J Stat Phys (2008) 130: 1063–1085
DOI 10.1007/s10955-008-9486-2

Using the Renormalization Group to Classify Boolean Functions

S.N. Coppersmith

Received: 24 December 2006 / Accepted: 10 January 2008 / Published online: 26 January 2008

Abstract This paper argues that the renormalization group technique used to characterize phase transitions in condensed matter systems can be used to classify Boolean functions. A renormalization group transformation is presented that maps an arbitrary Boolean function of N Boolean variables to one of $N-1$ variables. Applying this transformation to a generic Boolean function (one whose output for each input is chosen randomly and independently to be one or zero with equal probability) yields another generic Boolean function. Moreover, applying the transformation to some other functions known to be non-generic, such as Boolean functions that can be written as polynomials of degree ξ with $\xi \ll N$ and functions that depend on composite variables such as the arithmetic sum of the inputs, yields non-generic results. One can thus define different phases of Boolean functions as classes of functions with different types of behavior upon repeated application of the renormalization transformation. Possible relationships between different phases of Boolean functions and computational complexity classes studied in computer science are discussed.

Keywords Renormalization group · Computational complexity

1 Introduction

This paper argues that the set of Boolean functions of N Boolean variables can, as $N \to \infty$, be classified into phases using a method known in statistical physics as the renormalization group (RG) [19, 24, 50, 51]. The RG technique, originally formulated to provide insight into the nature of phase transitions in statistical mechanical systems [24, 50], involves taking a problem with N variables and rewriting it as a problem involving fewer variables, and then investigating the properties of the resulting sequence of functions as this procedure is iterated. Here, we will define a procedure that transforms a Boolean function of N Boolean

S.N. Coppersmith (✉)
Department of Physics, University of Wisconsin-Madison, 1150 University Avenue, Madison, WI 53706, USA
e-mail: snc@physics.wisc.edu

variables into a Boolean function of $N-1$ variables.[1] The transformation used here is very simple—the new function is one if the original function changes its output value when a given input variable's value is changed, and is zero if it does not.

It is shown that different classes of functions have different behavior upon repeated application of a renormalization group transformation. In analogy with well-known results in statistical mechanics [19], we interpret functions exhibiting different behaviors after many renormalizations as being in different phases. Applying the renormalization transformation to a generic Boolean function, whose output for each input is chosen randomly and independently to be one or zero with equal probability, yields another generic Boolean function; this "fixed point" behavior is evidence of the presence of a generic phase of Boolean functions. Functions that can be written as low-order polynomials and functions of composite variables such as the arithmetic sum of the values of the inputs are demonstrated to yield non-generic behavior upon renormalization. Therefore, the RG distinguishes some individual functions (that happen to be functions that can be specified much more efficiently than by making a table of output values) as non-generic.[2]

Being able to classify Boolean functions is of great interest in the field of computational complexity, the study of how the computational resources needed to solve different problems scale with the size of the problem specification. A problem as defined in computational complexity theory corresponds to a family of Boolean functions, one function for each size of the input specification [46]; the statement of how difficulty scales with problem size is a statement about the computational resources needed to compute these functions as the number of function arguments tends to infinity. Therefore, determining the computational resources needed to solve a problem is equivalent to determining the dependence on N of the computational resources needed to evaluate a set of Boolean functions of N Boolean variables, $f(x_1, \ldots, x_N)$, for any of the 2^N possible input configurations.[3] Classifying Boolean functions by the computational resources required to compute them is an extremely powerful concept [35], but it is difficult to implement. Whether or not the two well-known complexity classes P (problems that can be solved with resources that scale polynomially with the size of the problem specification) is equal to NP (problems for which a solution can be verified with resources that scale polynomially with the size of the problem specification) [12, 31] is a great outstanding question in computational complexity theory and in mathematics generally [1, 11, 23, 42, 49], and indeed, it has not been proven whether P is distinguishable from PSPACE, the class of problems that can be solved using polynomially bounded memory (but possibly exponential time) [35]. These difficulties in distinguishing computational complexity classes motivate the consideration of other methods of classifying Boolean functions.

As in statistical mechanics, the distinctions between phases introduced here for functions of N variables are completely sharp only in the thermodynamic limit $N \to \infty$.[4] However,

[1] It is more usual for renormalization group transformations to reduce the number of variables by a factor of two instead of by one. Examples of renormalization groups for condensed matter systems that eliminate one variable at a time are described in [9, 33, 48].

[2] Of course, we expect many other types of non-generic functions to exist.

[3] Computational complexity classes are often defined by referring to the computational resources needed to answer the decision problem of whether or not a given input string is in a given language [23], but the terminology in terms of functions used here is also accepted [7, 46].

[4] Distinctions between phases are sharp only in the limit $N \to \infty$ because they are based on being able to distinguish quantities that are exponentially large in N from those that are polynomially large in N. For fixed N any large number can be written as a very large coefficient times a polynomially bounded quantity, so for a fixed but large N, one examines whether or not the exponent and the coefficients are both less than some fixed bounds.

in our numerical examples we will apply the RG transformation to large but finite systems, similarly to the calculations in the classic paper of Wilson [51].

Any Boolean function $f(x_1, \ldots, x_N)$ of the N Boolean variables $x_1, \ldots, x_N$ can be written as a polynomial in the x_j using modulo-two addition. This follows because the variables and function all can be only 0 or 1, so $f(x_1, \ldots, x_N)$ can be written as

$$\begin{aligned} f(x_1, \ldots, x_N) = {} & A_{00\ldots00}(1 \oplus x_1)(1 \oplus x_2) \cdots (1 \oplus x_{N-1})(1 \oplus x_N) \\ & \oplus A_{00\ldots01}(1 \oplus x_1)(1 \oplus x_2) \cdots (1 \oplus x_{N-1})(x_N) \\ & \cdots \\ & \oplus A_{11\ldots10}(x_1)(x_2) \cdots (x_{N-1})(1 \oplus x_N) \\ & \oplus A_{11\ldots11}(x_1)(x_2) \cdots (x_{N-1})(x_N), \end{aligned} \tag{1}$$

where $A_{x_1,\ldots,x_N} = f(x_1, \ldots, x_N)$.[5] As Shannon pointed out [41], the number of different possible functions is 2^{2^N} (this follows because each of the 2^N coefficients $A_{\alpha_1,\ldots,\alpha_N}$ can be either one or zero), which is much larger than the number of functions that can be computed using resources that scale no faster than as a polynomial of N, which scales asymptotically as $(CN)^t$, where C is a constant and t is a polynomial in N [40]. This counting argument demonstrates that evaluating almost any Boolean function with N arguments requires computational resources that are exponentially large in N. However, it does not provide a means for determining whether or not a given function can be computed with resources bounded by a polynomial of N.

The proposed relevance of phases of Boolean functions for computational complexity relies on the observation that typical functions whose values are chosen independently and randomly to be one or zero with equal probability for each input configuration are hard to compute; there is essentially no simpler way to specify the function than to enumerate the output for each input separately [32].[6] Though it is true that when one chooses the output values randomly, then the probability of obtaining, e.g., a function whose output is one for all inputs is exactly the same as obtaining any other specific function, the constant function is atypical. The RG approach identifies whether or not an individual function is in the phase of generic Boolean functions.

The relationships between phases as defined here and computational complexity classes are not simple. It is shown below that there are problems that are in P that correspond to functions that are in the generic phase, and some non-generic functions correspond to problems that are not in P. The possible utility of characterizing phases of Boolean functions to the study of computational complexity classes arises because of the plausibility of the conjecture that the functions in the generic phase that can be computed with polynomially bounded resources have the property that they are close to a phase boundary in the sense that they differ from a non-generic function on a small fraction of input configura-

[5] In (1) the use of modulo-two addition is not necessary, but using modulo-two addition is extremely convenient when one is characterizing the properties of the functions obtained when the RG transformation is applied.

[6] Some non-generic functions are also hard to compute, but their presence or absence is irrelevant to the question of whether a typical Boolean function in which the function values are chosen randomly can be computed efficiently.

tions. If all functions that can be computed with polynomially bounded resources are either in a non-generic phase or near a non-generic phase boundary, then demonstrating that P and NP are distinct could be done if one could identify a problem in NP that gives rise to functions that (in the limit $N \to \infty$) are in the generic phase and also have the property that all functions yielding the same output on almost all the input configurations are also in the generic phase.

The paper is organized as follows. Section 2 defines the RG transformation that maps a Boolean function of N variables into a Boolean function of $N-1$ variables. It also characterizes the behavior of generic Boolean functions when the RG transformation is iterated repeatedly, and demonstrates that an attracting generic fixed point (and hence a generic phase) exists, and that functions exist that are non-generic. Section 3 examines the RG flows of functions in the generic phase, in particular how many iterations of the renormalization transformation are required to transform a function whose outputs are chosen independently and randomly to be one or zero with probabilities p and $1-p$ to one that is indistinguishable from a fixed point function, for which the two outputs are equally probable. In Sect. 4 the relationships between phases and computational complexity classes are explored. It is shown that the generic phase appears to include functions of N variables that can be computed with resources bounded by a polynomial of N, and that there are non-generic functions that cannot be computed with resources bounded by a polynomial of N. It is conjectured that every function corresponding to a problem that is in P is close to being non-generic in the sense that it can be written as the sum of a non-generic function plus a contribution that is nonzero on a small fraction of the input configurations. Such a function can be identified by checking whether or not each of the functions whose outputs differ from the original one on a small fraction of the input configurations is non-generic, so if the conjecture holds, it provides a means to demonstrate that an individual function cannot be computed with resources bounded polynomially in N. Section 5 discusses a specific class of Boolean functions that correspond to problems that are in NP and have properties that make it plausible that they are in the generic phase and also not near a non-generic phase boundary. Section 6 discusses the results in the framework of phase transitions in condensed matter systems, which renormalization group transformations are typically used to study, and also discusses how the strategy discussed here avoids the difficulties of "natural proofs" for addressing the P versus NP problem that are described in [39]. Section 7 presents the conclusions. Appendix A presents the arguments for why it is plausible that most functions that can be computed with polynomially bounded resources can be written as a non-generic function plus a term that is nonzero for a small fraction of input configurations. Appendix B shows that almost all Boolean functions cannot be written either as a low-order polynomial plus a term that is nonzero on a small fraction of the inputs or as a function that can be computed with resources bounded by a polynomial of N plus a term that is nonzero on a small fraction of the inputs.

2 Renormalization Group Transformation

The renormalization group (RG) procedure we define takes a given function of N variables and generates a function of $N-1$ variables [19, 24, 48, 50, 51]. The variable that is eliminated is called the "decimated" variable. The procedure can be iterated, mapping a function of $N-1$ variables into one of $N-2$ variables, etc.

The RG transformation proposed here specifies whether the original function's value changes if a given input variable is changed. Specifically, given a function $f(x_1, \ldots, x_N) \equiv f(\mathbf{x})$, we define

$$g_{i_1}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_N) \equiv g_{i_1}(\mathbf{x}') = f(x_1, \ldots, x_{i_1-1}, 0, x_{i_1+1}, \ldots, x_N) \oplus f(x_1, \ldots, x_{i_1-1}, 1, x_{i_1+1}, \ldots, x_N), \quad (2)$$

where $\oplus$ denotes addition modulo two,[7] and the vector $\mathbf{x}'$ denotes the set of undecimated variables. The function $g_{i_1}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_N)$ is one if the output of the function f changes when the value of the decimated variable x_{i_1} is changed and zero if it does not. Once g_{i_1} has been obtained, the procedure can be repeated and one can define g_{i_1,i_2} as

$$\begin{aligned} g_{i_1,i_2}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2+1}, x_N) &\equiv g_{i_1,i_2}(\mathbf{x}') \\ &= g_{i_1}(x_1, \ldots, x_{i_2-1}, 0, x_{i_2+1}, \ldots, x_N) \\ &\quad \oplus g_{i_1}(x_1, \ldots, x_{i_2-1}, 1, x_{i_2+1}, \ldots, x_N) \\ &= f(x_1, \ldots, x_{i_1-1}, 0, x_{i_1+1}, \ldots, x_{i_2-1}, 0, x_{i_2+1}, \ldots, x_N) \\ &\quad \oplus f(x_1, \ldots, x_{i_1-1}, 0, x_{i_1+1}, \ldots, x_{i_2-1}, 1, x_{i_2+1}, \ldots, x_N) \\ &\quad \oplus f(x_1, \ldots, x_{i_1-1}, 1, x_{i_1+1}, \ldots, x_{i_2-1}, 0, x_{i_2+1}, \ldots, x_N) \\ &\quad \oplus f(x_1, \ldots, x_{i_1-1}, 1, x_{i_1+1}, \ldots, x_{i_2-1}, 1, x_{i_2+1}, \ldots, x_N). \end{aligned} \quad (3)$$

It is straightforward to verify that the function $g_{x_{i_1},\ldots,x_{i_m}}(\mathbf{x}')$ obtained by decimating the m variables $x_{i_1}, \ldots, x_{i_m}$ does not depend on the order in which the variables are decimated.

2.1 Generic Phase

First we examine functions for which each coefficient $A^{(0)}_{\alpha_1,\alpha_2,\ldots,\alpha_N}$ in (1) is an independent random variable chosen to be one with probability p_0 and zero with probability $q_0 = 1 - p_0$, where $0 < p_0 < 1$. Numerical results of the RG procedure applied to such functions are shown in Fig. 1.

The coefficients $A^{(i_1)}_{x_1,\ldots,x_{i_1-1},x_{i_1+1},\ldots,x_N}$ that characterize the function $g_{i_1}(\mathbf{x}')$ obtained by decimating the variable i_1 via (2) are

$$A^{(i_1)}_{x_1,\ldots,x_{i_1-1},x_{i_1+1},\ldots,x_N} = A_{x_1,\ldots,x_{i_1-1},0,x_{i_1+1},\ldots,x_N} \oplus A_{x_1,\ldots,x_{i_1-1},1,x_{i_1+1},\ldots,x_N}. \quad (4)$$

The original $A^{(0)}$'s are independent random variables, so it follows that the $A^{(i_1)}$'s are independent random variables that are one with probability $p_1 = 2p_0q_0$ and zero with probability $1 - p_1$. After decimating ℓ variables, the coefficients are still independent random variables, and they are now one with probability p_ℓ and zero with probability $1 - p_\ell$, where the p_ℓ satisfy

$$p_{\ell+1} = 2p_\ell(1 - p_\ell). \quad (5)$$

[7] These polynomials have a natural interpretation in terms of arithmetic circuits, which are Boolean circuits composed of two types of gates, AND and EXCLUSIVE-OR. See, e.g., [37].

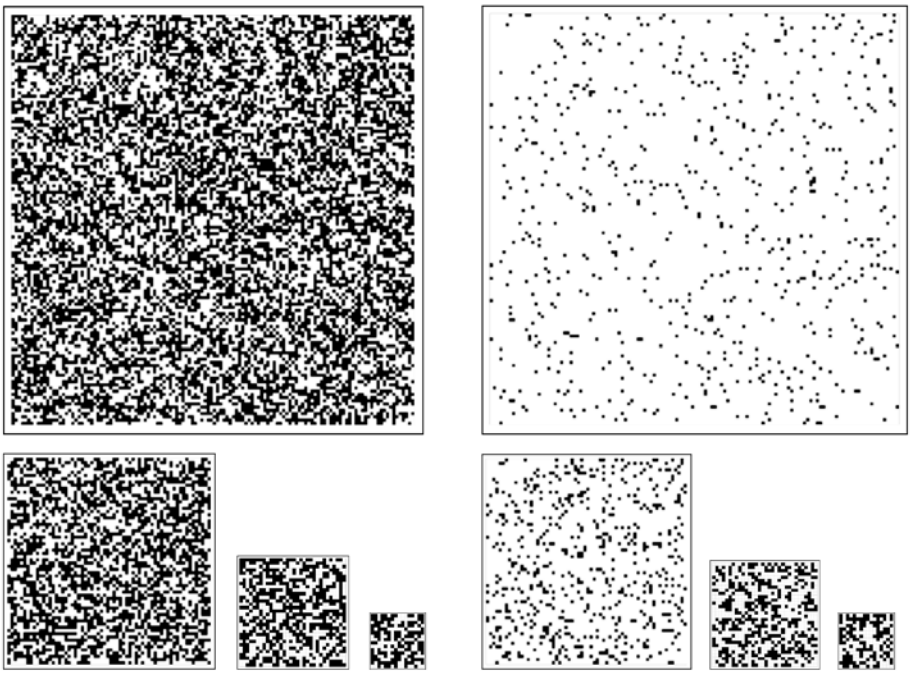

Fig. 1 Results of renormalization group (RG) transformation applied to functions in the generic phase. In both panels, the original function (*top*) is constructed by choosing the output value for each input configuration of $N = 14$ variables independently and pseudo-randomly to be 1 with probability p and 0 with probability $1 - p$. Configurations are shown after each two RG steps, so that in the four plots for each function, N takes on the values 14, 12, 10, and 8. In the plots, the x-coordinates are the binary expansions of the values of the first $N/2$ variables, and the y-coordinates are the binary expansions of the last $N/2$ variables. *Left*: Initial function has $p = 1/2$, characteristic of the generic fixed point. *Right*: Initial function has $p = 0.04$; the "flow" towards $p = 1/2$ as the RG is iterated is apparent (the fractions of nonzero outputs in the four plots in the *right panel* are 0.0402, 0.141, 0.388, and 0.473)

Solving (5) yields

$$p_\ell = \frac{1}{2}\left(1 - (1 - 2p_0)^{2^\ell}\right). \tag{6}$$

For any p_0 satisfying $0 < p_0 < 1$, the values of the p_ℓ flow as ℓ increases and eventually approach the fixed-point value of $1/2$, behavior that is analogous to that displayed by the parameters used to specify the partition functions describing thermodynamic phases in statistical mechanical systems [19], and so we interpret this behavior as evidence that there is a phase of generic Boolean functions.

It is consistent to call the phase generic because it is likely that the attracting fixed point describes almost all Boolean functions. Evidence for this is that $\mathcal{F}_\epsilon$, the fraction of Boolean functions that have an output of one on a fraction $(1 + \epsilon)/2$ of the $\Omega \equiv 2^N$ input configurations, is

$$\mathcal{F}_\epsilon = \frac{1}{2^\Omega} \frac{\Omega!}{[\frac{\Omega}{2}(1+\epsilon)]![\frac{\Omega}{2}(1-\epsilon)]!} \approx \sqrt{\frac{2}{\pi\Omega}} e^{-\epsilon^2\Omega/2}, \tag{7}$$

where the second line holds for $\epsilon \ll 1$. Therefore, $\mathcal{F}_\epsilon$ vanishes as a double exponential of N whenever $\epsilon > 2^{-\beta N}$ with $\beta < 1/2$. This property is characteristic of functions with $p = 1/2$. We choose to define the generic phase in terms of the results obtained after $N/2$ steps of the RG process. With this definition, functions in which the original $A^{(0)}$'s are chosen randomly and independently with $p_0 > 2^{-N(\frac{1}{2}-\epsilon)}$, with ϵ infinitesimal, are in the generic phase.[8] Conversely, functions that yield a nonzero output for a fraction of input configurations less than $C2^{-N(\frac{1}{2}+\epsilon)}$ or greater than $1 - C2^{-N(\frac{1}{2}+\epsilon)}$, with C a constant of order unity, are nongeneric.[9]

For functions in the generic phase, all the functions generated by the renormalization procedure applied more than $N/2$ times but which still have of order N undecimated variables have the property that their output is nonzero for a fraction of input configurations that deviates from $1/2$ by an amount that is exponentially small in N. For any $\eta \ll N$ that is greater than $N/2$, the probability that *all* of the $N!/[\eta!(N-\eta)!]$ functions yielded by η renormalizations yield one for a fraction of the inputs that differs from $1/2$ by an amount less than $2^{-\beta N}$ differs from unity by an amount that vanishes as a double exponential of N for some positive β bounded away from zero.

2.2 Non-generic Functions

We next demonstrate that Boolean functions that can be written as polynomials of degree of ξ or less with $\xi < N$ have the property that they yield zero after $\xi + 1$ renormalizations, for any choice of the decimated variables.

First we examine a simple example. The parity function $\mathcal{P}(x_1, \ldots, x_N)$, which is 1 if an odd number of input variables are 1 and 0 if an even number of the input variables are 1 [18, 21, 49, 52], can be written as

$$\mathcal{P}(x_1, \ldots, x_N) = x_1 \oplus x_2 \oplus \cdots \oplus x_N. \tag{8}$$

There are many less efficient ways to write the parity function, but the result of the renormalization procedure does not depend on how one has chosen to write the function, since it can be computed knowing only the values of the function for all different input configurations. For the parity function, one finds, for any choice of decimated variables x_{j_1} and x_{j_2}, the functions resulting from one and two renormalizations, $g^P_{j_1}(\mathbf{x}')$ and $g^P_{i_{j_1,j_2}}(\mathbf{x}')$, are:

$$\begin{aligned} g^P_{j_1}(\mathbf{x}') &= x_{j_1} \oplus (1 - x_{j_1}) = 1, \\ g^P_{i_{j_1,j_2}}(\mathbf{x}') &= 0. \end{aligned}$$

Thus, applying the renormalization transformation to the parity function yields zero after two iterations, in contrast to the behavior of a generic Boolean function.

More generally, for any term of the form $T = y_{i_1} y_{i_2} \cdots y_{i_m}$, with $y_i = x_i$ or $1 - x_i$, the quantity $T(x_i = 1) \oplus T(x_i = 0)$ is either zero (if y_i does not occur in T) or else is

[8] This result follows by writing (5) as $p_{\ell+1} - p_\ell = p_\ell - 2p_\ell^2$ and assuming that the variation with ℓ is slow, so that it may be approximated using the differential equation $dp(\ell)/d\ell = p(\ell) - 2p(\ell)^2$, which for $p_0 < 1/2$ has the solution $p(\ell) = 1/[2 + ((1-2p_0)/p_0)\exp(-\ell)] \approx 1/(2 + \exp(-\ell)/p_0)$.

[9] This follows because the inequality $p_{\ell+1} \leq 2p_\ell$ holds even for non-random functions, where p_ℓ is defined as the fraction of input configurations for which the output is one.

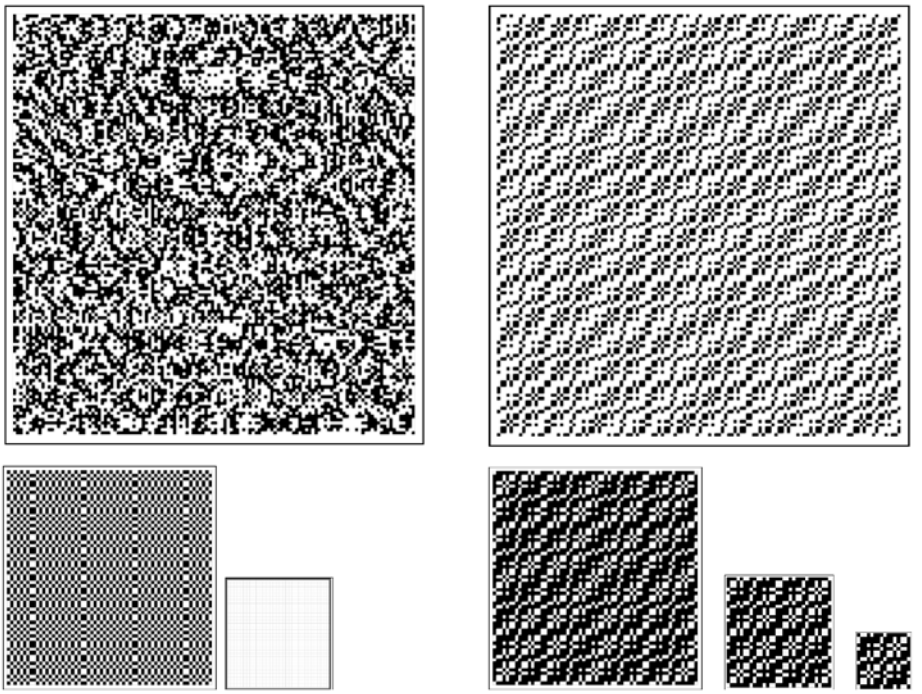

Fig. 2 Results of renormalization group (RG) transformation applied to two functions that are in nongeneric phases. In both panels, the original function (*top*) depends on $N = 14$ variables. Configurations are shown after every other RG step. In the plots, the x-coordinates are the binary expansions of the values of the first $N/2$ variables, and the y-coordinates are the binary expansion of the last $N/2$ variables. *Left*: Initial function is a third order polynomial, $f = a_0 + \sum_i b_i x_i + \sum_{ij} c_{ij} x_i x_j + \sum_{ijk} d_{ijk} x_i x_j x_k$, where the sums are all modulo two, and each coefficient is chosen to be zero or one pseudo-randomly with equal probability. The fixed point is the function in which the output is zero for every input configuration. *Right*: Initial function is one if the sum of the input values is divisible by 3, and zero otherwise; after one iteration a fixed point is reached in which the output value is unity for 2/3 of the input configurations, clearly different than that for a generic function, where the fraction of input configurations yielding nonzero output is very close to 1/2

the product of $m - 1$ instead of m of the y's; for example

$$T(y_{i_1} = 1) \oplus T(y_{i_1} = 0) = y_{i_2} \cdots y_{i_m}. \tag{9}$$

Because the effect of the RG procedure on the sum of terms is equal to the sum of the results of the transformation applied to the individual terms, any function that is the mod-2 sum of terms that are all products of fewer than m y's will yield zero after m renormalizations, for any choice of the decimated variables. It follows immediately that a function that is a polynomial of degree ξ or less has the property that applying the RG transformation to it $\xi + 1$ times yields zero for any choice of the decimated variables. This behavior is illustrated in the left panel of Fig. 2.

Next we note that the sum of a low-order polynomial and a small perturbation (a function that is nonzero on a very small fraction of input configurations) is a non-generic function. This follows because the renormalization group transformation is linear: if a function $f(x_1, \ldots, x_N)$ can be written as the sum of a polynomial of order ξ with $\xi < N/2$ and a

term that is nonzero on a fraction of input configurations that is less than $2^{-\frac{N}{2}(1+\epsilon)}$, then the functions resulting after $N/2$ renormalizations are nonzero on a fraction of inputs that is less than $2^{-\epsilon N}$.

Thus we have demonstrated that the RG transformation distinguishes generic Boolean functions from functions that can be written as polynomials of degree ξ or less, when $\xi < N/2$. Moreover, perturbing one of these functions slightly by adding a component that is nonzero on an exponentially small fraction of input configurations yields a nongeneric function. The qualitatively different behavior upon renormalization of polynomials of degree ξ from generic Boolean functions can be interpreted as evidence that these two classes of functions are in different phases.

We now demonstrate that the RG method also identifies as non-generic functions that depend on a composite quantity such as the arithmetic sum of the variables. Efficiently computable functions with this property include MAJORITY (which is one if more than half the inputs are set to one, and zero otherwise) [38] and DIVISIBILITY MOD p (which is one if the number of inputs that are set to one is divisible by an odd prime p and zero otherwise) [43, 44]. The renormalization group approach distinguishes such functions from generic Boolean functions because the output of all the functions in the sequence is constrained to be identical for very large sets of input configurations. We first show that MAJORITY and DIVISIBILITY MOD p are both distinguished from a generic Boolean function by the renormalization group procedure, and then we argue that the RG procedure distinguishes any function of the arithmetic sum of the inputs from a generic Boolean function. We expect that the argument will be generalizable to apply to a broad class of functions that depend on other composite quantities that are specific combinations of the input variables.

First we consider the behavior when the RG transformation is applied to DIVISIBILITY MOD 3. Since this function is nonzero when the arithmetic sum $\sum_{j=1}^{N} x_j$ is divisible by 3, changing an input x_i changes the output value when the sum of the other input variables is either zero or two. Thus, the renormalized function $g_i(\mathbf{x}')$ is nonzero for any i on a fraction of the input configurations that is very close to $2/3$. Every succeeding renormalization also yields a function that is nonzero when the sum of the remaining variables is either zero or two. This behavior differs from that of a generic Boolean function, in which the renormalized functions are nonzero for a fraction of inputs that is very close to $1/2$. More generally, when the RG is applied to DIVISIBILITY MOD p, with p an odd prime, the behavior of the sequence of functions is determined by the value of the mod p remainder of the undecimated variables. The functions in the sequence yield the output one when the remainder mod p takes on certain values, and typically, after a small number of iterations, these values cycle with a finite period. Therefore, the fraction of input configurations that lead to a nonzero input essentially cycles also (the cycling is not exact only because the fraction of input configurations with a given value of the remainder mod p changes very slightly with N), and, since p is odd, none of the fractions in the cycle is close to $1/2$.

The behavior obtained when the RG procedure is applied to the MAJORITY function is also significantly different from that of a generic Boolean function. The first renormalization step yields a function that is nonzero when the sum of the undecimated variables is $N/2-1$, and the second step yields a function that is nonzero when the sum of the undecimated variables is either $N/2-2$ or $N/2-1$. The functions obtained after j decimations are nonzero on a fraction of inputs that is bounded above by $Cj/\sqrt{N}$, where C is a constant of order unity, so long as $j \ll \sqrt{N}$. The original function is thus identified as non-generic because so long as the number of renormalizations applied is much smaller than $\sqrt{N}$ the renormalized functions are all nonzero on a fraction of input configurations that is much less than $1/2$.

Next we argue that the renormalization group approach distinguishes any function of the arithmetic sum of the inputs from a generic Boolean function. The intuition underlying the argument is that all the functions in the sequence depend only on the arithmetic sum of the undecimated variables, and when the number of undecimated variables is $\mathcal{N}$, the number of configurations of the undecimated variables whose arithmetic sum is constrained to be $\mathcal{S}$, is $\mathcal{N}!/\mathcal{S}!(\mathcal{N}-\mathcal{S})!$. Using Stirling's series [34], one can show explicitly that when N is large, then the number of configurations with a given value of $\mathcal{S}$ is a polynomial in $1/N$ times 2^N for a number of values of $\mathcal{S}$ that grows as the square root of N. Therefore, the *differences* in the fraction of configurations yielding different values of N decay polynomially with N, and the fraction of input configurations yielding one should either be exactly 1/2 or else must deviate from 1/2 by an amount that decreases only polynomially with N.

Many other nongeneric functions exist that are not described in this section, and exhaustive enumeration of all such functions is not likely to be feasible. The possible utility of being able to identify functions as nongeneric using the RG method depends on whether one can relate phases to other methods of classifying Boolean functions. This question is addressed in Sect. 4 below and in the appendices.

3 Renormalization Group Flows within the Generic Phase

In this section we investigate the behavior of a simple subset of the renormalization group flows in the generic phase. We consider the class of functions in which the output value for a given input configuration is chosen independently and randomly to be one with probability p and zero with probability $1-p$. A function at the generic fixed point has $p=1/2$, and yields an output of one on a fraction of input configurations that is $1/2+O(2^{-N/2})$ (reflecting the usual square-root fluctuations for a random process).[10] We ask how many renormalizations of a function with $p \neq 1/2$ are required before the renormalized value of p is indistinguishable from $1/2$.

This question can be addressed using (6), an explicit formula for p_j, the value of p after the jth application of the renormalization transformation. If one is interested in the behavior for large j, which is appropriate here because one is asking when p_j becomes exponentially close to $1/2$, one can write

$$\begin{aligned} p_j &= \frac{1}{2}\left(1-(1-2p_0)^{2^j}\right) \\ &= \frac{1}{2}\left(1-\left(1-\frac{2p_0 2^j}{2^j}\right)^{2^j}\right) \\ &\approx \frac{1}{2}\left(1-e^{-2p_0 2^j}\right). \end{aligned} \tag{10}$$

We wish to find j^* so that $|p_j - 1/2| < \delta$ for all $j > j^*$, where $\delta = O(\sqrt{1/\Omega})$, with $\Omega = 2^N$. We find

$$\begin{aligned} &\frac{1}{2}e^{-2p_0 2^{j^*}} = \delta \\ &\quad\Longrightarrow\quad j^* = \log_2\left(\ln\frac{1}{\delta}\right) - \log_2(p_0). \end{aligned} \tag{11}$$

[10] As shown in the previous section, the value $p=1/2$ is special not only because it is the fixed point value, but also because almost all Boolean functions yield an output of one on a fraction of outputs that is $1/2+O(2^{-N/2})$.

When p_0 is no smaller than N^z for some $z < \infty$, then j^* grows logarithmically with N for any $\delta \geq 1/2^N$. When $p_0 \propto 2^{-N^y}$ for some $y > 0$, then j^* grows as N^y. With our definition that the generic phase consists of functions with $j^* < N/2$, functions with $2^{-N/2} < p_0 < 1 - 2^{-N/2}$ are in the generic phase.

4 Relationships between the Phase of a Function and the Computational Resources Needed to Compute the Function

This section addresses the relationships between the phase of a Boolean function and whether or not the value of the function can be determined with computational resources bounded by a polynomial of N, the number of input variables.

We argue here that there are functions in the generic phase that can be computed with polynomially bounded resources, and that there are non-generic functions that cannot be computed with polynomially bounded resources. Some functions that can be computed efficiently that we expect to be in the generic phase are those of the form

$$\sum_{j=1}^{N} \sum_{k=1}^{k^*(j)} y_{i_1(k)} \cdots y_{i_j(k)}, \tag{12}$$

where the sums are modulo two, each $y_{i(k)}$ is either $x_{i(k)}$ or $1 - x_{i(k)}$, and each $k^*(j)$ grows no faster than polynomially with N. These functions are the sum of polynomially many terms that are each products of j factors, with $j = 1, \ldots, N$. A term with j factors is nonzero on the fraction 2^{-j} of the input configurations, and this fraction can increase by a factor of two at each decimation. The terms with $j \leq \xi$ yield zero after $\xi + 1$ renormalizations, but after $\xi + 1$ renormalizations the terms with $j = \xi + 1 + m$ are nonzero on the fraction 2^{-m} of the inputs, which need not be small. Therefore, for any number of renormalizations up to $N/2$ it is possible for the renormalized functions to yield a nonzero output for very close to half of the input configurations.

Conversely, the number of non-generic functions of N variables is much greater than the number of functions that can be computed with resources bounded polynomially in N, and so some non-generic functions must be associated with problems that are not in P. This can be seen by noting that the number of polynomials of N variables with degree ξ is $2^{\sum_{k=1}^{\xi} N!/(\xi!(N-\xi)!)}$,[11] which, when $\xi \ll N$, $\sim 2^{(Ne/\xi)^{\xi}}$. When ξ scales as a fractional power of N, this is much larger than the number of functions that can be computed with resources bounded by a polynomial of N [40]. Therefore, since we define a phase based on the behavior yielded by repeated renormalization, the functions that correspond to problems in P do not comprise a phase.

The conjectured relevance of phases for yielding insight into complexity classes relies on noting that a product of M variables is nonzero for only a fraction 2^{-M} of the input configurations (for example, the term $x_1 x_2 \cdots x_M$ is nonzero only for input configurations

[11] To obtain the number of polynomials of degree ξ or less, note that each can be written as a sum of terms of the form $x_{i_1} \cdots x_{i_k}$ for all $k \leq \xi$. There are $N!/k!(N-k)!$ ways to choose k indices out of N possibilities, so there are $\sum_{k=1}^{\xi} N!/k!(N-k)!$ different possible terms in the polynomial, each of which occurs with a coefficient of either one or zero. Thus, there are $2^{\sum_{k=1}^{\xi} N!/k!(N-k)!}$ different polynomials of degree ξ or less.

that have $x_1 = x_2 = \cdots = x_M = 1$). The sum of a polynomially large number $t(N)$ of terms that are the product of M variables is nonzero only on a fraction of inputs that is bounded above by $t(N)/2^M$. A typical Boolean function is obtained by adding up a number of terms that grows exponentially with N, each of which is nonzero on a fraction of the inputs that is exponentially small in N. This procedure is not available for functions for which the number of operations that can be used to construct them is bounded above by a polynomial of N. Functions that are constructed in a polynomially bounded number of operations that yield an output that is nonzero on a fraction of input configurations is close to $1/2$ either have some terms with a small number of factors or else involve a delicate balancing of sums and products to keep the fraction of configurations on which the terms are nonzero from becoming too small. It is plausible that if delicate balancing has occurred, the resulting function would be nongeneric. We conjecture that the any Boolean function of N variables $f(x_1, \ldots, x_N)$ that can be computed with polynomially bounded resources can be written as the sum:

$$f(x_1, \ldots, x_N) = \mathcal{N}(x_1, \ldots, x_N) \oplus \mathcal{R}(x_1, \ldots, x_N), \tag{13}$$

where $\mathcal{N}(x_1, \ldots, x_N)$ is a nongeneric function and the remainder term $\mathcal{R}(x_1, \ldots, x_N)$ is nonzero on a fraction of input configurations that is bounded above by $\mathcal{C}2^{-AN^B}$, where $\mathcal{C}$, A, and B are positive constants bounded away from zero. Appendix A presents arguments to support this conjecture, while Appendix B shows that almost all Boolean functions do not obey (13).

Using the RG transformation to identify functions that satisfy (13) is not entirely straightforward—the obvious strategy, renormalizing $\xi + 1$ times and checking whether or not each function in the sequence yields a nonzero output on a fraction of the input configurations that is exponentially close to $1/2$, fails because renormalization yields exponential growth in the fraction of input configurations for which the remainder term is nonzero. For instance, if $\mathcal{N}(x_1, \ldots, x_N)$ is a polynomial of order ξ with $\xi \propto N^y$ with $0 < y < 1$ and $\mathcal{R}(x_1, \ldots, x_N)$ is nonzero on a fraction of input configurations that is of order 2^{-AN^y}, then, because the remainder term grows upon renormalization, all the functions obtained by multiple renormalization steps are nonzero on a substantial fraction of the input configurations. To circumvent the difficulty caused by the growth of the remainder term upon renormalization, one can examine *all* functions that differ from the function in question on a small fraction of input configurations. If the original function obeys (13), then one of the "perturbed" functions will have a remainder term that is zero, and applying the renormalization transformation to it $\xi + 1$ times yields zero for all choices of the decimated variables. A similar procedure can be used to identify other functions that can be written as sums of nongeneric functions and small generic pieces—one can examine all functions that yield the same output for all but a small fraction of the input configurations and determine whether or not one or more of those functions exhibits nongeneric behavior upon renormalization.

5 Application of RG to the Kauffman Net Predecessor Problem

This section discusses Kauffman nets, a specific class of dynamical models that we will use to define a function that we propose as a candidate for investigating fundamental differences between functions that correspond to problems that are in P and functions that correspond to problems that are in NP. A Kauffman net (also called a Boolean network or an N-K model) [2, 25, 29] has N elements $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$, each of which is a Boolean variable $\sigma_i \in \{0, 1\}$,

$i = 1, 2, \ldots, N$. The value of the ith element σ_i at time $t + 1$ is determined by the value of its k inputs $j_1(i), j_2(i), \ldots, j_K(i)$ at time t, $\sigma_{j_1(i)}(t), \sigma_{j_2(i)}(t), \ldots, \sigma_{j_K(i)}(t)$, via

$$\sigma_i(t+1) = f_i(\sigma_{j_1(i)}(t), \sigma_{j_2(i)}(t), \ldots, \sigma_{j_K(i)}(t)), \tag{14}$$

where each f_i is a randomly chosen Boolean function that depends on K arguments. The K inputs for each element and the Boolean functions f_i are all chosen randomly before beginning and then fixed throughout the computation. We will denote N (14) for all the elements as $\{\sigma(t+1)\} = f(\{\sigma(t)\})$.

Kauffman nets have been studied because of their relevance to physics [4–6, 10, 36], social sciences [3, 22], and biology (Kauffman's original motivation was to study gene regulation and control [25–30, 45]). The model exhibits a phase transition as K is varied; $K < 2$ is a "frozen" phase, while $K > 2$ exhibits chaotic dynamics.

Kauffman nets with any $K \leq A \log_2 N$ can be specified using a number of bits that grows as a polynomial of N, and calculation of a successor configuration can be done with polynomially bounded resources. In contrast, when $K \geq 3$, all known algorithms for determining whether a given configuration has a predecessor appear to require computational resources that grow exponentially with N [20]. Our particular interest here is in characterizing the predecessor problem when $K \propto \log N$.

When $K = N$, although specifying the model requires space that grows exponentially with N, one can still ask how many evaluations of the Boolean functions are required to determine whether such a predecessor exists. A candidate solution can be verified with a single evaluation of each Kauffman net function, but because in this case each configuration is a truly random function of its predecessor [15–17], the only way to determine whether a predecessor exists is to check all of the exponentially many candidates [8]. In the parlance of computer science, the Kauffman net with $K = N$ is an oracle relative to which P and NP are not equal [8].

When $K \propto \log N$, the successor function of a Kauffman net is non-generic. This can be seen by noting that each output element only depends on K input elements, so changing an input element must affect fewer than K of the N output bits, and applying K renormalizations to the successor function must yield zero. Conversely, since the predecessor problem is nonlocal, it is extremely plausible that whether or not changing one element of the target configuration affects whether a predecessor exists depends on the values of a large number of other elements in the successor [13]. However, it is not known whether the function that is one if a given configuration has a predecessor and zero if not is in the generic phase, and if so, how far from a non-generic phase boundary it is.

The predecessor function is very unlikely to have an output value of one for a fraction of input configurations that is very close to $1/2$, but it is extremely plausible that this fraction is no smaller than 2^{-N^z} for some $z < 1$. When $K = N$, as $N \to \infty$ the probability that a configuration has no predecessor approaches $1/e$; this follows because the successor to each of the 2^N configurations is chosen independently and randomly, so that the probability that a configuration is not the successor to any input configuration is $(1 - 1/2^N)^{2^N}$, which approaches $1/e$ as $N \to \infty$ [8]. When K is finite, the fraction of configurations with predecessors decreases exponentially with N, because the probability that a randomly chosen configuration has a predecessor is bounded above by the value $(1 - 2^{-(K+1)})^N$. This bound follows because with probability 2^{-K} the function determining the value of any single element is independent of the values of all its inputs, and for such a function, with probability $1/2$ the output value will be inconsistent with the target [14]. For fixed K this upper bound on the probability that a randomly chosen configuration has a predecessor vanishes as $N \to \infty$, while when $K = A \log_2(N)$, this bound approaches $1 - \frac{1}{2} N^{-(A-1)}$, which

approaches unity as $N \to \infty$ when $A > 1$. Thus, when $K \propto \log N$, it is plausible that as $N \to \infty$ the fraction of configurations that have predecessors is either nonzero or else decays more slowly than exponentially in N. This condition is necessary but not sufficient for the predecessor function to be in the generic phase and also far from a non-generic phase boundary. One must also show that performing $N/2$ RG iterations yields a function that is nonzero on a fraction of input configurations that differs from $1/2$ by an amount that is exponentially small in N, which essentially requires that correlations between the function values for different input configurations become negligible as the RG transformation is iterated. It is plausible that the extreme sensitivity of the Kauffman net predecessor problem to small changes in the problem statement [13] could be a useful property to exploit in demonstrating that correlations between the function values for different input configurations can be neglected, but demonstrating this is a challenging outstanding problem.

6 Discussion

This paper presents a renormalization group approach that distinguishes generic Boolean functions of N variables from non-generic functions, examples of which include those that can be written as a polynomial of degree ξ, with $\xi \ll N$, and also those that depend only on composite quantities such as the arithmetic sum of all the input variables. The method provides a consistent framework for identifying many different functions as non-generic—one examines whether every function in the sequence of renormalized functions yields an output of one on a fraction of input configurations that differs from $1/2$ by an amount that is exponentially small in N. The identification of phases of Boolean functions is useful because it provides a method for determining whether an individual function has properties characteristic of typical randomly chosen functions.

The procedure used here of using the behavior yielded by a renormalization group transformation to identify different phases of Boolean functions is entirely analogous to a procedure presented by Wilson [51] to identify different thermodynamic phases of the Ising model, used to describe magnetism in solids. Wilson showed that individual configurations of Ising models could be identified as being in either a ferromagnetic phase or paramagnetic phase by repeatedly eliminating spins and examining the resulting configurations—if after many renormalizations all the spins are aligned, then the system is in the ferromagnetic phase, while if after many renormalizations the spin orientations are random, then the system is in the paramagnetic phase.

The relationship between phases and computational complexity classes is not simple. Functions computable with polynomially bounded resources do not comprise a phase—there are functions that are in a non-generic phase that cannot be computed with polynomially bounded resources, and there are functions that can be computed with polynomially bounded resources that are in the generic phase. The possibility of using the RG approach to demonstrate that a given Boolean function arises from a problem that is not in P arises from the conjecture that the functions that can be computed with polynomially bounded resources that are in the generic phase are all close to a phase boundary of a non-generic phase. If this conjecture holds, then this work provides a natural framework for understanding why the P versus NP question is so difficult: distinguishing computational complexity classes involves finding the quantitative location of functions in the phase diagram, a property that is not robust upon renormalization.

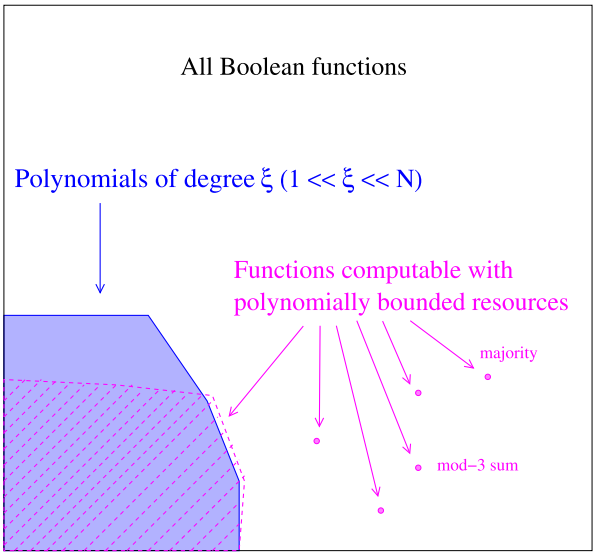

Fig. 3 (Color online) Schematic phase diagram for Boolean functions. Within the set of all Boolean functions of N Boolean variables as $N \to \infty$, there is a generic phase, and there are also many non-generic phases, two of which are functions that can be written as polynomials of order no greater than ξ with $\xi \ll N$, and functions of composite variables such as the arithmetic sum of all the inputs. As discussed in Appendix B, almost all Boolean functions are generic and also far from any non-generic phase. Though P does not correspond to a phase, Appendix A argues for the conjecture that all functions that arise from problems that are in P are either non-generic or else very close to a phase boundary of a non-generic phase

Based on the analogy between RG results for magnets and the qualitatively different behavior of the renormalization group flows for generic Boolean functions and for nongeneric functions such as low-degree polynomials and functions of composite variables, we propose the schematic phase diagram for Boolean functions shown in Fig. 3.

If this conjecture holds, then the procedure described here leads to a specific algorithmic approach to the P versus NP question—if a given function that is obtained as the answer to a problem in NP fails to be close enough to a non-generic phase, then one has shown that P is not equal to NP. Section 5 presents a specific family of candidate functions that may be useful for implementing the strategy proposed in this paper (see also [13]), but the strategy can be implemented for any candidate function. Appendix A argues that the construction of a function that can be computed with polynomially bounded resources that does not satisfy (13) requires delicate balancing that leads to nongenericity (such as the existence of a composite variable), but the argument is only speculative. Progress on this issue is the key to using the RG approach to be able to address the P versus NP question. Appendix B shows that almost all Boolean functions are far from all low-order polynomials and also from all functions that can be computed with polynomially bounded resources.

The strategy discussed in Sect. 4 for using the renormalization group approach to show that a function cannot be computed with polynomially bounded resources requires determining not only that it is not in a non-generic phase but also that it is not near a phase boundary, a task that appears to require resources that grow faster than exponentially with N. This superexponential scaling means that the procedure proposed here cannot be used to break pseudorandom number generators, a difficulty that would arise if the procedure were a "nat-

ural proof" that could be implemented with resources that scale no faster than exponentially with N [39]. However, direct numerical implementation of the procedure is not likely to be computationally feasible.

These work leaves many unanswered questions. One important one is, of course, whether the conjecture that all functions that can be computed with polynomially bounded resources are close to a non-generic phase boundary is valid. Another is whether the Kauffman net predecessor problem is in the generic phase and far from any generic phase boundary (that is, whether a number of renormalizations of order $\log(N)$ yields a series of functions that all yield an output of one on a fraction of input configurations that differs from $1/2$ by an amount that decays exponentially with N). But many other questions not aimed at understanding computational complexity classes are also unanswered at this stage. We do not know how to estimate of the number of functions that are in non-generic phases and therefore have not shown that almost all functions are generic (though this is extremely reasonable intuitively). Characterizing the relevant and irrelevant operators of the RG would also be extremely useful and important, for it would provide a method for, e.g., demonstrating that certain correlations between function values for different input configurations were irrelevant operators and hence would disappear at the RG fixed point.

7 Conclusions

This paper presents a renormalization group approach that can be applied to Boolean functions of N Boolean variables. It can be used to identify a generic phase of Boolean functions and distinguish functions in this phase from nongeneric functions such as polynomials of degree ξ with $\xi \ll N$ and functions that depends only on a composite variable such as the arithmetic sum of the values of the individual inputs. It is shown that phases do not correspond to computational complexity classes, but understanding the relationships between generic and non-generic phases may yield new insight into the distinctions between computational complexity classes.

Acknowledgements The author is grateful to Prof. Daniel Spielman for pointing out an error in a previous version of the manuscript, and for support from NSF grants CCF 0523680 and DMR 0209630.

Appendix A: Characterization of the Functions that Can Be Constructed with a Polynomially Large Number of Operations

In this appendix we examine the properties of functions that can be computed with polynomially bounded resources, and we present arguments to support the conjecture that all such functions can written in the form (13), which is the sum of a nongeneric function plus a correction term that is nonzero on a fraction of input configurations that is less than $\mathcal{C}2^{-AN^B}$ for positive $\mathcal{C}$, A, and B bounded away from zero. To motivate this conjecture, we first discuss the intuitive motivation for thinking that functions that can be computed with polynomially bounded resources are all close to non-generic functions.

Any Boolean function can be written as the result of a computation performed by a set of AND and XOR (exclusive-or) gates [47]. There is a significant difference in computational power of a set of gates of a given size depending on whether or not the output of the gates can be used as the input for more than one other gate (in other words, whether or not the gates have fan-out that is greater than or equal to one). For example, it has been proven that

computing DIVISIBILITY MOD 3 requires exponentially many gates if the gate fan-out is one but polynomially many gates if the gate fan-out is two or more [43, 44, 47]. Problems that are in P give rise to functions that can be computed using a set of gates whose size scales as a polynomial of N, the size of the input, when the gates have fanout of two or more. Using the output of some gates as inputs of other gates can reduce the number of gates required to perform a given computation from exponential to polynomial in N only if some outputs are reused many times,[12] meaning that the same composite quantity were to enter into many different intermediate results in the calculation. This observation leads to the conjecture that a function that can be computed with polynomially many gates either is close to a low-order polynomial or else contains a dependence on particular fixed combinations of variables that enter a large number of times and hence lead to non-generic behavior.

To argue further why it is plausible that all functions that can be computed with resources bounded by a polynomial of N satisfy (13), we consider the process by which functions can be constructed. We first examine functions that can be computed with polynomially bounded resources that are close to low-order polynomials, and then examine known functions that can be computed with polynomially bounded resources that are not approximable by low-order polynomials, arguing that their construction involves delicate balancing that may give rise to nongeneric properties, one example being the emergence of a composite variable on which the function depends.

First we show that a starting polynomial that is the sum of polynomially many terms whose factors are all either x_i or $(1 - x_i)$ satisfies (13). Then we show that the sum of two functions that each obey (13) also satisfies (13), and also that the coefficient multiplying the correction term grows sufficiently slowly that the bound remains true even after a number of additions that grows polynomially with N. We then consider products of such functions. The behavior is more complicated, but we argue that a similar decomposition works because one of two things will happen: either terms with a large number of products will be nonzero on a small fraction of input configurations, or else special balancing of terms will have taken place, which plausibly gives rise to nongeneric behavior.

First consider a polynomial $A(x_1, \ldots, x_N)$ that is the mod-2 sum of polynomially many terms that are all of the form $y_{i_1} \cdots y_{i_m}$, where y_i is either x_i or $1 - x_i$:

$$A(x_1, \ldots, x_N) = C_0 + \sum_{\eta=1}^{N} \sum_{k_\eta=1}^{M_\eta} y_{i_1(\eta,k_\eta)} \ldots y_{i_\eta(\eta,k_\eta)}. \tag{15}$$

Here, C_0 is a constant, η denotes the number of factors of y_i in a term, k_η is the index labeling the different terms with η factors, $i_j(\eta, k_\eta)$ denotes the index of the jth factor in the term k_η, and each M_η, the number of terms with η factors, is bounded above by a polynomial of N. We will obtain bounds on the number of configurations for which the remainder is nonzero by considering standard addition instead of modulo-two addition, which means that we will overcount by including configurations for which an even number of terms in the polynomial expansion are nonzero. Each term with η factors is nonzero only on a fraction $2^{-\eta}$ of the inputs. Therefore, if we define $\rho_A(\eta)$ to be the fraction of inputs of $A(x_1, \ldots, x_N)$ for which the sum of all the terms with η factors is nonzero, we have

$$\rho_A(\eta) \leq C_A 2^{-\alpha\eta}, \tag{16}$$

[12] A fan-out of two is sufficient for generating the same output a large number of times because one of the outputs could be the input of a gate that takes the AND of that input and one, yielding two more outputs with the same value.

for constant C_A and $\alpha = \frac{1}{2} - \epsilon$, with ϵ infinitesimal. If one chooses $\eta \propto N^B$ for some $B > 0$, then (13) holds.

Now consider the addition of two functions $F_1(x_1, \ldots, x_N)$ and $F_2(x_1, \ldots, x_N)$ with remainder terms $\mathcal{R}_1$ and $\mathcal{R}_\in$ that are each less than 2^{N^B} for some $B > 0$. Again we use standard addition instead of modulo-two addition, and note that because the sum $\mathcal{R}_S(x_1, \ldots, x_N) = \mathcal{R}_1(x_1, \ldots, x_N) + \mathcal{R}_2(x_1, \ldots, x_N)$ is nonzero only if one of the summands is, we have that the fraction of inputs on which the remainder of the sum is nonzero, $\rho_{\mathcal{R}_S}$, satisfies

$$\rho_{\mathcal{R}_S} \leq \rho_{\mathcal{R}_1} + \rho_{\mathcal{R}_2}, \tag{17}$$

and the sum also obeys (13). Adding polynomially many terms can increase the prefactor only by an amount that grows no faster than polynomially in N, so the remainder stays exponentially small in N^B.

We next consider the product of two functions that satisfy (16). We write

$$\begin{aligned} A(\mathbf{x}) &= \mathcal{N}_A(\mathbf{x}) + \mathcal{R}_A(\mathbf{x}), \\ B(\mathbf{x}) &= \mathcal{N}_B(\mathbf{x}) + \mathcal{R}_B(\mathbf{x}), \end{aligned} \tag{18}$$

where $\mathcal{N}_A$ and $\mathcal{N}_B$ are nongeneric, and $\mathcal{R}_A(\mathbf{x})$ and $\mathcal{R}_B(\mathbf{x})$ are both nonzero on a fraction of inputs that is less than 2^{-AN^B} for some positive y bounded away from zero.

We write the product of $A(\mathbf{x})$ and $B(\mathbf{x})$ as

$$\begin{aligned} D(\mathbf{x}) &= A(\mathbf{x})B(\mathbf{x}) \\ &= (\mathcal{N}_A(\mathbf{x}) + \mathcal{R}_A(\mathbf{x}))(\mathcal{N}_B(\mathbf{x}) + \mathcal{R}_B(\mathbf{x})) \\ &= \mathcal{N}_A(\mathbf{x})\mathcal{N}_B(\mathbf{x}) + \mathcal{N}_A(\mathbf{x})\mathcal{R}_B(\mathbf{x}) + \mathcal{R}_A(\mathbf{x})\mathcal{N}_B(\mathbf{x}) + \mathcal{R}_A(\mathbf{x})\mathcal{R}_B(\mathbf{x}). \end{aligned} \tag{19}$$

Now $\mathcal{N}_A(\mathbf{x})\mathcal{R}_B(\mathbf{x})$ can only be nonzero for an input configuration if $\mathcal{R}_B(\mathbf{x})$ is (this follows since a product is nonzero only if each of its factors is nonzero), and, similarly, $\mathcal{R}_A(\mathbf{x})\mathcal{R}_B(\mathbf{x})$ can only be nonzero for an input configuration if both $\mathcal{R}_A(\mathbf{x})$ and $\mathcal{R}_B(\mathbf{x})$ are, so the sum of the last three terms must be nonzero on a fraction of inputs that is less than $\rho_{\mathcal{R}_A} + \rho_{\mathcal{R}_B}$. Therefore, these contributions to the remainder term in the product remain exponentially small in N^B even after polynomially many multiplications. Therefore, it only remains to consider the properties of the product of the nongeneric functions $\mathcal{N}_A(\mathbf{x})\mathcal{N}_B(\mathbf{x})$. That the product of nongeneric functions need not be nongeneric can be seen by considering the case where the $\mathcal{N}_\alpha(\mathbf{x})$ are low-order polynomials (the product of $O(N)$ low-order polynomials can be a polynomial of order N). So for now let us assume that $\mathcal{N}_A(\mathbf{x})$ and $\mathcal{N}_B(\mathbf{x})$ are polynomials of degree $\xi = N^B$, and then write

$$\mathcal{N}_A(\mathbf{x})\mathcal{N}_B(\mathbf{x}) = P_D^\xi(\mathbf{x}) + R_D^\xi(\mathbf{x}), \tag{20}$$

where $P_D^\xi(\mathbf{x})$ is a polynomial of degree ξ and $R_D^\xi(\mathbf{x})$ is a remainder term that we would like to bound.

To bound the magnitude of the remainder, let us multiply out the polynomials in (20) so that they are all sums of terms that are products of the form $y_{i_1} \cdots y_{i_j}$, terms that we will denote as "primitive." Let T_A be the number of primitive terms in $P_A^\xi(\mathbf{x})$, and T_B be the number of primitive terms in $P_B^\xi(\mathbf{x})$. Note that every primitive term in the product with more than ξ factors is nonzero on a fraction $2^{-\xi}$ or less of the input configurations.

Since the total number of primitive terms in $R_D^\xi(\mathbf{x})$ is bounded above by $T_A T_B$, the fraction of inputs on which the sum of the terms with at least ξ factors is nonzero is bounded

above by $T_A T_B 2^{-\xi}$. So long as T_A and T_B are both less than exponentially large in ξ, then this remainder term is exponentially small in ξ. The multiplication process must start with values of T_A and T_B that are both bounded by a polynomial of N, but because multiplications can be composed, we need to examine the behavior of T_D, the number of primitive terms in $P_D^{\xi}(\mathbf{x})$.

A simple upper bound for T_D is obtained by ignoring all possible simplifications that could reduce the total number of terms in the product:

$$T_D \leq T_A T_B. \tag{21}$$

This equation describes geometric growth. If $\mathcal{M}$ polynomials are multiplied together, all of which have fewer than CN^Y terms for fixed C and Y, then the total number of terms in the product, $T_{\mathcal{M}}$, satisfies the bound

$$T_{\mathcal{M}} \leq (CN^Y)^{\mathcal{M}}. \tag{22}$$

This bound on the number of terms in the product is much smaller than 2^{ξ} so long as $\mathcal{M}$ satisfies

$$\mathcal{M} \ll \xi/(Y \log_2 N + \log_2 C). \tag{23}$$

A useful bound on multiplicative terms that are products of more than $\xi/(Y \log_2 N)$ factors can be obtained by exploiting the fact that the product of two functions is nonzero for a given input only if each of the factors is. Specifically, consider the product AB, and say that A is nonzero on a set of M_A inputs. If B is nonzero on less than a fraction σ of the inputs in this set for some $1/2 < \sigma < 1$, then the product AB is nonzero on fewer than σM_A inputs, and if not, then the product $A(1-B)$ is nonzero on fewer than $(1-\sigma)M_A$ inputs, and one can write $AB = A + A(1-B)$.[13]

The result of $\mathcal{M}$ multiplications is then nonzero only on a fraction of inputs bounded above by $2^{-\mathcal{M}\log_2\sigma}$. Therefore, a product of more than $\xi/Y\log_2(N)$ factors is nonzero on no more than a fraction $2^{-\tilde{C}\xi/\log_2(N)}$ of the inputs, where $\tilde{C}$ is a positive constant, and the entire product can be moved into the remainder term.

The arguments above indicate that the remainder term tends to be small for products because the number of terms in the polynomial that are of order ξ or less can be bounded for products of small numbers of terms, and products of many terms are nonzero on a small enough fraction of the input configurations that they can be considered to be part of the remainder term. However, there are functions that can be computed with polynomially bounded resources that do not obey (13). Two examples of functions that can be computed efficiently that have been proven to violate (13) are MAJORITY (which is one when more than half input variables have been set to one and zero otherwise) [38] and DIVISIBILITY MOD p, which is one if the sum of the input variables is divisible by an odd prime p [43, 44]. However, there are doubtless many others. The reason for considering these simple examples is that it is instructive to consider algorithms for computing these functions to see how they "avoid" being close to low-order polynomials.

[13] One might worry that products of the form $A_1 A_2 \cdots A_M$, where each A_i is nonzero on more than half of the inputs, and M is of order N, might pose a problem, for if one writes $A_1 A_2 \cdots A_M = (1 - A_1')(1 - A_2') \cdots (1 - A_M')$, then the number of terms with m factors is $M!/(M-m)!m!$, which can be as large as $2^{M/2}$ (when $m = M/2$). This term proliferation is not a problem if one chooses σ to be strictly greater than $1/2$ (say, $3/4$), since the number of terms with a given number of terms in the product is overwhelmed by the decrease in the fraction of inputs for which each individual term is nonzero.

Some pseudocode for a simple algorithm for solving DIVISIBILITY mod 3 is:

divisibility mod 3:

start: remainder0[0] = 1, remainder1[0] = remainder2[0] = 0

for each $i > 0$

remainder0$[i+1]$ = remainder0$[i] * (1 - x_{i+1}) \oplus$ remainder2$[i] * x_{i+1}$

remainder1$[i+1]$ = remainder1$[i] * (1 - x_{i+1}) \oplus$ remainder0$[i] * x_{i+1}$

remainder2$[i+1]$ = remainder2$[i] * (1 - x_{i+1}) \oplus$ remainder1$[i] * x_{i+1}$

answer = remainder0$[N]$

The quantity remainder0$[i]$ + remainder1$[i]$ + remainder2$[i]$ is unity for every i, and the fraction of inputs for which each remainder variable is nonzero is very close to $1/3$ and does not decay exponentially with i. The fractions do not decay or grow because the equation for each remainder for a given i is the sum of two products. The product remainder0$[i](1 - x_{i+1})$ is nonzero on half the inputs on which remainder0$[i]$ is nonzero, and similarly for the other term remainder2$[i] * x_{i+1}$. Because remainder0$[i+1]$ is the sum of two terms, each of which is nonzero on almost exactly half the outputs for which remainder0$[i]$ is nonzero, remainder0$[j]$ remains of order of but less than unity for all j. It is plausible that this repeated exquisite cancellation in the algorithm is necessary for the products of many terms to remain of order unity, and underlies the non-generic behavior upon renormalization. Because constructing a function using polynomially many operations that cannot be written as a low-order polynomial plus a term that is nonzero except for a small fraction of input configurations requires a series of delicate cancellations, it is also extremely plausible that the fraction of functions that do not satisfy (13) that can be computed with polynomially bounded resources is extremely small.

To summarize, in this appendix we discuss the restrictions on Boolean functions of N variables that can be computed with resources that are bounded above by a polynomial in N. We present arguments to support the conjecture that all functions in P can be written as the sum of a nongeneric function and a remainder term that is nonzero on a fraction of input configurations that is bounded above by 2^{-N^y} for a positive value of y that is bounded away from zero.

Appendix B: Demonstration that Almost All Boolean Functions Differ from Functions that Can Be Computed with Polynomially Bounded Resources on a Substantial Fraction of All Input Configurations

In this appendix it is shown almost all Boolean functions are far from every function that can be computed with polynomially bounded resources. This is done by bounding from above the number of Boolean functions that differ from a function that can be computed with polynomially bounded resources on a fraction ϵ of the input configurations, and showing that when ϵ is small, it is much less than the number of Boolean functions of N variables.

The number of functions that differ from all functions that can be computed with polynomially bounded resources on a fraction ϵ of the input configurations is bounded above by the product of an upper bound to the number of functions that can be computed with polynomially bounded resources and an upper bound to the number of functions whose output

differs from that of a given Boolean function on a fraction ϵ of the input configurations. The number of functions that can be computed with polynomially bounded resources, $\mathcal{N}_P$, is bounded above by $\mathcal{N}_P < \mathcal{P}_1(N)^{\mathcal{P}_2(N)}$, where $\mathcal{P}_1(N)$ and $\mathcal{P}_2(N)$ are polynomials in N [41]. This follows by considering a Boolean circuit of AND and OR gates, and noting that the input to each of the polynomially many gates can be either a constant or the output of one of the other gates. This upper bound can also be written $\mathcal{N}_P < 2^{\mathcal{P}'(N)}$.

The number of Boolean functions of N variables that differ from a reference function on a fraction of input configurations that is less than ϵ, $\mathcal{N}_N(\epsilon)$, is

$$\mathcal{N}_N(\epsilon) = \sum_{j=0}^{j=\Omega\epsilon} \binom{\Omega}{j}, \tag{24}$$

where $\Omega = 2^N$ is the number of different input configurations and $\binom{\Omega}{j}$ is the number of ways to choose j items out of Ω possibilities. We have, when $\epsilon < 1/2$,

$$\begin{aligned} \mathcal{N}_N(\epsilon) &= \sum_{j=1}^{\Omega\epsilon} \frac{\Omega!}{j!(\Omega - j)!} \\ &\leq (\Omega\epsilon)\frac{\Omega!}{(\Omega\epsilon)!(\Omega - \Omega\epsilon)!}. \end{aligned} \tag{25}$$

When $1 \ll \Omega\epsilon \ll \Omega$, we have

$$\begin{aligned} \mathcal{N}_N(\epsilon) &\sim (\Omega\epsilon)\left(\frac{\Omega e}{\Omega\epsilon}\right)^{\Omega\epsilon} \\ &= (\Omega\epsilon)(e/\epsilon)^{\epsilon\Omega} \\ &= 2^{2^N \epsilon \log_2(e/\epsilon)}. \end{aligned} \tag{26}$$

Thus we have that the fraction of Boolean functions that differ from a function that can be computed with polynomially bounded resources on a fraction of the input configurations that is no greater than ϵ is

$$\frac{2^{\mathcal{P}'(N)} 2^{2^N \epsilon \log_2(e/\epsilon)}}{2^{2^N}} = 2^{\mathcal{P}'(N)} 2^{-2^N(1-\epsilon\log_2(e/\epsilon))}, \tag{27}$$

which vanishes extremely quickly as $N \to \infty$ when $\epsilon \ll 1$.

An analogous argument demonstrates that a negligible fraction of Boolean functions differ from polynomials of order ξ with $\xi \ll N$ on a small fraction of input configurations. All polynomials of degree ξ or less can be written as a sum over all terms that are products of the form $x_{i_1} \cdots x_{i_j}$ with $j \leq \xi$. There are $\sum_{j=1}^{\xi} N!/[j!(N-j)!]$ such terms, and each coefficient can be either 1 or 0. Thus, when $1 \ll \xi \ll N$, the total number of polynomials of degree up to ξ is bounded above by $\xi(Ne/\xi)^{\xi}$, and the fraction of Boolean functions that differ from a polynomial of degree ξ or less on a fraction of the input configurations that is no greater than epsilon is bounded above by

$$\left(2^{e(N/\xi)^{\xi}}\right)\left(2^{-2^N(1-\epsilon\log_2(e/\epsilon))}\right), \tag{28}$$

which tends to zero as a double exponential of N as $N \to \infty$ for any $\epsilon \ll 1$.

A second non-rigorous but informative argument to see that almost all Boolean functions do not satisfy (13) is to consider a generic Boolean function in which each coefficient $A_{i_1,\ldots,i_N}$ is chosen independently and randomly to be 1 or 0 with equal probability. For such a function, one can always find a configuration satisfying (13) by changing just about $2^N/2$ of the output values so that the function has the same value for all inputs. The question is whether one can obtain $g_{x_{i_1},\ldots,x_{i_M}}(\mathbf{x}') = 0$ for all choices of the M decimated variables by changing the function for many fewer configurations than that. For a given g in which M variables have been decimated, one can find a configuration satisfying $g_{x_{i_1},\ldots,x_{i_M}}(\mathbf{x}') = 0$ for the 2^{N-M} different possible $\mathbf{x}'$ by changing the output for just about 2^{N-M-1} different input configurations. But one must arrange for $g_{x_{j_1},\ldots,x_{j_M}}(\mathbf{x}')$ to vanish for all possible choices of the M variables to be decimated. So the outputs for 2^{N-M-1} input configurations need to be changed for each of the $N!/[M!(N-M)!]$ different ways to choose the decimated variables. Assuming $M \ll N$, this yields a naive estimate that one must change the output value for $2^{N-M-1+M\log_2(Ne/M)}$ different input configurations, which exceeds 2^N for all $M \ll N$. This argument is useful because it makes it clear why one must examine all choices of the decimated variables to identify functions that differ from non-generic functions on a small fraction of input configurations.

References

1. Aaronson, S.: Is P versus NP formally independent? Bull. Eur. Assoc. Theor. Comput. Sci. **81**, 109 (2003)
2. Aldana, M., Coppersmith, S., Kadanoff, L.: Boolean dynamics with random couplings. In: Kaplan, E., Marsden, J., Sreenivasan, K. (eds.) Perspectives and Problems in Nonlinear Science. Springer Applied Mathematical Sciences Series. Springer, Berlin (2003)
3. Axelrod, R.: Advancing the art of simulation in the social sciences. In: Conte, R., Hegselmann, R., Terna, P. (eds.) Simulating Social Phenomena, pp. 21–40. Springer, Berlin (1997)
4. Bastolla, U., Parisi, G.: Closing probabilities in the Kauffman model: an annealed computation. Physica D **98**, 1–25 (1996)
5. Bastolla, U., Parisi, G.: The modular structure of Kauffman networks. Physica D **115**, 203–218, 219–233 (1998)
6. Bastolla, U., Parisi, G.: Relevant elements, magnetization and dynamical properties in Kauffman networks: a numerical study. Physica D **115**, 203–218 (1998)
7. Bennett, C.: Logical depth and physical complexity. Oxford University Press, London (1988), pp. 227–257
8. Bennett, C., Gill, J.: Relative to a random oracle A, $\mathrm{P}^{\mathrm{A}} \neq \mathrm{NP}^{\mathrm{A}} \neq \text{co-}\mathrm{NP}^{\mathrm{A}}$ with probability 1. SIAM J. Comput. **10**, 96–113 (1981)
9. Bhatt, R., Lee, P.: Scaling studies of highly disordered spin-1/2 antiferromagnetic systems. Phys. Rev. Lett. **48**, 344–347 (1982)
10. Bhattacharjya, A., Liang, S.: Power-law distributions in some random Boolean networks. Phys. Rev. Lett. **77**, 1644–1647 (1996)
11. Boppana, R., Sipser, M.: The complexity of finite functions. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, pp. 759–804. Elsevier, Amsterdam (1990)
12. Cook, S.: The complexity of theorem proving procedures. In: Third Annual ACM Symposium on the Theory of Computing, pp. 151–158. Assoc. Comput. Mach., New York (1971)
13. Coppersmith, S.: Complexity of the predecessor problem in Kauffman networks. Phys. Rev. E **75**, 51108 (2007)
14. Coppersmith, S., Kadanoff, L.P., Zhang, Z.: Reversible Boolean networks I: Distribution of cycle lengths. Physica D **149**, 11–29 (2001)
15. Derrida, B.: Random-energy model: An exactly solvable model of disordered systems. Phys. Rev. B **24**, 2613–2626 (1981)
16. Derrida, B., Flyvbjerg, H.: The random map model: a disordered model with deterministic dynamics. J. Phys. **48**, 971–978 (1987)
17. Derrida, B., Weisbuch, G.: Evolution of overlaps between configuration in random networks. J. Phys. (Paris) **47**, 1297–1303 (1986)

18. Furst, M., Saxe, J., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Syst. Theory **17**, 13–27 (1984)
19. Goldenfeld, N.: Lectures on Phase Transitions and the Renormalization Group. Perseus, New York (1992)
20. Goldreich, O.: Candidate one-way functions based on expander graphs (2000). Cryptology ePrint Archive, Report 200/063, available at http://www.wisdom/weizmann.ac.il/~oded/ow-candid.html
21. Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Proc. 18th ACM Symp. on Theory of Computing, pp. 6–20 (1986)
22. Hurford, J.: Random Boolean nets and features of language. IEEE Trans. Evol. Comput. **5**, 111–116 (2001)
23. Institute, C.M.: Clay Mathematics Institute Millennium Problems. http://www.claymath.org/millenium/
24. Kadanoff, L.: Scaling laws for Ising models near T_c. Physics **2**, 263–272 (1966)
25. Kauffman, S.: Homeostasis and differentiation in random genetic control networks. Nature **244**, 177–178 (1969)
26. Kauffman, S.: Metabolic stability and epigenesis in random constructed genetic nets. J. Theor. Biol. **22**, 437–467 (1969)
27. Kauffman, S.: The large scale structure and dynamics of genetic control circuits: An ensemble approach. J. Theor. Biol. **44**, 167–190 (1974)
28. Kauffman, S.: Emergent properties in random complex automata. Physica D **10**, 145–156 (1984)
29. Kauffman, S.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, London (1993)
30. Kauffman, S.: At Home in the Universe: The Search for Laws of Self-Organization and Complexity. Oxford University Press, London (1995)
31. Levin, L.: Universal sorting problems. Probl. Pered. Inf. **9**, 115–116 (1973) (in Russian). English translation: Universal search problems, in: Trakhtenbrot, B.A.: A survey of Russian approaches to Perebor (Brute-Force searches) algorithms. Ann. Hist. Comput. **6**(4), 384–400 (1984)
32. Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, New York (1997)
33. Ma, S.K., Dasgupta, C., Hu, C.K.: Random antiferromagnetic chain. Phys. Rev. Lett. **43**, 1434–1437 (1979)
34. Marsaglia, G., Marsaglia, J.: A new derivation of Stirling's approximation to $n!$. Am. Math. Mon. **97**, 826–829 (1990)
35. Papadimitriou, C.: Computational Complexity. Addison-Wesley, Reading (1994)
36. Qu, X., Aldana, M., Kadanoff, L.: Numerical and theoretical studies of noise effects in the Kauffman model. J. Stat. Phys. **109**, 967–985 (2002)
37. Raz, R.: Lecture notes on arithmetic circuits. http://www.cs.mcgill.ca/~denis/notes05.ps
38. Razborov, A.: Lower bounds on the size of bounded-depth networks over a complete basis with logical addition. Math. Notes Acad. Sci. USSR **41**, 333–338 (1987)
39. Razborov, A., Rudich, S.: Natural proofs. In: Proc. 26th ACM Symp. on Theory of Computing, pp. 204–213 (1994)
40. Riordan, J., Shannon, C.: The number of two-terminal series-parallel networks. J. Math. Phys. **21**, 83–93 (1942)
41. Shannon, C.: The synthesis of two-terminal switching circuits. Bell Syst. Tech. J. **28**, 59–98 (1949)
42. Sipser, M.: The history and status of the P versus NP question. In: Proceedings of ACM STOC'92, pp. 603–618 (1992)
43. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: STOC'87: Proc. of 19th STOC, pp. 77–82 (1987)
44. Smolensky, R.: On representations by low-degree polynomials. IEEE, pp. 130–138 (1993)
45. Socolar, J., Kauffman, S.: Scaling in ordered and critical random Boolean networks. Phys. Rev. Lett. **90**, 068702 (2003)
46. Vollmer, H.: Introduction to Circuit Complexity: A Uniform Approach. Springer, New York (1999)
47. Wegener, I.: The Complexity of Boolean Functions. Wiley, New York (1987). http://citeseer.ist.psu.edu/694854.html
48. White, S.: Density matrix formulation for quantum renormalization groups. Phys. Rev. Lett. **69**, 2863–2866 (1992)
49. Wigderson, A.: P, NP and mathematics—a computational complexity perspective. In: Proceedings of the ICM 06 (2006). http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W06/W06.pdf
50. Wilson, K.: The renormalization group and critical phenomena I: Renormalization group and the Kadanoff scaling picture. Phys. Rev. B **4**, 3174–3183 (1971)
51. Wilson, K.: Problems in physics with many scales of length. Sci. Am. **241**, 158–179 (1979)
52. Yao, A.: Separating the polynomial-time hierarchy by oracles. In: Proc. 26th IEEE Symposium on Foundations of Computer Science, pp. 1–10 (1985)