Reduction of Boolean network models

Matthew Macauley

Department of Mathematical Sciences
Clemson University
http://www.math.clemson.edu/~macaule/

Math 4500, Spring 2017

Motivation

In the previous lecture, we modeled time-delays and dilution & degradation by adding a number of Booleans variables.

This can causes the state space to grow enormously, though in many cases, this shouldn't affect the qualitative nature of the dynamics.

In other cases, certain Boolean network models are huge and too big for direct analysis.

In this lecture, we'll see how large Boolean networks can be "reduced" to much smaller models in a way that preserves the key feature such as fixed points.

Wiring diagrams

Definition

A Boolean network (BN) in the Boolean variables $x_1, \dots, x_n$ is a function

$$f = (f_1, \dots, f_n) \colon \{0,1\}^n \longrightarrow \{0,1\}^n$$

where each $f_i \colon \{0,1\}^n \to \{0,1\}$ is called a coordinate or local function.

Definition

The wiring diagram of a Boolean network is a directed graph G on with vertex set $x_1, \dots, x_n$ (or just $1, \dots, n$) and a directed edge (x_i, x_j) if f_j depends on x_i.

An edge $x_i \longrightarrow x_j$ is positive if

$$f_j(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \leqslant f_j(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

and negative if the inequality is reversed.

Negative edges are denoted with circles or blunt arrows instead of traditional arrowheads.

Definition

A Boolean function f_i is unate (or monotone) if every edge in the wiring diagram is either positive or negative.

Wiring diagrams

- A positive edge $x_i \longrightarrow x_j$ represents a situation where i activates j.

  Examples.

  - $f_j = x_i \wedge y$: $\quad 0 = f_j(x_i = 0, y) \leqslant f_j(x_i = 1, y) \leqslant 1.$
  - $f_j = x_i \vee y$: $\quad 0 \leqslant f_j(x_i = 0, y) \leqslant f_j(x_i = 1, y) = 1.$

- A negative edge $x_i \relbar\joinrel\relbar\joinrel\rightfootline x_j$ represents a situation where i inhibits j.

  Examples.

  - $f_j = \overline{x_i} \wedge y$: $\quad 1 \geqslant f_j(x_i = 0, y) \geqslant f_j(x_i = 1, y) = 0.$
  - $f_j = \overline{x_i} \vee y$: $\quad 1 = f_j(x_i = 0, y) \geqslant f_j(x_i = 1, y) \geqslant 0.$

- Occasionally, edges are neither positive nor negative:

  Example. (The logical "XOR" function):

  - $f_j = (x_i \wedge \overline{y}) \vee (\overline{x_i} \wedge y)$: $\quad \begin{array}{l} 0 = f_j(x_1 = 0, y = 0) < f_j(x_1 = 1, y = 0) = 1 \\ 1 = f_j(x_1 = 0, y = 1) > f_j(x_1 = 1, y = 1) = 0 \end{array}$

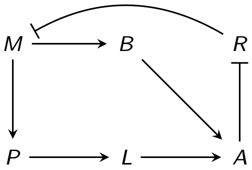
Most edges in Boolean networks arising from models are either positive or negative because most biological interactions are either simple activations or inhibitions.

A motivating example

Toy model of the *lac* operon

$f_M = \overline{R}$	R represses mRNA production
$f_P = M$	P is produced by translation of mRNA
$f_B = M$	B is produced by translation of mRNA
$f_R = \overline{A}$	A inactivates the repressor protein
$f_A = L \wedge B$	A is produced by lactose and β-galactosidase
$f_L = P$	*Lac* permease transports lactose into the cell

Here is the wiring diagram:



We won't show the state space because it's large (64 nodes), but it has two fixed points, both of which are biologically reasonable:

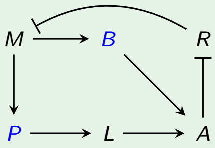
$$(M, P, B, R, A, L) = (0, 0, 0, 1, 0, 0) \qquad \text{and} \qquad (1, 1, 1, 0, 1, 1)\,.$$

Our goal is to "reduce" this model in a way that in some senes, *preserves the fixed points*.

A motivating example (cont.)

Toy model of the *lac* operon

$$f_M = \overline{R}$$
$$f_P = M$$
$$f_B = M$$
$$f_R = \overline{A}$$
$$f_A = L \wedge B$$
$$f_L = P$$

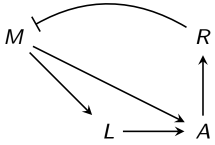


Consider the variable P. At equilibrium, $P(t) = P(t+1) = f_P(x(t)) = M(t)$.

Similarly, we can conclude that $B(t) = B(t+1) = f_B(x(t)) = M(t)$.

Thus, we can replace every instance of P and B with M:

$$f_M = \overline{R}$$
$$\cancel{f_P = M}$$
$$\cancel{f_B = M}$$
$$f_R = \overline{A}$$
$$f_A = L \wedge M$$
$$f_L = M$$



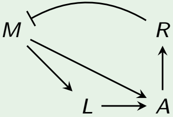
There are two steady-states of this reduced network: $(M, R, A, L) = (0, 1, 0, 0)$, $(1, 0, 1, 1)$.

Moreover, since $B = M$, $P = M$, we can recover the steady-states of the original network.

A motivating example (cont.)

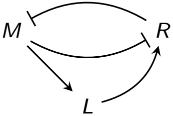
Partially reduced model of the *lac* operon

$$\begin{aligned} f_M &= \overline{R} \\ f_R &= \overline{A} \\ f_A &= L \wedge M \\ f_L &= M \end{aligned}$$



We can reduce further. At equilibrium, $A = f_A = L \wedge M$, so we can replace every instance of A with $L \wedge M$:

$$\begin{aligned} f_M &= \overline{R} \\ f_R &= \overline{L \wedge M} = \overline{L} \vee \overline{M} \\ &\cancel{f_A = L \wedge M} \\ f_L &= M \end{aligned}$$



There are two fixed points of this reduced network: $(M, R, L) = (0, 1, 0), (1, 0, 1)$.

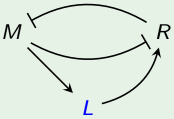
Moreover, since $B = P = M$, $A = L \wedge M$, we can recover the fixed points of the original network by *back-substituting*.

$$(M, P, B, R, A, L) = (M, M, M, R, L \wedge M, L) = (0, 0, 0, 1, 0, 0), \quad \text{and} \quad (1, 1, 1, 0, 1, 1).$$

A motivating example (cont.)

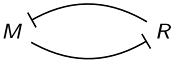
Partially reduced model of the *lac* operon

$$f_M = \overline{R}$$
$$f_R = \overline{L} \vee \overline{M}$$
$$f_L = M$$



We can reduce further. At equilibrium, $L = f_L = M$, so we can replace every instance of L with M:

$$f_M = \overline{R}$$
$$f_R = \overline{M} \vee \overline{M} = \overline{M}$$
$$\cancel{f_L = M}$$



There are two fixed points of this reduced network $(M, R) = (0, 1), (1, 0)$.

Moreover, since $L = B = P = M$ and $A = L \wedge M = M$, we can recover the steady-states of the original network by *back-substituting*.

$$(M, P, B, R, A, L) = (M, M, M, R, M, M) = (0, 0, 0, 1, 0, 0), \quad \text{and} \quad (1, 1, 1, 0, 1, 1).$$

General reduction

Reduction steps

1. Simplify the Boolean functions and wiring diagram.

   1.1 Reduce / simplfy Boolean expressions using Boolean algebra.

   1.2 Remove unnecessary edges from the wiring diagram.

2. Delete vertices x_i with no self-loop (equivalently, f_{x_i} doesn't depend on x_i), by doing the following:

   2.1 For all vertices y such that $x_i \longrightarrow y$, substitute f_{x_i} into x_i:

$$f_y(x_1 \ldots, \underbrace{\cdots x_i \cdots}_{\text{pos. } y}, \ldots, x_n) \qquad \text{becomes} \qquad f_y((x_1 \ldots, \underbrace{\cdots f_{x_i} \cdots}_{\text{pos. } y}, \ldots, x_n)\,.$$

   2.2 Replace edges $v \longrightarrow x_i \longrightarrow y$ by $v \longrightarrow y$ and remove x_i (and all edges to/from x_i).

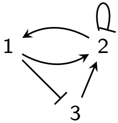
Exercise (HW)

In Step 2.2 above, how should you replace replace edges of the form:

- $v \longrightarrow x_i \relbar\joinrel\dashv y$
- $v \relbar\joinrel\dashv x_i \longrightarrow y$
- $v \relbar\joinrel\dashv x_i \relbar\joinrel\dashv y$

General reduction: an example

Consider the Boolean network $f(x) = (x_2,\ (x_1 \wedge x_3) \vee \overline{x_2},\ \overline{x_1})$.

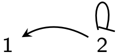


Let's remove $x_3 = \overline{x_1}$. The new Boolean functions are

$$\begin{aligned} h_1(x_1,x_2) &= f_1(x_1,x_2,x_3) = f_1(x_1,x_2,\overline{x_1}) = x_2\,, \\ h_2(x_1,x_2) &= f_2(x_1,x_2,x_3) = f_2(x_1,x_2,\overline{x_1}) = (x_1 \wedge \overline{x_1}) \vee \overline{x_2} \end{aligned}$$

However, $x_1 \wedge \overline{x_1} = 0$, and so

$$h_2(x_1,x_2) = (x_1 \wedge \overline{x_1}) \vee \overline{x_2} = 0 \vee \overline{x_2} = \overline{x_2}\,.$$



The reduced Boolean network is thus $h(x_1,x_2) = (x_2, \overline{x_2})$

To find the fixed points, we must solve the system $h_i = x_i$ for $i = 1, 2$:

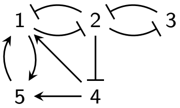
$$\begin{cases} h_1(x_1,x_2) = x_2 = x_1 \\ h_2(x_1,x_2) = \overline{x_2} = x_2\,. \end{cases}$$

Since $x_2 \neq \overline{x_2}$, there are no fixed points in the reduced BN, and thus none in the original BN.

General reduction: an example

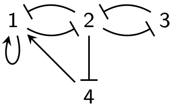
Consider the Boolean network:

$$f = (x_5 \vee \overline{x_2} \vee x_4,\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2},\ \overline{x_2},\ x_1 \vee x_4).$$



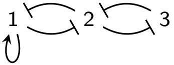
Remove $x_5 = x_1 \vee x_4$:

$$\begin{aligned} f &= ((x_1 \vee x_4) \vee \overline{x_2} \vee x_4,\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2},\ \overline{x_2}) \\ &= (x_1 \vee \overline{x_2} \vee x_4,\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2},\ \overline{x_2})\,. \end{aligned}$$



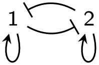
Remove $x_4 = \overline{x_2}$:

$$f = (x_1 \vee \overline{x_2} \vee \overline{x_2},\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2}) = (x_1 \vee \overline{x_2},\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2})$$



Remove $x_3 = \overline{x_2}$:

$$f = (x_1 \vee \overline{x_2},\ \overline{x_1} \wedge \overline{\overline{x_2}}) = (x_1 \vee \overline{x_2},\ \overline{x_1} \wedge x_2)$$



This yields the system:

$$\begin{cases} h_1(x_1, x_2) = x_1 \vee \overline{x_2} \\ h_2(x_1, x_2) = \overline{x_1} \wedge x_2 \\ x_3 = \overline{x_2} \\ x_4 = \overline{x_2} \\ x_5 = x_1 \vee x_4 \end{cases}$$

The reduced system (h_1, h_2) has 2 fixed points:

$$(x_1, x_2) = (1, 0),\quad (0, 1).$$

Thus, the original system has two fixed points:

$$(x_1, x_2, x_3, x_4, x_5) = (1, 0, 1, 1, 1),\quad (0, 1, 0, 0, 0).$$

Computational algebra software: Macaulay2 and Sage

Macaulay2 is a free computer algebra system developed by Dan Grayson (UIUC) and Mike Stillman (Cornell). It is named after the English mathematician Francis Macaulay (1862–1937).

It can be downloaded or used online at www.math.uiuc.edu/Macaulay2. Alternatively, it has been incorporated into the Sage Math Cloud: https://cloud.sagemath.org.

Let's see how to use Macaulay2 in Sage to do the Boolean reduction from the previous slide. First, tell Sage that we want to use Macaulay2 (hit *Shift-Enter* after each command):

```
%default_mode macaulay2
```

We want polynomials in variables $x_1, \dots x_5$, over the field $\mathbb{F}_2$, and $x_i^2 = x_i$:

```
R = ZZ/2[x1,x2,x3,x4,x5] / ideal(x1^2-x1, x2^2-x2, x3^2-x3, x4^2-x4, x5^2-x5);
```

For convenience, let's define $a|b := a + b + ab$ and $a\&b := a * b$:

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

Computational algebra software: Macaulay2 and Sage

Input the Boolean network $f = (f_1, f_2, f_3, f_4, f_5) = (x_5 \vee \overline{x_2} \vee x_4,\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2},\ \overline{x_2},\ x_1 \vee x_4)$:

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

Now, typing `f1` gives the following output:

```
x2*x4*x5 + x2*x4 + x2*x5 + x2 + 1
```

We can use the following commands to reduce the BN by substituting $x_5 = x_1 \vee x_4$:

```
f1=sub(f1,{x5=>f5});
f2=sub(f2,{x5=>f5});
f3=sub(f3,{x5=>f5});
f4=sub(f4,{x5=>f5});
```

Computational algebra software: Macaulay2 and Sage

The original Boolean network: $f = (f_1, f_2, f_3, f_4, f_5) = (x_5 \vee \overline{x_2} \vee x_4,\ \overline{x_1} \wedge \overline{x_3},\ \overline{x_2},\ \overline{x_2},\ x_1 \vee x_4)$.

Let's reduce further by removing x_4:

```
f1=sub(f1,{x4=>f4});
f2=sub(f2,{x4=>f4});
f3=sub(f3,{x4=>f4});
```

Finally, let's remove x_3:

```
f1=sub(f1,{x3=>f3});
f2=sub(f2,{x3=>f3});
```

To see the reduced network, type:

```
(f1,f2);
```

The output is:

```
(x1*x2 + x2 + 1, x1*x2 + x2)
Sequence
```

Application: Boolean model of Th-cell differentiation

White blood cells or *leukocytes* are in the immune system and fight diseases and infections.

One subtype are the *lymphocytes*, which includes the natural killer (NK) cells, B cells, and T cells, all which have different cellular functions.

The T-cells circulate throughout our bodies in the lymph fluid, looking for cellular abnormalities, infections, and diseases.

Helper T-cells (Th-cells) are a certain type of T-cells. They begin as *naïve*, or *Th0 cells*, and then differentiate into one of two phenotypes:

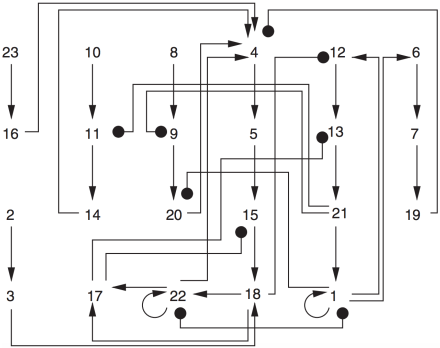
1. Type 1 are the *Th1 cells* which fight intracellular bacteria and protozoa.
2. Type 2 are the *Th2 cells* which fight extracellular parasites.

Malfunctions of immune responses involving Th1 phenotypes can result in autoimmune diseases, whereas malfunctions involving Th2 phenotypes can result in allergic reactions.

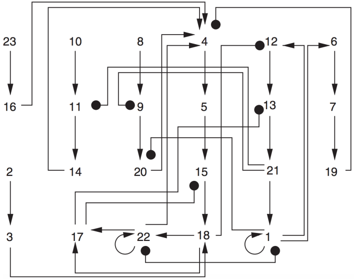
The biochemical signals that determine Th1 and Th2 differentiation act as a bistable switch, which permits either GATA3 or T-bet to be expressed, but not both. This was modeled using a 23-node Boolean network in Mendoza et. al (2006).

Boolean model of Th-cell differentiation (Mendoza, 2006)

$x_1 = \text{GATA3}$	$f_1 = (x_1 \vee x_{21}) \wedge \overline{x_{22}}$
$x_2 = \text{IFN-}\beta$	$f_2 = 0$
$x_3 = \text{IFN-}\beta\text{R}$	$f_3 = x_2$
$x_4 = \text{IFN-}\gamma$	$f_4 = (x_{14} \vee x_{16} \vee x_{20} \vee x_{22}) \wedge \overline{x_{19}}$
$x_5 = \text{IFN-}\gamma\text{R}$	$f_5 = x_4$
$x_6 = \text{IL-10}$	$f_6 = x_1$
$x_7 = \text{IL-10R}$	$f_7 = x_6$
$x_8 = \text{IL-12}$	$f_8 = 0$
$x_9 = \text{IL-12R}$	$f_9 = x_8 \wedge \overline{x_{21}}$
$x_{10} = \text{IL-18}$	$f_{10} = 0$
$x_{11} = \text{IL-18R}$	$f_{11} = x_{10} \wedge \overline{x_{21}}$
$x_{12} = \text{IL-4}$	$f_{12} = x_1 \wedge \overline{x_{18}}$
$x_{13} = \text{IL-4R}$	$f_{13} = x_{12} \wedge \overline{x_{17}}$
$x_{14} = \text{IRAK}$	$f_{14} = x_{11}$
$x_{15} = \text{JAK1}$	$f_{15} = x_5 \wedge \overline{x_{17}}$
$x_{16} = \text{NFAT}$	$f_{16} = x_{23}$
$x_{17} = \text{SOCS1}$	$f_{17} = x_{18} \vee x_{22}$
$x_{18} = \text{STAT1}$	$f_{18} = x_3 \vee x_{15}$
$x_{19} = \text{STAT3}$	$f_{19} = x_7$
$x_{20} = \text{STAT4}$	$f_{20} = x_9 \wedge \overline{x_1}$
$x_{21} = \text{STAT6}$	$f_{21} = x_{13}$
$x_{22} = \text{T-bet}$	$f_{22} = (x_{18} \vee x_{22}) \wedge \overline{x_1}$
$x_{23} = \text{TCR}$	$f_{23} = 0$



Boolean model of Th-cell differentiation



Reduced model (by removing, in order, $x_{23}, x_{21}, x_{20}, \ldots$):

Variable	Boolean function	Polynomial function
x_1 = GATA3	$h_1(x_1, x_{22}) = x_1 \wedge \overline{x_{22}}$	$h_1(x_1, x_{22}) = x_1 x_{22} + x_1$
x_{22} = T-bet	$h_{22}(x_1, x_{22}) = \overline{x_1} \wedge x_{22}$	$h_{22} = x_1 x_{22} + x_{22}$

There are three fixed points:

- $(0, 0)$: GATA3 and T-bet are inactive, the "signature" of Th0 cells.
- $(0, 1)$: Only T-bet is active, the signature of Th1 cells.
- $(1, 0)$: Only GATA3 is active, the signature of Th2-cells.

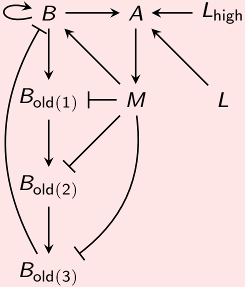
Application: Modeling time delays and degradration & dilution

In the last lecture, we saw how to add Boolean variables to model time delays and loss of concentration due to degradation / dilution.

Consider the following model of the *lac* operon (slightly modified from last lecture) that assumes that β-galactosidase takes several time-steops to degrade.

Example model

$$\begin{cases} f_M = A \\ f_A = (B \wedge L) \vee L_{\text{high}} \\ f_B = M \vee \Big(B \wedge \overline{B_{\text{old}(3)}}\Big) \\ f_{B_{\text{old}(1)}} = \overline{M} \wedge B \\ f_{B_{\text{old}(2)}} = \overline{M} \wedge B_{\text{old}(1)} \\ f_{B_{\text{old}(3)}} = \overline{M} \wedge B_{\text{old}(2)} \end{cases}$$



Do you see why the precise number of $B_{\text{old}(i)}$ variables is unimportant, regarding the number and quatitative nature of the fixed points? (HW exercise.)