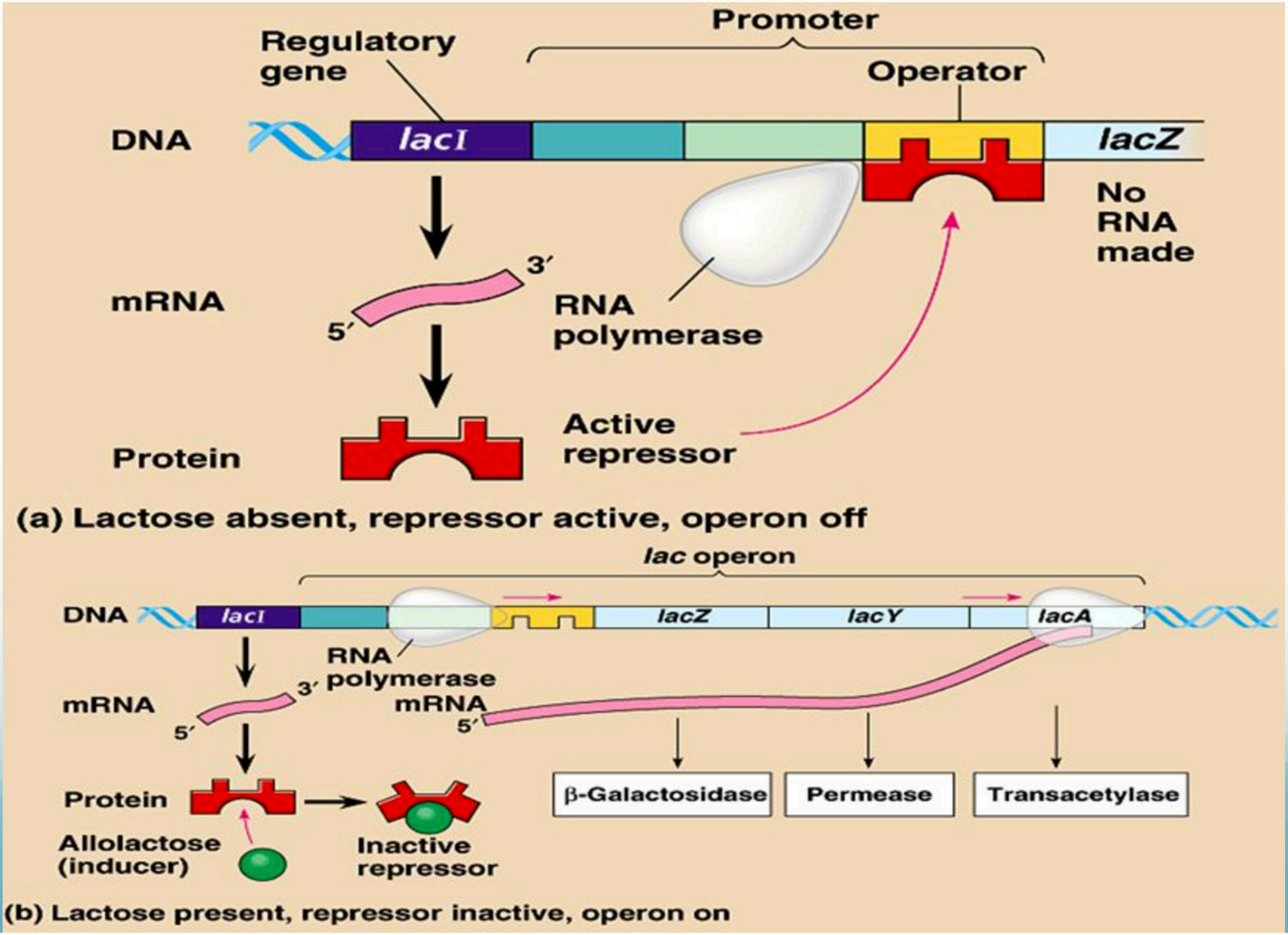
The *lac* operon in *E. coli*

Matthew Macauley
Math 4500: Mathematical Modeling
Clemson University
Spring 2016

The *lac* operon

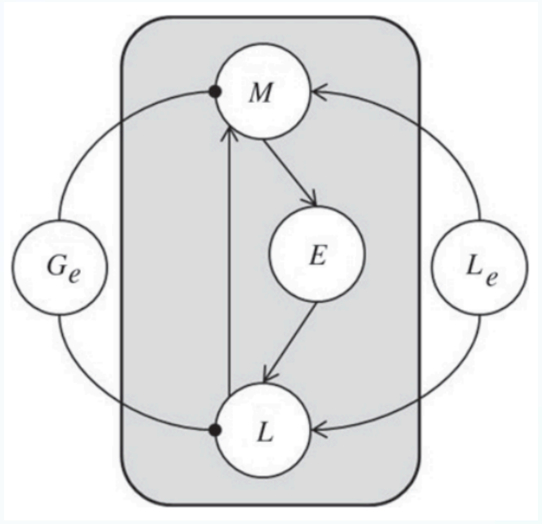


lac operon, with lactose present

- Lactose is brought into the cell by the *lac* permease transporter protein
- $\beta-$galactosidase breaks up lactose into glucose and galactose..
- $\beta-$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 continues: 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.)

Our first simple 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 drew the state space for all four choices of the parameters:
  - (L_e, G_e) = (0, 0). Every state ended up in the "OFF" fixed point (0,0,0).
  - (L_e, G_e) = (0, 1). Every state ended up in the "OFF" fixed point (0,0,0).
  - (L_e, G_e) = (1, 0). Every state ended up in the "ON" fixed point (1,1,1).
  - (L_e, G_e) = (1, 1). Every state ended up in the "OFF" fixed point (0,0,0).

A more refined model

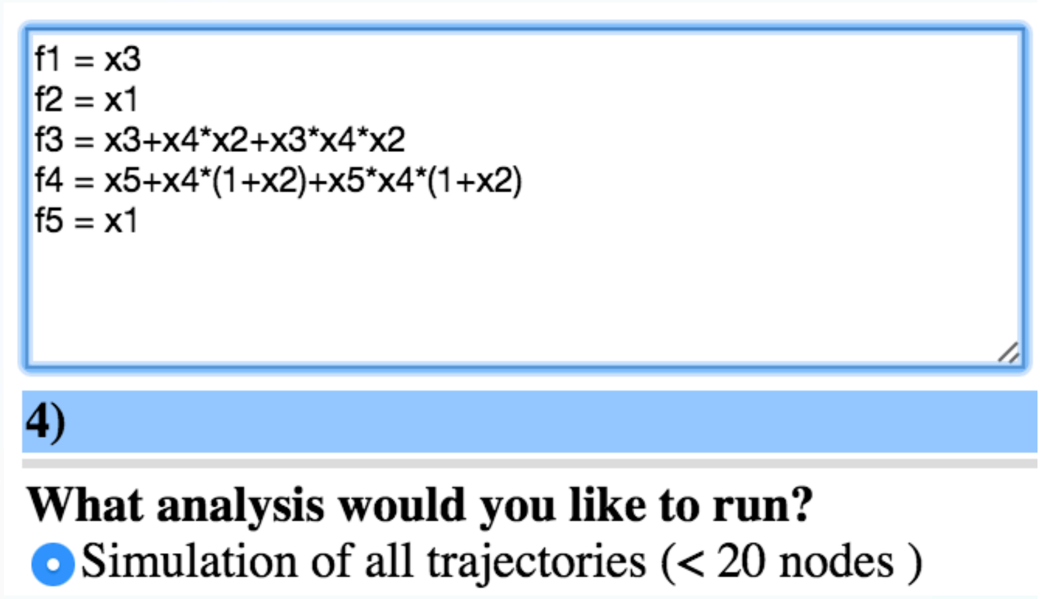
- Our model only used 3 variables: mRNA (M), enzyme (E), and lactose (L).
- Let's propose a new model with 5 variables:
  - M: mRNA
  - B: β–galactosidase
  - A: allolactose
  - L: intracellular lactose
  - P: *lac* permease (transporter protein)

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

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

Using ADAM to compute the state space

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

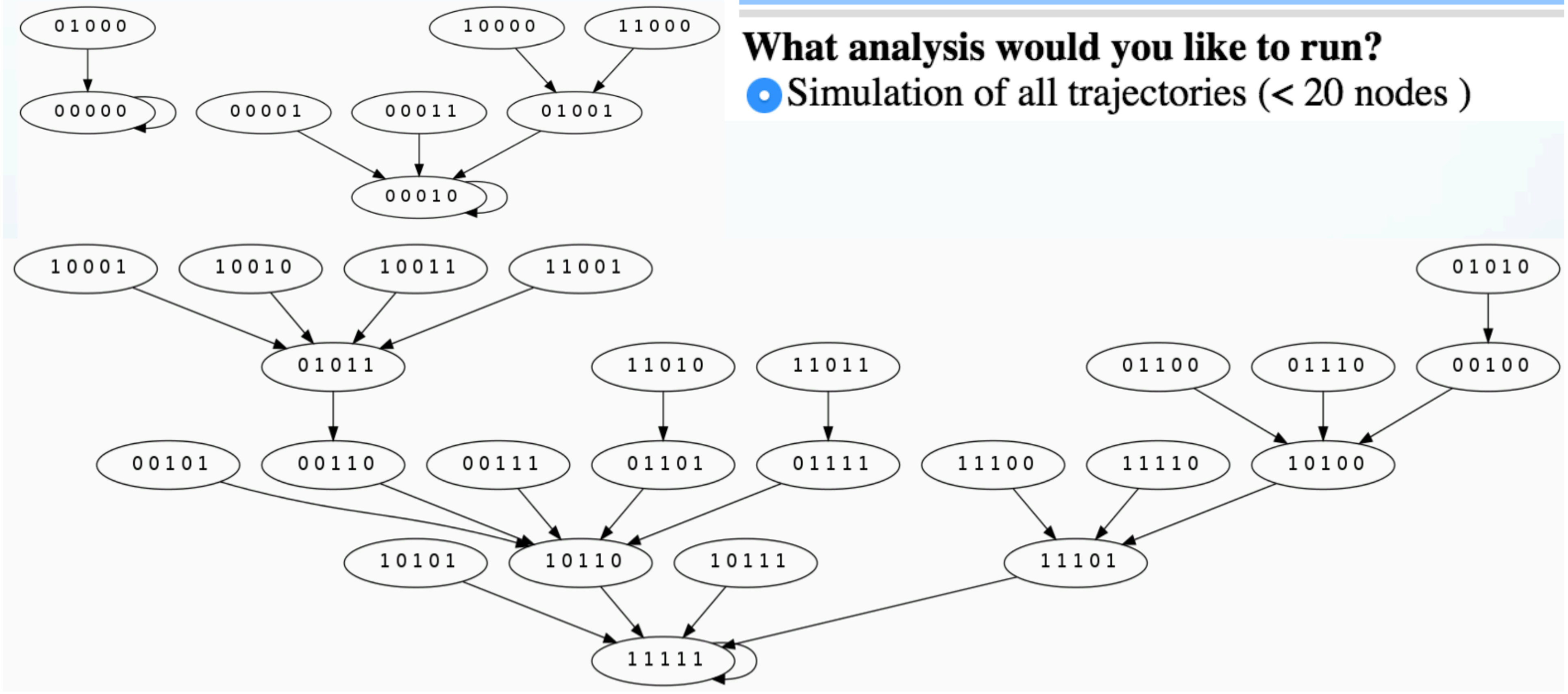


```
f1 = x3
f2 = x1
f3 = x3+x4*x2+x3*x4*x2
f4 = x5+x4*(1+x2)+x5*x4*(1+x2)
f5 = x1
```

4)

What analysis would you like to run?

[✓ Simulation of all trajectories (< 20 nodes)]



Problems with our refined model

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

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

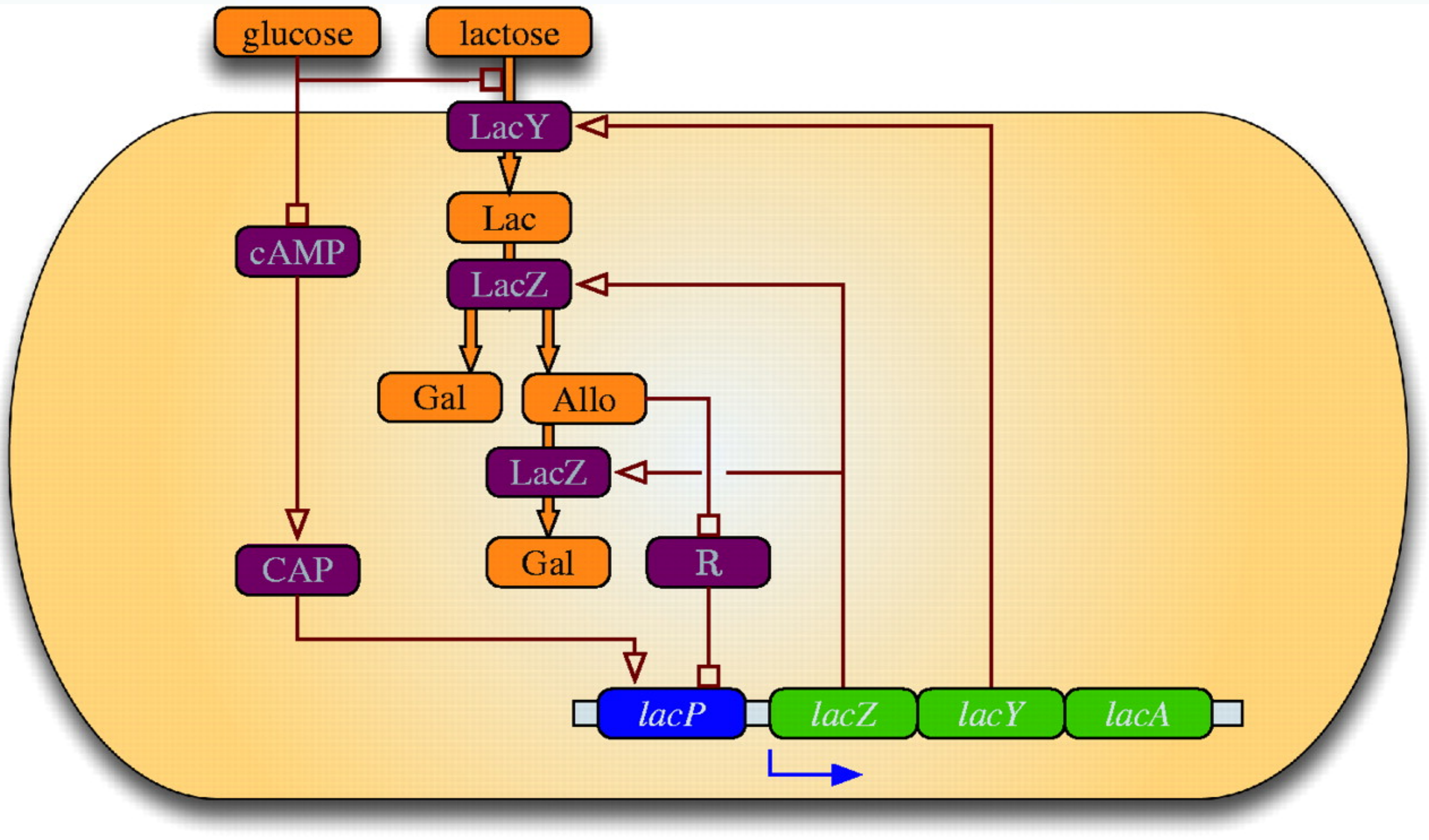
- Problems:
  - The fixed point (M,B,A,L,P) = (0,0,0,0,0) should not happen with lactose present but not glucose. [though let's try to justify this…]
  - The fixed point point (M,B,A,L,P) = (0,0,0,1,0) is not biologically feasible: it would describe a scenario where the bacterium does not metabolize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

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

Catabolite repression

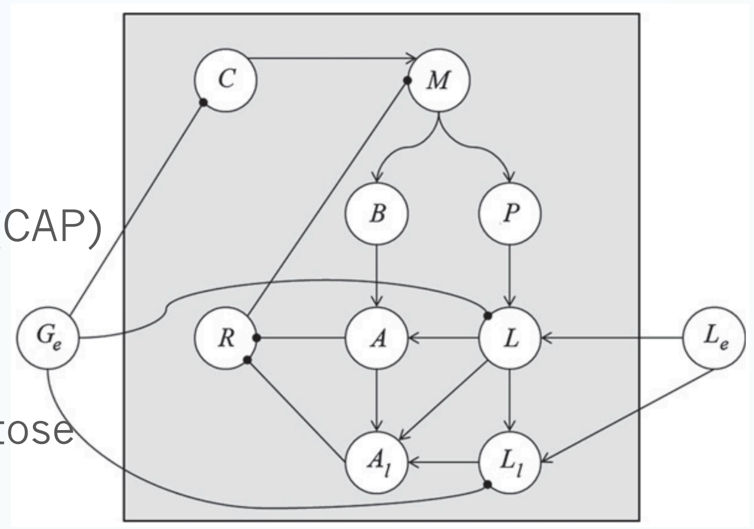
- We haven't yet discussed the cellular mechanism that turns the *lac* operon OFF when both glucose and lactose are present. This is done by catabolite repression.
- The *lac* operon promoter region has 2 binding sites:
  - One for RNA polymerase (this "unzips" and reads the DNA)
  - One for the CAP-cAMP 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: allolactose
  - A_l: at least low levels of allolactose
  - L: intracellular lactose
  - L_l: at least low levels of intracellular lactose



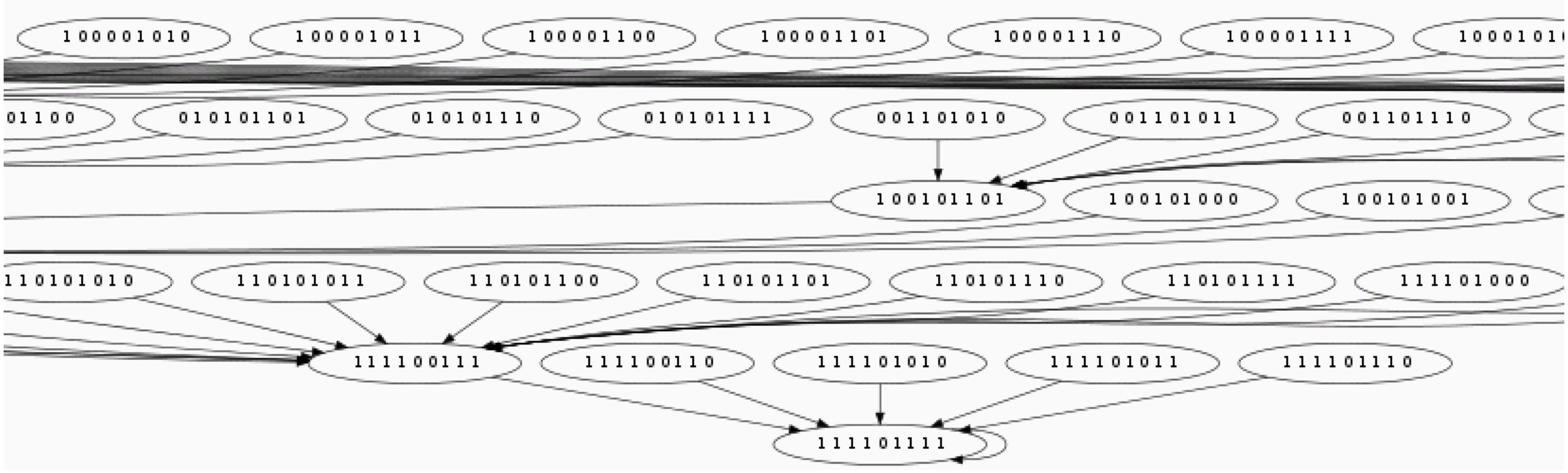
$$f_M = \overline{R} \wedge C$$
$$f_P = M$$
$$f_B = M$$
$$f_C = \overline{G_e}$$
$$f_R = \overline{A} \wedge \overline{A_l}$$
$$f_A = L \wedge B$$
$$f_{A_l} = A \vee L \vee L_l$$
$$f_L = \overline{G_e} \wedge P \wedge L_e$$
$$f_{L_l} = \overline{G_e} \wedge (L \vee L_e)$$

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

A more refined model

- This 9-variable model is about as big as ADAM can render a state space.
- In fact, it doesn't work using the "Open Polynomial Dynamical System (oPDS)" option (variables + parameters).
- Instead, it works under "Polynomial Dynamical System (PDS)", if we manually enter numbers for the parameters.
- Here's a sample piece of the state space:

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



What if the state space is too big?

- The previous 9-variable model is about as big as ADAM 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 systems like these, we would like to be able to analyze them without actually constructing the entire state space.
- One of the first goals is how to find the fixed points. This amounts to solving a system of equations:

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

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

How to find the fixed points

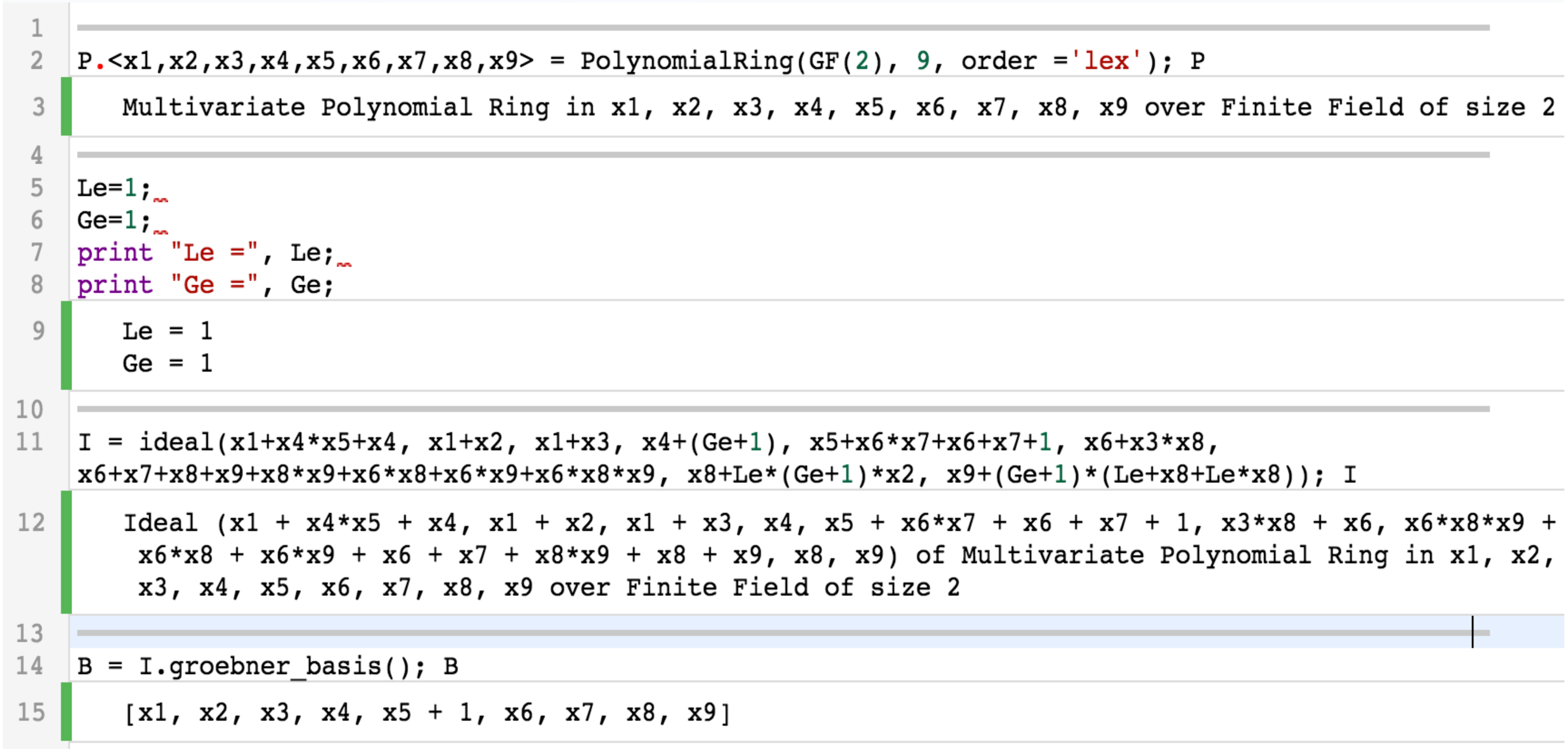
- Let's rename variables: $(M,P,B,C,R,A,A_l,L,L_l)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$
- Writing each function in polynomial form, and then $f_{x_i}=x_i$ for each i=1,...,9 yields the following system:

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

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

How to find the fixed points

- 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 (https://cloud.sagemath.com/), the free open-source mathematical software:



```
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}, and so 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)

What does a Gröbner basis tell us?

The output of `B = I.groebner_basis(); B` is the following:

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

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

$$\{x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ x_4 = 0,\ x_5 + 1 = 0,\ x_6 = 0,\ x_7 = 0,\ x_8 = 0,\ x_9 = 0\}$$

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:

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

  - 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

- Note that 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:

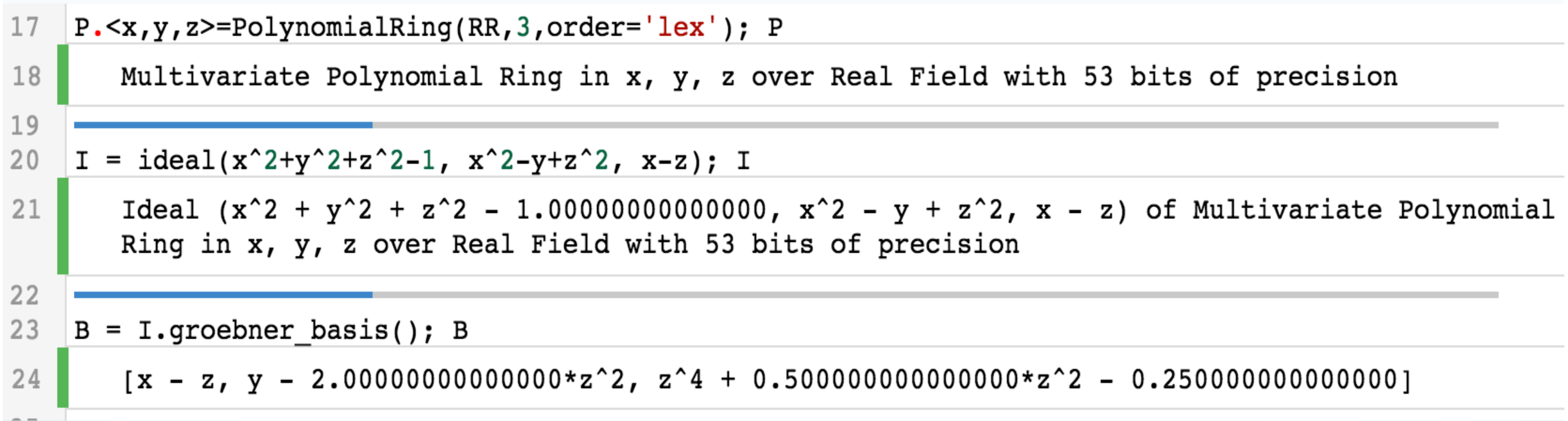
$$\begin{cases} x + \phantom{y - {}} z = 2 \\ \phantom{x + {}} y - z = 8 \\ \phantom{x + y - {}} 0 = 0 \end{cases}$$

- 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: $z = \pm\sqrt{\frac{-1+\sqrt{5}}{4}}$
  - 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: $z = \pm\sqrt{\frac{-1+\sqrt{5}}{4}}$
- 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: $(M,P,B,C,R,A,A_l,L,L_l)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$
- 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:

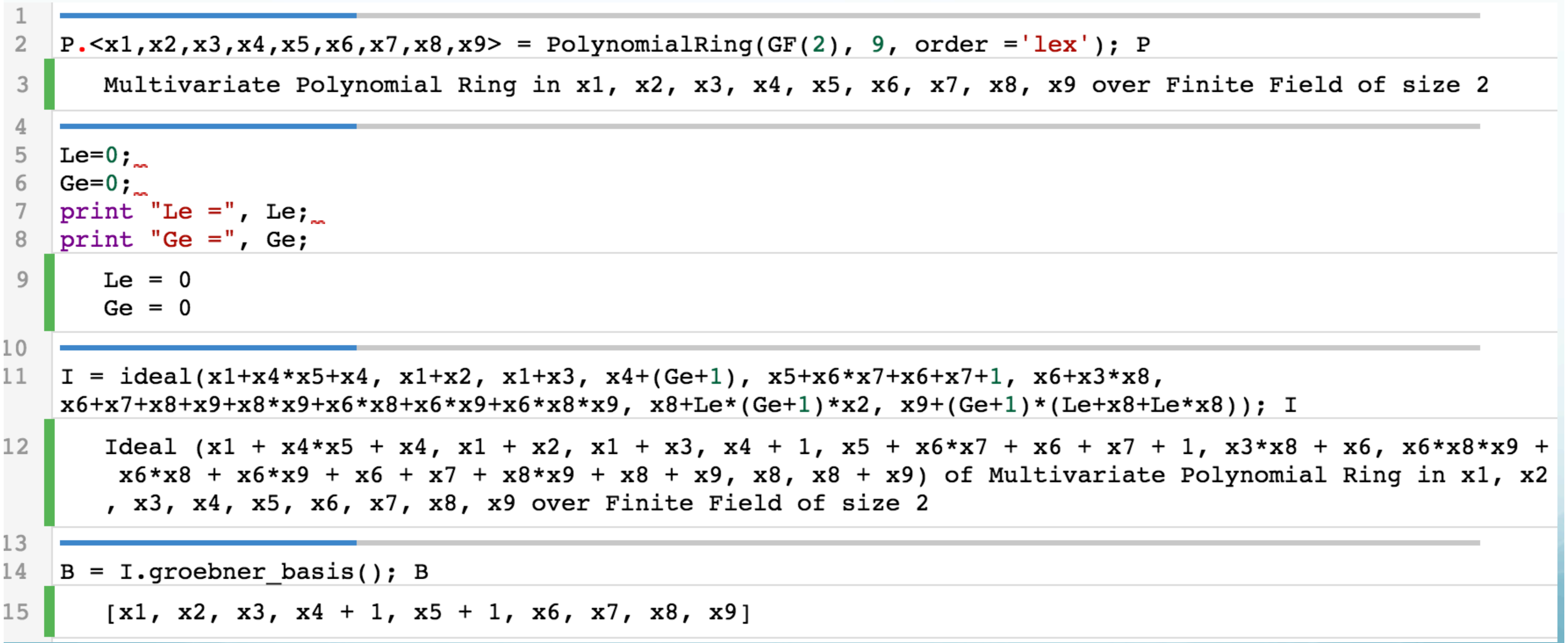
$$
\begin{aligned}
f_M &= \overline{R}\wedge C = M\\
f_P &= M = P\\
f_B &= M = B\\
f_C &= \overline{G_e} = C\\
f_R &= \overline{A}\wedge\overline{A_l} = R\\
f_A &= L\wedge B = A\\
f_{A_l} &= A\vee L\vee L_l = A_l\\
f_L &= \overline{G_e}\wedge P\wedge L_e = A_l\\
f_{L_l} &= \overline{G_e}\wedge(L\vee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

- 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 $(M,P,B,C,R,A,A_l,L,L_l)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)$

  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]
```

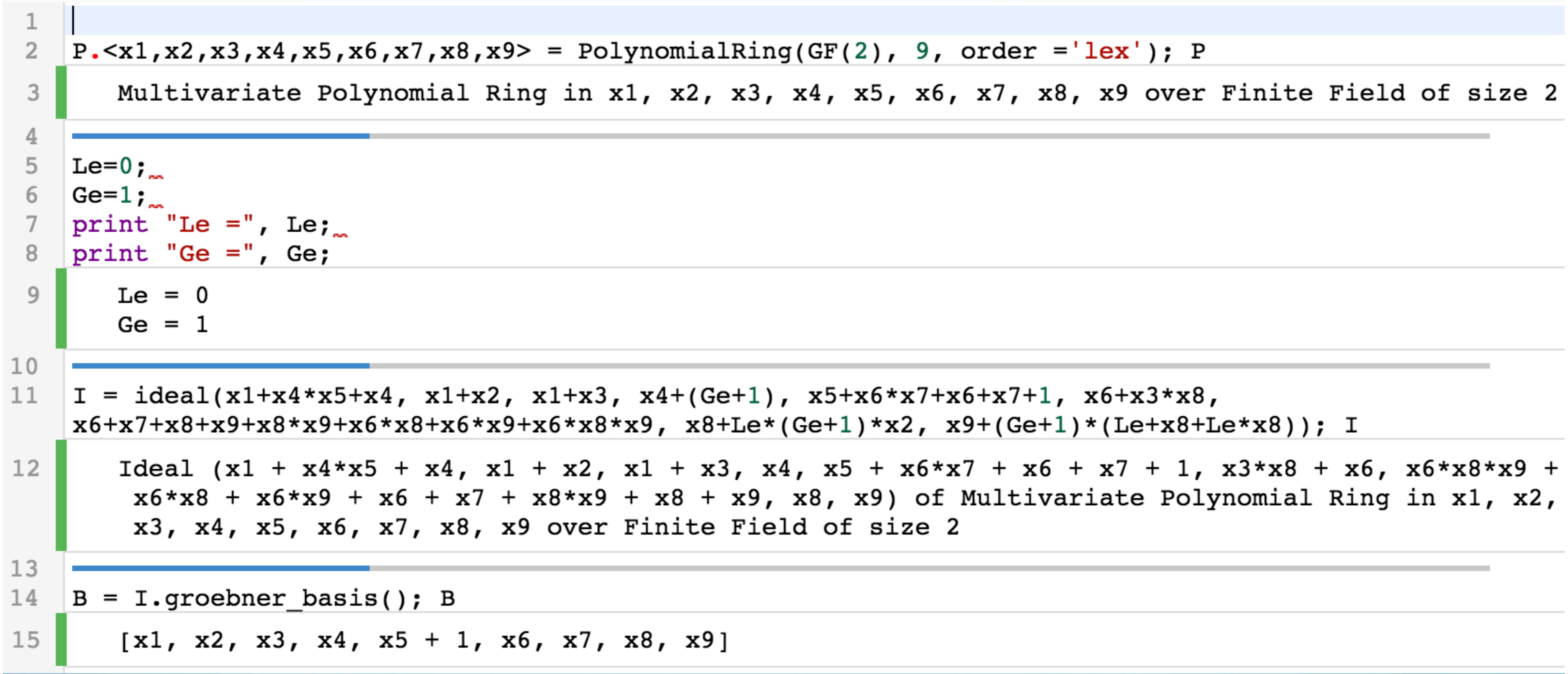
$$(M,P,B,C,R,A,A_l,L,L_l)=(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9)=(0,0,0,1,1,0,0,0,0)$$

Returning to the *lac* operon

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

  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]
```

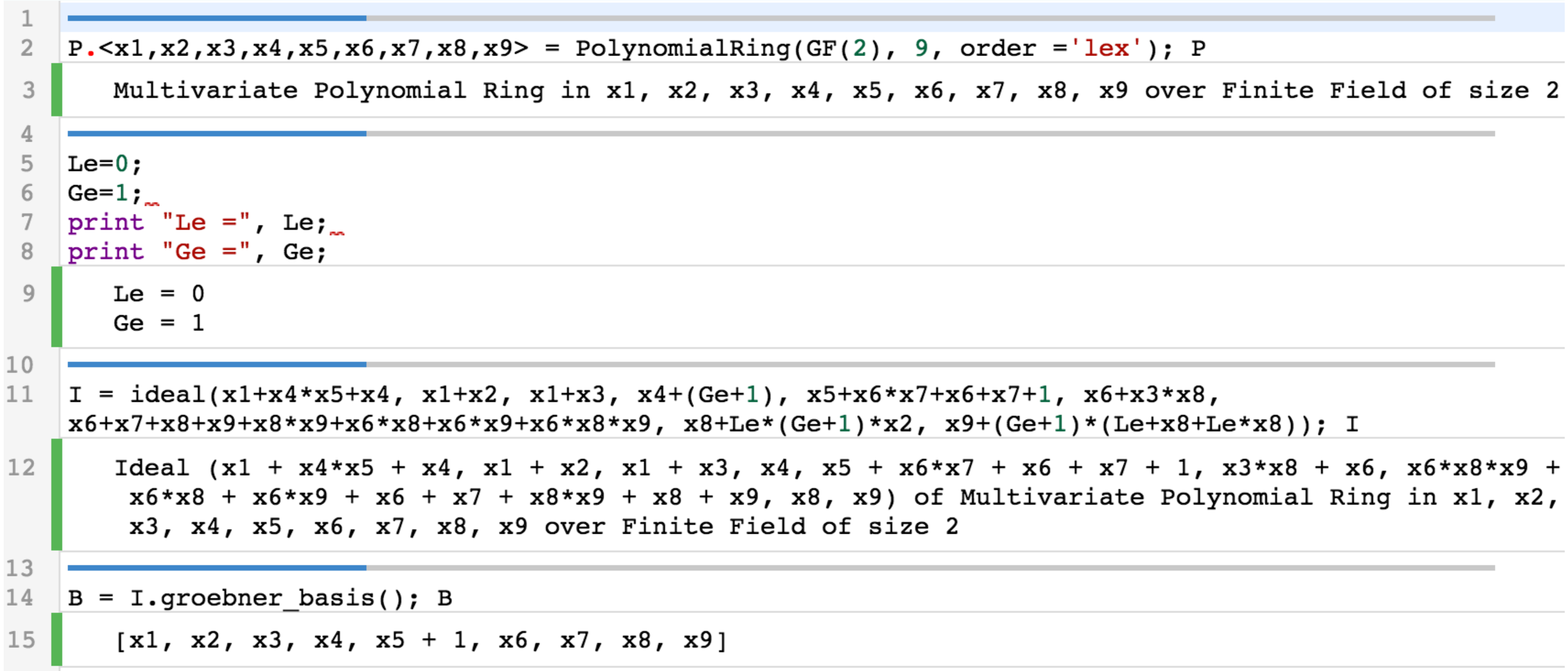
$$(M, P, B, C, R, A, A_l, L, L_l) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = (0,0,0,0,1,0,0,0,0)$$

Returning to the *lac* operon

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

  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]
```

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

Fixed point analysis of the *lac* operon

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

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!