1

Introduction to the Theory of Information

'Do not worry about your difficulties in Mathematics. I can assure you mine are still greater.'
Albert Einstein

1.1 Introduction

Information processing is the most important and the most energy consuming human activity. Our brains contain approximately 3×10^9 neurons and each of them has approximately 10^4 connections with other neurons. This impressive network is dense, as each cubic millimetre of neural tissue contains up to 10^9 synaptic junctions. While the brain constitutes only 2% of body mass, is consumes 20% of the energetic demand at rest. We really must need our personal 'CPUs' as Mother Nature invests so much energy in nervous systems. The importance of information storage, transfer and processing has been greatly appreciated through the ages. Nowadays various techniques have revolutionized all aspects of information processing via digital electronic technologies.

It is impossible to find a direct relation between brains and computers, but both systems show some functional and structural analogies. Their building blocks are relatively simple and operate according to well-defined rules, the complex functions they can perform is a result of the structural complexity (i.e. is an emergent feature of the system) and communication between structural elements is digital. This is quite obvious for electronic computers, but spikes of *action potential* can also be regarded as digital

Infochemistry: Information Processing at the Nanoscale, First Edition. Konrad Szaciłowski.

signals, as it is not the amplitude of the signal, but the sequence of otherwise identical pulses that carries information.

1.2 Definition and Properties of Information

We all intuitively use and understand the notion of *information*, but it defies precise definition. The concept of information has many meanings, depending on the context. It is usually associated with language, data, knowledge or perception, but in thermodynamics it is a notion closely related to entropy. Its technical definition is usually understood to be an ordered sequence of symbols. Information can be also regarded as any kind of sensory input for humans, animals, plants and artificial devices. It should carry a pattern that influences the interaction of the system with other sensory inputs or other patterns. This definition separates information from consciousness, as interaction with patterns (or pattern circulation) can take place in unanimated systems as well.

While the psychological definition of information is ambiguous, the technological applications must be based on strict definitions and measures. Information can be regarded as a certain physical or structural feature of any system. It can be understood as a degree of order of any physical system. This (structural) form of information is usually regarded as a third (along with matter and energy) component of the Universe. Every object, phenomenon or process can be described in terms of matter (type and number of particles), energy (physical movements) and information (structure). In other words, information can be another manifestation of a primary element. In the same way that the special theory of relativity expresses the equivalence of mass and energy (1.1) [1],

$$E = mc^2, \tag{1.1}$$

the equivalence of energy and information can be shown within information theory (*vide infra*).

The most precise definition of information is given by the *syntactic theory* of Hartley and Shannon. According to this theory, information is a measure of the probability of a certain event. It is the amount of uncertainty removed on occurrence of an event or data transmission. The less probable the event, the higher its information value. According to Hartley the amount of information (I_i) given by an event x_i can be formulated as (1.2) [2,3]:

$$I_i = \log_r \frac{1}{p_i}, \tag{1.2}$$

where p_i denotes the probability of an event x_i and r is the base of logarithm. Such an expressed amount of information is also a measure of the entropy associated with the event x_i. The average amount of information carried by an event from a set of events (X) is the weighted average of entropies of all the events within this set (1.3):

$$H(X) = \sum_{i=1}^{n} p_i \log_r \frac{1}{p_i} = -\sum_{i=1}^{n} p_i \log_r p_i \tag{1.3}$$

This definition automatically defines the unit of information. Depending on the logarithm base the basic information units are bit ($r = 2$), nit ($r = e$) and dit ($r = 10$). With $r = 2$ the information content of the event is measured as the number of binary digits necessary to describe it, provided there is no redundancy in the message.

The average amount of information in the system is related to the system entropy by (1.4):

$$S = -k_B \sum_{i=1}^{n} p_i \log p_i, \tag{1.4}$$

where k_B is the Boltzmann constant. As derived by Landauer, the energetic equivalent binary transition of one bit at $T = 298$ K amounts to (1.5) [4]:

$$E = k_B T \ln 2 \approx 2.8 \times 10^{-21}\ \mathrm{J} \cdot \mathrm{bit}^{-1} \tag{1.5}$$

More precisely, this is the minimum amount of energy that must be dissipated on erasure of one bit of information. This calculation, initially based on the *Second Law of Thermodynamics* was later generalized on the basis of the *Fokker–Planck equation* concerning the simplest memory model and a Brownian motion particle in a *potential well* [5]. The same value has also been derived microscopically without direct reference to the Second Law of Thermodynamics for classical systems with continuous space and time and with discrete space and time, and for a quantum system [6]. Interestingly, exactly the same value is obtained as the energetic limit required for switching of a single binary switch [7]. Any physical system which can be regarded as a binary switch must exist in two stable, equienergetic states separated by an energy barrier (Figure 1.1a). The barrier must be high enough to prevent thermal equilibration of these two distinct states. At any temperature, mechanical vibration of atoms and the thermal electromagnetic field may induce spontaneous switching between the states.

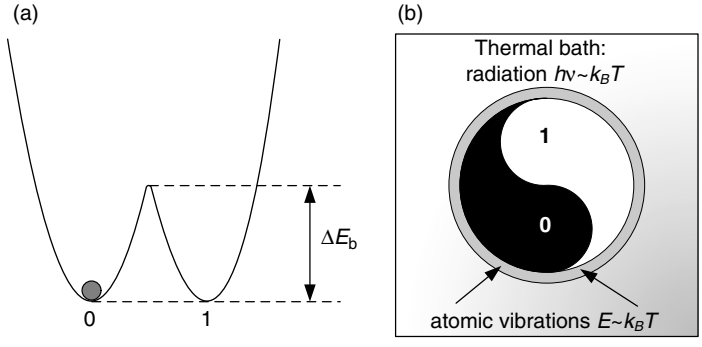


Figure 1.1 *Energy diagram of a binary switch (a) (Adapted from [5] Copyright (1995) American Physical Society) and schematic representation of a physical system corresponding to a binary switch immersed in a thermal bath with two components: radiation and atomic vibrations (b). (Adapted from [7] Copyright (2006) Springer Science + Business Media.)*

The probability of this spontaneous process can be quantified by the error probability Π_{err} obtained from the Bolzmann distribution (1.6) [7]:

$$\Pi_{err} = \exp\left(-\frac{\Delta E_b}{k_B T}\right) \tag{1.6}$$

The minimum barrier height (below this value the two states are indistinguishable) can be calculated assuming the error probability is equal to 0.5. The value of the limiting energy thus obtained for a binary switching process is identical to the energetic equivalent of one bit of information (1.7).

$$\Delta E_{b,\min} = k_B T \ln 2 \tag{1.7}$$

At very low temperatures ($T \to 0$ K), however, the *Landauer principle* is not valid because of *quantum entanglement* [8]. Recently measured energetic demand for single bit processing in conventional computers (Pentium II, 400 MHz) amounts to $\sim 8.5 \times 10^{-11}$ J bit^{-1} [9]. The combination of Equations (1.1) and (1.5) yields a mass equivalent of information amounting to $\sim 3 \times 10^{-38}$ kg bit^{-1}.

The above definition of information assumes a finite number of distinct events (e.g. transmission of characters) and so may somehow represent a digitalized form of information. The digital representation of information, along with *Boolean logic* (*vide infra*) constitutes the theoretical basis for all contemporary computing systems, excluding the quantum approach.

The strict mathematical definition of information concerns only information in the sense of a stream of characters and other signals, and is not related to its meaning. There are, however three distinct levels at which information may have different meanings. According to Charles S. Pierce and Charles W. Morris we can define three levels of information: *syntactic* level, *semantic* level and *pragmatic* level. Information on the *syntactic level* is concerned with the formal relation between the elements of information, the rules of corresponding language, the capacity of communication channels and the design of coding systems for information transmission, processing and storage. The meaning of information and its practical meaning are neglected at this level. The *semantic level* relates information to its meaning, and semantic units (words and groups of words) are assigned more or less precisely to their meaning. For correct information processing at the syntactic level semantics are not necessary. On the *pragmatic level* the information is related to its practical value. It strongly depends on the context and may be of economical, political or psychological importance. Furthermore, at the pragmatic level the information value is time dependent and its practical value decreases with time, while correct prediction of future information may be of high value [10,11].

1.3 Principles of Boolean Algebra

One of the definitions of the amount of information (Equation 1.2) in the case of $r = 2$ implies that the total information contained in a system or event can be expressed using two symbols, for example binary digits. This situation is related to *prepositional calculus*,

where any sentence has an attributed logic value: TRUE or FALSE. Therefore *Boolean algebra* based on a two-element set and simple operators can be used for any information processing. Unlike algebra, Boolean algebra does not deal with real numbers, but with the notions of truth and falsehood. These notions, however, are usually assigned symbols of 0 and 1, respectively. This is the most common symbolic representation, but others (e.g. $\bot$, $\top$; TRUE, FALSE) are also in use. Furthermore, the numerical operations of multiplication, addition and negation are replaced with the logic operations of *conjunction* ($\wedge$, AND, logic product), *disjunction* ($\vee$, OR, logic sum) and *complement* ($\neg$, NOT). Interestingly, the same structure would have algebra of the integers modulo 2; these two algebras are fully equivalent [12,13]. The operation can be easily defined if Boolean algebra is understood as the algebra of sets, where 0 represents an empty set and 1 a complete set. Then, conjunction is equivalent to the intersection of sets and disjunction to the union of sets, while complement is equivalent to the complement of a set [10]. These operations can be simply illustrated using *Venn diagrams* (Figure 1.2).

Conjunction in Boolean algebra has exactly the same properties as multiplication in algebra. If any of the arguments of the operation is 0 (i.e. FALSE) the operation yields 0, while if both arguments are equal to 1, the result of conjunction is also 1. Disjunction, unlike addition, yields 1 if both arguments are unity, while in other cases its properties are similar to addition. The properties of the complement operation can be described as follows (1.8), (1.9):

$$x \wedge \neg x = 0 \tag{1.8}$$

$$x \vee \neg x = 1 \tag{1.9}$$

Put simply, this operation exchanges the two Boolean values, that is $\neg 0 = 1$ and $\neg 1 = 0$. Therefore the double complement yields the initial logic value (1.10):

$$\neg\neg x = x \tag{1.10}$$

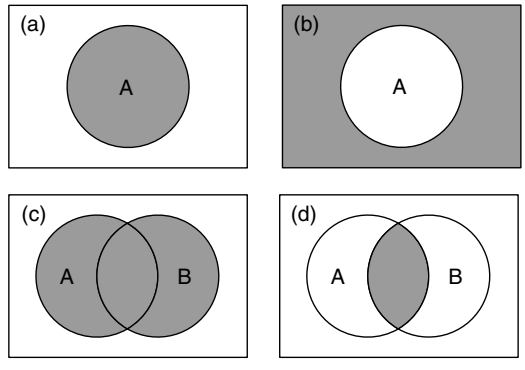


Figure 1.2 *Venn diagrams of set A (a), its complement (b), and the union (disjunction) (c) and intersection (conjunction) (d) of two sets, A and B.*

Boolean algebra is based on a set of axioms: *associativity*, *commutativity*, *distributivity*, *absorption* and *idempotence*. Furthermore, it assumes the existence of *neutral element*s and *annihilator element*s for binary operators.

The associativity rule states that the grouping of the variables in disjunction and conjunction operations does not change the result (1.11), (1.12).

$$(a \vee b) \vee c = a \vee (b \vee c) \tag{1.11}$$

$$(a \wedge b) \wedge c = a \wedge (b \wedge c) \tag{1.12}$$

Moreover, the operations of disjunction and conjunction are *commutative*, that is the result does not depend on the order of arguments (1.13), (1.14):

$$a \vee b = b \vee a \tag{1.13}$$

$$a \wedge b = b \wedge a \tag{1.14}$$

Both operations are *distributive* over the other one, that is (1.15), (1.16):

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \tag{1.15}$$

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \tag{1.16}$$

While the first distributivity law (1.15) is rather intuitive and true also in ordinary algebra, the other (1.16) is true only in Boolean algebra.

The *absorption law* is an identity linking a pair of binary operations (1.17):

$$a \wedge (a \vee b) = a \vee (a \wedge b) = a \tag{1.17}$$

Therefore Boolean algebra with two elements (0 and 1) and two commutative and associative operators ($\vee$ and $\wedge$), which are connected by the absorption law, is a lattice. In every lattice the following relation is always fulfilled (1.18):

$$a \vee b = b \Leftrightarrow a \wedge b = a \tag{1.18}$$

Therefore the ordering relation '$\leq$' can be defined as follows (1.19):

$$a \leq b \Leftrightarrow a \vee b = b \tag{1.19}$$

The $\vee$ and $\wedge$ operators can be defined as *infimum* and *supremum* of sets of arguments, respectively (1.20), (1.21):

$$a \wedge b \equiv \inf(a, b) \tag{1.20}$$

$$a \vee b \equiv \sup(a, b) \tag{1.21}$$

While in binary logic this is quite intuitive, this analysis is necessary to understand basic binary operators in ternary and higher logic systems (*vide infra*). The binary Boolean

operators are *idempotent*, that is when applied (many times) to one logic variable, its logic value is preserved (1.22), (1.23):

$$a \wedge a = a \tag{1.22}$$

$$a \vee a = a \tag{1.23}$$

For each binary operator there exists a neutral element, which does not change the value of the logic variable. For disjunction this element is 0, while for conjunction it is 1 (1.24), (1.25):

$$a \vee 0 = a \tag{1.24}$$

$$a \wedge 1 = a \tag{1.25}$$

*Annihilator*s are the elements that destroy information contained in Boolean variables (1.26), (1.27):

$$a \wedge 0 = 0 \tag{1.26}$$

$$a \vee 1 = 1 \tag{1.27}$$

Boolean algebra is dual, the exchange of 0 and 1 or $\wedge$ and $\vee$ operators will also result in also a Boolean algebra, but with different properties. Concomitant interchange of values and operators, however, yields the same algebra. This so-called *De Morgan duality* can be formulated as follows (1.28), (1.29):

$$(\neg a) \vee (\neg b) = \neg(a \wedge b) \tag{1.28}$$

$$(\neg a) \wedge (\neg b) = \neg(a \vee b) \tag{1.29}$$

De Morgan duality has very important practical consequences. It allows construction of any Boolean function from only two operators: $\neg$, and either $\vee$ or $\wedge$. This is especially important in electronics, because the realization of any combinatorial function requires only a limited number of building blocks (e.g. all binary functions can be achieved via combination of several NAND gates, *vide infra*).

1.4 Digital Information Processing and Logic Gates

1.4.1 Simple Logic Gates

The simple rules discussed in preceding sections allow any binary logic operations to be performed, and all the complex logic functions can be produced using the basic set of functions: *OR*, *AND* and *NOT*. It is not important whether the information is encoded as electric signals (classical electronics), light pulses (*photonic*s) or mechanical movements. The only important issues are the distinguishability of signals assigned to logic values,

and the principles of Boolean algebra. Any physical system whose state can be described as a Boolean function of input signals (also Boolean in nature) is a *logic gate*. Therefore it is not important if the signals are of electrical, mechanical, optical or chemical nature [14]. Information can be represented by transport of electric charge (classical electronics), ionic charge (electrochemical devices), mass, electromagnetic energy and so on. Furthermore, the actual state of any physical system can be also regarded as a representation of information, for example electrical charge, *spin orientation*, *magnetic flux quantum*, phase of an electromagnetic wave, chemical structure or mechanical geometry [15]. Usually the term 'logic gate', however, is associated with electronic devices capable of performing Boolean operations on binary variables.

There are two types of one-input electronic logic gates: *YES* (also called a buffer) and *NOT*. The *YES gate* transfers the unchanged signal from the input to the output, while the NOT gate computes its complement (Table 1.1).

There are 16 possible combinations of binary two-input logic gates, but only eight of them have any practical meaning. These include OR, AND and *XOR*, as well as their combinations with NOT: *NOR*, *NAND*, *XNOR*, *INH* and *IMP* (Table 1.2).

The OR gate is one of the basic gates from which all other functions can be constructed. The *OR gate* produces high output when any of the inputs is in the high state and the output is low when all the inputs are in the low state. Therefore the gate detects any high state at any of the inputs. It computes the logic sum of input variables, that is it performs the disjunction operation.

The AND gate is another of the principal logic gates, it has two or more inputs and one output. The *AND gate* produces high output (logical 1) only when all the inputs are in the high state. If any of the inputs is in the low state the output is also low (Figure 1.6b). The main role of the gate is to determine if the input signals are simultaneously true. Other words, it performs the conjunction operation or computes the logic product of input variables.

A more complex logic function is performed by exclusive-OR (XOR) gate. This is not a fundamental gate, but it is actually formed by a combination of the gates described above (usually four NAND gates). However, due to its fundamental importance in numerous applications, this gate is treated as a basic logic element and it has been assigned a unique

Table 1.1 *Truth tables, symbols and Venn diagrams for YES and NOT binary logic gates.*

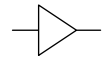
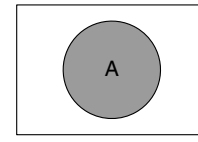
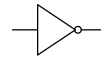
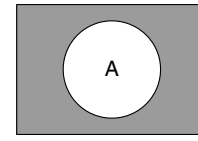
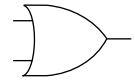
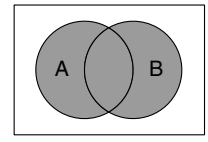
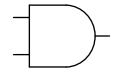
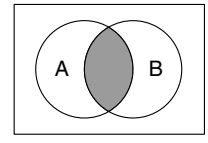
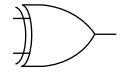
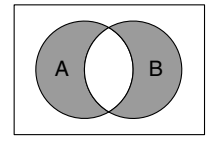
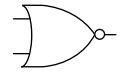
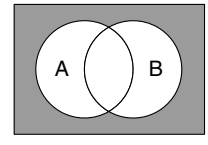
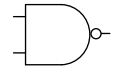
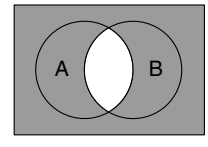
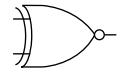
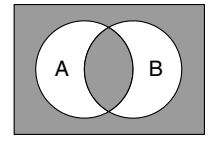
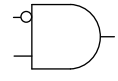
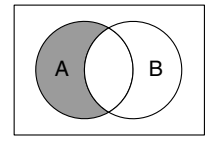
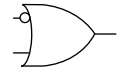
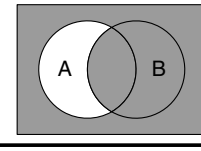
Name	Input	Output	Symbol	Venn diagram
YES	0	0		
	1	1		
NOT	0	1		
	1	0		

Table 1.2 *Truth tables, symbols and Venn diagrams for two-input binary logic gates.*

Name	Input A	Input B	Output	Symbol	Venn diagram
OR	0	0	0		
	1	0	1		
	0	1	1		
	1	1	1		
AND	0	0	0		
	1	0	0		
	0	1	0		
	1	1	1		
XOR	0	0	0		
	1	0	1		
	0	1	1		
	1	1	0		
NOR	0	0	1		
	1	0	0		
	0	1	0		
	1	1	0		
NAND	0	0	1		
	1	0	1		
	0	1	1		
	1	1	0		
XNOR	0	0	1		
	1	0	0		
	0	1	0		
	1	1	1		
INH	0	0	0		
	1	0	1		
	0	1	0		
	1	1	0		
IMP	0	0	1		
	1	0	0		
	0	1	1		
	1	1	1		

symbol (⊗). The *XOR gate* yields a high output when the two input values are different, but yields a low output when the input signals are identical. The main application of the XOR gate is in a *binary half-adder*, a simple electronic circuit enabling transition from Boolean logic to arithmetic.

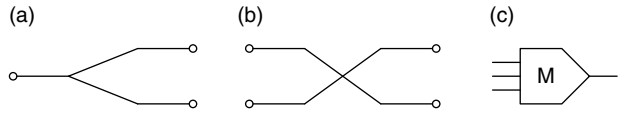


Figure 1.3 *Schematics of FAN-OUT (a), SWAP (b) and MAJORITY (c) gates.*

The whole family of logic gates is formed by the concatenation of OR, AND and XOR gates with the NOT gate, which can be connected to the input or output of any of the above gates. The various connection modes and resulting gates are presented in Table 1.2. Gates resulting from concatenation with NOT are obviously not basic gates, but due to their importance they are usually treated as fundamental logic gates together with NOT, OR and AND logic gates.

Along with fundamental and NOT-concatenated devices (Table 1.2) there are several other devices which are not fundamental (or cannot even be described in terms of Boolean logic), but are important for construction of both electronic and non-classical logic devices.

The *FAN-OUT* operation drives a signal transmitted through one line onto several lines, thus directing the same information into several outputs (Figure 1.3a). A *SWAP gate* (Figure 1.3b) is a two-input two-output device; it interchanges values transmitted through two parallel lines. This device is especially interesting from the point of view of reversible computation, as it is an element of the *Fredkin gate* (*vide infra*). While the FAN-OUT and SWAP operations are seen to be extremely simple devices in electronic implementations (forked connectors and crossed insulated connectors, respectively), in molecular systems it is not that straightforward. FAN-OUT, for example, requires replication of the signalling molecule [16]. A useful device, which is regarded as universal in *quantum computing* with *cellular automata*, is the *MAJORITY gate* (Figure 1.3c). This is a multiple-input single output device. It performs the MAJORITY operation on input bits, that is yields 1 if more than 50% of inputs are in the high state, otherwise the output is zero. A three-input majority gate can be regarded as a universal gate as it can be easily transformed into OR and AND gates (*vide infra*).

1.4.2 Concatenated Logic Circuits

Single logic gates, even with multiple inputs, allow only basic logic operations on single bits of information. More complex operations, or on larger sets of bits require more complex logic systems. These systems, usually called combinatorial circuits, are the result of connecting several gates. The gates must, however, be connected in a way that eliminates all possible feedback loops, as the state of the circuit should depend only on the input data, not on the device's history. The most important circuits are the binary half-adder and half-subtractor, and the full adder (Figure 1.4a) [17]. These circuits enable arithmetic operations on bits of information in a binary fashion, which is one of the pillars on which all information technology has been built.

The *half-adder* is a device composed of two gates: AND and XOR. It has two inputs (two bits to be added) and two outputs (*sum* and *carry*). The *half-subtractor* is a related circuit (the only difference lies in one NOT gate at input) which performs the reverse

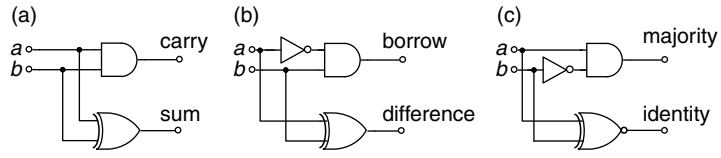


Figure 1.4 *Logic circuits of half-adder (a), half-subtractor (b) and binary comparator (c).*

operation: it subtracts the value of one bit from the other yielding one bit of *difference* and one bit of *borrow* (Figure 1.4b).

An interesting device, closely related to the half-adder and the half-subtractor is the *binary comparator*. It takes two bit inputs (x and y) and yields two bit outputs, which are determined by the relationship between the input quantities. If $x = y$ one output is set to high (*identity* bit) and the other to low (*majority* bit). If $x > y$ the identity bit is zero, while the majority bit equals 1. In the case of $x < y$ both output bits are 0 (Table 1.3, Figure 1.4c).

The appropriate connection of two binary half-adders or binary half-subtractors results in two more complex circuits, the binary adder and the binary subtractor, respectively. A *full adder* consists of two half-adders and an OR gate (Figure 1.5a). The circuit performs full addition of three bits yielding two-bit results. Similarly, a *full subtractor* is built from two half-subtractors and an OR gate (Figure 1.5b). This device can subtract three one-bit numbers yielding a two-bit binary result. The schematics of the binary full adder and full subtractor are shown in Figure 1.5 and the corresponding logic values in Table 1.4.

The simple concatenated logic circuits show the infinite possibilities of combinations of simple building blocks (logic gates) into large functional circuits.

1.4.3 Sequential Logic Circuits

A circuit comprised of connected logic gates, devoid of feedback loops (memory), is a *combinatorial logic circuit*, a device whose output signal is a unique Boolean function of input variables. A combinatorial logic circuit with added memory forms a *sequential logic*

Table 1.3 *Truth table for binary half-adder, half-subtractor and comparator.*

Input		Half-adder		Half-subtractor		Comparator
a	b	(c,s)[a]	Decimal value	(b,d)[b]	Decimal value	(i,m)[c]
0	0	(0,0)	0	(0,0)	0	(1,0)
0	1	(0,1)	1	(1,1)	−1	(0,0)
1	0	(0,1)	1	(0,1)	1	(0,1)
1	1	(1,0)	2	(0,0)	0	(1,0)

[a](carry, sum).
[b](borrow, difference).
[c](identity, majority).

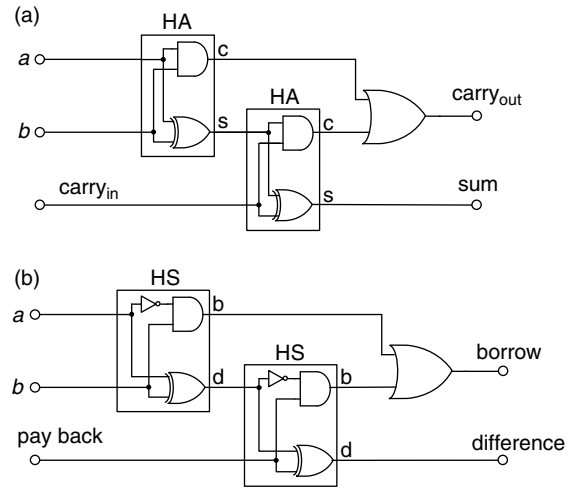


Figure 1.5 *Electronic diagrams for binary full adder (a) and full subtractor (b). HA stands for half-adder and HS for half-subtractor, respectively. In the case of subtractors, a stands for subtrahend* and b for *minuend.*

circuit, often referred to as an *automaton* (Figure 1.6). Memory function can be simply obtained by the formation of a feedback loop between the outputs and inputs of individual gates within the circuit. The output state of an automaton depends on the input variables and the inner state (memory) of the device.

The memory function can be simply realized as a feedback loop connecting one of the outputs of the logic device to one of the inputs of the same device. The simplest (but not very useful) memory cell can be made on the basis of an OR gate via feeding back the output to the input (Figure 1.7).

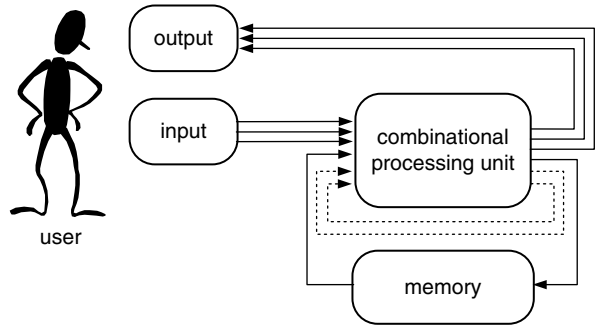


Figure 1.6 *Schematic of a sequential information processing device (an automaton). The simplest memory forf the device can be realized by a feedback loop (dashed arrow), feeding some of the outputs of the device to the input.*

Table 1.4 *Truth table for full adder and full subtractor.*

Input			Adder		Subtractor	
a	b	c/p[a]	(c,s)[b]	Decimal value	(b,d)[c]	Decimal value
0	0	0	(0,0)	0	(0,0)	0
0	0	1	(0,1)	1	(1,1)	−1
0	1	0	(0,1)	1	(1,1)	−1
0	1	1	(1,0)	2	(0,1)	−2
1	0	0	(0,1)	1	(1,0)	1
1	0	1	(1,0)	2	(0,0)	0
1	1	0	(1,0)	2	(0,0)	0
1	1	1	(1,1)	3	(1,1)	−1

[a]carry/pay back.
[b](carry, sum).
[c](borrow, difference).

Initially the device yields an output of 0. However, when the input is set to high, the output also switches to the high state. As the output is directed back to the input, the device will remember the state until power-off. Loops involving XOR and NAND gates tend to generate oscillations. These oscillations render the circuits unusable. This problem, however, can be simply solved in feedback circuits consisting in two gates (NOR, NAND, etc.) and two feedback loops (Figure 1.7b). This circuit is the simplest example of a *latch (flip-flop)*, a device, the state of which is a Boolean function of both the input data and the state of switch. This device can serve as a simple memory cell and after some modification can be used as a component of more complex circuits: shift registers, counters, and so on.

The two inputs of the latch, named R and S (after *set* and *reset*) change the state of the outputs in a complex way (Table 1.5), provided they are never equal to 1 at the same time (i.e. $R = S = 1$) as this particular combination of inputs results in oscillations of the latch.

In the case of most input combinations, the output state of the device is not changed, but the (1,0) state induces $0 \rightarrow 1$ switching, while the (0,1) state results in $1 \rightarrow 0$ switching.

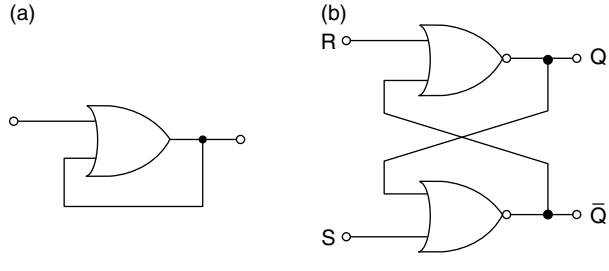


Figure 1.7 *Looped OR gate as a model of the simplest memory cell (a) and RS-type flip-flop built from two NOR gates (b).*

Table 1.5 *Truth table for the R-S latch.*

Current Q state	S	R	Next Q state
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1
1	0	1	0

1.5 Ternary and Higher Logic Calculi

Binary logic can be generalized for any system with a finite number of *orthogonal* logic states. In *ternary logic* any sentence may have three different values: FALSE, TRUE or UNKNOWN. Analagous to binary logic, numerical values can be associated with these values, as shown in Table 1.6.

Logic operations are defined in an analogous way to the case of binary logic:

The *unary ternary operator* NOT is defined as (1.30):

$$\neg a \equiv 1 - a \tag{1.30}$$

while the *binary ternary operator*s are defined as follows:

$$a \wedge b \equiv \inf(a, b) \tag{1.31}$$

$$a \vee b \equiv \sup(a, b) \tag{1.32}$$

$$a \otimes b \equiv \sup[\inf(a, 1 - b), \inf(1 - a, b)] \tag{1.33}$$

The logic values for unary and binary ternary operators are shown in Tables 1.7 and 1.8.

In any multivalued logic the unary and binary operators can be defined as follows:

$$\neg a \equiv T_0 - a \tag{1.34}$$

$$a \wedge b \equiv \inf(a, b) \tag{1.35}$$

Table 1.6 *Numerical representations of ternary logic values in unbalanced and balanced system.*

Logic value	Numerical representation	
	Unbalanced	Balanced
FALSE	0	−1
UNKNOWN	$^1/_2$, #	0
TRUE	1	1

Table 1.7 *Truth table for the unary ternary NOT.*

A	NOT A
logic values	
FALSE	TRUE
UNKNOWN	UNKNOWN
TRUE	FALSE
numerical representation	
0	1
$^1/_2$	$^1/_2$
1	0

$$a \vee b \equiv \sup(a, b) \tag{1.36}$$

$$a \otimes b \equiv \sup[\inf(a, T_0 - b), \inf T(0 - a, b)], \tag{1.37}$$

where T_0 represents the numerical value associated with the TRUE value. These definitions hold for any ordered unbalanced numerical representation of a multinary logic system.

Table 1.8 *Truth table for the binary ternary OR, AND and XOR operators.*

A	B	A OR B	A AND B	A XOR B
		Logic values		
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	UNKNOWN	UNKNOWN	FALSE	UNKNOWN
FALSE	TRUE	TRUE	FALSE	TRUE
UNKNOWN	FALSE	UNKNOWN	FALSE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	TRUE	TRUE	UNKNOWN	UNKNOWN
TRUE	FALSE	TRUE	FALSE	TRUE
TRUE	UNKNOWN	TRUE	UNKNOWN	UNKNOWN
TRUE	TRUE	TRUE	TRUE	FALSE
		Numerical representation		
0	0	0	0	0
0	$^1/_2$	$^1/_2$	0	$^1/_2$
0	1	1	0	1
$^1/_2$	0	$^1/_2$	0	$^1/_2$
$^1/_2$	$^1/_2$	$^1/_2$	$^1/_2$	$^1/_2$
$^1/_2$	1	1	$^1/_2$	$^1/_2$
1	0	1	0	1
1	$^1/_2$	1	$^1/_2$	$^1/_2$
1	1	1	1	0

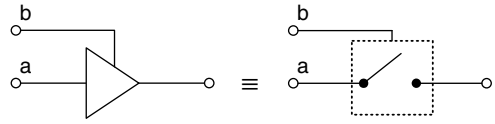


Figure 1.8 *The electronic symbol for a three-state buffer and its functional equivalent.*

Table 1.9 *Truth table for the three-state buffer.*

Inputs		Output
a	b	
0	0	HiZ
1	0	HiZ
0	1	0
1	1	1

The main advantage of ternary logic consists in lower demand for memory and computing power, however the electronic implementation of ternary logic gates is not as straightforward as in the case of binary logic gates. In the second half of the twentieth century Russian Setun (Сетунь) and Setun-70 (Сетунь-70) computers, based on ternary logic, were developed at the Moscow State University [18].

Along with ternary logic, so called *three-valued logic* has been developed. Three-valued electronic logic combines two-state Boolean logic with a third state, where the output of the gate is disconnected from the circuit. This state, usually called *HiZ or Z* (as this is a high impedance state) is used to prevent shortcuts in electronic circuits. The most common device is a *three-state buffer* (Figure 1.8, Table 1.9).

1.6 Irreversible vs Reversible Logic

The energetic equivalent of information (*vide supra*) is dissipated to the environment when information is destroyed. This is one of the fundamental limits of information processing technologies (see Chapter 3). In order to avoid this limit, computation should be

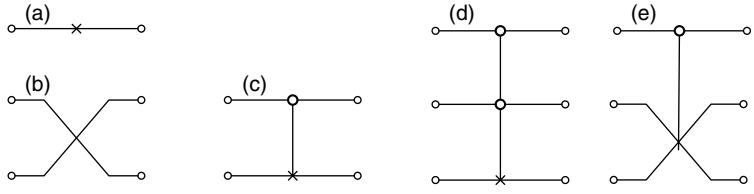


Figure 1.9 *Selected reversible logic gates presented in Feynman quantum gate notation [19,20]: NOT (a), SWAP (b), C-NOT(c) CC-NOT, that is Toffoli gate (d) and Fredkin gate (e).*

performed in such a way that no bits of information are destroyed. This approach is usually called *reversible computing*, but another term, *non-destructive computing*, is also in use. It concerns all the computational techniques that are reversible in the time domain, so the input and the output data are interchangeable. First of all, this approach implies that the number of inputs of the device equal the number of outputs. Other words, the output of the reversible logic device must contain original information supplied to the input. In this sense amongst classical Boolean logic gates only YES and NOT can be regarded as reversible (cf. Table 1.1).

All of these gates can be described in terms of permutation of states, therefore they can be easily described by unitary matrices (Pauli matrices) [19,21]. The construction of such matrices is shown in Figure 1.10.

The unitary matrices represent mapping of input states into output states, as reversible logic functions can be regarded as *bijective mapping* of *n*-dimensional space of data into itself. This ensures that no information is lost during processing, as the mapping is unique and reversible.

NOT and SWAP gates have already been discussed in the preceding section. More complex is the *C-NOT* (controlled NOT) gate, also known at the *Feynman gate*. This two-input two-output device (1.38) transfers one of the input bits directly to the output, but the second bit is replaced with its complement if the first input is in the high state. This output is thus a XOR function of the inputs.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.38}$$

The C-NOT gate, however, is not universal, as any combination of C-NOT gates cannot perform all the basic Boolean operations (cf. Table 1.2). Introduction of one more control line to the C-NOT gate results in *CC-NOT* (controlled-controlled NOT, Figure 1.9d). This gate, called also the *Toffoli gate*, is *universal*, as it can perform all simple Boolean functions. Its unitary matrix is shown as Equation (1.39).

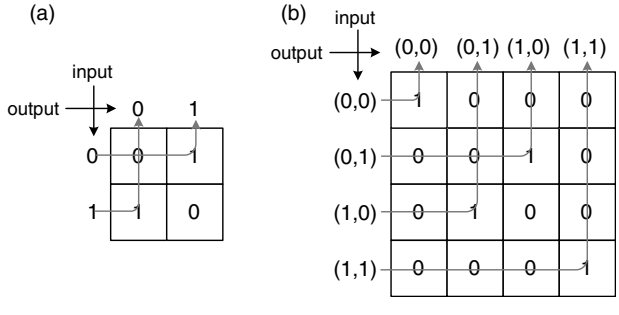


Figure 1.10 *Construction of unitary permutation matrices representing reversible logic operations in the case of NOT (a) and SWAP (b) logic gates.*

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.39}$$

A device which combines the ideas of SWAP and C-NOT is a so called *Fredkin gate* (Equation 1.40, Figure 1.9d). It can be shown that this gate is universal as well.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.40}$$

The universality of Toffoli and Fredkin gates is, however, not a unique feature. Let us look at three-input-three-output binary devices. Altogether there are $2^{24} = 16\,777\,216$ different truth tables for 3×3 logic devices. Reversible logic gates must, however, map directly each input state to a different output state (cf. Figure 1.10). This makes $8! = 40\,320$ different reversible logic gates in a 3×3 device. Only 269 of them are not fundamental, that is their combinations cannot generate all Boolean functions.

1.7 Quantum Logic

The elemental unit of information in quatum systems is the qubit (*quantum bit*). Contrary to the bit, its value is not confined to one of the two allowed states of '0' and '1', but the state of any qubit may be any linear combination of $|0\rangle$ and $|1\rangle$ eigenstates (1.41):

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \tag{1.41}$$

where c_0 and c_1 are complex coefficients normalized to unity. Even though a qubit has discrete orthogonal eigenstates of $|0\rangle$ and $|1\rangle$, it can be regarded as an analogue variable in the sense that it has a continuous range of available superpositions (1.41). This state $|\psi\rangle$ of a qubit collapses to $|0\rangle$ or $|1\rangle$ if the information is read from the qubit (i.e. when any measurement of the qubit is performed). In other words, upon measurement a qubit loses its quantum character and reduces to a bit [22]. The graphical representation of a qubit as a point on a *Bloch sphere* is shown in Figure 1.11.

Quantum systems containing more than one qubit exist in a number of orthogonal states corresponding to the products of eigenstates. A two-qubit system, for example,

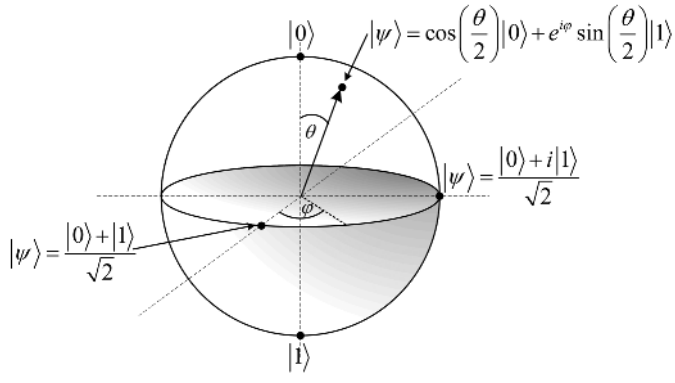


Figure 1.11 *Bloch sphere representing a single qubit. The orthogonal $|0\rangle$ and $|1\rangle$ eigenstates are located at the opposite poles of the sphere, while any superpositions are located on the surface of the sphere. (Adapted from Ref. [24].)*

may have four eigenstates: $|00\rangle$,$|01\rangle$,$|10\rangle$ and $|11\rangle$. Unlike classical systems, interference between individual qubits will result in quantum states of the form (1.42):

$$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle \tag{1.42}$$

Furthermore, two interfering qubits can exist in an entangled state, when the result of measurement on one qubit determines the result of the measurement on the other, like in the Einstein–Podolski–Rosen state (1.43):

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \tag{1.43}$$

All classical logic gates can also be implemented in quantum systems. There are, however, logic operations that can be only performed on qubits [23]. The operation of these functions can be understood with the help of the Bloch representation of a qubit (Figure 1.11) and can be regarded as various symmetry operations on the qubit vector.

The first and the simplest gate of this type is $\sqrt{NOT}$. Its unitary matrix has the form (1.44):

$$\begin{bmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix} \tag{1.44}$$

This operation, which has no classical equivalent, has the following property (1.45):

$$\left(\sqrt{NOT}\left(\sqrt{NOT}a\right)\right) = NOTa \tag{1.45}$$

that is, it flips the vector in the Bloch space by $\frac{\pi}{2}$ (cf. Figure 1.11), while the NOT gate results in π radian flipping along the x axis. There are two other possible rotations, along y and z axes, respectively, with the corresponding matrices (1.46):

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1.46}$$

respectively. Another important gate is the *Hadamard gate* with a unitary matrix of the form (1.47):

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.47}$$

The operation of the gate is simple: from $|0\rangle$ or $|1\rangle$ states the gate yields the superposition of $|0\rangle$ and $|1\rangle$ with equal propabilities, that is (1.48)–(1.49):

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{1.48}$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{1.49}$$

A universal quantum logic gate that corresponds to the Toffoli gate is the *Deutsch quantum gate* (1.50) [25]:

$$|a, b, c\rangle \rightarrow \begin{cases} \mathrm{i} \cos\theta |a, b, c\rangle + \sin\theta |a, b, 1-c\rangle, & \text{if } a = b = 1 \\ |a, b, c\rangle, & \text{otherwise} \end{cases} \tag{1.50}$$

Reversible (or non-destructive) information processing with application of the above and other devices is not restricted by the *Shannon–Landauer–von Neuman energetic limit* (cf. Equation 1.5 and Section 3.1) and therefore the thermodynamic limits of binary information processing do not apply to quantum computation [26].

References

(1) Einstein, A. (1905) Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Ann. Phys.*, **323**, 639–641.
(2) Shannon, C.E. (1948) A mathematical theory of communication. *Bell. Syst. Tech. J.*, **27**, 379–423.
(3) Hartley, R.V.L. (1928) Transmission of information. *Bell. Syst. Tech. J.*, **7**, 535–563.
(4) Landauer, R. (1961) Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, **5**, 183–191.
(5) Shizume, K. (1995) Heat generation by information erasure. *Phys. Rev. E.*, **52**, 3495–3499.
(6) Piechocinska, B. (2000) Information erasure. *Phys. Rev. A.*, **61**, 062314.
(7) Cavin, R.K.III, Zhirnov, V.V., Herr, D.J.C. *et al.* (2006) Research directions and challenges in nanoelectronics. *J. Nanopart. Res.*, **8**, 841–858.

(8) Allahverdyan, A.E. and Nieuwenhuizen, T.M. (2001) Breakdown of the Landauer bound for information erasure in the quantum regime. *Phys. Rev. E.*, **64**, 056117.
(9) Landar', A.I. and Ablamskii, V.A. (2000) Energy equivalent of information. *Cyber Syst. Anal.*, **36**, 791–792.
(10) Waser, R. (2003) Properties of information, in *Nanoelectronics and Information Technology* (ed. R. Waser), Wiley-VCH Verlag GmbH, Weinheim, pp. 11–23.
(11) Szaciłowski, K. (2008) Digital information processing in molecular systems. *Chem. Rev.*, **108**, 3481–3548.
(12) Zhegalkin, I.I. (1927) O tekhnike vychisleniya predlozhenii v simvolicheskoi logike (in Russian, On the technique of calculating propositions in symbolic logic). *Mat. Sbor.*, **43**, 9–28.
(13) Stone, M.H. (1936) The theory of representation for Boolean algebras. *Trans. Amer. Math Soc.*, **40**, 37–111.
(14) Hillis, W.D. (1999) *The Pattern on the Stone. The Simple Ideas that Make Computers Work*, Perseus Publishing, Boulder.
(15) Waser, R. (2003) Logic gates, in *Nanoelectronics and Information Technology* (ed. R. Waser), Wiley-VCH Verlag GmbH, Weinheim, pp. 321–358.
(16) Zaune, K.P. (2005) Molecular information technology. *Crit. Rev. Solid State Mat. Sci.*, **30**, 33–69.
(17) Gibilisco, S. (ed.) (2001) *The Illustrated Dictionary of Electronics*, McGraw-Hill, New York.
(18) Brousentsov, N.P., Maslov, S.P., Ramil Alvarez, J. and Zhogolev, E.A. (2010) Development of ternary computers at Moscow State University. [cited 2010 2010-03-26]; Available from: http://www.computer-museum.ru/english/setun.htm.
(19) Feynman, R. (1996) *Feynman Lectures on Computation*, Addison-Wesley, Reading, Mass.
(20) Barenco, A., Bennett, C.H., Cleve, R. *et al.* (1995) Elementary gates for quantum computation. *Phys. Rev. A.*, **52**, 3457–3467.
(21) Kitaev, A.Y. (1997) Quantum computations: algorithms and error correction. *Russ. Math. Surv.*, **52**, 1191–1249.
(22) Vedral, V. and Plenio, M.B. (1998) Basics of quantum computation. *Progr. Quant. Electron.*, **22**, 1–39.
(23) Galindo, A. and Martín-Delgado, M.A. (2002) Information and computation: Classical and quantum aspects. *Rev. Mod. Phys.*, **74**, 347–423.
(24) Vandersypen, L.M.K. and Chuang, I.L. (2004) NMR techniques for quantum control and computation. *Rev. Mod. Phys.*, **76**, 1037–1069.
(25) Deutsch, D. (1985) Quantum computational networks. *Proc. R. Soc. Lond. A.*, **425**, 73–90.
(26) Zhirnov, V.V., Cavin, R.K. III, Hutchby, J.A. and Bourianoff, G.I. (2003) Limits to binary logic switch scaling. A gedanken model. *Proc IEEE.*, **91**, 1934–1939.