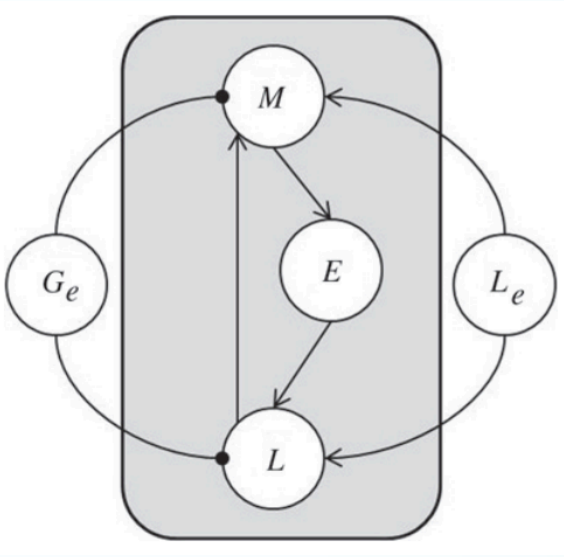
$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

- Assume that the variables are updated synchronously.

# How to analyze a Boolean model

- At the bare minimum, we should expect:
  - Lactose absent => operon OFF.
  - Lactose present, glucose absent => operon ON.
  - Lactose and glucose present => operon OFF.




$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

- The state space (or phase space) is the directed graph (V, T), where

$$V = \{(x_M, x_E, x_L) : x_i \in \{0,1\}\} \qquad T = \{(x, f(x)) : x \in V\}$$

- We'll draw the state space for all four choices of the parameters:
  - ($L_e$, $G_e$) = (0, 0). We hope to end up in a fixed point (0,0,0).
  - ($L_e$, $G_e$) = (0, 1). We hope to end up in a fixed point (0,0,0).
  - ($L_e$, $G_e$) = (1, 0). We hope to end up in a fixed point (1,1,1).
  - ($L_e$, $G_e$) = (1, 1). We hope to end up in a fixed point (0,0,0).

# How to analyze a Boolean model

- We can plot the state space using the "Cyclone" software package: at http://cyclone.algorun.org/.

- First, we need to convert our logical functions into polynomials.

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$
$$x_E(t+1) = f_E(t+1) = M(t)$$
$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

- Here is the relationship between Boolean logic and polynomial algebra:

| Boolean operations | logical form | polynomial form |
|---|---|---|
| AND | $z = x \wedge y$ | $z = xy$ |
| OR | $z = x \vee y$ | $z = x + y + xy$ |
| NOT | $z = \overline{x}$ | $z = 1 + x$ |

- Also, everything is modulo 2, so 1+1=0, and 1=-1, and $x^2=x$, and thus $x(x+1)=0$.

ALGORUN | Cyclone | Run | About | Web API | Help

# Cyclone

Calculate Dynamics of a discrete dynamical system using exhaustive search

More Information: https://github.com/PlantSimLab/cyclone

Packaged by: Abdelrahman Hosny

## Authors

Reinhard Laubenbacher (PI)
Center for Quantitative Medicine

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

**input**

```
1  MODEL NAME: 3-variable lac operon model
2  SIMULATION NAME: Lactose but not glucose
3  NUMBER OF VARIABLES: 3
4  VARIABLE NAMES: x1 x2 x3
5  NUMBER OF STATES: 2 2 2
6  SPEED OF VARIABLES: 1 1 1
7
8  f1 = 1
9  f2 = x1
10 f3 = x2 + x3 + x2*x3
```

**output** | visualization

```
1  digraph test {
2  node0 [label=" 0 0 0"];
3  node1 [label=" 0 0 1"];
4  node2 [label=" 0 1 0"];
5  node3 [label=" 0 1 1"];
6  node4 [label=" 1 0 0"];
7  node5 [label=" 1 0 1"];
8  node6 [label=" 1 1 0"];
9  node7 [label=" 1 1 1"];
10 node0 -> node0
11 node1 -> node5
12 node2 -> node1
13 node3 -> node5
14 node4 -> node2
15 node5 -> node7
16 node6 -> node3
17 node7 -> node7
18 }
19
```

Load sample data | Change parameters | Reset computation | RUN COMPUTATION

ALGORUN | Cyclone | Run | About | Web API | Help

# Cyclone

Calculate Dynamics of a discrete dynamical system using exhaustive search

More Information: https://github.com/PlantSimLab/cyclone

Packaged by: Abdelrahman Hosny

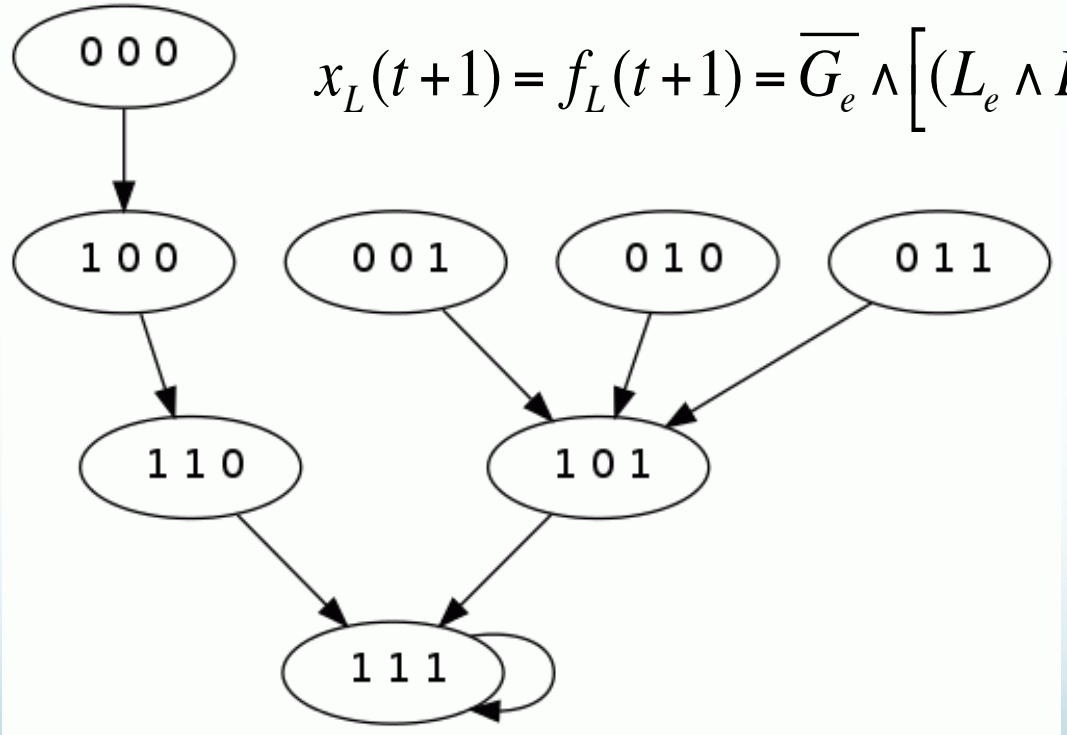
## Authors

Reinhard Laubenbacher (PI)
Center for Quantitative Medicine

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$




State space when ($G_e$, $L_e$) = (0, 1). The operon is ON.

ALGORUN | Cyclone | Run | About | Web API | Help

# Cyclone

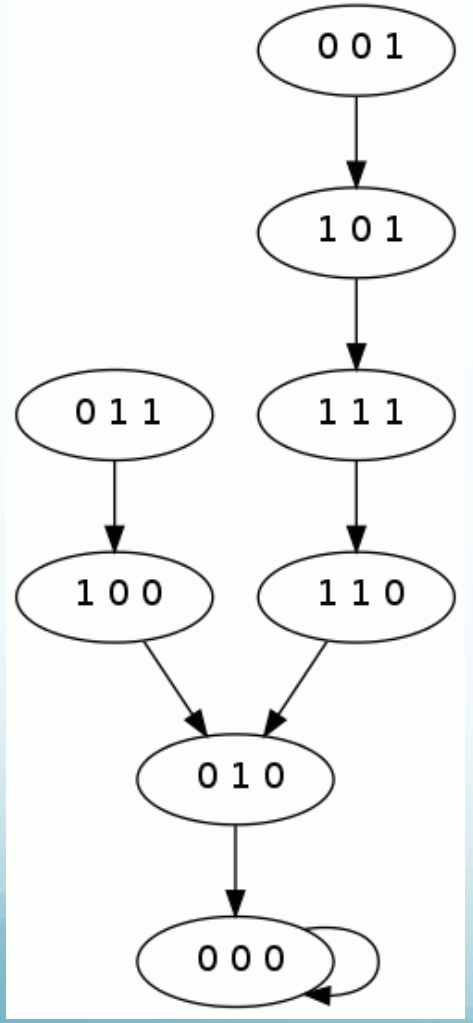
Calculate Dynamics of a discrete dynamical system using exhaustive search

More Information: https://github.com/PlantSimLab/cyclone

Packaged by: Abdelrahman Hosny

## Authors

Reinhard Laubenbacher (PI)
Center for Quantitative Medicine




$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$

State space when ($G_e$, $L_e$) = (0, 0).

The operon is OFF.

ALGORUN | Cyclone | Run | About | Web API | Help

# Cyclone

Calculate Dynamics of a discrete dynamical system using exhaustive search

More Information: https://github.com/PlantSimLab/cyclone

Packaged by: Abdelrahman Hosny

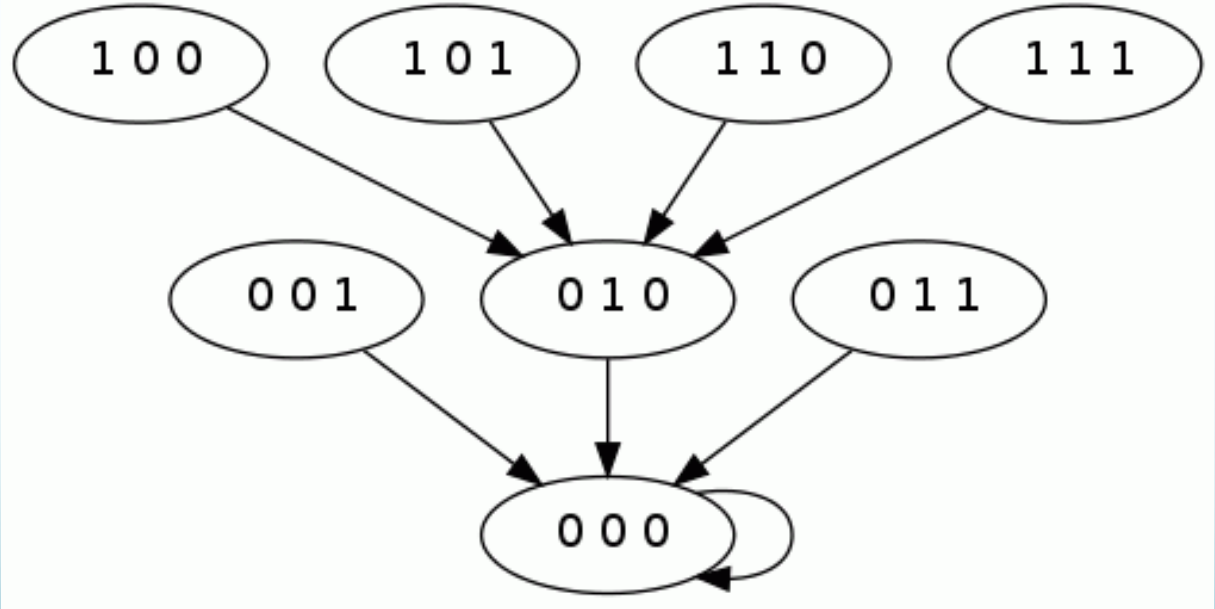
## Authors

Reinhard Laubenbacher (PI)
Center for Quantitative Medicine

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[(L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)})\right]$$




State space when ($G_e$, $L_e$) = (1, 0). The operon is OFF.

ALGORUN | Cyclone | Run | About | Web API | Help

# Cyclone

Calculate Dynamics of a discrete dynamical system using exhaustive search

More Information: https://github.com/PlantSimLab/cyclone

Packaged by: Abdelrahman Hosny

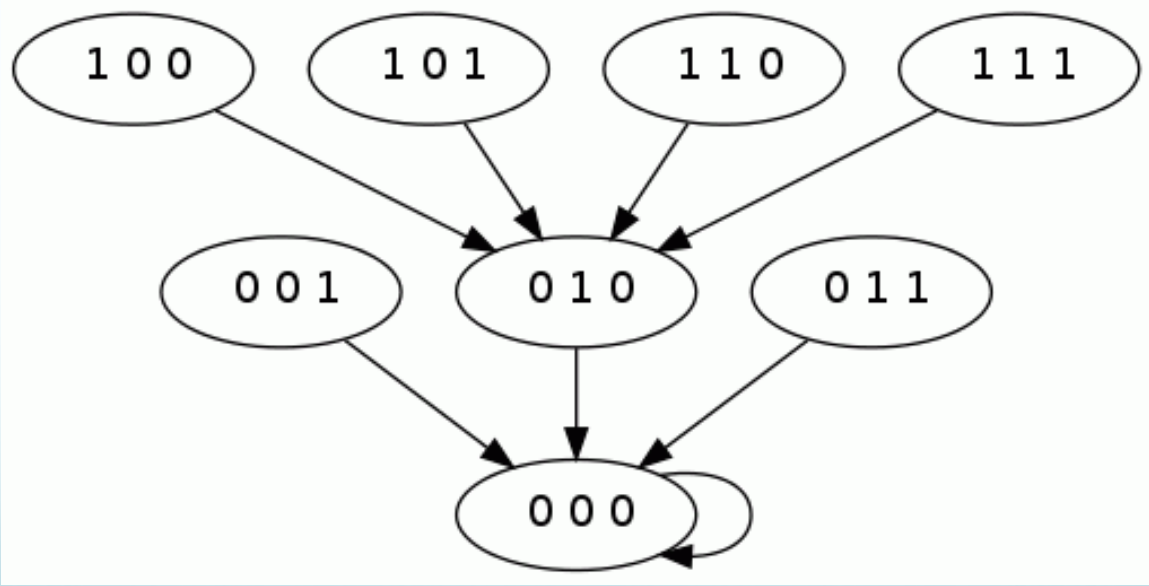
## Authors

Reinhard Laubenbacher (PI)
Center for Quantitative Medicine

$$x_M(t+1) = f_M(t+1) = \overline{G_e} \wedge (L(t) \vee L_e)$$

$$x_E(t+1) = f_E(t+1) = M(t)$$

$$x_L(t+1) = f_L(t+1) = \overline{G_e} \wedge \left[ (L_e \wedge E(t)) \vee (L(t) \wedge \overline{E(t)}) \right]$$




State space when ($G_e$, $L_e$) = (1, 1). The operon is OFF.

# Summary so far

- Gene regulatory networks consist of a collection of gene products that interact with each other to control a specific cell function.
- Classically, these have been modeled quantitatively with differential equations (continuous models).
- Boolean networks take a different approach. They are discrete models that are inherently qualitative.
- The state space graph encodes all of the dynamics. The most important features are the fixed points, and a necessary step in model validation is to check that they are biologically meaningful.
- The model of the *lac* operon shown here is a "toy model". Next, we will see more complicated models of the *lac* operon that capture intricate biological features of these systems.
- Modeling with Boolean logic is a relatively new concept, first done in the 1970s. It is a popular research topic in the field of systems biology.

# A more refined model

- Our first model only used 3 variables: mRNA (M), enzymes (E), and lactose (L).
- Let's propose a new model with 5 variables:
  - M: mRNA
  - B: β–galactosidase
  - A: allolactose
  - L: intracellular lactose
  - P: *lac* permease (transporter protein)

$$f_M = A$$
$$f_B = M$$
$$f_A = A \vee (L \wedge B)$$
$$f_L = P \vee (L \wedge \overline{B})$$
$$f_P = M$$

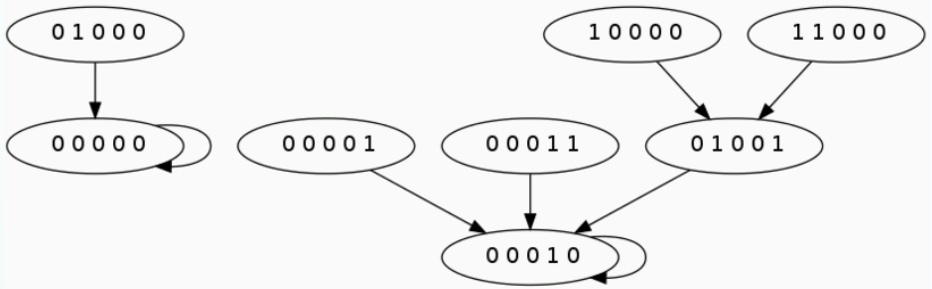
- Assumptions
  - Extracellular lactose is always available.
  - Extracellular glucose is always unavailable.
  - Translation and transcription require one unit of time.
  - Protein and mRNA degradation require one unit of time
  - Lactose metabolism require one unit of time

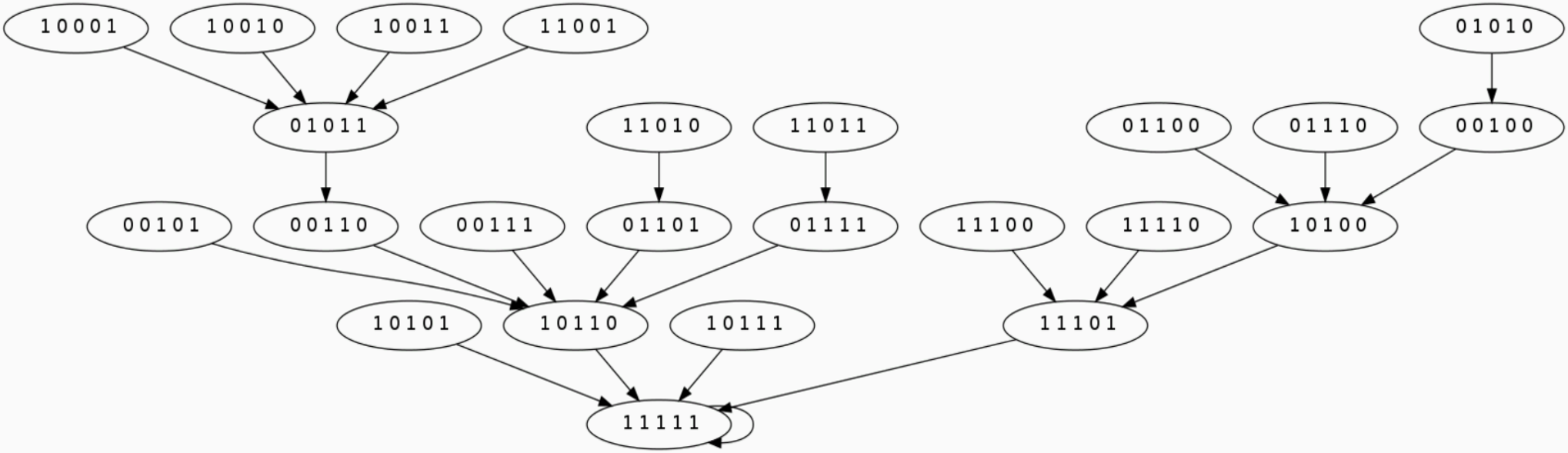
# Using Cyclone to compute the state space

$$f_M = A$$
$$f_B = M$$
$$f_A = A \vee (L \wedge B)$$
$$f_L = P \vee (L \wedge \overline{B})$$
$$f_P = M$$

input

```
1  MODEL NAME: lac-operon
2  SIMULATION NAME: sim1
3  NUMBER OF VARIABLES: 5
4  VARIABLE NAMES: M B A L P
5  NUMBER OF STATES: 2 2 2 2 2
6  SPEED OF VARIABLES: 1 1 1 1 1
7
8  f1 = A
9  f2 = M
10 f3 = B*A*L + B*L + A
11 f4 = B*L*P + B*L + L*P + L + P
12 f5 = M
```



# Problems with our refined model

- Model variables:
  - M: mRNA
  - B: β–galactosidase
  - A: allolactose
  - L: intracellular lactose
  - P: *lac* permease (transporter protein)

$$f_M = A$$
$$f_B = M$$
$$f_A = A \vee (L \wedge B)$$
$$f_L = P \vee (L \wedge \overline{B})$$
$$f_P = M$$

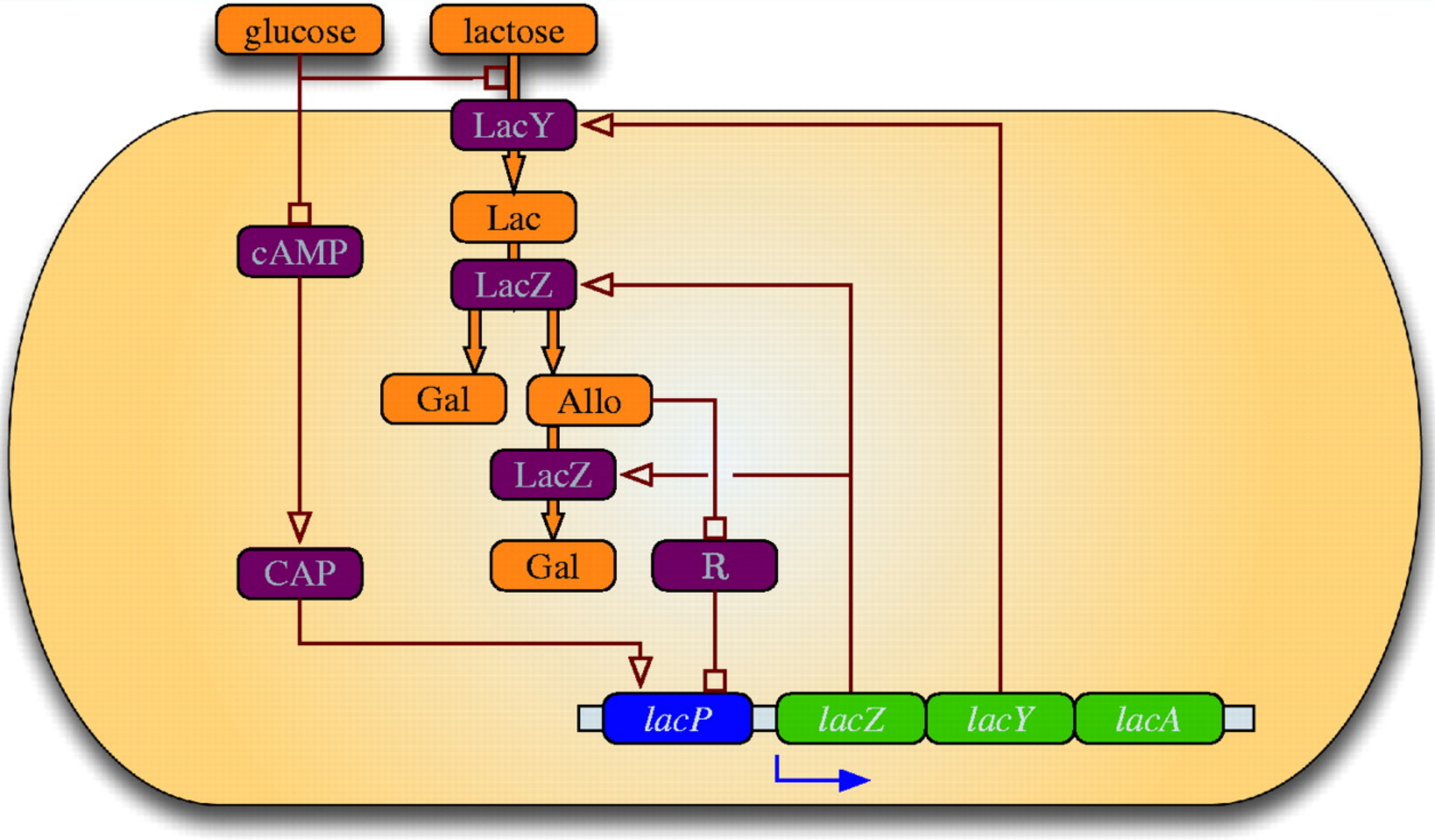
- Problems:
  - The fixed point (M,B,A,L,P) = (0,0,0,0,0) should not happen with lactose present but not glucose. [though let's try to justify this…]
  - The fixed point (M,B,A,L,P) = (0,0,0,1,0) is not biologically feasible: it would describe a scenario where the bacterium does not metabolize intracellular lactose.

- Conclusion: *The model fails the initial testing and validation, and is in need of modification*. (Homework!)

# Catabolite repression

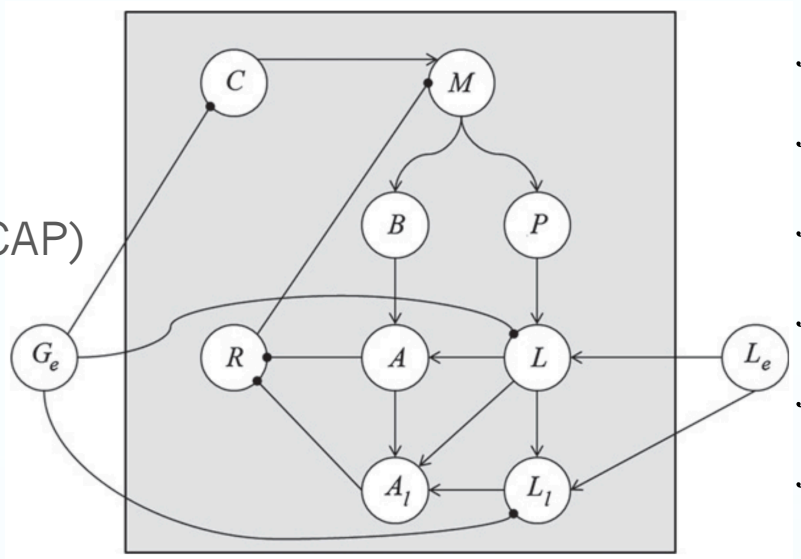
- We haven't yet discussed the cellular mechanism that turns the *lac* operon OFF when both glucose and lactose are present. This is done by catabolite repression.
- The *lac* operon promoter region has 2 binding sites:
  - One for RNA polymerase (this "unzips" and reads the DNA)
  - One for the CAP-cAMP complex. This is a complex of two molecules: catabolite activator protein (CAP), and the cyclic AMP receptor protein (cAMP, or *crp*).
- Binding of the CAP-cAMP complex is required for transcription for the *lac* operon.
- Intracellular glucose causes the cAMP concentration to decrease.
- When cAMP levels get too low, so do CAP-cAMP complex levels.
- Without the CAP-cAMP complex, the promoter is inactivated, and the *lac* operon is OFF.

# *Lac* operon gene regulatory network



# A more refined model

- Variables:
  - M: mRNA
  - P: *lac* permease
  - B: β–galactosidase
  - C: catabolite activator protein (CAP)
  - R: repressor protein (LacI)
  - A: high allolactose
  - $A_m$: at least med. allolactose
  - L: high (intracellular) lactose
  - $L_m$: at least med. levels of lactose



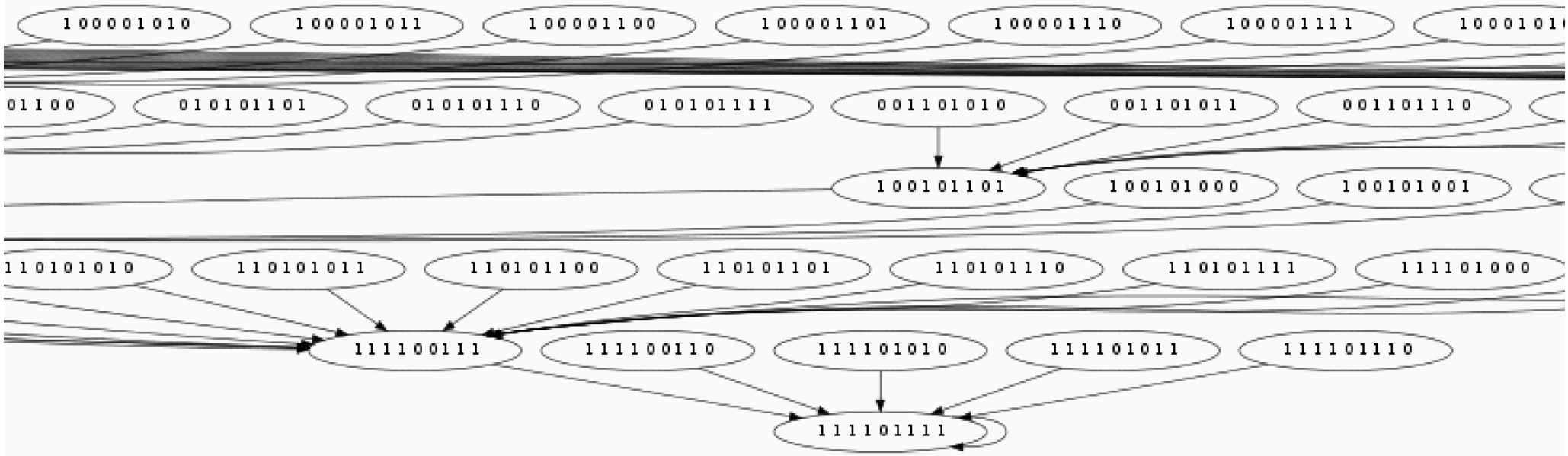

$$f_M = \overline{R} \wedge C$$
$$f_P = M$$
$$f_B = M$$
$$f_C = \overline{G_e}$$
$$f_R = \overline{A} \wedge \overline{A_m}$$
$$f_A = L \wedge B$$
$$f_{A_m} = A \vee L \vee L_m$$
$$f_L = \overline{G_e} \wedge P \wedge L_e$$
$$f_{L_m} = \overline{G_e} \wedge (L \vee L_e)$$

- Assumptions:
  - Transcription and translation require 1 unit of time.
  - Degradation of all mRNA and proteins occur in 1 time-step.
  - High levels of lactose or allolactose at any time *t* imply (at least) medium levels for the next time-step *t+1*.

# A more refined model

- This 9-variable model is about as big of a state space that can be rendered.
- Here's a sample piece of the state space:

$$f_M = \overline{R} \wedge C$$
$$f_P = M$$
$$f_B = M$$
$$f_C = \overline{G_e}$$
$$f_R = \overline{A} \wedge \overline{A_m}$$
$$f_A = L \wedge B$$
$$f_{A_m} = A \vee L \vee L_m$$
$$f_L = \overline{G_e} \wedge P \wedge L_e$$
$$f_{L_m} = \overline{G_e} \wedge (L \vee L_e)$$



# What if the state space is too big?

- The previous 9-variable model is about as big as Cyclone can handle.
- However, many gene regulatory networks are much bigger.
  - A Boolean network model (2006) of T helper cell differentiation has 23 nodes, and thus a state space of size $2^{23} = 8{,}388{,}608$.
  - A Boolean network model (2003) of the segment polarity genes in Drosophila melanogaster (fruit fly) has 60 nodes, and a state space of size $2^{60} \approx 1.15 \times 10^{18}$.
  - There are many more examples...
- For these systems, we need to be able to analyze them without constructing the entire state space.
- Our first goal is to find the fixed points. This amounts to solving a system of equations:

$$\begin{cases} f_{x_1} = x_1 \\ f_{x_2} = x_2 \\ \vdots \\ f_{x_n} = x_n \end{cases}$$

$$f_M = \overline{R} \wedge C$$
$$f_P = M$$
$$f_B = M$$
$$f_C = \overline{G_e}$$
$$f_R = \overline{A} \wedge \overline{A_m}$$
$$f_A = L \wedge B$$
$$f_{A_m} = A \vee L \vee L_m$$
$$f_L = \overline{G_e} \wedge P \wedge L_e$$
$$f_{L_m} = \overline{G_e} \wedge (L \vee L_e)$$

# How to find the fixed points

- Let's rename variables: $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$
- Writing each function in polynomial form, and then $f_{x_i}=x_i$ for each i=1,...,9 yields the following system:

$$f_M=\overline{R}\wedge C=M$$
$$f_P=M=P$$
$$f_B=M=B$$
$$f_C=\overline{G_e}=C$$
$$f_R=\overline{A}\wedge\overline{A_m}=R$$
$$f_A=L\wedge B=A$$
$$f_{A_m}=A\vee L\vee L_m=A_m$$
$$f_L=\overline{G_e}\wedge P\wedge L_e=L$$
$$f_{L_m}=\overline{G_e}\wedge(L\vee L_e)=L_m$$

$$\begin{cases} x_1+x_4x_5+x_4=0 \\ x_1+x_2=0 \\ x_1+x_3=0 \\ x_4+(G_e+1)=0 \\ x_5+x_6x_7+x_6+x_7+1=0 \\ x_6+x_3x_8=0 \\ x_6+x_7+x_8+x_9+x_8x_9+x_6x_8+x_6x_9+x_6x_8x_9=0 \\ x_8+x_2L_e(G_e+1)=0 \\ x_9+(G_e+1)(x_8+x_8L_e+L_e)=0 \end{cases}$$

- We need to solve this for all 4 combinations: $(G_e,L_e)=(0,0),(0,1),(1,0),(1,1)$

# How to find the fixed points with Macaulay2

- Let's first consider the case when $(G_e, L_e) = (0,1)$
- We can solve the system by typing the following commands into Macaulay2 an open-source software package for computational algebraic geometry:

```
-- Define a ring of polynomials in 9 variables.

R = ZZ/2[x1,x2,x3,x4,x5,x6,x7,x8,x9];

-- Define a quotient ring, where each x_i^2 = x_i.

I = ideal(x1^2-x1, x2^2-x2, x3^2-x3, x4^2-x4, x5^2-x5, x6^2-x6, x7^2-x7, x8^2-x8, x9^2-x9);
Q = R / I;

-- Shortcut for AND and OR functions.

RingElement | RingElement :=(x,y)->x+y+x*y;
RingElement & RingElement :=(x,y)->x*y;

-- Set the parameters (constants).

Ge = 0_Q
Le = 1_Q

-- This is the 9-variable lac operon model.

f1 = (1+x5) & x4;
f2 = x1;
f3 = x1;
f4 = 1+Ge;
f5 = (1+x6) & (1+x7);
f6 = x8 & x3;
f7 = x6 | x8 | x9;
f8 = (1+Ge) & x2 & Le;
f9 = (1+Ge) & (x8 | Le);

-- Compute the ideal to find the fixed point(s).

I = ideal(f1+x1, f2+x2, f3+x3, f4+x4, f5+x5, f6+x6, f7+x7, f8+x8, f9+x9)

-- Compute a Groebner basis.

G = gens gb I
```

# What does this code mean?

The output of `G = Gens gb I;` is the following:

|x9+1, x8+1, x7+1, x6+1, x5, x4+1, x3+1, x2+1, x1+1|

This is short-hand for the following system of equations:

$$x_9 + 1 = 0, x_8 + 1 = 0, \dots, x_4 + 1 = 0, x_5 = 0, x_3 + 1 = 0, \dots, x_1 + 1 = 0$$

This simple system has the same set of solutions as the much more complicated system we started with:

$$\begin{cases} x_1+x_4x_5+x_4 = 0 \\ x_1+x_2 = 0 \\ x_1+x_3 = 0 \\ x_4+(G_e+1) = 0 \\ x_5+x_6x_7+x_6+x_7+1 = 0 \\ x_6+x_3x_8 = 0 \\ x_6+x_7+x_8+x_9+x_8x_9+x_6x_8+x_6x_9+x_6x_8x_9 = 0 \\ x_8+x_2L_e(G_e+1) = 0 \\ x_9+(G_e+1)(x_8+x_8L_e+L_e) = 0 \end{cases}$$

# What does a Gröbner basis tell us?

The output of `G = Gens gb I;` is the following:

|x9+1, x8+1, x7+1, x6+1, x5, x4+1, x3+1, x2+1, x1+1|

This is short-hand for the following system of equations:

$$x_9 + 1 = 0, x_8 + 1 = 0, \ldots, x_4 + 1 = 0, x_5 = 0, x_3 + 1 = 0, \ldots, x_1 + 1 = 0$$

This simple system has the same set of solutions as the much more complicated system we started with:

$$\begin{cases} x_1+x_4x_5+x_4 = 0 \\ x_1+x_2 = 0 \\ x_1+x_3 = 0 \\ x_4+(G_e+1) = 0 \\ x_5+x_6x_7+x_6+x_7+1 = 0 \\ x_6+x_3x_8 = 0 \\ x_6+x_7+x_8+x_9+x_8x_9+x_6x_8+x_6x_9+x_6x_8x_9 = 0 \\ x_8+x_2L_e(G_e+1) = 0 \\ x_9+(G_e+1)(x_8+x_8L_e+L_e) = 0 \end{cases}$$

# How to find the fixed points with Sage

- Let's first consider the case when $(G_e, L_e) = (1,1)$
- We can solve the system by typing the following commands into Sage the free open-source mathematical software.
- Here, we did not take the quotient ideal, but we still could have.

```
P.<x1,x2,x3,x4,x5,x6,x7,x8,x9> = PolynomialRing(GF(2), 9, order ='lex'); P
```

```
Multivariate Polynomial Ring in x1, x2, x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
```

```
Le=1;
Ge=1;
print "Le =", Le;
print "Ge =", Ge;
```

```
Le = 1
Ge = 1
```

```
I = ideal(x1+x4*x5+x4, x1+x2, x1+x3, x4+(Ge+1), x5+x6*x7+x6+x7+1, x6+x3*x8,
x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8+Le*(Ge+1)*x2, x9+(Ge+1)*(Le+x8+Le*x8)); I
```

```
Ideal (x1 + x4*x5 + x4, x1 + x2, x1 + x3, x4, x5 + x6*x7 + x6 + x7 + 1, x3*x8 + x6, x6*x8*x9 +
 x6*x8 + x6*x9 + x6 + x7 + x8*x9 + x8 + x9, x8, x9) of Multivariate Polynomial Ring in x1, x2,
 x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
```

```
B = I.groebner_basis(); B
```

```
[x1, x2, x3, x4, x5 + 1, x6, x7, x8, x9]
```

# What those Sage commands mean

Let's go over what the following commands mean:

- `P.<x1,x2,x3,x4,x5,x6,x7,x8,x9> = PolynomialRing(GF(2),9,order='lex');`
  - Define P to be the polynomial ring over 9 variables, x1,...,x9.
  - GF(2)={0,1} because the coefficients are binary.
  - order='lex' specifies a monomial order. More on this later.
- `Le=1; Ge=1; print "Le =", Le; print "Ge =", Ge;`
  - This defines two constants $(G_e, L_e) = (1,1)$ and prints them.
- `I = ideal(x1+x4*x5+x4, x1+x2, x1+x3, x4+(Ge+1), x5+x6*x7+x6+x7+1, x6+x3*x8, x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8+Le*(Ge+1)*x2, x9+(Ge+1)*(Le+x8+Le*x8)); I`
  - Defines I to be the ideal generated by those following 9 polynomials, i.e.,

$$I = \{p_1 f_1 + \cdots + p_k f_k : p_k \in P\}$$

- `B = I.groebner_basis(); B`
  - Define B to be the Gröbner basis of I w.r.t. the lex monomial order. (More on this later)

# Gröbner bases vs. Gaussian elimination

- Gröbner bases are a generalization of Gaussian elimination, but for systems of polynomials (instead of systems of linear equations)
- In both cases:
  - The input is a complicated system that we wish to solve.
  - The output is a simple system that we can easily solve by inspection.
- Consider the following example:
  - Input: The 2x2 system of linear equations

$$\begin{cases} x+2y=1 \\ 3x+8y=1 \end{cases}$$

  - Gaussian elimination yields the following:

$$\left[\begin{array}{cc|c} 1 & 2 & 1 \\ 3 & 8 & 1 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 2 & 1 \\ 0 & 2 & -2 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 0 & 3 \\ 0 & 2 & -2 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 0 & 3 \\ 0 & 1 & -1 \end{array}\right]$$

  - This is just the much simpler system with the same solution!

$$\begin{cases} x+0y=3 \\ 0x+y=-1 \end{cases}$$

# Back-substitution & Gaussian elimination

- We don't necessarily need to do Gaussian elimination until the matrix is the identity. As long as it is upper-triangular, we can back-substitute and solve by hand.
- For example:

$$\begin{cases} x + z = 2 \\ y - z = 8 \\ 0 = 0 \end{cases}$$

- Similarly, when Sage outputs a Gröbner basis, it will be in "upper-triangular form", and we can solve the system easily by back-substituting.
- We'll do an example right away. For this part of the class, you can think of Gröbner bases as a mysterious "black box" that does what we want.
- We'll study them in more detail shortly, and understand what's going on behind the scenes.

# Gröbner bases: an example

- Let's use Sage to solve the following system:

$$\begin{cases} x^2 + y^2 + z^2 = 1 \\ x^2 - y + z^2 = 0 \\ x - z = 0 \end{cases}$$

```
17  P.<x,y,z>=PolynomialRing(RR,3,order='lex'); P
18      Multivariate Polynomial Ring in x, y, z over Real Field with 53 bits of precision
19
20  I = ideal(x^2+y^2+z^2-1, x^2-y+z^2, x-z); I
21      Ideal (x^2 + y^2 + z^2 - 1.00000000000000, x^2 - y + z^2, x - z) of Multivariate Polynomial
        Ring in x, y, z over Real Field with 53 bits of precision
22
23  B = I.groebner_basis(); B
24      [x - z, y - 2.00000000000000*z^2, z^4 + 0.500000000000000*z^2 - 0.250000000000000]
```

- From this, we get an "upper-triangular" system:
- This is something we can solve by hand.

$$\begin{cases} x - z = 0 \\ y - 2z^2 = 0 \\ z^4 + .5z^2 - .25 = 0 \end{cases}$$

# Gröbner bases: an example (cont.)

- To solve the reduced system:

$$\begin{cases} x - z = 0 \\ y - 2z^2 = 0 \\ z^4 + .5z^2 - .25 = 0 \end{cases}$$

  - Solve for z in Eq. 3: $z = \pm\sqrt{\frac{-1+\sqrt{5}}{4}}$
  - Plug z into Eq. 2 and solve for y: $y = 2z^2 = \frac{-1+\sqrt{5}}{2}$
  - Plug y & z into Eq. 1 and solve for x: $x = z = \pm\sqrt{\frac{-1+\sqrt{5}}{4}}$

$$\begin{cases} x^2 + y^2 + z^2 = 1 \\ x^2 - y + z^2 = 0 \\ x - z = 0 \end{cases}$$

- Thus, we get 2 solutions to the original system:

$$(x_1, y_1, z_1) = \left(\sqrt{\frac{-1+\sqrt{5}}{4}}, \frac{-1+\sqrt{5}}{2}, \sqrt{\frac{-1+\sqrt{5}}{4}}\right) \qquad (x_2, y_2, z_2) = \left(-\sqrt{\frac{-1+\sqrt{5}}{4}}, \frac{-1+\sqrt{5}}{2}, -\sqrt{\frac{-1+\sqrt{5}}{4}}\right)$$

# Returning to the *lac* operon

- We have 9 variables: $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$
- Writing each function in polynomial form, we need to solve the system $f_{x_i}=x_i$ for each i=1,...,9, which is the following:

$$
\begin{aligned}
f_M &= \overline{R}\wedge C = M \\
f_P &= M = P \\
f_B &= M = B \\
f_C &= \overline{G_e} = C \\
f_R &= \overline{A}\wedge\overline{A_m} = R \\
f_A &= L\wedge B = A \\
f_{A_m} &= A\vee L\vee L_m = A_m \\
f_L &= \overline{G_e}\wedge P\wedge L_e = A_m \\
f_{L_m} &= \overline{G_e}\wedge(L\vee L_e) = L_m
\end{aligned}
\qquad
\begin{cases}
x_1+x_4x_5+x_4=0 \\
x_1+x_2=0 \\
x_1+x_3=0 \\
x_4+(G_e+1)=0 \\
x_5+x_6x_7+x_6+x_7+1=0 \\
x_6+x_3x_8=0 \\
x_6+x_7+x_8+x_9+x_8x_9+x_6x_8+x_6x_9+x_6x_8x_9=0 \\
x_8+x_2L_e(G_e+1)=0 \\
x_9+(G_e+1)(x_8+x_8L_e+L_e)=0
\end{cases}
$$

- We need to solve this for all 4 combinations: $(G_e,L_e)=(0,0),(0,1),(1,0),(1,1)$ (we already did (1,1)).

# Returning to the *lac* operon

- Again, we use variables $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$

  and parameters $(G_e,L_e)=(0,0)$

- Here is the output from Sage:

```
P.<x1,x2,x3,x4,x5,x6,x7,x8,x9> = PolynomialRing(GF(2), 9, order ='lex'); P
```

```
Multivariate Polynomial Ring in x1, x2, x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
```

```
Le=0;
Ge=0;
print "Le =", Le;
print "Ge =", Ge;
```

```
Le = 0
Ge = 0
```

```
I = ideal(x1+x4*x5+x4, x1+x2, x1+x3, x4+(Ge+1), x5+x6*x7+x6+x7+1, x6+x3*x8,
x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8+Le*(Ge+1)*x2, x9+(Ge+1)*(Le+x8+Le*x8)); I
```

```
Ideal (x1 + x4*x5 + x4, x1 + x2, x1 + x3, x4 + 1, x5 + x6*x7 + x6 + x7 + 1, x3*x8 + x6, x6*x8*x9 +
 x6*x8 + x6*x9 + x6 + x7 + x8*x9 + x8 + x9, x8, x8 + x9) of Multivariate Polynomial Ring in x1, x2
, x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
```

```
B = I.groebner_basis(); B
```

```
[x1, x2, x3, x4 + 1, x5 + 1, x6, x7, x8, x9]
```

- $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,1,1,0,0,0,0)$

# Returning to the *lac* operon

- Again, we use variables $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$

  and parameters $(G_e,L_e)=(1,0)$

- Here is the output from Sage:

```
1
2  P.<x1,x2,x3,x4,x5,x6,x7,x8,x9> = PolynomialRing(GF(2), 9, order ='lex'); P
3      Multivariate Polynomial Ring in x1, x2, x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
4
5  Le=0;
6  Ge=1;
7  print "Le =", Le;
8  print "Ge =", Ge;
9      Le = 0
       Ge = 1
10
11 I = ideal(x1+x4*x5+x4, x1+x2, x1+x3, x4+(Ge+1), x5+x6*x7+x6+x7+1, x6+x3*x8,
   x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8+Le*(Ge+1)*x2, x9+(Ge+1)*(Le+x8+Le*x8)); I
12     Ideal (x1 + x4*x5 + x4, x1 + x2, x1 + x3, x4, x5 + x6*x7 + x6 + x7 + 1, x3*x8 + x6, x6*x8*x9 +
        x6*x8 + x6*x9 + x6 + x7 + x8*x9 + x8 + x9, x8, x9) of Multivariate Polynomial Ring in x1, x2,
        x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
13
14 B = I.groebner_basis(); B
15     [x1, x2, x3, x4, x5 + 1, x6, x7, x8, x9]
```

- $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,0,1,0,0,0,0)$

# Returning to the *lac* operon

- Again, we use variables $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$

  and parameters $(G_e,L_e)=(0,1)$

- Here is the output from Sage:

```
1
2  P.<x1,x2,x3,x4,x5,x6,x7,x8,x9> = PolynomialRing(GF(2), 9, order ='lex'); P
3     Multivariate Polynomial Ring in x1, x2, x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
4
5  Le=0;
6  Ge=1;
7  print "Le =", Le;
8  print "Ge =", Ge;
9     Le = 0
      Ge = 1
10
11 I = ideal(x1+x4*x5+x4, x1+x2, x1+x3, x4+(Ge+1), x5+x6*x7+x6+x7+1, x6+x3*x8,
   x6+x7+x8+x9+x8*x9+x6*x8+x6*x9+x6*x8*x9, x8+Le*(Ge+1)*x2, x9+(Ge+1)*(Le+x8+Le*x8)); I
12    Ideal (x1 + x4*x5 + x4, x1 + x2, x1 + x3, x4, x5 + x6*x7 + x6 + x7 + 1, x3*x8 + x6, x6*x8*x9 +
       x6*x8 + x6*x9 + x6 + x7 + x8*x9 + x8 + x9, x8, x9) of Multivariate Polynomial Ring in x1, x2,
       x3, x4, x5, x6, x7, x8, x9 over Finite Field of size 2
13
14 B = I.groebner_basis(); B
15    [x1, x2, x3, x4, x5 + 1, x6, x7, x8, x9]
```

- $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(1,1,1,1,0,1,1,1,1)$

# Fixed point analysis of the *lac* operon

Using the variables $(M,P,B,C,R,A,A_m,L,L_m)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$

we got the following fixed points for each choice of parameters $(G_e,L_e)$

- Input: $(G_e,L_e)=(0,0)$

  Fixed point: $(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,1,1,0,0,0,0)$

- Input: $(G_e,L_e)=(1,0)$

  Fixed point: $(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,0,1,0,0,0,0)$

- Input: $(G_e,L_e)=(1,1)$

  Fixed point: $(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,0,1,0,0,0,0)$

- Input: $(G_e,L_e)=(0,1)$

  Fixed point: $(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(1,1,1,1,0,1,1,1,1)$

All of these fixed points make biological sense!