Once the specific model and information that the recipient has about the source enters into an evaluation of the information transfer, there is a certain and quite reasonable degree of relativity in the amount of information transferred. An extreme example would be if the recipient has already received a long message and knows the same message is being repeated,then no new information is being transmitted.A person who has memorized the Gettysburg Address will receive very little new information upon hearing or reading it again. The prior knowledge is part of the model possessed by the recipient about the source.

Can we incorporate this in our definition of information? In every case where we have measured the information of a message, we have made use of a model of the source of the information. The underlying assumption is that this model is possessed by the recipient. It should now be recognized that there is a certain amount of information necessary to describe this model. As long as the amount of information in the model is small compared to the amount of information in the message, we can say that we have an absolute estimate of the information content of the message. As soon as the information content of the model approaches that of the message itself, then the amount of information transferred is sensitive to exactly what information is known. It might be possible to develop a theory of information that incorporates the information in the model,and thus to arrive at a more absolute measure of information. Alternatively, it might be necessary to develop a theory that considers the recipient and source more completely, since in actual communication between human beings, both are nonergodic systems possessed of a large amount of information. There is significant overlap of the information possessed by the recipient and the source. Moreover, this common information is essential to the communication itself.

One effort to arrive at a universal definition of information content of a message has been made by formally quantifying the information contained in models. The resulting information measure, Kolmogorov complexity, is based on computation theory discussed in the next section. While there is some success with this approach,two difficulties remain. In order for a universal definition of information to be agreed upon,models must still have an information content which is less than the message—knowledge possessed must be smaller than that received. Also, to calculate the information contained in a particular message is essentially impossible, since it requires computational effort that grows exponentially with the length of the message. In any practical case,the amount of information contained in a message must be estimated using a limited set of models of the source. The utilization of a limited set of models means that any estimate of the information in a message is an upper bound.

1.9 Computation

The theory of computation describes the operations that we perform on numbers, including addition, subtraction, multiplication and division. More generally, a computation is a sequence of operations each of which has a definite/unique/well-defined result. The fundamental study of such operations is the theory of logic. Logical operations do not necessarily act upon numbers, but rather upon abstract objects called statements. Statements can be combined together using operators such as AND and OR, and acted upon by the negation operation NOT. The theory of logic and the theory of computation are at root the same. All computations that have been conceived of can be constructed out of logical operations. We will discuss this equivalence in some detail.

We also discuss a further equivalence, generally less well appreciated, between computation and deterministic time evolution. The theory of computation strives to describe the class of all possible discrete deterministic or causal systems. Computations are essentially causal relationships. Computation theory is designed to capture all such possible relationships. It is thus essential to our understanding not just of the behavior of computers, or of human logic, but also to the understanding of causal relationships in all physical systems. A counterpoint to this association of computation and causality is the recognition that certain classes of deterministic dynamical systems are capable of the property known as universal computation.

One of the central findings of the theory of computation is that many apparently different formulations of computation turn out to be equivalent. The sense in which they are equivalent is that each one can simulate the other. In the early years of computation theory, there was an effort to describe sets of operations that would be more powerful than others. When all of them were shown to be equivalent it became generally accepted (the Church-Turing hypothesis) that there is a well-defined set of possible computations realized by any of several conceptual formulations. This has become known as the theory of universal computation.

1.9.1 *Propositional logic*

Logic is the study of reasoning, inference and deduction. Propositional logic describes the manipulation of statements that are either true or false. It assumes that there exists a set of statements that are either true or false at a particular time, but not both. Logic then provides the possibility of using an assumed set of relationships between the statements to determine the truth or falsehood of other statements.

For example,the statements Q_1 = “I am standing” and Q_2 = “I am sitting” may be related by the assumption: Q_1 is true implies that Q_2 is not true. Using this assumption,it is understood that a statement “Q_1 AND Q_2” must be false. The falsehood depends only on the relationship between the two sentences and not on the particular meaning of the sentences. This suggests that an abstract construction that describes mechanisms of inference can be developed. This abstract construction is propositional logic.

Propositional logic is formed out of statements (propositions) that may be true (T) or false (F), and operations. The operations are described by their actions upon statements. Since the only concern of logic is the truth or falsehood of statements, we can describe the operations through tables of truth values (truth tables) as follows. NOT (^) is an operator that acts on a single statement (a unary operator) to form a new statement. If Q is a statement then ^Q (read “not Q”) is the symbolic representation of “It is not true that Q.” The truth of ^Q is directly (causally) related to the truth of Q by the relationship in the table:

Q	**^Q**
T	F
F	T

(1.9.1)

The value of the truth or falsehood of Q is shown in the left column and the corresponding value of the truth or falsehood of ^Q is given in the right column.

Similarly, we can write the truth tables for the operations AND (&) and OR (|):

Q_1	**Q_2**	**Q_1&Q_2**
T	T	T
T	F	F
F	T	F
F	F	F

(1.9.2)

Q_1	**Q_2**	**$Q_1 \vert Q_2$**
T	T	T
T	F	T
F	T	T
F	F	F

(1.9.3)

As the tables show, Q_1&Q_2 is only true if both Q_1 is true and Q_2 is true. $Q_1|Q_2$ is only false if both Q_1 is false and Q_2 is false.

Propositional logic includes logical theorems as statements. For example, the statement Q_1 is true if and only if Q_2 is true can also be written as a binary operation $Q_1 \equiv Q_2$ with the truth table:

Q_1	**Q_2**	**$Q_1 \equiv Q_2$**
T	T	T
T	F	F
F	T	F
F	F	T

(1.9.4)

Another binary operation is the statement Q_1 implies Q_2, Q_1 Q_2. When this statement is translated into propositional logic,there is a difficulty that is usually bypassed by the following convention:

Q_1	**Q_2**	**Q_1 Q_2**
T	T	T
T	F	F
F	T	T
F	F	T

(1.9.5)

The difficulty is that the last two lines suggest that when the antecedent Q_1 is false,the implication is true, whether or not the consequent Q_2 is true. For example, the statement "If I had wings then I could fly" is as true a statement as "If I had wings then I couldn't fly," or the statement "If I had wings then potatoes would be flat." The problem originates in the necessity of assuming that the result is true or false in a unique way based upon the truth values of Q_1 and Q_2. Other information is not admissible, and a third choice of "nonsense" or "incomplete information provided" is not allowed within propositional logic. Another way to think about this problem is to say that there are many operators that can be formed with definite outcomes. Regardless of how we relate these operators to our own logical processes, we can study the system of operators that can be formed in this way. This is a model, but not a complete one, for human logic. Or, if we choose to define logic as described by this system, then human thought (as reflected by the meaning of the word "implies") is not fully characterized by logic (as reflected by the meaning of the operation " ").

In addition to unary and binary operations that can act upon statements to form other statements, it is necessary to have parentheses that differentiate the order of operations to be performed. For example a statement $((Q_1 \equiv Q_2)\&(\wedge Q_3)|Q_1)$ is a series of operations on primitive statements that starts from the innermost parenthesis and progresses outward. As in this example, there may be more than one innermost parenthesis. To be definite, we could insist that the order of performing these operations is from left to right. However, this order does not affect any result.

Within the context of propositional logic, it is possible to describe a systematic mechanism for proving statements that are composed of primitive statements. There are several conclusions that can be arrived at regarding a particular statement. A tautology is a statement that is always true regardless of the truth or falsehood of its component statements. Tautologies are also called theorems. A contradiction is a statement that is always false. Examples are given in Question 1.9.1.

Question 1.9.1 Evaluate the truth table of:

a. $(Q_1 \quad Q_2)|((\wedge Q_2)\&Q_1)$

b. $(\wedge(Q_1 \quad Q_2))\equiv((\wedge Q_1)|Q_2)$

Identify which is a tautology and which is a contradiction.

Solution 1.9.1 Build up the truth table piece by piece:

a. Tautology:

Q_1	Q_2	$Q_1 \quad Q_2$	$(\wedge Q_2)\&Q_1$	$(Q_1 \quad Q_2)\|((\wedge Q_2)\&Q_1)$
T	T	T	F	T
T	F	F	T	T
F	T	T	F	T
F	F	T	F	T

(1.9.6)

b. Contradiction:

Q_1	Q_2	$\wedge(Q_1 \quad Q_2)$	$(\wedge Q_1)\vert Q_2$	$(\wedge(Q_1 \quad Q_2)) \equiv ((\wedge Q_1)\vert Q_2)$
T	T	F	T	F
T	F	T	F	F
F	T	F	T	F
F	F	F	T	F

(1.9.7) ▮

Question 1.9.2: Construct a theorem (tautology) from a contradiction.

Solution 1.9.2: By negation. ▮

1.9.2 *Boolean algebra*

Propositional logic is a particular example of a more general symbolic system known as a Boolean algebra. Set theory, with the operators complement,union and intersection, is another example of a Boolean algebra. The formulation of a Boolean algebra is convenient because within this more general framework a number of important theorems can be proven. They then hold for propositional logic,set theory and other Boolean algebras.

A Boolean algebra is a set of elements $B=\{Q_1,Q_2, \ldots\}$, a unary operator (^), and two binary operators, for which we adopt the notation (+,•),that satisfy the following properties for all Q_1, Q_2, Q_3 in B:

1. Closure: $\wedge Q_1$, Q_1+Q_2, and $Q_1 \bullet Q_2$ are in B
2. Commutative law: $Q_1+Q_2=Q_2+Q_1$, and $Q_1 \bullet Q_2=Q_2 \bullet Q_1$
3. Distributive law: $Q_1 \bullet (Q_2+Q_3)=(Q_1 \bullet Q_2)+(Q_1 \bullet Q_3)$ and $Q_1+(Q_2 \bullet Q_3)=(Q_1+Q_2) \bullet (Q_1+Q_3)$
4. Existence of identity elements, 0 and 1: $Q_1+0=Q_1$, and $Q_1 \bullet 1=Q_1$
5. Complementarity law: $Q_1+(\wedge Q_1)=1$ and $Q_1 \bullet (\wedge Q_1)=0$

The statements of properties 2 through 5 consist of equalities. These equalities indicate that the element of the set that results from operations on the left is the same as the element resulting from operations on the right. Note particularly the second part of the distributive law and the complementarity law that would not be valid if we interpreted + as addition and • as multiplication.

Assumptions 1 to 5 allow the proof of additional properties as follows:

6. Associative property: $Q_1+(Q_2+Q_3)=(Q_1+Q_2)+Q_3$ and $Q_1 \bullet (Q_2 \bullet Q_3)=(Q_1 \bullet Q_2) \bullet Q_3$
7. Idempotent property: $Q_1+Q_1=Q_1$ and $Q_1 \bullet Q_1=Q_1$
8. Identity elements are nulls: $Q_1+1=1$ and $Q_1 \bullet 0=0$
9. Involution property: $\wedge(\wedge Q_1)=Q_1$

10. Absorption property: Q_1+(Q_1•Q_2)=Q_1 and Q_1•(Q_1+Q_2)=Q_1
11. DeMorgan's Laws: ^(Q_1+Q_2)=(^Q_1)•(^Q_2) and ^(Q_1•Q_2)=(^Q_1)+(^Q_2)

To identify propositional logic as a Boolean algebra we use the set B={T,F} and map the operations of propositional logic to Boolean operations as follows:(^ to ^), (| to +) and (& to •). The identity elements are mapped:(1 to T) and (0 to F). The proof of the Boolean properties for propositional logic is given as Question 1.9.3.

Question 1.9.3: Prove that the identification of propositional logic as a Boolean algebra is correct.

Solution 1.9.3: (1) is trivial; (2) is the invariance of the truth tables of Q_1&Q_2, Q_1|Q_2 to interchange of values of Q_1 and Q_2; (3) requires comparison of the truth tables of Q_1|(Q_2&Q_3) and (Q_1|Q_2)&(Q_1|Q_3) (see below). Comparison of the truth tables of Q_1&(Q_2|Q_3) and (Q_1&Q_2)|(Q_1&Q_3) is done similarly.

Q_1	**Q_2**	**Q_3**	**Q_2&Q_3**	**Q_1|(Q_2&Q_3)**	**Q_1|Q_2**	**Q_1|Q_3**	**(Q_1|Q_2)&(Q_1|Q_3)**
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

(1.9.8)

(4) requires verifying Q_1&T=T, and Q_1|F=F (see the truth tables for & and | above);(5) requires constructing a truth table for Q|^Q and verifying that it is always T (see below). Similarly, the truth table for Q&^Q shows that it is always F.

Q	**^Q**	**Q|^Q**
T	F	T
F	T	T

(1.9.9) ▮

1.9.3 *Completeness*

Our objective is to show that an arbitrary truth table, an arbitrary logical statement, can be constructed out of only a few logical operations. Truth tables are also equivalent to numerical functions—specifically, functions of binary variables that have binary results (binary functions of binary variables). This can be seen using the Boolean representation of T and F as {1,0} that is more familiar as a binary notation for numerical functions. For example, we can write the AND and OR operations (functions) also as:

Q_1	Q_2	$Q_1 \bullet Q_2$	Q_1+Q_2
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

(1.9.10)

Similarly for all truth tables,a logical operation is a binary function of a set of binary variables. Thus,the ability to form an arbitrary truth table from a few logical operators is the same as the ability to form an arbitrary binary function of binary variables from these same logical operators.

To prove this ability, we use the properties of the Boolean algebra to systematically discuss truth tables. We first construct an alternative Boolean expression for Q_1+Q_2 by a procedure that can be generalized to arbitrary truth tables. The procedure is to look at each line in the truth table that contains an outcome of 1 and write an expression that provides unity for that line only. Then we combine the lines to achieve the desired table. $Q_1 \bullet Q_2$ is only unity for the first line,as can be seen from its column. Similarly, $Q_1 \bullet (\wedge Q_2)$ is unity for the second line and $(\wedge Q_1) \bullet Q_2$ is unity for the third line. Using the properties of + we can then combine the terms together in the form:

$$Q_1 \bullet Q_2 + Q_1 \bullet (\wedge Q_2) + (\wedge Q_1) \bullet Q_2 \qquad (1.9.11)$$

Using associative and identity properties, this gives the same result as Q_1+Q_2.

We have replaced a simple expression with a much more complicated expression in Eq.(1.9.11). The motivation for doing this is that the same procedure can be used to represent any truth table. The general form we have constructed is called the disjunctive normal form. We can construct a disjunctive normal representation for an arbitrary binary function of binary variables. For example, given a specific binary function of binary variables, $f(Q_1,Q_2,Q_3)$, we construct its truth table, e.g.,

Q_1	Q_2	Q_3	$f(Q_1,Q_2,Q_3)$
1	1	1	1
1	0	1	0
0	1	1	1
0	0	1	0
1	1	0	0
1	0	0	1
0	1	0	0
0	0	0	0

(1.9.12)

The disjunctive normal form is given by:

$$f(Q_1,Q_2,Q_3) = Q_1 \bullet Q_2 \bullet Q_3 + (\wedge Q_1) \bullet Q_2 \bullet Q_3 + Q_1 \bullet (\wedge Q_2) \bullet (\wedge Q_3) \qquad (1.9.13)$$

as can be verified by inspection. An analogous construction can represent any binary function.

We have demonstrated that an arbitrary truth table can be constructed out of the three operations $(\wedge, +, \bullet)$. We say that these form a complete set of operations. Since there are 2^n lines in a truth table formed out of n binary variables, there are 2^{2^n} possible functions of these n binary variables. Each is specified by a particular choice of the 2^n possible outcomes. We have achieved a dramatic simplification by recognizing that all of them can be written in terms of only three operators. We also know that at most $(1/2)n2^n$ (^) operations, $(n - 1)\ 2^n$ (•) operations and $2^n - 1$ (+) operations are necessary. This is the number of operations needed to represent the identity function 1 in disjunctive normal form.

It is possible to further simplify the set of operations required. We can eliminate either the + or the • operations and still have a complete set. To prove this we need only display an expression for either of them in terms of the remaining operations:

$$\begin{aligned} Q_1 \bullet Q_2 &= \wedge((\wedge Q_1)+(\wedge Q_2)) \\ Q_1 + Q_2 &= \wedge((\wedge Q_1)\bullet(\wedge Q_2)) \end{aligned} \qquad (1.9.14)$$

Question 1.9.4: Verify Eq. (1.9.14).

Solution 1.9.4: They may be verified using DeMorgan's Laws and the involution property, or by construction of the truth tables, e.g.:

Q_1	Q_2	$\wedge Q_1$	$\wedge Q_2$	$Q_1 \bullet Q_2$	$(\wedge Q_1)+(\wedge Q_2)$
1	1	0	0	1	0
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	1

(1.9.15) ▮

It is possible to go one step further and identify binary operations that can represent all possible functions of binary variables. Two possibilities are the NAND ($\hat{\&}$) and NOR ($\hat{|}$) operations defined by:

$$\begin{aligned} Q_1 \,\hat{\&}\, Q_2 &= \wedge(Q_1 \& Q_2) \rightarrow \wedge(Q_1 \bullet Q_2) \\ Q_1 \,\hat{|}\, Q_2 &= \wedge(Q_1 | Q_2) \rightarrow \wedge(Q_1 + Q_2) \end{aligned} \qquad (1.9.16)$$

Both the logical and Boolean forms are written above. The truth tables of these operators are:

Q_1	Q_2	$\wedge(Q_1 \bullet Q_2)$	$\wedge(Q_1 + Q_2)$
1	1	0	0
1	0	1	0
0	1	1	0
0	0	1	1

(1.9.17)

We can prove that each is complete by itself (capable of representing all binary functions of binary variables) by showing that they are capable of representing one of the earlier complete sets. We prove the case for the NAND operation and leave the NOR operation to Question 1.9.5.

$$\begin{gathered} {}^\wedge Q_1 = {}^\wedge(Q_1 \bullet Q_1) = Q_1 \mathbin{\hat{\&}} Q_1 \\ (Q_1 \bullet Q_2) = {}^\wedge({}^\wedge(Q_1 \bullet Q_2)) = {}^\wedge(Q_1 \mathbin{\hat{\&}} Q_2) = (Q_1 \mathbin{\hat{\&}} Q_2) \mathbin{\hat{\&}} (Q_1 \mathbin{\hat{\&}} Q_2) \end{gathered} \tag{1.9.18}$$

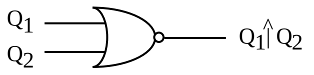
Question 1.9.5: Verify completeness of the NOR operation.

Solution 1.9.5: We can use the same formulas as in the proof of the completeness of NAND by replacing $\bullet$ with $+$ and $\hat{\&}$ with $\hat{|}$ everywhere. ∎

1.9.4 *Turing machines*

We have found that logical operators can represent any binary function of binary variables. This means that all well-defined mathematical operations on integers can be represented in this way. One of the implications is that we might make machines out of physical elements, each of which is capable of performing a Boolean operation. Such a machine would calculate a mathematical function and spare us a tedious task. We can graphically display the operations of a machine performing a series of Boolean operations as shown in Fig. 1.9.1. This is a simplified symbolic form similar to forms used in the design of computer logic circuits.

By looking carefully at Fig. 1.9.1 we see that there are several additional kinds of actions that are necessary in addition to the elementary Boolean operation. These actions are indicated by the lines that might be thought of as wires. One action is to transfer information from the location where it is input into the system, to the place where it is used. The second is to duplicate the information. Duplication is represented in the figure by a branching of the lines. The branching enables the same



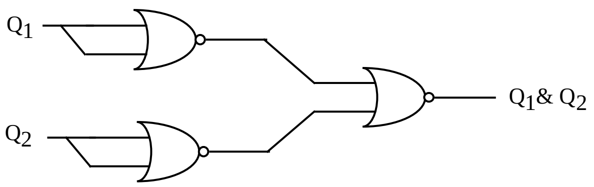


Figure 1.9.1 Graphical representation of Boolean operations. The top figure shows a graphical element representing the NOR operation $Q_1 \hat{|} Q_2 = {}^\wedge(Q_1 | Q_2)$. In the bottom figure we combine several operations together with lines (wires) indicating input, output, data duplication and transfer to form the AND operation, $(Q_1 \hat{|} Q_1) \hat{|} (Q_2 \hat{|} Q_2) = ({}^\wedge Q_1) \hat{|} ({}^\wedge Q_2) = Q_1 \& Q_2$. This equation may be used to prove completeness of the NOR operation. ∎

information to be used in more than one place. Additional implicit actions involve timing, because the representation makes an assumption that time causes the information to be moved and acted upon in a sequence from left to right. It is also necessary to have mechanisms for input and output.

The kind of mathematical machine we just described is limited to performing one prespecified function of its inputs. The process of making machines is time consuming. To physically rearrange components to make a new function would be inconvenient. Thus it is useful to ask whether we might design a machine such that part of its input could include a specification of the mathematical operation to be performed. Both information describing the mathematical function, and the numbers on which it is to be performed, would be encoded in the input which could be described as a string of binary characters.

This discussion suggests that we should systematically consider the properties/qualities of machines able to perform computations. The theory of computation is a self-consistent discussion of abstract machines that perform a sequence of prespecified well-defined operations. It extends the concept of universality that was discussed for logical operations. While the theory of logic determined that all Boolean functions could be represented using elementary logic operations, the theory of computation endeavors to establish what is possible to compute using a sequence of more general elementary operations. For this discussion many of the practical matters of computer design are not essential. The key question is to establish a relationship between machines that might be constructed and mathematical functions that may be computed. Part of the problem is to define what a computation is.

There are several alternative models of computation that have been shown to be equivalent in a formal sense since each one of them can simulate any other. Turing introduced a class of machines that represent a particular model of computation. Rather than maintaining information in wires, Turing machines (Fig. 1.9.2) use a storage device that can be read and written to. The storage is represented as an infinite one-dimensional tape marked into squares. On the tape can be written characters, one to a square. The total number of possible characters, the alphabet, is finite. These characters are often taken to be digits plus a set of markers (delimiters). In addition to the characters, the tape squares can also be blank. All of the tape is blank except for a finite number of nonblank places. Operations on the tape are performed by a roving read-write head that has a specified (finite) number of internal storage elements and a simple kind of program encoded in it. We can treat the program as a table similar to the tables discussed in the context of logic. The table operation acts upon the value of the tape at the current location of the head, and the value of storage elements within the read head. The result of an operation is not just a single binary value. Instead it corresponds to a change in the state of the tape at the current location (write), a change in the internal memory of the head, and a shift of the location of the head by one square either to the left or to the right.

We can also think about a Turing machine (TM) as a dynamic system. The internal table does not change in time. The internal state $s(t)$, the current location $l(t)$, the current character $a(t)$ and the tape $c(t)$ are all functions of time. The table consists of

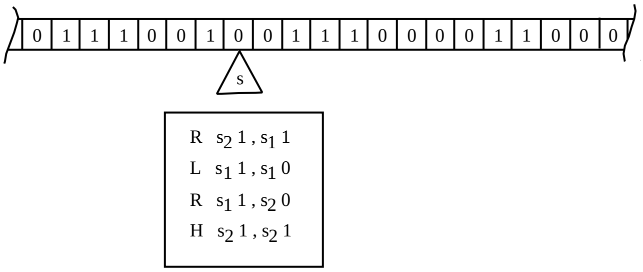


Figure 1.9.2 Turing's model of computation — the Turing machine (TM) — consists of a tape divided into squares with characters of a finite alphabet written on it. A roving "head" indicated by the triangle has a finite number of internal states and acts by reading and writing the tape according to a prespecified table of rules. Each rule consists of a command to read the tape, write the tape, change the internal state of the TM head and move either to the left or right. A simplified table is shown consisting of several rules of the form $\{\mu, s', a', s, a\}$ where a and a' are possible tape characters, s and s' are possible states of the head and μ is a movement of the head right (R), left (L) or halt (H). Each update the TM starts by finding the rule $\{\mu, s', a', s, a\}$ in the table such that a is the character on the tape at the current location of the head, and s is its current state. The tape is written with the corresponding a' and the state of the TM head is changed to s'. Then the TM head moves according to the corresponding μ right or left. The illustration simplifies the characters to binary digits 0 and 1 and the states of the TM head to s_1 and s_2. ▮

a set of instructions or rules of the form $\{\mu,s',a',s,a\}$ corresponding to a deterministic transition matrix. s and a are the current internal state and current tape character respectively. s' and a' are the new internal state and character. μ is the move to be made, either right or left (R or L).

Using either conceptual model, the TM starts from an initial state and location and a specified tape. In each time interval the TM head performs the following operations:

1. Read the current tape character
2. Find the instruction that corresponds to the existing combination of (s,a)
3. Change the internal memory to the corresponding s'
4. Write the tape with the corresponding character a'
5. Move the head to the left or right as specified by μ

When the TM head reaches a special internal state known as the halt state, then the outcome of the computation may be read from the tape. For simplicity, in what follows we will indicate entering the halt state by a move μ = H which is to halt.

The best way to understand the operation of a TM is to construct particular tables that perform particular actions (Question 1.9.6). In addition to logical operations, the possible actions include moving and copying characters. Constructing particular actions using a TM is tedious, in large part because the movements of the head are limited to a single displacement right or left. Actual computers use direct addressing that enables access to a particular storage location in its memory using a number (address) specifying its location. TMs do not generally use this because the tape is arbitrarily long, so that an address is an arbitrarily large number, requiring an arbitrarily large storage in the internal state of the head. Infinite storage in the head is not part of the computational model.

Question 1.9.6 The following TM table is designed to move a string of binary characters (0 and 1) that are located to the left of a special marker M to blank squares on the tape to the right of the M and then to stop on the M. Blank squares are indicated by B. The internal states of the head are indicated by s_1, s_2 . . . These are not italicized, since they are values rather than variables. The movements of the head right and left are indicated by R and L. As mentioned above, we indicate entering the halt state by a movement H. Each line has the form $\{\mu, s', a', s, a\}$.

Read over the program and convince yourself that it does what it is supposed to. Describe how it works. The TM must start from state s_1 and must be located at the leftmost nonblank character. The line numbering is only for convenience in describing the TM, and has no role in its operation.

1.	R	s_2	B	s_1	0
2.	R	s_3	B	s_1	1
3.	R	s_2	0	s_2	0
4.	R	s_2	1	s_2	1
5.	R	s_2	M	s_2	M
6.	R	s_3	0	s_3	0
7.	R	s_3	1	s_3	1
8.	R	s_3	M	s_3	M
9.	L	s_4	0	s_2	B
10.	L	s_4	1	s_3	B
11.	L	s_4	0	s_4	0
12.	L	s_4	1	s_4	1
13.	L	s_4	M	s_4	M
14.	R	s_1	B	s_4	B
15.	H	s_1	M	s_1	M

(1.9.19)

Solution 1.9.6 This TM works by (lines 1 or 2) reading a nonblank character (0 or 1) into the internal state of the head; 0 is represented by s_2 and 1 is represented by s_3. The character that is read is set to a blank B. Then the TM moves to the right, ignoring all of the tape characters 0, 1 or M (lines 3 through 8) until it reaches a blank B. It writes the stored character (lines 9 or 10), changing its state to s_4. This state specifies moving to the left,ignoring all characters 0,1 or M (lines 11 through 13) until it reaches a blank B. Then (line 14) it moves one step right and resets its state to s_1. This starts the procedure from the beginning. If it encounters the marker M in the state s_1 instead of a character to be copied, then it halts (line 15). ∎

Since each character can also be represented by a set of other characters (i.e.,2 in binary is 10), we can allow the TM head to read and write not one but a finite prespecified number of characters without making a fundamental change. The following TM, which acts upon pairs of characters and moves on the tape by two characters at a time, is the same as the one given in Question 1.9.6.

1.	01	01	00	00	01	
2.	01	11	00	00	11	
3.	01	01	01	01	01	
4.	01	01	11	01	11	
5.	01	01	10	01	10	
6.	01	11	01	11	01	
7.	01	11	11	11	11	
8.	01	11	10	11	10	(1.9.20)
9.	10	10	01	01	00	
10.	10	10	11	11	00	
11.	10	10	01	10	01	
12.	10	10	11	10	11	
13.	10	10	10	10	10	
14.	01	00	00	10	00	
15.	00	00	10	00	10	

The particular choice of the mapping from characters and internal states onto the binary representation is not unique. This choice is characterized by using the left and right bits to represent different aspects. In columns 3 or 5, which represent the tape characters, the right bit represents the type of element (marker or digit), and the left represents which element or marker it is: 00 represents the blank B, 10 represents M, 01 represents the digit 0,and 11 represents the digit 1. In columns 2 or 4, which represent the state of the head,the states s_1 and s_4 are represented by 00 and 10, s_2 and s_3 are represented by 01 and 11 respectively. In column 1, moving right is 01, left is 10, and halt is 00.

The architecture of a TM is very general and allows for a large variety of actions using complex tables. However, all TMs can be simulated by transferring all of the responsibility for the table and data to the tape.A TM that can simulate all TMs is called a universal Turing machine (UTM). As with other TMs,the responsibility of arranging the information lies with the "programmer." The UTM works by representing the table,current state,and current letter on the UTM tape. We will describe the essential concepts in building a UTM but will not explicitly build one.

The UTM acts on its own set of characters with its own set of internal states. In order to use it to simulate an arbitrary TM, we have to represent the TM on the tape of the UTM in the characters that the UTM can operate on. On the UTM tape, we

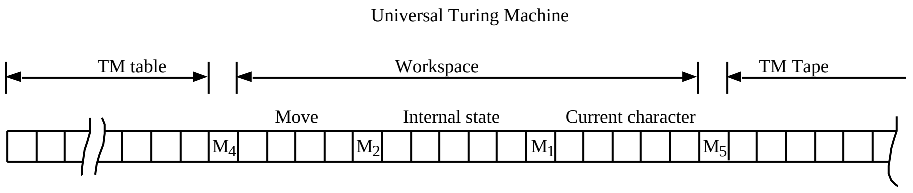


Figure 1.9.3 The universal Turing machine (UTM) is a special TM that can simulate the computation performed by any other TM. The UTM does this by executing the rules of the TM that are encoded on the tape of the UTM. There are three parts to the UTM tape, the part where the TM table is encoded (on the left), the part where the tape of the TM is encoded (on the right) and a workspace (in the middle) where information representing the current state of the TM head, the current character of the TM tape, and the movement command, are encoded. See the text for a description of the operation of the UTM based on its own rule table. ▮

must be able to represent four types of entities: a TM character, the state of the TM head, the movement to be taken by the TM head, and markers that indicate to the UTM what is where on the tape. The markers are special to the UTM and must be carefully distinguished from the other three. For later reference, we will build a particular type of UTM where the tape can be completely represented in binary.

The UTM tape has three parts,the part that represents the table of the TM,a work area,and the part that represents the tape of the TM (Fig. 1.9.3). To represent the tape and table of a particular but arbitrary TM, we start with a binary representation of its alphabet and of its internal states

$$\begin{gathered} a_1 \rightarrow 00000,\ a_2 \rightarrow 00001,\ a_3 \rightarrow 00010,\ \ldots \\ s_1 \rightarrow 000,\ s_2 \rightarrow 001,\ \ldots \end{gathered} \tag{1.9.21}$$

where we keep the left zeros, as needed for the number of bits in the longest binary number. We then make a doubled binary representation like that used in the previous example, where each bit becomes two bits with the low order bit a 1. The doubled binary notation will enable us to distinguish between UTM markers and all other entities on the tape. Thus we have:

$$\begin{gathered} a_1 \rightarrow 01\ 01\ 01\ 01\ 01,\ a_2 \rightarrow 01\ 01\ 01\ 01\ 11,\ a_3 \rightarrow 01\ 01\ 01\ 11\ 01,\ \ldots \\ s_1 \rightarrow 01\ 01\ 01,\ s_2 \rightarrow 01\ 01\ 11,\ \ldots \end{gathered} \tag{1.9.22}$$

These labels of characters and states are in a sense arbitrary, since the transition table is what gives them meaning.

We also encode the movement commands. The movement commands are not arbitrary, since the UTM must know how to interpret them. We have allowed the TM to displace more than one character, so we must encode a set of movements such as R_1, L_1, R_2, L_2, and H. These correspond respectively to moving one character right, one character left, two characters right, two characters left, and entering the halt state. Because the UTM must understand the move that is to be made, we must agree once and for all on a coding of these movements. We use the lowest order bit as a direction bit (1 Right, 0 Left) and the rest of the bits as the number of displacements in binary

$$\begin{aligned} R_1 &\rightarrow 011, R_2 \rightarrow 101, \ldots, \\ L_1 &\rightarrow 010, L_2 \rightarrow 100, \ldots, \\ H &\rightarrow 000 \text{ or } 001 \end{aligned} \tag{1.9.23}$$

The doubled binary representation is as before: each bit becomes two bits with the low order bit a 1,

$$\begin{aligned} R_1 &\rightarrow 01\ 11\ 11, R_2 \rightarrow 11\ 01\ 11, \ldots, \\ L_1 &\rightarrow 01\ 11\ 01, L_2 \rightarrow 11\ 01\ 01, \ldots, \\ H &\rightarrow 01\ 01\ 01 \text{ or } 01\ 01\ 11 \end{aligned} \tag{1.9.24}$$

Care is necessary in the UTM design because we do not know in advance how many types of TM moves are possible. We also don't know how many characters or internal states the TM has. This means that we don't know the length of their binary representations.

We need a number of markers that indicate to the UTM the beginning and end of encoded characters, states and movements described above. We also need markers to distinguish different regions of the tape. A sufficient set of markers are:

M_1—the beginning of a TM character,

M_2—the beginning of a TM internal state,

M_3—the beginning of a TM table entry, which is also the beginning of a movement command,

M_4—a separator between the TM table and the workspace,

M_5—a separator between the workspace and the TM tape,

M_6—the beginning of the current TM character (the location of the TM head),

M_7—the identified TM table entry to be used in the current step, and

B—the blank, which we include among the markers.

Depending on the design of the UTM, these markers need not all be distinct. In any case, we encode them also in binary

$$B \rightarrow 000, M_1 \rightarrow 001, M_2 \rightarrow 010, \ldots \tag{1.9.25}$$

and then doubled binary form where the second character is now zero:

$$B \rightarrow 00\ 00\ 00, M_1 \rightarrow 00\ 00\ 10, M_2 \rightarrow 00\ 10\ 00, \ldots \tag{1.9.26}$$

We are now in a position to encode both the tape and table of the TM on the tape of the UTM. The representation of the table consists of a sequence of representations of the lines of the table, L_1L_2..., where each line is represented by the doubled binary representation of

$$M_3\,\mu\,M_2\,s'\,M_1\,a'\,M_2\,s\,M_1 a \tag{1.9.27}$$

The markers are definite but the characters and states and movements correspond to those in a particular line in the table. The UTM representation of the tape of the TM, $a_1a_2 \ldots$, is a doubled binary representation of

$$\mathrm{M}_1\ a_1\ \mathrm{M}_1\ a_2\ \mathrm{M}_1\ a_3 \ldots \qquad (1.9.28)$$

The workspace starts with the character M_4 and ends with the character M_5. There is room enough for the representation of the current TM machine state, the current tape character and the movement command to be executed. At a particular time in execution it appears as:

$$\mathrm{M}_4\ \mu\ \mathrm{M}_2\ s\ \mathrm{M}_1\ a\ \mathrm{M}_5 \qquad (1.9.29)$$

We describe in general terms the operation of the UTM using this representation of a TM. Before execution we must indicate the starting location of the TM head and its initial state. This is done by changing the corresponding marker M_1 to M_6 (at the UTM tape location to the left of the character corresponding to the initial location of the TM), and the initial state of the TM is encoded in the workspace after M_2.

The UTM starts from the leftmost nonblank character of its tape. It moves to the right until it encounters M_6. It then copies the character after M_6 into the work area after M_1. It compares the values of (s,a) in the work area with all of the possible (s,a) pairs in the transition table pairs until it finds the same pair. It marks this table entry with M_7. The corresponding s' from the table is copied into the work area after M_2. The corresponding a' is copied to the tape after M_6. The corresponding movement command μ is copied to the work area after M_4. If the movement command is H the TM halts. Otherwise, the marker M_6 is moved according to the value of μ. It is moved one step at a time (i.e., the marker M_6 is switched with the adjacent M_1) while decrementing the value of the digits of μ (except the rightmost bit) and in the direction specified by the rightmost bit. When the movement command is decremented to zero, the TM begins the cycle again by copying the character after M_6 into the work area.

There is one detail we have overlooked: the TM can write to the left of its nonblank characters. This would cause problems for the UTM we have designed, since to the left of the TM tape representation is the workspace and TM table. There are various ways to overcome this difficulty. One is to represent the TM tape by folding it upon itself and interleaving the characters. Starting from an arbitrary location on the TM tape we write all characters on the UTM tape to the right of M_5, so that odd characters are the TM tape to the right, and even ones are the TM tape to the left. Movements of the M_6 marker are doubled, and it is reflected (bounces) when it encounters M_5.

A TM is a dynamic system. We can reformulate Turing's model of computation in the form of a cellular automaton (Section 1.5) in a way that will shed some light on the dynamics that are being discussed. The most direct way to do this is to make an automaton with two adjacent tapes. The only information in the second strip is a single nonblank character at the location of the head that represents its internal state. The TM update is entirely contained within the update rule of the automaton. This update rule may be constructed so that it acts at every point in the space, but is enabled by the nonblank character in the adjacent square on the second tape. When the dynamics reaches a steady state (it is enough that two successive states of the automaton are the same),the computation is completed. If desired we could reduce this CA to one tape by placing each pair of squares in the two tapes adjacent to each other, interleaving the two tapes. While a TM can be represented as a CA,any CA with only a finite number of active cells can be updated by a Turing machine program (it is computable). There are many other CA that can be programmed by their initial state to perform computations. These can be much simpler than using the TM model as a starting point. One example is Conway's Game of Life, discussed in Section 1.5.Like a UTM, this CA is a universal computer—any computation can be performed by starting from some initial state and looking at the final steady state for the result.

When we consider the relationship of computation theory to dynamic systems, there are some intentional restrictions in the theory that should be recognized. The conventional theory of computation describes a single computational unit operating on a character string formed from a finite alphabet of characters. Thus, computation theory does not describe a continuum in space,an infinite array of processors, or real numbers. Computer operations only mimic approximately the formal definition of real numbers. Since an arbitrary real number requires infinitely many digits to specify, computations upon them in finite time are impossible. The rejection by computation theory of operations upon real numbers is not a trivial one. It is rooted in fundamental results of computation theory regarding limits to what is inherently possible in any computation.

This model of computation as dynamics can be summarized by saying that a computation is the steady-state result of a deterministic CA with a finite alphabet (finite number of characters at each site) and finite domain update rule.One of the characters (the blank or vacuum) must be such that it is unchanged when the system is filled with these characters. The space is infinite but the conditions are such that all space except for a finite region must be filled with the blank character.

1.9.5 *Computability and the halting problem*

The construction of a UTM guarantees that if we know how to perform a particular operation on numbers, we can program a UTM to perform this computation. However, if someone gives you such a program—can you determine what it will compute? This seemingly simple question turns out to be at the core of a central problem of logic theory. It turns out that it is not only difficult to determine what it will compute,it is,in a formal sense that will be described below, impossible to figure out if it will compute anything at all. The requirement that it will compute something is that eventually it will halt. By halting, it declares its computation completed and the answer given. Instead of halting, it might loop forever or it might continue to write on ever larger regions of tape. To say that we can determine whether it will compute something is equivalent to saying that it will eventually halt. This is called the halting problem. How could we determine if it would halt? We have seen above how to represent an arbitrary TM on the tape of a particular TM. Consistent with computation theory, the halting problem is to construct a special TM, T_H, whose input is a description of a TM and whose output is a single bit that specifies whether or not the

TM will halt. In order for this to make sense,the program T_H must itself halt. We can prove by contradiction that this is not possible in general, and therefore we say that the halting problem is not computable. The proof is based on constructing a paradoxical logical statement of the form “This statement is false.”

A proof starts by assuming we have a TM called T_H that accepts as input a tape representing a TM Y and its tape y. The output, which can be represented in functional form as $T_H(Y,y)$, is always well-defined and is either 1 or 0 representing the statement that the TM Y halts on y or doesn’t halt on y respectively. We now construct a logical contradiction by constructing an additional TM based on T_H. First we consider $T_H(Y,Y)$, which asks whether Y halts when acting on a tape representing itself. We design a new TM T_{H1} that takes only Y as input, copies it and then acts in the same way as T_H. So we have

$$T_{H1}(Y) = T_H(Y,Y) \tag{1.9.30}$$

We now define a TM T_{H2} that is based on T_{H1} but whenever T_{H1} gives the answer 0 it gives the answer 1,and whenever T_{H1} gives the answer 1 it enters a loop and computes forever. A moment’s meditation shows that this is possible if we have T_{H1}. Applying T_{H2} to itself then gives us the contradiction, since $T_{H2}(T_{H2})$ gives 1 if

$$T_{H1}(T_{H2}) = T_H(T_{H2},T_{H2}) = 0 \tag{1.9.31}$$

By definition of T_H this means that $T_{H2}(T_{H2})$ does not halt, which is a contradiction. Alternatively, $T_{H2}(T_{H2})$ computes forever if

$$T_{H1}(T_{H2}) = T_H(T_{H2},T_{H2}) = 1$$

by definition of T_H this means that $T_{H2}(T_{H2})$ halts, which is a contradiction.

The noncomputability of the halting problem is similar to Gödel’s theorem and other results denying the completeness of logic, in the sense that we can ask a question about a logical construction that cannot be answered by it.Gödel’s theorem may be paraphrased as: In any axiomatic formulation of number theory (i.e.,integers),it is possible to write a statement that cannot be proven T or F. There has been a lot of discussion about the philosophical significance of these theorems.A basic conclusion that may be reached is that they describe something about the relationship of the finite and infinite. Turing machines can be represented,as we have seen, by a finite set of characters. This means that we can enumerate them, and they correspond one-to-one to the integers. Like the integers, there are (countably) infinitely many of them. Gödel’s theorem is part of our understanding of how an infinite set of numbers must be described. It tells us that we cannot describe their properties using a finite set of statements. This is appealing from the point of view of information theory since an arbitrary integer contains an arbitrarily large amount of information. The noncomputability of the halting problem tells us more specifically that we can ask a question about a system that is described by a finite amount of information whose answer (in the sense of computation) is not contained within it. We have thus made a vague connection between computation and information theory. We take this connection one step further in the following section.

1.9.6 *Computation and information in brief*

One of our objectives will be to relate computation and information. We therefore ask, Can a calculation produce information? Let us think about the results of a TM calculation which is a string of characters—the nonblank characters on the output tape. How much information is necessary to describe it? We could describe it directly, or use a Markov model as in Section 1.8. However, we could also give the input of the TM and the TM description, and this would be enough information to enable us to obtain the output by computation. This description might contain more or fewer characters than the direct description of the output. We now return to the problem of defining the information content of a string of characters. Utilizing the full power of computation, we can define this as the length of the shortest possible input tape for a UTM that gives the desired character string as its output. This is called the algorithmic (or Kolmogorov) complexity of a character string. We have to be careful with the definition, since there are many different possible UTM. We will discuss this in greater detail in Chapter 8. However, this discussion does imply that a calculation cannot produce information. The information present at the beginning is sufficient to obtain the result of the computation. It should be understood, however, that the information that seems to us to be present in a result may be larger than the original information unless we are able to reconstruct the starting point and the TM used for the computation.

1.9.7 *Logic, computation and human thought*

Both logic and computation theory are designed to capture aspects of human thought. A fundamental question is whether they capture enough of this process—are human beings equivalent to glorified Turing machines? We will ask this question in several ways throughout the text and arrive at various conclusions, some of which support this identification and some that oppose it. One way to understand the question is as one of progressive approximation. Logic was originally designed to model human thought. Computation theory, which generalizes logic, includes additional features not represented in logic. Computers as we have defined them are instruments of computation. They are given input (information) specifying both program and data and provide well-defined output an indefinite time later. One of the features that is missing from this kind of machine is the continuous input-output interaction with the world characteristic of a sensory-motor system. An appropriate generalization of the Turing machine would be a robot. As it is conceived and sometimes realized, a robot has both sensory and motor capabilities and an embedded computer. Thus it has more of the features characteristic of a human being. Is this sufficient, or have we missed additional features?

Logic and computation are often contrasted with the concept of creativity. One of the central questions about computers is whether they are able to simulate creativity. In Chapter 3 we will produce a model of creativity that appears to be possible to simulate on a computer. Hidden in this model, however, is a need to use random numbers. This might seem to be a minor problem, since we often use computers to generate random numbers. However, computers do not actually generate randomness,they generate pseudo-random numbers. If we recall that randomness is the same as information, by the discussion in the previous section,a computer cannot generate true randomness.A Turing machine cannot generate a result that has more information than it is given in its initial data. Thus creativity appears to be tied at least in part to randomness, which has often been suggested, and this may be a problem for conventional computers. Conceptually, this problem can be readily resolved by adding to the description of the Turing machine an infinite random tape in addition to the infinite blank tape. This new system appears quite similar to the original TM specification.A reasonable question would ask whether it is really inherently different. The main difference that we can ascertain at this time is that the new system would be capable of generating results with arbitrarily large information content, while the original TM could not. This is not an unreasonable distinction to make between a creative and a logical system. There are still key problems with understanding the practical implications of this distinction.

The subtlety of this discussion increases when we consider that one branch of theoretical computer science is based on the commonly believed assumption that there exist functions that are inherently difficult to invert—they can only be inverted in a time that grows exponentially with the length of the nonblank part of the tape. For all practical purposes, they cannot be inverted, because the estimated lifetime of the universe is insufficient to invert such functions. While their existence is not proven, it has been proven that if they do exist, then such a function can be used to generate a string of characters that, while not random, cannot be distinguished from a random string in less than exponential time. This would suggest that there can be no practical difference between a TM with a random tape,and one without. Thus,the possibility of the existence of noninvertible functions is intimately tied to questions about the relationship between TM, randomness and human thought.

1.9.8 *Using computation and information to describe the real world*

In this section we review the fundamental relevance of the theories of computation and information in the real world. This relevance ultimately arises from the properties of observations and measurements.

In our observations of the world, we find that quantities we measure vary. Indeed, without variation there would be no such thing as an observation. There are variations over time as well as over space. Our intellectual effort is dedicated to classifying or understanding this variation. To concretize the discussion, we consider observations of a variable s which could be as a function of time $s(t)$ or of space $s(x)$. Even though x or t may appear continuous, our observations may often be described as a finite discrete set $\{s_i\}$. One of the central (meta)observations about the variation in value of $\{s_i\}$ is that sometimes the value of the variable s_i can be inferred from, is correlated with, or is not independent from its value or values at some other time or position s_j.

These concepts have to do with the relatedness of s_i to s_j. Why is this important? The reason is that we would like to know the value of s_i without having to observe it.

We can understand this as a problem in prediction—to anticipate events that will occur. We would also like to know what is located at unobserved positions in space; e.g., around the corner. And even if we have observed something, we do not want to have to remember all observations we make. We could argue more fundamentally that knowledge/information is important only if prediction is possible. There would be no reason to remember past observations if they were uncorrelated with anything in the future. If correlations enable prediction, then it is helpful to store information about the past. We want to store as little as possible in order to make the prediction. Why? Because storage is limited, or because accessing the right information requires a search that takes time. If a search takes more time than we have till the event we want to predict, then the information is not useful. As a corollary (from a simplified utilitarian point of view), we would like to retain only information that gives us the best, most rapid prediction, under the most circumstances, for the least storage.

Inference is the process of logic or computation. To be able to infer the state of a variable s_i means that we have a definite formula $f(s_j)$ that will give us the value of s_i with complete certainty from a knowledge of s_j. The theory of computation describes what functions f are possible. If the index i corresponds to a later time than j we say that we can predict its value. In addition to the value of s_j we need to know the function f in order to predict the value of s_i. This relationship need not be from a single value s_j to a single value s_i. We might need to know a collection of values $\{s_j\}$ in order to obtain the value of s_i from $f(\{s_j\})$.

As part of our experience of the world, we have learned that observations at a particular time are more closely related to observations at a previous time than observations at different nearby locations. This has been summarized by the principle of causality. Causality is the ability to determine what happens at one time from what happened at a previous time. This is more explicitly stated as microcausality—what happens at a particular time and place is related to what happened at a previous time in its immediate vicinity. Causality is the principle behind the notion of determinism, which suggests that what occurs is determined by prior conditions. One of the ways that we express the relationship between system observations over time is by conservation laws. Conservation laws are the simplest form of a causal relationship.

Correlation is a looser relationship than inference. The statement that values s_i and s_j are correlated implies that even if we cannot tell exactly what the value s_i is from a knowledge of s_j, we can describe it at least partially. This partial knowledge may also be inherently statistical in the context of an ensemble of values as discussed below. Correlation often describes a condition where the values s_i and s_j are similar. If they are opposite, we might say they are anticorrelated. However, we sometimes use the term "correlated" more generally. In this case, to say that s_i and s_j are correlated would mean that we can construct a function $f(s_j)$ which is close to the value of s_i but not exactly the same. The degree of correlation would tell us how close we expect them to be. While correlations in time appear to be more central than correlations in space, systems with interactions have correlations in both space and time.

Concepts of relatedness are inherently of an ensemble nature. This means that they do not refer to a particular value s_i or a pair of values (s_i, s_j) but rather to a collection of such values or pairs. The ensemble nature of relationships is often more explicit for correlations, but it also applies to inference. This ensemble nature is hidden by functional terminology that describes a relationship between particular values. For example, when we say that the temperature at 1:00 P.M. is correlated with the temperature at 12:00 P.M., we are describing a relationship between two temperature values. Implicitly, we are describing the collection of all pairs of temperatures on different days or at different locations. The set of such pairs are analogs. The concept of inference also generally makes sense only in reference to an ensemble. Let us assume for the moment that we are discussing only a single value s_i. The statement of inference would imply that we can obtain s_i as the value $f(s_j)$. For a single value, the easiest way (requiring the smallest amount of information) to specify $f(s_j)$ would be to specify s_i. We do not gain by using inference for this single case. However, we can gain if we know that, for example, the velocity of an object will remain the same if there are no forces upon it. This describes the velocity $v(t)$ in terms of $v(t')$ of any one object out of an ensemble of objects. We can also gain from inference if the function $f(s_j)$ gives a string of more than one s_i.

The notion of independence is the opposite of inference or correlation. Two values s_i and s_j are independent if there is no way that we can infer the value of one from the other, and if they are not correlated. Randomness is similar to independence. The word "independent" is used when there is no correlation between two observations. The word "random" is stronger, since it means that there is no correlation between an observed value and anything else. A random process, like a sequence of coin tosses, is a sequence where each value is independent of the others. We have seen in Section 1.8 that randomness is intimately related with information. Random processes are unpredictable, therefore it makes no sense for us to try to accumulate information that will help predict it. In this sense, a random process is simple to describe. However, once a random process has occurred, other events may depend upon it. For example, someone who wins a lottery will be significantly affected by an event presumed to be random. Thus we may want to remember the results of the random process after it occurs. In this case we must remember each value. We might ask, Once the random process has occurred, can we summarize it in some way? The answer is that we cannot. Indeed, this property has been used to define randomness.

We can abstract the problem of prediction and description of observations to the problem of data compression. Assume there are a set of observations $\{s_i\}$ for which we would like to obtain the shortest possible description from which we can reconstruct the complete set of observations. If we can infer one value from another, then the set might be compressed by eliminating the inferable values. However, we must make sure that the added information necessary to describe how the inference is to be done is less than the information in the eliminated values. Correlations also enable compression. For example, let us assume that the values are biased ON with a probability $P(1) = .999$ and OFF with a probability $P(-1) = 0.001$. This means that one in a thousand values is OFF and the others are ON. In this case we can remember which ones are OFF rather than keeping a list of all of the values. We would say they are ON except for numbers 3, 2000, 2403, 5428, etc. This is one way of coding the information. This method of encoding has a problem in that the numbers representing the locations of the OFF values may become large. They will be correlated because the first few digits of successive locations will be the same (...,431236,432112,434329,...). We can further reduce the list if we are willing to do some more processing, by giving the intervals between successive OFF values rather than the absolute numbers of their location.

Ultimately, when we have reached the limits of our ability to infer one observation from another, the rest is information that we need. For example, differential equations are based on the presumption that boundary conditions (initial conditions in time,and boundary conditions in space) are sufficient to predict the behavior of a system. The values of the initial conditions and the boundary conditions are the information we need. This simple model of a system, where information is clearly and simply separated from the problem of computation, is not always applicable.

Let us assume that we have made extensive observations and have separated from these observations a minimal set that then can be used to infer all the rest.A minimal set of information would have the property that no one piece of information in it could be obtained from other pieces of information. Thus,as far as the set itself is concerned, the information appears to be random. Of course we would not be satisfied with any random set; it would have to be this one in particular, because we want to use this information to tell us about all of the actual observations.

One of the difficulties with random numbers is that it is inherently difficult to prove that numbers are random. We may simply not have thought of the right function f that can predict the value of the next number in a sequence from the previous numbers. We could argue that this is one of the reasons that gambling is so attractive to people because of the use of "lucky numbers" that are expected by the individual to have a better-than-random chance of success. Indeed,it is the success of science to have shown that apparently uncorrelated events may be related. For example, the falling of a ball and the motion of the planets. At the same time, science provides a framework in which noncausal correlations, otherwise called superstitions, are rejected.

We have argued that the purpose of knowledge is to succinctly summarize information that can be used for prediction. Thus,in its most abstract form, the problem of deduction or prediction is a problem in data compression. It can thus be argued that science is an exercise in data compression. This is the essence of the principle of Occam's razor and the importance of simplicity and universality in science. The more universal and the more general a law is,and the simpler it is,then the more data compression has been achieved. Often this is considered to relate to how valuable is the contribution of the law to science. Of course, even if the equations are general and simple,if we cannot solve them then they are not particularly useful from a practical point of view. The concept of simplicity has always been poorly defined. While science seeks to discover correlations and simplifications in observations of the universe around us,ultimately the minimum description of a system (i.e.,the universe) is given by the number of independent pieces of information required to describe it.

Our understanding of information and computation enters also into a discussion of our models of systems discussed in previous sections. In many of these models, we assumed the existence of random variables, or random processes. This randomness represents either unknown or complex phenomena. It is important to recognize that this represents an assumption about the nature of correlations between different aspects of the problem that we are modeling. It assumes that the random process is independent of (uncorrelated with) the aspects of the system we are explicitly studying. When we model the random process on a computer by a pseudo-random number generator, we are assuming that the computations in the pseudo-random number generator are also uncorrelated with the system we are studying. These assumptions may or may not be valid, and tests of them are not generally easy to perform.

1.10 Fractals, Scaling and Renormalization

The physics of Newton and the related concepts of calculus, which have dominated scientific thinking for three hundred years,are based upon the understanding that at smaller and smaller scales—both in space and in time—physical systems become simple,smooth and without detail.A more careful articulation of these ideas would note that the fine scale structure of planets, materials and atoms is not without detail. However, for many problems, such detail becomes irrelevant at the larger scale. Since the details are irrelevant, formulating theories in a way that assumes that the detail does not exist yields the same results as a more exact description.

In the treatment of complex systems, including various physical and biological systems,there has been a recognition that the concept of progressive smoothness on finer scales is not always a useful mathematical starting point. This recognition is an important fundamental change in perspective whose consequences are still being explored.

We have already discussed in Section 1.1 the subject of chaos in iterative maps. In chaotic maps, the smoothness of dynamic behavior is violated. It is violated because fine scale details matter. In this section we describe fractals,mathematical models of the spatial structure of systems that have increasing detail on finer scales.Geometric fractals have a self-similar structure, so that the structure on the coarsest scale is repeated on finer length scales. A more general framework in which we can articulate questions about systems with behavior on all scales is that of scaling theory introduced in Section 1.10.3.One of the most powerful analytic tools for studying systems that have scaling properties is the renormalization group. We apply it to the Ising model in Section 1.10.4, and then return full cycle by applying the renormalization group to chaos in Section 1.10.5.A computational technique,the multigrid method, that enables the description of problems on multiple scales is discussed in Section 1.10.6. Finally, we discuss briefly the relevance of these concepts to the study of complex systems in Section 1.10.7.

1.10.1 *Fractals*

Traditional geometry is the study of the properties of spaces or objects that have integral dimensions. This can be generalized to allow effective fractional dimensions of objects, called fractals, that are embedded in an integral dimension space. In recent years the recognition that fractals can play an important role in modeling natural phenomena has fueled a whole area of research investigating the occurrence and properties of fractal objects in physical and biological systems.

Fractals are often defined as geometric objects whose spatial structure is self-similar. This means that by magnifying one part of the object, we find the same structure as of the original object. The object is characteristically formed out of a collection of elements: points, line segments, planar sections or volume elements. These elements exist in a space of the same or higher dimension to the elements themselves. For example, line segments are one-dimensional objects that can be found on a line, plane, volume or higher dimensional space. We might begin to describe a fractal by the objects of which it is formed. However, geometric fractals are often described by a procedure (algorithm) that creates them in an explicitly self-similar manner.

One of the simplest examples of a fractal object is the Cantor set (Fig. 1.10.1). This set is formed by a procedure that starts from a single line segment. We remove the middle third from the segment. There are then two line segments left. We then remove the middle third from both of these segments, leaving four line segments. Continuing iteratively, at the kth iteration there are 2^k segments. The Cantor set, which is the limiting set of points obtained from this process,has no line segments in it. It is self-similar by direct construction,since the left and right third of the original line segment can be expanded by a factor of three to appear as the original set.

An analog of the Cantor set in two dimensions is the Sierpinski gasket (Fig. 1.10.2). It is constructed from an equilateral triangle by removing an internal triangle which is half of the size of the original triangle. This procedure is then iterated for all of the smaller triangles that result. We can see that there are no areas that are left in this shape. It is self-similar, since each of the three corner triangles can be expanded by a factor of two to appear as the original set.

For self-similar objects, we can obtain the effective fractal dimension directly by considering their composition from parts. We do this by analogy with conventional

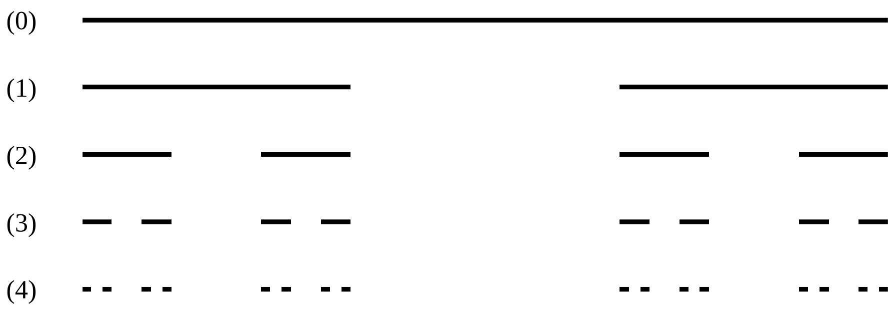


Figure 1.10.1 Illustration of the construction of the Cantor set, one of the best-known fractals. The Cantor set is formed by iteratively removing the middle third from a line segment, then the middle third from the two remaining line segments, and so on. Four iterations of the procedure are shown starting from the complete line segment at the top. ∎

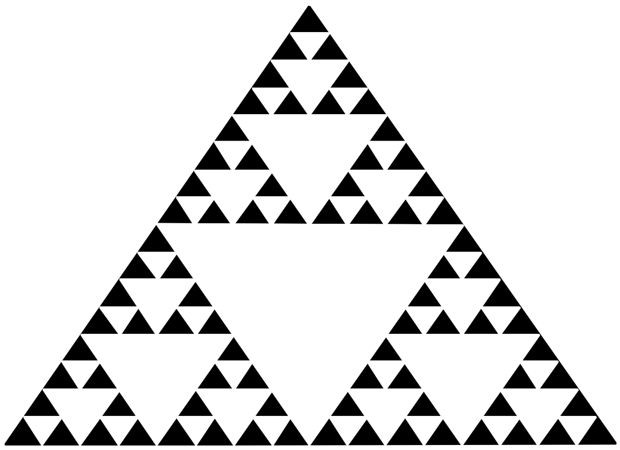


Figure 1.10.2 The Sierpinski gasket is formed in a similar manner to the Cantor set. Starting from an equilateral triangle, a similar triangle one half the size is removed from the middle leaving three triangles at the corners. The procedure is then iteratively applied to the remaining triangles. The figure shows the set that results after four iterations of the procedure. ▮

geometric objects which are also self-similar. For example,a line segment,a square, or a cube can be formed from smaller objects of the same type. In general, for a *d*-dimensional cube, we can form the cube out of smaller cubes. If the size of the smaller cubes is reduced from that of the large cube by a factor of η, where η is inversely proportional to their diameter, $\eta \quad 1/R$, then the number of smaller cubes necessary to form the original is $N = \eta^d$. Thus we could obtain the dimension as:

$$d = \ln(N) / \ln(\eta) \tag{1.10.1}$$

For self-similar fractals we can do the same, where *N* is the number of parts that make up the whole.Each of the parts is assumed to have the same shape, but reduced in size by a factor of η from the original object.

We can generalize the definition of fractal dimension so that we can use it to characterize geometric objects that are not strictly self-similar. There is more than one way to generalize the definition. We will adopt an intuitive definition of fractal dimension which is closely related to Eq. (1.10.1). If the object is embedded in *d*-dimensions, we cover the object with *d*-dimensional disks. This is illustrated in Fig. 1.10.3 for a line segment and a rectangle in a two-dimensional space. If we cover the object with two-dimensional disks of a fixed radius, *R*, using the minimal number of disks possible, the number of these disks changes with the radius of the disks according to the power law:

$$N(R) \quad R^{-d} \tag{1.10.2}$$

where *d* is defined as the fractal dimension. We note that the use of disks is only illustrative. We could use squares and the result can be proven to be equivalent.

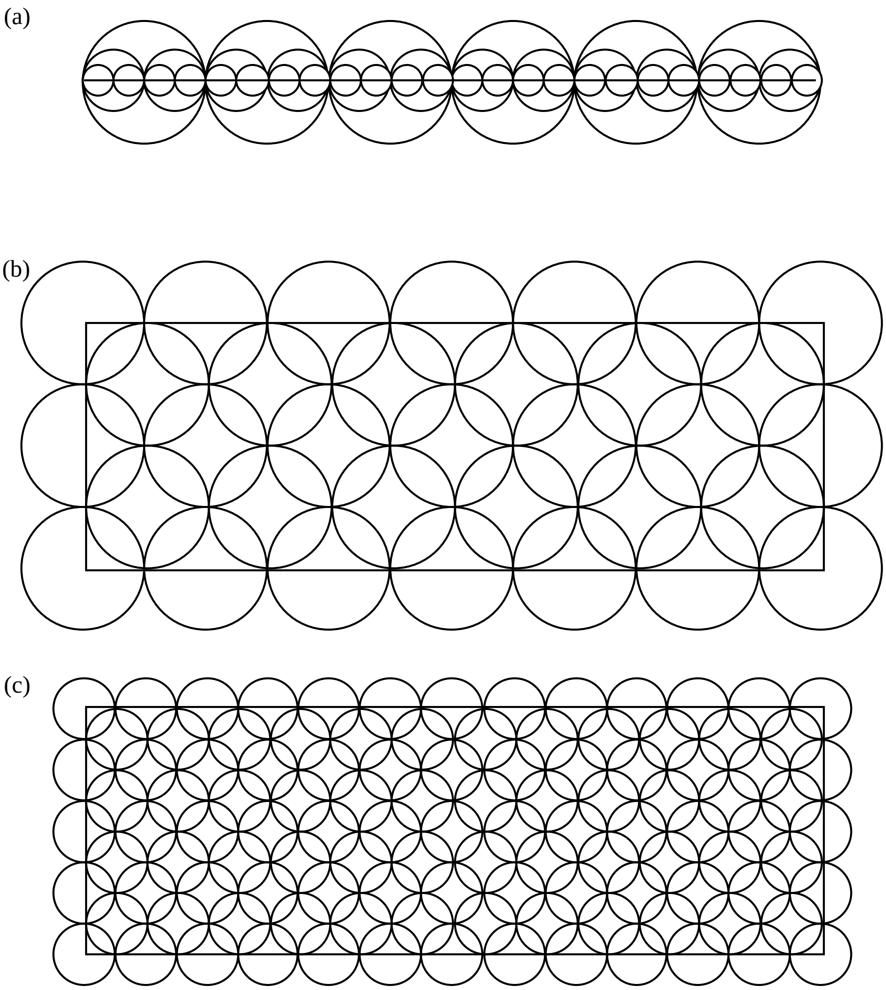


Figure 1.10.3 In order to define the dimension of a fractal object, we consider the problem of covering a set with a minimal number of disks of radius R. (a) shows a line segment with three different coverings superimposed. (b) and (c) show a rectangle with two different coverings respectively. As the size of the disks decreases the number of disks necessary to cover the shape grows as R^{-d}. This behavior becomes exact only in the limit $R \to 0$. The fractal dimension defined in this way is sometimes called the box-counting dimension, because d-dimensional boxes are often used rather than disks. ∎

We can use either Eq. (1.10.1) or Eq. (1.10.2) to calculate the dimension of the Cantor set and the Sierpinski gasket. We illustrate the use of Eq. (1.10.2). For the Cantor set, by construction, 2^k disks (or line segments) of radius $1/3^k$ will cover the set. Thus we can write:

$$N(R / 3^k) = 2^k N(R) \qquad (1.10.3)$$

Using Eq. (1.10.2) this is:

$$(R / 3^k)^{-\mathrm{d}} = 2^k R^{-d} \tag{1.10.4}$$

or:

$$3^d = 2 \tag{1.10.5}$$

which is:

$$d = \ln(2) / \ln(3) \quad 0.631 \tag{1.10.6}$$

We would arrive at the same result more directly from Eq. (1.10.1).

For the Sierpinski gasket, we similarly recognize that the set can be covered by three disks of radius 1/2, nine disks of radius 1/4,and more generally 3^k disks of radius $1/2^k$. This gives a dimension of:

$$d = \ln(3) / \ln(2) \quad 1.585 \tag{1.10.7}$$

For these fractals there is a deterministic algorithm that is used to generate them. We can also consider a kind of stochastic fractal generated in a similar way, however, at each level the algorithm involves choices made from a probability distribution. The simplest modification of the sets is to assume that at each level a choice is made with equal probability from several possibilities.For example,in the Cantor set, rather than removing the middle third from each of the line segments, we could choose at random which of the three thirds to remove. Similarly for the Sierpinski gasket, we could choose which of the four triangles to remove at each stage. These would be stochastic fractals,since they are not described by a deterministic self-similarity but by a statistical self-similarity. Nevertheless, they would have the same fractal dimension as the deterministic fractals.

Question 1.10.1 How does the dimension of a fractal,as defined by Eq. (1.10.2), depend on the dimension of the space in which it is embedded?

Solution 1.10.1 The dimension of a fractal is independent of the dimension of the space in which it is embedded. For example, we might start with a d-dimensional space and increase the dimension of the space to $d + 1$ dimensions. To show that Eq. (1.10.2) is not changed, we form a covering of the fractal by $d + 1$ dimensional spheres whose intersection with the d-dimensional space is the same as the covering we used for the analysis in d dimensions. ▮

Question 1.10.2 Prove that the fractal dimension does not change if we use squares or circles for covering an object.

Solution 1.10.2 Assume that we have minimal coverings of a shape using $N_1(\mathbf{R}) = c_1 R^{-d_1}$ squares, and minimal coverings by $N_2(R) = c_2 R^{-d_2}$ circles, with $d_1 \neq d_2$. The squares are characterized using R as the length of their side, while the circles are characterized using R as their radius. If d_1 is less than d_2, then for smaller and smaller R the number of disks becomes arbitrarily smaller than the number of squares. However, we can cover the same shape using squares that circumscribe the disks. The number of these squares is $N'_1(R) = c_1(R / 2)^{-d_1}$. This is impossible, because for small enough R, $N'_1(R)$ will be smaller than $N_1(R)$, which violates the assumption that the latter is a minimal covering. Similarly, if d is greater than d', we use disks circumscribed around the squares to arrive at a contradiction. ▮

Question 1.10.3 Calculate the fractal dimension of the Koch curve given in Fig. 1.10.4.

Solution 1.10.3 The Koch curve is composed out of four Koch curves reduced in size from the original by a factor of 3. Thus, the fractal dimension is $d = \ln(4) / \ln(3) \approx 1.2619$. ▮

Question 1.10.4 Show that the length of the Koch curve is infinite.

Solution 1.10.4 The Koch curve can be constructed by taking out the middle third of a line segment and inserting two segments equivalent to the one that was removed. They are inserted so as to make an equilateral triangle with the removed segment. Thus, at every iteration of the construction procedure, the length of the perimeter is multiplied by 4/3, which means that it diverges to infinity. It can be proven more generally that any fractal of dimension $2 > d > 1$ must have an infinite length and zero area,since these measures of size are for one-dimensional and two-dimensional objects respectively. ▮

Eq. (1.10.2) neglects the jumps in $N(R)$ that arise as we vary the radius R. Since $N(R)$ can only have integral values,as we lower R and add additional disks there are discrete jumps in its value. It is conventional to define the fractal dimension by taking the limit of Eq.(1.10.2) as $R \rightarrow 0$, where this problem disappears. This approach,however, is linked philosophically to the assumption that systems simplify in the limit of small length scales. The assumption here is not that the system becomes smooth and featureless, but rather that the fractal properties will continue to all finer scales and remain ideal. In a physical system,the fractal dimension cannot be taken in this limit. Thus, we should allow the definition to be applied over a limited domain of length scales as is appropriate for the problem. As long as the domain of length scales is large, we can use this definition. We then solve the problem of discrete jumps by treating the leading behavior of the function $N(R)$ over this domain.

The problem of treating distinct dimensions at different length scales is only one of the difficulties that we face in discussing fractal systems. Another problem is inhomogeneity. In the following section we discuss objects that are inherently inhomogeneous but for which an alternate natural definition of dimension can be devised to describe their structure on all scales.

1.10.2 *Trees*

Iterative procedures like those used to make fractals can also be used to make geometric objects called trees. An example of a geometric tree, which bears vague

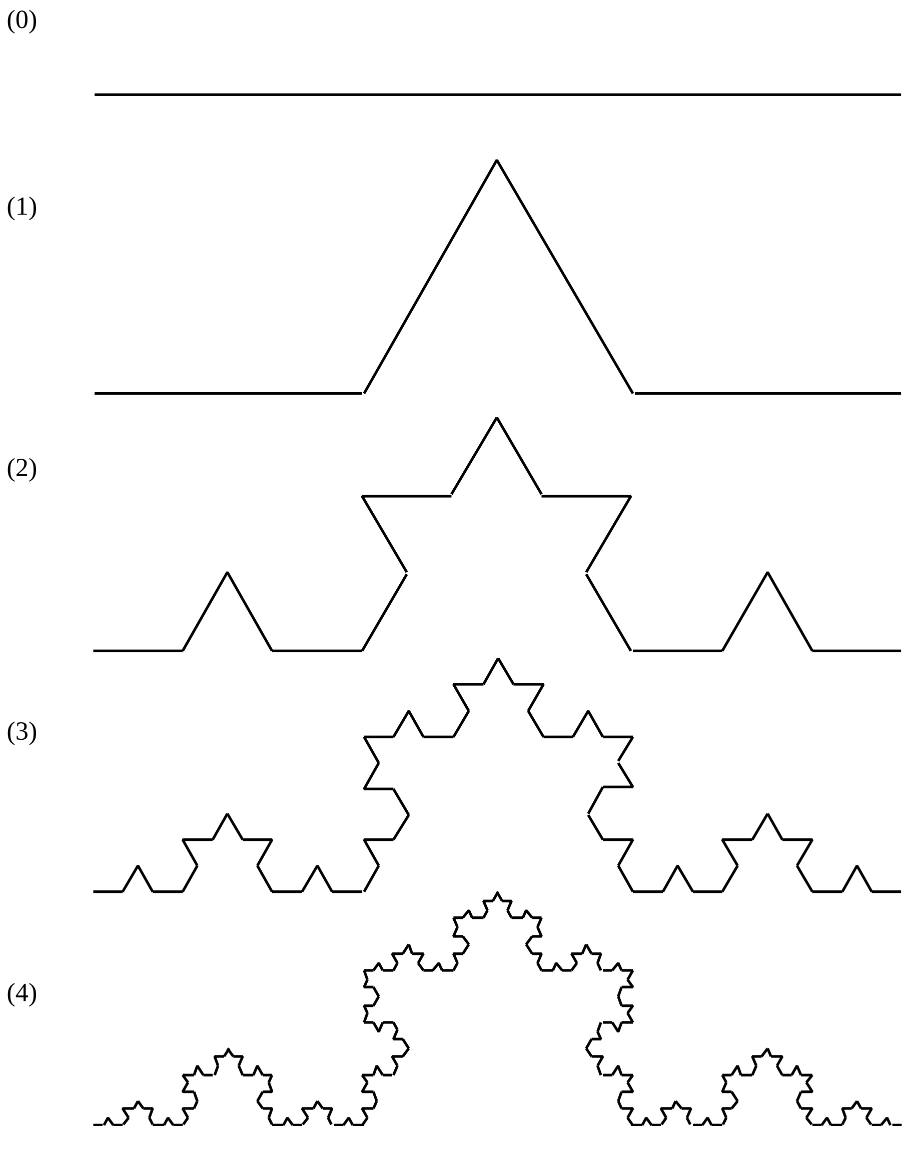


Figure 1.10.4 Illustration of the starting line segment and four successive stages in the formation of the Koch curve. For further discussion see Questions 1.10.3 and 1.10.4. ∎

resemblance to physical trees, is shown in Fig. 1.10.5. The tree is formed by starting with a single object (a line segment), scaling it by a factor of 1/2, duplicating it two times and attaching the parts to the original object at its boundary. This process is then iterated for each of the resulting parts. The iterations create structure on finer and finer scales.

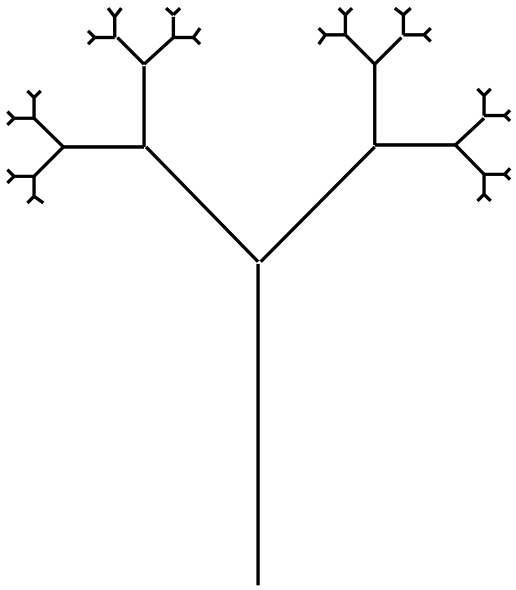


Figure 1.10.5 A geometric tree formed by an iterative algorithm similar to those used in forming fractals. This tree can be formed starting from a single line segment. Two copies of it are then reduced by a factor of 2, rotated by 45° left and right and attached at one end. The procedure is repeated for each of the resulting line segments. Unlike a fractal, a tree is not solely composed out of parts that are self-similar. It is formed out of self-similar parts, along with the original shape — its trunk. ❚

We can generalize the definition of a tree to be a set formed by iteratively adding to an object copies of itself. At iteration t, the added objects are reduced in size by a factor η^t and duplicated N^t times, the duplicated versions being rotated and then shifted by vectors whose lengths converge to zero as a function of t. A tree is different from a fractal because the smaller versions of the original object, are not contained within the original object.

The fractal dimension of trees is not as straightforward as it is for self-similar fractals. The effective fractal dimension can be calculated; however, it gives results that are not intuitively related to the tree structure. We can see why this is a problem in Fig. 1.10.6. The dimension of the region of the tree which is above the size R is that of the embedded entity (line segments), while the fractal dimension of the region which is less than the size R is determined by the spatial structure of the tree. Because of the changing value of R in the scaling relation, an intermediate value for the fractal dimension would typically be found by a direct calculation (Question 1.10.5).

It is reasonable to avoid this problem by classifying trees in a different category than fractals. We can define the tree dimension by considering the self-similarity of the tree structure using the same formula as Eq. (1.10.1), but now applying the definition to the number N and scaling η of the displaced parts of the generating structure, rather than the embedded parts as in the fractal. In Section 1.10.7 we will encounter a treelike structure; however, it will be more useful to describe it rather than to give a dimension that might characterize it.

Question 1.10.5 A simple version of a tree can be constructed as a set of points $\{1/k\}$ where k takes all positive integer values. The tree dimension

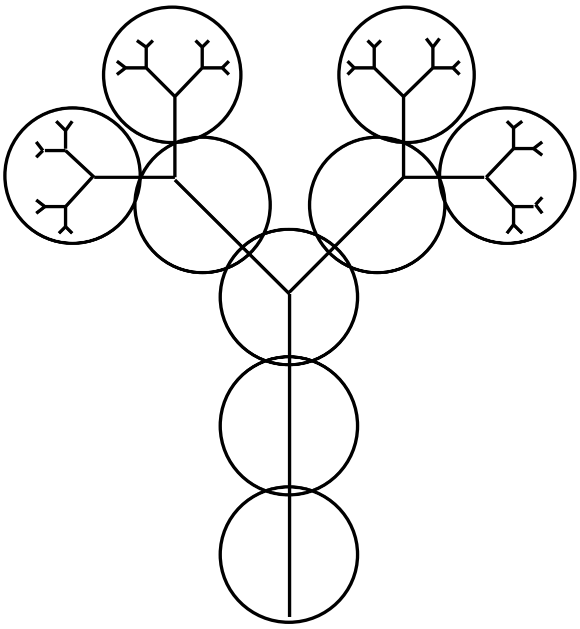


Figure 1.10.6 Illustration of the covering of a geometric tree by disks. The covering shows that the larger-scale structures of the tree (the trunk and first branches in this case) have an effective dimension given by the dimension of their components. The smaller scale structures have a dimension that is determined by the algorithm used to make the tree. This inhomogeneity implies that the fractal dimension is not always the natural way to describe the tree. ∎

of this set is zero because it can be formed from a point which is duplicated and then displaced by progressively smaller vectors. Calculate the fractal dimension of this set.

Solution 1.10.5 We construct a covering of scale R from line segments of this length. The covering that we construct will be formed out of two parts. One part is constructed from segments placed side by side. This part starts from zero and covers infinitely many points of the set. The other part is constructed from segments that are placed on individual points. The crossing point between the two sets can be calculated as the value of k where the difference between successive points is R. For k below this value,it is not possible to include more than one point in one line segment. For k above this value, there are two or more points per line segment. The critical value of k is found by setting:

$$\frac{1}{k_c} - \frac{1}{k_c + 1} = \frac{1}{k_c(k_c + 1)} \approx \frac{1}{k_c^2} = R \tag{1.10.8}$$

or $k_c = R^{-1/2}$. This means that the number of segments needed to cover individual points is given by this value. Also, the number of segments that are placed side by side must be enough to go up to this point, which has the value $1/k_c$. This number of segments is given by

$$\frac{1/k_c}{R} = R^{-1/2} \approx k_c \tag{1.10.9}$$

Thus we must cover the line segment up to the point $R^{1/2}$ with $R^{-1/2}$ line segments, and use an additional $R^{-1/2}$ line segments to cover the rest of the points. This gives a total number of line segments in a covering of $2R^{-1/2}$. The fractal dimension is thus $d = 1/2$.

We could have used fewer line segments in the covering by covering pairs of points and triples of points rather than covering the whole line segment below $1/k_c$. However, each partial covering of the set that is concerned with pairs, triples and so on consists of a number of segments that grows as $R^{-1/2}$. Thus our conclusion remains unchanged by this correction. ∎

Trees illustrate only one example of how system properties may exist on many scales, but are not readily described as fractals in the conventional sense. In order to generalize our concepts to enable the discussion of such properties, we will introduce the concept of scaling.

1.10.3 *Scaling*

Geometric fractals suggest that systems may have a self-similar structure on all length scales. This is in contrast with the more typical approach of science, where there is a specific scale at which a phenomenon appears. We can think about the problem of describing the behavior of a system on multiple length scales in an abstract manner. A phenomenon (e.g., a measurable quantity) may be described by some function of scale, $f(x)$. Here x represents the characteristic scale rather than the position. When there is a well-defined length scale at which a particular effect occurs, for longer length scales the function would typically decay exponentially:

$$f(x) \quad e^{-x/\lambda} \tag{1.10.10}$$

This functional dependence implies that the characteristic scale at which this property disappears is given by λ.

In order for a system property to be relevant over a large range of length scales, it must vary more gradually than exponentially. In such cases, typically, the leading behavior is a power law:

$$f(x) \quad x^{\alpha} \tag{1.10.11}$$

A function that follows such power-law behavior can also be characterized by the scaling rule:

$$f(ax) = a^{\alpha} f(x) \tag{1.10.12}$$

This means that if we characterize the system on one scale, then on a scale that is larger by the factor a it has a similar appearance, but scaled by the factor a^{α}. α is called the scaling exponent. In contrast to the behavior of an exponential, for a power law there is no particular length at which the property disappears. Thus, it may extend over a wide range of length scales. When the scaling exponent is not an integer, the function $f(x)$ is nonanalytic. Non-analyticity is often indicative of a property that cannot be treated by assuming that it becomes smooth on small or large scales. However, fractional scaling exponents are not necessary in order for power-law scaling to be applicable.

Even when a system property follows power-law scaling, the same behavior cannot continue over arbitrarily many length scales. The disappearance of a certain power law may occur because of the appearance of a new behavior on a longer scale. This change is characterized by a crossover in the scaling properties of $f(x)$. An example of crossover occurs when we have a quantity whose scaling behavior is

$$f(x) \sim A_1 x^{\alpha_1} + A_2 x^{\alpha_2} \tag{1.10.13}$$

If $A_1 > A_2$ and $\alpha_1 < \alpha_2$ then the first term will dominate at smaller length scales, and the second at larger length scales. Alternatively, the power-law behavior may eventually succumb to exponential decay at some length scale.

There are three related approaches to applying the concept of scaling in model or physical systems. The first approach is to consider the scale x to be the physical size of the system, or the amount of matter it contains. The quantity $f(x)$ is then a property of the system measured as the size of the system changes. The second approach is to keep the system the same, but vary the scale of our observation. We assume that our ability to observe the system has a limited degree of discernment of fine details—a finest scale of observation. Finer details are to be averaged over or disregarded. By moving toward or away from the system, we change the physical scale at which our observation can no longer discern details. x then represents the smallest scale at which we can observe variation in the system structure. Finally, in the third approach we consider the relationship between a property measured at one location in the system and the same property measured at another location separated by the distance x. The function $f(x)$ is a correlation of the system measurements as a function of the distance between regions that are being considered.

Examples of quantities that follow scaling relations as a function of system size are the extensive properties of thermodynamic systems (Section 1.3) such as the energy, entropy, free energy, volume, number of particles and magnetization:

$$U(ax) = a^d U(x) \tag{1.10.14}$$

These properties measure quantities of the whole system as a function of system size. All have the same scaling exponent—the dimension of space. Intrinsic thermodynamic quantities are independent of system size and therefore also follow a scaling behavior where the scaling exponent is zero.

Another example of scaling can be found in the random walk (Section 1.2). We can generalize the discussion in Section 1.2 to allow a walk in d dimensions by choosing steps which are ± 1 in each dimension independently. A random walk of N steps in three dimensions can be thought of as a simple model of a molecule formed as a chain of molecular units—a polymer. If we measure the average distance between the ends of the chain as a function of the number of steps $R(N)$, we have the scaling relation:

$$R(aN) = a^{1/2} R(N) \tag{1.10.15}$$

This scaling of distance traveled in a random walk with the number of steps taken is independent of dimension. We will consider random walks and other models of polymers in Chapter 5.

Often our interest is in knowing how different parts of the system affect each other. Direct interactions do not always reflect the degree of influence. In complex systems, in which many elements are interacting with each other, there are indirect means of interacting that transfer influence between one part of a system and another. The simplest example is the Ising model, where even short-range interactions can lead to longer-range correlations in the magnetization. The correlation function introduced in Section 1.6.5 measures the correlations between different locations. These correlations show the degree to which the interactions couple the behavior of different parts of the system. Correlations of behavior occur in both space and time. As we mentioned in Section 1.3.4, near a second-order phase transition, there are correlations between different places and times on every length and time scale, because they follow a power-law behavior. This example will be discussed in greater detail in the following section.

Our discussion of scaling also finds application in the theory of computation (Section 1.9) and the practical aspects of simulation (Section 1.7). In addition to the question of computability discussed in Section 1.9, we can also ask how hard it is to compute something. Such questions are generally formulated by describing a class of problems that can be ordered by a parameter N that describes the size of the problem. The objective of the theory of computational complexity is to determine how the number of operations necessary to solve a problem grows with N. A scaling analysis can also be used to compare different algorithms that may solve the same problem. We are often primarily concerned with the scaling behavior (exponential, power law and the value of the scaling exponent) rather than the coefficients of the scaling behavior, because in the comparison of the difficulty of solving different problems or different methodologies this is often, though not always, the most important issue.

1.10.4 *Renormalization group*

General method The renormalization group is a formalism for studying the scaling properties of a system. It starts by assuming a set of equations that describe the behavior of a system. We then change the length scale at which we are describing the system. In effect, we assume that we have a finite ability to see details. By moving away from a system, we lose some of the detail. At the new scale we assume that the same set of equations can be applied, but possibly with different coefficients. The objective is to relate the set of equations at one scale to the set of equations at the other scale. Once this is achieved, the scale-dependent properties of the system can be inferred.

Applications of the renormalization group method have been largely to the study of equilibrium systems, particularly near second-order phase transitions where mean field approaches break down (Section 1.6). The premise of the renormalization group is that exactly at a second-order phase transition, the equations describing the system are independent of scale. In recent years, dynamic renormalization theory has been developed to describe systems that evolve in time. In this section we will describe the more conventional renormalization group for thermodynamic systems.

We illustrate the concepts of renormalization using the Ising model. The Ising model, discussed in Section 1.6, describes the interactions of spins on a lattice. It is a first model of any system that exhibits simple cooperative behavior, such as a magnet.

In order to appreciate the concept of renormalization, it is useful to recognize that the Ising model is not a true microscopic theory of the behavior of a magnet. It might seem that there is a well-defined way to identify an individual spin with a single electron at the atomic level. However, this is far from apparent when equations that describe quantum mechanics at the atomic level are considered. Since the relationship between the microscopic system and the spin model is not manifest, it is clear that our description of the magnet using the Ising model relies upon the macroscopic properties of the model rather than its microscopic nature. Statistical mechanics does not generally attempt to derive macroscopic properties directly from microscopic reality. Instead, it attempts to describe the macroscopic phenomena from simple models. We might not give up hope of identifying a specific microscopic relationship between a particular material and the Ising model, however, the use of the model does not rely upon this identification.

Essential to this approach is that many of the details of the atomic regime are somehow irrelevant at longer length scales. We will return later to discuss the relevance or irrelevance of microscopic details. However, our first question is: What is a single spin variable? A spin variable represents the effective magnetic behavior of a region of the material. There is no particular reason that we should imagine an individual spin variable as representing a small or a large region of the material. Sometimes it might be possible to consider the whole magnet as a single spin in an external field. Identifying the spin with a region of the material of a particular size is an assignment of the length scale at which the model is being applied.

What is the difference between an Ising model describing the system at one length scale and the Ising model describing it on another? The essential point is that the interactions between spins will be different depending on the length scale at which we choose to model the system. The renormalization group takes this discussion one step further by explicitly relating the models at different scales.

In Fig. 1.10.7 we illustrate an Ising model in two dimensions. There is a second Ising model that is used to describe this same system but on a length scale that is twice as big. The first Ising model is described by the energy function (Hamiltonian):

$$E[\{s_i\}] = -c\sum_i 1 - h\sum_i s_i - J\sum_{<ij>} s_i s_j \tag{1.10.16}$$

For convenience, in what follows we have included a constant energy term $-cN = -c\,\Sigma 1$. This term does not affect the behavior of the system, however, its variation from scale to scale should be included. The second Ising model is described by the Hamiltonian

$$E'[\{s'_i\}] = -c'\sum_i 1 - h'\sum_i s'_i - J'\sum_{<ij>} s'_i s'_j \tag{1.10.17}$$

where both the variables and the coefficients have primes. While the first model has N spins, the second model has N' spins. Our objective is to relate these two models. The general process is called renormalization. When we go from the fine scale to the coarse scale by eliminating spins, the process is called decimation.

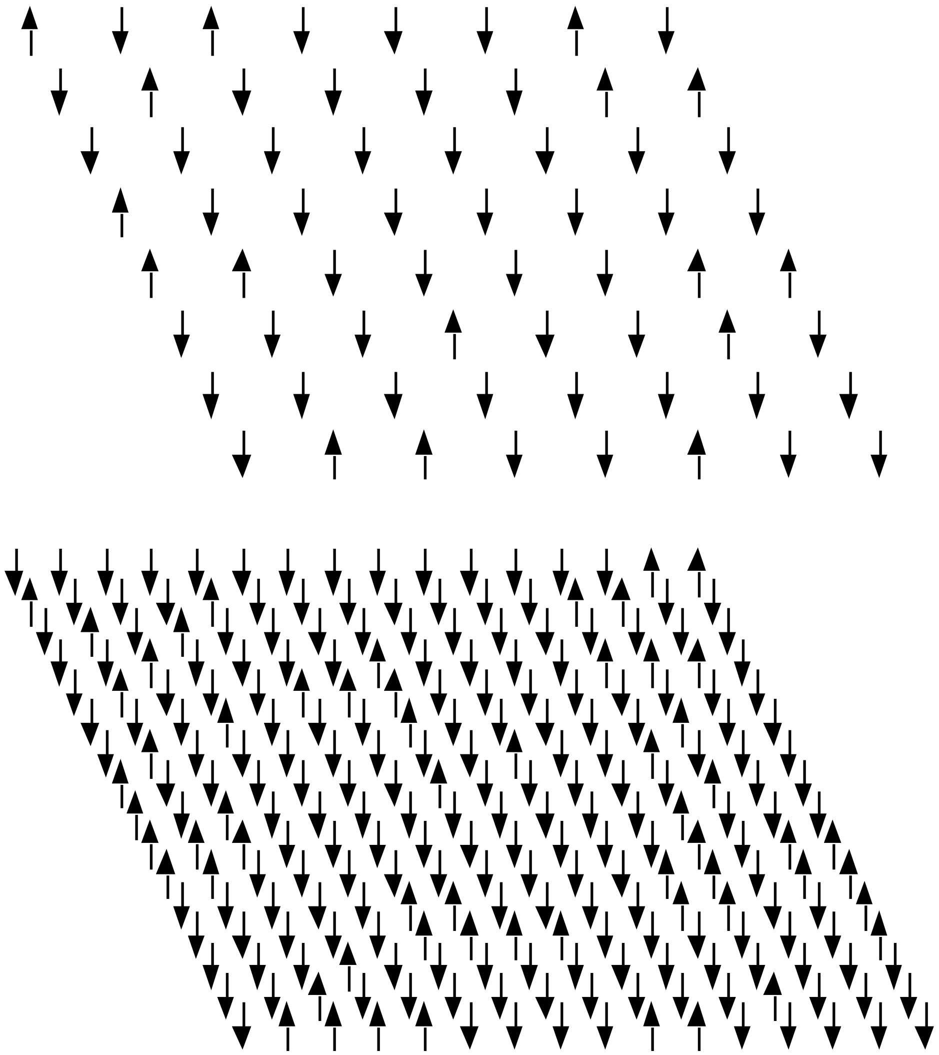


Figure 1.10.7 Schematic illustration of two Ising models in two dimensions. The spins are indicated by arrows that can be UP or DOWN. These Ising models illustrate the modeling of a system with different levels of detail. In the upper model there are one-fourth as many spins as in the lower model. In a renormalization group treatment the parameters of the lower model are related to the parameters of the upper model so that the same system can be described by both. Each of the spins in the upper model, in effect, represents four spins in the lower model. The interactions between adjacent spins in the upper model represent the net effect of the interactions between groups of four spins in the lower model. ▮

There are a variety of methods used for relating models at different scales. Each of them provides a distinct conceptual and practical approach. While in principle they should provide the same answer, they are typically approximated at some stage of the calculation and therefore the answers need not be the same. All the approaches we describe rely upon the partition function to enable direct connection from the microscopic statistical treatment to the macroscopic thermodynamic quantities. For a particular system, the partition function can be written so that it has the same value, independent of which representation is used:

$$Z = \sum_{\{s_i\}} e^{-\beta E[\{s_i\}]} = \sum_{\{s_i'\}} e^{-\beta E'[\{s_i'\}]} \tag{1.10.18}$$

It is conventional and convenient when performing renormalization transformations to set $\beta = 1/kT = 1$. Since β multiplies each of the parameters of the energy function, it is a redundant parameter. It can be reinserted at the end of the calculations.

The different approaches to renormalization are useful for various models that can be studied. We will describe three of them in the following paragraphs because of the importance of the different conceptual treatments. The three approaches are (1) summing over values of a subset of the spins, (2) averaging over a local combination of the spins, and (3) summing over the short wavelength degrees of freedom in a Fourier space representation.

1. Summing over values of a subset of the spins. In the first approach we consider the spins on the larger scale to be a subset of the spins on the finer scale. To find the energy of interaction between the spins on the larger scale we need to eliminate (decimate) some of the spins and replace them by new interactions between the spins that are left. Specifically, we identify the larger scale spins as corresponding to a subset $\{s_i\}_A$ of the smaller scale spins. The rest of the spins $\{s_i\}_B$ must be eliminated from the fine scale model to obtain the coarse scale model. We can implement this directly by using the partition function:

$$e^{-E'[\{s_i'\}]} = \sum_{\{s_i\}_B} e^{-E[\{s_i'\}_A, \{s_i\}_B]} = \sum_{\{s_i\}} e^{-E[\{s_i\}]} \prod_{i\ A} \delta_{s_i', s_i} \tag{1.10.19}$$

   In this equation we have identified the spins on the larger scale as a subset of the finer scale spins and have summed over the finer scale spins to obtain the effective energy for the larger scale spins.

2. Averaging over a local combination of the spins. We need not identify a particular spin of the finer scale with a particular spin of the coarser scale. We can choose to identify some function of the finer scale spins with the coarse scale spin. For example, we can identify the majority rule of a certain number of fine scale spins with the coarse scale spins:

$$e^{-E[\{s_i'\}]} = \sum_{\{s_i\}} \prod_{i\ A} \delta_{s_i', \mathrm{sign}(\sum s_i)} e^{-E[\{s_i\}]} \tag{1.10.20}$$

This is easier to think about when an odd number of spins are being renormalized to become a single spin. Note that this is quite similar to the concept of defining a collective coordinate that we used in Section 1.4 in discussing the two-state system. The difference here is that we are defining a collective coordinate out of only a few original coordinates, so that the reduction in the number of degrees of freedom is comparatively small. Note also that by convention we continue to use the term "energy," rather than "free energy," for the collective coordinates.

3. Summing over the short wavelength degrees of freedom in a Fourier space representation. Rather than performing the elimination of spins directly, we may recognize that our procedure is having the effect of removing the fine scale variation in the problem. It is natural then to consider a Fourier space representation where we can remove the rapid changes in the spin values by eliminating the higher Fourier components. To do this we need to represent the energy function in terms of the Fourier transform of the spin variables:

$$s_k = \sum_i e^{ikx_i} s_i \tag{1.10.21}$$

Writing the Hamiltonian in terms of the Fourier transformed variables, we then sum over the values of the high frequency terms:

$$e^{-E[\{s_k\}]} = \sum_{\{s_k\}_{k<k_0}} e^{-E[\{s_k\}]} \tag{1.10.22}$$

The remaining coordinates s_k have $k > k_0$.

All of the approaches described above typically require some approximation in order to perform the analysis. In general there is a conservation of effort in that the same difficulties tend to arise in each approach, but with different manifestation. Part of the reason for the difficulties is that the Hamiltonian we use for the Ising model is not really complete. This means that there can be other parameters that should be included to describe the behavior of the system. We will see this by direct application in the following examples.

Ising model in one dimension We illustrate the basic concepts by applying the renormalization group to a one-dimensional Ising model where the procedure can be done exactly. It is convenient to use the first approach (number 1 above) of identifying a subset of the fine scale spins with the larger scale model. We start with the case where there is an interaction between neighboring spins, but no magnetic field:

$$E[\{s_i\}] = -c\sum_i 1 - J\sum_{<ij>} s_i s_j \tag{1.10.23}$$

We sum the partition function over the odd spins to obtain

$$Z = \sum_{\{s_i\}_{\text{even}}} \sum_{\{s_i\}_{\text{odd}}} e^{c\sum_i 1 + J\sum_i s_i s_{i+1}} = \sum_{\{s_i\}_{\text{even}}} \prod_{i\,\text{even}} 2\cosh(J(s_i + s_{i+2}))e^{2c} \tag{1.10.24}$$

We equate this to the energy for the even spins by themselves, but with primed quantities:

$$Z = \sum_{\{s_i\}_{\text{even}}} e^{\sum_i c' + J' \sum_i s_i s_{i+2}} = \sum_{\{s_i\}_{\text{even}}} \prod_{i\,\text{even}} 2\cosh(J(s_i + s_{i+2}))e^{2c} \qquad (1.10.25)$$

This gives:

$$e^{c' + J' s_i s_{i+2}} = 2\cosh(J(s_i + s_{i+2}))e^{2c} \qquad (1.10.26)$$

or

$$c' + J' s_i s_{i+2} = \ln(2\cosh(J(s_i + s_{i+2}))) + 2c \qquad (1.10.27)$$

Inserting the two distinct combinations of values of s_i and s_{i+2} ($s_i = s_{i+2}$ and $s_i = -s_{i+2}$), we have:

$$\begin{aligned} c' + J' &= \ln(2\cosh(2J)) + 2c \\ c' - J' &= \ln(2\cosh(0)) + 2c = \ln(2) + 2c \end{aligned} \qquad (1.10.28)$$

Solving these equations gives the primed quantities for the larger scale model as:

$$\begin{aligned} J' &= (1/2)\ln(\cosh(2,J)) \\ c' &= 2c + (1/2)\ln(4\cosh(2J)) \end{aligned} \qquad (1.10.29)$$

This is the renormalization group relationship that we have been looking for. It relates the values of the parameters in the two different energy functions at the different scales.

While it may not be obvious by inspection, this iterative map always causes J to decrease. We can see this more easily if we transform the relationship of J to J' to the equivalent form:

$$\tanh(J') = \tanh(J)^2 \qquad (1.10.30)$$

This means that on longer and longer scales the effective interaction between neighboring spins becomes smaller and smaller. Eventually the system on long scales behaves as a string of decoupled spins.

The analysis of the one-dimensional Ising model can be extended to include a magnetic field. The decimation step becomes:

$$Z = \sum_{\{s_i\}_{\text{even}}} \sum_{\{s_i\}_{\text{odd}}} e^{c\sum_i 1 + h\sum_i s_i + J\sum_i s_i s_{i+1}} = \sum_{\{s_i\}_{\text{even}}} \prod_{i\,\text{odd}} 2\cosh(h + J(s_i + s_{i+2}))e^{2c} \qquad (1.10.31)$$

We equate this to the coarse scale partition function:

$$Z = \sum_{\{s_i\}_{\text{odd}}} e^{c' + h'\sum_i s_i' + J'\sum_i s_i' s_{i+1}'} = \sum_{\{s_i\}_{\text{odd}}} \prod_{i\,\text{odd}} 2\cosh(h + J(s_i + s_{i+2}))e^{2c} \qquad (1.10.32)$$

which requires that:

$$\begin{aligned} c' + h' + J' &= h + \ln(2\cosh(h + 2J)) + 2c \\ c' - J' &= \ln(2\cosh(h)) + 2c \\ c' - h' + J' &= -h + \ln(2\cosh(h - 2J)) + 2c \end{aligned} \qquad (1.10.33)$$

We solve these equations to obtain:

$$\begin{aligned} c' &= 2c + (1/4)\ln(16\cosh(h + 2J)\cosh(h - 2J)\cosh(h)^2) \\ J' &= (1/4)\ln(\cosh(h + 2J)\cosh(h - 2J)/\cosh(h)^2) \\ h' &= h + (1/2)\ln(\cosh(h + 2J)/\cosh(h - 2J)) \end{aligned} \qquad (1.10.34)$$

which is the desired renormalization group transformation. The renormalization transformation is an iterative map in the parameter space (c, h, J).

We can show what happens in this iterative map using a plot of changes in the values of J and h at a particular value of these parameters. Such a diagram of flows in the parameter space is illustrated in Fig. 1.10.8. We can see from the figure or from Eq. (1.10.34) that there is a line of fixed points of the iterative map at $J = 0$ with arbitrary

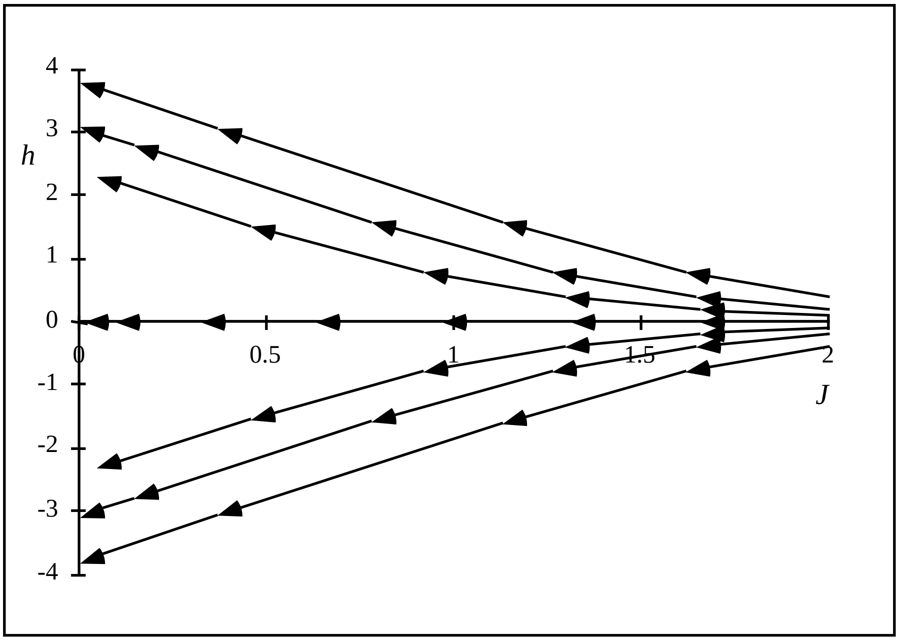


Figure 1.10.8 The renormalization transformation for the one-dimensional Ising model is illustrated as an iterative flow diagram in the two-dimensional (h,J) parameter space. Each of the arrows represents the effect of decimating half of the spins. We see that after a few iterations the value of J becomes very small. This indicates that the spins become decoupled from each other on a larger scale. The absence of any interaction on this scale means that there is no phase transition in the one-dimensional Ising model. ∎

value of h. This simply means that the spins are decoupled. For $J = 0$ on any scale,the behavior of the spins is determined by the value of the external field.

The line of fixed points at $J = 0$ is a stable (attracting) set of fixed points. The flow lines of the iterative map take us to these fixed points on the attractor line. In addition, there is an unstable fixed point at $J = \infty$. This would correspond to a strongly coupled line of spins, but since this fixed point is unstable it does not describe the large scale behavior of the model. For any finite value of J, changing the scale rapidly causes the value of J to become small. This means that the large scale behavior is always that of a system with $J = 0$.

Ising model in two dimensions In the one-dimensional case treated in the previous section, the renormalization group works perfectly and is also, from the point of view of studying phase transitions,uninteresting. We will now look at two dimensions, where the renormalization group must be approximated and where there is also a phase transition.

We can simplify our task in two dimensions by eliminating half of the spins (Fig. 1.10.9) instead of three out of four spins as illustrated previously in Fig. 1.10.7. Eliminating half of the spins causes the square cell to be rotated by 45°,but this should not cause any problems. Labeling the spins as in Fig. 1.10.9 we write the decimation step for a Hamiltonian with $h = 0$:

$$\begin{aligned} Z &= \sum_{\{s_i\}_A} \sum_{\{s_i\}_B} e^{c\sum_i 1 + J\sum_i s_0(s_1+s_2+s_3+s_4)} \\ &= \sum_{\{s_i\}_A} \prod_{i\,B} 2\cosh(J(s_1+s_2+s_3+s_4))e^c \\ &= \sum_{\{s_i\}_A} \prod_{i\,B} e^{c'+(J'/2)(s_1s_2+s_2s_3+s_3s_4+s_4s_1)} \end{aligned} \tag{1.10.35}$$

In the last expression we take into consideration that each bond of the form s_1s_2 appears in two squares and each spin appears in four squares.

In order to solve Eq. (1.10.35) for the values of $c'$and J' we must insert all possible values of the spins (s_1,s_2,s_3,s_4). However, this leads to a serious problem. There are four distinct equations that arise from the different values of the spins. This is reduced from $2^4 = 8$ because, by symmetry, inverting all of the spins gives the same answer. The problem is that while there are four equations, there are only two unknowns to solve for, c' and J'. The problem can be illustrated by recognizing that there are two distinct ways to have two spins UP and two spins DOWN. One way is to have the spins that are the same be adjacent to each other, and the other way is to have them be opposite each other across a diagonal. The two ways give the same result for the value of $(s_1 + s_2 + s_3 + s_4)$ but different results for $(s_1s_2 + s_2s_3 + s_3s_4 + s_4s_1)$.

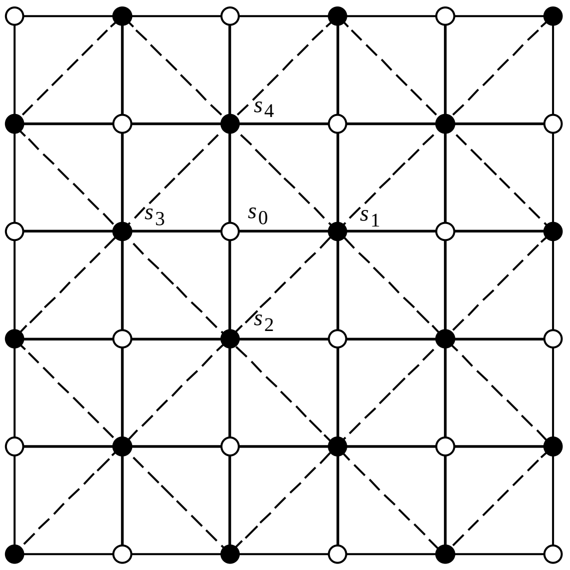


Figure 1.10.9 In a renormalization treatment of the two-dimensional Ising model it is possible to decimate one out of two spins as illustrated in this figure. The black dots represent spins that remain in the larger-scale model, and the white dots represent spins that are decimated. The nearest-neighbor interactions in the larger-scale model are shown by dashed lines. As discussed in the text, the process of decimation introduces new interactions between spins across the diagonal, and four spin interactions between spins around a square. ∎

In order to solve this problem, we must introduce additional parameters which correspond to other interactions in the Hamiltonian. To be explicit, we would make a table of symmetry-related combinations of the four spins as follows:

(s_1,s_2,s_3,s_4)	(1,1,1,1)	(1,1,1,−1)	(1,1,−1,−1)	(1,−1,1,−1)
1	1	1	1	1
$(s_1+s_2+s_3+s_4)$	4	2	0	0
$(s_1s_2+s_2s_3+s_3s_4+s_4s_1)$	4	0	0	−4
$(s_1s_3+s_2s_4)$	2	0	−2	2
$s_1s_2s_3s_4$	1	−1	1	1

(1.10.36)

In order to make use of these to resolve the problems with Eq.(1.10.35), we must introduce new interactions in the Hamiltonian and new parameters that multiply them. This leads to second-neighbor interactions (across a cell diagonal),and four spin interactions around a square:

$$E[\{s_i\}]=-c'\sum_i 1-J'\sum_{<ij>}s_is_j-K'\sum_{<<ij>>}s_is_j-L'\sum_{<ijkl>}s_is_js_ks_l \tag{1.10.37}$$

where the notation $<<ij>>$ indicates second-neighbor spins across a square diagonal, and $<ijkl>$ indicates spins around a square. This might seem to solve our problem. However, we started out from a Hamiltonian with only two parameters,and now we are switching to a Hamiltonian with four parameters. To be self-consistent, we should start from the same set of parameters we end up with. When we start with the additional parameters K and L this will,however, lead to still further terms that should be included.

Relevant and irrelevant parameters In general, as we eliminate spins by renormalization, we introduce interactions between spins that might not have been included in the original model. We will have interactions between second or third neighbors or between more than two spins at a time. In principle, by using a complete set of parameters that describe the system we can perform the renormalization transformation and obtain the flows in the parameter space. These flows tell us about the scale-dependent properties of the system.

We can characterize the flows by focusing on the fixed points of the iterative map. These fixed points may be stable or unstable. When a fixed point is unstable, renormalization takes us away from the fixed point so that on a larger scale the properties of the system are found to be different from the values at the unstable fixed point. Significantly, it is the unstable fixed points that represent the second-order phase transitions. This is because deviating from the fixed point in one direction causes the parameters to flow in one direction, while deviating from the fixed point in another direction causes the parameters to flow in a different direction. Thus, the macroscopic properties of the system depend on the direction microscopic parameters deviate from the fixed point—a succinct characterization of the nature of a phase transition.

Using this characterization of fixed points, we can now distinguish between different types of parameters in the model. This includes all of the additional parameters that might be introduced in order to achieve a self-consistent renormalization transformation. There are two major categories of parameters: relevant or irrelevant. Starting near a particular fixed point, changes in a relevant parameter grow under renormalization. Changes in an irrelevant parameter shrink. Because renormalization indicates the values of system parameters on a larger scale, this tells us which microscopic parameters are important to the macroscopic scale. When observed on the macroscopic scale, relevant parameters change at the phase transition, while irrelevant parameters do not. A relevant parameter should be included in the Hamiltonian because its value affects the macroscopic behavior. An irrelevant parameter may often be included in the model in a more approximate way. Marginal parameters are the borderline cases that neither grow nor shrink at the fixed point.

Even when we are not solely interested in the behavior of a system at a phase transition, but rather are concerned with its macroscopic properties in general, the definition of “relevant” and “irrelevant” continues to make sense. If we start from a particular microscopic description of the system, we can ask which parameters are relevant for the macroscopic behavior. The relevant parameters are the ones that can affect the macroscopic behavior of the system. Thus, a change in a relevant microscopic parameter changes the macroscopic behavior. In terms of renormalization, changes in relevant parameters do not disappear as a result of renormalization.

We see that the use of any model, such as the Ising model, to model a physical system assumes that all of the parameters that are essential in describing the system have been included. When this is true, the results are universal in the sense that all microscopic Hamiltonians will give rise to the same behavior. Additional terms in the Hamiltonian cannot affect the macroscopic behavior. We know that the microscopic behavior of the physical system is not really described by the Ising model or any other simple model. Thus, in creating models we always rely upon the concept, if not the process, of renormalization to make many of the microscopic details disappear, enabling our simple models to describe the macroscopic behavior of the physical system.

In the Ising model, in addition to longer range and multiple spin interactions, there is another set of parameters that may be relevant. These parameters are related to the use of binary variables to describe the magnetization of a region of the material. It makes sense that the process of renormalization should cause the model to have additional spin values that are intermediate between fully magnetized UP and fully magnetized DOWN. In order to accommodate this, we might introduce a continuum of possible magnetizations.Once we do this,the amplitude of the magnetization has a probability distribution that will be controlled by additional parameters in the Hamiltonian. These parameters may also be relevant or irrelevant. When they are irrelevant,the Ising model can be used without them. However, when they are relevant, a more complete model should be used.

The parameters that are relevant generally depend on the dimensionality of space. From our analysis of the behavior of the one-dimensional Ising model,the parameter J is irrelevant. It is clearly irrelevant because not only variations in J but J itself disappears as the scale increases. However, in two dimensions this is not true.

For our purposes we will be satisfied by simplifying the renormalization treatment of the two-dimensional Ising model so that no additional parameters are introduced. This can be done by a fourth renormalization group technique which has some conceptual as well as practical advantages over the others. However, it does hide the importance of determining the relevant parameters.

Bond shifting We simplify our analysis of the two-dimensional Ising model by making use of the Migdal-Kadanoff transformation. This renormalization group technique is based on the recognition that the correlation between adjacent spins can enable us to, in effect, substitute the role of one spin for another. As far as the coarser scale model is concerned, the function of the finer scale spins is to mediate the interaction between the coarser scale spins. Because one spin is correlated to the behavior of its neighbor, we can shift the responsibility for this interaction to a neighbor, and use this shift to simplify elimination of the spins.

To apply these ideas to the two-dimensional Ising model, we move some of the interactions (bonds) between spins, as shown in Fig. 1.10.10. We note that the distance over which the bonds act is preserved. The net result of the bond shifting is that we form short linear chains that can be renormalized just like a one-dimensional chain. The renormalization group transformation is thus done in two steps.First we shift the bonds, then we decimate. Once the bonds are moved, we write the renormalization of the partition function as:

$$
\begin{aligned}
Z &= \sum_{\{s_i\}_A} \sum_{\{s_i\}_B} \sum_{\{s_i\}_C} e^{c\sum_i 1 + 2J\sum_i s_0(s_1+s_2)} \\
&= \sum_{\{s_i\}} \prod_{i\ A} 2\cosh(2J(s_1+s_2))e^{4c} \qquad (1.10.38) \\
&= \sum_{\{s_i\}} \prod_{i\ A} e^{c'+J'(s_1 s_2)}
\end{aligned}
$$

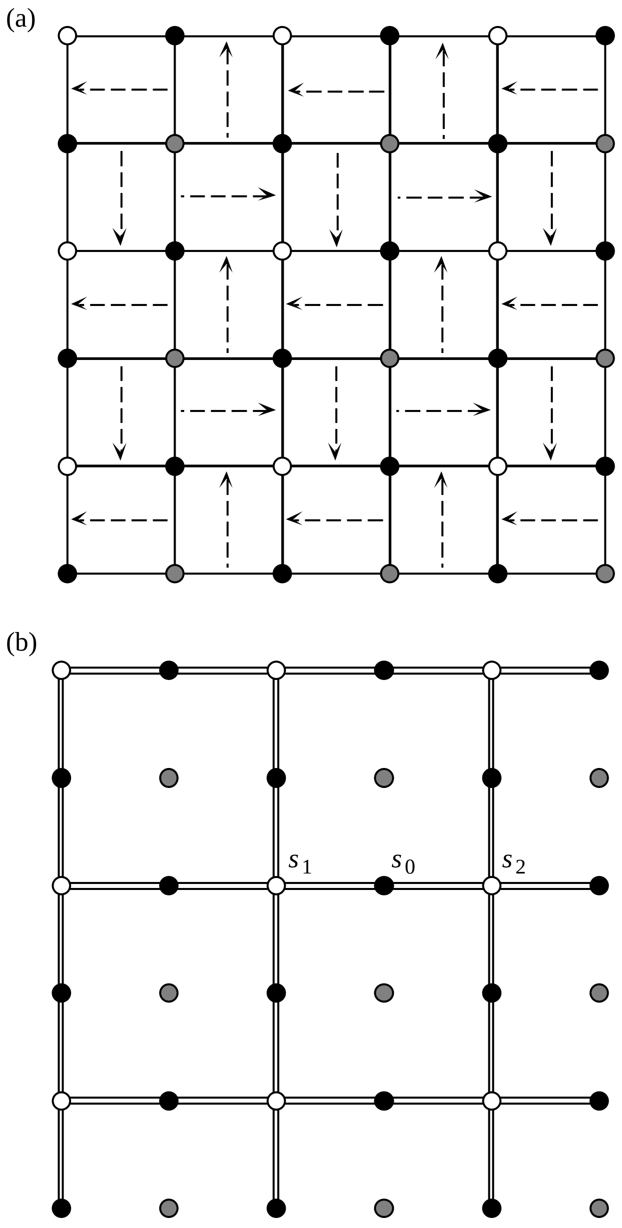


Figure 1.10.10 Illustration of the Migdal-Kadanoff renormalization transformation that enables us to bypass the formation of additional interactions. In this approach some of the interactions between spins are moved to other spins. If all the spins are aligned (at low temperature or high J), then shifting bonds doesn't affect the spin alignment. At high temperature, when the spins are uncorrelated, the interactions are not important anyway. Near the phase transition, when the spins are highly correlated, shifting bonds still makes sense. A pattern of bond movement is illustrated in (a) that gives rise to the pattern of doubled bonds in (b). Note that we are illustrating only part of a periodic lattice, so that bonds are moved into and out of the region illustrated. Using the exact renormalization of one-dimensional chains, the gray spins and the black spins can be decimated to leave only the white spins. ▮

The spin labels s_0, s_1, s_2 are assigned along each doubled bond, as indicated in Fig. 1.10.10. The three types of spin A, B and C correspond to the white, black and gray dots in the figure. The resulting equation is the same as the one we found when performing the one-dimensional renormalization group transformation with the exception of factors of two. It gives the result:

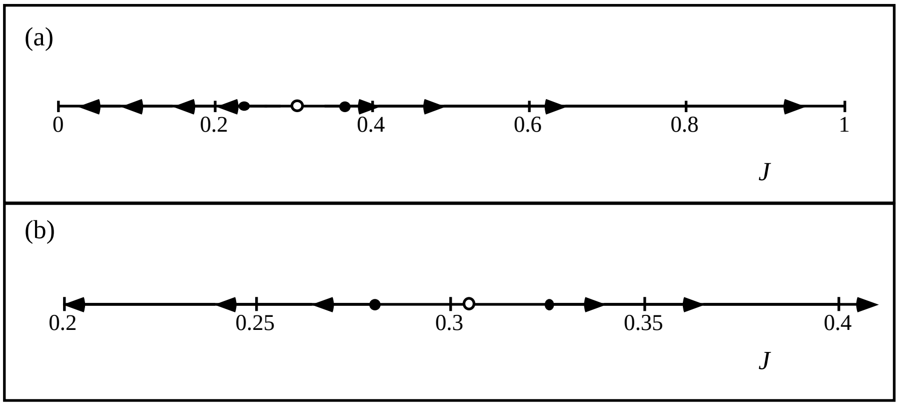


Figure 1.10.11 The two-dimensional Ising model renormalization group transformation obtained from the Migdal-Kadanoff transformation is illustrated as a flow diagram in the one-dimensional parameter space (J). The arrows show the effect of successive iterations starting from the black dots. The white dot indicates the position of the unstable fixed point, J_c, which is the phase transition in this model. Starting from values of J slightly below J_c, iteration results in the model on a large scale becoming decoupled with no interactions between spins ($J \rightarrow 0$). This is the high-temperature phase of the material. However, starting from values of J slightly above J_c iteration results in the model on the large scale becoming strongly coupled ($J \rightarrow \infty$) and spins are aligned. (a) shows only the range of values from 0 to 1, though the value of J can be arbitrarily large. (b) shows an enlargement of the region around the fixed point. ∎

$$\begin{aligned} J' &= (1/2)\ln(\cosh(4J)) \\ c' &= 4c + (1/2)\ln(4\cosh(4J)) \end{aligned} \tag{1.10.39}$$

The renormalization of J in the two-dimensional Ising model turns out to behave qualitatively different from the one-dimensional case. Its behavior is plotted in Fig. 1.10.11 using a flow diagram. There is an unstable fixed point of the iterative map at $J \approx .305$. This nonzero and noninfinite fixed point indicates that we have a phase transition. Reinserting the temperature, we see that the phase transition occurs at $\beta J = .305$ which is significantly larger than the mean field result $\beta zJ = 1$ or $\beta J = .25$ found in Section 1.6. The exact value for the phase transition for this lattice, $\beta J \approx .441$, which can be obtained analytically by other techniques, is even larger.

It turns out that there is a trick that can give us the exact transition point using a similar renormalization transformation. This trick begins by recognizing that we could have moved bonds in a larger square. For a square with b cells on a side, we would end up with each bond on the perimeter being replaced by a bond of strength b. Using Eq.(1.10.30) we can infer that a chain of b bonds of strength bJ gives rise to an effective interaction whose strength is

$$J'(b) = \tanh^{-1}(\tanh(bJ)^b) \tag{1.10.40}$$

The trick is to take the limit $b \rightarrow 1$, because in this limit we are left with the original Ising model. Extending b to nonintegral values by analytic continuation may seem a little strange, but it does make a kind of sense. We want to look at the incremental change in J as a result of renormalization, with b incrementally different from 1. This can be most easily found by taking the hyperbolic tangent of both sides of Eq. (1.10.40), and then taking the derivative with respect to b. The result is:

$$\left.\frac{dJ'(b)}{db}\right|_{b=1} = J + \sinh(J)\cosh(J)\ln(\tanh(J)) \qquad (1.10.41)$$

Setting this equal to zero to find the fixed point of the transformation actually gives the exact result for the phase transition.

The renormalization group gives us more information than just the location of the phase transition. Fig. 1.10.11 shows changes that occur in the parameters as the length scale is varied. We can use this picture to understand the behavior of the Ising model in some detail. It shows what happens on longer length scales by the direction of the arrows. If the flow goes toward a particular point,then we can tell that on the longest (thermodynamic) length scale the behavior will be characterized by the behavior of the model at that point. By knowing how close we are to the original phase transition, we can also learn from the renormalization group what is the length scale at which the behavior characteristic of the phase transition will disappear. This is the length scale at which the iterative map leaves the region of the repelling fixed point and moves to the attracting one.

We can also characterize the relationship between systems at different values of the parameters: temperatures or magnetic fields. Renormalization takes us from a system at one value of βJ to another. Thus, we can relate the behavior of a system at one temperature to another by performing the renormalization for both systems and stopping both at a particular value of βJ. At this point we can directly relate properties of the two systems, such as their free energies. Different numbers of renormalization steps in the two cases mean that we are relating the two systems at different scales. Such descriptions of relationships of the properties of one system at one scale with another system at a different scale are known as scaling functions because they describe how the properties of the system change with scale.

The renormalization group was developed as an analytic tool for studying the scaling properties of systems with spatially arrayed interacting parts. We will study another use of renormalization in Section 1.10.5. Then in Section 1.10.6 we will introduce a computational approach—the multigrid method.

Question 1.10.6 In this section we displayed our iterative maps graphically as flow diagrams, because in renormalization group transformations we are often interested in maps that involve more than one variable. Make a diagram like Fig. 1.1.1 for the single variable J showing the iterative renormalization group transformation for the two-dimensional Ising model as given in Eq. (1.10.39).

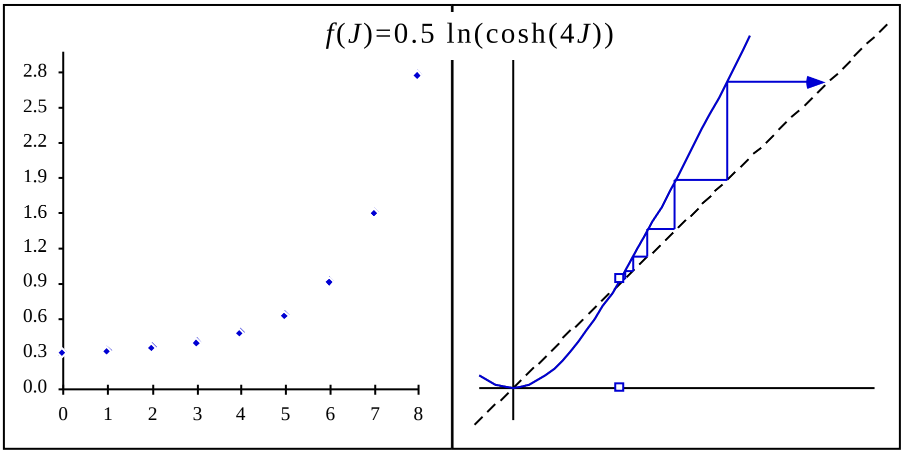


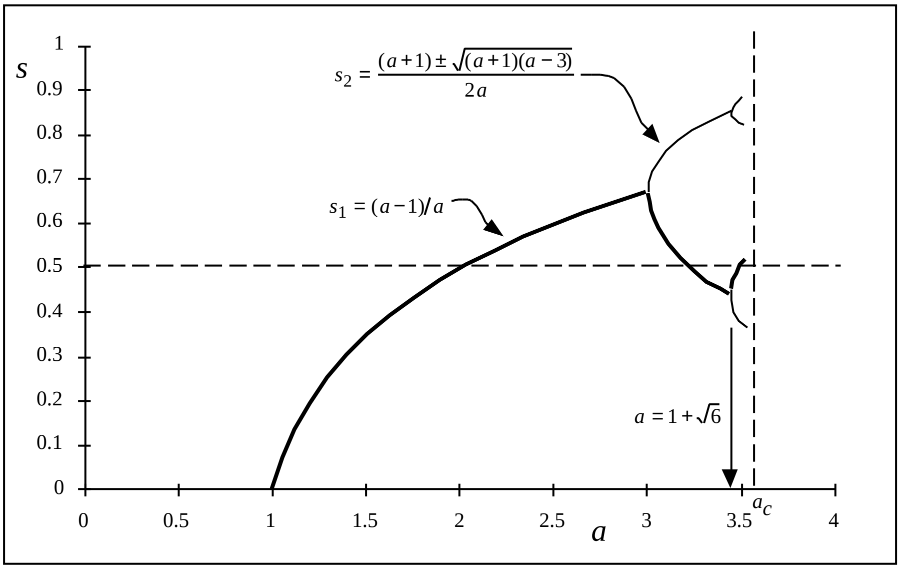
Figure 1.10.12 The iterative map shown as a flow diagram in Fig. 1.10.11 is shown here in the same manner as the iterative maps in Section 1.1. On the left are shown the successive values of J as iteration proceeds. Each iteration should be understood as a loss of detail in the model and hence an observation of the system on a larger scale. Since in general our observations of the system are macroscopic, we typically observe the limiting behavior as the iterations go to ∞. This is similar to considering the limiting behavior of a standard iterative map. On the right is the graphical method of determining the iterations as discussed in Section 1.1. The fixed points are visible as intersections of the iterating function with the diagonal line. ▮

Solution 1.10.6 See Fig. 1.10.12. The fixed point and the iterative behavior are readily apparent. ▮

1.10.5 *Renormalization and chaos*

Our final example of renormalization brings us back to Section 1.1, where we studied the properties of iterative maps and the bifurcation route to chaos. According to our discussion, cycles of length 2^k, k = 0,1,2,..., appeared as the parameter a was varied from 0 to a_c = 3.56994567, at which point chaotic behavior appeared.Fig. 1.1.3 summarizes the bifurcation route to chaos.A schematic of the bifurcation part of this diagram is reproduced in Fig. 1.10.13. A brief review of Section 1.1 may be useful for the following discussion.

The process of bifurcation appears to be a self-similar process in the sense that the appearance of a 2-cycle for $f(s)$ is repeated in the appearance of a 2-cycle for $f^2(s)$, but over a smaller range of a. The idea of self-similarity seems manifest in Fig. 1.10.13, where we would only have to change the scale of magnification in the s and a directions in order to map one bifurcation point onto the next one. While this mapping might not work perfectly for smaller cycles, it becomes a better and better



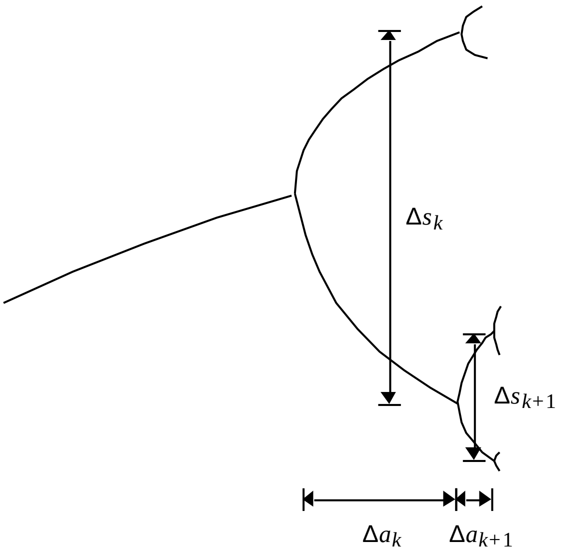


Figure 1.10.13 Schematic reproduction of Fig. 1.1.4, which shows the bifurcation route to chaos. Successive branchings are approximately self-similar. The bottom figure shows the definition of the scaling factors that relate the successive branchings. The horizontal rescaling of the branches, δ, is given by the ratio of Δa_k to Δa_{k+1}. The vertical rescaling of the branches, α, is given by the ratio of Δs_k to Δs_{k+1}. The top figure shows the values from which we can obtain a first approximation to the values of α and δ, by taking the ratios from the zeroth, first and second bifurcations. The zeroth bifurcation point is actually the point $a = 1$. The first bifurcation point occurs at $a = 3$. the second occurs at $a = 1 + \sqrt{6}$. The values of s at the bifurcation points were obtained in Section 1.1, and formulas are indicated on the figure. When the scaling behavior of the tree is analyzed using a renormalization group treatment, we focus on the tree branches that cross $s = 1/2$. These are indicated by bold lines in the top figure. ▮

approximation as the number of cycles increases. The bifurcation diagram is thus a treelike object. This means that the sequence of bifurcation points forms a geometrically converging sequence, and the width of the branches is also geometrically converging. However, the distances in the *s* and *a* directions are scaled by different factors. The factors that govern the tree rescaling at each level are δ and α, as shown in Fig. 1.10.13 (b):

$$\delta = \lim_{k \to \infty} \frac{\Delta a_k}{\Delta a_{k+1}}$$
$$\alpha = \lim_{k \to \infty} \frac{\Delta s_k}{\Delta s_{k+1}} \tag{1.10.42}$$

By this convention,the magnitude of both α and δ is greater than one. α is defined to be negative because the longer branch flips up to down at every branching. The values are to be obtained by taking the limit as $k \to \infty$ where these scale factors have well-defined limits.

We can find a first approximation to these scaling factors by using the values at the first and second bifurcations that we calculated in Section 1.1. These values, given in Fig. 1.10.13, yield:

$$\delta \approx (3-1)/(1+\sqrt{6}-3) = 4.449 \tag{1.10.43}$$

$$\alpha \approx \frac{2s_1\big|_{a=3}}{\left(s_2^+ - s_2^-\right)\Big|_{a=1+\sqrt{6}}} = \frac{4}{3}\frac{a}{\sqrt{(a+1)(a-3)}}\Bigg|_{a=1+\sqrt{6}} = 3.252 \tag{1.10.44}$$

Numerically, the asymptotic value of δ for large *k* is found to be 4.6692016. This differs from our first estimate by only 5%. The numerical value for α is 2.50290787, which differs from our first estimate by a larger margin of 30%.

We can determine these constants with greater accuracy by studying directly the properties of the functions $f, f^2, \ldots f^{2^k} \ldots$ that are involved in the formation of 2^k cycles. In order to do this we modify our notation to explicitly include the dependence of the function on the parameter *a*. $f(s,a), f^2(s,a)$, etc. Note that iteration of the function *f* only applies to the first argument.

The tree is formed out of curves $s_{2^k}(a)$ that are obtained by solving the fixed point equation:

$$s_{2^k}(a) = f^{2^k}(s_{2^k}(a), a) \tag{1.10.45}$$

We are interested in mapping a segment of this curve between the values of *s* where

$$\frac{df^{2^k}(s,a)}{ds} = 1 \tag{1.10.46}$$

and

$$\frac{df^{2^k}(s,a)}{ds} = -1 \tag{1.10.47}$$

onto the next function, where k is replaced everywhere by $k+1$. This mapping is a kind of renormalization process similar to that we discussed in the previous section. In order to do this it makes sense to expand this function in a power series around an intermediate point, which is the point where these derivatives are zero. This is known as the superstable point of the iterative map. The superstable point is very convenient for study, because for any value of k there is a superstable point at $s = 1/2$. This follows because $f(s,a)$ has its maximum at $s = 1/2$, and so its derivative is zero. By the chain rule,the derivative of $f^{2^k}(s,a)$,is also zero. As illustrated in Fig. 1.10.13,the line at $s = 1/2$ intersects the bifurcation tree at every level of the hierarchy at an intermediate point between bifurcation points. These intersection points must be superstable.

It is convenient to displace the origin of s to be at $s = 1/2$,and the origin of a to be at the convergence point of the bifurcations. We thus define a function g which represents the structure of the tree. It is approximately given by:

$$g(s,a) \approx f(s + 1/2, a + a_c) - 1/2 \tag{1.10.48}$$

However, we would like to represent the idealized tree rather than the real tree. The idealized tree would satisfy the scaling relation exactly at all values of a. Thus g should be the analog of the function f which would give us an ideal tree. To find this function we need to expand the region near $a = a_c$ by the appropriate scaling factors. Specifically we define:

$$g(s,a) = \lim_{k\to\infty} \alpha^k f^{2^k}(s/\alpha^k + 1/2, a/\delta^k + a_c) - 1/2 \tag{1.10.49}$$

The easiest way to think about the function $g(s,a)$ is that it is quite similar to the quadratic function $f(s,a)$ but it has the form necessary to cause the bifurcation tree to have the ideal scaling behavior at every branching. We note that $g(s,a)$ depends on the behavior of $f(s,a)$ only very near to the point $s = 1/2$. This is apparent in Eq. (1.10.49) since the region near $s = 1/2$ is expanded by a factor of α^k.

We note that $g(s,a)$ has its maximum at s = 0. This is a consequence of the shift in origin that we chose to make in defining it.

Our objective is to find the form of $g(s,a)$ and, with this form, the values of α and δ. The trick is to recognize that what we need to know can be obtained directly from its scaling properties. To write the scaling properties we look at the relationship between successive iterations of the map and write:

$$g(s,a) = \alpha g^2(s/\alpha, a/\delta) \tag{1.10.50}$$

This follows either from our discussion and definition of the scaling parameters α and δ or directly from Eq. (1.10.49).

For convenience, we analyze Eq. (1.10.50) first in the limit $a \to 0$. This corresponds to looking at the function $g(s,a)$ as a function of s at the limit of the bifurcation sequence. This function (Fig. 1.10.14) still looks quite similar to our original function $f(s)$, but its specific form is different. It satisfies the relationship:

$$g(s,0) = g(s) = \alpha g^2(s/\alpha) \tag{1.10.51}$$

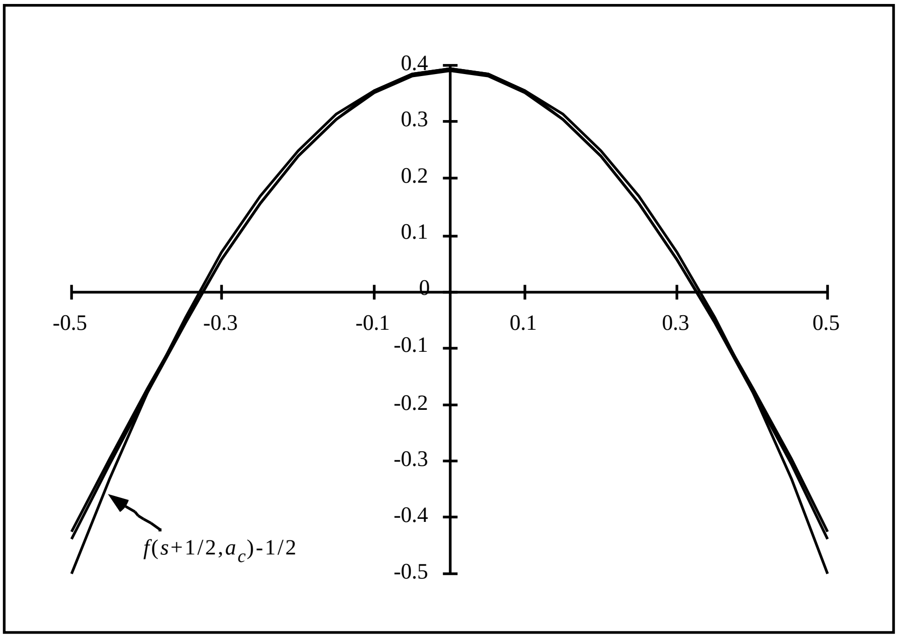


Figure 1.10.14 Three functions are plotted that are successive approximations to $g(s) = g(s, 0)$. This function is essentially the limiting behavior of the quadratic iterative map $f(s)$ at the end of the bifurcation tree a_c. The functions plotted are the first three k values inserted in Eq. (1.10.49): $f(s + 1/2, a + a_c) - 1/2$, $af^2(s/\alpha + 1/2, a_c) - 1/2$ and $a^2f^4(s/\alpha^2 + 1/2, a_c) - 1/2$. The latter two are almost indistinguishable, indicating that the sequence of functions converges rapidly. ▮

We approximate this function by a quadratic with no linear term because $g(s)$ has its maximum at $s = 0$:

$$g(s) \approx g_0 + cs^2 \tag{1.10.52}$$

Inserting into Eq. (1.10.51) we obtain:

$$g_0 + cs^2 \approx \alpha(g_0 + c(g_0 + c(s / \alpha)^2)^2) \tag{1.10.53}$$

Equating separately the coefficients of the first and second terms in the expansion gives the solution:

$$\begin{aligned} \alpha &= 1 / (1 + cg_0) \\ \alpha &= 2cg_0 \end{aligned} \tag{1.10.54}$$

We see that c and g_0 only appear in the combination cg_0, which means that there is one parameter that is not determined by the scaling relationship. However, this does not prevent us from determining α. Eq. (1.10.54) can be solved to obtain:

$$\begin{aligned} cg_0 &= (-1 \pm \sqrt{3})/2 = -1.3660254 \\ \alpha &= (-1 \pm \sqrt{3}) = -2.73205081 \end{aligned} \tag{1.10.55}$$

We have chosen the negative solutions because the value of α and the value of cg_0 must be negative.

We return to consider the dependence of $g(s,a,)$ on a to obtain a new estimate for δ. Using a first-order linear dependence on a we have:

$$g(s,a,) \approx g_0 + cs^2 + ba \tag{1.10.56}$$

Inserting into Eq. (1.10.50) we have:

$$g_0 + cs^2 + ba \approx \alpha(g_0 + c(g_0 + c(s / \alpha)^2 + ba / \delta)^2 + ba / \delta) \tag{1.10.57}$$

Taking only the first order term from this equation in a we have:

$$\delta = 2\alpha cg_0 + \alpha = 4.73205 \tag{1.10.58}$$

Eq. (1.10.55) and Eq.(1.10.58) are a significant improvement over Eq.(1.10.44) and Eq.(1.10.43). The new value of α is less than 10% from the exact value. The new value of δ is less than 1.5% from the exact value. To improve the accuracy of the results, we need only add additional terms to the expansion of $g(s,a)$ in s. The first-order term in a is always sufficient to obtain the corresponding value of δ.

It is important, and actually central to the argument in this section, that the explicit form of $f(s,a)$ never entered into our discussion. The only assumption was that the functional behavior near the maximum is quadratic. The rest of the argument follows independent of the form of $f(s,a)$ because we are looking at its properties after many iterations. These properties depend only on the region right in the vicinity of the maximum of the function. Thus only the first-order term in the expansion of the original function $f(s,a)$ matters. This illustrates the notion of universality so essential to the concept of renormalization—the behavior is controlled by very few parameters. All other parameters are irrelevant—changing their values in the original iterative map is irrelevant to the behavior after many iterations (many renormalizations) of the iterative map. This is similar to the study of renormalization in models like the Ising model, where most of the details of the behavior at small scales no longer matter on the largest scales.

1.10.6 *Multigrid*

The multigrid technique is designed for the solution of computational problems that benefit from a description on multiple scales. Unlike renormalization, which is largely an analytic tool,the multigrid method is designed specifically as a computational tool. It works well when a problem can be approximated using a description on a coarse lattice, but becomes more and more accurate as the finer-scale details on finer-scale lattices are included. The idea of the method is not just to solve an equation on finer and finer levels of description, but also to correct the coarser-scale equations based on the finer-scale results. In this way the methodology creates an improved description of the problem on the coarser-scale.

The multigrid approach relies upon iterative refinement of the solution. Solutions on coarser scales are used to approximate the solutions on finer scales. The finer-scale solutions are then iteratively refined. However, by correcting the coarser-scale equations,it is possible to perform most of the iterative refinement of the fine-scale solution on the coarser scales. Thus the iterative refinement of the solution is based both upon correction of the solution and correction of the equation. The idea of correcting the equation is similar in many ways to the renormalization group approach. However, in this case it is a particular solution, which may be spatially dependent, rather than an ensemble averaging process, which provides the correction.

We explain the multigrid approach using a conventional problem, which is the solution of a differential equation. To solve the differential equation we will find an approximate solution on a grid of points.Our ultimate objective is to find a solution on a fine enough grid so that the solution is within a prespecified accuracy of the exact answer. However, we will start with a much coarser grid solution and progressively refine it to obtain more accurate results. Typically the multigrid method is applied in two or three dimensions, where it has greater advantages than in one dimension. However, we will describe the concepts in one dimension and leave out many of the subtleties.

For concreteness we will assume a differential equation which is:

$$\frac{d^2 f(x)}{dx^2} = g(x) \tag{1.10.59}$$

where $g(x)$ is specified. The domain of the equation is specified, and boundary conditions are provided for $f(x)$ and its derivative.On a grid of equally spaced points we might represent this equation as:

$$\frac{1}{d^2}\left(f(i+1) + f(i-1) - 2f(i)\right) = g(i) \tag{1.10.60}$$

This can be written as a matrix equation:

$$\sum_j A(i,j)\, f(j) = g(i) \tag{1.10.61}$$

The matrix equation can be solved for the values of $f(i)$ by matrix inversion (using matrix diagonalization). However, diagonalization is very costly when the matrix is large, i.e., when there are many points in the grid.

A multigrid approach to solving this equation starts by defining a set of lattices (grids), $L_j, j \quad \{0,\ldots,q\}$, where each successive lattice has twice as many points as the previous one (Fig. 1.10.15). To explain the procedure it is simplest to assume that we start with a good approximation for the solution on grid L_{j-1} and we are looking for a solution on the grid L_j. The steps taken are then:

1. Interpolate to find $f_0^j(i)$, an approximate value of the function on the finer grid L_j.

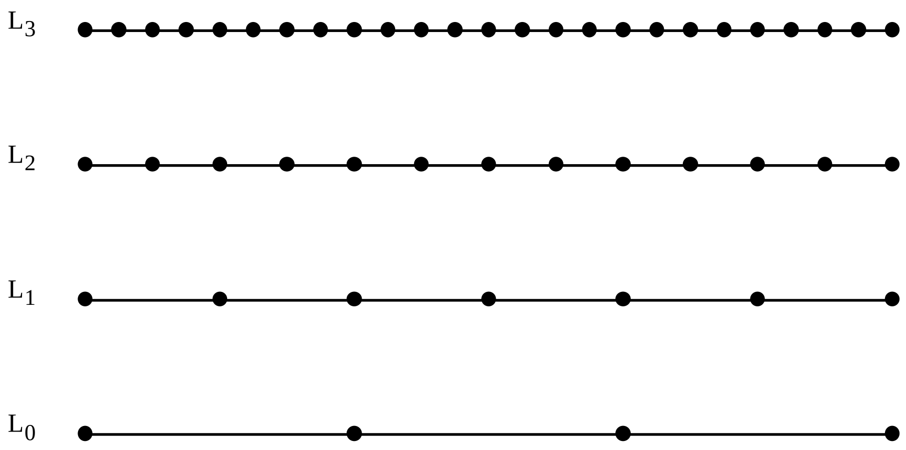


Figure 1.10.15 Illustration of four grids for a one-dimensional application of the multigrid technique to a differential equation by the procedure illustrated in Fig. 1.10.16. ▮

2. Perform an iterative improvement (relaxation) of the solution on the finer grid. This iteration involves calculating the error

$$\sum_{i'} A(i,i')\, f_0^j(i') - g(i) = r^j(i) \tag{1.10.62}$$

where all indices refer to the grid L_j. This error is used to improve the solution on the finer grid, as in the minimization procedures discussed in Section 1.7.5:

$$f_1^j(i) = f_0^j(i) - c r^j(i) \tag{1.10.63}$$

The scalar c is generally replaced by an approximate inverse of the matrix $A(i,j)$ as discussed in Section 1.7.5. This iteration captures much of the correction of the solution at the fine-scale level; however, there are resulting corrections at coarser levels that are not captured. Rather than continuing to iteratively improve the solution at this fine-scale level, we move the iteration to the next coarser level.

3. Recalculate the value of the function on the coarse grid L_{j-1} to obtain $f_1^{j-1}(i)$. This might be just a restriction from the fine-grid points to the coarse-grid points. However, it often involves some more sophisticated smoothing. Ideally, it should be such that interpolation will invert this process to obtain the values that were found on the finer grid. The correction for the difference between the interpolated and exact fine-scale results are retained.

4. Correct the value of $g(i)$ on the coarse grid using the values of $r^j(i)$ restricted to the coarser grid. We do this so that the coarse-grid equation has an exact solution that is consistent with the fine-grid equation. From Eq. (1.10.62) this essentially means adding $r^j(i)$ to $g(i)$. The resulting corrected values we call $g_1^{j-1}(i)$.

5. Relax the solution $f_1^{j-1}(i)$ on the coarse grid to obtain a new approximation to the function on the coarse grid $f_2^{j-1}(i)$. This is done using the same procedure for relaxation described in step 3; however $g(i)$ is replaced by $g_1^{j-1}(i)$.

   The procedure of going to coarser grids in steps 3 through 5 is repeated for all grids $L_{j-2}, L_{j-3}, \ldots$ till the coarsest grid, L_0. The values of the function $g(i)$ are progressively corrected by the finer-scale errors. Step 5 on the coarsest grid is replaced by exact solution using matrix diagonalization. The subsequent steps are designed to bring all of the iterative refinements to the finest-scale solution.

6. Interpolate the coarse-grid solution L_0 to the finer-grid L_1.

7. Add the correction that was previously saved when going from the fine to the coarse grid.

   Steps 6–7 are then repeated to take us to progressively finer-scale grids all the way back to L_j.

This procedure is called a V-cycle since it appears as a V in a schematic that shows the progressive movement between levels. A V-cycle starts from a relaxed solution on grid L_{j-1} and results in a relaxed solution on the grid L_j. A full multigrid procedure involves starting with an exact solution at the coarsest scale L_0 and then performing V-cycles for progressively finer grids. Such a multigrid procedure is graphically illustrated in Fig. 1.10.16.

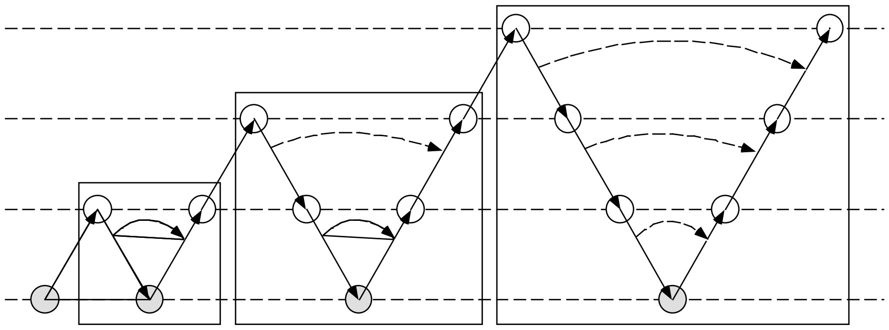


Figure 1.10.16 The multigrid algorithm used to obtain the solution to a differential equation on the finest grid is described schematically by this sequence of operations. The operation sequence is to be read from left to right. The different grids that are being used are indicated by successive horizontal lines with the coarsest grid on the bottom and the finest grid on the top. The sequence of operations starts by solving a differential equation on the coarsest grid by exact matrix diagonalization (shaded circle). Then iterative refinement of the equations is performed on finer grids. When the finer-grid solutions are calculated, the coarse-grid equations are corrected so that the iterative refinement of the fine-scale solution can be performed on the coarse grids. This involves a V-cycle as indicated in the figure by the boxes. The horizontal curved arrows indicate the retention of the difference between coarse- and fine-scale solutions so that subsequent refinements can be performed. ∎

There are several advantages of the multigrid methodology for the solution of differential equations over more traditional integration methods that use a single-grid representation. With careful implementation, the increasing cost of finer-scale grids grows slowly with the number of grid points, scaling as $N \ln(N)$. The solution of multiple problems of similar type can be even more efficient,since the corrections of the coarse-scale equations can often be carried over to similar problems, accelerating the iterative refinement. This is in the spirit of developing universal coarse-scale representations as discussed earlier. Finally, it is natural in this method to obtain estimates of the remaining error due to limited grid density, which is important to achieving a controlled error in the solution.

1.10.7 *Levels of description, emergence of simplicity and complexity*

In our explorations of the world we have often discovered that the natural world may be described in terms of underlying simple objects, concepts, and laws of behavior (mechanics) and interactions. When we look still closer we see that these simple objects are composite objects whose internal structure may be complex and have a wealth of possible behavior. Somehow, the wealth of behavior is not relevant at the larger scale. Similarly, when we look at longer length scales than our senses normally are attuned to, we discover that the behavior at these length scales is not affected by objects and events that appear important to us.

Examples are found from the behavior of galaxies to elementary particles: galaxies are composed of suns and interstellar gases, suns are formed of complex plasmas and are orbited by planets, planets are formed from a diversity of materials and even life, materials and living organisms are formed of atoms,atoms are composed of nuclei and electrons, nuclei are composed of protons and neutrons (nucleons),and nucleons appear to be composed of quarks.

Each of these represents what we may call a level of description of the world. A level is an internally consistent picture of the behavior of interacting elements that are simple. When taken together, many such elements may or may not have a simple behavior, but the rules that give rise to their collective behavior are simple. We note that the interplay between levels is not always just a self-contained description of one level by the level immediately below. At times we have to look at more than one level in order to describe the behavior we are interested in.

The existence of these levels of description has led science to develop the notion of fundamental law and unified theories of matter and nature. Such theories are the self-consistent descriptions of the simple laws governing the behavior and interplay of the entities on a particular level. The laws at a particular level then give rise to the larger-scale behavior.

The existence of simplicity in the description of underlying fundamental laws is not the only way that simplicity arises in science. The existence of multiple levels implies that simplicity can also be an emergent property. This means that the collective behavior of many elementary parts can behave simply on a much larger scale.

The study of complex systems focuses on understanding the relationship between simplicity and complexity. This requires both an understanding of the emergence of complex behavior from simple elements and laws, as well as the emergence of simplicity from simple or complex elements that allow a simple larger-scale description to exist.

Much of our discussion in this section was based upon the understanding that macroscopic behavior of physical systems can be described or determined by only a few relevant parameters. These parameters arise from the underlying microscopic description. However, many of the aspects of the microscopic description are irrelevant. Different microscopic models can be used to describe the same macroscopic phenomenon. The approach of scaling and renormalization does not assume that all the details of the microscopic description become irrelevant, however, it tries to determine self-consistently which of the microscopic parameters are relevant to the macroscopic behavior in order to enable us to simplify our analysis and come to a better understanding.

Whenever we are describing a simple macroscopic behavior, it is natural that the number of microscopic parameters relevant to model this behavior must be small. This follows directly from the simplicity of the macroscopic behavior. On the other hand,if we describe a complex macroscopic behavior, the number of microscopic parameters that are relevant must be large.

Nevertheless, we know that the renormalization group approach has some validity even for complex systems. At long length scales, all of the details that occur on the smallest length scale are not relevant. The vibrations of an individual atom are not generally relevant to the behavior of a complex biological organism. Indeed, there is a pattern of levels of description in the structure of complex systems. For biological organisms, composed out of atoms, there are additional levels of description that are intermediate between atoms and the organism: molecules, cells, tissues, organs and systems. The existence of these levels implies that many of the details of the atomic behavior are not relevant at the macroscopic level. This should also be understood from the perspective of the multi-grid approach. In this picture, when we are describing the behavior of a complex system, we have the possibility of describing it at a very coarse level or a finer and yet finer level. The number of levels that are necessary depends on the level of precision or level of detail we wish to achieve in our description. It is not always necessary to describe the behavior in terms of the finest scale. It is essential, however, to identify properly a model that can capture the essential underlying parameters in order to discuss the behavior of any system.

Like biological organisms, man-made constructs are also built from levels of structure. This method of organization is used to simplify the design and enable us to understand and work with our own creations. For example, we can consider the construction of a factory from machines and computers,machines constructed from individual moving parts, computers constructed from various components including computer chips, chips constructed from semiconductor devices, semiconductor devices composed out of regions of semiconductor and metal. Both biology and engineering face problems of design for function or purpose. They both make use of interacting building blocks to engineer desired behavior and therefore construct the complex out of the simple. The existence of these building blocks is related to the existence of levels of description for both natural and artificial systems.

Our discussion thus brings us to recognize the importance of studying the properties of substructure and its relationship to function in complex systems. This relationship will be considered in Chapter 2 in the context of our study of neural networks.