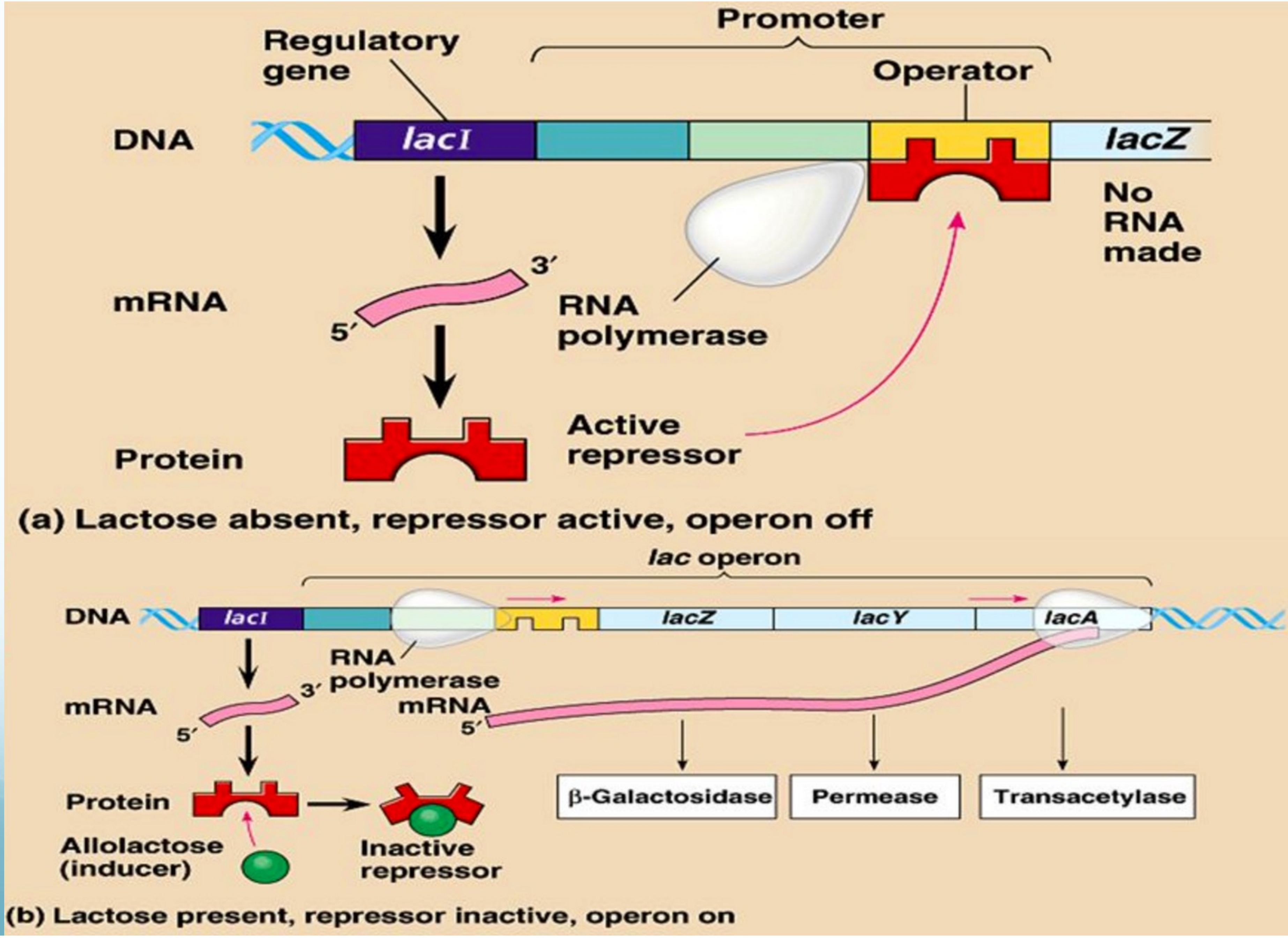
Advanced Boolean models of the *lac* operon

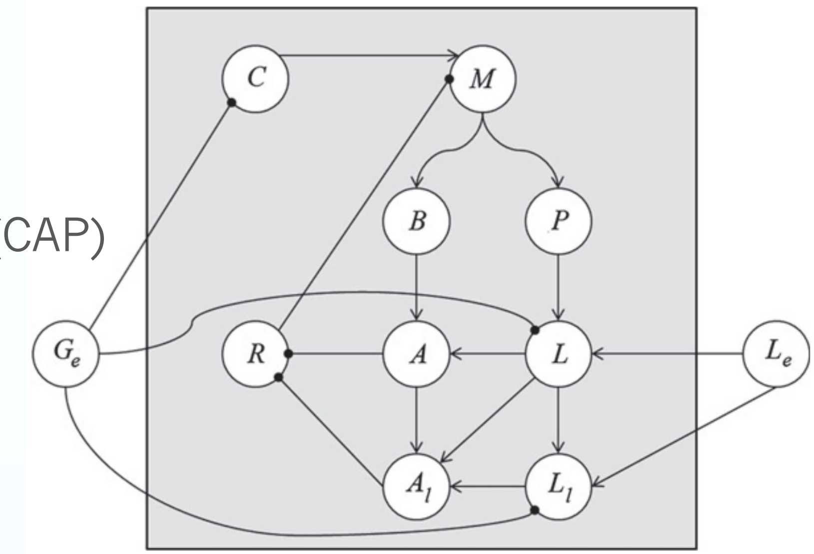
Matthew Macauley
Clemson University

The *lac* operon



A 9-variable 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 9-variable 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:

$$f_M = \overline{R} \wedge C$$

$$f_P = M$$

$$f_B = M$$

$$f_C = \overline{G_e}$$

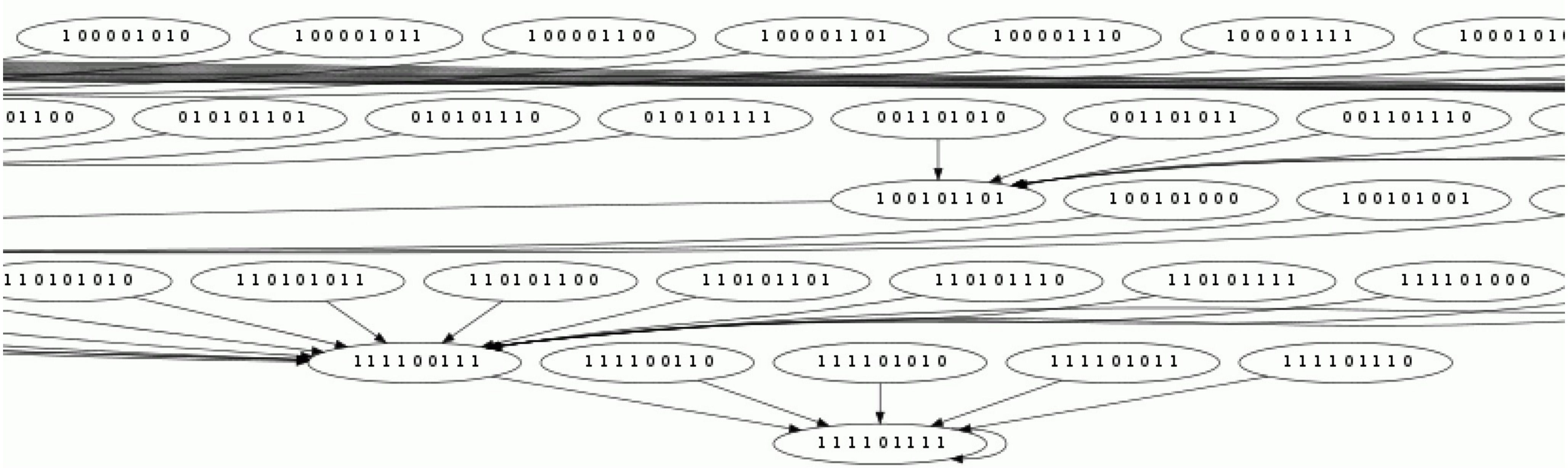
$$f_R = \overline{A} \wedge \overline{A_m}$$

$$f_A = L \wedge B$$

$$f_{A_m} = A \vee L \vee L_m$$

$$f_L = \overline{G_e} \wedge P \wedge L_e$$

$$f_{L_m} = \overline{G_e} \wedge (L \vee L_e)$$



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:

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

- 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:

$$\begin{cases} x_1 + x_4 x_5 + x_4 = 0 \\ x_1 + x_2 = 0 \\ x_1 + x_3 = 0 \\ x_4 + (G_e + 1) = 0 \\ x_5 + x_6 x_7 + x_6 + x_7 + 1 = 0 \\ x_6 + x_3 x_8 = 0 \\ x_6 + x_7 + x_8 + x_9 + x_8 x_9 + x_6 x_8 + x_6 x_9 + x_6 x_8 x_9 = 0 \\ x_8 + x_2 L_e (G_e + 1) = 0 \\ x_9 + (G_e + 1)(x_8 + x_8 L_e + L_e) = 0 \end{cases}$$

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:

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

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

$$\begin{cases} x_1 + x_4 x_5 + x_4 = 0 \\ x_1 + x_2 = 0 \\ x_1 + x_3 = 0 \\ x_4 + (G_e + 1) = 0 \\ x_5 + x_6 x_7 + x_6 + x_7 + 1 = 0 \\ x_6 + x_3 x_8 = 0 \\ x_6 + x_7 + x_8 + x_9 + x_8 x_9 + x_6 x_8 + x_6 x_9 + x_6 x_8 x_9 = 0 \\ x_8 + x_2 L_e (G_e + 1) = 0 \\ x_9 + (G_e + 1)(x_8 + x_8 L_e + L_e) = 0 \end{cases}$$

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:

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

  - 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:

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

- 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:

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

```
P.<x,y,z>=PolynomialRing(RR,3,order='lex'); P
    Multivariate Polynomial Ring in x, y, z over Real Field with 53 bits of precision

I = ideal(x^2+y^2+z^2-1, x^2-y+z^2, x-z); I
    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

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

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

- To solve the reduced system:
  - Solve for z in Eq. 3: $z = \pm\sqrt{\dfrac{-1+\sqrt{5}}{4}}$
  - Plug z into Eq. 2 and solve for y: $y = 2z^2 = \dfrac{-1+\sqrt{5}}{2}$
  - Plug y & z into Eq. 1 and solve for x: $x = z = \pm\sqrt{\dfrac{-1+\sqrt{5}}{4}}$
- Thus, we get 2 solutions to the original system:

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

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

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:

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

- 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:

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

```
Le = 0
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]
```

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

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:

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

```
Le = 0
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]
```

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

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!