Regulatory motifs: structural and functional building blocks of genetic computation

T. M. A. Fink

London Institute for Mathematical Sciences, Royal Institution, 21 Albermarle St, London W1S 4BS, UK

Developing and maintaining life requires a lot of computation. This is done by gene regulatory networks. But we have little understanding of how this computation is organized. I show that there is a direct correspondence between the structural and functional building blocks of regulatory networks, which I call regulatory motifs. I derive a simple bound on the range of function that these motifs can perform, in terms of the local network structure. I prove that this range is a small fraction of all possible functions, which severely constrains global network behavior. Part of this restriction is due to redundancy in the function that regulatory motifs can achieve—there are many ways to perform the same task. Regulatory motifs help us understanding how genetic computation is organized and what it can achieve.

Introduction

Genetic computation

Genetic computation is the computation done by gene regulatory networks. It is responsible for the creation of body structures [1], the determination of cell identity [2] and resilience to fluctuations in the environment. Scientists have investigated genetic computation for over 50 years [3, 4]. Their work tends to be at two ends of a spectrum. On the one hand, they have studied the average behavior of large networks. This work, exemplified by Boolean networks [5–7], considers the global dynamics of a large number interacting genes, typically with random connectivities and update rules. On the other hand, scientists have studied the specific behavior of small networks. This work, exemplified by the dynamics of network motifs [8, 9], considers the specific function of small genetic circuits [10, 11], typically comprising just a handful of nodes.

Understanding how genetic computation is organized is one of the top mathematical challenges of our time [12]. However, any comprehensive theory requires that we know how global behavior emerges from local function. Biology makes extensive use of modularity: the repeated use of specific design elements that can be usefully combined [13]. Modularity is likely to feature in genetic computation as well: the existence of functional subroutines that can be combined to create more advanced functionality. We see this in modern software design, in which new software is rarely written from scratch. Instead, it tends to combine existing subroutines, like how programs written in JavaScript combine a library of existing modular programs [14].

Structure and function

In any system that performs computation, structure and function are different. Structure dictates which parts of the system can interact, whereas function relates to the dynamics of the system as it evolves over time. For example, in electronic computing, the structure is the circuit architecture; in neural networks, it is the synapse connectivity; in cellular automata, it is the neighbors that define the lattice; in gene regulatory networks, it is the network connectivity of molecules that can bind together. In contrast to structure, function takes place in dynamical space [15]. This is much larger than structure space, since the number of states grows exponentially with the size of the system.

In general, at the most local level, structural and functional elements must correspond, because computation always takes place via a physical substrate—a physical dynamical system. In electronic computing and neural networks, for example, these elements are the transistor and the neuron. But as we progress to higher organizational length scales, this correspondence breaks down. Instead, function gets distributed over different parts of the structure, like how a computer program is computed by different parts of a circuit.

Any correspondence between structure and function in gene regulatory networks is valuable because it provides a window into how genetic computation is organized. The essence of this paper is that there is a formal correspondence between the two that extends beyond individual molecular logics, and this tells us how genetic computation is built up and constrained.

Genes and transcription factors

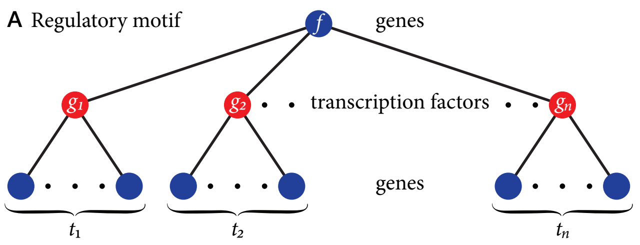
One of the distinctive features of gene regulatory networks is their bipartite nature: genes and transcription factors talk to each other but not themselves [16–18]. To see why, recall that a transcription factor is a protein or complex of proteins which are synthesized from expressed genes. Thus a transcription factor depends on one or more genes. A gene is a particular segment of DNA that codes for a protein, flanked by one or more binding sites for different transcription factors, which together promote or block the transcription of the gene. Thus a gene depends on one or more transcription factors. In this way, the expression levels of genes are determined by those of other genes, but only indirectly—transcription factors act as middlemen (The same is true if we swap genes and transcription factors, but I stick to the gene-centric perspective for simplicity.)

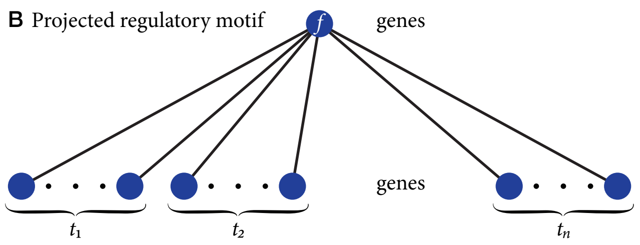
In previous work [18], I showed that a consequence of this bipartite nature is that the logical dependencies that one gene can have on other genes—what I call biological logics—are restricted. In other words, many of the possible logical dependencies are forbidden. I was able to enumerate these biological logics, some of which are given in [18], but I was unable to derive a simple expression for their number. By casting the problem in terms of structural and functional building blocks, in this work I derive a deeper understanding of the restriction on biological logics, and how they combine to perform more advanced computation.

Logics

I take genes and transcription factors to be either expressed or not expressed. This means that the expression of a gene is a Boolean function of the transcription factors that regulate it, and the expression of a transcription factor is a Boolean function of the genes that code for its proteins. A Boolean function is just a logic gate—a lookup table for the state of the output given the states of the inputs. I refer to a Boolean function and a logic interchangeably, with a bias towards logic, which is simpler. (Debate about discrete versus continuous expression neglects a deeper concern over whether sophisticated continuous computation is even possible [19].)

There are 2^{2^n} logics of n inputs. For example, for $n = 2$, these are true, false, a, b, $\overline{a}$, $\overline{b}$ ab, $a\overline{b}$, $\overline{a}b$, $\overline{a}\overline{b}$, $a+b$, $a+\overline{b}$, $\overline{a}+b$, $\overline{a}+\overline{b}$, $ab+\overline{a}\overline{b}$ and $a\overline{b}+\overline{a}b$. In this notation, $\overline{a}$ means NOT a, ab means a AND b, and $a+b$ means a OR b. Notice that two of these 16 logics depend on no inputs, four depend on one input, and 10 depend on two inputs. As we shall see, an important quantity is the number of logics of n inputs which depend on all n inputs, which we call $s(n)$, shown in eq. (4). The first few $s(n)$ are 2, 2, 10, 218, 64594 (OEIS A000371 [20]), starting at $n = 0$.





C

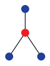
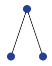
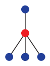
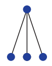
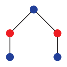
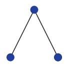
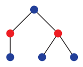
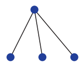
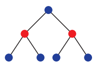
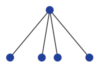
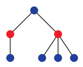
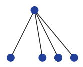
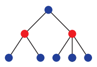
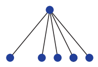
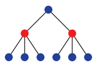
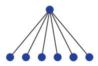
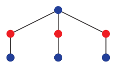
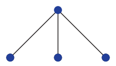
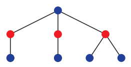
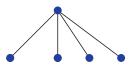
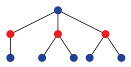
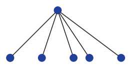
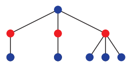
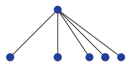
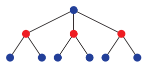
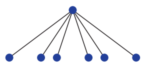
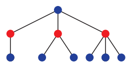
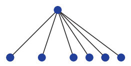
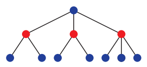
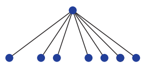
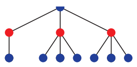
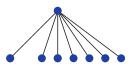
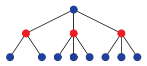
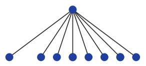
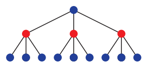
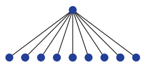
Short-hand	*Regulatory motifs*	*Lower bound* c_{lower}	*Biological logics* c	*Uppper bound* c_{upper}	*Projected regulatory motifs*	*All logics* ℓ
$\{1\}$		2	4	4		4
$\{2\}$		14	16	16		16
$\{3\}$		254	256	256		256
$\{1,1\}$		10	16	40		16
$\{1,2\}$		70	88	160		256
$\{2,2\}$		490	520	640		$65,536$
$\{1,3\}$		1,270	1,528	2,560		$65,536$
$\{2,3\}$		8,890	9,160	10,240		4.3×10^{9}
$\{3,3\}$		161,290	161,800	163,840		1.8×10^{19}
$\{1,1,1\}$		218	256	1,744		256
$\{1,1,2\}$		1,526	1,696	6,976		65,536
$\{1,2,2\}$		10,682	11,344	27,904		4.3×10^{9}
$\{1,1,3\}$		27,686	30,496	111,616		4.3×10^{9}
$\{2,2,2\}$		74,774	76,288	111,616		1.8×10^{19}
$\{1,2,3\}$		193,802	204,304	446,464		1.8×10^{19}
$\{2,2,3\}$		1,356,614	1,375,168	1,785,856		3.4×10^{38}
$\{1,3,3\}$		3,516,122	3,680,464	7,143,424		1.8×10^{19}
$\{2,3,3\}$		24,612,854	24,792,448	28,573,696		1.2×10^{77}
$\{3,3,3\}$		446,547,494	447,032,128	457,179,136		1.3×10^{154}

FIG. 1: **Regulatory motifs and their projections, and the number of logics they can support. A** In a regulatory motif, a gene (blue) depends on n transcription factors (red), each of which depends on $t_1,\ldots,t_n$ genes (blue). As a shorthand, we write $\{t_1,\ldots,t_n\}$, which counts the number of genes in the n branches of the tree. **B** In a projected motif, a gene depends on $t_1+\ldots+t_n$ genes; there are no transcription factor middlemen. **C** For the 19 simplest regulatory motifs (left), I show the number of biological logics c, and the lower and upper bounds on c. For their projections (right), I show the number of all possible logics ℓ. This tends to be much larger than the number of biological logics c.

In this paper

In this paper I do four things, which correspond to the four parts of the Results. First, I show that gene regulatory networks can be uniquely broken into structural building blocks (Fig. 1C left), and that these correspond to the functional building blocks of the network. Just as the global network structure is a combination of these regulatory motifs, the global network function is a combination of the biological logics that these motifs can carry out. Second, I study the number of different biological logics that these regulatory motifs can perform. By bounding it from below and above, I provide a simple estimate of this number in terms of the local network structure. Third, I prove that this number is a tiny fraction of the number of all possible logics of the same number of inputs. This puts severe constraints on the global behavior of regulatory networks. Fourth, I prove that part of the restriction on biological logics is due to their redundancy: different assignments of update rules to the nodes in a regulatory motif can produce the same biological logic. I calculate the average redundancy in terms of the local network structure. I conclude in the Discussion with implications of these results on the organization of genetic computation.

Results

Regulatory motifs: structural and functional building blocks

I start by showing there is a direct correspondence between the structural and functional building blocks of regulatory networks. Let's first look at the structure. The primitive structural building blocks are the connectivities that a gene has with other genes via transcription factor middlemen. For a gene that depends on n transcription factors, each of which depends on $t_1, \ldots, t_n$ genes (Fig. 1A), we use as a shorthand $\{t_1, \ldots, t_n\}$. For example, $\{2, 2\}$ denotes a gene which depends on two transcription factors, each of which depends on two genes. We call these pieces regulatory motifs, the 19 simplest of which are shown in Fig. 1C left. Any bipartite network can be broken into such pieces in a unique way. The recipe for doing so is to pick a gene (blue node), find all its second-nearest neighbors, then break each of these outer genes in half. (I could have drawn the building blocks in Fig. 1 with the genes as half-nodes, but I kept them whole for convenience.)

Now let's consider the functionality of these regulatory motifs. The top gene in Fig. 1A is a function of its transcription factors, which in turn are functions of their genes. Ultimately, the state of the top gene is set by the state of the bottom genes. We can work out this dependence by composing the logics. Let $x_{i,j}$ be the state of the jth gene in the ith branch in Fig. 1A. Then we can compose the logics to get a unique new logic:

$$\begin{aligned} &h(x_{1,1}, \ldots, x_{1,t_1}; \ldots; x_{n,1}, \ldots, x_{n,t_n}) = \\ &f\big(g_1(x_{1,1}, \ldots, x_{1,t_1}), \ldots, g_n(x_{n,1}, \ldots, x_{n,t_n})\big). \end{aligned} \quad (1)$$

For example, consider the regulatory motif $\{1, 2\}$. With the $x_{i,j}$ denoted a, b, c for convenience, $h(a, b, c) = f(g_1(a), g_2(b, c))$. Setting $f = g_1$ AND g_2, $g_1 = a$, and $g_2 = b$ OR c, the composed logic is $h = (a$ AND $b)$ OR $(a$ AND $c)$. Different choices of the logics for f and for g_1 and g_2 can give other composed logics.

Number of functions that a regulatory motif can perform

The number of biological logics c is the number of logical dependencies that one gene can have on other genes, that is, the number of different forms that h can take. My main mathematical result is that c is bounded by

$$c(t_1, \ldots, t_n) \;\geq\; s(n) \prod_{i=1}^{n} \left(2^{2^{t_i}}/2 - 1\right) = c_{\text{lower}} \quad (2)$$

$$c(t_1, \ldots, t_n) \;\leq\; s(n) \prod_{i=1}^{n} 2^{2^{t_i}}/2 = c_{\text{upper}}, \quad (3)$$

where $s(n)$ is the number of logics of n inputs which depend on all n inputs,

$$s(n) = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} 2^{2^i}. \quad (4)$$

These bounds are derived in the Methods. For example, for the regulatory motif $\{2, 3\}$, $c_{\text{upper}} = s(2) \cdot 2^{2^2}/2 \cdot 2^{2^3}/2 = 10{,}240$, which is not much higher than the true value, 9,160. Values of c and its bounds are given in Fig. 1C and plotted as error bars in Fig. 2.

The upper bound on the number of biological logics is also a good approximation to it. We know this because the ratio of the lower and upper bounds approaches one as the t_i increase. Dividing eqs. (2) and (3) by (3), the ratio of the bounds is

$$\frac{c_{\text{lower}}}{c_{\text{upper}}} = \prod_{i=1}^{n} \left(1 - 2/2^{2^{t_i}}\right).$$

For example, for the regulatory motif $\{3, 3\}$, the ratio is $(1 - 2/2^{2^3})^2 = 0.984$. So each bound is within 1.6% of c itself.

Function is restricted

In the absence of the transcription factor middlemen, we can simplify the regulatory motifs by projecting them: drawing an edge between genes connected to the same transcription factor. The projections are shown in Fig. 1C right.

The top gene in Fig. 1B depends on $t_1 + \ldots + t_n$ other genes. Since the number of logics of n inputs is 2^{2^n}, the number ℓ of all logical dependencies that a gene can have in the absence of the transcription factors is

$$\ell = 2^{2^{t_1 + \ldots + t_n}}.$$

The range of functionality c for the regulatory motifs is much smaller than the range ℓ for their projections (see Fig. 1C and Fig. 2). Our upper bound on c lets us quantify this difference:

$$\frac{c}{\ell} \leq \frac{c_{\text{upper}}}{\ell} = \frac{s(n)}{2^n} \frac{2^{2^{t_1}} \ldots 2^{2^{t_n}}}{2^{2^{t_1 + \ldots + t_n}}}. \quad (5)$$

Taking the logarithm gives a better sense of this ratio:

$$\log_2\left(\frac{c}{\ell}\right) \leq 2^n - n + \sum_{i=1}^{n} 2^{t_i} - \prod_{i=1}^{n} 2^{t_i},$$

where we make use of $s(n) \leq 2^{2^n}$ (see Methods).

For example, for the regulatory motif $\{2, 3\}$, the number of biological logics c is 9,160 and the number of possible logics ℓ is 2^{32}. Then $\log_2(c/\ell) = -18.8$, which is less than the bound of $2^2 - 2 + 2^2 + 2^3 - 2^2 2^3 = -18$.

Function is redundant

We know that the number of logical dependencies that one gene can have on other genes cannot be greater than the number of assignments of logics to the gene and transcription factors, that is, the number of choices of f and $g_1, \ldots, g_n$ in Fig. 1A. Since the number of logics of n inputs is 2^{2^n}, the number of assignments is

$$v = 2^{2^n} 2^{2^{t_1}} \ldots 2^{2^{t_n}}.$$

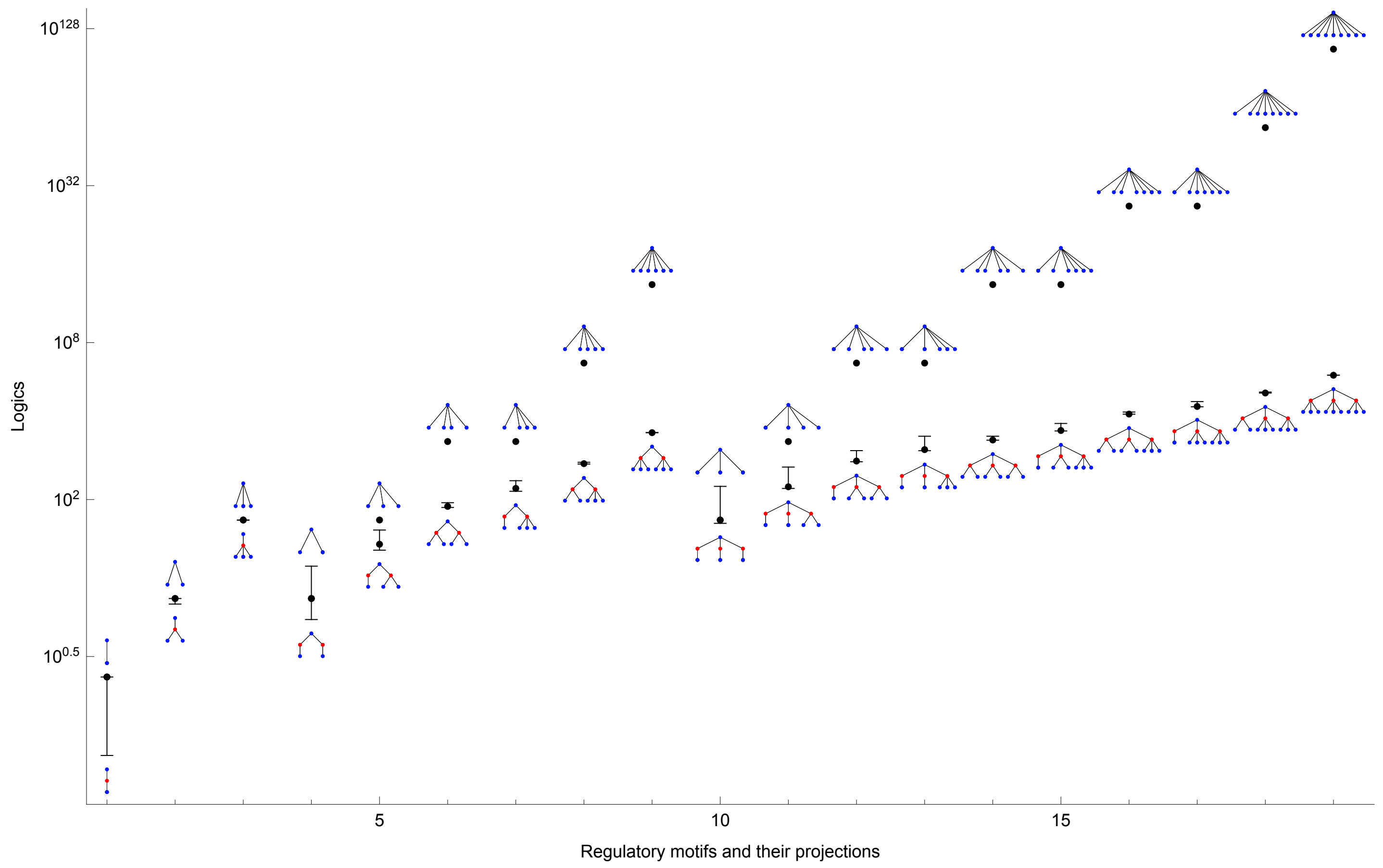


FIG. 2: **Plot of the number of logics for regulatory motifs and their projections.** Each row in Fig. 1C is plotted along the y-axis, in the same order that the rows appear in Fig. 1C. The bounds for the number of biological logics c are shown as points with error bars. The number of all possible logics ℓ are shown as points.

Let's compare v to the number of biological logics c:

$$\frac{v}{c} \geq \frac{v}{c_{\text{upper}}} = 2^n \frac{2^{2^n}}{s(n)} \geq 2^n, \tag{6}$$

where we make use of $s(n) \leq 2^{2^n}$. So the average redundancy is at least 2^n—though it can be considerably larger than this. Values of the average redundancy are shown for the 19 simplest regulatory motifs in Fig. 3. One instance of redundancy is that there is another way to compose the logic $h = (a \text{ AND } b) \text{ OR } (a \text{ AND } c)$ mentioned above. It also results from setting $f = \overline{g}_1$ AND $\overline{g}_2$, $g_1 = \overline{a}$, and $g_2 = \overline{b}$ AND $\overline{c}$.

For example, consider the regulatory motif $\{2, 2\}$. The number of assignments $v = 2^{2^2} 2^{2^2} 2^{2^2} = 4{,}096$ and the number of biological logics c is 520. So the average redundancy is 7.9, which is indeed greater than v/c_{upper}, which is $2^2 2^{2^2}/10 = 6.4$

Discussion

Correspondence of structure and function

In general, for any network that performs computation—genetic or otherwise—functional subroutines do not correspond to network substructures. Rather, function tends to be distributed over different parts of the structure. However, for gene regulatory networks, structure and function do correspond at the length scale of regulatory motifs. The functional subroutines—what I call biological logics—are precisely those that run on these network substructures. Just as global network structure is built from the regulatory motifs in Fig. 1C left, global network function is built from the logics that run on them.

The reason for this correspondence is because the network is bipartite: genes and transcription factors talk to each other but not themselves [16–18]. In the building blocks in Fig. 1C, all of the downstream transcription factors (red nodes) are surrounded by genes (blue nodes). This allows us to effectively integrate out the transcription factor logics, generating a restricted range of logical dependencies that the top gene can have on the bottom genes. Incidentally, bipartite networks are not the only kind of networks for which structure and function correspond. Tripartite networks and more complex network architectures also have this feature.

Function is restricted

The extent to which biological logics are restricted is bounded by eq. (5). For $n = 1$, or for all of the $t_i = 1$, there is no restriction; the number of biological and possible logics are the same, and $c/\ell = 1$. But the restriction quickly becomes severe for other regulatory motifs. The highest values that c/ℓ can take after 1 are 11/32 for the motif $\{1, 2\}$ and 0.026 for $\{1, 1, 2\}$. The ratio drops off for other regulatory motifs, reaching 10^{-14} for $\{2, 2, 2\}$ and 10^{-145} for $\{3, 3, 3\}$. When all of the $t_i = t$, the logarithm of c/ℓ is less than $2^n + n(2^t - 1) - 2^{nt}$.

For most of the regulatory motifs in Fig. 1C, the restriction on function c/ℓ is astronomical. This puts severe constraints on the global function of gene regulatory networks, because the global function is a composition of the local function performed by each of the regulatory motifs. As the number of motifs that are combined increases, the restriction on global behavior grows exponentially.

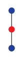
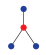
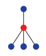
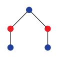
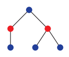
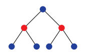
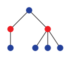
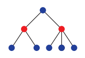
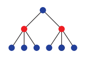
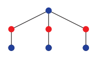
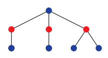
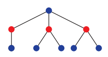
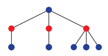
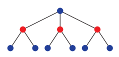
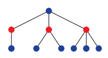
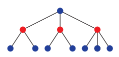
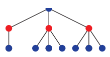
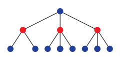
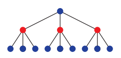
Short-hand	*Regulatory motifs*	*Logic assignments* v	*Biological logics* c	*Average redundancy* v/c
$\{1\}$		2^4	4	4
$\{2\}$		2^6	16	4
$\{3\}$		2^{10}	256	4
$\{1,1\}$		2^8	16	16
$\{1,2\}$		2^{10}	88	11.6
$\{2,2\}$		2^{12}	520	7.9
$\{1,3\}$		2^{14}	1,528	10.7
$\{2,3\}$		2^{16}	9,160	7.2
$\{3,3\}$		2^{20}	161,800	6.5
$\{1,1,1\}$		2^{14}	256	64
$\{1,1,2\}$		2^{16}	1,696	38.6
$\{1,2,2\}$		2^{18}	11,344	23.1
$\{1,1,3\}$		2^{20}	30,496	34.4
$\{2,2,2\}$		2^{20}	76,288	13.7
$\{1,2,3\}$		2^{22}	204,304	20.5
$\{2,2,3\}$		2^{24}	1,375,168	12.2
$\{1,3,3\}$		2^{26}	3,680,464	18.2
$\{2,3,3\}$		2^{28}	24,792,448	10.8
$\{3,3,3\}$		2^{32}	447,032,128	9.6

FIG. 3: **The average redundancy for regulatory motifs.** For each of the 19 simplest regulatory motifs, I show: the number of ways v of assigning logics to the gene and transcription factors (f and the g_i in Fig. 1A); the number of distinct logics c that these compose to (what I call biological logics); and their ratio, which is the average redundancy of this many-to-one map. The average redundancy is at least 2^n, where n is the number of branches, but it can be considerably higher than this.

Function is redundant

We know from eq. (6) that biological logics are on average redundant. If we think of the assignment of logics to the genes and transcription factors as the genotype (instructions), and the composed gene-gene logic as the phenotype (behavior), then the ratio of the number of genotypes to the number of phenotypes is at least $2^n 2^{2^n}/s(n)$, which is itself at least 2^n. The mapping of many genotypes to fewer phenotypes is highly prevalent in biological systems, ranging from RNA secondary structure to protein folding to biological clocks. This gives rise to neutral networks and can confer robustness to errors [21].

Intriguingly, computational evidence suggests that some biological logics are much more redundant than others. These phenotypes are more likely to be observed simply because there are more ways to design them. For example, for the regulatory motif $\{2,3\}$, $2^{16} = 65{,}536$ assignments map to 9,160 biological logics. But of the 2^{16} assignments, 18% map to true and false, 11% map to 14 simple logics such as a, b, and a OR b, 17% map to 254 more complex logics, such as c OR d OR e, and the rest map to 8,890 more complex logics still. If this trend holds for regulatory motifs in general, it would imply that biological logics are not only restricted, but also tend to be simple. This is an open theoretical question with important consequences, and I hope that others will try to answer it.

More generally, the composition of logics is an archetypal system for understanding input-output maps. Across a broad range of systems in nature and mathematics, input-output maps tend to be highly biased towards simple outputs [25, 26]. A theory for the distribution of redundancies mentioned above could help provide an explanation of this empirical trend.

Methods

Logics that depend on all of their inputs

By the principle of inclusion and exclusion, the number of logics of n inputs which depend on all n inputs is

$$s(n) = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} 2^{2^i}, \qquad (7)$$

which is the inverse binomial transform of 2^{2^n} (OEIS A000371 [20]). It is bounded from below and above by

$$2^{2^n} - n\, 2^{2^{n-1}} \le s(n) \le 2^{2^n}. \qquad (8)$$

The right side follows from the definition of $s(n)$. The left side can be deduced by showing that the magnitude of the $(i-1)$th term in (7) is less than half that of the ith term, that is,

$$\binom{n}{i-1} 2^{2^{i-1}} \le \frac{1}{2} \binom{n}{i} 2^{2^i}.$$

This implies that $2i \le (n-i+1) 2^{2^{i-1}}$. Since $n-i+1$ is at least 1, we need only that $2i \le 2^{2^{i-1}}$. It is indeed for all $i \ge 1$, establishing the lower bound in (8). It implies that $s(n)$ rapidly approaches 2^{2^n}.

Proof of upper bound on biological logics

The upper bound in eq. (3) is not at all obvious. It is also delicate, in that it is an equality for $n = 1$. My first proof of the bound, not given here, was cumbersome. But, like a climber who on reaching the summit sees a superior path of ascent, as soon as I proved it I found a much simpler proof the next day. I thank my colleague Yang-Hui He for prompting one of the steps.

Here is the simpler proof. My starting point is a result from a recent paper [18], in which I derived an exact—but difficult to apply—expression for the number of biological logics. It is

$$c(t_1, \ldots, t_n) = \sum_{m=0}^{n} s(m) \sum_{\sigma_1 \ldots \sigma_m} \alpha_{\sigma_1} \ldots \alpha_{\sigma_n}, \qquad (9)$$

where

$$\alpha_i = 2^{2^i}/2 - 1,$$

and the second sum adds up the product of all m-tuples of the α_i. A few examples illustrate what this means:

$$\begin{aligned} c(i) &= 2 + 2\,\alpha_i, \\ c(i,j) &= 2 + 2(\alpha_i + \alpha_j) + 10\,\alpha_i\alpha_j, \\ c(i,j,k) &= 2 + 2(\alpha_i + \alpha_j + \alpha_k) \\ &+ 10\big(\alpha_i\alpha_j + \alpha_j\alpha_k + \alpha_i\alpha_k\big) + 218\,\alpha_i\alpha_j\alpha_k. \end{aligned}$$

Let's start by rewriting (9) as

$$c(t_1,\ldots,t_n) \;=\; \sum_{m=0}^{n} s(m)A_m,$$

where A_m is the sum over the product of all m-tuples of the α_i:

$$\begin{aligned}A_0 &= 1 \quad \text{(by definition)},\\ A_1 &= \alpha_1+\alpha_2+\ldots+\alpha_n,\\ A_2 &= \alpha_1\alpha_2+\ldots+\alpha_1\alpha_n+\ldots\alpha_n\alpha_1+\ldots+\alpha_n\alpha_{n-1},\\ &\;\vdots\\ A_n &= \alpha_1\ldots\alpha_n.\end{aligned}$$

We need to show that

$$\sum_{m=0}^{n} s(m)A_m \le s(n)\prod_{i=1}^{n}(\alpha_i+1). \qquad (10)$$

To do so, consider the product

$$\prod_{i=1}^{n}(x+\alpha_i) = A_0x^n + A_1x^{n-1}+\ldots+A_nx^0.$$

Setting $x=1$, this gives

$$\prod_{i=1}^{n}(\alpha_i+1) = A_0+A_1+\ldots+A_n.$$

Substituting this into eq. (10) gives

$$\sum_{m=0}^{n} s(m)A_m \le s(n)\sum_{m=0}^{n} A_m.$$

Since $s(m)$ is a non-decreasing function of m, $s(m)\le s(n)$, and the above is true by inspection. Therefore

$$c(t_1,\ldots,t_n) \le s(n)\prod_{i=1}^{n} 2^{2^{t_i}}/2 = c_{\text{upper}},$$

proving eq. (3).

Proof of lower bound on biological logics

In addition to the upper bound, we can write a lower bound of a similar form. To do so, we just take the last ($m=n$) term in eq. (9) and substitute in α_i. This gives

$$c_{\text{lower}} = s(n)\prod_{i=1}^{n}\Big(2^{2^{t_i}}/2-1\Big) \le c(t_1,\ldots,t_n),$$

proving eq. (2).

Acknowledgements

Funding: This research was supported by a grant from bit.bio. **Competing interests:** The author declares that he has no competing interests.

[1] C. Gomez et al., Control of segment number in vertebrate embryos, Nature **454**, 335 (2008).
[2] M. Pawlowski et al., Inducible and deterministic forward programming of human pluripotent stem cells into neurons, skeletal myocytes, and oligodendrocytes, Stem Cell Reports **8**, 803 (2017).
[3] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, J Theor Biol 22, 437 (1969).
[4] S. Huang, G. Eichler, Y. Bar-Yam, and D. E. Ingber, Cell fates as high-dimensional attractor states of a complex gene regulatory network, Phys Rev Lett **94**, 128701 (2005).
[5] J. E. Socolar and S. A. Kauffman, Scaling in ordered and critical random Boolean networks, Phys Rev Lett **90**, 068702 (2003).
[6] B. Samuelsson and C. Troein, Superpolynomial growth in the number of attractors in Kauffman networks, Phys Rev Lett **90**, 098701 (2003).
[7] I. Shmulevich and S. A. Kauffman, Activities and sensitivities in Boolean network models, Phys Rev Lett **93**, 048701 (2004).
[8] R. Milo et al., Network motifs: Simple building blocks of complex networks, Science **298**, 824 (2002).
[9] R. Milo et al., Superfamilies of evolved and designed networks, Science **303**, 1538 (2004).
[10] S. Mangan and U. Alon, Structure and function of the feed-forward loop network motif, P Natl Acad Sci USA **100**, 980 (2003).
[11] S. E. Ahnert and T. M. A. Fink, Form and function in gene regulatory networks J. R. Soc. Interface **13**, 20160179 (2016).
[12] T. Whipple, "23 Mathematical Challenges", *The Times*, June 2021.
[13] G. P. Wagner, M. Pavlicev, J. M. Cheverud, The road to modularity, Nat Rev Genet **8**, 921 (2007).
[14] A. Decan, T. Mens and E. Constantinou, On the evolution of technical lag in the npm package dependency network," IEEE ICSME, p. 404 (2018).
[15] S. Wolfram, *A New Kind of Science* (Wolfram Media, 2002), p. 637.
[16] R. Hannam, R. Kuhn, and A. Annibale, Percolation in bipartite Boolean networks and its role in sustaining life, J Phys A **52**, 334002 (2019).
[17] R. Hannam, Cell states, fates and reprogramming, Ph.D. thesis, King's College London (2019).
[18] T. M. A. Fink and R. Hannam, Biological logics are restricted, submitted to Science Advances (2021).
[19] S. Wolfram, *A New Kind of Science* (Wolfram Media, 2002), pp. 729, 1128.
[20] N. J. A. Sloane, editor, The On-Line Encyclopedia of Integer Sequences, published electronically at https://oeis.org, 2021.
[21] A. Wagner, Robustness and evolvability: a paradox resolved, P R Soc B, **275** 91 (2008).
[22] C. Buccitelli and M. Selbach, mRNAs, proteins and the emerging principles of gene expression control, Nat. Rev. Genet **21**, 630 (2020).
[23] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Random graphs with arbitrary degree distributions and their applications, Phys Rev E **64**, 026118 (2001).
[24] J. L. Payne and A. Wagner, Mechanisms of mutational robustness in transcriptional regulation, Front Genet **6**, 322 (2015).
[25] K. Dingle, C. Q. Camargo, and A. A. Louis, Input-output maps are strongly biased towards simple outputs, Nat. Commun. **9** (2018).
[26] I. G. Johnston et al., Symmetry and simplicity spontaneously emerge from the algorithmic nature of evolution, P Natl Acad Sci USA **119**, e2113883119 (2022).
[27] K. Raman and A. Wagner, The evolvability of programmable hardware, J. R. Soc. Interface **8**, 269 (2011).
[28] S. Bilke and F. Sjunnesson, Stability of the Kauffman model, Phys Rev E **65**, 016129 (2001).
[29] M. Reed, Why is mathematical biology so hard?, Not Am Math Soc **51**, 338 (2004).
[30] M. Reed, Mathematical biology is good for mathematics, Not Am Math Soc **62**, 1172 (2015).