Vincenzo Manca

EMERGENCE, COMPLEXITY AND COMPUTATION

ECC

Infobiotics

Information in Biotic Systems

Springer

Vincenzo Manca

Infobiotics

Information in Biotic Systems

Springer

Author
Prof. Vincenzo Manca
Dipartimento di Informatica
Universita' degli Studi di Verona
Verona
Italy

ISSN 2194-7287 e-ISSN 2194-7295
ISBN 978-3-642-36222-4 e-ISBN 978-3-642-36223-1
DOI 10.1007/978-3-642-36223-1
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012956231

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To little Elena

Preface

This book presents topics in discrete biomathematics, by promoting and exploring new approaches at the borders between biology and discrete mathematics. Its main claim is that, starting from the general principles of molecular biology discovered in the last century, an informational perspective could give hints and solutions to some of the challenges posed by biological problems. This approach is not new in biology. A remarkable example of deductive reasoning, in biological investigation, is the booklet written by Schrödinger in 1945 entitled "What is life?". In a few pages, by means of a lucid chain of general arguments, Schrödinger deduced that the information responsible for heredity has to be encoded inside cells by a kind of molecular structure that he defined "aperiodic crystals" (in essence, a surprising anticipation of what was discovered a few years later). The scientists who discovered the DNA structure in 1953 were impressed by Schrödinger's analysis, and surely it was a strong motivation in the achievement of their results. The current situation of science encourages a revival of such an approach. In fact, in the last century the cruciality of information in living processes became more and more clear. Mathematics has been widely used in modeling biological phenomena. However, the molecular and discrete nature of basic life processes suggests that many aspects underlying the logic of these processes follow principles which are intrinsically based on informational mechanisms.

Computer science is essential for the elaboration of huge amounts of biological data. For example, it was essential in genome sequencing. However, science is not only data processing. If we want to disclose the deep logic of basic mechanisms of life, we need new scientific theories, and therefore new conceptual frameworks. Discrete mathematics, algorithms, and computational approaches are good candidates for introducing new scientific ideas in life sciences. For this reason the discipline evoked by the title of this text, *Infobiotics*, is viewed as the reverse side of *Bioinformatics*. The two roots "info" and "bio" are inverted in these words. In bioinformatics the biologists ask computer scientists to assist them in elaborating the data they obtain. Conversely, in infobiotics computer scientists and mathematicians provide biologists with explanations and theories which biologists need to verify by means of specific experiments. As I like to say, *life is too important to be investigated only* *by biologists*. Computers are essential for processing data coming from biological laboratories, but many crucial questions about life can be appropriately answered by a substantial intervention of mathematicians, computer scientists, and physicists, in order to complement the work of chemists, biochemists, and biologists, The role of computer scientists cannot be limited to the use of computers, in the same manner as physicists are essential to astronomy not only in connection with telescopes.

An example of "mental experiment", similar to Schrödinger's approach, is Galileo's argument (reported by Alfred Whitehead in his *Introduction to Mathematics*) against the Aristotelian principle that heavier bodies fall faster than lighter ones. Galileo's reasoning is a masterpiece of logical analysis, which was the beginning of mechanics and of the scientific revolution initiated in the 17th century. Galileo showed that Aristotle was wrong by means of the following argument. Let H and L be two bodies, such that H is heavier than L. Let us assume that Aristotle is right, and let us link H with L, by means of very thin string, thus obtaining a new body H+L, which we drop to the ground from a higher position. If L is slower to fall, then it provides a force which slows down the fall of H. This means that H+L falls slower than H. But, this contradicts Aristotle's claim because, of course, H+L is heavier than H. This contradiction confutes Aristotle's claim. Galileo proved experimentally the validity of his argument (with the famous experiment from the leaning tower in Pisa), however the motivation of that physical experiment had its conceptual support in the previous mental experiment.

In many aspects Darwin's arguments for his theory of evolution, based on natural selection, have an analogous conceptual rigor in taking into account the evidence of crucial factors related to species (the fitness principle of living organisms, the geological transformations, the inheritance of biological characters, and the Malthusian population growth).

When we read texts of biology describing cell structures and functions, we are immediately overwhelmed by the huge amount of things and facts. But let us reverse the situation, by assuming that we are asked to create life starting from some basic kinds of organic molecules. How can we manage in order to provide some basic functions? How can we keep them together by ensuring some transformations? How can we maintain them, by ensuring some input/output channels, and by integrating them in a stable manner? Any answer to these general questions inevitably leads to the notions of metabolism and replication, which, in any possible biological realization, are logically necessary for basic phenomena on which life relies. A basic feature of cell function is the genes/proteins duality. Does this duality have an intrinsic necessity, implied by some more basic principles? Solving such kind of problems is not an abstract logico-mathematical exercise. Appropriate answers to these questions could provide important clues to facts which are relevant from a biological and medical viewpoint. These answers cannot be obtained from millions of data, or better, they may require a lot of biological and computational experiments, but surely they need to be driven by mental experiments resembling Galilean analysis of falling bodies.

Paraphrasing Spinoza's famous masterpiece ("Ethica more geometrico demonstrata"), infobiotics could be regarded as a a kind of biology "more geometrico demonstrata" and should be strictly related to *artificial life*, *computational synthetic biology*, and *natural computing*. However, these other disciplines follow different lines of investigation. Artificial life, initiated by John von Neumann and Robert Langton, concerns the realization of artificial computational systems exhibiting typical properties of living organisms (von Neumann's cellular automata were designed as auto-replicating devices). Computational synthetic biology is more specifically involved in the simulation, *in silico*, of life phenomena, while natural computing investigates the computational capabilities of natural systems. Of course, infobiotics can take advantage of all these disciplines, but its main aim is the identification of the informational mechanisms driving life phenomena. We know that life relies on information, but so far we know very little about the specific aspects of biological information.

Starting from the basic organic molecules, the first attempts at life appeared only when some special biopolymers emerged, which were able to encode information. Thus, if life is based on information, discrete mathematics and informatics must be crucial in the analysis of **biotic systems** as forms of life that we know, but also of forms that we do not know, or that are so far unrealized.

Infobiotics is aimed at considering life, in a wide sense, with the "glasses" of information. This book tries to show that discrete structures, algorithms, languages, grammars, and automata are strictly related to life structures and processes, to genes and proteins, enzymes and ribosomes, species and biological dynamics.

Verona, May 2012 Vincenzo Manca

Acknowledgements

I would like to express my gratitude to many collaborators, who contributed to develop the research lines presented in this book, and who read preliminary versions of this book by giving important suggestions toward improving its coherence and quality.

Marco Domenico Martino, and Francesco Bernardini were my students in Pisa, where my biological investigation started in 1996. Some papers that I published with them were the seeds of the development of a research line in algorithmic modeling of biological phenomena. Giuditta Franco, who was PhD student in Verona, has been a precious research collaborator of mine along the last ten years, in particular, she has worked with me on many problems of DNA computing, and developed with me the XPCR algorithm presented in the second chapter. Luca Bianco, when he was Phd student in Verona, participated in the first steps in the definition of metabolic P systems and developed the Java implementation of the "metabolic P algorithm", the first method for computing the evolution of metabolic systems in terms of multiset rewriting grammars (P systems). Luca Marchetti, who was PhD student in Verona, worked with me on solving the dynamical inverse problem in metabolic P system (MP systems, related to Păun's P systems). He developed the Matlab implementation of LGSS, the main algorithm presented in the third chapter. Alberto Castellini and Roberto Pagliarini, who were PhD students in Verona, collaborated, in many aspects, on software projects for the simulation, design, and analysis of MP systems. Recently, Giuditta Franco and Alberto Castellini are collaborating with me in a research project, outlined at end of the second Chapter, called *Infogenomics*, which is based on the notion of genomic dictionary. In this context, interesting ideas, in course of investigation, were suggested by Rosario Lombardo, and Vincenzo Bonnici, PhD students in Verona, who are developing software tools focused on non-conventional representation and visualization of genomes. Zsuzsanna Liptak, who recently joined our research group, and Iain Halliday suggested improvements for the exposition of some chapters of the book.

Ennio De Giorgi and Alfonso Caracciolo di Forino were my mentors in Pisa. They gave me a scientific imprinting and were "models" in the development of my research.

Gheorghe Păun, my friend and collaborator, was the scholar who introduced P systems and founded the scientific community of Membrane Computing. His papers and books on DNA Computing and Membrane Computing were influential for my research. Membrane Computing, DNA Computing, and Natural Computing meetings, which I regularly attended since their beginnings, were a constant source of inspiration. Discussions and ideas, during the scientific meetings of these communities, were “rivers” flowing in the underground of my mind which, after mixing their waters, very often sprang up during my conscious research activity.

Giuseppe Scollo and Angelo Spena, my colleagues in Verona, from the mathematical and and biological points of view respectively, stimulated my theoretical analyses of life phenomena from an algorithmic perspective. Ivan Zelinka, at VSB Technical University in Ostrava, supported enthusiastically the project of this book. Many other colleagues at the University of Verona helped me very much in projects and activity promoting bioinformatics curricula in Verona. This activity surely had a “catalytic” effect on many research topics related to this book.

The content of this book is based on my research and teaching mostly developed in the last ten years. In this period, I taught and coordinated many courses at the Department of Computer Science of the University of Verona (*Information theory*, *Unconventional models of computation*, *Informational methods*, *Models of natural computing*, *Discrete biological models*, *Algorithms and languages of bioinformatics*) for the curricula in Computer Science and Bioinformatics (bachelor, master and PhD courses). I am grateful to many students, for their questions, projects, and theses, which were a source of continuous clarification of my research and teaching. I taught regularly at the universities of Pisa, Udine, and Verona, where I met a huge number of students. I remember clearly a lot of them, and I know that they were essential in the construction of my scientific profile.

The seven chapters of the book are associated with the five Platonic solids, the sphere, and the Archimedean truncated icosahedron (the soccer ball), which, very often, I contemplate as symbols of rational harmony.

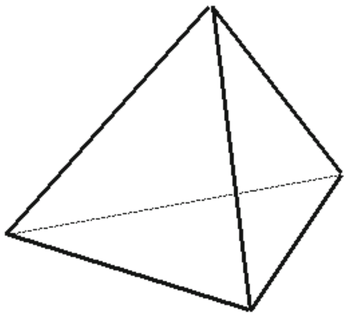
Contents

Acronyms

ACO	Ant Colony Optimization
ADA	Avogadro Dalton Action
ANN	Artificial Neural Network
BT	Binary rooted Tree
BZ	Belousov-Zhabotinsky reaction
CF	Context-Free languages
CS	Context-Sensitive languages
DNA	DeoxyriboNucleic Acid
DIP	Dynamical Inverse Problem
EDVAC	Electronic Discrete Variable Automatic Computer
EMA	Equation Metabolic Algorithm
EMBL	European Molecular Biology Laboratory
EPD	Eukaryotic Promoter Database
FIN	FINite languages
FLT	Formal Language Theory
FOL	First Order Logic
FSA	Finite State Automaton
HTML	Hyper Text Markup Language
IFF	IF and only iF
KEGG	Kyoto Encyclopedia of Genes and Genomes
k-mer	DNA string of length *k*
LGSS	Log Gain Stoichiometric Stepwise regression
LSE	Least Square Evaluation
LUCA	Last Common Universal Ancestor
LV	Lotka-Volterra dynamics
MATLAB	MATrix LABoratory, developed by MathWorks
MetaPlab	Metabolic P systems virtual laboratory
MF	Minimal Forbidden length
MP	Metabolic P (Multiset Processing, Minus Plus), prefix related to MP grammars (MP system, MP rule, ...)
MR	Maximal Repeat length

MPF	Metabolic P prefix related to flux regulation maps (regulators)
MPR	Metabolic P prefix related to reaction maps
MSE	Mean Squared Error
NCBI	National Center for Biotechnology Information
NP	languages decidable in Polynomial time by a Non-deterministic TM
NPQ	Non-Photochemical Quenching
P	Languages decidable in Polynomial time by a deterministic TM
PCR	Polymerase Chain Reaction
PSim	Java Simulator of MP dynamics
P systems	Păun systems
REG	REGular languages
REC	RECursive, or decidable, languages
RE	Recursively Enumerable, or semidecidable, languages
RNA	RiboNucleic Acid
SAT	Boolean SATisfiability problem
SIR	Susceptible-Infective- Removed epidemic
SSE	Sum of Squared Errors
SSR	Sum of Squared Regression deviations
SST	Sum of Squared Total deviations
TSS	Transcription Start Site
TTL	Test Tube Language
TM	Turing Machine
XML	eXtensible Markup Language
XPCR	Cross pairing Polymerase Chain Reaction
UB	Unrooted three-Branched tree
UCSC	University of California Santa Cruz website
UTR	Untranslated Terminal Region
VIF	Variance Inflation Factor

Part I
Topics in Discrete Biomathematics


Platonic Tetrahedron

1 Discrete Information and Life

Abstract. Information and life are intrinsically related and many challenges for a deep understanding of life phenomena require analyzing the role of information, and its specific mechanisms, in biological systems. In this chapter, the main kinds of discrete informational structures will be informally presented in many biological contexts where they occur: sets, sequences, multisets, trees, and graphs, which are mathematically defined and analyzed in the second part of the book, are here seen as fundamental mechanisms of aggregation and organization of biological components.

1.1 Symbols and Molecules

A univocal and clear definition of information is almost impossible. Information, as energy, matter, time, space, and life are such pervasive notions that any precise characterization of them necessarily misses aspects which are crucial for any comprehensive understanding of their nature. For this reason, it is useful to start with an informal idea to which we can anchor our initial intuition. We encounter information in data, which are physical objects, and can be transformed, stored, and transmitted. But they are meaningful when seen as containers of some kind of "information". This sounds clearly tautological, therefore, we need to add something to avoid vacuity. Data, recognized by some "agents", have the capability of directing and regulating their actions. In other words, they direct actions. When a flag movement or a trumpet sound is executed, then soldiers attack the enemy. The more complex are the actions, the more the information system regulating them has to be

V. Manca: *Infobiotics*, ECC 3, pp. 3–26.
DOI: 10.1007/978-3-642-36223-1_1 © Springer-Verlag Berlin Heidelberg 2013

complex. No system can exhibit a complex behavior, without a reliable system of data processing. This is the reason why living organisms need sophisticated mechanisms of data manipulation.

Information occurs with data transformations, while energy occurs with states transformations. Information needs energy for transforming, sending, receiving, and storing data, but simultaneously, it may control actions. In those cases where it does not provide a direct control of actions, it can be always located in a chain of effects which terminates with a control, or a potential control. For example, some keys in a computer keyboard send signals with direct effects to some device (for example, a printer), while pushing other keys provides only the memorization of characters somewhere (by forgetting their side effects on the screen). Typically, a document does not have a direct effect of data transformation, but it is aimed at provoking or evoking actions. Moreover, very often, the effects of some data are not easily definable, because their information can act at different levels. For example, the production of a book could have effects at the level of the book market, but also effects in the diffusion of ideas and consequent behaviors.

The term "discrete" in mathematics refers to any aggregation of objects where components are clearly distinguishable. A finite set of objects is the simplest example of a discrete structure. A segment, considered as a subset of points on a line is an example of "continuous" structure, because the geometrical representation of lines assumes that in the middle between two points there are other points, and moreover, that infinite intermediate points can be found with an endless process of resolution refinement. Space and time are usually modeled as continuous magnitudes. This is probably an idealization, with respect to the physical reality, but it provides powerful mathematical concepts for their analysis. The surprising results of classical physics (mechanics and electromagnetism) are based on continuous mathematical structures (real numbers, limit process, differential calculus). The 19th century concluded (Planck's famous quantum theory is dated November 1900) with the discovery of the discrete nature of matter and energy (the origin of Dalton's atomic theory is dated September 1803).

Information is discrete when data are based on discrete mathematical structures (the fundamental ones are presented in the second part of this book). **Digital information** is the discrete information, which refers to data realized by sequences of *digits* (elements of a finite set).

Data are represented by means of symbols, which are physical entities. Their physicality is an intrinsic aspect of their nature; even the abstract symbols of mathematics introduced for algebraic manipulation need a physical realization, that is, a quantity of matter/energy in order to be produced and perceived. A finite set of *symbols* (digits, characters, letters, signs) is called an **alphabet**. The English alphabet has 26 letters. Most alphabetic writing systems have around 20-30 characters (Archaic Latin has 21 letters). The chemistry alphabet of Mendeleev's table is given by the symbols of atomic species (around 100 elements): $\{H, He, Br, \ldots, C, O, N, \ldots\}$. The alphabet of amino acids is of 20 symbols (21 including one stop symbol). The alphabet of the usual Indo-Arabic decimal notation has ten digit symbols.

The DNA nucleotide alphabet consists of four symbols $\{A,T,C,G\}$, while the RNA nucleotide alphabet is $\{A,U,C,G\}$. The binary alphabet has two symbols. A **bit**, usually denoted by an element of $\{0,1\}$, is an element of a binary alphabet. Computer keyboards have around 80 keys. Striking them (one, two, or sometimes three of them) a character is generated belonging to an alphabet of 128 standard elements (ASCII characters) plus another 128 characters with specific usages. Any of these 256 characters can be represented by a sequence of eight binary symbols, called a **byte**. Table 1.1 collects some important examples of alphabets.

Table 1.1 Important alphabets

Binary Alphabet	$\{0,1\}$
Number Decimal Alphabet	$\{0,1,\dots,9\}$
DNA Alphabet	$\{A,T,C,G\}$
Archaic Latin Alphabet	21 letters
Amino acids Alphabet	21 letters (including stop symbol)
Chemistry Alphabet	Around 100 symbols
Minimal Computer Keyboard Alphabet	Around 80 characters (text + control)

A definition of life is even more difficult than that a definition of information.

Life, in a full sense, includes at least seven essential features: i) **birth**, ii) **nutrition** (feeding resources from the environment, expelling degraded substances), iii) **growth**, iv) **interaction** (with stimuli from/to the environment), v) **reproduction**, vi) **death**, and vii) **evolution**.

The first five features refer to the individual living organisms, seen as particular instances of life, in time and space. The last two features refer to populations of living organisms instantiating forms of life, which by means of their cycles of birth-reproduction-death realize a second level dynamics, providing, along the arrow of time, a *tree of life*, that is, a "genealogy" of life *species* (at least one individual must reproduce for the survival of the population and all individuals eventually die).

Life exhibits a tremendous complexity of interacting processes which, at different levels, have to communicate for reaching successful realizations of specific functionalities. The complexity of this underlying dynamics needs information.

Life emerged only when an efficient system of data processing was possible at molecular level.

In fact, the fundamental discovery of the past century was the molecular structures and processes which life is based on. Molecules are discrete structures built on atoms.

In the middle of the 19th century, the discovery of DNA structure [63], and its informational and digital nature, introduced a new level in the analysis of life phenomena [1]. From this perspective, discrete information becomes the basis of crucial molecular processes inside the cell, and it follows that their logics can be completely understood if their symbolic role is recognized. In fact, DNA molecules represent symbols, and the discrete nature of these molecules implies the digital nature of processes involving them. Biochemistry provides nanotechnological support to symbolic elaborations, which follow principles analogous to those of formal systems investigated in formal language theory or mathematical logic.

1.1.1 Molecules and Protocells

According to some cosmological evaluations, at very beginning, about 10^{10} years ago, and in a time interval between around 10^{-44} and 10^{-35} seconds, at a temperature near 10^{28} Celsius degrees, matter appeared as subatomic particles. After a very short fraction of one second, they aggregated into the simplest atoms, from Hydrogen to Helium, Lithium, Beryllium, Boron, Carbon, Nitrogen, Oxygen, Fluorine, and Neon (the list is only an exemplification reporting the first period of Mendeleev's table) [3]. Then, the simplest organic molecules of hydrocarbons (carbon aggregated with hydrogens) appeared. In fact, the carbon atom has a structure which implies an enormous combinatorial power of aggregations (we do not enter into more details, but this aspect is related to its specific "tetravalent" nature). From hydrocarbons the organic molecular evolution starts, which is the basis for the emergence of life.

The more complex organic molecules are constituted by carbon, hydrogen, oxygen, nitrogen, and other crucial atoms (for example, phosphorus, sulfur, iron, calcium, potassium, sodium, chlorine). They are capable of providing a very rich catalog of chemical aggregations (hydrocarbons, alcohols, ethers, esters, amides, amines, fatty acids, and other organic groups) [4].

The prebiotic molecular evolution is the process at the end of which, from these basic organic molecules, more complex molecules emerged, which provided the basic pieces for the chemical realization of **polymers** and **membranes** [10, 6, 199].

Protocells are biomolecular aggregations which:

i) separate an inner region from the external environment, by means of membranes, *ii)* select within this internal region some kinds of molecules, *iii)* concentrate the molecules which are inside, for a better chemical interaction, *iv)* protect the internal space from external disturbances to the internal activity, and v) take matter from outside and expel matter outside for feeding the internal processes and eliminating matter which degrades or is dangerous to the internal activity.

These five points, referring to the compartmentalization of molecules, are realized by membranes. However, point ii) is not enough for an efficient realization of chemical transformation. In fact, vicinity in space is important, but some agents are necessary which recognize reactants and speed up their reactions, that is, which play a role of **catalysts**. This role can be played by complex molecules, able to discriminate forms. Linear polymers provide a solution to this complex task. By using some "ports" controlling the input/output flux of molecules, protocells maintain a stable flux of in-coming and out-going matter, in order to ensure that internal chemical species are kept within certain specific quantitative ranges. However, in this kind of protocell, two essential requirements of life are missing: growth and reproduction. When protocells exhibit some primitive form of these features, then they are close to cells.

The basic problem of protocell realization is: keeping the introduction-transformation-expulsion ability, and keeping the membrane structure separating the external environment from the interior space. It is important to realize that even these easy actions are not as simple as they may appear at a first glance. In fact, they are performed without any intervention of external agents. This is not so trivial if only molecules are available for performing all these tasks. In fact, some kinds of molecules have to be arranged in order to construct membranes, some have to control selectively the input/output (ensuring that only certain molecules can enter and only certain others can exit), and some molecules have to direct the chemical transformations. The problem becomes even more challenging if we consider that any kind of matter aggregation, the more complex it is, the more easily it is subjected to degradation, therefore, after some time (depending on the kind of aggregation), it is naturally destined to be destroyed, or to lose its functionality.

1.2 Sequences and Polymers

Sets or **classes** (in this context we use these terms as synonyms) are collections of distinct objects. A set is completely specified by the elements which belong to it, and if it has a finite number of elements, then it is completely described by a list of them. In this list, the appearance order of the elements is not relevant, and moreover, each element has to appear once.

Consider the set of decimal digits $\{0,1,2,3,4,5,6,7,8,9\}$. If we arrange them in the following way:

$$314159265358979$$

we have the first 15 digits (decimal dot is omitted) of the decimal representation of π (the ratio between any circle and its diameter). Here, two aspects are different with respect to sets: some symbols occur many times and the order of symbol occurrences is relevant. Structures with a linear arrangement of components, possibly occurring many times, where the order and the number of occurrences is relevant are called **sequences**. They are expressed in many ways, for example, by putting the occurrences between parentheses and by separating them by commas:

$$(3,1,4,1,5,9,2,6,5,3,5,8,9,7,9)$$

however, what is essential is a clear indication of the order and the number of occurrences. The total number of the elements occurring in a sequence, counting the repetitions, constitutes its **length**. In general, sequences are referred to a preliminary fixed set. For example, the sequence above is a sequence over the set of decimal digits.

It is easy to realize that the number of sequences grows exponentially with respect to their length. For example, the number of sequences of length 100 over 20 elements is 20^{100}, that is, a number overcoming any possibility of enumeration, within the whole existence of the universe. Therefore, the aggregation of basic kinds of molecules, in linear arrangements, opens an enormous possibility in the search of complex molecules. In fact, it provides a space of molecular variability which life can explore in the search of the complex molecular functions for assembling living organisms, starting from molecules.

Polymers are sequences over a set of component molecules called **monomers**. They have a crucial relevance in fundamental mechanisms of life, and are the molecular basis of replication, which is essential to life reproduction.

1.2.1 LUCA and the Sequence Paradox

What we described in Sect. 1.1.1 is more precisely a **prebiotic protocell**. It is only able to maintain, for some time, an introduction-transformation-expulsion ability. However, another feature is missing for the passage from a prebiotic protocell to a cell in a full sense: the reproduction in two similar organisms which, in turn, are able to perform the same task. Surely, many attempts of reproduction happened, but over time, the reproductive ability of descendants degraded in two possible ways: i) they lost the reproductive ability due to some internal degradation, or ii) the environment changed and they were not able to survive. This means that, among a huge number of attempts, possibly in different places, and in different times, life emerged, in a full sense, when a protocell was able to produce descendants for a sufficient number of generations. One lineage acquired the property of generating stable progenies. Therefore:

Cells originating from that ancestor protocell, called **LUCA (Least Universal Common Ancestor)** are the cells of the life we know. This event is dated around 3.8 billions of years ago.

A different possibility for the origin of life would suggest a different scenario where the process outlined above happened many times, providing different sources of life. In this case, it is possible that different LUCAs cooperated, in some way, by exchanging some molecules, by keeping a sort of similarity, and surely a kind of competition was established, in sharing common resources [7]. On the contrary, if no cooperation/competition had been established among biotic protocells, then almost certainly, the lineages originating from them would have followed very different paths. This situation seems nowadays confuted by the fact that all the known forms of life exhibit an evident common molecular basis.

A famous paradox related to polymers, attributable to Eigen [128, 137], is called the *sequence paradox* and is related to the impossibility of producing replicating polymers with a length greater than the inverse percentage of replication error. That implies the impossibility of improving the replication efficiency beyond a fixed threshold (around 100 units according to Eigen's theory). In fact, the ability of a faithful auto-replication improves with the complexity of the polymers, and consequently, with an increment of their length (longer programs are more powerful than short programs). This means that replicating polymers need to be sufficiently long in order to replicate without errors, and to keep this special property. But, at the same time, the longer sequences are, the more errors are cumulated during their replication (increasing the length of a text, the errors in copying it increase too).

Life had to solve the sequence paradox, and other paradoxes, in order to emerge from the prebiotic chemistry, and in order to evolve toward more complex organisms.

However, even if science discloses some important aspects of these paradoxes, probably no rational reconstruction can completely disclose all the reality of life's mystery.

1.3 Multisets and Membranes

The notion of **multiset** is something between sets and sequences. In fact in a multiset an element can occur more then once (zero times means that it does not occur), but the order of occurrences is not relevant. A standard way for denoting this multiset is the **multiset polynomial notation** such as:

$$2a+3b+c.$$

The non-negative number of occurrences of an element a in a multiset X is also called the **multiplicity** of a in X and we denote it by:

$$X(a).$$

We write:

$$a \in X$$

if $X(a) > 0$, that is, when a occurs in X. Of course, a set is a multiset where all its elements occur with multiplicity 1. A multiset Y is **included** in a multiset X if $Y(a) \leq X(a)$, for every $a \in X$. In this case, we write also $Y \subseteq X$.

Sequences and multisets over a set A, called also the **support** of these structures, can be seen as particular functions. In fact, a sequence can be seen as a function from a subset of natural numbers (position from 1 to the sequence length) to elements of a set, while a multiset can be seen as a function from a set to a subset of natural numbers (the multiplicities assigned to the elements of the set).

A sequence α of length n over A is a function: $\alpha : \{1,2,\ldots n\} \rightarrow A$

A multiset X over A, where any element occurs at most n times, is a function: $X : A \rightarrow \{0,1,2,\ldots n\}$.

A very useful notation which assumes many meanings, in dependence on the context where it occurs, is the absolute value sign $|\ |$. When it is applied to a (finite) set, X, then $|X|$ means the number of elements of X, also called **cardinality** (in set theory it extends also to infinite sets, providing *transfinite cardinal numbers*). For a sequence α, the expression $|\alpha|$ denotes the **length** of α, while for a (finite) multiset X, notation $|X|$ denotes its **size**, that is, the sum of the multiplicities of all elements occurring in X.

Given an order among the elements of a finite set A of cardinality n, any finite multiset X over A, is completely identified by the numeric sequence, of length n, of the multiplicities that the elements of A have in X (according to the order fixed over A). In this sense, finite multisets coincide with finite sequences of numbers.

The main principles of aggregation and selection mechanisms, on which life is based, rely on the basic discrete mathematical structures of sequences and multisets. We claim that many choices of life for realizing its strategies of expansion and development are intrinsic to the informational logic of these fundamental structures.

1.4 Chemistry Multisets

Any molecule is a multiset of atoms providing a stable physical structure. In molecules, multiplicities are indicated by indexes, thus CO_2 has the same meaning of $C+2O$. Chemical substances are multisets of molecules. In fact, 100 molecules of CO_2 correspond to a multiset of molecules (100 copies of the same molecule). A quantity of 12 grams of Carbon dioxide CO_2 is a multiset of about 6.2×10^{23} of CO_2 molecules.

Chemistry suggests some natural operation on multisets. Firstly, given two multisets X,Y over a set A, their **multiset sum** $X+Y$ can be defined by setting that, for any $a \in A$, the multiplicity of a in $X+Y$ is given by:

$$(X+Y)(a) = X(a) + Y(a)$$

where, for the sake of simplicity, the same symbol $+$, which aggregates elements in multiset polynomial notation, here denotes the multiset sum operation.

Multiset sum is, of course, commutative and associative, that is for any multisets X, Y, Z:

$$X + Y = Y + X$$
$$(X + Y) + Z = X + (Y + Z).$$

The multiset without elements is denoted by $\emptyset$ (which also denotes the empty set).

Multiset multiplication $m \cdot X$ of a multiset X, over a set A, by a natural number m is defined by the following equation:

$$(m \cdot X)(a) = m \cdot X(a)$$

where symbol $\cdot$ on the right hand denotes the product between numbers. If we identify an element a as a multiset of a single object (with multiplicity 1), then $m \cdot a = ma$, that is, for single multisets, multiplication can be identified with multiplicity.

Multiset operations naturally occur in chemistry. In fact, a **chemical reaction** is an operation transforming a multiset of molecules (reactants) into another multiset of molecules (products).

A chemical reaction is usually denoted by an ordered pair of two multisets, with an arrow between them:

$$\text{Reactants} \quad \rightarrow \quad \text{Products}$$

A reaction such as:

$$X + Y \rightarrow 2Z$$

can be viewed as an operation which takes the molecules, viewed as multisets of atoms, X and Y and transforms them into the multiset $2Z$. The elements on the left of the arrow are also called substrates of the reaction.

Let us consider a reaction of n reactants $X_1, X_2, \ldots X_n$ and m products $Y_1, Y_2, \ldots Y_m$:

The so called **stoichiometric balance** of such a reaction is the procedure which provides the minimum multiplicities (if they exist) $h_1, h_2, \ldots h_n, k_1, k_2, \ldots k_m$ of reactants and products such that:

$$(h_1 \cdot X_1) + (h_2 \cdot X_2) + \ldots (h_n \cdot X_n) = (k_1 \cdot Y_1) + (k_2 \cdot Y_2) + \ldots (k_m \cdot Y_m)$$

The **law of multiple proportions** is one of the fundamental laws of stoichiometry and was first discovered by the English chemist John Dalton in 1803. The law states that when chemical elements combine, they do so in a ratio of whole numbers:

In any chemical reaction, the coefficients of the reactants and products are always multiples of some positive integer coefficients, which are called **stoichiometric coefficients**.

This principle is a consequence of the fact that molecules are multisets of atoms and the atoms in each molecule occur with an integer positive multiplicity, therefore, for the conservation principle, the multisets of atoms corresponding to molecules have to occur with positive integer multiplicities. The Italian scientist Avogadro introduced a number as a standard value for indicating a population of molecules. Its value corresponds to the number of hydrogen molecules contained in one gram of this substance, its value was estimated as (the two digits between parentheses are the standard deviation of the last two digits):

$$6.02214179(30) \times 10^{23}.$$

A concrete example of stoichiometric coefficients, for balancing a chemical reaction, is given by one of the most important reactions for life. The sugar glucose $C_6(H_2O)_6$ and oxygen O_2 are formed from water H_2O and carbon dioxide CO_2, by means of chlorophyll synthesis. In this case, the stoichiometric coefficients which provide the chemical balance are the following (the light energy is converted into the chemical energy of glucose, which, burning in presence of oxygen, reverts the process by producing energy):

$$6CO_2 + 6H_2O \rightarrow C_6(H_2O)_6 + 6O_2.$$

1.5 Liposome Membranes

If we need to put together a number of molecules of some types, we need to collect them in a space containing them. A realization of this compartmentalization is biologically provided by **liposomes**. They are membranes realized in a very simple and efficient manner, by using a special kind of organic molecules, called *p*hospholipids, which, put in the water, are subjected to two opposite forces. In fact, they are asymmetric, with one head and one tail. The head part is hydrophilic, while the tail part is hydrophobic. In the water, they can solve the hydrophilicity versus hydrophobicity contradiction by aggregating each one, side by side, in such a way that all heads remain externally in contact with water, while the tail of any phospholipid is opposite to the tail of another phospholipid (see Fig. 1.1). In this way, a "bilayer" structure is realized, where tails remain dry, while heads are wet.

The emergence of life requires that a number of reactions may work, and persist for some time. Reactions need membranes where reactants are collected as multisets of molecules, spatially concentrated and protected from external noises. In conclusion, the following statement summarizes the biological cruciality of membranes.

Membranes built by phospholipids need water, and stable reactions need membranes, therefore life, as we know it, needs water.

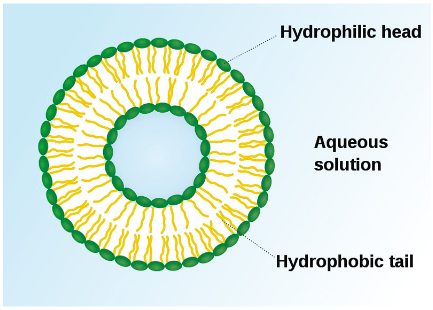


Fig. 1.1 The schema of a liposome (GNU Free Documentation License)

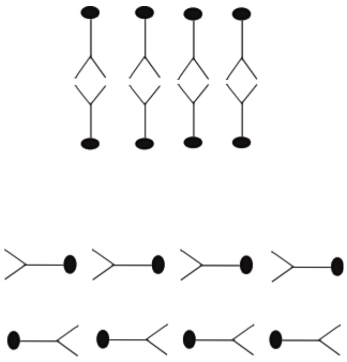


Fig. 1.2 Aggregation according to phospholipid bonds (top) and to phosphodiester bonds (bottom)

Figure 1.2 shows the two fundamental kinds of biological aggregations. The one at the top is typical of biological membranes. The one at bottom refers to DNA molecules. Both aggregations are based on double links. But, in phospholipids we have a *b*ilayer schema where firstly bilayer double monomers of type: (head-tail)+(tail-head) are realized, and then these double monomers become elements of aggregations along all directions (almost) parallel with the axis passing through their heads. In this way, two surfaces, having heads externally, are in contact with

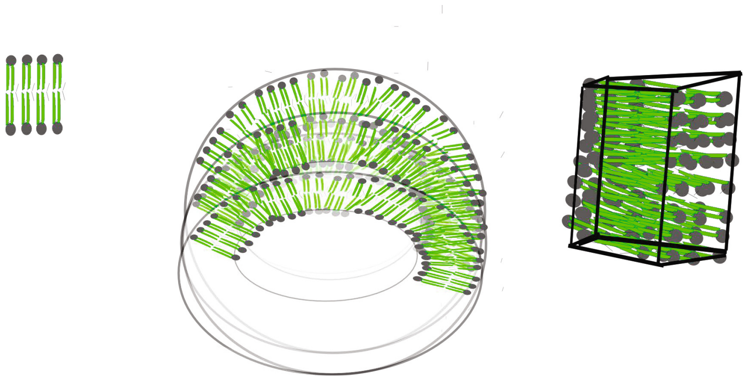


Fig. 1.3 Stylized images of phospholipid aggregation of bilayer monomers around their head-head axis

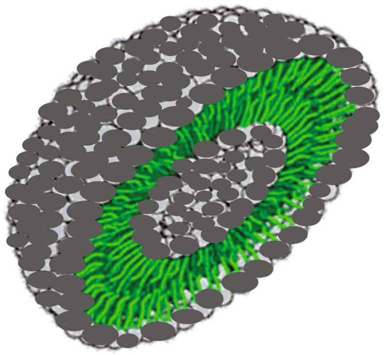


Fig. 1.4 A semi-complete liposome aggregation with a tail-uncovered border

water, while double tails are waterproofed by the double tails around them in the bilayer aggregation. The two surfaces showing the heads of double monomers provide one internal and one external spherical surface (sphere is the minimum surface enclosing a given volume). The second arrangement develops according to a bilinear schema. In this case, firstly a linear *c*atenative arrangement is realized by sequences of type: (head-tail)+(head-tail)...+(head-tail), then two lines are paired (at least in regular situations). In this case, pairing can be established only between two corresponding monomers of the paired lines, by means of a specific molecular bond between them (having complementary types, see Fig. 1.13). A crucial aspect

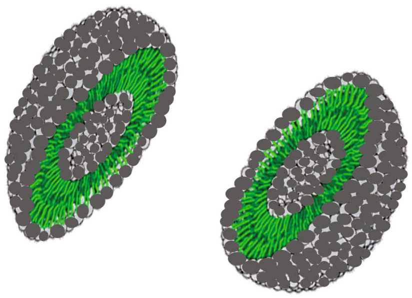


Fig. 1.5 Two semi-complete liposome aggregations that provide a membrane by joining their tail-uncovered borders

explaining the difference between the spherical and bilinear shapes is given by the symmetric nature of double monomers with respect to their head-head axis, in opposition with the asymmetric nature of the head-tail monomers which are catenated in the two lines. DNA monomers, as will be explained in the next chapter, are asymmetric with respect to the head-tail axis, and along this direction can be established a pairing only with another monomer. Therefore, the double DNA arrangement is a "bilinear concatenation", in opposition to the liposome arrangement which is a "bilayer (closed) convolution". As will be explained in the next chapter, under very general hypotheses, bilinear catenative arrangements will provide helical shapes.

Another crucial difference between phospholipid and DNA aggregations is due to the kind of molecules that they aggregate. In the phospholipid case, there is only one type of head-tail monomers, while in the DNA case there are four different head-tail monomers. This is the reason for the enormous number of combinatorial possibilities for DNA molecules.

We conclude with the following statement.

Both aggregations of phospholipids and DNA molecules are based on asymmetric monomers. Phospholipid aggregation is based on homogeneous bilayer double monomers, which are symmetric with respect to the head-head axis. DNA aggregation is based on heterogeneous double linear concatenation of monomers, which are asymmetric with respect to the head-tail axis.

1.6 Populations and Hypermultisets

Chemistry deals with multisets at two different levels. In fact CO_2 is a multiset over the set $\{C, O\}$, while $6CO_2 + 6H_2O$ is a multiset over the set $\{CO_2, H_2O\}$. In other words:

$$6CO_2 + 6H_2O = 6(C + 2O) + 6(2H + O).$$

In general, the notion of multiset can be considered at any level, because multisets of second level can be considered as objects which can be aggregated by providing multisets of third level, and the same kind of construction can be iterated. We reserve the word **population** for denoting multisets of the first level, while the term **hypermultisets** is used for multisets at a level greater than one. Therefore, a **population** over a set A is a multiset of elements which belong to A, and a multiset of level $k > 0$, over a set A, is a multiset of elements which are multisets of levels lower than k (over A).

Populations are ubiquitous in life phenomena, at any level, from populations of molecules, to populations of organisms and species. However, the peculiarity of membranes is that they provide a biochemical counterpart of parentheses of mathematical constructs. In fact, when molecules are put inside membranes, these become elements of other aggregation levels, where they are elements (possibly occurring in many copies) of a second level multiset. We note the following statement emphasizing the structural role of membranes.

Membranes constitute the biochemical realization of hierarchical multiset aggregation of biological components.

If we consider an atom as a multiset of subatomic particles, that is, protons, neutrons, and electrons, then a molecule such as CO_2 is a multiset of two levels:

$$CO_2 = C + 2O = (6 \cdot (p+n+e)) + 2(8 \cdot (p+n+e)).$$

Analogously to hypermultisets, we can consider hypersets and hypersequences (however, this is not standard terminology). A **hyperset** is a set including sets as its elements. For example, $\{a, \{a,b\}\}$ includes as element $\{a,b\}$ which is the set of elements a and b. A **hypersequence** is a sequence of sequences. For example, $(a, (a,b), (a, (a,b)))$ is a hypersequence. Any sequence can be represented by a hypersequence constituted by pairs (sequences of length 2). For example (a,b,c) can be represented by $((a,b),c)$. The notion of level for hypersets and hypersequences can be defined as in the case of hypermultisets. It is easy to realize that hyperstructures are expressed, up to here, using parentheses. For example, a hypersequence of level two has two levels of parentheses. This concept is the basis of hierarchies, and trees are the mathematical concept behind any notion of hierarchy, as we show in the next section.

Before considering trees and graphs in basic structures of life, let us conclude regarding the intertwined roles of membranes and polymers. The basic mathematical structure underlying molecules and reactions are multisets. Efficient realizations of chemical reactions require complex molecules facilitating and driving them (enzymes), and a compartmentalization of molecules in order to select, concentrate, and protect all the elements involved in reactions. Membranes provide the biological solution to compartmentalization, and polymers, which are sequences of monomers, provide the enormous molecular variety and complexity within which molecular functionalities can be found for life finalities. Membranes and polymers are generated by suitable mechanisms of biochemical aggregation. However, only the aggregations that are involved in processes of organisms able to survive, reproduce, and evolve, are maintained by life genealogies. In conclusion, aggregations in space, based on the mathematical structures of multiset and sequence, are selected along time for life propagation and evolution. Tables 1.2 and 1.3, and Fig. 1.6 summarize and visualize basic aspects of biological aggregation.

Table 1.2 Basic mechanisms of matter organization

Aggregation	Stability in space
Selection	Stability in time

Table 1.3 Correspondence between basic mathematical and biochemical aggregations

Multisets	Membranes
Sequences	Polymers

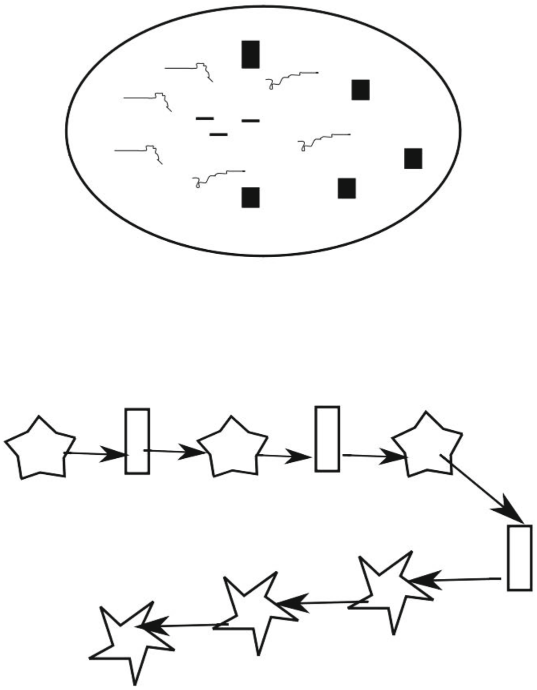


Fig. 1.6 The basic aggregation mechanisms of life: membranes and polymers

1.7 Trees and Hierarchies

A natural way to express a hypermultiset, built from some initial elements, is by means of diagrams such as those given in Fig. 1.7. Each level of aggregation corresponds to a closed plane curve embracing the elements which are aggregated, possibly occurring in many identical copies. Moreover, the 2-dimensional nature of this kind of diagram represents adequately the lack of order among the components inside each closed curve. The diagram at the bottom of Fig. 1.7 is a diagram of a **rooted tree**, a typical structure representing genealogies. The root is the node at the top of the diagram (the origin of the genealogy), and each element that is the *parent* of some *child* node is **an internal node** (the root is an internal node), while other nodes, called *leaves*, do not have child nodes, and the nodes having the same parent are *sibling* nodes (some authors use the masculine designation ”father, son, brother”, while others use the feminine designation ”mother, daughter, sister” instead of ”parent, child, sibling”). Terms *ancestor* and *descendant* refer to nodes that are the beginning and the end, respectively, of a chain of parent-child relationship (another way of expressing a tree is by means of a *parent function*, assigning to any node different from the root its parent node).

Life tree represents the hierarchical structure of classification of living organisms. The tree of Fig. 1.8 (http://en.wikipedia.org/wiki/Phylogenetic_tree) is a life evolution tree.

The trees of Fig. 1.9 show the difference between prebiotic and biotic trees. The biotic tree has leaves connected to the Last Universal Common Ancestor by an uninterrupted genealogical chain going from it down to the current living organisms.

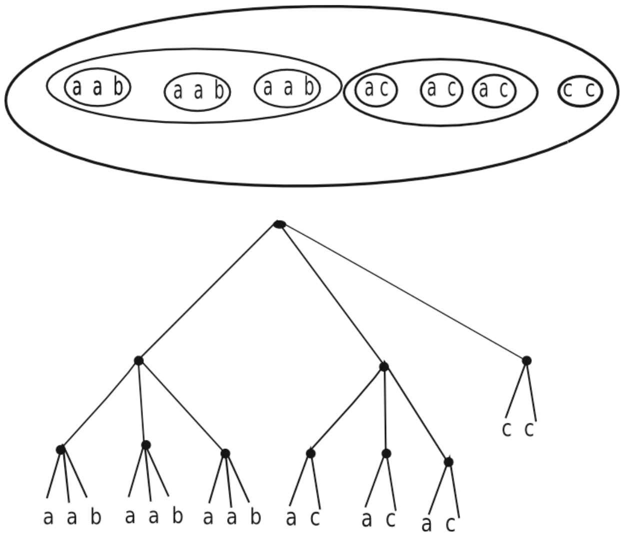


Fig. 1.7 The membrane diagram of the multiset $3[2a+b]+2c+3[a+c]$ on the top, and the tree diagram of the same multiset on the buttom

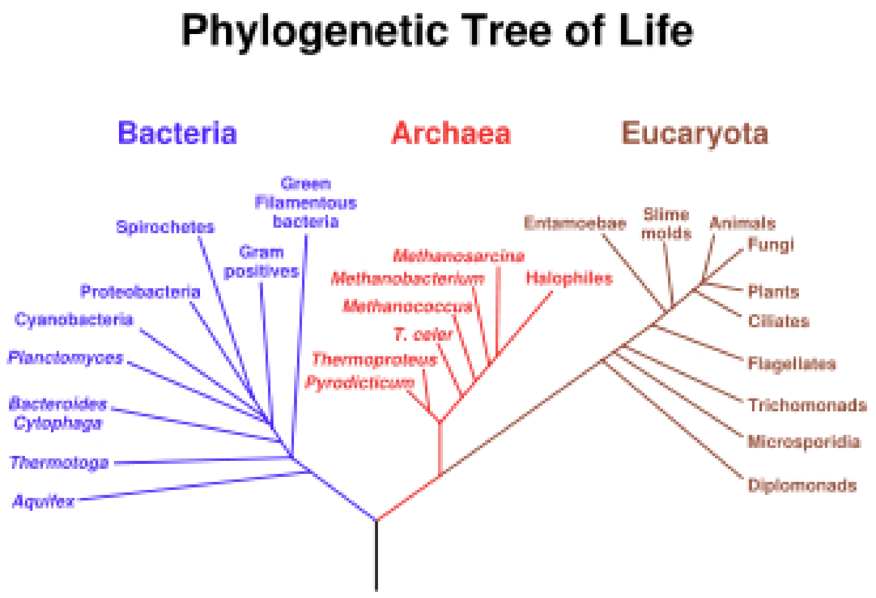


Fig. 1.8 Life as a tree, reconstructed by ribosome RNA sequencing (Credit: Nasa Astrobiology Institute, Public Domain)

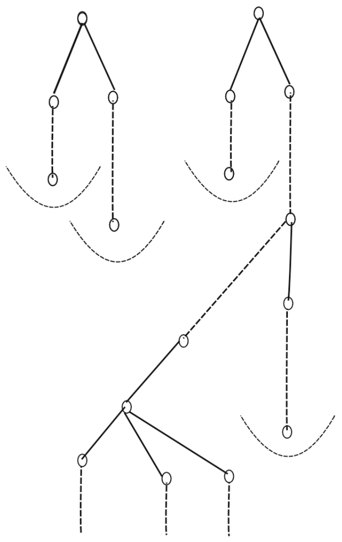


Fig. 1.9 The left tree is a pre-biotic evolution which ends. The right tree is a biotic evolution with non-ending paths.

1.8 Graphs and Interactions

A molecule is built over a multiset of atoms, but this multiset does not provide all the information about the relations holding among its atoms. In chemistry, the multiset of a molecule is also called molecular formula, while the *structural formula* is a graph, that is a set of *nodes* connected by *edges* or *arcs* (usually drawn by lines or curves, oriented or not, and possibly tagged with specific data). Two molecules having the same molecular formula, but different structural formulae are called **isomers**.

A graph is the mathematical abstraction of point-line (or point-arrow) diagrams. In any context where some entities are related, possibly in many different ways, graphs naturally emerge. Mathematically speaking, a **graph** is a set of "arrows" (edges, arcs), each of which connects two nodes (tags, or marks may be associated to nodes and/or arrows). The terminology of graphs is very intuitive and all the important features of a graph, such as *path, cycle, degree, component* can be mathematically expressed by using the basic notions of set, sequence, multiset, operation, and relation. For example, a **path** is a sequence of nodes, where consecutive nodes are connected by an edge. A graph is **connected** if any two nodes are connected by some path. A **cycle** is a path where the first node coincides with the last one. A graph is **acyclic** if it does not have cycles. The graphs which are connected and without cycles are also called **unrooted trees**.

From an unrooted tree, a rooted tree can be univocally obtained by choosing one node as root. Connection and absence of cycles ensure this possibility. In fact, any node has to be connected to some other node. Therefore, from the chosen root we get its sons and, iteratively, from them we get other nodes until we reach some leaves. No node can be reached twice, because the graph is assumed without cycles.

1.8.1 Graphs of Molecule Structures

Molecules are graphs where nodes are atoms (or atom aggregates) and edges are the chemical bonds among atoms (some tags can be added to nodes and/or edges for expressing specific properties of chemical interest, such as the types or strengths of bonds). For example, a basic components of RNA and DNA molecules are **pentoses**, sugar molecules constituted by carbons and water molecules. **Ribose** $C_5H_{10}O_5$ is a pentose which includes five carbons and five molecules of water (H_2O). The positions of the five carbons in pentoses are identified by primed numbers. **Deoxyribose** $C_5H_{10}O_4$ derives from ribose by removing an Oxygen in position $2'$.

Figure 1.10 represents the graph of the structural formula of a **nucleotide**, the basic component of DNA polymers. It is constituted by three parts:

1. **deoxyribose**,
2. **nitrogenous base** (*adenine* in Fig. 1.10),
3. **phosphate group** PO_4.

In the graph of Fig. 1.10, nodes are atoms or groups of atoms, and edges represent the chemical bonds between them. Table 1.4 gives the complete list of **Nitrogen Bases**, which correspond to the letters of DNA and RNA alphabets.

In the deoxyribose molecule, Carbon at position $1'$ is connected to the nitrogenous base. Position $2'$ is where Oxygen is missing, respect to ribose, and position $5'$ is the position where Carbon shares an Oxygen with the phosphate group. Bases are connected by means of a bond between Nitrogen and Carbon (a *glycosidic* bond) which is established by liberating a water molecule.

When a nucleotide is linked to another nucleotide, then phosphate group becomes the bridge between the $5'$ Carbon with the $3'$ Carbon of the other nucleotide. In this link, OH molecule at position $3'$ of pentose (deoxyribose in DNA case, or ribose in RNA case) is replaced by an Oxygen of the phosphate group.

Table 1.4 The nitrogen Bases of DNA. Uracil substitutes Thymine in RNA.

Adenine	A	$C_5H_5N_5$
Guanine	G	$C_5H_5N_2O_2$
Thymine	T	$C_5H_6N_2O_2$
Cytosine	C	$C_4H_5N_3O$
Uracil	U	$C_4H_4N_2O_2$

It is interesting that the nucleotide corresponding to the letter A of the DNA alphabet has a crucial role from the point of view of biological energy. In fact, the Adenosine triphosphate is constituted by the adenine nucleotide, where deoxyribose is replaced by ribose and the phosphate group is a triphosphate (see Fig. 1.12). This molecule is the so-called unit of energetic currency in biomolecular transformations. When energy is required, then a reaction, from ATP to the molecule ADT (Adenosine diphosphate, with a diphosphate group) is simultaneously associated, which liberates both a phosphate group and an energy quantity (7 Kcal/mole at 37^o Celsius). Figure 1.12 shows this transformation, which is essentially an operation transforming a connected graph into another one with two connected components.

RNA structure is similar to that of DNA, with the only difference that in RNA the basic components are ribonucleotides (in a ribose the Carbon at 2' binds H_2O instead of H_2). Ribo-nucleotides concatenate each other, by providing RNA strands.

DNA nucleotide concatenate in double strands, according to the schema of Fig. 1.13, where a nucleotide of a strand is paired with a nucleotide of the other strand if the two corresponding bases are a complementary pair. The two complementary pairs are $\{A,T\}$ and $\{C,G\}$ (a pairing of a DNA strand with a RNA strand is also possible, according to the complementarity $\{A,U\}$, $\{C,G\}$). The three principles of this molecule arrangement are: *bilinearity, complementarity,* and *antiparallelism*.

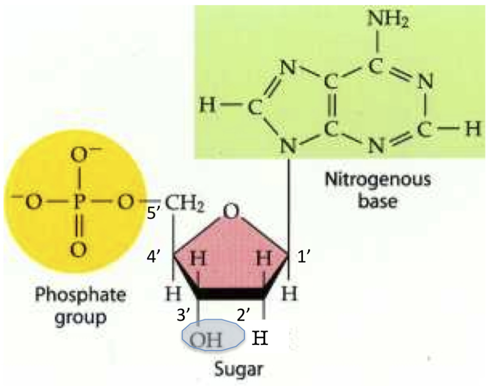


Fig. 1.10 The graph of the structural formula of Adenine nucleotide. The atoms of Carbon in positions $1', 2', 3', 4'$ are not indicated.

1.8.2 Graphs of Reactions

Molecule structures are represented by graphs, but even chemical reactions acting cooperatively and competitively over some substances are helpfully represented by reaction graphs. A chemical reaction is naturally represented by means of a **multi-edge**, which extends the notion of edge, because it has a set of source nodes and a set of target nodes (it can be represented by a *body* with possibly many tails and many arrows, as it is illustrated by Fig. 1.14). Namely, in a chemical reaction many reactants are connected to many products. **Multigraphs**, which are based on **Multi-edges**, can be also seen as graphs with two kinds of nodes (circles representing substances and full circles representing reactions), and with two types of edges (tail edges and head edges). Figure 1.14 describes chemical reactions expressed by a multigraph.

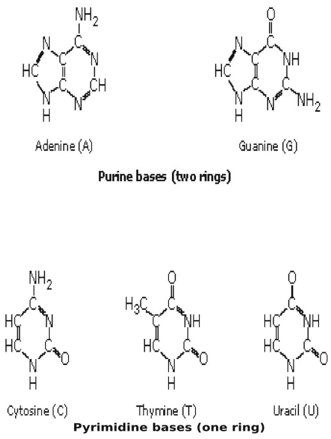


Fig. 1.11 The Nitrogenous bases

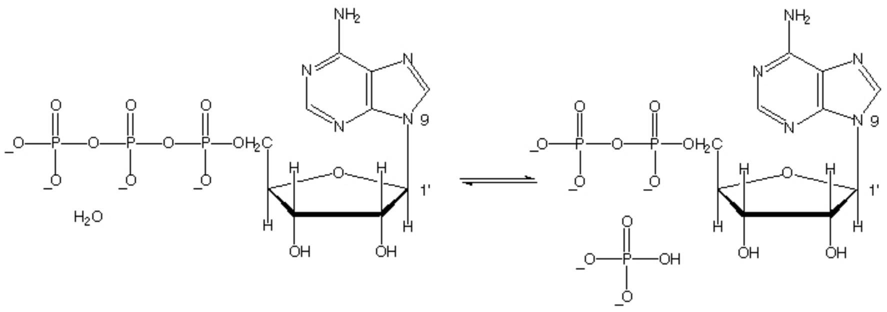


Fig. 1.12 The transformation from ATP to ADT molecule. On the right, an excess of Hydrogen ions is present, which is not indicated.

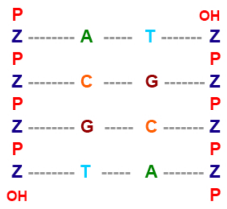


Fig. 1.13 The schema of a DNA double strand, where Z denotes the sugar, P the phosphate, and A, T. C, G the bases

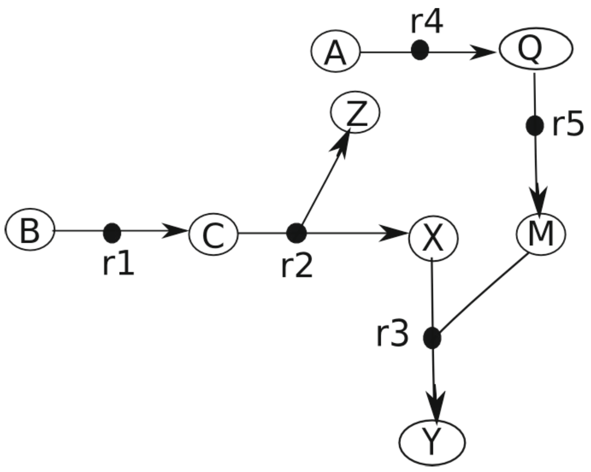


Fig. 1.14 A chemical reaction as a multigraph

1.8.3 Neuron Graph

The structure of our brain is a graph built on a networks of about 10^{11} nodes (neurons) connected by means of 10^{14} edges (synapses). Figure 1.15 reports an original drawing by Ramon y Cajal, the Spanish scientist who discovered the neuron, on the basis of a visualization technique introduced by the Italian scientist Camillo Golgi (both scientists received the Nobel prize for their discoveries), while Fig. 1.16 (http://www.sciencecases.org/split_brain/split_brain.asp) provides a schematic representation of its structure.

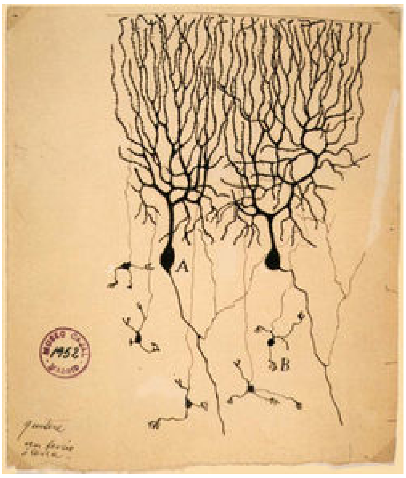


Fig. 1.15 An original drawing of a neuron by Ramon y Cajal

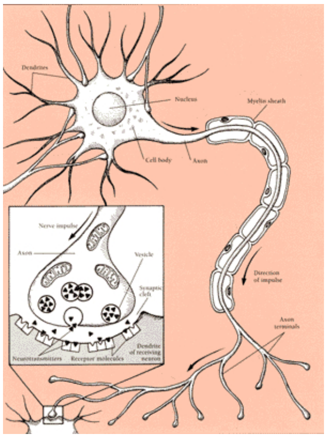


Fig. 1.16 A schematic structure of a neuron (Credit: National Institute on Drug Abuse)

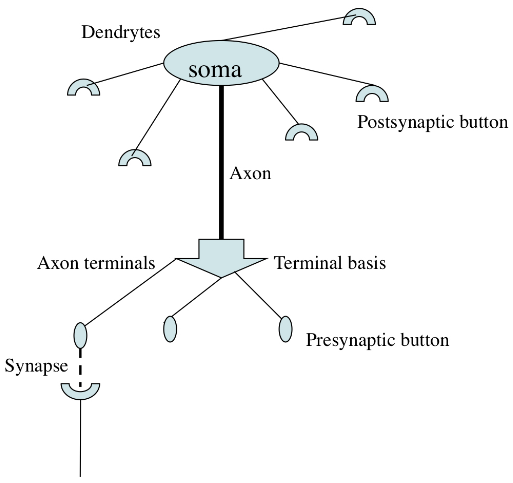


Fig. 1.17 A kind of graph corresponding to a neuron structure

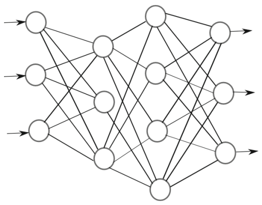
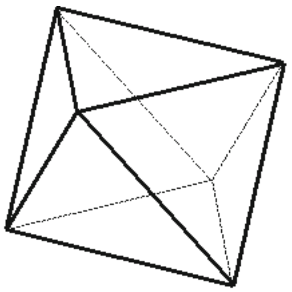


Fig. 1.18 The structure of an artificial neural network

Neurons suggested one of the first computation models elaborated by McCullouch and Pitts in 1943 [9], which was the basis of circuits of the EDVAC computer designed by John von Neumann in 1945 [207]. Artificial Neural Networks (ANN) became popular, in many variants, in typical problems of artificial intelligence (recognition, learning, classification, and robotics) [8]. Fig. 1.18 shows a graph representing a general kind of neural network. In this case, nodes are disposed in successive levels (columns of nodes, usually from left to right). Edges connect nodes of one level to nodes of the next level, and are labeled with numerical coefficients called *weights*. A node contains a *current value* (initially it may be some default value). The current values of nodes change in time according to the following strategy. A *activation function* is associated to each node. The current value of a node is *passed*, to the connected nodes of the next level, after multiplying it by the corresponding weights of the connecting edges. The weighted values passed to a node are given as arguments of the activation function of the node, and the result becomes the new current value of the node. In this way, when some input values are given as current values of the leftmost nodes, after a number of steps, depending on the number of levels of a given ANN, the current values of the rightmost nodes provide the output values of the ANN in correspondence to the given input values. In this sense, an ANN with n input nodes and m output nodes computes a function from numerical n-sequences to numerical m-sequences. A typical problem of ANNs is the following inverse problem: given a set of k pairs (X_i, Y_i), for $i = 1, 2, \ldots, k$, where X_i and Y_i are input and output sequences, finding an ANN able to provide, in the most faithful way, the given correspondence, that is, providing the best approximating function that underlies the given input-output pairs.



Platonic Octahedron

2 Strings and Genomes

Abstract. Strings constitute the mathematical structures of informational biopolymers. Genomes are long strings (hundreds of thousands, or millions, or billions of characters) built over the four nucleotides, and many typical operations over genomes are naturally expressed by string operations. In this chapter we present basic concepts about DNA molecules and genomes in algorithmic terms, by emphasizing the roles of strings, formal languages, and multisets of strings in the analysis of typical biological and biotechnological DNA manipulations. We conclude by outlining some research lines of genome analysis which are based on genomic dictionaries. The chapter is mostly based on the author's published papers (see **References for Chapter 2**).

2.1 Biological Monomers and Polymers

Words of written alphabetic languages are the most usual intuition of symbolic linear forms. However, if we were to create, at a biomolecular level, structures similar to words, then we would experience an essential difficulty. In fact, letters are arranged over an external rigid support (paper) maintaining their linear arrangement stable and robust despite the movements of the support, while molecules are floating in a liquid environment. Therefore, we need a different way to arrange them. In other words, the linearity has to be implemented by means of a feature internal to molecules. This is the reason for the following structure which is common to the most important biological monomers: *body, head, tail, flag, and bridge*. The body is the component of the monomer to which head, tail, and flag are connected. The link

DOI: 10.1007/978-3-642-36223-1_2

between two monomers is obtained by connecting the head of a monomer with the tail of another monomer in a resulting component which is the bridge of the composed structure. When the link is performed we get a polymer, or more precisely a dimer, having a head and a tail: the head is the head of the monomer which fused its tail in the bridge, while the tail is the tail of the monomer which fused its head in the bridge (see Fig. 2.1 and Fig. 2.2). In this construction we have a result the linear arrangement of the two flags of the linked monomers, and we can continue the process by linking, in the same way, the obtained polymer with another monomer or with another polymer. This means that the flag is the variable component of monomers, or what makes its specific role in the construction of polymers.

The reader is invited to verify that this structure is present in the nucleotide given in Fig. 1.10, where the body is the sugar, the head and also the bridge is the phosphate group, the tail is the oxhydryl group OH in 3', while flags are the nitrogenous bases.

For peptides we have that the body is a Carbon-Hydrogen group, the head is the amine group NH_2, the tail is the carboxyl group $COOH$, the bridge is the group $CONH$, and flags are the 20 peptide residues.

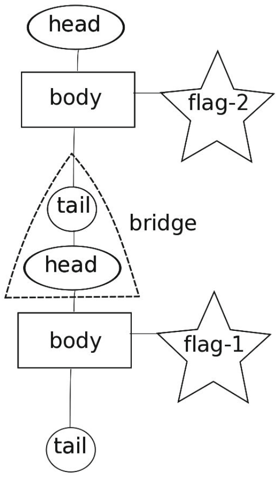


Fig. 2.1 The structure of a biological monomer

2.2 DNA Strings and DNA Helix

DNA molecules realize a special structure of strings. For this reason, DNA manipulation can be formally described in terms of string operations. However, DNA strings are more precisely **double strings** and their physical nature implies their specific geometric form of a helix. In the following section we discuss these aspects.

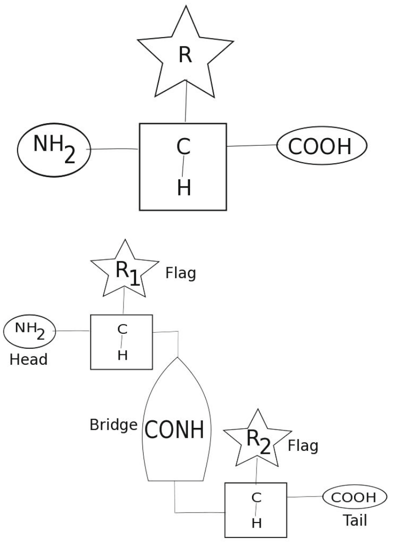


Fig. 2.2 The structure of a peptide link. The peptide structure above, the peptide bond below. The bridge links two peptides, by fusing the head NH_2 of one peptide with the tail $COOH$ of the other in $CONH$ (a water molecule H_2O is released).

2.2.1 DNA Notation and Double String Operations

Let us recall and extend some standard notation from Formal Language Theory (see [213], and Chap. 6) in order to formalize fundamental operations related to the structure of DNA molecules.

Let us consider the usual alphabet of bases $\Gamma = \{A,T,C,G\}$. The set Γ^* of *strings* over this alphabet is comprised of the sequences (words) that can be arranged with these four symbols (letters). Strings of Γ^* will be indicated by Greek letters. On these strings, a binary associative operation of *concatenation* is defined, that given two strings of Γ^* yields a new string where all the symbols of the second sequence are put, in the given order, after the last symbol of the first one. Concatenation between two strings α and β is denoted by the juxtaposition $\alpha\beta$. The length of a string α is the number of its symbol occurrences (each symbol is counted as many times as it occurs) and is indicated by $|\alpha|$. A special string is that of length 0, denoted by λ, that is, the *empty* string, where no symbol occurs (an abstract notion similar to zero for numbers). Symbols are special strings of length 1. Mathematically speaking, the structure Γ^* is referred to as the *free monoid* over the alphabet Γ. Any subset of Γ^* is a (formal) *language* over the alphabet Γ. Since languages are sets, all the usual set theoretical notions extend to languages (such as membership $\in$, inclusion $\subseteq$, empty set $\emptyset$).

The symbol of α that occurs in position i ($1 \leq i \leq |\alpha|$) is denoted by $\alpha(i)$, the sequence of symbols of α occurring (in the given order) from position i to position j

($1 \leq i \leq j \leq |\alpha|$) is denoted by $\alpha[i,j]$ and is called a *substring* of α. In particular, it is called a *prefix* if $i = 1$, and a *suffix* if $j = |\alpha|$. A string with prefix α and suffix β is also denoted by $\alpha \dots \beta$. The *complementation* function, denoted by the superscript c, is defined on Γ by $A^c = T, T^c = A, C^c = G, G^c = C$. It extends naturally to Γ^* by the conditions $\lambda^c = \lambda$ and $(\alpha\beta)^c = \alpha^c\beta^c$. Another operation on Γ^* is the *reverse* operation *rev* such that, for every $X \in \Gamma$, $rev(X) = X$, $rev(\alpha X) = X rev(\alpha)$, and $rev(\lambda) = \lambda$.

Reversing and complementation operations are *involutive* and *commute*, that is:

$$rev(rev(\alpha)) = \alpha$$

$$(\alpha^c)^c = \alpha$$

and

$$rev(\alpha^c) = (rev(\alpha))^c.$$

The *mirroring* of α is defined by:

$$mir(\alpha) = rev(\alpha^c)$$

and is an involutive operation. We abbreviate $mir(\alpha)$ as $\bar{\alpha}$.

In DNA molecules, an intrinsic concatenation verse is given which goes from the Phosphate to the Oxydryl (that respectively correspond to Carbon $5'$ and $3'$ positions in the sugar DNA backbone). This verse is symbolically denoted by an arrow at one side of non-null strings α, as in:

$$\alpha \rightarrow$$

that can be seen as an indication of the extremity where the Oxydryl terminal is located (while Phosphate is at the other extremity). However, when we omit the arrow, the usual reading from left to right is assumed, which corresponds to the $5' - 3'$ verse.

An *exact pairing* (symmetric) relation $||$ is defined over Γ^* such that:

$$\alpha || \beta$$

if $\beta = mir(\alpha)$.

Let γ be the longest string such that α and β include γ and $\bar{\gamma}$ respectively. Let h_{hybr} be a non-null length that we call *hybridization threshold* (it represents a biological parameter we leave unspecified). If $|\gamma| > h_{hybr}$, then we say that α and β *pair by hybridizing* on γ (or shortly, that they *hybridize*) and write:

$$\alpha]\gamma[\beta.$$

In this case a double DNA string composed of α and β is defined, that will be denoted by:

$$\frac{\alpha}{rev(\beta)}.$$

In this fraction notation the following concatenation verses are implicitly assumed:

$$\frac{\alpha \rightarrow}{\leftarrow rev(\beta)}$$

that is, the Oxydryl terminal of the inferior string is located on the opposite side with respect to that of the superior string. Sometimes, fraction notation is a way to make explicit the verse of single strings. In fact, we put (superior notation):

$$\frac{\alpha}{\lambda} = \alpha \rightarrow$$

and (inferior notation):

$$\frac{\lambda}{\alpha} = \leftarrow \alpha = rev(\alpha).$$

For notation completeness it is assumed that:

$$\frac{\lambda}{\lambda} = \lambda.$$

We write:

$$\alpha][\beta$$

if α and β hybridize (with an exact pairing on some internal portions) and

$$\alpha \,]\!\!\not[\, \beta$$

when they do not hybridize. Pairing is a crucial operation of double strings formation, where the phenomena of *bilinearity*, *complementarity*, and *antiparallelism* are jointly involved. It is a partial operation that is defined on two strings only when they hybridize. The set of single or double DNA strings will be denoted by Γ^*/Γ^*. By the symmetry of the pairing relation, double strings satisfy the following equation:

$$\frac{\alpha}{rev(\beta)} = \frac{\beta}{rev(\alpha)}.$$

This is an important aspect of double strings, that formalizes the mobility of DNA strands in a fluid environment (superior and inferior positions are relative concepts). On the other hand, any α hybridizes exactly with $\bar{\alpha}$ by definition of exact pairing, hence the double string:

$$\frac{\alpha}{rev(\bar{\alpha})}$$

is defined, which is called a *blunt* double string and is denoted by $<\alpha>$. We note that:

$$<\alpha>= \frac{\alpha}{rev(\bar{\alpha})} = \frac{\alpha}{rev(rev(\alpha^c))} = \frac{\alpha}{\alpha^c}$$

and, as a consequence of the equations above we obtain:

$$<\bar{\alpha}> = <\alpha>$$

in fact, we have that:

$$<\bar{\alpha}>= \frac{\bar{\alpha}}{rev(\bar{\bar{\alpha}})} = \frac{\bar{\alpha}}{rev(\alpha)} = \frac{\alpha}{rev(\bar{\alpha})} = \frac{\alpha}{rev(rev(\alpha^c))} =<\alpha> .$$

Operations on DNA strings Γ^*/Γ^* are summarized in Table 2.1.

We call *strand* any object s on which an operation *type* is defined which assigns to it a string in Γ^*/Γ^*. In particular, if $type(s)$ is a single string, then we say that s is a *single strand*, while if $type(s)$ is a double string, then we say that s is a *double strand*. Intuitively, strands are DNA molecules which, as physical objects, are different from the base sequence they realize. Therefore, in our terminology, strands having the same base sequence are strands with the same type. We restrict ourselves to consider only single or double strands. In fact, for the needs of the following discussion, we may avoid considering more complex forms of DNA molecules that combine more than two DNA strands.

We consider a DNA pool P as a set of strands, or equivalently, as a *multiset* of single or double strings of Γ^*/Γ^*, which is specified by a *multiplicity* function $mult_P$ from Γ^*/Γ^* to natural numbers. In fact, $mult_P(\eta) = n$ means that the pool P contains n (indiscernible) strands of type η. We write $P = \{n_1 : \eta_1,\ n_2 : \eta_2,\ \ldots,\ n_k : \eta_k\}$ when $mult_P(\eta_1) = n_1, mult_P(\eta_2) = n_2, \ldots, mult_P(\eta_k) = n_k$ and $mult_P(\eta) = 0$ for $\eta \notin \{\eta_1, \eta_2, \ldots, \eta_k\}$. Mixing and splitting DNA pools correspond to the standard multiset operations of sum and difference, denoted by $+, -$ (sum and difference of their multiplicity functions).

In virtue of these two ways of considering a pool, we can use (ambiguously) both notations: $\eta \in P$, for a string $\eta \in \Gamma^*/\Gamma^*$, meaning $mult_P(\eta) \neq 0$, and $s \in P$, when s is a strand of P. The type of pool P is the set of strings (a language):

$$Type(P) = \{\eta \in \Gamma^*/\Gamma^* \mid \eta \in P\} = \{type(s) \in \Gamma^*/\Gamma^* \mid s \in P\}.$$

We remark the difference between *type*, which assigns a string to a strand, and *Type* (with capital T) which assigns, to a pool of DNA strands, the set of types of its strands. Although strings and strands are different things, very often the two terms are used almost synonymously. In fact, any string is physically implemented by strands having its type, and conversely, the expression “a string”, in a given context, could refer to a physical occurrence of a string, which is just a strand.

Table 2.1 Basic DNA operations

$\alpha[i,j]$	Substring
$\alpha\beta$	Concatenation
$(\alpha)^c$	Complementation
$rev(\alpha)$	Reversing
$mir(\alpha)$	Mirroring
$\alpha][\beta$	Hybridization
$\frac{\alpha}{\beta}$	Pairing
$< \alpha >$	Blunt Pairing

2.2.2 DNA Helix

DNA molecules are constituted by nucleotides arranged in a bilinear structure. **Bilinearity** is accompanied by two other features: **complementarity** and **anti-parallelism**. Both these features are consequences of the template driven duplication of DNA, which seems to be essential to the nature of this molecule. In fact, template duplication is performed in two basic steps: removing the bonds which tie the two paired sequences and then using each of them as a template by restoring the missing paired sequence with the appropriate nucleotides matching the templates. In this manner the two templates produce two equal double strands. In this procedure, it is essential that paired strands could be unpaired. For this reason any nucleotide has to be paired by a weak chemical bond, which is better realized between different (complementary) molecules. In biological organisms this kind of duplication is performed by a class of enzymes called Polymerase. Their action is essentially called **extension**, in fact they extend a strand in the verse $5'$ to $3'$ by copying the missing (complementary) nucleotides according to the sequence specified by the template strand (by using nucleotides floating in the environment). However, their action can be performed only when an initial part of the missing strand, usually called **primer** is already present (see Fig. 2.3).

A bilinear structure which realizes a template driven duplication is described in Fig. 2.4.

From an abstract viewpoint, a monomer M of a bilinear structure is characterized by a triangle, say a **monomeric triangle**. In fact, let us consider an internal point P of this monomer, for example its barycenter, which we call its X-point. Then, the monomeric triangle is defined by the three point P, P', P'' where P' is the X-point of the monomer M' concatenated to M, and P'' is the middle point between P and the X-point of the monomer M'' paired with M (see Fig. 2.4).

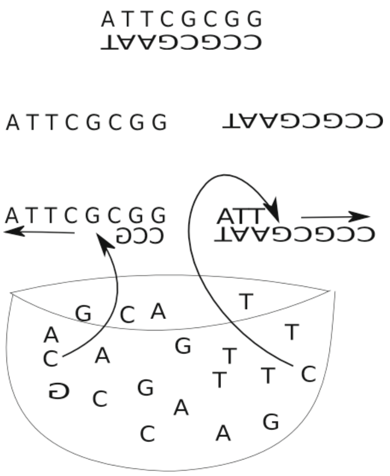


Fig. 2.3 Template driven sequential duplication

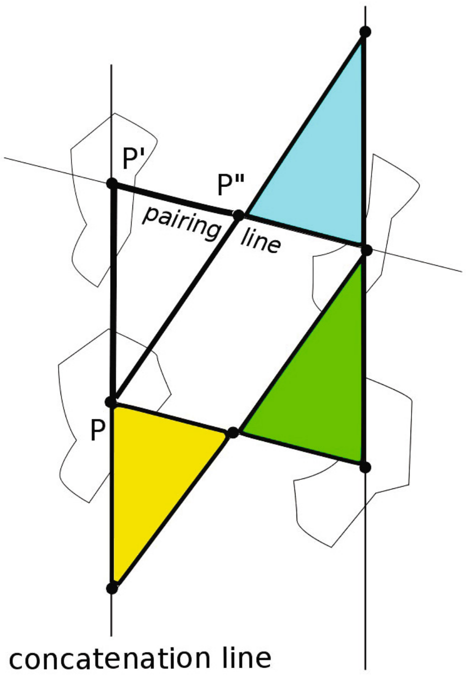


Fig. 2.4 The bilinear arrangement of monomeric triangles

A monomeric triangle defines a Cartesian system where abscissa, or x-axis, goes from P to P', ordinate or y-axis goes from P' to P'', and altitude or z-axis is orthogonal to the xy-plane and oriented according to a counterclockwise rotation of x-axis toward y-axis. The pairing has a strictly dual nature: only two monomers can be paired, because this relationship is exclusive, and when two monomers are paired, both of them are unavailable to be paired with something else. In conclusion, monomers need to be asymmetric with respect to two distinct directions, the concatenation direction and the pairing direction. In fact, both these two relations are intrinsically oriented. For this reason, monomers are **chiral** (from a Greek term for hand), that is, each monomer defines univocally a three-dimensional Cartesian coordinate system.

A monomeric triangle clearly defines a plane. Therefore, we can consider different possibilities for the planes where concatenated and paired monomeric triangles lie.

Figure 2.5 shows some possible planar bilinear arrangements of monomers in the different cases of right, acute, or obtuse monomeric triangles in parallel and antiparallel arrangements. Apart from the difficulty of keeping planar structures in a fluid environment, the space occupancy of these structures would become prohibitive for long DNA molecules. In all these cases, no rotation of paired monomeric triangles is allowed around any axis lying in the plane of concatenation. Namely, as indicated at the bottom of the figure (in the case of a right angle parallel arrangement) such a kind of rotation would be in conflict with the parallelism (or anti-parallelism) of the these concatenated structures.

If concatenated triangles, lying on the same or different planes, form an angle along the concatenation line, then we can have the possibilities illustrated in Fig. 2.6. In both possibilities the concatenation angles vary along the concatenation line, because in a spiral the curvature increases from the periphery to the center (a logarithmic spiral could avoid the angular variability, but its space occupancy would be prohibitive). Therefore, this kind of arrangement is impossible, because it would imply an angle between two concatenated monomers that depend on their positions in concatenation line.

In the cases of acute or obtuse monomeric triangles, parallel arrangements are impossible, as indicated in Fig. 2.7, because the bilinear structure cannot be realized.

When monomeric triangles are acute or obtuse, they can be arranged in antiparallel way (see Fig. 2.8), and the arrangement of both concatenation lines in the same plane can be avoided by means of a rotation along the pairing line, as indicated at the bottom of the figure. Of course, acute monomeric triangles realize more compact arrangements than obtuse monomeric triangle, therefore are preferable.

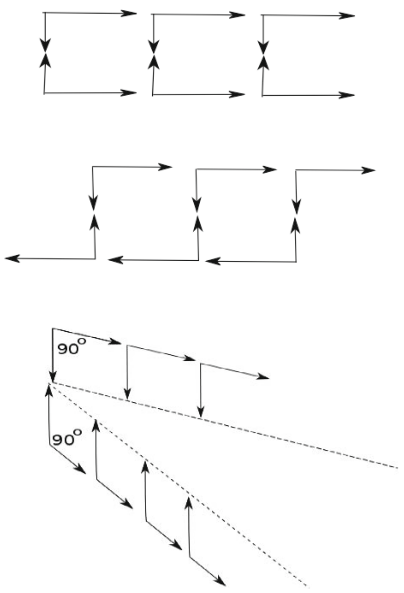


Fig. 2.5 Parallel and antiparallel arrangements of bilinear monomeric triangles

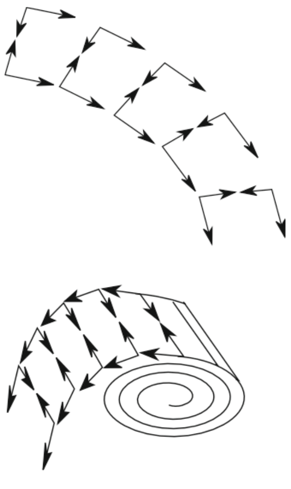


Fig. 2.6 Impossible linear arrangement of right monomeric triangles with a rotation angle between concatenated monomers: Top: rotation of concatenated triangles around z-axis; Bottom: rotation of concatenated triangles around y-axis (xyz, the Cartesian system of the monomeric triangles)

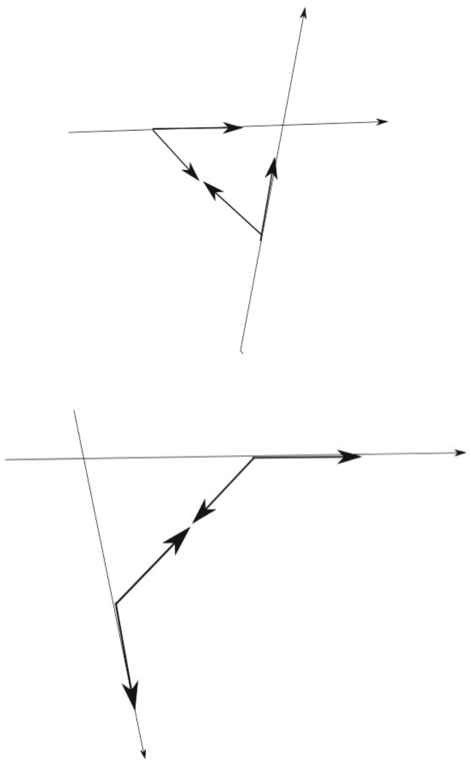


Fig. 2.7 Parallel arrangement of acute (top) or obtuse (bottom) monomeric triangles are impossible

Figures 2.9 and 2.10 are different looks of the same structure, where paired monomeric triangles form a rotation angle around the pairing line. This implies that they can be located inside a cylinder shape that is completely characterized by a radius and by three angles: the rotation angle ρ, between two concatenated triangles, the rotation angle Φ between two paired triangles, and the angle τ between the concatenation segment of the triangle with the cylinder axis. From an evaluation of these values, and of cylinder radius, the average length of edges of monomeric triangles can be estimated (fractions of a nanometer) [44].

When many modules of the kind given in Fig. 2.10 are arranged, we obtain the bilinear structure with a spiral shape, which grows internally to a cylinder, as shown in Fig. 2.11, that is, the DNA double helix structure appears in its pure geometrical form.

In Figure 2.12, we can see the empty internal cylinder formed by the paired monomeric triangles along the DNA helix.

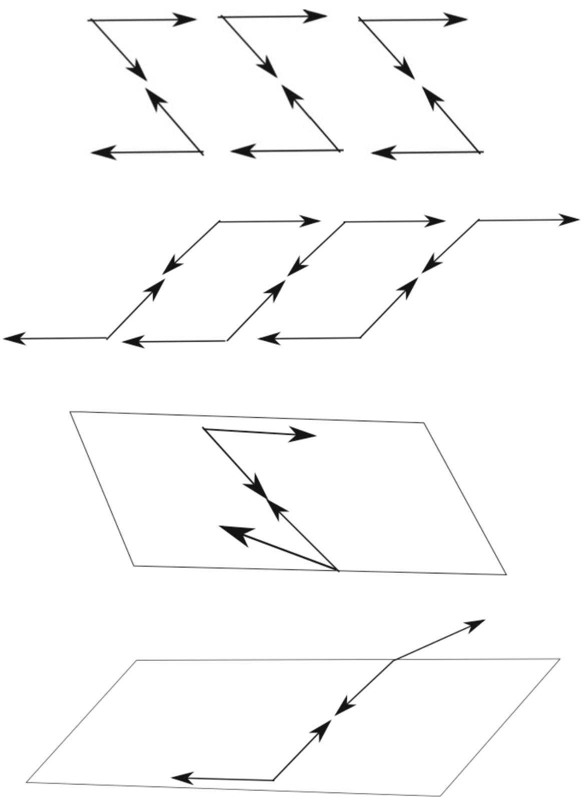


Fig. 2.8 Planar arrangement of acute or obtuse monomeric triangles is possible in an antiparallel way

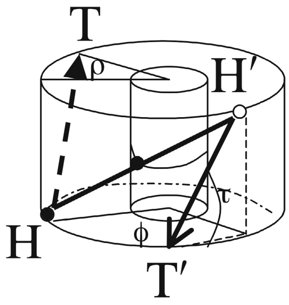


Fig. 2.9 Two paired monomers inside a cylinder

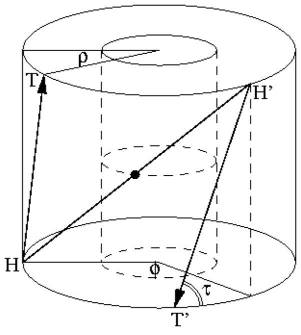


Fig. 2.10 Another look of two paired monomers inside a cylinder

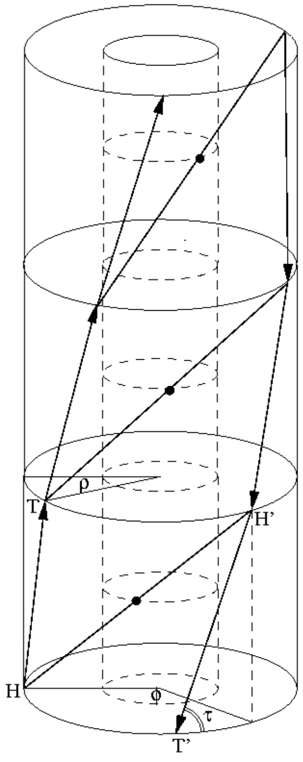


Fig. 2.11 The helix of paired monomers inside a cylinder

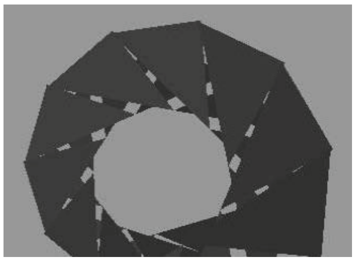


Fig. 2.12 A view of the helix from upstairs

We know that DNA helix presents two groves: a major grove and a minor grove. Figure 2.13 presents a stylized view of this structure, as a consequence of the double rotation of the two paired lines around the same cylinder axis. This is the ideal geometric structure underlying the fundamental discovery by Watson and Crick in 1953 [63].

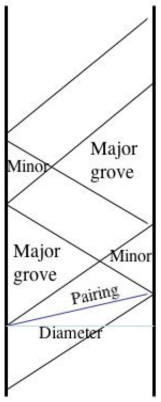


Fig. 2.13 A stylized representation of DNA groves

In conclusion, anti-parallelism is a direct consequence of the bilinear nonplanar arrangement of nucleotides, as deduced by the analysis developed in terms of monomeric triangles. Moreover, the chirality of monomers joined to their anti-parallelism implies that if they have the same chirality, then a reading agent can read both the two concatenation lines along the same reading plane. Therefore, DNA structure is implied by algorithmic arguments related to the fundamental duplication mechanism of this molecule, and DNA helix is implied by simple geometrical principles aimed at satisfying economy in space occupancy and an efficient associative mechanism based on hybridization. An important consequence of the helix arrangement of the two strands is the possibility of a second level of DNA packing where the DNA "rope", packed in the helix cylinder, can be again wound by wrapping it around some support (histone proteins, where DNA is wrapped around forming spherical packing units, glued by chromatin proteins, called nucleosomes). As is well, known in the craft of rope-making, even since ancient times, a "screw thread" is a helical ridge on the outside of a screw, or bolt, or on the inside of a cylindrical hole, to allow two parts to be screwed together. The same logic underlies a DNA formation, where strands are twisted together along a helix. This structure, not only provides the bilinearity postulated by the template driven duplication, but increases the robustness of the linear structure, making it able to keep its integrity in spite of its length, and in spite of other wrapping levels. Another illuminating comparison regarding DNA anti-parallelism can be found in folk dances where many dancers are arranged in two paired rows. Each dancer D corresponds to the dancer D' in front of him/her in the paired row, but due to the chirality of the human body, the dancer who is on the right of D corresponds to the dancer who is on the left of D'. It is surprising that rope technology and the art of dancing share important aspects with DNA structure.

Figure 2.14 shows 10 different forms of double DNA strings, where parallelism between lines refers to hybridization between strings, while Y forms correspond to the cases where extremal parts of single strings do not hybridize (in YY forms this happens on both sides). Of course, these forms do not cover all possible DNA forms, but they identify all the possibilities of pairing two different strands. Circular forms, hetero-duplex and hairpins are considered in Fig. 2.15.

A DNA simple branch is formed by three linear strands such that each of them hybridize with the other two. Such kinds of branches can produce any kind of graph, if we identify a cycle of branches as a single node. Figure 2.17 shows the way branches are combined for producing a node with degree five.

Table 2.2 Basic Requirements in DNA Pool Operations

$mix(P_1, P_2) = P_1 + P_2$
$split(P) = (P_1, P_2) \Leftrightarrow P = P_1 + P_2,\ Type(P_1) = Type(P_2) = Type(P)$
$length(P) = \{\lvert\eta\rvert \mid\ < \eta > \in Type(P)\}$
$separate(P, n) = \{s \in P \mid \lvert type(s)\rvert = n\}$
$Type(denature(P)) \supseteq \{\alpha \mid \frac{\alpha}{\beta} \in Type(P)\}$
$Type(hybridize(P)) \supseteq \{\frac{\alpha}{rev(\beta)} \mid \alpha, \beta \in Type(P), \alpha][\beta\}$
$Type(extend(P)) \supseteq \{\frac{\alpha\gamma\beta}{(\delta\gamma\beta)^c} \mid \frac{\alpha\gamma}{(\delta\gamma\beta)^c} \in P\}$
$Type(infix(P, \gamma, \delta)) \supseteq \{< \gamma\alpha\delta > \mid\ < \alpha > \in Type(P)\}$

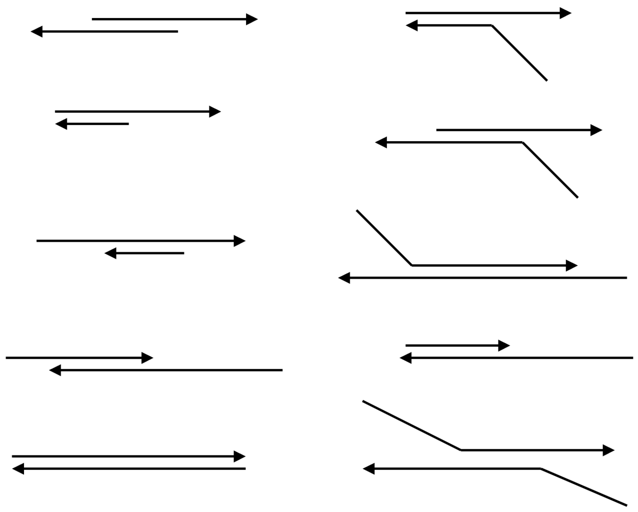


Fig. 2.14 Ten bilinear antiparallel forms

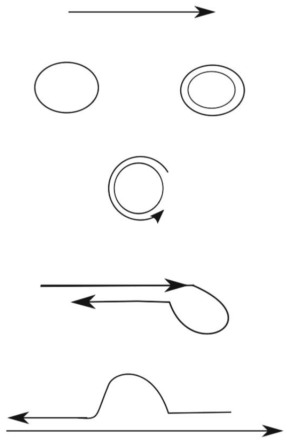


Fig. 2.15 Linear, circular, hairpin and heteroduplex DNA

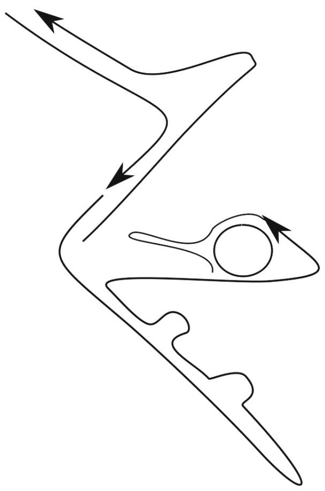


Fig. 2.16 A composition of heteroduplex, hairpin, and circular DNA

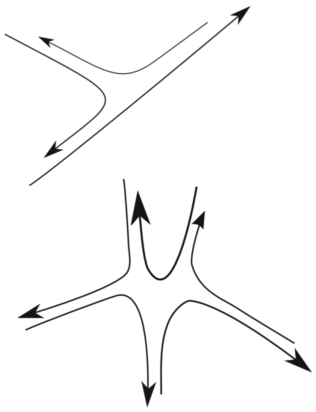


Fig. 2.17 Branching DNA and a composition of five branches

2.3 DNA Pool Operations

A pool of DNA molecules can be mathematically simplified by a multiset of double or single strings. In Table 2.2, some DNA pool operations are defined which have to be considered as high level mechanisms, analogous to the basic operations of a high level programming language. This implies that the algorithms we specify by means of them are not exactly experimental protocols, but rather computational procedures, implementable by laboratory procedures. These DNA algorithms are naturally expressed in terms of a *Test Tube register Language* (TTL for short), where registers are Test Tubes which contain DNA pools, rather than numbers, and DNA pool operations apply to them. The symbol $:=$ denotes the usual *assignment* command typical of imperative languages. In an assignment $P := op(Q)$, the operation op is intended to be applied to the content of the test tube Q and the resulting DNA pool becomes the content of the test tube P.

The exact "microscopic" effect of operations in Table 2.2 is not defined, because in many cases it is hard to say exactly what are the multisets before and/or after the application of these operations. However, despite such a kind of incomplete knowledge at a microscopic level, DNA algorithms can be designed which are based on these operations. In fact, it is enough to assume that, after performing a given DNA operation, an input pool P is transformed into a pool P' where a specific relationship holds between the types of P and P' (see Table 2.2 and Figs. 2.18, 2.19, 2.20).

The realization of operations in Table 2.2 is performed by means of standard laboratory procedures based on physical and biological phenomena. Operations *mix* and *split* are simply obtained by merging the content of two test tubes, or by splitting the content of one test tube in two different test tubes. Operations *length* and *separate* are realized by means of gel-electrophoresis, a specific electrochemical tool for discriminating DNA strands according to their length. The operation *hybridize* is obtained by raising temperature, while *renature* by (slowly) decreasing it. Operations *extend*, *ligate*, and *infix* are realized by Polymerase and Ligase enzymes (see Figs. 2.21, 2.22).

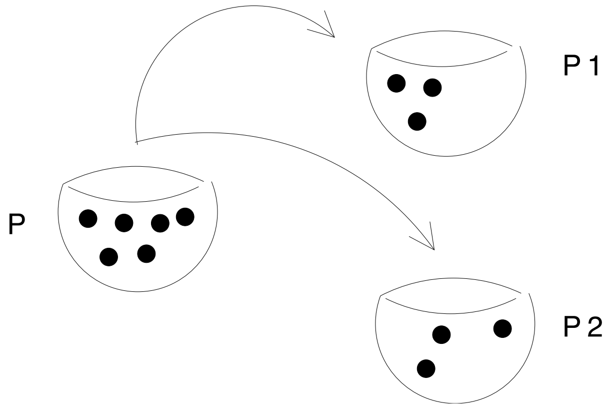


Fig. 2.18 Split operation (Mix is realized in the inverse way)

In Gel-electrophoresis DNA strands are put over a plate with a gel producing resistance to the movement of DNA molecules. An electrical field is applied between the two extremal borders of the plate (see Fig. 2.19). In column 1, strands of different (known) length are located at different levels. In the other columns, strands of unknown lengths stop at some level after the effect of the electrical field (DNA is negatively charged). Their positions, compared with the position of reference strands, provide a precise estimation of their lengths. The strands at a given

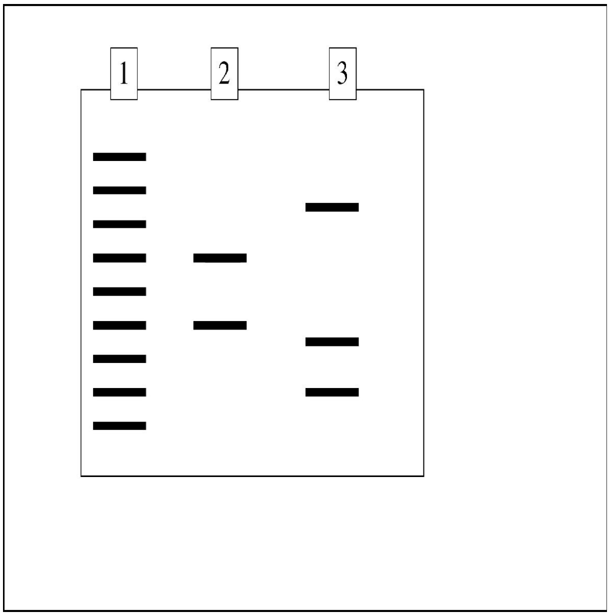


Fig. 2.19 Gel-electrophoresis implementing Length operation

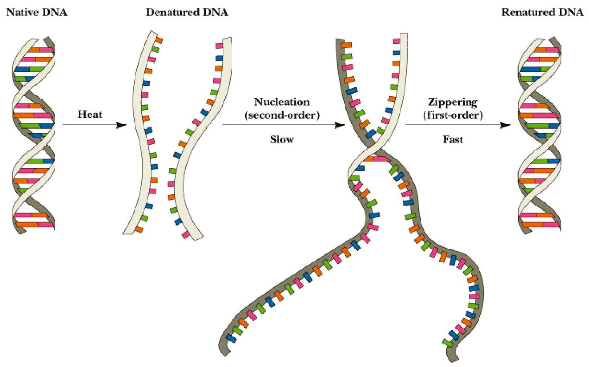


Fig. 2.20 DNA denaturation and renaturation

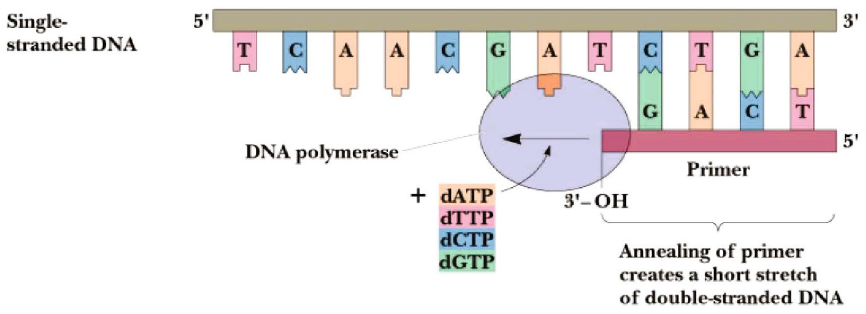


Fig. 2.21 Polymerase extension (Garret & Grisham: Biochemistry, Saunders College Publishing, 2e)

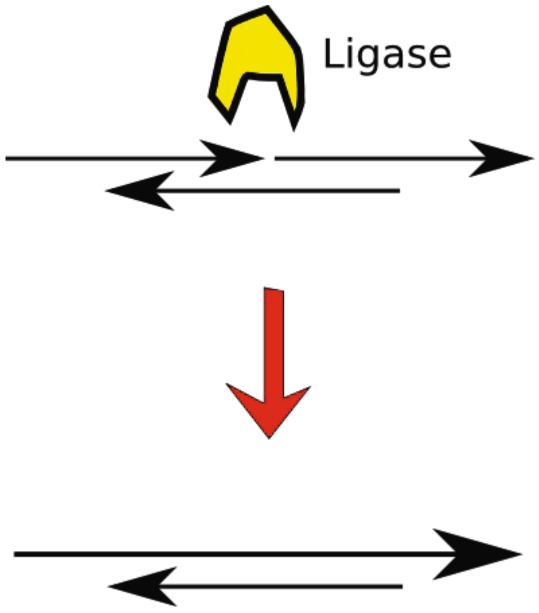


Fig. 2.22 Ligase joins 5′ phosphate to 3′ hydroxyl

position can be extracted from the plate, by cutting the gel slice where they are located, and, after removing the gel by *washing* the selected strands, the *separate* operation, providing strands of a required length, can be performed.

2.3.1 Writing and Reading DNA

A DNA **clone** is a multiset of DNA strands constituted by many copies of the same DNA molecule, or equivalently, a set of DNA strands or double strands having the same type. Two important aspects of DNA manipulation are the basic operation of any system of information representation: writing and reading. Writing DNA consists in generating a clone of DNA molecules having as type a given string over the alphabet of nucleotides. Reading means the inverse operation, that is, given a clone of DNA molecules, discovering which is the string of bases corresponding to their type.

The first operation, which is called *DNA synthesis*, is now a routine operation and is based on the following principle. Consider nucleotides where one of the two extremities is removed, say the $5'$ of the phosphoric group. Let us group in different vessels containing a given number of each nucleotide deprived of the P-terminal: a vessel A containing A_{OH} (instead of ${}_PA_{OH}$), a vessel T containing T_{OH}, a vessel C containing C_{OH}, and a vessel G containing G_{OH}. Let us write, for example, the sequence ATTCG. We start with a nucleotide (or better, a clone of nucleotides) ${}_PG_{OH}$ bound to a solid support S by means of some affinity mechanism, which anchor them to S, say it $S{+}_PG_{OH}$. Then we put $S{+}_PG_{OH}$ in the vessel C. The loss of P-extremity of nucleotides in this vessel ensures that when $S{+}_PG_{OH}$ is put in the vessel only $S{+}C_{OH}\,{}_PG_{OH}$ can be formed, where C nucleotides are without the P-terminal. Therefore, we add to $S{+}C_{OH}\,{}_PG_{OH}$ the missing P-terminal. by obtaining molecules $S{+}_PC_{OH}\,{}_PG_{OH}$. The same process can start again with sequences CG anchored to the support S and having both the extremities, for completing the process, by adding the remaining TCG part. In general, writing is performed by anchoring to a solid support S the last nucleotide of the sequence we want to write ($3'$ terminal), and then, by proceeding in the verse $3' - 5'$, by iterating the cycle of: i) introducing the solid support S in the right vessel for linking molecules inside it to the OH-terminal of molecules anchored to S, ii) extracting S from the vessel, and iii) adding the P-terminal to the molecule anchored to S.

Reading DNA is the process usually indicated as *DNA sequencing*. In fact, it provides the sequence bases corresponding to the type of a given DNA clone. In the reading process, it is essential to assume that DNA which has to be read is provided as a clone. We will explain the main idea of sequencing by using a metaphor. Let us assume the availability of a sequence of balls of four colors, say A, T, C, and G. Let us assume also that these balls are so small that we cannot distinguish their colors. However, let us suppose that we are able to obtain many copies of the given original sequence of balls. Then, we can distribute these copies in four different vessels. In each vessel we use some special scissors S_A, S_T, S_C, S_G such that S_A cuts after A, S_T cuts after T, S_C cuts after C, and S_G cuts after G. Moreover, in order to keep the right analogy with DNA, our balls are asymmetric, with a left part and a right part, in such a way that scissors can act only once for each strand, by cutting and by keeping in the vessels only the left parts. In these hypotheses, after using the corresponding scissors in each vessel, we get some strands which are copies of prefixes of the original sequence. If scissors perform their task in a completely random way, then in each vessel, say in A, we get sequences of different lengths which correspond to the positions where a ball of color A is located in the original sequence. In this manner, by measuring the lengths of the sequences obtained in the four vessels, we can discover the positions of the original sequence where A are located, those where T are located, those where C are located, and those where G are located. In conclusion, the structure of the original sequence can be completely deduced. This process was realized by Sanger, who won his second Nobel Prize for it, by using in each vessel a small part of nucleotides without the Oxydryl, in such a way that when polymerase enzyme uses one of them, then the extension process stops. It is important that the amount of these nucleotides is not so big and not so small. In fact in the first case only short fragments are obtained, while in the second case only long fragments are obtained. However, in almost all sequencing methods the role of polymerase extension is essential. In a new class of methods each step is constituted by four sub-steps, where polymerase can use only one of the four nucleotides, and its use is made evident by coupling to it a phenomenon which produces some effect. In this way according to which sub-step provides the effect, the kind of nucleotides added by polymerase can be deduced; therefore all the extended sequence can be deduced.

2.3.2 Plasmide Cloning Algorithm

DNA cloning of double strands is a process that produces many identical copies of a given double DNA molecule. Plasmid cloning is performed by means of bacteria. In fact, in many bacteria there are some double circular forms of DNA, called **plasmids**. Therefore, the main idea of plasmid cloning procedure consists in inserting a target DNA molecule inside a bacterium plasmid and then letting the bacterium proliferate in many cells which, being descendant of the same cell, have the same DNA and consequently the same plasmid including the target DNA molecule. In this way, we can get many copies of the initially inserted DNA, just by recovering all these plasmids and by extracting from them the copies of the target molecule.

The following is the detailed (abstract) procedure.

1. Choose a plasmid, called also **vector** which includes a gene encoding the resistance to an antibiotic A and a second gene encoding the resistance to an antibiotic B.
2. Cut this circular vector by means of a suitable *restriction enzyme* occurring once in the plasmid and in the middle of the sequence of gene A (see Fig. 2.23). As indicated in the figure, the cut of the restriction enzyme realizes two specific single strand flanking terminals.
3. Extend the target molecule in a such way that it begins and ends with sticky ends that can hybridize with the sticky ends provided by the restriction enzyme (see Fig. 2.24).

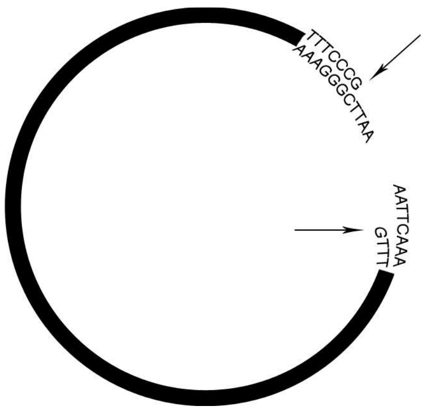


Fig. 2.23 Cutting the plasmid

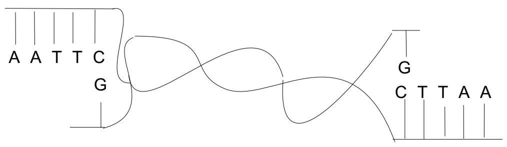


Fig. 2.24 Extending target DNA with suitable sticky ends

4. Let the target molecule hybridize with the broken plasmid, and use Ligase enzyme in order to obtain a circular double strand DNA molecule (see Fig. 2.25).

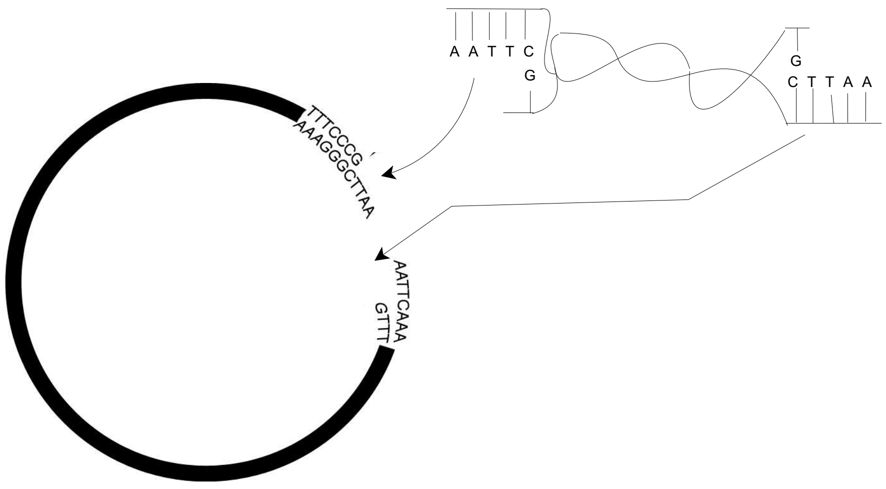


Fig. 2.25 Inserting the target DNA molecule into the plasmid

5. Infect bacteria that are not resistant to the antibiotics A and B with the plasmids obtained at the end of the previous step. The adopted technology (by suitable electrical impulses) ensures that only one plasmid can enter in one bacterium. In this operation three possible results can occur for each bacterium: i) one plasmid including the target molecule enters in the bacterium, ii) one plasmid without the target molecule enters in the bacterium, iii) no plasmid enters in the bacterium (the first situation occurs usually only for one in 10,000 bacteria).
6. Put bacteria obtained at the previous step in a microbial culture (each bacterium is put alone in one compartment) where the antibiotic B is put with the nutrient. In this way only bacteria including the plasmid can survive.
7. Select the bacteria which survive at the previous step and construct a **mirror bacterial colony** by choosing from each compartment one bacterium and putting it in a compartment having the same position it has in the original culture plate.
8. After a growing phase, put the antibiotic A in the mirror colony and observe in which position bacteria die. These positions are those where in the original colony there are bacteria hosting the plasmid which includes the target DNA molecule. In fact in these plasmids the resistance to the antibiotic A was removed. For example, if in the mirror colony bacteria in wells $2, 5, 8$ die, then we deduce that in the original colony these bacteria include the altered plasmid.
9. In the original colony keep only bacteria which correspond to dead bacteria, in the mirror colony. They include the plasmids hosting the target molecule. Remove the external membrane of these bacteria. and recover their plasmids.
10. Put in the obtained pool of plasmids the same restriction enzyme used at the second step. In this way copies of the target molecules, which were included in the plasmids, are recovered and can be selected by length with a gel-electrophoresis.

2.4 DNA Computing

In 1994 Leonard Adleman started the new research field of DNA Computing. [11, 12]. He showed that an instance of a famous combinatorial problem can be translated in terms of DNA strands, put in a test tube in such a way that, by means of typical laboratory manipulations, a final DNA pool is obtained where the solution of the problem is encoded. Since then, a great deal of research has been carried out, and many technical and theoretical achievements have been reached in DNA computing [39, 17, 18, 75]. Recently, new research perspectives have emerged that widen the possibilities of this field, among them: DNA self-assembly [61, 55, 58], and DNA automata [15, 14], as well as tools for DNA and RNA manipulation inspired by algorithmic analyses. [55, 13].

In the attempt to implement algorithms over a DNA-based "bioware", along with the DNA computing trend, it became increasingly apparent that the logic of DNA operations presents deep combinatorial and algorithmic aspects.

2.4.1 *Adleman's Experiment*

The main idea of Adleman's experiment is the representation of a graph in a DNA pool. To this end, it is enough to encode nodes with single DNA strands and edges as other strands in such a way that when an edge f connects two nodes a and b, then it is encoded by a strand which hybridizes with one half of the strand encoding a and one half of the strand encoding b. Figure 2.26 illustrates this idea of realizing connections with hybridization of single DNA strands. In this manner, when many copies of strands encoding all the nodes and edges of a graph are put in a DNA pool, then, in a very short time, they hybridize by providing all the possible paths which can be formed in the graph. Some specific operations involving the ligase and polymerase enzymes are necessary for having stable and abundand strands. Ligase fuses all the strands in the paths by performing the missing OH-P bonds between consecutive strands which are linearly arranged, while polymerase provides a generation of many copies of the formed paths, according to the polymerase chain reaction which we will study in a following section.

The question posed by Adleman, solved by DNA operations, was the determination of a Hamiltonian path connecting a start node with an end node (a path passing exactly once for each node of the graph). In terms of DNA strands, this corresponds to finding a DNA strand beginning with a given prefix and ending with a given suffix, having a given length (nk length if in the graph there are k nodes, all encoded by strands of length n) and where the encoding of each node is included. The solution (only one solution was possible in the problem considered by Adleman) was found by selecting by electrophoresis all the strands of the required length, and then by keeping only those which are *complete*, in the sense of including the encoding of all nodes. The check of completeness is performed by the crucial operation of **extraction**. Given a DNA pool P, the extraction of its γ-strands, provides strands of P having a type including the string γ. In Adleman's experiment, if $\alpha_1, \alpha_2, \ldots, \alpha_k$ are the strings encoding the nodes, then the strands solving the problem have to include types $\alpha_1, \alpha_2, \ldots, \alpha_k$. Let us consider k probes, that is, strands having types complementary to the strings $\alpha_1, \alpha_2, \ldots, \alpha_k$. Therefore, by using the first probe, all the strands are selected, by complementarity (we avoid the biochemical details), with types including the string α_1, then, from this selection, with a second probe, the strands can be selected which include α_2, and by proceeding in this way, after k steps (k the number of nodes of the graph), the required solution can be obtained. Sequencing the final strands, the required Hamiltonian path can be easily determined.

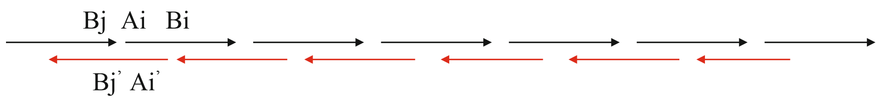


Fig. 2.26 Adleman experiment

The main steps of Adleman's experiments are synthesized in the following list.

1. Encode nodes and edges of the graph under investigation by means of suitable linear strands which hybridize by providing double strands encoding all the possible paths of the graph;
2. Add Ligase enzyme in order to transform hybridization paths into double strands (by concatenating contiguous strands);
3. Amplify, that is, realizes many copies (by using PCR, as will be explained later) of the double strands starting with the start node and ending with the end node of the graph;
4. Separate by length the double strands of the previous step encoding paths with 7 nodes (all the nodes of the graph);
5. Extract, from the paths of the previous step, those where all the nodes occur (each node once, because paths of the previous step contain 7 nodes);
6. The double strands remaining, after the extraction of the previous step, encode Hamiltonian paths of the given graph.

The procedure of Adleman's experiment has a general validity. In fact, in any combinatorial problem we can distinguish two different phases (see Fig. 2.27): the generation of a solution space where all possible solutions are produced, and a following phase where the true solution of the given problem is selected. Selection requires tools for discriminating between false and true solutions. The conceptual strength of DNA computing relies in the possibility of nature in performing massive string recombination processes, on the basis of the massive parallelism of strand hybridization in DNA pools with billions of billions of strands. The extraction operation is technologically more complex, expensive, and usually not very reliable. We will consider an extraction method in a following section. However, in principle, both generation and extraction operations can be performed in linear time, with respect to the size of the combinatorial problems. This general model of DNA computation, usually called *Adleman-Lipton extract model* [39] was tested and applied in many specific cases. After almost 10 years of research in this context, the initial enthusiasm about the computational efficiency of DNA computing sensibly diminished, but many aspects and many related fields of investigation were pushed by the Adleman's experiment and by all the research following it. We want only to mention the problem 3-SAT, on which a great deal of research effort was spent in recent years, for its centrality in the theory of combinatorial problems and for its natural setting in the context of DNA computing.

The problem 3-SAT consists in the determination of boolean assignments, to the variables $X_1, X_2, \ldots, X_n$, which satisfy a number of boolean equations $C_1, C_2, \ldots, C_m$ involving $X_1, X_2, \ldots, X_n$. It can be easily shown that without loss of generality boolean equations can be put in the form of *clauses*, that is, boolean sums of at most 3 *literals* (this explains the name 3-SAT) implicitly equated to the truth value true, where each literal can be a boolean variable or the negation of a boolean variable. For example, the system of boolean equations 2.1 ($= 1$ is implicit in every line) is a 3-SAT problem of 4 variables and three clauses which has, for example, the assignment $(X_1, X_2, \neg X_3, X_4)$ as solution.

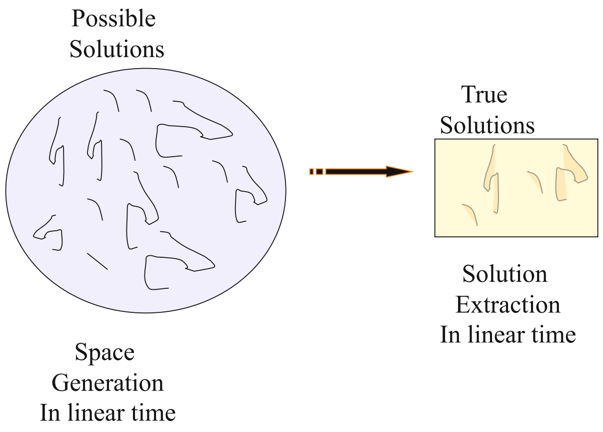

Possible Solutions
True Solutions
Solution Extraction In linear time
Space Generation In linear time

Fig. 2.27 The extract model

$$\begin{aligned} X_1 + \neg X_2 + X_3 \\ X_2 + X_3 + \neg X_4 \\ \neg X_1 + \neg X_2 + \neg X_3 \\ X_1 + \neg X_2 + X_3 \\ \neg X_2 + \neg X_3 + X_4. \end{aligned} \tag{2.1}$$

A natural way for generating a DNA space representing all possible assignments of a 3-SAT is the Mix-and-split method, which is based on a very simple idea depicted by Fig. 2.28 and expressed in TTL notation in Table 2.3.

Table 2.3 Mix-and-split procedure

```
Mix $X1$ and $\neg X1$ in a tube $T$
For J := 2 to $N$ do
                    Split $T$ into $A$ and $B$
                    Extend strands of $A$ with $Xj$
                    Extend strands of $B$ with $\neg Xj$
                    Mix $A$ and $B$ into $T$
```

By using a split-and-mix procedure all assignments of a 3-SAT problem can be generated and by applying to them m consecutive selection steps for filtering the assignments satisfying all the clauses, then the required solution can be found. The algorithm given in Fig. 2.29 is due to Lipton and provides 3-SAT solutions by means of $3m$ extraction steps.

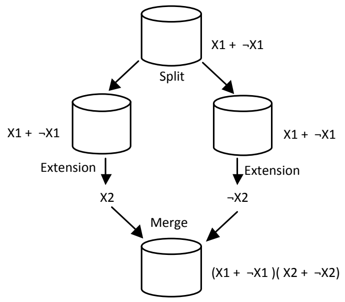
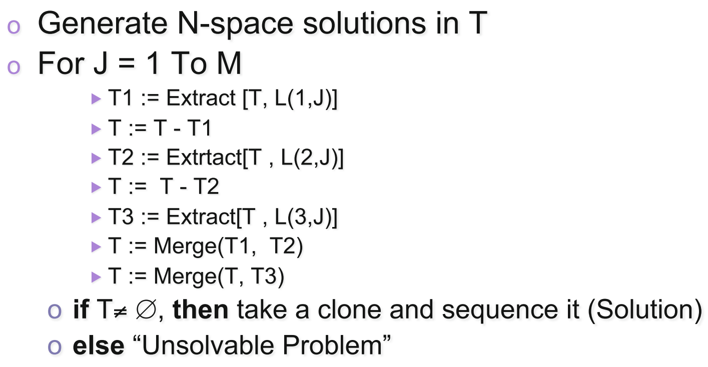


Fig. 2.28 Mix-and-split method



```
o Generate N-space solutions in T
o For J = 1 To M
      ▸ T1 := Extract [T, L(1,J)]
      ▸ T := T - T1
      ▸ T2 := Extrtact[T , L(2,J)]
      ▸ T :=  T - T2
      ▸ T3 := Extract[T , L(3,J)]
      ▸ T := Merge(T1,  T2)
      ▸ T := Merge(T, T3)
   o if T≠ ∅, then take a clone and sequence it (Solution)
   o else "Unsolvable Problem"
```

Fig. 2.29 Lipton's algorithm for 3-SAT. Expression $T - T'$ denotes the strands of T where the extracted strands of T' are removed.

We conclude by mentioning another three different ideas for solving 3-SAT in terms of DNA computing. The first one due to Jonoska et al. represents a given instance of the 3-Sat problem by producing, by means of DNA branches, a graph of the type given in Fig. 2.30, where clauses are represented by joining double DNA encodings of clauses (by means of branches as depicted in Fig. 2.17). Given a great number of DNA structures representing the graph of a 3-SAT problem, we can proceed in m steps ($j = 1, m$) by splitting, at step j, the pool of graphs in two pools and cutting (by using specific enzymes able to recognize some strings and to cut them) the literal X_j in the first pool and the literal $\neg X_j$, in the second pool, and then, by mixing again the two pools. It is easy to verify that if the problem has solutions, then a connected graph remains linking all the clauses, where each path of literals represents a solution of the original problem.

An algorithm due to Sakamoto et al. yields solutions according to an idea similar to that of Jonoska's algorithm. Let us suppose to order the clauses of a given problem. Let us encode the literals of every clause in such a way that they are linkable

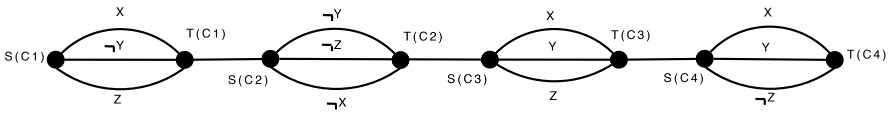


Fig. 2.30 3-SAT solutions represented by graphs encoding assignments

by means of two sticky ends (left and right) with the literals of the previous clause (on the left) and with the literals of the next clause (on the right, see Fig. 2.31). Moreover, for every variable X, let the literal $\neg X$ be encoded with a string that is the the mirror of the encoding of X. Of course, literals of the first clause are linkable only with those of the second, while literals of the last clause are linkable only with those of clause of order $m-1$ (m number of clauses). In this way, if a a path, of linked literals, is formed where a variable and its negation are encoded, then a DNA hairpin is formed. Therefore, in general with n variables, the true solutions are the paths encoding n literals that are not DNA hairpin.

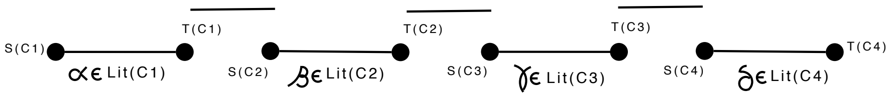


Fig. 2.31 3-SAT solutions represented by sequences encoding literals

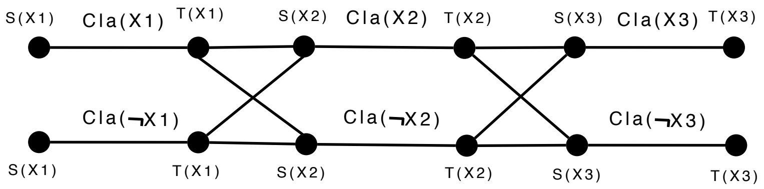


Fig. 2.32 3-SAT solutions represented by sequences encoding clauses

An algorithm due to Manca and Zandron [46] provides $2n$ strings (n number of boolean variables) $Cla(X_i)$ and $Cla(\neg X_i)$, for $1 \leq i \leq n$. The string $Cla(X_i)$ is a sequence (double string with sticky ends) of DNA encodings of all clauses having the literal X_i, and $Cla(\neg X_i)$ is a sequence of DNA encodings of all clauses having the literal $\neg X_i$. Moreover, $Cla(X_i)$ and $Cla(\neg X_i)$ are left-linkable to $Cla(X_{i-1})$ and to $Cla((\neg X_{i-1})$, while they are right-linkable to $Cla(X_{i+1})$and to $\neg Cla((\neg X_{i+1})$ (by means of suitable sticky ends, see Fig. 2.32). It is easy to show that the solutions of a 3-SAT problem are represented by those strands which encode sequences of n strings of type $Cla(--)$ containing the encodings of all the clauses of the given problem. In the case of m clauses, m extractions are sufficient to solve a 3-SAT problem.

2.5 PCR and XPCR Protocols

Polymerase Chain Reaction process (PCR) is one of the most important and efficient tools in biotechnological manipulation and analysis of DNA molecules, where the polymerase enzyme implements a very simple and efficient duplication algorithm on double oriented strings. Kary Mullis discovered this method in 1983 [56], and for this reason, the Nobel Prize in chemistry 1993 was awarded to him (jointly with Michael Smith for contributions to site-directed mutagenesis).

The main result of PCR is the **exponential amplification** of target double strands having a given sequence of bases. As will be explained in this section, in the PCR process, an initial number of target double strands (even only one of them) provides a final number of them which is a product of the initial number by a multiplicative factor consisting of a power of the times a basic PCR step is repeated. The bilinearity of DNA molecules and the antiparallel orientation of their two linear components are essential aspects of the logic underlying PCR. The computational schema of PCR in TTL notation is given in Table 2.4, while its combinatorial schema in terms of string transformations is given in Fig. 2.33.

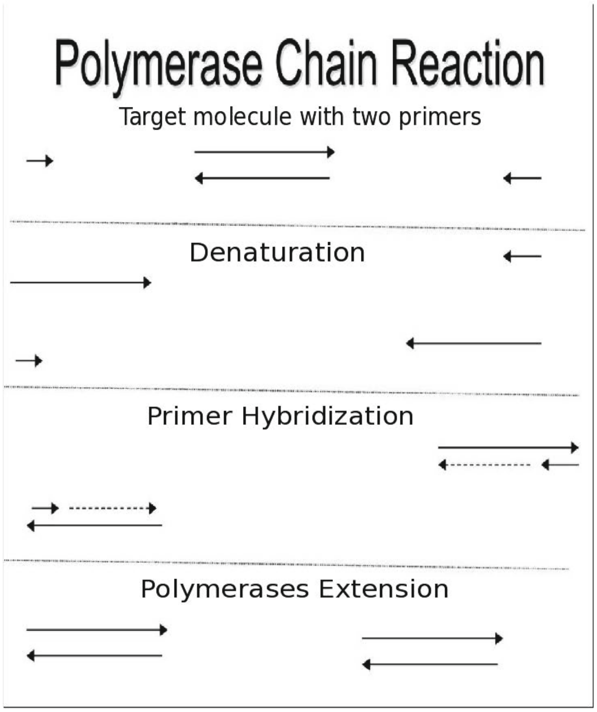


Fig. 2.33 The schema of Polymerase Chain Reaction

A substantial improvement in the efficiency and reliability of PCR was obtained when the process was realized by means of heat-stable DNA polymerase, such as Taq polymerase (an enzyme isolated from the bacterium Thermus aquaticus). Namely, in this way the three phases realizing the denaturation, primer hybridization, and polymerase extension can be realized by alternately heating and cooling the PCR sample to a defined series of temperature steps (depending on the length and nucleotide composition of the hybridization regions of the involved strands).

Polymerase Chain Reaction, which is a milestone in DNA recombinant technology [1], shows many interesting combinatorial properties. In fact, it is in some cases a complicated process, and, when it is used in non-standard ways, it yields very complex behaviors. Very often, anomalies are ascribed to experimental noise, but, if we frame PCR within a rigorous symbolic notation, then non-trivial combinatorial aspects appear and, under suitable hypotheses, a formula can be derived which describes the general form of sequences that are exponentially amplified. This approach is not of merely mathematical interest. On the contrary, it can suggest new methods that enjoy biological relevance for in vitro DNA manipulation. In fact, starting from DNA computing problems [46], we investigated specific methods for DNA extraction and recombination [28, 30, 26]. In these attempts, where theoretical issues were supported by the experiments, we realized that a special kind of PCR, called *Cross Pairing PCR* or *XPCR* for short, can be the basis for new algorithms that solve a wide class of DNA extraction and recombination problems. Within a basic DNA symbolic notation, we will show a crucial computational property of Polymerase Chain Reaction, we call *PCR Lemma*, which suggested to us the idea of Cross Pairing PCR.

Table 2.4 PCR Algorithm

```
$PCR(P,n) =$
let $Type(P) = \{<\gamma\ldots\delta>, \gamma, \bar{\delta}\}$, $n$ integer;
input $P$;
for $i = 1,n$ do
begin
    $P := denature(P)$;
    $P := hybridize(P)$;
    $P := extend(P)$;
end;
output $P$.
```

We denote by $PCR(P,n)$ the DNA pool obtained by applying to the pool P the procedure of Table 2.4 (the usual PCR process). The parameter n denotes the number of the fundamental PCR steps (also called PCR cycles, which we avoid

Table 2.5 Combinatorial Schema of Polymerase Chain Reaction

$\frac{\gamma\,\zeta\,\delta}{\gamma^c\zeta^c\delta^c} \rightarrow \frac{\gamma\zeta\delta}{\lambda}, \frac{\lambda}{\gamma^c\zeta^c\delta^c}$	*template denaturation*
$\frac{\gamma\zeta\delta}{\lambda}, \frac{\lambda}{\gamma^c\zeta^c\delta^c}, \frac{\gamma}{\lambda}, \frac{\lambda}{\delta^c} \rightarrow \frac{\gamma}{\gamma^c\zeta^c\delta^c}, \frac{\gamma\zeta\delta}{\delta^c}$	*primer hybridization*
$\frac{\gamma}{\gamma^c\zeta^c\delta^c}, \frac{\gamma\zeta\delta}{\delta^c} \rightarrow \frac{\gamma\,\zeta\,\delta}{\gamma^c\zeta^c\delta^c}, \frac{\gamma\,\zeta\,\delta}{\gamma^c\zeta^c\delta^c}$	*polymerase extension*

mentioning when its value is not essential in the discussion). It is well-known that if $Type(P) = \{< \gamma\varphi\delta >, \gamma, \bar{\delta}\}$, then $PCR(P,n)$ generates an *exponential amplification* of the *target molecule* $< \gamma\varphi\delta >$, that is, copies of this molecule in a number that is exponential with respect to the number n of steps, and only a minor quantity of strands in P have types different from $< \gamma\varphi\delta >$.

Polymerase extension *ext*, depicted in Fig. 2.21, of a single string $\alpha\gamma$, according to a template η, is defined as:

$$ext(\alpha\gamma, \bar{\beta}\bar{\gamma}) = \alpha\gamma\beta$$

(assuming that γ does not occur as a substring of α, and β), that is:

$$ext(\alpha\gamma, \frac{\lambda}{\gamma^c\beta^c}) = \alpha\gamma\beta$$

If $\alpha\gamma \,]\!\!/\!\![\, \eta$, then we set $ext(\alpha\gamma, \eta) = \alpha\gamma$.

It is useful to consider another extension operation, which we continue to denote by *ext*, having a double string as argument, by setting (if the argument has a form different from that here considered, this *ext* leaves it unchanged):

$$ext(\frac{\alpha\gamma}{\gamma^c\beta^c}) = \frac{\alpha\gamma\beta}{\alpha^c\gamma^c\beta^c}.$$

PCR processes are easily representable by diagrams like that in Fig. 2.34, where dotted arrows represent *ext* operation performed by polymerase enzymes.

All the possible PCR diagrams which result from the different forms of target strings and from the different positions where primers may hybridize, are close to 100. Therefore, a natural question arises: in which cases does PCR provide exponential amplification? And, in these cases, is it possible to characterize, in general terms, the form of strings that are exponentially amplified? Our DNA notation allows us to answer these questions with the following lemma, which identifies important cases of exponential amplification by PCR processes.

We say that single DNA strings φ and ψ **overlap** when they hybridize according to the pattern depicted in Fig. 2.35. We define their **overlap concatenation**, by setting:

$$< \varphi >]\!\lfloor < \psi > = ext(\frac{\varphi}{\psi^c})$$

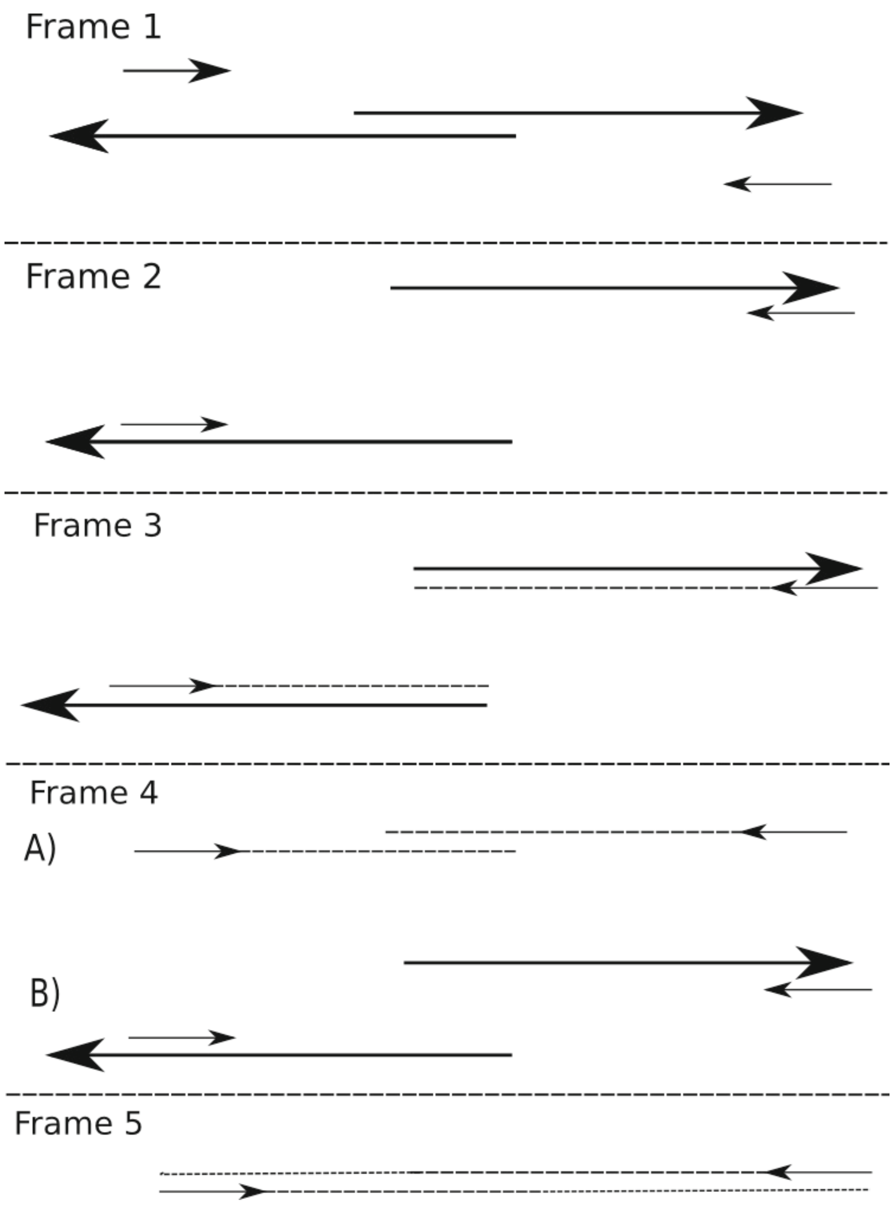


Fig. 2.34 PCR on a $3'3'$ form along with one internal and one external primer. Reading from the top: after the hybridization of the primers (short arrows), and the formation of their extensions, these extensions overlap and, with the further extensions of the overlapping strings, a blunt string is formed where primers correspond to the two (forward and backward) extremities.

which produces the DNA string $<\varphi'\gamma\psi'>$ depicted in Fig. 2.36, where γ is the substring of φ, $\bar{\gamma}$ of ψ, and φ', ψ' are the strings φ and $\bar{\psi}$ without the substrings γ and $\bar{\gamma}$, respectively.

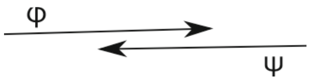


Fig. 2.35 Two DNA single strings that overlap

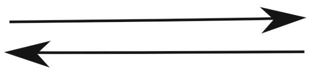


Fig. 2.36 The overlap concatenation of two strings $<\varphi>$ and $<\psi>$ from the (overlap) hybridization displayed in Fig. 2.35

Lemma 2.1 (PCR Lemma). *Let P be a DNA pool such that $Type(P) = \{\frac{\alpha}{rev(\beta)}, \gamma, \bar{\delta}\}$ and let:*

$\gamma \,]\!\!/\!\![\, \bar{\delta}$,
$\gamma \,][\, \beta$,
$\bar{\delta} \,][\, \alpha$.

In the following cases:

i) $\gamma \,][\, ext(\bar{\delta}, \alpha)$,
ii) $\bar{\delta} \,][\, ext(\gamma, \beta)$,
iii) $ext(\gamma, \beta)$ *and* $ext(\bar{\delta}, \alpha)$ *overlap,*

$PCR(P, n)$ exponentially amplifies blunt strings, where primers hybridize at extremal regions of their single strands, having the following forms, respectively:

$$ext(\bar{\delta}, ext(\gamma, ext(\bar{\delta}, \alpha))) \tag{2.2}$$

$$ext(\gamma, ext(\bar{\delta}, ext(\gamma, \beta))) \tag{2.3}$$

$$< ext(\gamma, \beta) >]\!\!\lfloor < ext(\bar{\delta}, \alpha) > . \tag{2.4}$$

Proof. By the hypotheses, in the cases i) and ii) the extensions (2.2) and (2.3) have to be proper (giving results different from the primers). It is easy to check that they are blunt double strings including the primers in the extremal parts of their strands, therefore they are seeds of exponential amplifications. In the last case iii), the situation of Fig. 2.35 occurs, then the overlap concatenation (2.4) (realized by the polymerase extension) yields a blunt string (with extremal primers) that is a seed of exponential amplification. □

The phenomenon of the blunt form of PCR exponential amplification is empirically well-known, but it is interesting that it is a mathematical consequence of the combinatorial mechanism on which PCR is based. The following corollary is a direct consequence of PCR lemma.

Corollary 2.2. *In the PCR process of lemma above where an exponential amplification occurs, then, at most at the third step of the process, a blunt string appears which is a seed of an exponential amplification.*

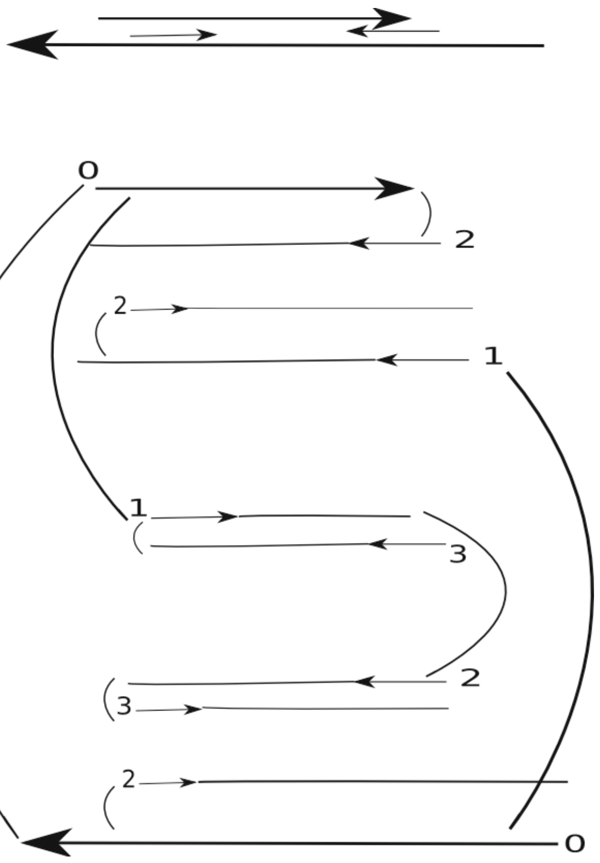


Fig. 2.37 PCR with two internal primers, where curved lines represent pairing

In Fig. 2.37, a PCR process is displayed, where the first two steps are completely represented, while, for easier reading, only two double strands of the third step are shown. One of them is a blunt double strand that is a seed of exponential amplification.

We remark that PCR lemma presented here is different from the analogous lemma given in [45]. Namely, here we do not claim to cover all the possible cases of amplification. In fact, some laboratory experiments have shown cases of exponential amplification with YY forms, where primers hybridize with flanking regions (and produce amplifications having patterns different from those given in Eqs. 2.2, 2.3, 2.4). Here we do not analyze all the cases of PCR with YY forms. The example of Fig. 2.39 shows the complexity of the combinatorial patterns related to PCR amplification. In this figure, starting with a YY molecule M1 and two primers that hybridize on two flanking regions, the molecule M2 is realized, which consists of the extensions of the two primers. Molecules M3 and M4 are obtained by hybridization of molecules of type M2 with the two strands of M1. In these molecules, it easy to realize that the extensions of the primers stop when polymerase reaches the other flanking regions. The resulting extensions hybridize and produce the molecule M5, that, after polymerase extension, becomes the blunt molecule M6, which is seed of an exponential amplification.

An example where PCR lemma does not apply is displayed in Fig. 2.38.

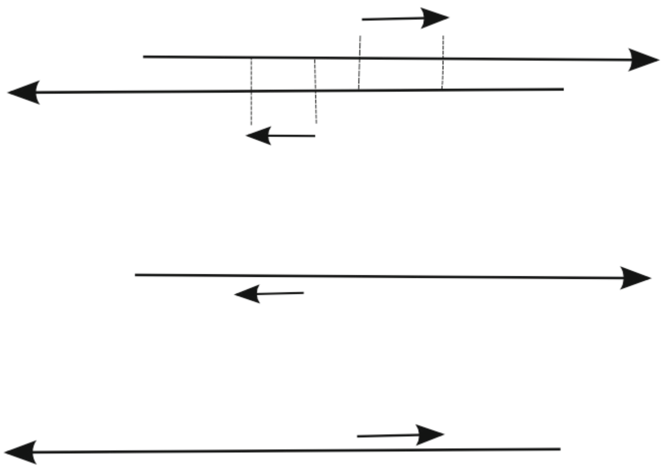


Fig. 2.38 A case where PCR lemma does not apply

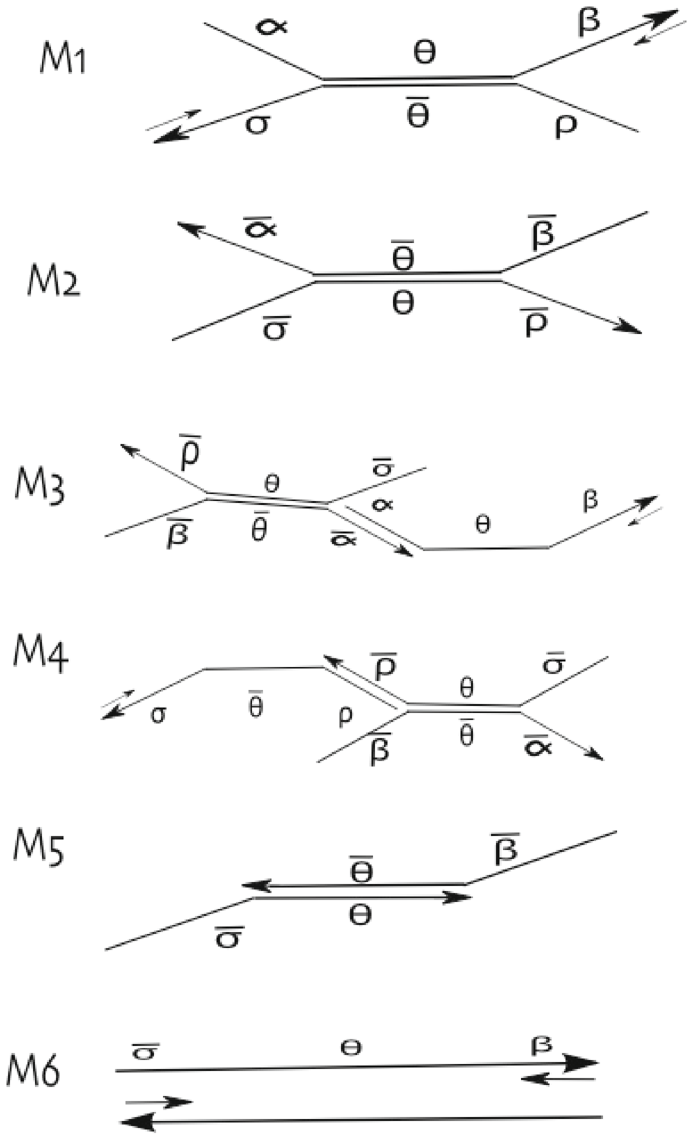


Fig. 2.39 An exponential amplification based on a YY form

2.5.1 XPCR

The role of the overlap concatenation in PCR amplification suggested to us an interesting form of PCR where, rather than just one target string, we put two target DNA double strings of types $< \alpha\varphi\gamma >$ and $< \gamma\psi\beta >$ in a test tube.

Let us analyze what happens if a PCR is performed, starting from such a pool, extended with primers $\alpha, \bar{\beta}$:

$$< \alpha\varphi\gamma > \ , \ < \gamma\psi\beta > \ , \ \alpha \ , \ \bar{\beta}$$

In this case, after denaturation the following strings will be present in the pool:

$$\frac{\alpha\varphi\gamma}{\lambda} \ , \ \frac{\lambda}{(\alpha\varphi\gamma)^c} \ , \ \frac{\gamma\psi\beta}{\lambda} \ , \ \frac{\lambda}{(\gamma\psi\beta)^c} \ , \ \alpha \ , \ \bar{\beta} \ .$$

If the temperatures at which α, β, and γ hybridize with their mirror strings are ‘close enough’, then the following hybridizations can occur, where the first string, in the pool given above, overlaps with the fourth one, while the strings in second and third positions hybridize with the primers (other hybridizations are possible, but it is easy to realize that they are not “productive” in terms of amplification, because they can only delay the effect of these “canonical” hybridizations, see Fig. 2.40):

$$\frac{\alpha\varphi\gamma\qquad}{\qquad(\gamma\psi\beta)^c} \ , \ \frac{\alpha\qquad}{(\ \alpha\ \varphi\ \gamma)^c} \ , \ \frac{\gamma\ \psi\ \beta}{\qquad\beta^c} .$$

At this point polymerase can extend strings:

$$ext(\frac{\alpha\varphi\gamma\qquad}{\qquad(\gamma\psi\beta)^c}) \ , \ ext(\frac{\alpha\qquad}{(\ \alpha\ \varphi\ \gamma)^c}) \ , \ ext(\frac{\gamma\ \psi\ \beta}{\qquad\beta^c})$$

that become:

$$< \alpha\varphi\gamma\psi\beta > \ , \ < \alpha\varphi\gamma > \ , \ < \gamma\psi\beta >$$

in such a way that the blunt string $< \alpha\varphi\gamma\psi\beta >$ is produced, which results from the overlap concatenation of $< \alpha\varphi\gamma >$ and $< \gamma\psi\beta >$ with two more strings equal to the initial target strings. We call this special kind of *PCR cross pairing PCR* or *XPCR* for short; its combinatorial schema implements an operation, which theoretically can be seen as a special kind of Tom Head’s *null context splicing rule*, given in the seminal paper [37] where a Formal Language Theory perspective of DNA strings was introduced (see also [47, 43]). In conclusion, if

$$Type(P) = \{< \alpha\varphi\gamma >, < \gamma\psi\beta >\}$$

then

$$Type(PCR(P, \alpha, \beta)) \approx \{< \alpha\varphi\gamma\psi\beta >\}.$$

Types of strands different from $< \alpha\varphi\gamma\psi\beta >$ are present in a minor quantity, moreover, recombinations different from those considered above are possible (for example, that of type $3' - 3'$ between strands $\frac{\gamma\psi\beta}{\lambda}$ and $\frac{\lambda}{(\alpha\varphi\gamma)^c}$). Figure 2.40 displays the possibility of hybridizations in a pool with two target strings and two extremal primers (boxes includes the equal substrings in the two target molecules). It is easy to check that the overall effect, in any case, is the amplification of the overlap concatenation of the two target molecules. In fact, either the doubling of the two target molecules plus a seed of overlap concatenation, or two seeds of overlap concatenation plus the initial pair of target molecules is obtained after hybridization.

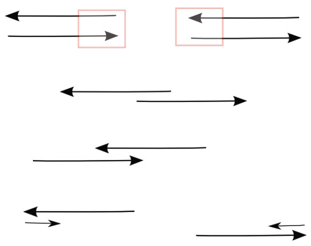


Fig. 2.40 Possible hybridizations of *XPCR*

We tested *XPCR* in several different experimental conditions and every time it provided correct results [28, 30]. In most cases, the amplification signal is very clear and a small noise consisting of unspecific products is reported in the electrophoresis results. Surprisingly enough, on the basis of the simple mechanism of *XPCR*, we have been able to build more sophisticated procedures, which find two main kinds of applications in DNA extraction and DNA recombination.

2.5.2 *DNA Extraction by XPCR*

DNA extraction is a fundamental procedure where the "good solutions" are discriminated in a space of possible solutions. For example, given a DNA pool consisting of a family of genes with an indefinite identity (their sequences are not known), one might be interested in extracting the subfamily of those genes where a given subsequence γ occurs, which, for instance, refers to an important biological property. The classic extraction procedure *by affinity* uses a probe $\bar{\gamma}$ which is "marked" in such a way that, after denaturation, single strands where γ occurs hybridize with the probe, and so are selected from the original pool. Here we show a different way of performing extraction, based on *XPCR*, outlined in TTL notation, in Table 2.6 ($P - separate(P, n)$ denotes the pool P after removing its strands of length n).

The main idea of the algorithm is as follows. Consider all the lengths of strands in a given pool. For each length n, pieces of strands which include the substring γ in their types are copied. This is performed by means of usual PCR to amplify both strands of type $< \alpha\gamma >$ and $< \gamma\beta >$. Strings shorter than n are then separated by length and finally joined by an overlapping concatenation performed by *XPCR*.

Table 2.6 A DNA extraction algorithm providing the strings of length n including γ as substring, in a given a pool P

```
$XPCR - Extract(P, n, \gamma)$
1.  $P := infix(P, \alpha, \beta)$;
2.  $P_0 := separate(P, n)$;
3.  $(P_1, P_2) := split(P_0)$;
4.  $P_1 := PCR(P_1, \alpha, \bar{\gamma})$;
5.  $P_2 := PCR(P_2, \gamma, \bar{\beta})$;
6.    $P_1 := P_1 - separate(P_1, n)$;
7.    $P_2 := P_2 - separate(P_2, n)$;
8.  $Q := mix(P_1, P_2)$;
9.  $Q := PCR(Q, \alpha, \bar{\beta})$;
10. $Q := separate(Q, n)$;
11. **output** $Q$.
```

In the joining process, pieces which do not have length n are removed. For this reason, the process must be iterated for each length of strings of the initial pool. The algorithm reported in Table 2.6 provides all the strings where γ occurs, previously elongated by the prefix α and by the suffix β. This algorithm was tested in vitro where, in a very heterogeneous DNA pool, all the types of strands including substrands of a given type, and only they, were extracted. Therefore, *XPCR* extraction proved to be correct and complete [28].

A useful warning about the *XPCR*-Extract algorithm is given by the following observation. If in a family of initial genes there are two different genes, say $< \varphi\gamma\psi >$ and $< \sigma\gamma\rho >$, that have the same length and where the substring γ occurs in exactly the same position, then the method will give, as extracted genes, also their chimeric combinations $< \sigma\gamma\psi >$ and $< \varphi\gamma\rho >$. In other words, if we define $Recombine(L, \gamma) = \{< \alpha\gamma\beta > \mid < \alpha\gamma\delta >, < \eta\gamma\beta > \in L\}$, then $Extract(P, \gamma)$ coincides with $Recombine(Type(P), \gamma)$. In this case, further checks are necessary for realizing a reliable extraction.

2.5.3 DNA Recombination by XPCR

Let P a DNA pool of type $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and Q a pool of type $\{\beta_1, \beta_2, \ldots, \beta_n\}$. The problem of generating all possible recombinations of pools P and Q is that of obtaining a pool of type $L = \{\eta_1\eta_2\ldots\eta_n \mid \eta_1 \in \{\alpha_1, \beta_1\}, \eta_2 \in \{\alpha_2, \beta_2\} \ldots \eta_n \in \{\alpha_n, \beta_n\}\}$. Of course L contains 2^n different strings; we call it the n dimensional complete recombination of P and Q. In DNA Computing this is an important step for encoding all the possible solutions of a combinatorial problem. For example, if α_i, β_i encode the two possible values of a Boolean variable x_i, then any string of L usually encodes a possible solution of the problem. True solutions are obtained by generating L and then by extracting strings which satisfy the requirements of the problem. But, apart from this DNA computing interest, DNA recombination has an important biological meaning. For example, in the immunological system recombination is a key feature for generating the antibody repertory which is essential to the security system preserving the biological identity. However, in general, DNA pools generated by *XPCR* recombination could be very useful in the analysis of many aspects related to DNA hyper-variability.

Let us present a DNA recombination method based on *XPCR*. We show that a simple and efficient algorithm based on *XPCR* can provide the n dimensional complete recombination of P and Q. Let us consider the following four *initial sequences*, where n is an odd number (if n is even, the roles of α_n and β_n are inverted in the last two sequences):

Positive: $\eta_1 =< \alpha_1\alpha_2\alpha_3\alpha_4\dots\alpha_n >$
Negative: $\eta_2 =< \beta_1\beta_2\beta_3\beta_4\dots\beta_n >$
Positive-Negative: $\eta_3 =< \alpha_1\beta_2\alpha_3\beta_4\dots\alpha_n >$
Negative-Positive: $\eta_4 =< \beta_1\alpha_2\beta_3\alpha_4\dots\beta_n >$.

Let us call α-string any element of $\{\alpha_1,\alpha_2,\dots,\alpha_n\}$ and β-string any element of $\{\beta_1,\beta_2,\dots,\beta_n\}$. The language L is the set of all the possible ordered combinations of n α-strings and β-strings. We call *XPCR* rule r_γ, and write it as $\xi_1,\xi_2 \xrightarrow{r_\gamma} \zeta$, the relation between strings ξ_1,ξ_2,ζ that holds when $\xi_1 =< \alpha\delta\gamma\dots >$, $\xi_2 =< \dots\gamma\theta\beta >$, and $\zeta =< \alpha\delta\gamma\theta\beta >$. Any string that is a combination of α-strings and β-strings can be obtained from η_1, η_2, η_3, η_4 by suitable *XPCR* rules r_γ where γ is an α-string or a β-string. For example, with $n = 7$, the string $\alpha_1\alpha_2\beta_3\alpha_4\beta_5\beta_6\beta_7$, can be obtained in the following way:

$$\eta_1\ ,\ \eta_4 \xrightarrow{r_{\alpha_2}} \alpha_1\alpha_2\beta_3\alpha_4\beta_5\alpha_6\beta_7\ ,\ \eta_2 \xrightarrow{r_{\beta_5}} \alpha_1\alpha_2\beta_3\alpha_4\beta_5\beta_6\beta_7.$$

It can be easily shown that the order of application of the rules is not relevant, because the same string can be also obtained by permuting the order of application of the rules (from the same initial strings).

Let us consider the set of *XPCR* rules $R = \{r_{\alpha_2}, r_{\alpha_3}, \dots r_{\alpha_{n-1}}, r_{\beta_2}, r_{\beta_3}, \dots r_{\beta_{n-1}}\}$. A "quaternary *XPCR* recombination" which produces L from $\{\eta_1,\eta_2,\eta_3,\eta_4\}$ is effectively specified by the algorithm displayed in Table 2.7. A completeness claim of our quaternary *XPCR* recombination method is given by the following proposition, which is an easy consequence of *XPCR* definition and of the particular structure of the pool to which we apply this recombination method [30].

Proposition 2.3 (Recombination Method). *If all the XPCR rules of R are applied to a DNA pool of Type* $\{\eta_1,\ \eta_2,\ \eta_3,\ \eta_4\}$*, then a final DNA pool is obtained of Type* $\{\xi_1\xi_2\dots\xi_n \mid \xi_1 \in \{\alpha_1,\beta_1\}, \xi_2 \in \{\alpha_2,\beta_2\}\dots\xi_n \in \{\alpha_n,\beta_n\}\}$.

Table 2.7 The Quaternary Recombination Algorithm

```
$XPCR-Recombination(\{\alpha_1,\alpha_2,\ldots\alpha_n\},\{\beta_1,\beta_2,\ldots\beta_n\}) =$
**let** $Type(P) = \{\eta_1,\eta_2,\eta_3,\eta_4\}$;
$P := infix(P,\alpha,\beta)$;
**for** $i = 2, n-1$ **do**
**begin**
    $P := PCR(P,\alpha,\bar{\alpha}_i)$;
    $P := PCR(P,\alpha_i,\bar{\beta})$;
    $P := PCR(P,\alpha,\bar{\beta}_i)$;
    $P := PCR(P,\beta_i,\bar{\beta})$;
    $P := PCR(P,\alpha,\bar{\beta})$;
**end**;
**output** $P$.
```

The quaternary recombination algorithm has the additional advantage of being equipped with a couple of special strings, called *recombination witnesses*, such that, if they are present in the final pool then the whole library of possible recombinations is present too. Namely, let us consider a pool P including the four strings of quaternary recombination (prefixed by the string α and suffixed by the string β), then a set W of stings is a *XPCR* **witness** set, if when all the strings of W are included into P, then all the possible XPCR recombinations where realized from the four initial strings. Let us call i-**trio-factor** any substring of three components $\gamma_{i-1}\gamma_i\gamma_{i+1}$ where exactly two consecutive components are positive (α strings) or negative (β strings), then the following lemmas can be shown [30, 26].

Proposition 2.4. *The recombination according to the XPCR rule r_γ ($\gamma = \alpha_i$ or $\gamma = \beta_i$ for $1 < i < n$) was realized in the pool P, at the end of the quaternary recombination algorithm starting with the four initial strings, if the i-trio-factor including as component γ is present in some string of P.*

Proposition 2.5. *The set consisting only of the following two strings is a XPCR witness set ($n \geq 6$): $w_1 = \alpha\alpha_1\alpha_2\beta_3\beta_4\alpha_5\alpha_6\ldots\beta$ and $w_2 = \alpha\beta_1\beta_2\alpha_3\alpha_4\beta_5\beta_6\ldots\beta$.*

2.6 L-Systems and Morphogenesis

L-systems, introduced by Aristid Lindenmayer in 1968 [48, 53, 213], are grammars with parallel rewriting. In particular, *EOL* is the class of L-systems which can be defined as *CF* grammars where rules are applied, by a parallel rewriting of all the symbols occurring in the string. It is really surprising that a wide class of developmental processes can be described by suitable L-systems, possibly extended with elements

Table 2.8 An EOL system generating the tri-somatic language

S	$\rightarrow$	ABC
A	$\rightarrow$	AA'
B	$\rightarrow$	BB'
C	$\rightarrow$	CC'
A	$\rightarrow$	a
B	$\rightarrow$	b
C	$\rightarrow$	c
A'	$\rightarrow$	A'
B'	$\rightarrow$	B'
C'	$\rightarrow$	C'
A'	$\rightarrow$	a
B'	$\rightarrow$	b
C'	$\rightarrow$	c
a	$\rightarrow$	F
b	$\rightarrow$	F'
c	$\rightarrow$	F
F	$\rightarrow$	F

conveying important morphological parameters (e. g., lengths, angles, sizes). The EOL system of Table 2.8 generates the tri-somatic language.

The first application of L-systems were the developments of some algae (red alga). In these organisms all cells grow, at each developmental step. This means that, in symbols, this mechanism corresponds to a parallel rewriting of all the symbols occurring in a string.

A DOL system (Deterministic OL system) is defined by a triple $(\mathbf{A},\mu,\alpha)$ where $A=\{a_1,a_2,\cdots,a_n\}$ is the *alphabet*, α is the **initial word** over the alphabet A, and μ is a **string morphism** over A, that is a function from A to A^* which extends to a function from A^* to A^* by the condition

$$\mu(\alpha\beta)=\mu(\alpha)\mu(\beta).$$

The following is the usual representation of a DOL system, where the morphism is expressed in a evident arrow notation.

$$\mu=\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ \downarrow & \downarrow & & \downarrow \\ \beta_1 & \beta_2 & \cdots & \beta_n \end{pmatrix}$$

When $\mu : A \rightarrow \mathscr{P}(A^*)$, we call it a **poly-morphism**, and the system is said to be an OL system. This means that each symbol can be rewritten with many possible strings, and, at any (parallel) rewriting step, one of the strings of $\mu(a)$ is chosen to replace the symbol a (for any symbol a of the rewritten string).

The strings generated by L-systems are *translated* into forms, according to the **turtle interpretation**, where symbols are associated to *segments*, and other *control* symbols are used, say $+, -, [,]$. In this case, a symbol F stands for a segment having a given initial length and angle α_0. The turtle encoding of strings is obtained by translating symbols into movements of a point (the turtle) that can draw while it moves. When an L-rewriting is applied, the segment of given length l, associated to the symbol F, is replaced by a poly-segment of sub-segments of length l/k. The fractal dimension of a rule is given by $\log_k n$ where n is the number of sub-segments of the poly-segment replacing the original segment. This means that, at each rewriting of the rule, the segment associated to symbol F is resized by the factor $1/k$. The symbols $+$ and $-$ stand for a positive or negative deviations of some prefixed angle δ. Brackets represent mechanisms of internal rewriting. In fact, when an open bracket appears, a form is generated which encodes the string between the open and closed brackets. After that, the turtle goes back to the same position and angle it had before the open bracket. In other words, we can assume that during its movement the turtle can move and draw or can only move (movement without drawing could be denoted by putting a special symbol F'). With these assumptions, a string within brackets encodes the generation of a curve such that, after its generation, the turtle moves back along this curve with the inverse movement it performed in its generation, but without drawing (all its symbols F become F'). In Figs. 2.41, 2.43, and 2.44 forms generated by L systems are expressed by means of the turtle interpretation.

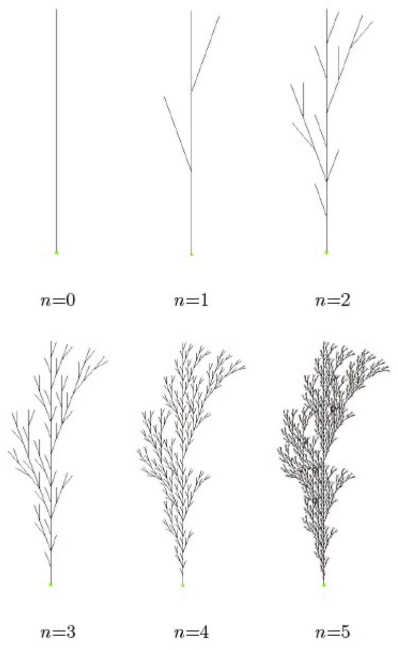


Fig. 2.41 Example of forms generated by L systems

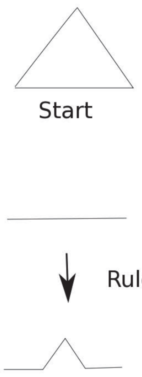


Fig. 2.42 Start string $F--F--F$ and L-rule $F \to F+F--F+F$ with $\delta = 60^o$

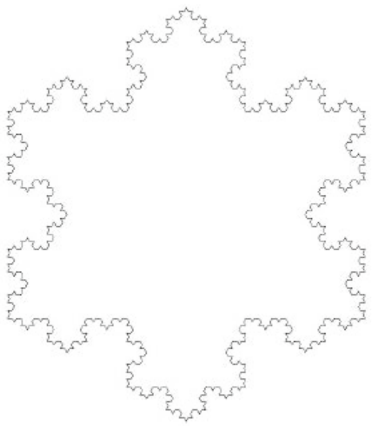
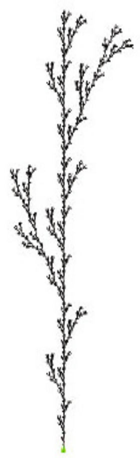


Fig. 2.43 An L form generated by the L system of Fig. 2.42

The classes EDOL and EOL are obtained by adding to the classes DOL and EOL the feature "D" of determinism and the feature "E" of extended alphabet (with nonterminal symbols).

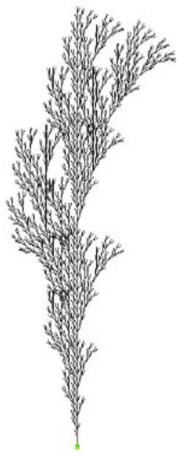
Example 2.6. The following EOL system generates the tri-somatic language (lower case symbols are terminal). In fact, the only case of producing terminal strings is when symbols A, B, C or A', B', C' are rewritten in a synchronized way. The morphism μ is given below, terminals are $\{a, b, c\}$, and abc is the initial word.

$$\begin{pmatrix} A & B & C & A' & B' & C' & a & b & c & F \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \{AA', a\} & \{BB', b\} & \{CC', c\} & \{A', a\} & \{B', b\} & \{C', c\} & F & F & F & F \end{pmatrix}$$



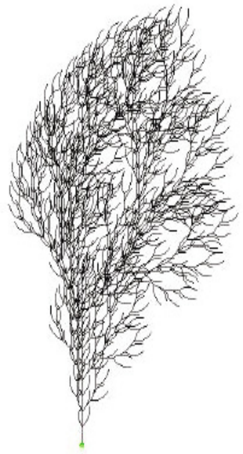
$\alpha_0 = 90, \delta = 25.7^{\circ}, n{=}5$
$\omega \ : \ F$
$p \ : \ F \rightarrow F[+F]F[-F]F$

(a)



$\alpha_0 = 90, \delta = 20^{\circ}, n{=}5$
$\omega \ : \ F$
$p \ : \ F \rightarrow F[+F]F[-F][F]$

(b)



$\alpha_0 = 90, \delta = 22.5^{\circ}, n{=}4$
$\omega \ : \ F$
$p \ : \ F \rightarrow FF-[-F+F+F]+[+F-F-F]$

(c)

Fig. 2.44 Example of other forms generated by L systems

ETOL-systems and EDTOL-systems are EOL-systems and EDOL-systems, respectively, where instead of a poly-morphism, a set of morphisms is given, which is called a *table*. In this case, at each rewriting step, one morphism of the table is chosen.

Example 2.7. The following EDTOL-system, with a table of two morphisms, generates the tri-somatic language.

$$\mu_1 = \begin{pmatrix} A & B & C & a & b & c \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ aA & bB & cC & a & b & c \end{pmatrix} \; \mu_2 = \begin{pmatrix} A & B & C & a & b & c \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ a & b & c & a & b & c \end{pmatrix}.$$

2.6.1 String Models and Theories

A string model is a relational structure (see Chap. 5) having a set of strings as domain. A powerful way of expressing rewriting rules and generation strategies is by means of logical formalisms [204].

A **string theory** or a **monoidal theory** $\mathscr{T}$, over the alphabet A, is a theory including, as terms, the *free monoid* generated by the alphabet A. This means that the signature of $\mathscr{T}$ includes the symbols of A (as individual constants), a symbol for the empty string λ, and a symbol for concatenation, which we denoted in the usual way (by juxtaposition), and its axioms include the monoid axioms (x, y, z variables of strings):

$$(xy)z = x(yz)$$
$$x\lambda = \lambda x = x.$$

In a string theory over the alphabet A terms, possibly with variables, include the symbols of A and variables ranging over strings of A^*.

A language L is a set of strings, therefore L can be defined by means of a formula; within a string theory $\varphi(x)$ (with a free variable x):

$$L = \{\alpha \mid \mathscr{T} \models \varphi(\alpha)\}.$$

In this case $\models$ is the logical consequence relation that can be computed by any logical calculus of predicate logic.

Given a grammar $G = (A, T, S, R)$, the string theory of Table 2.9, over the alphabet A, provides the logical deduction of a formula $Generate(\varphi)$ if and only if $\varphi \in \mathbb{L}(G)$ (the language generated by the grammar G).

The following example is an (equational) monoidal theory that deduces the development $R(n)$ at stage n of a Red Alga, a primitive organism that grows according to the following law.

Table 2.9 A string theory associated to a Chomsky grammar G

$Derive(S)$	
$Terminal(a)$	for every $a \in T$
$Rule(\alpha, \beta)$	for every $\alpha \to \beta \in R$
$\forall x((Terminal(x) \wedge Terminal(y)) \to Terminal(xy))$	
$\forall x, y, z((Derive(xyz) \wedge Rule(y, w)) \to Derive(xwz))$	
$\forall x((Derive(x) \wedge Terminal(x)) \to Generate(x))$	

Cells of this organism are indicated by the symbol F, cells inside brackets are "branches", inclined (alternatively in opposite verses) with respect to a main growing axis. For $n > 5$ we distinguish in $R(n)$ two parts $B(n), A(n)$ called a basal part and an apical part. Every second cell in the basal part carries a non-branching filament. These filaments develop linearly in time, they add at each stage one new cell. At stage 6 the lengths of these filaments are 3, 2, 1, respectively. The apical part at stage 6 consists of four cells without any branches. In the following stages, the apical part is a repeat of the apical part at the previous stage, together with two new cells at the end of the apical part. The second of these new cells carries a branch, identical to the whole organism six stages before.

- $R(0) = F$
- $R(1) = FF$
- $R(2) = FFFF$
- $R(3) = FF[F]FF$
- $R(4) = FF[FF]FF[F]FF$
- $R(5) = FF[FFF]FF[FF]FF[F]FFFF$
- $R(x+6) = B(x+6)A(x+6)$
- $A(5) = FF$
- $A(x+6) = FF[R(x)]A(x+5)$
- $B(x+6) = FF[L(x+3)]FF[L(x+2)]FF[L(x+1)]$
- $L(0) = \lambda$
- $L(x+1) = FL(x)$.

The following derivation is based on the equations defining Red Algae structure:

- $R(10) = B(10)A(10)$
- $A(10) = FF[R(4)]A(5) = FF[R(4)]FF =$
- $FF[FF[FF]FF[F]]FF$
- $B(10) = FF[L(7)]FF[L(6)]FFL(5) =$
- $FF[FL(6)]FF[FL(5)]FFL(5) =$
- $FF[F^7]FF[F^6]FFF^5$
- $R(10) = FF[FF[FF]FF[F]]FFFF[F^7]FF[F^6]FFF^5$.

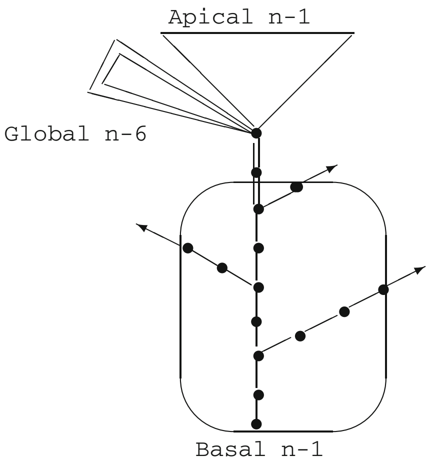


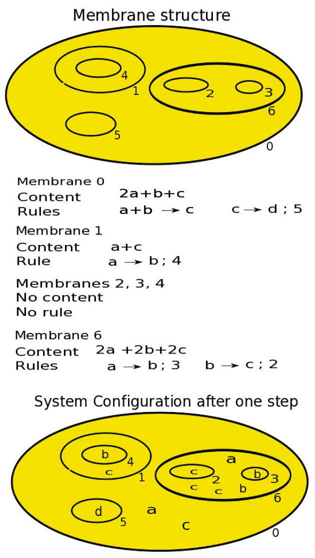
Fig. 2.45 The development of Red Alga

2.7 Membrane Computing

Membranes are one of the basic ingredients in the organization of living organisms. They determine a space partition in internal and external points, separated by a surface, called skin. Biomolecules inside a membrane are selected, protected, and concentrated. In this way, the specificity and efficiency of biochemical reactions is guaranted. Membrane systems are mathematical objects where abstract structures of membrane organization are investigated in the perspective of information processing.

Objects and membrane are dual concepts which can be reciprocally reduced (an analogous situation arises in set theory duality between elements and sets). This duality is a special case of the space/matter duality formulated in the context of a discrete framework. In fact a physical object, having a spatial extension comprises a portion of space, the space occupied by it, that can be delimited by an implicit membrane delimiting its internal region. Conversely, a membrane is an object with an internal region which can include other objects. Therefore, we may consider an object of type a as equivalent to an empty membrane $[\,]_a$. Analogously an object a inside the membrane of label j, $[a]_j$, is represented by an object a_j with the index denoting the localization of a. In general, we may reverse the relationship of containment of membranes and objects, by expressing the localization of an object by labeling it with a *membrane address* (a string of membrane labels). Here we do not enter into further details. However, many aspects deserve a careful analysis.

Membrane computing, introduced by Gheorghe Păun in 1998 [49, 50, 74, 51], develops an analysis of computations and languages in terms of multisets of objects and their localization inside compartments. The class of membrane systems he introduced are also referred as **P systems** (P is the initial letter of Păun). Differently from classical Formal Language theory, which is based on strings, in this framework multisets are the basic structure, and multiset rewriting rules are usually applied in a maximal parallel way (a set of rules which can be simultaneously applied must be applied). Rules are located into membranes and can be applied only to objects inside the membranes where they are located, even if results produced by the rule applications can be moved into other membranes. Figure 2.46 shows a membrane system and one step of computation applied to a configuration where contents are indicated for each membrane (0 is the index of the *skin*, the most external membrane). In standard membrane computing, multisets are denoted by **commutative strings** (strings where the order of symbols is not relevant), however, here for notational coherence with the other chapters, we adopt the multiset polynomial notation. We refer to the books cited above for the main results and application fields of this theory. Here, after a short definition of a membrane system, we want to list some aspects, related to the notion of membrane, that show its capability of modeling biological phenomena.



Membrane 0
Content 2a+b+c
Rules a+b → c c → d ; 5

Membrane 1
Content a+c
Rule a → b ; 4

Membranes 2, 3, 4
No content
No rule

Membrane 6
Content 2a +2b+2c
Rules a → b ; 3 b → c ; 2

Fig. 2.46 A membrane system

Membrane configurations and rules can be easily represented by bracket expressions. For example, the membrane structure given in Fig. 2.46 can be written in the following way:

$$[_0 [_1 [_4 \,]_4 \,]_1 + [_5 \,]_5 + [_6 \,[_2 \,]_2 + [_3 \,]_3 \,]_6 \,]_0$$

and the system given in Fig. 2.46 (membrane structure with initial configuration and membrane rules) is synthetically expressed in Table 2.10.

Table 2.10 The bracket representation of the membrane system of Fig. 2.46

$[_0\, 2a+b+c+\ [_1\, a+c+\ [_4\,]_4\,]_1 + [_5\,]_5 + [_6\, 2a+2b+2c+\ [_2\,]_2 +\ [_3\,]_3\,]_6]_0$
$[_0\, a+b \rightarrow [_0\, c$
$[_0\, c \rightarrow [_5\, d$
$[_1\, a \rightarrow [_4\, b$
$[_6\, a \rightarrow [_3\, b$
$[_6\, b \rightarrow [_2\, c$

Usually, membrane systems are represented by means of Venn diagrams with symbols and rules inside them. Many different notations can be used, very often with a natural interpretation. However, when a multiset m inside a membrane of index i is denoted by $[m]_i$, then we mean that m is all the content of $[\ ]_i$, while we write $[_i m$ for indicating that m is a sub-multiset of the content of $[\ ]_i$.

This notation is also called **boundary notation** and was introduced [65] in order to cope with more general membrane rules. In fact, in Păun's original formulation, rules are inside membranes and everything outside the membrane where a rule is located remains unknown to the rule. But, in many cases a wider visibility is required. The essential point of boundary representation is the idea of rules with a greater level of localization knowledge about the objects which they apply to. An important case of this situation is present in *anti-port rules* which postulate to consider objects inside and outside membranes.

The example of Fig. 2.47 is not standard in membrane computing, but it is very intuitive because it constitutes a membrane representation of a classical abacus for performing sums.

Multiset rewriting with maximal parallel strategy means that at any step a set of rules is chosen in a maximal way, because they can be simultaneously applied, and no other rule can be applied together with them (and rules have to consume as many objects as they can). This rewriting strategy provides universal computation even in the case of only one membrane. The proof of this statement is given by means of a membrane representation of register machines, in a formulation due to Minsky (which are proved to be computationally universal). They use only two types of instructions. Any instruction has a number as label (indicated on the left), and applies when this label is equal to the content of the *program counter* (a special register). Instruction $i : inc_k j$ increments the content of the register R_k and sets to j the value of the program counter. The instruction $i : condec_k j, h$ decrements the content of the register R_k, by putting j as the value of the program counter, if the content of R_k is different from zero, while it does not alter any register, and sets to h the value of the program counter, if the content of R_k is zero.

Let us represent the register R_k containing the value m with the multiset of m copies of symbol r_k, and let us represent by the presence of symbol P_i the fact

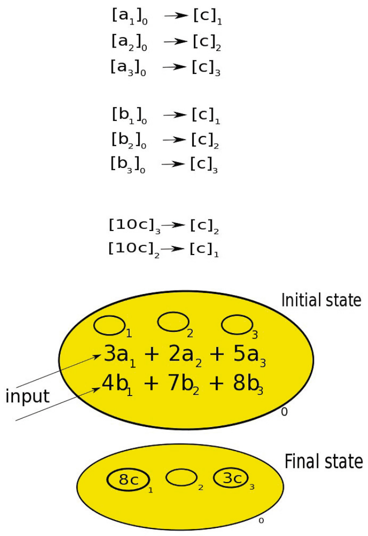


Fig. 2.47 A membrane system which realizes the sum of two decimal numbers of three digits. Input numbers are represented by multisets of three symbols (a_1, a_2, a_3 for the first input and b_1, b_2, b_3 for the second input). The output is obtained after three steps (by maximal parallel applications of rules) by the multiplicities of c in membranes $[\,]_1$, $[\,]_2$, $[\,]_3$. The case of the sum $325 + 478$ is indicated in the Venn diagram of the initial and final state of the computation.

that instruction of order i is the current instruction. With this representation of the computation states, the instructions of the register machine will be given by the following multiset rules.

1. An instruction $i : inc_k\ j$ is expressed by the rule $P_i \rightarrow r_k + P_j$.
2. An instruction $i : condec_k\ j, h$ is expressed by the rules:

$$P_i \rightarrow S_i + \#$$
$$S_i + r_k \rightarrow Q_i$$
$$\# \rightarrow \$$$
$$Q_i + \$ \rightarrow P_j$$
$$S_i + \$ \rightarrow P_h$$

It is easy to realize that, with maximal parallel rewriting, the above translation provides, in the membrane register representation, the effects of the corespondent Minski's instructions in the original registers.

Many features of biological membranes were introduced in membrane computing in order to analyze their computational relevance; moreover many different membrane systems were introduced for expressing a great variety of distributed, parallel, and non-deterministic computations [49, 50, 74, 51]. Many of them were analyzed in specific papers of membrane computing (see http://ppage.psystems.eu).

A great deal of research in membrane computing was devoted to results establishing the computational universality of specific classes of membrane systems. Many applications of membrane formalisms consist in discrete models of biological phenomena.

The following list summarizes some important aspects of membrane that are relevant to biological discrete modeling.

1. Membrane or object polarities (positive, negative, and neuter) can be considered. Therefore, rules can be applied according to the presence of certain polarities and may also transform polarities.
2. Membrane thickness can prevent the membrane permeability, that is, exit/entrance of objects from/to a membrane and rules may change a membrane thickness.
3. Rule priorities can establish an order among the rules of a membrane, and rules with a lower priority can be applied only if no rule with a higher priority can be applied.
4. Catalysts are objects that participate in a rule as reactant and products, which are not transformed during its application. They can play an important role as controllers. In fact, if a catalyst is not available the rule cannot be applied, and the amount of a catalyst determine the level of activity of a rule where it is involved.
5. Synport/antiport rules realize the entrance/exit of objects from/to membranes only in a synchronized way among pair of objects (*a* can enter/exit only if *b* enters/exits or vice versa).
6. Promoters/Inhibitors may be associated to the rules: their presence can be used for allowing or for avoiding the application of some rules.
7. Multi-membranes may be considered, that is, membranes which occur in many copies with possibly different contents. In this case when a rule is associated to this kind of membrane, it may be applied to all its copies, or only to some of them (a specific strategy has to be defined).
8. Parameters could be useful in representing physical variables which are not objects, but may influence the applicability of rules (temperature or pressure, or electrical potential).
9. Deterministic strategies could be imposed in several ways, for example, by combining priority among rules with promoters and inhibitors, or even programs could be added to membranes which at each step activate at most one rule, or groups of independent rules. The metabolic P systems, which will be introduced in the next chapter, are a special case of deterministic P systems.
10. Non-deterministic strategies can be used in a completely free manner or with some specific criteria. In the original formulation maximal parallelism was adopted; however, intermediate forms of non-determinism could be considered, where maximal parallelism could be confined to some membranes or to some kinds of rules.
11. Movement of objects could be considered only from a membrane to an immediately included or immediately including membrane.

12. Special rules may change the membrane structure, for example by removing a membrane (releasing its content in the membrane including it), or by creating membranes, or by including a membrane (with its content) inside another membrane, or expelling a membrane (with its content) from a membrane which includes it. These operations correspond to basic biological phenomena (dissolution, vesiculation, endocytosis, exocytosis). They are easily described in terms of bracket expressions, and may be extended and specialized in many ways.
13. Porters or carriers of objects could be used which control the object movement according to some transport/addressing mechanisms.
14. Many levels of membrane skins could be useful for internal localizations. For example, only objects which are at the first level could have some kind of interaction with external objects.
15. Membrane structures could assume relations which are different from membrane containment; for example, specific channels or communication lines could be assumed and rules changing these connections dynamically could be postulated.
16. Hyper-rules refer to rules which involve not only multisets, but more generally substructures. For example a rule such as $[_h c[_i a[_j b \rightarrow [_j b[_i a[_h c$ acts on three membranes and implies a complex transformation where the existence of three specific objects in the three membranes changes their containment relationship. In general, it would be very useful to introduce the notion of P-term, realized by a bracket expression possibly including variables of multisets or membrane structures. In this way, some very general operations could be represented in a very synthetic and powerful way, For example, let X,Y,Z be variables of multisets, then the following rule changes radically the inclusion structure and the content of some membranes:

$$[_i a + X + [_j b + Y]_j Z]_i \rightarrow [_i b + Y + [_j a + X]_j Z]_i$$

17. Structured objects like strings or trees could replace the simple objects of the original membrane systems, allowing specific operations over these structures in the transformations described by rules.
18. Input/output strategies establish where to put the multisets which encode the input data (according to some encoding mechanism) and where outputs of a terminating computation have to be read. However, instead of focusing on terminating computations which provide some kind of results, some external rules (outside the skin) may yield resources, by modeling the typical interactions of cells with the environment. In this case, the behavior of the system is intended to cope with the realization of some dynamical regimes of interest, keeping some variables within certain intervals, while external rules satisfy some specific constraints [65].
19. Termination strategies could be defined in many ways; for example, the presence or the absence of some symbols in some membranes could denote the end of computation, but even the absence of applicable rules can play the same role.

20. Loci (Latin word for “places”) as localization mechanisms more general than membranes, could be added. For example, if specific zones of membrane skin are defined, the position of objects could be given in terms of coordinates which refer to them. Moreover, implicit forms of localization could be based on concentration gradients along some directions.

By using some of the features mentioned in the list above it is possible to prove a great number of results of computational universality [50]. We will mention only two results of this kind:

$$NRE = NOP_1(cat_2)$$

that is, the recursively enumerable sets of natural numbers coincide with the set of numbers which can be generated by P systems with one membrane and rules allowing at most two catalysts. Moreover:

$$NRE = NOP_3(sym_1, anti_1)$$

that is, the recursively enumerable sets of natural numbers coincide with the set of numbers which can be generated by P systems with three membranes and rules allowing one *symport* rule and one *antiport* rule.

2.8 Informational Analysis of Genomes

Genomes are containers of biological information, they direct the functions of the organisms and they are what is transmitted to the generations of their organisms. During their transmission variations are introduced, which change the organisms hosting them. Therefore, according to the evolutionary mechanisms, genomes are selected indirectly through the effects they produce in the organism they direct. The selective force driving this process pushes genomes of organisms with the best capacity of spreading in the environments and of reproducing abundantly. From this point of view, we could see organisms as means for selecting genomes, instead of genomes as means for realizing organisms. The situation is a typical case of duality, in the sense that each of the two realities needs the other one. However, what it is essential to stress here, is the specific informational nature of genomes. If they are responsible for the most important part of biological information, then they have to follow the general rules which information mechanisms follow. This simple statement implies some important informational perspectives in the analysis of genomes, which we will outline in the following.

Many seminal works were developed in recent years where concepts from information theory and computer science [198, 200, 64, 22], formal language theory [213], and linguistics [16] were used in the investigation of genomes considered as particular texts ([19, 37, 52, 57, 59, 60, 24, 33, 31, 54]). These researches applied information theoretic notions (information, entropy, mutual information, encoding, compressibility, complexity measures, and randomness), or formal language theory (grammars, automata, patterns), or linguistic concepts (words, dictionaries, lexical categories, distributional classes) for recognizing and analyzing functional roles of DNA fragments.

Here we briefly outline an approach, which we call **Infogenomics** [20, 27], under development, where we apply methods of informational text analysis specifically conceived for genome analysis. Infogenomics is aimed at devising and computing informational indexes able to provide systematic "localization" of genomes in many-dimension spaces where these indexes may vary. If these indexes are defined in an adequate manner, then the genome characterization by means of them could correspond to important biological or evolutive properties which reflect their internal organization, and could provide profiles for comparing genomes of different species or even of different physiological/pathological situations.

Bioinformatics had a crucial role in the analysis of biopolymers by means of the notion of *sequence similarity* and *sequence alignment*. In this field, many algorithms on strings were essential in a huge number of important applications. The epochal sequencing of complex genomes was surely impossible without the existence of these algorithms and of their efficient implementation. However, now many genomes, of different types of organisms, are available as files in public sites. The number of sequenced genomes is near to 1000, from bacteria to Homo sapiens (without counting viruses). They are *treasures* and an integrated analysis of their informational, mathematical, and linguistic features could reveal new clues in the challenge of understanding their *languages*. This perspective requires a systemic approach where genomes are considered not only strings, but structures based on strings and the components and features of these structures could be discovered by comparing them. This emerging perspective [35, 36, 34, 62] is based on *alignment free* methods of genome analysis, where global properties of genomes are investigated, rather than local similarities based on classical methods of string alignment.

On the side of molecular biology and biochemistry many international projects are active for deciphering genomes, in order to pass from the knowledge of genome sequences to their biological functions. In particular, the project ENCODE (ENCyclopedia Of DNA Elements) [23] is mainly aimed to extract lexicons, and catalogs of biochemically annotated DNA elements, in the human genome. In this context, biochemical functions were assigned to 80% of the genome, mainly outside the protein-coding regions (with a clear evidence of their crucial role in regulation of gene expression). A very complex dynamics of interactions results among DNA regions, proteins, and RNAs, with a lot of newly identified elements, and with a huge number of data (see websites: http://nature.com/encode, http://epd.vital-it.ch). This scenario has certainly an informational basis, linked to the DNA strings related to these elements. Therefore, an integration between biochemical and informational perspective could provide important synergies, with new possibilities for interpreting data and for discovering principles of genome organizations and functions.

A simple argument can show the crucial aspect that dictionaries play in the analyses of genomes. Let us consider a genome G with a length of 10^6 bases. All the sequences of length 40 which we encounter by scanning all the genomes are $(10^6 - 39)$, moreover possibly many of them occur many times. However, the number of possible different words of four letters having length 40 is 4^{40} which is a value greater than 10^{24}. This means that the part of sequences of length 40 which occur in that genome is a portion 10^{-18} smaller than the whole set of possible words. This simple numerical evaluation tells us that surely sequences of length 40 have to be meaningful. In fact if they occur, surely they were selected among a huge number of other possible sequences. The real sequences of this length which we meet in a genome are a small part of those which were evaluated, during the evolutive process. In other words, the meaning of words is related to the relationship between the real words and all possible words.

Let us denote by Γ the **genomic alphabet** of four symbols (characters, or letters, associated to nucleotides): $\Gamma = \{A, T, C, G\}$ (then $\Gamma^\star$, as usual, denotes the set of all possible words over Γ).

A genome G is representable by a sequence over Γ, that is, a table assigning a symbol of Γ to each position (from 1 to the length of G). Symbols are written in a linear order, from left to right, according to the standard writing system of western languages, and to the chemical orientation $5' - 3'$ of DNA molecules.

We remark that other equivalent representations of sequences are possible. For example, we could represent G, by a function associating to each symbol of Γ the set of positions where it occurs. In this way G is identified by four sets of numbers, say $N(A), N(T), N(C), N(G)$. It is also important to distinguish between **subsequences** and **substrings** (also called **words**, **factors**, **k-mers**) of G. Indeed, a subsequence is a sequence of symbols occurring on a set of positions (considered in their order), while a substring is a subsequence of symbols which are (contiguously) associated to all the positions between an initial and a final position (of course, any string is also a sequence). If a genome has length n, then according to the Gauss triangular formula, it has at most $n(n-1)/2$ different factors (n factors of length 1, $n-1$ of length 2, and so on, up to only one factor of length n), while all the possible subsequences are 2^n (the different ways of choosing sets of positions).

A dictionary D of a genome G is a **factorization** of G when the concatenation of all the elements of D, possibly with overlapping sub-strings, yields G (the overlapping concatenation of $\alpha\gamma$ with $\gamma\beta$ is $\alpha\gamma\beta$). It is intended that in this concatenation the elements of D may occur at least once, but possibly more than once. We remark that the problem of **genome sequencing** can be expressed in the following way. Given a genomic dictionary D (consisting of words of G, called *reads*, usually of average lengths under 1000 bp), find the *most probable* genome G such that D is a factorization of G, and where the concatenation of elements is always a proper overlapping concatenation. Despite this simple formulation, this problem is computationally complex and its solution is not uniquely defined, in mathematical terms, but can be found, with a certain probabilistic belief (supported by the empirical evidence) by means of different and repeated reconstruction experiments of G from different factorizations of it. Nowadays, many different sequencing methods are available, which are based on different technologies. Crucial parameters of these sequencing methods are the average length of reads and the number of hierarchical phases of string assembling (where fragments of increasing lengths are reconstructed from factorizations of these fragments).

Dictionary Based Indexes

We denote by $D(G)$ the set of all factors of a genome G, while we call **k-genomic dictionary of G** (for some $k \leq |G|$), denoted by $D_k(G)$, the set of all the k-long substrings of genome G. Starting from dictionary $D_k(G)$, the **k-genomic table** $T_k(G)$, which mathematically corresponds to a *multiset*, is defined by equipping the words of $D_k(G)$ with their *multiplicities*, that is, the number of their respective occurrences in G. Let $\alpha(G)$ denote the multiplicity of α in a genome G, and $pos_G(\alpha)$ give the set of positions of α in G (that is, the positions where the first symbol of α is placed). Of course, it holds $\alpha(G) = |pos_G(\alpha)|$. The table $T_k(G)$ may be represented by a list of associations of strings to their corresponding multiplicities: $\alpha \mapsto \alpha(G)$, with $\alpha \in D_k(G)$. The sum of all the multiplicities of elements in $D_k(G)$ is called *size* of $T_k(G)$, denoted by $|T_k(G)|$. It is easy to realize that:

$$|T_k(G)| = |G| - k + 1.$$

In general, the multiset $T_k(G)$ associated to the genome G does not univocally individuate G. In fact, let us assume that G has the following string structure:

$$G = ---\gamma_1 ----xxxxx----\gamma_2----\gamma_1----yyyyy----\gamma_2---$$

Now, if we exchange the two fragments included between γ_1 and γ_2 and if their lengths are equal to, or longer than k, the resulting genome G' is such that $T_k(G) = T_k(G')$, because the k-factors of these two genomes are the same. In fact, the k strings which occur internally in $\gamma_1 ----xxxxx------ \gamma_2$ and in $\gamma_1 ----yyyyy---- \gamma_2$ do not depend on the positions of these strings, while those which are partially inside and partially outside to the (left and right) borders depend on the $k-1$-*contexts*, that is, the strings of length $k-1$ which they have on the right and on the left. But, these contexts in this case are exactly the same, because $|\gamma_1| \geq k$ and $|\gamma_2| \geq k$.

We say that two genomes G_1 and G_2 are multiset k-equivalent when $T_k(G_1) = T_k(G_2)$.

Given a dictionary D of a genome G, the **Multiplicity-Comultiplicity** distribution MC, relative to D and G, may be defined by means of a graphical profile, where in the abscissa the multiplicities are given, in increasing order (0, 1, 2, ...), and in the ordinate the number of words of D having a given multiplicity of occurrence in G is indicated.

All the typical parameters of distributions (mean value, standard deviation, median, mode, ...) also determine specific values of distribution MC.

The same information of a multiplicity-comultiplicity distribution may be expressed as a **rank-multiplicity** Zipf map (usually employed to study word frequencies in natural languages). Zipf's distributions have in the abscissa the words in decreasing order of frequency (in alphabetical order when they have the same frequency), say this order rank, and in the ordinate the value of frequency associated to a rank. Zipf's curves prove to be sensibly different for different genomes, but in all cases, there are few elements with maximal multiplicity, indeed Zipf curves initially slope down steeply.

Selectivity, Lexicality, and Forbidden Words

We call k-**lexical fraction** of a genome G the value $|D_k(G)|/4^k$, which is the percentage of different k-mers occurring in G with respect to all the possible ones. Of course, $|T_k(G)|/4^k$ is an upper bound for $|D_k(G)|/4^k$. A better evaluation for such an upper bound is given by the value $1/(1+4^k/|G|)$ which approximates $|D_k(G)|/4^k$ for a random sequence over Γ having length $|G|$. In fact, let us assume that G is random, then if q is the fraction of k-mers occurring at least once in G, then the fraction of k-mers occurring at least twice in G is q^2, and in general the fraction of k-mers occurring at least i times is q^i, therefore, assuming $q < 1$, for a very long genome G, its length can be approximated in the following way [25]:

$$|G| = 4^k(q+q^2+\ldots q^i\ldots) = 4^k\frac{q}{1-q}.$$

Therefore,

$$|G|(1-q) = 4^k q$$

that is:

$$|G| = q(|G|+4^k)$$

which implies:

$$1/q = (|G|+4^k)/|G|$$

or equivalently, the fraction of k-mers occurring in a random genome of length $|G|$ (of length sensibly shorter than 4^k) is:

$$1/q = (1+4^k/|G|). \tag{2.5}$$

The computations of $|D_k(G)|/4^k$ for the genomes of Table 2.11 are in all cases sensibly under this estimation. For example, for *H. sapiens chr. 19*, $1/(1+4^{12}/|G|)$ is equal to 0.791, while $|D_{12}|/4^{12}$ is equal to 0.639. We define for a genome G its **k-dictionary selectivity** $DS_k(G)$ as the following difference:

$$DS_k(G) = 1/(1+4^k/|G|) - |D_k(G)|/4^k. \tag{2.6}$$

Dictionary selectivity very often proves more indicative than the k-empirical entropy of $E_k(G)$, which can be defined as:

$$E(T_k(G))$$

by applying to $T_k(G)$ the following general definition of entropy $E(X)$ of a multiset X of size n with m elements of multiplicities $n_1, n_2, \ldots, n_m$:

$$E(X) = -\sum_{j=1}^{m} \frac{n_j}{n} \log \frac{n_j}{n}.$$

The k-**co-entropy** $coE_k(G)$ is a measure of the distance from the maximum value of the empirical k-entropy of a genome with the same dictionary $D(G)$:

$$coE_k(G) = \lg(|D_k(G)|) - E_k(G). \tag{2.7}$$

On the basis of genomic k-entropy, many other genomic concepts can be developed which could have a great relevance in the analysis of genomes, such as mutual information or entropic divergence.

Another related index for a genome G is given by its **k-lexicality**, that is, the ratio:

$$|D_k(G)|/|T_k(G)| \tag{2.8}$$

which expresses the percentage of distinct k-factors of G with respect to the all the k-factors present in G. It is clear that the k-lexicality increases with the word length k, and does not exhibit any regularity with the genome length. Of course, the inverse of this ratio provides an average repeatability of the k-factors of G.

When $\Gamma^k = D_k(G)$ we say that a genome G is **k-complete**, meaning that all the possible genomic k-long strings occur (at least once) in G. If G is not k-complete, a non-empty set $F_k(G)$ of "non-appearing", say **forbidden k-words** (also called "nullomers" [34]), is given by the difference of sets:

$$F_k(G) = \Gamma^k \backslash D_k(G). \tag{2.9}$$

Of course, genomic k-completeness is related to the genome length. In fact, it is easy to find a genome length such that surely genomes of that length are k-complete. In fact we can construct such a genome by concatenating, in any order, all the k-mers. Therefore, we have $4^k!$ genomes k-complete of length $k4^k$. The search for the minimum length providing genomic k-completeness, and of algorithms for constructing such minimal genomes, is a non-trivial theoretical investigation of some possible interest.

For each G, we can define its **minimal forbidden length**, denoted by $MF(G)$, as the minimum k such that G is not k-complete.

The cardinality of $F_k(G)$, for k greater than $MF(G)$ and within a small range over $MF(G)$, seems to be a very specific feature of each genome. It is indeed remarkable that in all genomes we considered $MF(G)$ is at least 6 and below 12 (see Table 2.15), and it does not appear directly related to the biological complexity of corresponding organisms. Another interesting character of genomes is the **factor length selectivity** $LS(G)$, which expresses the gap between the length of factors which in principle could be all accommodated in a genome G, and the length of those which are actually present in G (according to the value of its minimal forbidden length):

$$LS(G) = \lfloor \lg_4 |G| \rfloor - (MF(G) - 1) \tag{2.10}$$

where $\lfloor x \rfloor$ is the floor (greatest integer less than the real value) of x. The value of LS(G) is around 5 in all the unicellular and primitive multicellular organisms, and is around 10 in the two human chromosomes we analyzed. A clear understanding of this behavior should be investigated in more general terms; however, $LS(G)$ is surely related to an evolution selectivity action over the strings constituting genomic dictionaries.

Hapax and Repeat Analysis

Two important types of factors of genomes are hapaxes and repeats. A **hapax** of a genome G is a factor α of G such that $\alpha(G) = 1$. The term hapax (from a Greek word meaning once) came from the analysis of literary texts (for stylistic analysis and text authorship attribution); however, now it is also used in informational text analysis [64].

A **repeat** of G is a factor α of G such that $\alpha(G) > 1$. Of course, the set $H(G)$ of hapaxes of G and the set $R(G)$ of repeats of G constitute a bipartition of $D(G)$ (at least one element of Γ is a repeat and G is a hapax, therefore $H(G)$ and $R(G)$ are non-empty, also disjoint sets, such that their union is $D(G)$). We set:

$$H_k(G) = \Gamma^k \cap H(G) \tag{2.11}$$

$$R_k(G) = \Gamma^k \cap R(G) \tag{2.12}$$

where $\cap$ is the set-theoretic intersection. Therefore, given a genome G of length n, for any $k \leq n$ we can read it according to the bi-partition of its k-genomic dictionaries $H_k(G)$ and $R_k(G)$.

A more refined measure for the **average k-factors repeatability** in G may be now given as:

$$AR_k(G) = \frac{|T_k(G) \setminus H_k(G)|}{|R_k(G)|} \tag{2.13}$$

where k-hapaxes have been excluded by both the k-genomic multiset and the k-genomic dictionary (the symbol\represents the set-theoretic difference). Index $AR_k(G)$ counts the proper (average) repeatability of k-repeats in genome G.

The concepts of hapax and repeat provide a great number of related notions. In the following section we will discuss experimental data, reported in tables, diagrams, and figures, which include the measure of the ratio between $|H_k(G)|$ and $|R_k(G)|$ as a function of k (that is, how the number of hapax words of a given length increases or decreases with respect to the number of repeats of that length). An important phenomenon guided us in the choice of the string lengths for the computed dictionaries. In fact, we observed a sort of *transition phase* effect in the passage from $D_{12}(G)$ to $D_{18}(G)$, in almost all genomes of Table 2.11, where a clear inversion appears in the ratio hapax-cardinality/repeat-cardinality.

Let us mention briefly other relevant indexes, related to hapax and repeat concepts, that will be reconsidered in the following (for definitions see Table 2.13): **minimal hapax length**, denoted by MH, **maximal repeat length** MR, **repeat**

positions RP. An important concept is the **maximal k-repeat distance**, defined as (symbols $\perp$ and $\top$ denote respectively minimum and maximum over sets of numbers):

$$MD_k(G) = \top\{|j-i| : \alpha \in R_k(G), i \in pos_G(\alpha),\ j = \perp\{k : k \in pos_G(\alpha), k > i\}\}.$$

In Table 2.11, some genomes are reported as case studies of infogenomics analyses. In Table 2.12, sizes of some genomic dictionaries are reported. In Table 2.13 some infogenomics indexes were collected, while in Tables 2.14, 2.16, 2.17 numerical values of these indexes are reported which were computed for genomes of Table 2.11 [20]. Sequences were downloaded as *fasta* files from *NCBI genome database*, UCSC Genome Bioinformatics website, EMBL-EBI website, or KEGG Kyoto Encyclopedia of Genes and Genomes website. They constitute biological models of remarkable relevance in the genomic analyses. A most detailed description of these genomes may be found in: `http://users.rcn.com/jkimball.ma.-ultranet/BiologyPages/G/GenomeSizes.html`.

Table 2.11 A list of genomes

Organism genome	Length	Genes	Type
Nanoarchaeum equitans	490,885	536	Minimal archaeum
Mycoplasma genitalium	580,076	476	Minimal bacterium
Mycoplasma mycoides	1,211,703	1,016	Venter's experiment bacterium
Haemophilus influenzae	1,830,138	1,717	First sequenced bacterium
Escherichia coli	4,639,675	4,685	Bacterium model (K-12)
Pseudomonas aeruginosa	6,264,404	5,566	Ubiquitous bacterium
Saccharomyces cerevisiae	12,070,898	6,275	Unicellular eukaryote (Yeast)
Sorangium cellulosum	13,033,779	9,700	Longest genome bacterium
Homo sapiens chr. 19	63,800,000	2,066	Highest gene density H. chromosome
Caenorhabditis elegans	100,267,632	19,000	Worm (around 1000 cells)
Drosophila melanogaster	129,663,327	14,000	Insect (Fruit fly)
Homo sapiens chr. 1	247,000,000	3,511	Longest Human chromosome

For all our genomes of Table 2.11, listed according to the increasing genome length, we report in Tables 2.14, 2.16, and 2.17 numerical data related to the computation of $D_k(G), F_k(G), H_k(G), R_k(G)$ for k= 6, 12, and 18, respectively.

It is easy to see that any genomic factor containing a hapax as a substring is a hapax as well. Hence a hapax within the genome may be elongated (by keeping its property of being a hapax) up to reach the genome itself, which is of course a hapax. It is then interesting to evaluate, for each genome G: *i)* how $|H_k(G)|$ varies with k, *ii)* the k-hapax positions (that is, how densely hapax words fall in the genetic regions), and *iii)* the shortest length of a hapax. Also, a k-similarity between genomes G and G' could be measured by $|H_k(G) \cap H_k(G')|$ (we have some work in progress on the computation of dictionary intersections).

Table 2.12 Sizes of genomic dictionaries

$\lvert\Gamma^6\rvert = 4^6$	$\lvert\Gamma^{12}\rvert = 4^{12}$	$\lvert\Gamma^{18}\rvert = 4^{18}$	$\lvert\Gamma^{24}\rvert = 4^{24}$
4096	$16,777,216$	68.719476736×10^9	$> 281 \times 10^{12}$

Organism genome	$\lvert D_6\rvert/4^6$	$\lvert D_{12}\rvert/4^{12}$	$\lvert D_{18}\rvert/4^{18}$
Nanoarchaeum equitans	0.99	0.025	$\approx 7 \times 10^{-9}$
Mycoplasma genitalium	0.99	0.029	$\approx 8 \times 10^{-9}$
Mycoplasma mycoides	0.99	0.038	$\approx 14 \times 10^{-9}$
Haemophilus influenzae	1	0.089	$\approx 26 \times 10^{-9}$
Escherichia coli	1	0.207	$\approx 67 \times 10^{-9}$
Pseudomonas aeruginosa	1	0.175	$\approx 90 \times 10^{-9}$
Saccharomyces cerevisiae	1	0.393	$\approx 169 \times 10^{-9}$
Sorangium cellulosum	1	0.23	$\approx 185 \times 10^{-9}$
H. sapiens chr. 19	1	0.639	$\approx 610 \times 10^{-9}$
Caenorhabditis elegans	1	0.83	$\approx 1{,}315 \times 10^{-9}$
Drosophila melanogaster	1	0.947	$\approx 1{,}712 \times 10^{-9}$

Table 2.13 Genomic indexes, dictionaries, and tables

Indexes/Dictionaries/Tables	Notation	Definition
Genomic Dictionary	D(G)	$\{\alpha \in \Gamma^* \,:\, \alpha \subset G\}$
k-Genomic Dictionary	$D_k(G)$	$\Gamma^k \cap D(G)$
k-Genomic Table	$T_k(G)$	$\alpha \mapsto \alpha(G) \,:\, \alpha \in D_k(G)$
k-Lexicality	$L_k(G)$	$\lvert D_k(G)\rvert/\lvert T_k(G)\rvert$
k-Dictionary Selectivity	$DS_k(G)$	$1/(1+4^k/\lvert G\rvert) - \lvert D_k(G)\rvert/4^k$
Multiplicity-coMultiplicity *k*-distribution	$MC_k(G)$	$j \mapsto \lvert\{\alpha \in D_k(G) : \alpha(G) = j\}\rvert$
Forbidden *k*-Factors	$F_k(G)$	$\Gamma^k \setminus D_k(G)$
Minimal Forbidden Length	$MF(G)$	$\perp\{k : \Gamma^k \not\subseteq D_k(G)\}$
Factor Length Selectivity	$LS(G)$	$\lfloor \lg_4 \lvert G\rvert \rfloor - (MF(G)+1)$
Hapaxes	$H(G)$	$\{\alpha \in D(G) : \alpha(G) = 1\}$
k-Hapaxes	$H_k(G)$	$\Gamma^k \cap H(G)$
k-Hapax-factor ratio	$HD_k(G)$	$\lvert H_k(G)\rvert/\lvert D_k(G)\rvert$
Minimal Hapax Length	$MH(G)$	$\perp\{\lvert\alpha\rvert \,:\, \alpha \in H(G)\}$
Repeats	$R(G)$	$\{\alpha \in D(G) : \alpha(G) > 1\}$
k-Repeats	$R_k(G)$	$\Gamma^k \cap R(G)$
Maximal Repeat Length	MR(G)	$\top\{\lvert\alpha\rvert : \alpha \in R(G)\}$
Repeat Positions	$RP(G)$	$\alpha \mapsto pos_G(\alpha) : \alpha \in R(G)$
Length-Multiplicity Repeatability	$LM(G)$	$j \mapsto \alpha(G) : \lvert\alpha\rvert = j \,,\, \alpha \in R(G)$
Average *k*-Repeatability	$AR_k(G)$	$\lvert T_k(G) \setminus H_k(G)\rvert/\lvert R_k(G)\rvert$
k-Repeat-factor ratio	$RD_k(G)$	$\lvert R_k(G)\rvert/\lvert D_k(G)\rvert$

Table 2.14 Indexes related to D_6 dictionaries

Genomic sequences	$\|D_6\|$	L_6	$\|H_6\|$	$\|R_6\|$	HR_6
Nanoarchaeum equitans	4,094	0.008	6	4,086	1.468×10^{-3}
Mycoplasma genitalium	4,082	0.007	35	4,047	8.65×10^{-3}
Mycoplasma mycoides	4,076	0.003	39	4,037	9.661×10^{-3}
Haemophilus influenzae	4,096	0.002	0	4,096	0
Escherichia coli	0	0.0009	0	4,096	0

Table 2.15 Genomic forbidden lengths and words

Genomic sequences	MF	$\|F_6\|$	$\|F_7\|$	$\|F_8\|$	$\|F_9\|$	$\|F_{10}\|$	$\|F_{11}\|$	$\|F_{12}\|$
Nanoarchaeum equitans	6	2	552	13,294	130,705	802,658	3,837,340	16,346,170
Mycoplasma genitalium	6	14	851	14,196	126,707	779,814	3,789,892	16,281,022
Mycoplasma mycoides	6	20	1,269	18,498	141,646	791,717	3,747,529	16,130,251
Haemophilus influenzae	7	0	12	1,077	33,891	442,572	3,083,682	15,281,515
Escherichia coli	7	0	1	176	5,617	150,468	1,997,469	13,298,293
Pseudomonas aeruginosa	8	0	0	276	21,387	326,432	2,536,746	13,827,364
Saccharomyces cerevisiae	9	0	0	0	99	31,619	1,007,904	10,179,957
Sorangium cellulosum	7	0	1	683	23,327	299,079	2,296,098	12,913,817
Homo sapiens chr. 19	9	0	0	0	53	12,763	469,379	6,041,533
C. elegans	10	0	0	0	0	38	54,011	2,847,301
D. melanogaster	11	0	0	0	0	0	3214	886,004
Homo sapiens chr. 1	9	0	0	0	1	2,698	149,365	2,830,885

Table 2.16 Indexes related to D_{12} dictionaries

Genomic sequences	$\|D_{12}\|$	L_{12}	$\|H_{12}\|$	$\|R_{12}\|$	RD_{12}	HR_{12}	AR_{12}
Nanoarchaeum equitans	431,046	0.87	385,146	45,900	0.11	8.39	2.30
Mycoplasma genitalium	496,194	0.85	435,502	60,692	0.13	7.175	2.38
Mycoplasma mycoides	646,965	0.53	442,836	204,129	0.32	2.169	3.76
Haemophilus influenzae	1,495,701	0.81	1,256,043	239,658	0.17	5.240	2.39
Escherichia coli	3,478,923	0.74	2,675,846	803,077	0.24	3.331	2.44
Pseudomonas aeruginosa	2,949,852	0.47	1,799, 637	1,150,215	0.39	1.564	3.88
Saccharomyces cerevisiae	6,597,259	0.54	3,977,392	2,619,867	0.40	1.518	3.08
Sorangium cellulosum	3,863,399	0.29	1,924,969	1,938,430	0.51	0.993	5.73
Homo sapiens chr19	10,735,683	0.19	3,359,705	7,375,978	0.69	0.455	6.99
C. elegans	13,929,915	0.13	3,099,744	10,830,171	0.78	0.286	8.97
D. melanogaster	15,891,212	0.12	1,632,045	14,259,167	0.9	0.114	8.89

Table 2.17 Indexes related to D_{18} dictionaries

Genomic sequences	$\|D_{18}\|$	L_{18}	$\|H_{18}\|$	$\|R_{18}\|$	RD_{18}	HR_{18}	AR_{18}
Nanoarchaeum equitans	489,465	0.99	488,802	663	0.001	737.25	3.11
Mycoplasma genitalium	569,202	0.98	563,045	6,157	0.01	91.44	2.76
Mycoplasma mycoides	987,645	0.81	913,599	74,046	0.07	12.33	4.025
Haemophilus influenzae	1,795,492	0.98	1,775,531	19,964	0.01	88.93	2.64
Escherichia coli	4,557,590	0.98	4,518,585	39,005	0.008	115.84	3.10
Pseudomonas aeruginosa	6,183,215	0.98	6,117,968	65,247	0.01	93.76	2.24
Saccharomyces cerevisiae	11,499,795	0.95	11,307,098	192,697	0.01	58.67	3.96
Sorangium cellulosum	12,640,960	0.96	12,340,846	300,114	0.02	41.12	2.30
Homo sapiens chr19	41,529,106	0.75	39,256,297	2,272,809	0.05	17.27	6.91
C. elegans	89,444,661	0.89	85,157,627	4,287,034	0.04	19.86	3.52
D. melanogaster	116,446,627	0.90	112,977,046	3,469,581	0.02	32.56	4.45

Table 2.18 MR index, positions of the only twice repeating word of length MR, and relative distance between the two occurrences (with respect to the genome lengths)

Genomic sequences	MR	$MD_{MR}/\|G\|$
Nanoarchaeum equitans	139	96.95%
Mycoplasma genitalium	243	0.15 %
Mycoplasma mycoides	10,963	0.019 %
Haemophilus influenzae	5,563	8.05%
Escherichia coli	2,815	0.89 %
Pseudomonas aeruginosa	5,304	12.37 %
Saccharomyces cerevisiae	8,375	0.07%
Sorangium cellulosum	2,720	27.68 %
Homo sapiens chr19	2,247	0.02%
C. elegans	38,987	0.10 %
D. melanogaster	30,892	0.02 %

The phenomenon regarding hapax statistical distribution may be observed passing from 12- to 18-genomic dictionaries (see Tables 2.14, 2.16, and 2.17). For all the genomes, by enlarging the k value, the number of hapax increases, even relatively to the number of repeats (roughly speaking, "most of the 12-words are repeats while most of 18-words are hapax"). Indeed, by computing HR_k, we see that repeatability generally almost disappears for $k = 18$, with respect to the number of hapaxes.

More interestingly, the (relative) amount of hapaxes increases by some order of magnitude with k passing from 12 to 18. Based on this observation coming from computational analyses, one could suppose that by increasing the word size, genomic dictionaries composed by only hapaxes may be computed. This intuition has been invalidated (see Table 2.18). In fact, repeats having lengths of several thousands have been found within each of our genomes, and $12 \rightarrow 18$ represents a sort of transition phase from scarce to abundant hapax/repeat distribution. This phenomenon would surely deserve a more detailed and generalized analysis.

Any substring of a repeat word is still a repeat, with its own multiplicity along the genome, and inside the repeat word itself. A further index is thus defined over genomes G, called $MR(G)$ (**maximal repeat length**), as the maximal length of words γ such that $\gamma(G) > 1$. An algorithmic way to find it (for our genomes) starts from repeats out of $D_{18}(G)$ (that are less than the hapaxes) and checks how much they may be elongated on the genome by keeping their status of repeat words. Data related to the MR index computed over our genomes are reported in Table 2.18, where the only MR-long repeat of each genome exhibits a non-trivial structure (that is, different than polymers with a same nucleotide or similar patterns), and complex repeats are obtained for many lengths.

The importance of word repeatability is crucial to understanding the information content of texts. A genome analysis in terms of (shortest) hapaxes and (maximal) repeats, providing their relative distribution within the genome, highlights the associative nature of DNA as a container of information. Localization and frequency of specific DNA fragments is indeed crucial to understand the information organization of genomes. Hapaxes, occurring once in the genome, by their nature have a role of address for the specific retrieval of functional elements, characterized by redundancy and repeatability. On the other hand, an important characterization of repeats may be given by means of their internal structure, that is, by the non-maximal repeats which compose them. These represent a second level repeatability, possibly exhibiting various and rich genomic structural properties of functional sequences (such as the presence of power strings).

Indexes, dictionaries, and tables given in this section identify a kernel of about 20 basic concepts, and many other notions may be derived from them. Namely, for any numerical index I_k with parameter k, the distribution $k \mapsto I_k$ can be defined, and its classical statistical parameters (mean, standard deviation, median, mode, etc.) may be derived as further indexes (the same possibility holds for multiplicity-comultiplicity factor distribution). Moreover, extending Shannon's notion of typical sequence in information theory, for any index I, a minimal I-typical sequence, for a given genome G, is a portion of G such that the index I, restricted to this portion, assumes (approximates) the same value which I assumes over the whole G. The length and number of these sequences are other genomic indexes. The power of some indexes in characterizing properties, which are relevant in specific contexts, is a kind of research requiring computational experiments, mathematical analyses, and biological interpretations and comparisons.

Bipartition of a genomic dictionary in hapax and repeat words emphasizes the roots of precise string categories which are related to the functional organization of genomes. The set of 18-repeats in our genomes has a size which is a couple of orders smaller than the whole genome, and it seems to have a role of "lexical" coding. Other elements, with a notably bigger size, seem to have a role of addressing, delimiting, coordinating, just like position-identification tags. While the lexical nature of repeated elements points out their semantic value, the "relative localization" nature of the others gives importance to their unrepeatability along the genomic sequence.

This phenomenon explains, at an informational level, what is apparent in complex genomes, where it is well-known that the DNA "coding portion" is much smaller than the rest. Understanding the logic of such a reality is a crucial challenge in genome analyses. We would like to remark here that such a dichotomy corresponds to statistical-informational evidence. On the basis of the hapax/repeat distinction, we started a genome analysis for building synthetic gene networks, called **repeat sharing gene networks** where genes (of a given genome) are the nodes, and two genes are connected by a labeled edge if they share a repeat, which is put as label of the edge. This procedure has so far been applied to small genomes, but many important phenomena were individuated and in some cases groups of genes strongly connected (often as complete subgraphs) correspond to genes which are related to a specific cell function.

Figures 2.48, 2.49, 2.50, 2.51, 2.52, and 2.53 are relative to gene networks obtained by connecting genes having some common words (repeats) in N. equitans and E. coli (for different lengths of words).

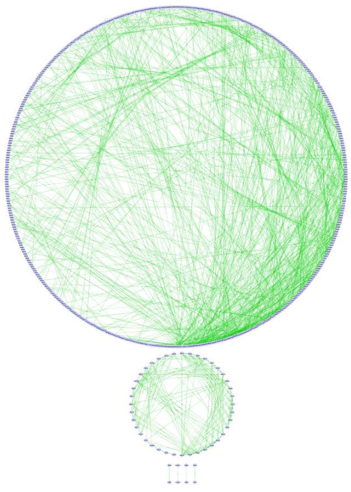


Fig. 2.48 The repeat-sharing whole network of Nanoarchaeum equitans (repeat length 16)

The definition, computation, and analysis of well-characterized dictionary-based genomic indexes have pointed out some phenomena of genomic regularity and specificity. They can enhance our knowledge about the internal logic of genome structure and organization, as well as about evolutional and functional attributes of genomes, specifically devoted to genome clustering. We trust this methodology of comparative genomics, after a suitable and specific tuning of our indexes, may be a basis for discrimination of genomic pathologies.

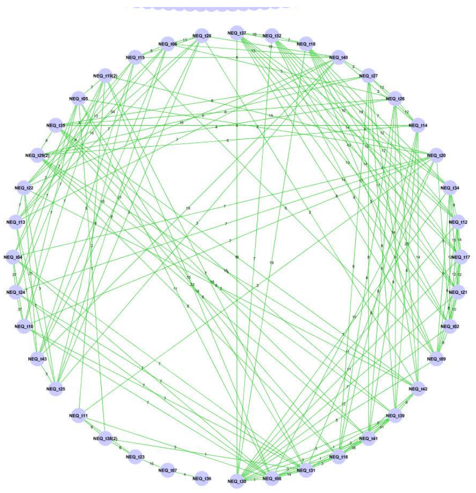


Fig. 2.49 A portion of the N. equitans network of Fig. 2.48

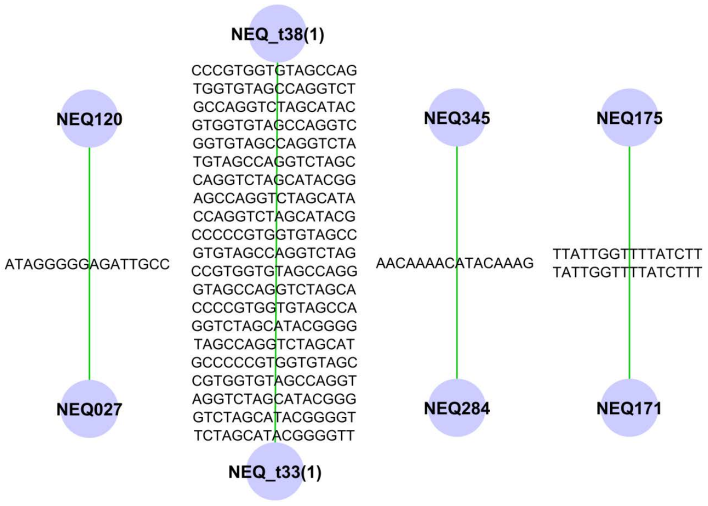


Fig. 2.50 A portion of the N. equitans network of Fig. 2.48 with the common repeats as labels of edges

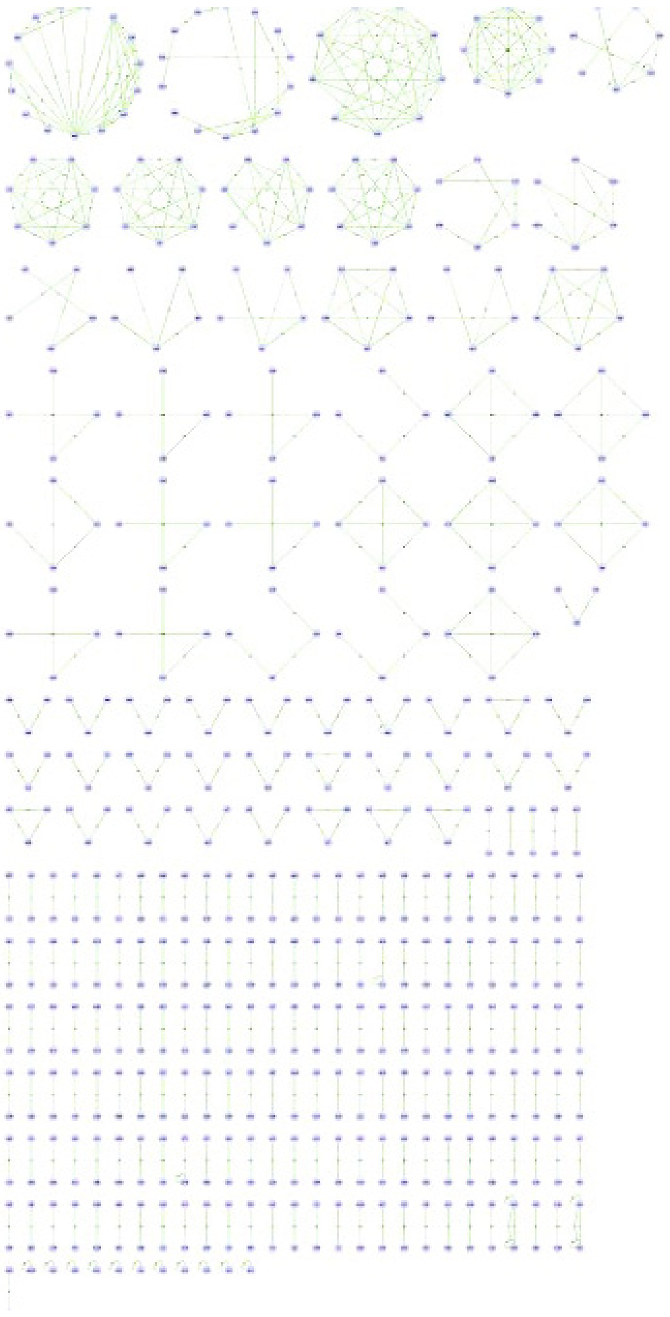


Fig. 2.51 The repeat-sharing whole network of Escherichia coli (repeat length 20)

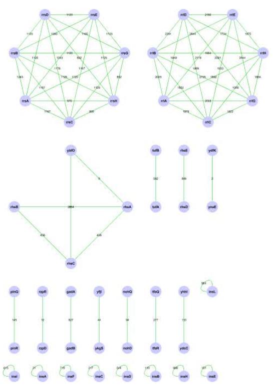


Fig. 2.52 The repeat-sharing whole network of Escherichia coli (repeat length 150)

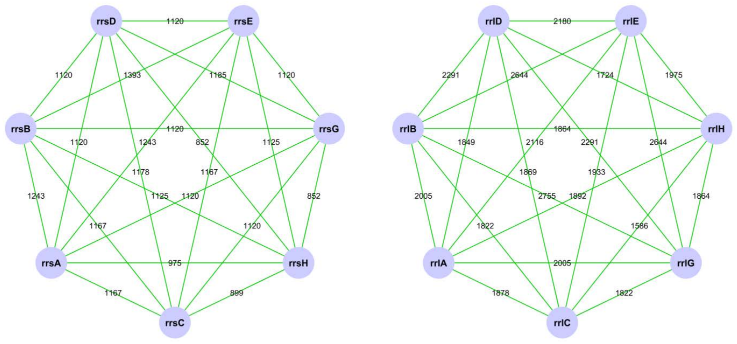


Fig. 2.53 A portion of the E. coli network of Fig. 2.52

There are several possible lines of development:

1. A systematic analysis of other genome sequences (for example of all human chromosomes) should give more arguments and hints for biological evaluations of the genomic indexes;
2. The systematic analysis of repeat sharing networks, based on specific dictionaries, could reveal/confirm important functional gene organization explaining the semantic roles of repeats;
3. Approaches based on genomic dictionaries could be exported to other kinds of genomic data, for example those obtained by means of RNA-Seq methodology;
4. Genomic dictionaries of k-mers for $k > 12$ are very often computationally very hard to be computed. Therefore, specific algorithms and data representations for "long" k-mers could be developed;

5. It would be interesting to compute intersections of genomic dictionaries and to investigate if words which are common to many genomes are conserved along evolutive lineages.
6. The inter-genomic character of hapaxes and repeats could be investigated, for determining which hapaxes (resp repeats) of a given genome keep or not their status of hapax (resp. repeat) within a given class of genomes.
7. New kinds of genomic representations could be investigated which could be useful for specific analyses of genomic features relevant in specific contexts.

The last point of the list above is the basis for possible important developments. Representing genomes, in non-conventional manners, could open new possibilities in genome analysis. In fact, within a certain genomic representation some aspects could emerge which are not evident in other conventional ways of expressing genomes. An interesting issue, in this context, concerns genome compression methods. A genome compression provides a compact way of identifying genome sequences, without loss of information. Many approaches have been investigated, where classical compression methods of information theory were applied and adapted to genomes [31]. In general, genomes are sequences not efficiently compressible, with standard methods.

Now, let us assume we find an encoding that, within some class of genomes, is able to provide efficient compressions, for example, by reducing a genome G of n bases to a sequence g of $n/10$ bits (an incredible compression ratio, with respect to the actual genomic compressions which rarely are below 1.7 bits per base, in the average case). Even if this encoding were hardly reversible, if injective, no loss of information would occur in the passage from G to g. We call this kind of encoding an *one-way compression*, because it reduces the digital information of G, but the recovering of G from g is not efficient, or even computationally hard, in the sense of computational complexity (for example NP-hard). Nevertheless, g could be very important for representing the global identity of G in comparisons with other genomes and, for its reduced size, more adequate for some specific genomic analyses.

Another kind of investigations concerns the lossy compression methods. Let us suppose that a genome G is represented by a shorter string g that does not identify G, but g retains some specific feature of G that is more evident, or more easily detectable (for the reduced size of g). If this is the case, the compression method adopted for generating g could provide an important clue, in the context particular situations of genome classification.

It is too early for evaluating the biological interest of these approaches, but it is reasonable to believe that new methods of genome representation could disclose relevant aspects in the informational organization of genomes.

Examples of graphics, related to genomic distributions defined above, are given in Figs 2.54 – 2.58.

Genomes cannot be fully considered as simple sequences, but more properly, texts in the usual sense of our written texts of natural or artificial languages. In fact, what we call a text is very often apparently a linear structure, because information is organized at several levels that we distinguish by means of different kinds of information which are added to the basic linear structure: different kinds of alphabets

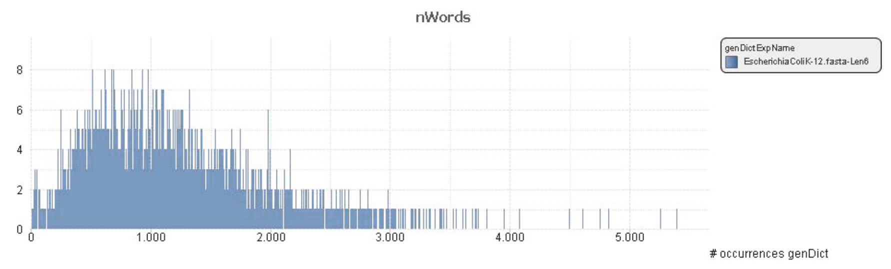


Fig. 2.54 Multiplicity-comultiplicity distribution of 6-words of Escherichia coli

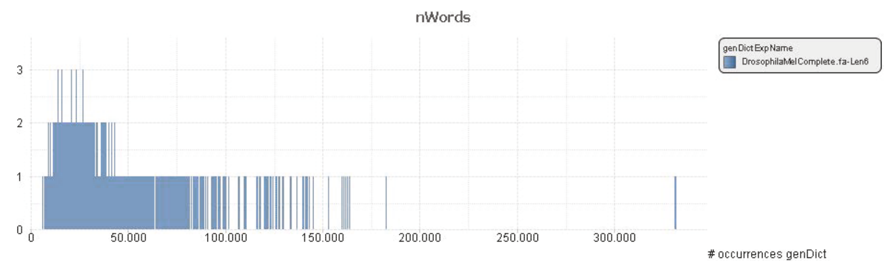


Fig. 2.55 Multiplicity-comultiplicity distribution of 6-words of Drosophila

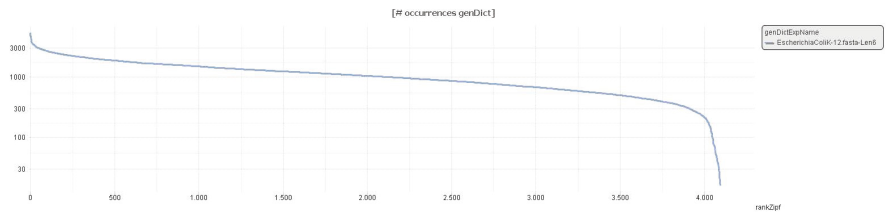


Fig. 2.56 Rank-frequency distribution of 6-words of Escherichia coli (ordinate in logarithm scale)

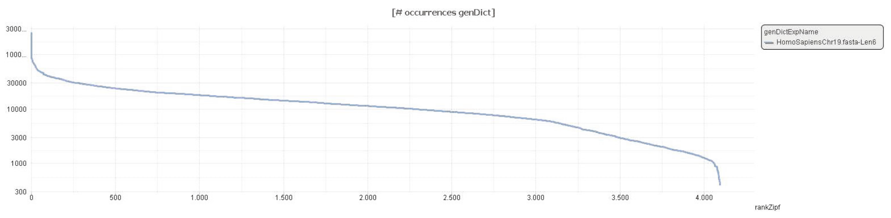


Fig. 2.57 Rank-frequency distribution of 6-words of Chromosome 19 of Homo sapiens (ordinate in logarithm scale)

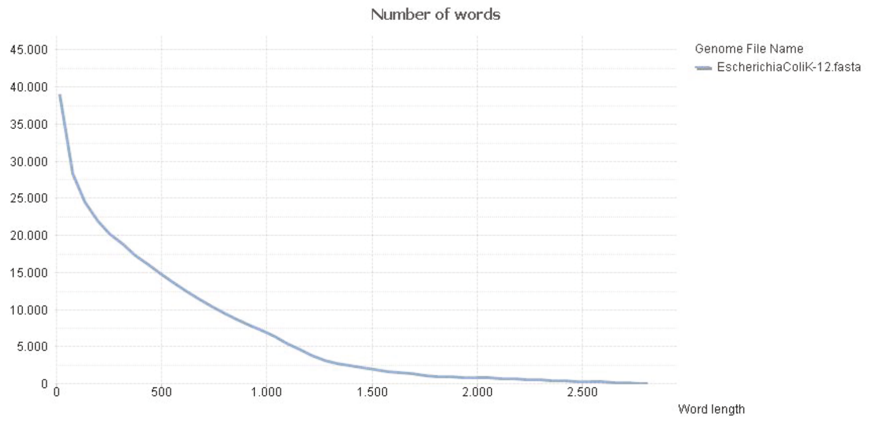


Fig. 2.58 Multiplicity-length distribution of repeats in Escherichia coli

and styles, character dimensions, and structural information (paragraphs, sections, titles, footnotes, ...). When such a rich structure of text is represented in a pure textual format, for example in terms of ascii code, then the whole information is encoded in suitable manners by means of a complex process of representation. For the sake of clarifying the idea, we can refer to modern languages for text representation such as Tex, Latex, or XML (see [41, 42] for examples of XML representation of linguistic structures). However, the encoding process is very strict. If we delete from a Latex file only a few characters, then it becomes useless and its compilation providing the text it represents cannot be successfully realized. Genomes need to realize a complex informational structure, but they are pure linear structures which require a high level of *flexibility*. DNA is a sort of associative memory, no addresses are available, as in electronic memories, therefore hapaxes can play an essential role in localizing and coordinating pieces containing specific meanings. In this perspective, the semantics of words do not depend on their absolute positions, but on their relative positions in a system of anchors or markers for correctly determining some elements within the genomic superstring where they occur.

Tables 2.19, 2.20, 2.21, 2.22, and 2.23 are examples of complex texts, such as a web page or a mathematical formula. They are represented at different levels. The information, at any level, is the same, but the "comprehension" at different levels is very different. In the first case the meaning underlying in Tables 2.19 and 2.20 is a famous web page, which everybody uses frequently as Web search engine. Its meaning is very clear and any component of the page can be easily recognized with its specific function. However, it is hard to recognize this meaning in the representations given in the mentioned tables. The second example is an elementary algebraic formula (for solving quadratic equations). Its meaning is clear to anybody having basic mathematical skills. However, also in this case, it is difficult to recognize its meaning from Tables 2.22 and 2.23. These examples intend to stress the importance of discovering the structure of a text from its basic linear representation where the organization levels are put in a common unidimensional structure. This is surely the case of genomic texts; therefore the examples help to imagine the kinds of difficulties faced in disclosing the internal logic of genomes.

Figures 2.59, 2.60, 2.61, 2.62, 2.63, and 2.64 are visual representations of the genome of E. coli. The genome of this organism (a strain different from the more common K12) is 5,528,445 base pairs. Colors denote bases according to the following code: A green, C blue, G yellow, and T red. The size of edges (or sectors, or circles) is proportional to the multiplicity of paths passing through them. Moreover, in all these visualization we can focus on different parts of the structure, by enlarging the view of some portions (Figs. 2.60, 2.62, 2.64 focus on parts of Figs. 2.60, 2.62, 2.64, respectively). The trees represented in these figures (with different visualization methods, see [40]) are obtained by means of an algorithm, called "Crescendo", which is based on the elongation of initial *seeds* (in our case fragments of length 6), which are chosen among the most frequent 6-mer words occurring in the genome, plus the initial 6-mer word, or overlapping concatenations of these words occurring in the genome. In the cases shown in the figures they are (multiplicities are between brackets):

$$AGCTTTCTGGCG[1], CTGGCG[6032],$$
$$CTGGCGCTGGCG[22], CTGGCGCTGGCGCTGGCG[1], AGCTTT[1280].$$

These words were elongated by starting from each of them and adding, in the order, all the bases that in the genome follow the seed, by stopping the elongation when the beginning of any seed is encountered. In this way the concatenation of these elongations covers the whole genome. The different elongation fragments are 7208 and occur 7336 times, therefore the majority of them are hapaxes. For a better visualization, the trees represented in the figures end when the number of possible continuations are less than 30. These figures suggest many possible analyses, but what we want to remark here is that they suggest perspectives of genome analyses that probably are hidden when we represent genomes in classical manners.

Genome visualization is not only a matter of presentation. In fact, when colors, forms, and geometrical shapes are used in visualizations, we use many-dimensional perspectives, which prove to be more adequate for expressing systemic phenomena. As shown in Tables 2.19, 2.20, 2.21, 2.22, and 2.23 the internal organization of texts is hardly recognized in their sequential representations, because it is very often based on many representation levels, and this aspect is what makes a text different from a string (see tables 2.21, 2.22, 2.23). An interesting phenomenon, common to the genome representations based on Crescendo, is their *noise amplification*. Namely, if a genome is changed, in a small percentage, for example, by randomly substituting some bases, then the corresponding (Crescendo) representations prove to be different (with respect to reasonable metrics) in a percentage that is between 3 and 10 times the original variation percentage. This effect was confirmed by many experiments, and can be explained by the additional structure of the representation that becomes more sensible to small variations in the genome linear structure.

Table 2.19 First 1000 among 98.870 HTML characters of Google home page (spaces included)

```
<!doctype html><html itemscope="itemscope" itemtype="http://schema.or
g/WebPage"><head><meta itemprop="image" content="/images/google_favi
con_128.eps"><title>Google</title><script>window.google={kEI:"cHB1UO7
WKcTGswa754HgBA",getEI:function(a){var b;while(a&&!(a.getAttribute&&(
b=a.getAttribute("eid"))))a=a.parentNode;return b||google.kEI},https:
function(){return window.location.protocol=="https:"},kEXPI:"31215,357
02,37102,38371,39523,39977,3300025,3300119,3300124,3300133,3300135,33
00137,3300152,3310000,4000116,4000354,4000553,4000624,4000648,4000742,
4000833,4000879,4000955,4001064,4001131,4001145,4001188,4001192,400126
7,4001281,4001437,4001441,4001449,4001459,4001568",kCSI:{e:"31215,3570
2,37102,38371,39523,39977,3300025,3300119,3300124,3300133,33001,35,330
01373300152,3310000,4000116,4000354,4000553,4000624,4000648,4000742,40
00833,4000879,4000955,4001064,4001131,4001145,4001188,4001192,4001267,
4001281,400144001441,4001449,4001459,4001568",ei:"cHB1UO7WKcTGswa 754H
g37,BA"},authuseml:funct r:0,
```

Table 2.20 Frst 500 hexadecimal characters of the HTML Google home page (spaces included)

```
3C 21 64 6F 63 74 79 70 65 20 68 74 6D 6C 3E 3C 68 74 6D 6C 20 69 74 65
6D 73 63 6F 70 65 3D 22 69 74 65 6D 73 63 6F 70 65 22 20 69 74 65 6D 74
79 70 65 3D 22 68 74 74 70 3A 2F 2F 73 63 68 65 6D 61 2E 6F 72 67 2F 57
65 62 50 61 20 67 65 22 3E 3C 68 65 61 64 3E 3C 6D 65 74 61 20 69 74 65
6D 70 72 6F 70 3D 22 69 6D 61 67 65 22 20 63 6F 6E 74 65 6E 74 3D 22 2F
69 6D 61 67 65 73 2F 67 6F 6F 67 6C 65 5F 66 61 76 69 63 6F 6E 5F 31 32
38 2E 70 6E 67 22 3E 3C 74 69 20 74 6C 65 3E 47 6F 6F 67 6C 65 3C 2F 74
69 74 6C 65 3E 3C 73 63 72 69 70 74 3E 77 69 6E 64 6F 77 2E 67 6F 6F 67
6C 65 3D 7B 6B 45 49 3A 22 63 48 42 31 55 4F 37 57 4B 63 54 47 73 77 61
37 35 34 48 67 42 41 22 2C 67 65 74 45 49 3A 66 75 20 6E 63 74 69 6F 6E
28 61 29 7B 76 61 72 20 62 3B 77 68 69 6C 65 28 61 26 26 21 28 61 2E 67
65 74 41 74 74 72 69 62 75 74 65 26 26 28 62 3D 61 2E 67 65 74 41 74 74
72 69 62 75 74 65 28 22 65 69 64 22 29 29 29 29 61 3D 61 2E 70 61 72 65
20 6E 74 4E 6F 64 65 3B 72 65 74 75 72 6E 20 62 7C 7C 67 6F 6F 67 6C 65
2E 6B 45 49 7D 2C 68 74 74 70 73 3A 66 75 6E 63 74 69 6F 6E 28 29 7B 72
65 74 75 72 6E 20 77 69 6E 64 6F 77 2E 6C 6F 63 61 74 69 6F 6E 2E 70 72
6F 74 6F 63 6F 6C 3D 20 3D 22 68 74 74 70 73 3A 22 7D 2C 6B 45 58 50 49
3A 22 33 31 32 31 35 2C 33 35 37 30 32 2C 33 37 31 30 32 2C 33 38 33 37
31 2C 33 39 35 32 33 2C 33 39 39 37 37 2C 33 33 30 30 30 32 35 2C 33 33
30 30 31 31 39 2C 33 33 30 30 31 32 34 2C 33 20 33 30 30 31 33 33 2C 33
33 35 2C 33 33 30 30 31 33 37 2C 33 33 30 30 31 33 30 30 31
```

Table 2.21 Formula for solving the second degree equation $ax^2 + bx + c = 0$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.14}$$

Table 2.22 Latex notation of formula of Table 2.21

```
\begin{equation}
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\label{Tartaglia}
\end{equation}
```

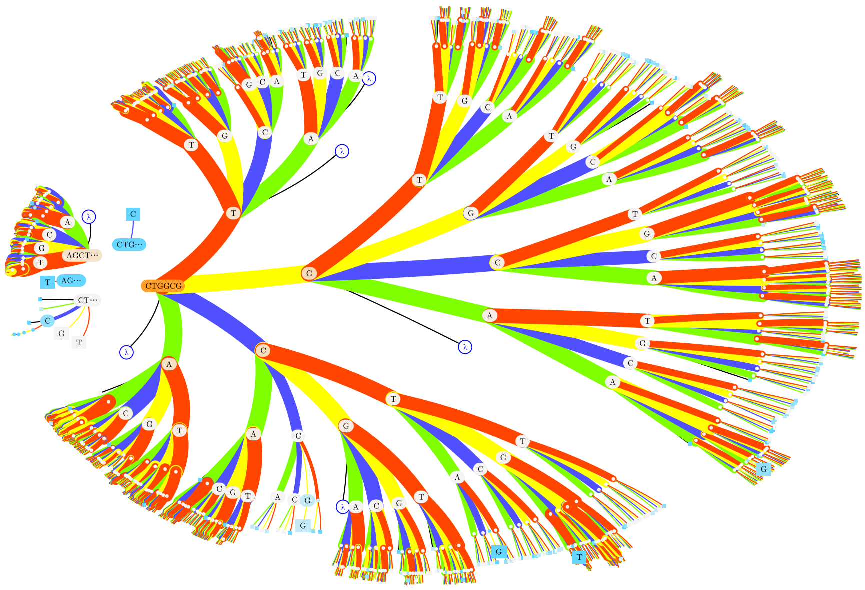


Fig. 2.59 A hyperbolic tree map of the genome of E. coli

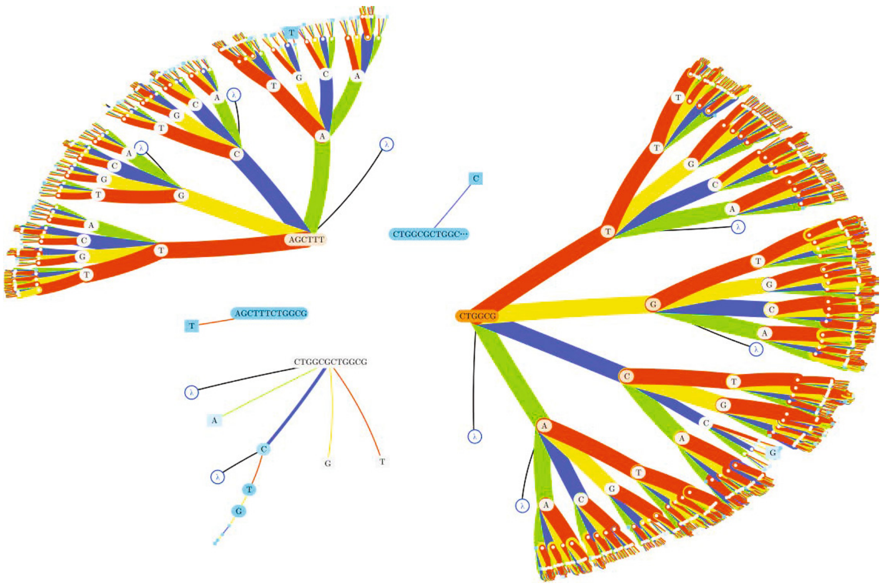


Fig. 2.60 Another view of the hyperbolic tree map of the genome of E. coli

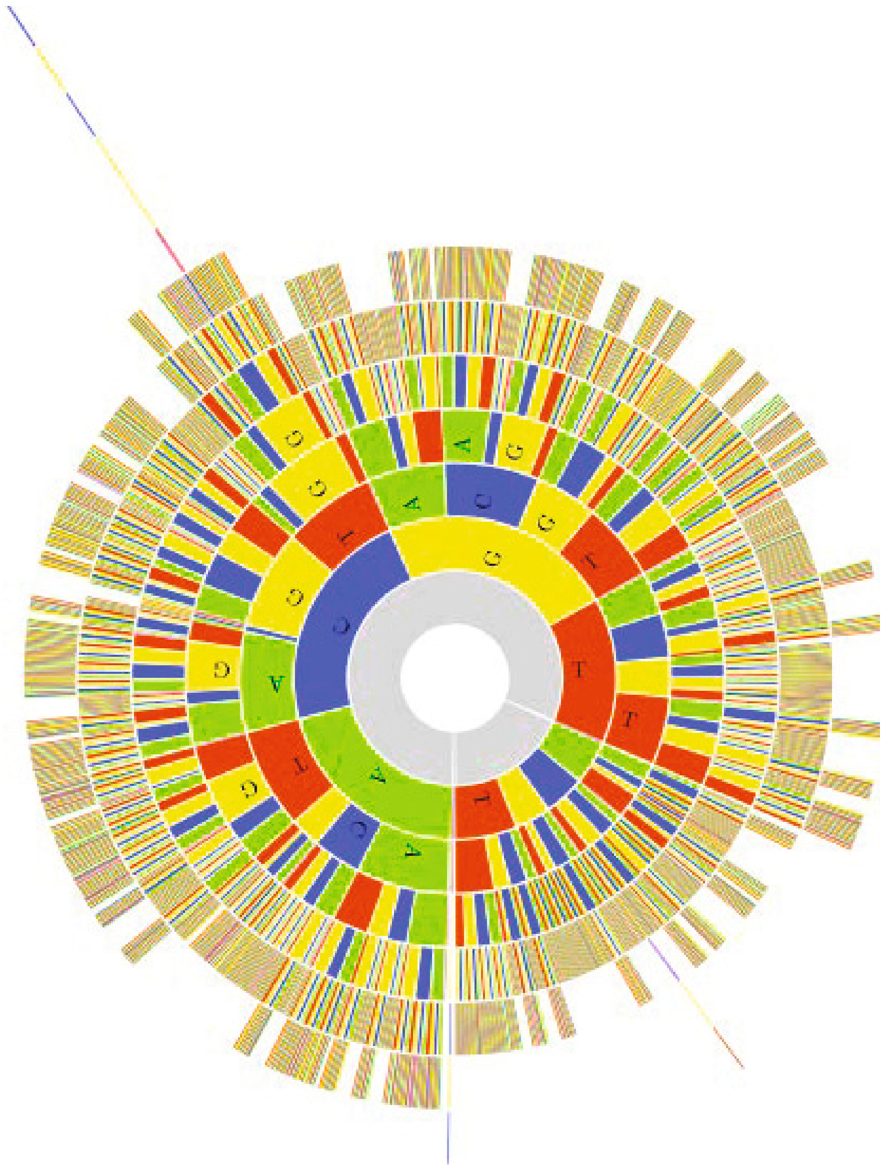


Fig. 2.61 A sector tree map of the genome of E. coli

Table 2.23 ASCII notation of formula of Table 2.21

```
92 92 98 101 103 105 110 123 101 113 117 97 116 105 111 110 125 32 120 32 61
32 92 102 114 97 99 123 45 98 32 92 112 109 32 92 115 113 114 116 123 98 94
50 32 45 32 52 97 99 125 125 123 50 97 125 32 92 108 97 98 101 108 123 84 97
114 116 97 103 108 105 97 125 92 101 110 100 123 101 113 117 97 116 105 111
110 125 32
```

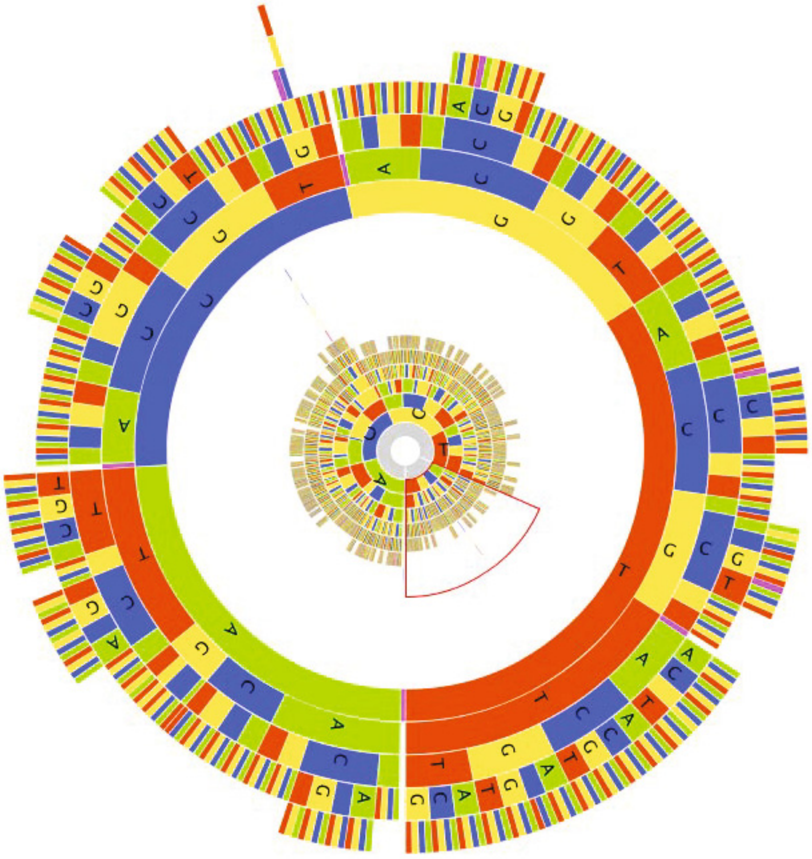


Fig. 2.62 A sector tree map of the genome of E. coli zooming out for a portion (the marked sector)

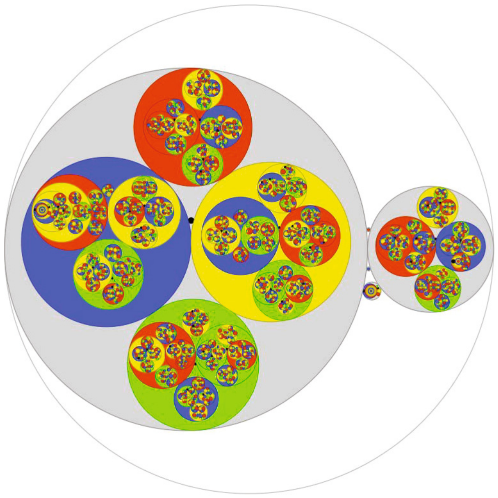


Fig. 2.63 A circle tree map of the genome of E. coli

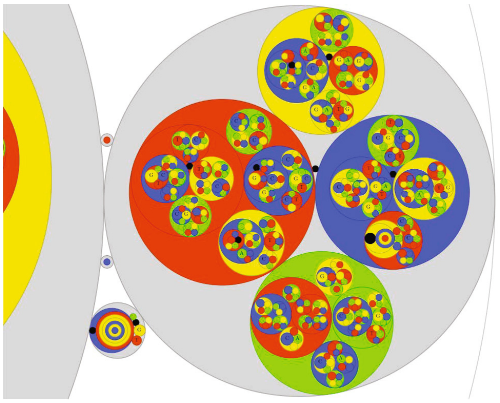
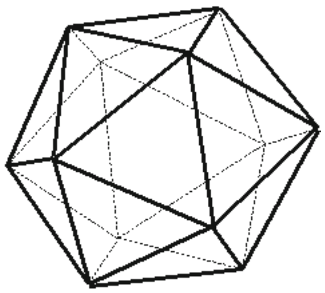


Fig. 2.64 A portion of the representation given in Fig. 2.63


Platonic Icosahedron

3 Algorithms and Biorhythms

Abstract. Algorithms define computation processes, while biorhythms constitute the dynamic of life. This chapter aims at showing the links between these two concepts, by giving examples where biorhythms can be described by suitable algorithms. Metabolism is the starting point of the analysis we develop, but the metabolic perspective is generalized in many directions, and a methodology is introduced for algorithmically solving dynamical inverse problems. The chapter is mostly based on the author's published papers (see **References for Chapter 3**).

3.1 Metabolic Grammars

P systems were introduced by Gheorghe Păun as a computation model inspired from the living cell [49, 50]. In 2004 a special kind of P systems, called MP systems, were introduced [98], specifically oriented to the symbolic modeling of biological processes [90, 106, 65, 68, 96]. This chapter is mainly devoted to present the basic concepts of MP theory and its applications to biological modeling [107, 101, 73, 105, 109].

Metabolism is one of the basic phenomena on which life is based. Any living organism has to maintain processes which: i) introduce matter of some kinds from the external environment, ii) transform internal matter by changing the molecule distribution of a number of biochemical species (substances, metabolites), and iii) expel outside matter that is not useful, or is dangerous for the organism. Molecule distribution identifies the metabolic state of a system, and can be represented as a multiset of type $n_A A + n_B B + \dots n_Z Z$ giving the numbers $n_A, n_B, \dots, +n_Z$ of molecules for each of molecular species $A, B, \dots, Z$.

DOI: 10.1007/978-3-642-36223-1_3 © Springer-Verlag Berlin Heidelberg 2013

A chemical reaction such as $A + B \rightarrow C$ means that a number u of molecules of kind A and the same number u of molecules B are replaced by u molecules of type C. This means that this reaction is a multiset rewriting rule. The value of u is the flux of the rule application. Let us observe a system at some time steps $0, 1, 2, \ldots, i$, and consider a substance x that is produced by rules r_1, r_3 and is consumed by rule r_2. If $u_1[i], u_2[i], u_3[i]$ are the fluxes of the rules r_1, r_2, r_3, respectively, in the passage from step i to step $i+1$, then the variation of substance x is given by:

$$x[i+1] - x[i] = u_1[i] - u_2[i] + u_3[i].$$

Then, passing from one step to the next (after a fixed time interval), the number of objects consumed and produced by a reaction r_i, for any single occurrence of reactant/product of r_i, is the **reaction flux** u_i, which depends on the state of the system by means of a map φ_i called the **regulator** of r_i. In this sense, a metabolic system is a dynamical system, because the quantities of its substances change along the time. In general, a **time series** $(X[i] \mid i \in \mathbb{N})$ is a sequence of real values intended as "equally spaced" in time. In the following we will study metabolic systems in a discrete mathematical perspective, by introducing **Metabolic P grammars**, or briefly **MP grammars**. These grammars are related to **P systems** introduced in 1998 by Gheorghe Păun [49, 50], and are a special kind of **multiset processing grammars** [98, 92, 93, 94, 95, 97].

Definition 3.1 (MP grammar). An MP grammar G is a generator of time series, determined by the following structure ($n, m \in \mathbb{N}$, the set of natural numbers):

$$G = (M, R, I, \Phi)$$

where:

1. $M = \{x_1, x_2, \ldots x_n\}$ is a finite set of elements called **metabolites** or **substances**. A **metabolic state** is given by a list of n values, each of which is associated to a metabolite;
2. $R = \{\alpha_j \rightarrow \beta_j \mid j = 1, \ldots m\}$ is a set of **rules**, or **reactions**, with α_j and β_j multisets over M for $j = 1, \ldots m$;
3. I are **initial values** of metabolites, that is, a list $x_1[0], x_2[0], \ldots x_n[0]$ providing the *metabolic state at step 0*;
4. $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ is a list of functions, called **regulators**, one for each rule, such that, for $1 \leq j \leq m$:
$$\varphi_j : \mathbb{R}^n \rightarrow \mathbb{R}. \quad (3.1)$$

G defines, for any $x \in M$, a time series:

$$(x[i] \mid i \in \mathbb{N}, i > 0) \quad (3.2)$$

in the following way. Let

$$s[i] = (x[i] \mid x \in M) \quad (3.3)$$

the **state vector** of G at step i, which can be seen as a function from the set of metabolites to $\mathbb{R}$, then the flux $u_j[i]$ of rule r_j at step i, is given by applying the regulator φ_j to the state $s[i]$:

$$u_j[i] = \varphi_j(s[i]). \tag{3.4}$$

For any $i \in \mathbb{N}$, the value of $x[i+1]$, for each $x \in M$ is given by the following equation, where $\alpha_j(x)$ and $\beta_j(x)$ denote the multiplicities of x in the multiset α_j and β_j, respectively:

$$x[i+1] = x[i] + \sum_{j=1}^{m} (\beta_j(x) - \alpha_j(x))u_j[i]. \tag{3.5}$$

An MP graph is a natural graphical representation of an MP grammar G (originated from [91]). An MP grammar G becomes an **MP system** when values for *time factor* τ, *population factor* ν, and *mass factor* μ are added to G (however, very often MP grammar and MP system are used synonymously):

- $\tau \in \mathbb{R}$ is the *time interval* between two consecutive steps;
- $\nu \in \mathbb{R}$ is the number of molecules which represents a *(conventional) mole* in the model;
- $\mu \in \mathbb{R}^n$ is the vector of the *mole masses* of metabolites.

An MP grammar G is **parametric**, when a set P of parameters is added to G, and metabolic states include also elements of P (to which, the state assigns real values). If G is parametric, the time series of parameters also has to be provided in order to specify G.

Table 3.1 gives an example of MP grammar, where $\emptyset$ denotes an empty multiset ($\emptyset$-rules correspond to introduction or expulsion of matter) and substance symbols occurring in regulators denote the corresponding substance quantities. The quantities that occur as arguments of a regulator (substances or parameters) are called the **tuners** of that regulator. In Table 3.1 p is a parameter, because it occurs as a tuner, but does not occur in any reaction. In the following, we are focused essentially on rules and regulators, then, often, initial values and parameters (if they are present) are given separately. Moreover, $X[i]$ will also be used for denoting the vector of metabolite values at step i (in the order they are specified), while $s[i]$ is the *state vector* extending $X[i]$ with the values of parameters at step i (if they are present). For the sake of simplification, we consider equivalent expressions such as: $\varphi(x_4[i], x_3[i], x_1[i])$ or $\varphi(x_4, x_3, x_1)[i]$, or $\varphi(s[i])$.

Three aspects are essential in the notion of MP grammar: **multiset rewriting**, **time discreteness**, and **molar perspective**. In fact:

1. rules transform multisets. We remark that an implicit assumption of Eqs. (3.5) in Definition 3.1 is that all the rules can be applied. This is a reasonable hypothesis if fluxes have values smaller than the quantities of available substances. If this is not the case, further strategies need to be specified which establish how to proceed when the hypothesis of metabolite abundance does not apply. For example, if the rules consume a metabolite quantity which is not available, then all the rules consuming it could be blocked at that step (or some of them according to some specified criterion);

Table 3.1 An MP grammar corresponding to the MP graph of Fig. 3.1

$r_1 : \emptyset \to A$	$\varphi_1 = 0.05p$
$r_2 : A \to B$	$\varphi_2 = 0.2C$
$r_3 : B \to \emptyset$	$\varphi_3 = 0.1$
$r_4 : A \to C$	$\varphi_4 = 0.6A/C$
$r_5 : C \to \emptyset$	$\varphi_5 = 0.4$
$A[0] = B[0] = C[0] = 1$	
$p[0] = 0.2,\ p[i+1] = p[i] + 0.2$	

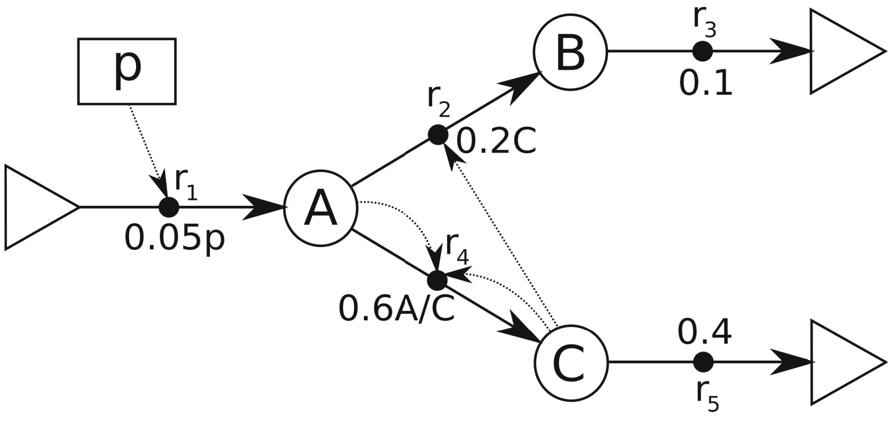


Fig. 3.1 The MP graph corresponding to the MP grammar of Table 3.1

2. fluxes refer to the substances consumed and produced between two consecutive steps at a fixed **time interval** with respect to which time is measured in the specific case of interest;
3. fluxes are expressed with respect a **conventional mole**, that is, a population size, depending on the particular phenomenon in question.

Any rule r of an MP grammar G determines two column vectors: the left vector $r^-$and the right vector r^+. The first vector provides the multiplicity of substances occurring in the multiset of reactants, while the second vector provides the multiplicities of the substances occurring in the multiset of products. The **stoichiometric balance** $r^\#$ of the rule r is defined as $r^+ - r^-$ (the difference of the two vectors). The matrix consisting of the column vectors $r^+ - r^-$ (in a conventional order of rules) is called the **stoichiometric matrix** of G. For example, the rules of MP grammar of Table 3.1 could be given by Table 3.2 reporting the pair of vectors and the corresponding stoichiometric balances.

Now, let us suppose we start from a state of a system where substances A, B, C have a 1-mole of elements (we do not enter into the specific value of our mole). Let us denote by $A[i], B[i], C[i]$ the molar values of these substances at step i. Then, the fluxes of our system are given, according to Table 3.1, by Table 3.3.

Table 3.2 Rule vectors of the MP grammar of Table 3.1

$r_1 = \left(\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}\right) \Rightarrow r_1^{\#} = \begin{pmatrix}1\\0\\0\end{pmatrix}$
$r_2 = \left(\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\right) \Rightarrow r_2^{\#} = \begin{pmatrix}-1\\1\\0\end{pmatrix}$
$r_3 = \left(\begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\end{pmatrix}\right) \Rightarrow r_3^{\#} = \begin{pmatrix}0\\-1\\0\end{pmatrix}$
$r_4 = \left(\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\right) \Rightarrow r_4^{\#} = \begin{pmatrix}-1\\0\\1\end{pmatrix}$
$r_5 = \left(\begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\0\end{pmatrix}\right) \Rightarrow r_5^{\#} = \begin{pmatrix}0\\0\\-1\end{pmatrix}$

Table 3.3 The fluxes of MP rules of Table 3.1

$u_1[i] : \varphi_1(s[i]) = 0.05p[i]$
$u_2[i] : \varphi_2(s[i]) = 0.2C[i]$
$u_3[i] : \varphi_3(s[i]) = 0.1$
$u_4[i] : \varphi_4(s[i]) = 0.6A[i]/C[i]$
$u_5[i] : \varphi_5(s[i]) = 0.4$

This means that if $U[0]$ denotes the (column) vector of fluxes at time 0, then the corresponding row vector of fluxes at time 0 is given by:

$$(u_1[0], u_2[0], u_3[0], u_4[0], u_5[0]) = (0.01, 0.2, 0.1, 0.6, 0.4). \tag{3.6}$$

In this notational setting it is easy to realize that the substance variation vector

$$(A[1] - A[0], B[1] - B[0], C[1] - C[0])$$

is given by the following matrix (row by column) product:

$$U[0] = (r_1^{\#}\ r_2^{\#}\ r_3^{\#}\ r_4^{\#}\ r_5^{\#}) \times U[0]$$

that is:

$$\begin{pmatrix}1 & -1 & 0 & -1 & 0\\ 0 & 1 & -1 & 0 & 0\\ 0 & 0 & 0 & 1 & -1\end{pmatrix} \times \begin{pmatrix}u_1[0]\\ u_2[0]\\ u_3[0]\\ u_4[0]\\ u_5[0]\end{pmatrix}.$$

Table 3.4 Substance variations, in the MP grammar of 3.1, in passing from step 0 to step 1

$\Delta_A[0] = A[1] - A[0] = u_1[0] - u_2[0] - u_4[0] = 0.01 - 0.2 - 0.6 = -0.79$
$\Delta_B[0] = B[1] - B[0] = u_2[0] - u_3[0] = 0.2 - 0.1 = 0.1$
$\Delta_C[0] = C[1] - C[0] = u_4[0] - u_5[0] = 0.6 - 0.4 = 0.2$

In fact, according to Eq. (3.6), we can compute the substance variations as reported in Table 3.4.

In this manner we compute the substance vector at time 1:

$$\begin{aligned} A[1] &= A[0] + \Delta_A[0] = 1.0 - 0.79 = 0.21 \\ B[1] &= B[0] + \Delta_B[0] = 1.0 + 0.1 = 1.1 \\ C[1] &= C[0] + \Delta_C[0] = 1.0 + 0.2 = 1.2 \end{aligned}$$

and, by applying the same method, we get the value of this vector in all the following steps. In other words, the metabolic grammar provides the time series of its metabolites. If $\Delta[i]$ is the vector of substance variation at step i:

$$\Delta[i] = (\Delta_x[i] \mid x \in M) \tag{3.7}$$

then, for any metabolite x:

$$x[i+1] = x[i] + \Delta_x[i] \tag{3.8}$$

therefore, the **dynamics generated** by a metabolic grammar with n substances and m reactions of regulators

$$\varphi_1, \varphi_2, \ldots, \varphi_m$$

and stoichiometric matrix

$$\mathbb{A} = (r_1^{\#}\ r_2^{\#}\ \ldots\ r_m^{\#}), \tag{3.9}$$

is completely expressed by the following **Equational Metabolic Algorithm**:

$$\Delta[i] = \mathbb{A} \times U[i] \tag{3.10}$$

where:

$$U[i] = \begin{pmatrix} \varphi_1(s[i]) \\ \varphi_2(s[i]) \\ \cdots\cdots\cdots \\ \varphi_m(s[i]) \end{pmatrix}. \tag{3.11}$$

In conclusion, an MP grammar G can be completely represented in four ways:

1. in textual format, by providing its reactions with regulators and its initial values (see, for example, Fig. 3.1);
2. by its corresponding MP graph (see, for example, Fig. 3.1);

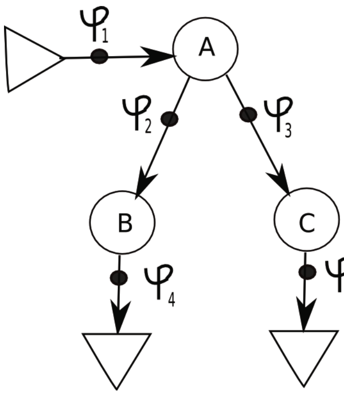


Fig. 3.2 The MP graph of MP grammar of Table 3.5

3. by the set of Eqs. (3.5) of Definition 3.1;
4. by the vector equation EMA(see 3.10).

The MP grammar given in Table 3.5 has the dynamics given in Fig. 3.3. It defines a synthetic oscillator, very often considered in the MP theory, called Sirius (see [92]). The oscillator is made of three substances A, B, and C and five reactions.

Table 3.5 The MP grammar of Sirius oscillator, corresponding to the MP graph of Fig. 3.2. Its dynamics with initial values $A[0] = 100$, $B[0] = 100$, $C[0] = 0.02$, is given in Fig. 3.3.

$r_1 : \emptyset \to A$	$\varphi_1 = 0.047 + 0.087A$
$r_2 : A \to B$	$\varphi_2 = 0.002A + 0.0002AC$
$r_3 : A \to C$	$\varphi_3 = 0.002A + 0.0002AB$
$r_4 : B \to \emptyset$	$\varphi_4 = 0.4B$
$r_5 : C \to \emptyset$	$\varphi_5 = 0.4C$

Table 3.6 The stoichiometric matrix of MP grammar of Table 3.5

$$\mathbb{A} = \begin{pmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

Table 3.7 reports a useful notation concerning MP rules which reveals an intrinsic duality between substances and reactions of an MP grammar. This notation can be easily extended to a set of substances and to a set of reactions. The *closure property* of a reaction graph of substances X and reactions Y can be expressed by the condition $R^o(S^o(Y)) = Y$ and $S^o(R^o(X)) = X$.

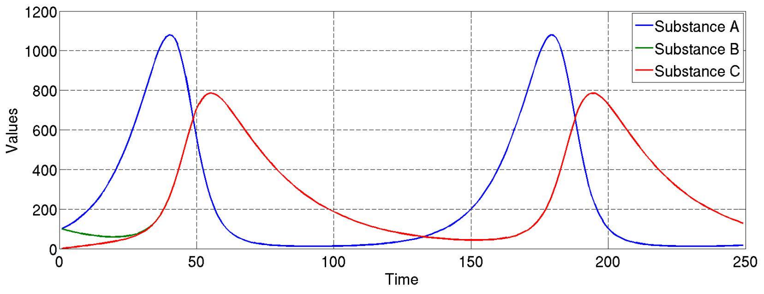


Fig. 3.3 The Sirius dynamics

Table 3.7 The duality between substances and reactions

Notation	Description
$R^{-}(x)$	reactions consuming substance x
$R^{+}(x)$	reactions producing substance x
$R^{o}(x)$	reactions consuming or producing substance x
$S^{-}(r)$	substances consumed by reaction r
$S^{+}(r)$	substances produced by reaction r
$S^{o}(r)$	substances consumed or produced by reaction r

3.1.1 Lotka-Volterra Dynamics

The Lotka-Volterra model is the simplest model of predator-prey interactions. The model was developed independently by Lotka ([89] and Volterra [116] and pertains to a population of individuals of two species: a prey species x and a predator species y. It was observed that, in time, the numbers of individuals of the two types oscillate according to a regular pattern. The reason for this oscillating dynamics can be qualitatively explained by considering that predators need prey for increasing their number but, at the same time, if the number of predators is too big, then prey cannot satisfy their sustainment, therefore predators decrease, and consequently prey can survive more easily so that prey population increases. The differential model of prey-predator dynamics is described by the following equations, where A, B, C, D are interaction parameters related to the average mortality and reproduction factors of the two species, and to the predator voracity, that is, the average percentage of prey eaten by predators in a time unit. In this way, under suitable simplifications (prey consumption is due only to predation, no other factors influence the dynamics, and these influences are almost stable within the considered ranges of the population sizes), the following equations are the invariant of the considered dynamics:

$$\frac{dx}{dt} = (A - By)x \tag{3.12}$$
$$\frac{dy}{dt} = (Cx - D)y.$$

The solution of these equations, starting from suitable initial values of the two populations exhibits a typical oscillation of both population sizes.

Now we show an MP analysis of the same phenomenon which provides the same conclusion. A prey is replicated, by means of a rule $x \rightarrow 2x$, in proportion to the number of prey (their average proliferation, and the availability of resources from the environment determine the proportionality factor). The rate of predator replication and prey death (rules r_2 and r_3) depends on both the number of prey and predators. Finally, predator death is proportional to predator population size (rule r_4). We do not discuss here the method for determining these parameters, which will be explained in the following sections, however, the MP grammar of Table 3.8 provides a typical Lotka-Volterra dynamics which is depicted on the left part of Fig. 3.4.

Table 3.8 MP grammar of Lotka-Volterra dynamics

Reactions	Regulators
$r_1 : x \rightarrow 2x$	$\varphi_1 = 0.03x$
$r_2 : x \rightarrow \emptyset$	$\varphi_2 = 0.0009 + 0.009xy + 0.0001x^2y$
$r_3 : y \rightarrow 2y$	$\varphi_3 = 0.0009 + 0.015xy + 0.0003x^2y$
$r_4 : y \rightarrow \emptyset$	$\varphi_4 = 0.066y$

We recall that, in accordance with our general notation, in Table 3.8, symbols x, y are used with different meanings. In fact, when they are used in MP-rules, they represent metabolites, while when they are arguments of regulators, they denote metabolite quantities in the given metabolic state.

3.1.2 The Brusselator (Belousov-Zhabotinsky Reaction)

Brusselator is an idealization, due to Prigogine's school [112], of a famous kind of chemical oscillating reaction, called BZ reactions, discovered by Belousov and then analyzed by Zhabotinsky. The best-known BZ reaction can be created with a mixture of potassium bromate $KBrO_3$, malonic acid $CH_2(COOH)_2$, and manganese sulfate $MnSO_4$ prepared in a heated solution of sulfuric acid H_2SO_4. Brusselator has the following form:

$$\begin{cases} a \rightarrow x \\ 2x + y \rightarrow 3x \\ b + x \rightarrow y + d \\ x \rightarrow e. \end{cases}$$

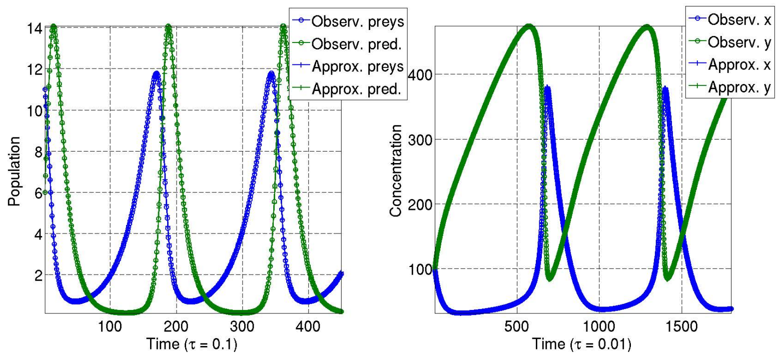


Fig. 3.4 On the left: observed prey and predator populations with $x[0] = 11$ and $y[0] = 6$. The approximated values are plotted against the ones calculated by the differential model given in Eq. (3.12) with $A = 0.3$, $B = 0.1$, $C = 0.18$ and $D = 0.7$. On the right: the Brusselator dynamics with $x[0] = y[0] = 100$.

We can further simplify these reactions by focusing on substances x,y, and by considering a,b,d,e as input/output substances providing the insertion or expulsion of x,y from/to the environment. Table 3.9 is an MP grammar of BZ reaction. In fact, its MP dynamics given on the right of Fig. 3.4 exhibits the classical form of Prigogine's Brusselator differential model. In this form a strict analogy of BZ with Lotka-Volterra dynamics can be envisaged, with the difference that a kind of y-to-x transformation and of x-to-y transformation is realized here by rules r_2 and r_3 respectively.

Table 3.9 MP grammar of Brusselator dynamics

Reactions	Regulators
$r_1 : \emptyset \rightarrow x$	$\varphi_1 = 1$
$r_2 : 2x + y \rightarrow 3x$	$\varphi_2 = 10^{-6}x^2y$
$r_3 : x \rightarrow y$	$\varphi_3 = 0.03x$
$r_4 : x \rightarrow \emptyset$	$\varphi_4 = 0.01x$

3.2 Time Series and Inverse Dynamics

Dynamical system is a pervasive concept in mathematics and in all sciences. A real number, seen as a process of generation of digits is a particular case of dynamical system. Analogously, a computation which starts from some initial data and transforms them along some elaboration steps determines a dynamical system. We find other examples of dynamical systems in a planet moving around a star, in the internal interactions of the particles inside an atom, in a chemical process, in a cyclone, in an economic process of resource exchange, in the course of an epidemic, or in the development of an organism.

The Greek philosopher Eracyitus says, "[...] Panta rei" ($\pi\grave{\alpha}\nu\tau\alpha\ \rho\grave{\varepsilon}\iota$), that is, "everything is changing" or also "existence is change". But if change follows a rule, something has not to change. In fact, in order to localize and describe a process as an entity, something has to remain stable during it. Existence is a mysterious mixing of variation and invariance underlying objects and events, at each level of reality. A **dynamical system** is a structure hosting a dynamics. The term dynamics points to the process while the term system points to the structure. A river, a city, a living organism, are always different in time, nevertheless, their individualities persist unchanged during their lives.

Since the epochal discovery of the laws of planetary motions, dynamical systems have been mathematically studied by means of differential equations. Recently the study of complex systems arising from life sciences, economics, meteorology, and many other fields, requires novel ideas and different approaches for the analysis and modeling of a wider class of dynamical systems, and for a great variety of aspects concerning the dichotomy structure/behavior. In particular, discrete dynamical systems seem to open new modeling possibilities and pose new problems where the algorithmic aspects replace the geometrical and differential perspective of classical dynamics.

In 1684 the astronomer Edmund Halley traveled to Cambridge to ask Newton about planetary motions. It was known that planets move around the Sun in ellipses. The question Halley posed to Newton was about the reason for this elliptic shape. Sir Isaac replied immediately: "I have calculated it". The general rigorous proof of this phenomenon was definitively given by Euler in 1749 (elliptic shape is a consequence of the inverse-square law and of the fundamental law of dynamics) [165].

Halley's question is a first example of the *dynamical inverse problem*, that is, given an observed behavior, is there a mathematical system that can explain this behavior? This is a crucial issue, in fact, when we can answer this question, then we "dominate" the system because we can predict or alter its behavior.

Any system changing in time can be identified with some variables. As they define a system, they must verify some relations which are the *invariant* of the behavior. Newton's solution to Halley's question was obtained first by determining some invariants of the planetary orbits, specified in terms of differential equations, then, by solving these equations, the mathematical form of these orbits was deduced. In general, in order to define a mathematical system related to a process, we need to determine its *variables* and *invariants*. This schema of analysis for dynamical systems is very general and continues to hold in different formal frameworks of dynamical representations. To suggest the wide range of this paradigm, let us consider the situation of a discrete phenomenon. Suppose we observe a device generating as output words (in a particular alphabet). A very natural question is: which is the grammar of the language generated by this device? And, in which manner does its internal structure realize the generation of these words? If we assume that any event can be discretely represented by a suitable word, then the search for rules underlying this process corresponds, in principle, to the search for a grammar. In the case of metabolic processes, we show that what is usually expressed in terms of differential equations can be formulated in terms of a special class of grammars, and very often these grammars are directly related to the biochemical mechanisms of the phenomenon under investigation.

Let us consider a biochemical system. Its state can be expressed by means of a multiset built on the set of its chemical species (molecule types). But we know that a chemical reaction is a rule transforming multisets of molecules into other multisets of molecules. This viewpoint implies that the whole dynamics of a biochemical system, where some reactions are active, corresponds to a grammar of multiset rewriting rules. Moreover, in order to provide a quantitative description of the effects produced by the reactions, these rules have to specify the amount of molecules they transform. If the evolution of a system is considered along a sequence of discrete steps, then a natural manner for expressing quantitative effects of rules can be expressed by equipping rules with functions (depending on the state of the system) which establish the fluxes, that is the quantities of substances transformed in any evolution step. This perspective of considering biochemical systems explains the central role of grammars in the description of biochemical dynamics, and consequently, the dynamical inverse problem for a biochemical system reduces to the search for suitable multiset grammars generating the sequence of states which we observe through time in the given system.

Given an MP system, the recurrent equation 3.10 of EMA generates the evolution of substances, according to the MP grammar of the system (starting from an initial metabolic state, and with the knowledge of parameter evolution in time). In other words, when regulators are given, the substance variations follow easily from the set of reactions defined in the system. The flux maps express the logic of an MP system in terms of its internal states.

Let us assume we “observe” a metabolic system for a number of steps (separated by some temporal intervals). The observation is represented by one **time series** for each observed substance, that is, the sequence of quantities of the substance in correspondence to the observation steps.

How can we discover an MP system which provides the dynamics which we observe? The substances and reactions in question are given by the particular phenomenon we want to describe. Moreover, the stoichiometry can be deduced by a basic knowledge of the phenomenon, but how do we know the fluxes of matter transformation that are responsible for the observed evolution? This is the problem of discovering the flux regulation maps, or simply, the **Regulation Discovery Problem**. Its solution can be described as the passage from the observation of the **behavior which a system exhibits in time** to the **determination of an internal logic that provides rules of transformation of its states**. In a metabolic system, an internal state is naturally representable by a multiset, therefore, we could synthesize the way of solving the dynamical inverse problem, for metabolic systems, by means of the following statement.

From time series to (multiset) grammars generating them.

A synthetic way to characterize the approach displayed above is **Gramma-dynamics**, because the solution to the a dynamical inverse problem is reduced to the solution of a regulation discovery problem, and then to the search of an MP grammar able to generate the observed time series. The methods that we develop, along this research line, are mainly based on algorithmic, algebraic, and statistical concepts, thus the more general term **Infodynamics** may be more appropriate.

3.3 Metabolic Approximation

Before introducing the algorithm that solves the regulation discovery problem in the context of MP systems, we present a procedure which discovers an MP grammar "approximating" some given time series exhibiting periodical behaviors [100].

Let us consider a real n-dimensional function $f : \mathbb{R} \to \mathbb{R}^n$. We pose the following approximation problem. Let $f[x_0], f[x_1], f[x_2], \dots, f[x_t]$ be the time series of f along a uniformly distributed sequence of values $x_0, x_1, x_2, \dots, x_t$. We are interested in finding an MP system such that, for some real vector K, and for some real vectors $\varepsilon[i]$ with $0 \leq \varepsilon[i] \leq \varepsilon$ and $0 \leq i \leq t$, the following equation holds (X is the vector of substance quantities of the system):

$$f[x_i] + K = X[i] \pm \varepsilon[i] \tag{3.13}$$

where the vector K provides *shift constants* for avoiding negative values, and vector ε is the error tolerated by the approximation.

In other words, we want to design an MP system exhibiting a dynamics that, within an ε-approximation, coincides with the time series $f[x_0], f[x_1], f[x_2], \dots, f[x_t]$ sampled from f. Another way to state synthetically this task could be: *providing an MP grammar approximating a real function*.

The previous formulation of the problem is too general, because no conditions are required to the structure of the MP system we are searching for. Of course, a natural indication could be the simplicity of the MP system. This means that the more the MP system is "parsimonious", the better our solution is. The parsimony of an MP system is given by two features: i) the reaction stoichiometry and ii) the form of its regulators. If we choose a class of regulation functions, we should search a minimal set of reactions providing the required behavior.

Now we show that there is a simple way, directly related to the definition of MP system, to approach our problem by using a mathematical regression method based on the solution of a *linear least-squares problem* [225]. The method of least-squares was first described by Carl Friedrich Gauss around 1794 and it is the standard approach to approximate solutions of overdetermined systems, that is, sets of equations in which there are more equations than unknowns. “Least-squares” means that the overall solution minimizes the sum of the squares of the errors made in solving the equation (see Sect. 7.7 for a mathematical explanation of the method).

Let us explain the procedure of metabolic approximation, in the particular case of a function from $\mathbb{R}$ to $\mathbb{R}^2$ providing the time series $(A[i], B[i])$ with $i = 0, \ldots, t$, by means of the following MP grammar ($\emptyset$ is the empty multiset) in which the values of c_1, c_2, ..., c_6 are unknowns:

$$\begin{array}{ll} \emptyset \to A & \varphi_1(A,B) = c_1 A + c_2 AB \\ 2A \to B & \varphi_2(B) = c_3 B + c_4 \\ B \to \emptyset & \varphi_3(A) = c_5 B + c_6. \end{array}$$

The stoichiometric matrix of the system is

$$\mathbb{A} = \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & -1 \end{pmatrix},$$

therefore the approximation system of equations will be, for $i = 0, 1, \ldots, t-1$,

$$\begin{array}{l} \Delta_A[i] = A[i+1] - A[i] = c_1 A[i] + c_2 A[i]B[i] - 2(c_3 B[i] + c_4) \\ \Delta_B[i] = B[i+1] - B[i] = c_3 B[i] + c_4 - (c_5 B[i] + c_6). \end{array} \tag{3.14}$$

If we call M the matrix of coefficients presented on the right-hand side of Eq. (3.14), the least-squares approximation of the coefficient vector $(c_1, c_2, c_3, c_4, c_5, c_6)^T$, expressed as a transposed row (exponent T denotes matrix transposition), for which our MP system exhibits a dynamics close to the time series we started from, is given by (see Sect. 7.7 and [225]):

$$\begin{pmatrix} \widehat{c_1} \\ \widehat{c_2} \\ \widehat{c_3} \\ \widehat{c_4} \\ \widehat{c_5} \\ \widehat{c_6} \end{pmatrix} = \left(M^T \times M\right)^{-1} \times M^T \times \begin{pmatrix} \Delta_A[0] \\ \Delta_B[0] \\ \Delta_A[1] \\ \Delta_B[1] \\ \cdots\cdots \\ \Delta_A[i] \\ \Delta_B[i] \end{pmatrix}$$

assuming that M has maximum rank[1] (i.e., equal to the number of its columns). At this point we can apply EMA to the MP system which uses $\widehat{c_1}, \widehat{c_2}, \ldots, \widehat{c_6}$ as values for $c_1, c_2, \ldots, c_6$. If the system provides a dynamics close enough to the time series $(A[i], B[i])$ with $i = 0, \ldots, t$, then the method stops. If the approximation error is already too big, the method tries to modify the initial MP grammar by introducing some rules deduced by a suitable correlation analysis of the approximation error.

[1] If M does not have a maximum rank, then the matrix given by the product $M^T \times M$ is not invertible.

3.3.1 Goniometricus: A Metabolic Grammar of Sine and Cosine

In this section we will outline a method to develop MP approximations of sine and cosine functions. This will be done by defining an MP model, called *Goniometricus*, which has two substances, *S* and *C*, approximating in time the evolution of the functions sine and cosine respectively (apart from a constant offset).

We start from two time series of the same length, related to the two target functions *sin* and *cos* along two oscillations (from 0 to 4π) with a step of 10^{-3}. As depicted in Fig. 3.5, each value has been increased by 3 units to make all values positive:

$$cos(0)+3\,,\ cos(0.001)+3\,,\ cos(0.002)+3\,,\ \ldots\ cos(4\pi)+3$$

$$sin(0)+3\,,\ sin(0.001)+3\,,\ sin(0.002)+3\,,\ \ldots\,,\ sin(4\pi)+3.$$

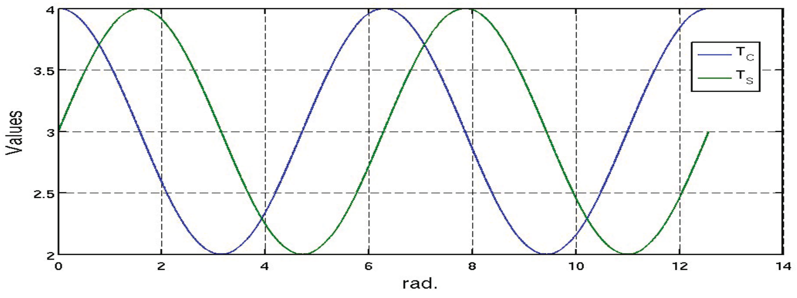


Fig. 3.5 The plot of the time series T_C and T_S

When we start with angle 0, then cosine is 1 and sine is 0. Going from 0 to $\pi/2$ cosine decreases and sine increases. Therefore, we can assume a transformation from cosine to sine. Moreover, initially the sum of sine and cosine is 1, but, when angle $\pi/4$ is reached, the sum of sine and cosine values is $\sqrt{2}$, which is greater than 1. This means that rules of increase and decrease of "matter" in the system are appropriate. This intuition suggests the MP grammar given in Table 3.10, and depicted by the MP graph of Fig. 3.6, which allows us to find, by means of Least Square Evaluation, the right constants providing a good approximation of the required dynamics.

Table 3.10 A form for Goniometricus's MP grammar

$r_1 : \emptyset \rightarrow C$	$\varphi_1 = c_1 + c_2\, C$
$r_2 : C \rightarrow S$	$\varphi_2 = c_3\, S + c_4\, C$
$r_3 : S \rightarrow \emptyset$	$\varphi_3 = c_5 + c_6\, S$

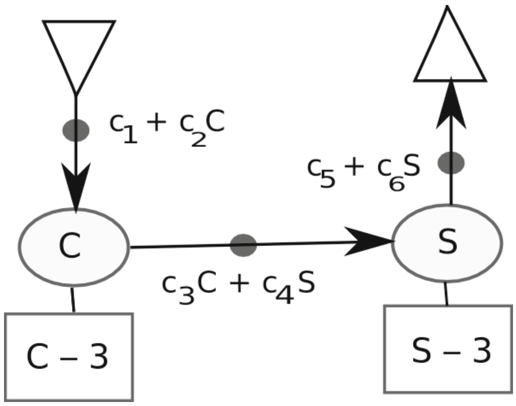


Fig. 3.6 Goniometricus's MP graph related to the MP grammar given in Table 3.10

The least square evaluation of the following system of equations with unknowns $c_1, \dots c_6$ will provide the regulators of the grammar we are searching for:

$$\begin{cases} c_1 + c_2\,C[0] - c_3\,S[0] - c_4\,C[0] & = \Delta_C[0] \\ c_3\,S[0] + c_4\,C[0] - c_5 - c_6\,S[0] & = \Delta_S[0] \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & = \dots\dots \\ c_1 + c_2\,C[n] - c_3\,S[0] - c_4\,C[n] & = \Delta_C[n] \\ c_3\,S[n] + c_4\,C[n] - c_5 - c_6\,S[0] & = \Delta_S[n]. \end{cases} \tag{3.15}$$

The least square evaluation of system (3.15) provides sine and cosine with an approximation error of 10^{-14}. The initial values of C and S are $4,3$ respectively (for keeping fluxes positive), $\tau = 10^{-3} \times 2\pi$ (the period is 2π), and $\nu = \mu = 1$. The values of the constants are reported in Table 3.11.

Table 3.11 Goniometricus's MP grammar. Parameters $C-3$ and $S-3$ shift the values of C and S in the interval $[-1,1]$.

$r_1 : \emptyset \to C$	$\varphi_1 = 0.0030015 + 0.0009995\,C$
$r_2 : C \to S$	$\varphi_2 = 0.001\,C + 0.001\,S$
$r_3 : S \to \emptyset$	$\varphi_3 = 0.0029985 + 0.0010005\,S$

3.3.2 Generalization of the Goniometricus Model

The approximation power of MP grammars may increase by considering "memories". Given a substance A, we denote by A_{-m} ($m \in \mathbb{N}$) the *memory* of A of level m. It is a substance obtained from A such that $A_{-m}[i] = A[i-m]$. Therefore its time series $T_{A_{-m}}$ is obtained from the time series $T_A = [t_{A,1}, t_{A,2}, \dots, t_{A,n}]$ of A by the following shifting operation: $T_{A_{-m}} = [0,\dots,0,t_{A,1},\dots,t_{A,n-m}]$. Usually, two memory levels are enough for defining MP approximations of very complex functions. MP grammars M_1 and M_2 [100], having the form given in Table 3.12, which approximate $(f_1(x), g_1(x)) = (cos(x)^3, sin(x)^3)$ and $(f_2(x), g_2(x)) = (cos(0.99x), sin(0.99x))$ respectively (initial values $3,2$, shift of each substance 2, $\tau = 10^{-3} \times 2\pi$ and $\nu = \mu = 1$).

M_1 provides approximation order 10^{-6} and uses two memory levels with:

$c_1 = 0.0133055979719, c_{2,0} = 38.2585289672269, c_{2,1} = -76.6872395145061,$
$c_{2,2} = 38.4397714886722, c_3 = 0.0053099516581, c_{4,0} = 35.6377416209758,$
$c_{4,1} = -70.8226212812310, c_{4,2} = 35.1955107358582, c_{5,0} = 14.5027497660384,$
$c_{5,1} = -29.4732582998089, c_{5,2} = 14.9749362227182, c_{6,0} = 57.3945207457749,$
$c_{6,1} = -115.0336102507111, c_{6,2} = 57.6568032453146.$

M_2 does not use memories, and provides approximation order 10^{-14} with:

$c_1 = 0.000196018399012, c_{2,0} = 0.009850833684540, c_3 = -0.019701667369079,$
$c_{4,0} = 0.009899838284292, c_{5,0} = 0.009899838284293, c_{6,0} = 0.009948842884046.$

Table 3.12 The Goniometricus MP grammar generalized with two memory levels

$r_1 : \emptyset \to A$	$\varphi_1 = c_1 + c_{2,0}\,A + c_{2,1}\,A_{-1} + c_{2,2}\,A_{-2}$
$r_2 : A \to B$	$\varphi_2 = c_3 + c_{4,0}\,B + c_{4,1}\,B_{-1} + c_{4,2}\,B_{-2} +$ $c_{5,0}\,A + c_{5,1}\,A_{-1} + c_{5,2}\,A_{-2}$
$r_3 : B \to \emptyset$	$\varphi_3 = c_{6,0}\,B + c_{6,1}\,B_{-1} + c_{6,2}\,B_{-2}$

3.4 Stoichiometric Expansion and Stepwise Regression

The method which we presented in the previous section, for approximating real functions, is based on grammars with very simple rules. In this case, we assumed regulators having linear forms. In general, an MP grammar generating a given dynamics can be very complex, since we cannot *a priori* restrict the forms of regulators. For this reason we need to extend our method for solving the Regulation Discovery Problem within MP grammars.

Let us suppose we know some time series of states[2], that is, the vector sequence

$$(s[i] | i \in \mathbb{N}),$$

[2] Very often the time series from which the inverse dynamical problem starts are not at regular time intervals. In this case a preprocessing phase is appropriate for determining an interpolation curve fitting the observed values along the observation points. After this, we will consider the time series given by the values of the interpolation curve at steps with the same time interval.

then we can read the equation *EMA* by reversing the known values with the unknown ones. In fact, by writing the substance variation vector:

$$s[i+1] - s[i] = \Delta[i]$$

and, assuming n substances and m reactions, we get the following system, which we call *ADA* (Avogadro-Dalton-Action, see [94]), for remarking that here, differently from *EMA* of Eqs. (3.5), the unknown values are fluxes $U[i]$:

$$\mathbb{A} \times U[i] = \Delta[i]. \tag{3.16}$$

For the determination of the regulators which provide the best approximate solution of Eq. (3.16), we apply a procedure which we call *stoichiometric expansion* (see [102, 103, 104]).

Given a positive integer d, let us assume that the regulators we are searching for can be expressed as linear combinations of some basic regressors

$$g_1, g_2, \dots, g_d$$

which usually include constants, powers, and products of substances, plus some basic functions which are considered suitable in the specific cases under investigation:

$$\begin{aligned} \varphi_1 &= c_{1,1}g_1 + c_{1,2}g_2 + \dots + c_{1,d}g_d \\ \varphi_2 &= c_{2,1}g_1 + c_{2,2}g_2 + \dots + c_{2,d}g_d \\ \dots &= \dots\dots\dots\dots \\ \varphi_m &= c_{m,1}g_1 + c_{m,2}g_2 + \dots + c_{m,d}g_d. \end{aligned} \tag{3.17}$$

Equation (3.17) can be written, in matrix notation, in the following way, where $U[i]$ is the column vector of regulators evaluated at state s_i, $\mathbb{G}[i]$ the column vector of regressors evaluated at the same state, and $\mathbb{C}^T$ is the matrix $m \times d$ of the unknown coefficients of regressors (exponent T denotes matrix transposition):

$$U[i] = \mathbb{C}^T \times \mathbb{G}[i]. \tag{3.18}$$

Substituting the right member of Eq. (3.18) into Eq. (3.16), we obtain the following system of equations ($\mathbb{A}$ is the stoichiometric matrix):

$$\mathbb{A} \times \mathbb{C}^T \times \mathbb{G}[i] = \Delta[i]. \tag{3.19}$$

Now, if we consider t systems of type (3.19), for $1 \le i \le t$, and if n is the number of variables, we obtain nt equations with md unknown coefficients of $\mathbb{C}$. If $nt > md$ and the system has maximum rank, then we can apply a Least Square Evaluation which provides the coefficients that minimize the errors between the left and right sides of the equations. These coefficients provide the regulator representations that we are searching for.

Now we provide a compact representation of the unknown coefficients constituting the matrix $\mathbb{C}$, by using suitable matrix operations. Let us consider the *t*-**expansions** $\phi_1^t, \phi_2^t, \ldots, \phi_m^t$ of regulators as the vectors constituted by the right members of Eqs. (3.17) evaluated along t steps (where the values of all the variables of the system are known), and analogously the t-expansions $G_1^t, G_2^t, \ldots, G_d^t$ of the regressors $g_1, g_2, \ldots, g_d$ along the same t steps. With this notation Eqs. 3.17 provide a linear system with $d \times m$ unknowns:

$$\begin{aligned} \phi_1^t &= c_{1,1}G_1^t + c_{1,2}G_2^t + \ldots + c_{1,d}G_d^t \\ \phi_2^t &= c_{2,1}G_1^t + c_{2,2}G_2^t + \ldots + c_{2,d}G_d^t \\ \ldots &= \ldots\ldots\ldots\ldots \\ \phi_m^t &= c_{m,1}G_1^t + c_{m,2}G_2^t + \ldots + c_{m,d}G_d^t. \end{aligned} \tag{3.20}$$

Now, let $C_1^d, C_2^d, \ldots, C_m^d$ be the unknown column vectors of dimension d constituted by the coefficients of the regressors providing the linear combinations of regulators $\varphi_1, \varphi_2, \ldots, \varphi_m$ we are searching for, and

$$\mathbb{C} = (C_1^d, C_2^d, \ldots, C_m^d)$$

the matrix having these vectors as columns.

Let also $\mathbb{F}$ be the following matrix constituted by m column vectors of t elements:

$$\mathbb{F} = (\phi_1^t, \phi_2^t, \ldots, \phi_m^t).$$

Finally, let

$$\mathbb{G} = (G_1^t, G_2^t, \ldots, G_d^t)$$

be the matrix of dimension $t \times d$ having as columns the t-expansions of regressors.

With the notation above Eq. (3.20) becomes:

$$\mathbb{G} \times \mathbb{C} = \mathbb{F} \tag{3.21}$$

Moreover, let $\Delta_1^t, \Delta_2^t, \ldots, \Delta_n^t$ be the column vectors of dimension t constituted by substance variations of substances, from step i to step $i+1$, for $0 \le i \le t$, and:

$$\mathbb{D} = (\Delta_1^t, \Delta_2^t, \ldots, \Delta_n^t)$$

the matrix having these vectors as columns.

By using matrix transposition (denoted by the exponent T), Eq. (3.16) $\mathbb{A} \times U[i] = \Delta[i]$ becomes:

$$U[i]^T \times \mathbb{A}^T = \Delta[i]^T = (\Delta_1[i], \Delta_2[i], \ldots, \Delta_n[i])$$

which, expanded along the t time points, provides:

$$\mathbb{F} \times \mathbb{A}^T = \mathbb{D} \tag{3.22}$$

Therefore, by combining Eqs. (3.21) and (3.22), it follows that:

$$\mathbb{G} \times \mathbb{C} \times \mathbb{A}^T = \mathbb{D}. \tag{3.23}$$

We show that regressor coefficients of $\mathbb{C}$ can be obtained by a least square estimation deduced by Eq. (3.23), by using the **direct product** $\otimes$ between matrices, also called the Kronecker product, a special case of *tensor product* used in linear algebra and in mathematical physics.

Given two real matrices A, B of dimension $n \times m$ and $t \times d$ respectively, the *direct product*:

$$A \otimes B$$

is the matrix of dimension $nt \times md$, constituted by nm blocks $(A \otimes B)_{i,j}$, such that, if $A = (a_{i,j} \mid 1 \leq i \leq n,\ 1 \leq j \leq m)$, then $(A \otimes B)_{i,j} = a_{i,j}B$ (all the elements of B are multiplied by $a_{i,j}$). If we use the block notation, $A \otimes B$ can be represented in the following way:

$$\begin{pmatrix} a_{1,1}B & a_{1,11}B & \dots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,mn}B \\ \dots & \dots & \dots & \dots \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,m}B \end{pmatrix}. \tag{3.24}$$

The Kronecker product is bilinear and associative, that is, it satisfies the following equations:

$$\begin{aligned} A \otimes (B + C) &= (A \otimes B) + (A \otimes C) \\ (A + B) \otimes C &= (A \otimes B) + (A \otimes C) \\ (kA) \otimes B &= A \otimes (kB) = k(A \otimes B) \\ (A \otimes B) \otimes C &= A \otimes (B \otimes C). \end{aligned}$$

Moreover, matrix direct product verifies the following equations (the last equation when matrices are invertible):

$$\begin{aligned} (A \otimes B) \times (C \otimes D) &= (A \times C) \otimes (B \times D) \\ (A \otimes B)^T &= A^T \otimes B^T \\ (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1}. \end{aligned}$$

The following Lemma asserts a useful property of the Kronecker product [87].

Lemma 3.2 (Vectorization Lemma). *Let us denote by $vec(W)$ the* vectorization *of a matrix W, obtained by concatenating all the columns of W, in their order, in a unique column vector. Then the following equation holds:*

$$vec(A \times X \times B) = (B^T \otimes A) \times vec(X). \tag{3.25}$$

Proof. Let A be a matrix $n \times h$, X a matrix $h \times k$, and B a matrix $k \times m$. Let us denote by A_i the i-row vectors of a matrix A and by X^j the j-column vectors of X. With this notation we evaluate both members of Eq. (3.25). The following matrix product:

$$A \times X \times B \tag{3.26}$$

is equal to:

$$\begin{pmatrix} A_1{\times}X^1 & A_1{\times}X^2 & \dots & A_1{\times}X^k \\ A_2{\times}X^1 & A_2{\times}X^2 & \dots & A_2{\times}X^k \\ \dots & \dots & \dots & \dots \\ A_n{\times}X^1 & A_n{\times}X^2 & \dots & A_n{\times}X^k \end{pmatrix} \times B\,.$$

Therefore, if A_iX^j abbreviates the scalar product $A_i{\times}X^j$ of the row vector A_i with the column vector X_j, then the product $A \times X \times B$ is equal to:

$$\begin{pmatrix} b_{1,1}A_1X^1 + b_{2,1}A_1X^2 + \dots + b_{k,1}A_1X^k & \dots & b_{1,m}A_1X^1 + b_{2,m}A_1X^2 + \dots + b_{k,m}A_1X^k \\ b_{1,1}A_2X^1 + b_{2,1}A_2X^2 + \dots + b_{k,1}A_2X^k & \dots & b_{1,m}A_2X^1 + b_{2,m}A_2X^2 + \dots + b_{k,m}A_2X^k \\ \dots\dots\dots & \dots & \dots\dots\dots \\ b_{1,1}A_nX^1 + b_{2,1}A_nX^2 + \dots + b_{k,1}A_nX^k & \dots & b_{1,m}A_nX^1 + b_{2,m}A_nX^2 + \dots + b_{k,m}A_nX^k \end{pmatrix}. \tag{3.27}$$

The right member of Eq. (3.25) is:

$$(B^T \otimes A) \times vec(X) \tag{3.28}$$

which is equal to:

$$\begin{pmatrix} b_{1,1}A & b_{2,1}A & \dots & b_{k,1}A \\ b_{1,2}A & b_{2,2}A & \dots & b_{k,2}A \\ \dots & \dots & \dots & \dots \\ b_{1,m}A & b_{2,m}A & \dots & b_{k,m}A \end{pmatrix} \times \begin{pmatrix} X^1 \\ X^2 \\ \dots \\ X^k \end{pmatrix}$$

that is, by making explicit the rows of A:

$$\begin{pmatrix} \begin{pmatrix} b_{1,1}A_1 & b_{2,1}A_1 & \dots & b_{k,1}A_1 \\ b_{1,1}A_2 & b_{2,1}A_2 & \dots & b_{k,1}A_2 \\ \dots & \dots & \dots & \dots \\ b_{1,1}A_n & b_{2,1}A_n & \dots & b_{k,1}A_n \end{pmatrix} \times \begin{pmatrix} X^1 \\ X^2 \\ \dots \\ X^k \end{pmatrix} \\ \\ \begin{pmatrix} b_{1,2}A_1 & b_{2,2}A_1 & \dots & b_{k,2}A_1 \\ b_{1,2}A_2 & b_{2,2}A_2 & \dots & b_{k,2}A_2 \\ \dots & \dots & \dots & \dots \\ b_{1,2}A_n & b_{2,2}A_n & \dots & b_{k,2}A_n \end{pmatrix} \times \begin{pmatrix} X^1 \\ X^2 \\ \dots \\ X^k \end{pmatrix} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \begin{pmatrix} b_{1,m}A_1 & b_{2,m}A_1 & \dots & b_{k,m}A_1 \\ b_{1,m}A_2 & b_{2,m}A_2 & \dots & b_{k,m}A_2 \\ \dots & \dots & \dots & \dots \\ b_{1,m}A_n & b_{2,m}A_n & \dots & b_{k,m}A_n \end{pmatrix} \times \begin{pmatrix} X^1 \\ X^2 \\ \dots \\ X^k \end{pmatrix} \end{pmatrix}$$

but this vector corresponds to the vectorization of matrix (3.27), therefore the proof is completed. □

If we apply the Vectorization Lemma to Eq. (3.23), we obtain:

$$(\mathbb{A} \otimes \mathbb{G}) \times vec(\mathbb{C}) = vec(\mathbb{D}) \tag{3.29}$$

where the *stoichiometric matrix* $\mathbb{A}$ is multiplied by the Kronecker product with the *regressor matrix* $\mathbb{G}$ and the result is multiplied with the vectorization of the *regressor coefficient matrix* $\mathbb{C}$, and then equated to the vectorization of the *substance variation matrix* $\mathbb{D}$, by providing nt equations with md unknown values.

According to the least square approximation method (see Sect. 7.7), if matrix $\mathbb{A} \otimes \mathbb{G}$ has maximum rank (that is, matrices $\mathbb{A}$ and $\mathbb{G}$ have maximum rank), and $nt \geq md$, then the best approximation to $vec(\mathbb{C})$ minimizing the difference between the two members of Eq. (3.29) is given by the following vector:

$$vec(\mathbb{C}) = \left((\mathbb{A} \otimes \mathbb{G})^T \times (\mathbb{A} \otimes \mathbb{G})\right)^{-1} \times (\mathbb{A} \otimes \mathbb{G})^T \times vec(\mathbb{D}). \tag{3.30}$$

Some constraints may be imposed to the fluxes provided by regulators, which may be of general nature, or may be specific to some classes of systems (for example, fluxes should not be negative, and the sum of fluxes of all reactions consuming a substance x cannot exceed the quantity of x).

However, the approximation given by the matrix expression (3.30) cannot in general be considered the best way for solving the dynamical inverse problem which we started from. In fact, apart from the computational cost of considering all the d regressors at the same time, several reasons suggest we should follow a gradual strategy in the determination of a subset of regressors, and their corresponding coefficients, which provide the best approximation of the given dynamics. In fact, two main requirements, which we will discuss in a future section are essential for an appropriate application of the least square method: the linear independence among the regressor expansions and the minimality of the set of regressors.

In conclusion, the best approximation is obtained by determining a minimal set of linearly independent regressors ensuring an error under a given threshold.

Linear independence is a requirement of least square method, while the minimality avoids problems of over-fitting of the approximate solution. In fact, the more regressors are considered, the less is the degree of freedom of the solution. This implies that the solution fits very well with the dynamics on the observation points, but it is too constrained to them for behaving in a satisfactory way outside them. Therefore, in order to cope with these requirements we integrate the least square method by developing, in the next sections, an algorithm, which we call **LGSS** (Log-Gain Stoichiometric Stepwise) for selecting the best regressors among a set of possible regressors.

The stoichiometric expansion method allows us to determine the best coefficients which approximate regulators, as combinations of some specified regulators. Another two crucial aspects of LGSS algorithm concern i) the determination of which regressors have to be considered for each flux regulation function, ii) how to choose those which provide the best approximations. We address these aspects by integrating the stoichiometric expansion with two mechanisms: i) a way for scoring the regressors of an initial dictionary of functions, by using the Log-gain principle, and ii) a way for choosing the best regressors for each flux regulator, by using an extension of the classical stepwise regression method, based on the least square approximation and on the F statistical test. In the following subsections we develop the details necessary for a complete description of the LGSS algorithm.

3.4.1 Log-Gain Principle

In the previous section, we have explained how stoichiometric expansion approximates the coefficients of regressors which provide the regulators of a given MP grammar when regressors are given. In the following subsections we will focus on the problem of finding which regressors provide the best approximation of regulators, among a set of d basic functions $g_1, g_2, \ldots, g_d$ depending on substance quantities and parameters. Here we introduce a principle, called *Log-gain principle*, which is a special case of a general criterion usually satisfied by the variations of quantities involved in biological phenomena.

Given a time series $v^t = (v[i] \mid 0 \leq i \leq t)$, of non-null real values its discrete log-gain vector is defined by:

$$Lg(v^t) = (Lg(v[i]) \mid 0 \leq i \leq t) = \left(\frac{v[i+1] - v[i]}{v[i]} \mid 0 \leq i < t \right). \tag{3.31}$$

The log-gain principle requires that the flux vector $u_l[i] = \varphi_l(s[i])$ associated to a reaction r_l has to be a linear combination of log-gains of the time series of tuners of φ_l. Namely, let $s^t = (s[i] \mid 0 \leq i \leq t)$ be the time series of states along some uniformly distributed observation points $0 \leq i \leq t$, with a corresponding time series of fluxes given by the regulator φ_l:

$$\varphi_l^t = (\varphi_l(s[i]) \mid 0 \leq i \leq t).$$

For any substance or parameter y, if the corresponding time series of values along the state time series s^t is:

$$y^t = (y[i] \mid 0 \leq i \leq t),$$

then the log-gain principle is satisfied by φ_l^t when the following equation holds for suitable real coefficients q_j, where T_l is the set of tuners of φ_l:

$$Lg(\varphi_l^t) = \sum_{y_j \in T_l} q_j Lg(y_j^t).$$

This principle is a discrete formulation [92, 93] of a general principle holding very often in biological systems, in order to keep a global equilibrium among all its components. The term log-gain is related to the fact that the continuous version of the log-gain of a real derivable function f, is the derivative of its logarithm, expressing its relative (infinitesimal) variation $\frac{df(x)}{dt}/x = \frac{d\log f(x)}{dt}$. In general terms, we can assume that in a biological system dynamics obeys a global equilibrium among variables. Namely, the relative variation of a variable, depending on certain independent variables, has to be a linear combination of its independent variables. In the influential book [66], the biological significance and relevance of this principle, in many specific contexts, is widely discussed.

For evaluating to which extent a regressor verifies the log-gain principle, we compute for each of them a log-gain score, by using a least- squares approximation, where the log-gain of the regressor, along the observation points, is equal to a linear combination of the log-gain of its tuners. For example, if we want to test the goodness of the regressor AB, given by the product of two time series (A and B), then we need to calculate the least- squares approximation for c_1, c_2 such that

$$Lg(A[i]B[i]) = c_1 Lg(A[i]) + c_2 Lg([B[i])$$

where $0 \leq i < t$. Then the obtained approximation error determines a criterion for establishing a *log-gain ordering* among regressors by assigning greater log-gain scores to the ones which provide smaller approximation errors.

3.4.2 The Stepwise Regression LGSS

Once regressors are ranked according to their log-gain scores, their choice is based on a regression procedure which develops the ideas introduced in Sect. 3.3, by extending the classical method of stepwise regression [216, 76, 85]. Before presenting LGSS regression, we will recall the classical *k-variable multiple regression*. The reader can find more details and statistical motivations in Chapter 7 (Sect. 7.7) and in Aczel and Sounderpandian's book [216], from which we adopt the notation.

The following equation is the general form of a linear regression. Statistics provides methods for finding the right coefficients, possibly null, if they exist, expressing the kind of relationship between a *dependent variable* Y and one or more *independent variables* $X_1, X_2, \ldots, X_k$ (if $k > 1$, then the regression equation above is called a **multiple regression model**):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon. \tag{3.32}$$

The LGSS algorithm, given in the gray box at the end of this section, is a procedure extending the one defined in Sect. 3.3, where MP systems were applied to the problem of approximating real functions. Also, LGSS is based on least squares estimation, but differently from standard regression, metabolic approximation in LGSS has the form of a grammar imposed by the phenomenon under investigation.

Moreover, the search for the best regulators of the given reactions is performed by using suitable forms of F-tests (based on Fisher distribution).

Let us consider the algebraic formulation of the dynamical inverse problem, given by Eq. (3.23), for determining the vector $\mathbb{C}$ of regressor coefficients providing the best regulators approximating a given dynamics:

$$\mathbb{G} \times \mathbb{C} \times \mathbb{A}^T = \mathbb{D} \tag{3.33}$$

which is equivalent to (see Eq. (3.29)):

$$(\mathbb{A} \otimes \mathbb{G}) \times vec(\mathbb{C}) = vec(\mathbb{D}). \tag{3.34}$$

Let us assume that the rank of the matrix associated to the above system is maximum. This happens when the stoichiometric matrix has maximum rank and the expanded regressors are linear independent. The system above can be considered as a multiple regression model, where $vec(\mathbb{D})$ is the (vector) dependent variable, while the independent (vector) variables are the columns of matrix $\mathbb{G}$. These assumptions make it possible to deal with system 3.34 as a multiple regression model, in the sense defined in Eq. 3.32.

For a better understanding of the following discussion we reformulate in Table 3.13 the deviations occurring in regression models in terms of the matrix $\mathbb{D}$, where Δ_h indicates variation vector of the substance of index h. We also extend the definition of MSE, R^2 and $\bar{R}^2$ in order to consider separately each substance of the system.

$$MSE_h = \frac{SSE_h}{t - d_h} = \frac{\sum_{i=0}^{t} (\Delta_h[i] - \widehat{\Delta}_h[i])^2}{t - d_h} \tag{3.35}$$

$$R_h^2 = \frac{SSR_h}{SST_h} = 1 - \frac{SSE_h}{SST_h} \tag{3.36}$$

$$\bar{R}_h^2 = 1 - \frac{SSE_h/[t - d_h]}{SST_h/t} = 1 - \frac{MSE_h}{MST_h}. \tag{3.37}$$

Table 3.13 The three deviations of an LGSS regression for each substance of the system. We indicate with h the index of the substance, with $\widehat{\Delta}_h^t$ the predicted value of Δ_h^t by means of the multiple regression model, and with $\bar{\Delta}_h^t$ the average of the values of Δ_h^t.

$\Delta_h^t - \bar{\Delta}_h^t$	=	$\Delta_h^t - \widehat{\Delta}_h^t$	+	$\widehat{\Delta}_h^t - \bar{\Delta}_h^t$
Total deviation for substance h		Unexplained deviation (error) for substance h		Explained deviation (regression) for substance h
$\sum_{i=0}^{t} (\Delta_h[i] - \bar{\Delta}_h^t)^2$	=	$\sum_{i=0}^{t} (\Delta_h[i] - \widehat{\Delta}_h[i])^2$	+	$\sum_{i=0}^{t} (\widehat{\Delta}_h[i] - \bar{\Delta}_h^t)^2$
SST_h		SSE_h		SSR_h
Sum of Squared Total deviations for substance h		Sum of Squared Errors for substance h		Sum of Squared Regressions deviations for substance h

The number of degrees of freedom is calculated here by considering that in Eq. (3.29) the total number of equations for each substance is equal to t and the number of regressors used for each substance is given by:

$$d_h = \left| \bigcup_{r_l \in R^o(h)} \left\{ g_j \mid \varphi_l = \sum_{j=1}^{d} c_{l,j} g_j \right\} \right| \tag{3.38}$$

where φ_l is the regulator of the reaction r_l and $R^o(h)$ indicates the reactions of the system consuming or producing the substance of index h (see Table 3.7). Following the same reasoning, we can also extend the notion of F-ratio:

$$F_h = \frac{SSR_h/d_h}{SSE_h/[t-d_h]} = \frac{MSR_h}{MSE_h} = \frac{R_h^2}{1-R_h^2} \cdot \frac{t-d_h}{d_h} \tag{3.39}$$

After having fixed a significance value α for the test, we can conclude that a linear relationship exists if (see F-distribution in Sect. 7.7):

$$F_h > F_{[\alpha;d_h,t-d_h]}.$$

The evaluation of the confidence intervals for the least squares estimation of each regressor can be done by computing, in the t-expansion G_j^t of regressor g_j, the right number of degrees of freedom of the t-distribution. It depends on the ADA stoichiometric expansion applied to the considered regressors. In fact, we recall from the k-Variable Multiple Regression Model that the number of the degrees of freedom is given by $n-(k+1)$, where n is the number of equations and $k+1$ is the number of regressor coefficients. Now, the role of n is played by the number of equations involved in the regressor G_j^t under consideration, which is given by (see Table 3.7):

$$n' = t \cdot |S^o(r_l)|.$$

The role of $k+1$, instead, is given by:

$$k' = \sum_{h \in S^o(r_l)} d_h$$

that is, the sum of all the regressors included in the model which modify at least one substance in $S^o(r_l)$. Finally, the formula that gives the confidence intervals for the coefficients computed in Eq. (3.29), is given by:

$$\beta_{l,j} = c_{l,j} \pm t_{[\alpha/2;n'-k']} \sqrt{e \cdot MSE_{S^o(r_l)}} \tag{3.40}$$

where $l = 1,2,\ldots,m$ and $j = 1,2,\ldots,d$ are the indexes related to reactions and regressors, respectively, e is the element related to the regressor under examination on the first diagonal of the matrix $\left((\mathbb{A} \otimes \mathbb{G})^T \times (\mathbb{A} \otimes \mathbb{G})\right)^{-1}$ used for the least squares estimation of Eq. (3.30), and:

$$SSE_{S^o(r_l)} = \sum_{h \in S^o(r_l)} SSE_h \tag{3.41}$$

$$MSE_{S^o(r_l)} = \frac{SSE_{S^o(r_l)}}{n' - k'}. \tag{3.42}$$

The same observations made for computing the degrees of freedom in Eq. (3.40) must to be applied to the formula of the partial F-test. When we want to test the statistical significance of an expanded regressor in LGSS, we apply the following partial F-test:

$$F_{[1,n'-k']} = \frac{SSE^R_{S^o(r_l)} - SSE^F_{S^o(r_l)}}{MSE^F_{S^o(r_l)}} \tag{3.43}$$

where r_l is the rule to which the regressor g_j is added (G^t_j is its expansion), $S^o(r_l)$ indicates the set of substances produced or consumed by r_l, $SSR^R_{S^o(r_l)}$ is the sum of squares for error of the reduced model (i.e. the model without the regressor under examination), $SSE^F_{S^o(r_l)}$ is the sum of squares for error of the full model (i.e. the model with the regressor under examination), and $MSE^F_{S^o(r_l)}$ is the mean square error of the full model.

After fixing a significance value α for the test, we can conclude that the full model is statistically better than the reduced one (i.e. that the regressor under examination should be included in the model) when the value of the partial F-test is greater than the threshold value:

$$F_{[\alpha;1,n'-k']}. \tag{3.44}$$

The partial F-statistics given in Eq. (3.43) makes possible the application of the stepwise regression in LGSS, because it allows us to select the right set of regressors that should be used in ADA expansion. We will test only a subset of the expanded regressors, that is, the ones which have log-gain scores higher than a fixed theshold (see Sect. 3.4.1). When the stepwise algorithm stops, LGSS saves the computed multiple regression model and then tries to modify it by running again the stepwise regression algorithm with a larger set of expanded regressors. We repeat this phase until there are no regressors that can be considered. The final regression model will be the one, among those saved, which will provide the best approximation. The estimation of the approximation of models is calculated by means of Eqs. (3.35), (3.36), (3.37), (3.39), and (3.40).

The least squares estimation used in LGSS gives the best approximations with respect to the "observed steps", while when the dynamics is produced by means of the regulators, substance variations are computed by means of the previous "computed step". For this reason the dynamics generated by the MP system provided by LGSS can be improved by a *tuning process* which systematically searches in small neighborhoods of the estimated coefficients, inside the range of the confidence intervals computed by means of Eq. (3.40). In this way, usually, values providing an MP dynamics have a significant improvement in the approximation of the observed dynamics.

LGSS Algorithm

1. Let t be the number of observation time points. Let α_1, α_2 be two user-defined significance values for partial F-tests. Let $\mathbb{S} = \emptyset$, the set of *saved models*. Let us identify any t-expanded matrix with the set of vectors constituted by its columns;
2. Compute the matrices $\mathbb{D}, \mathbb{G}$ of the t-expansions of substance differences and of regressors, respectively;
3. Compute the log-gain scores for all the (expanded) regressors of $\mathbb{G}$, and sort them accordingly in decreasing order. Let $\mathbb{G}'$ be the subset/submatrix of regressors of $\mathbb{G}$ having the best log-gain scores (according to a prefixed user-defined threshold);
4. Let $\mathbb{M}$ be a user-defined initial subset/submatrix of $\mathbb{G}'$, and compute the following LSE of unknown values $vec(\mathbb{C})$:

$$(\mathbb{A} \otimes \mathbb{M}) \times vec(\mathbb{C}) = vec(\mathbb{D}) \tag{3.45}$$

5. For each expanded regressor $g^t \in \mathbb{G}'/\mathbb{M}$, compute the LSE of Eq. (3.45), where $\mathbb{M} = \mathbb{M} \cup \{g^t\}$, then compute the partial F-test of the extended model $\mathbb{M} \cup \{g^t\}$ with respect to the current model $\mathbb{M}$, according to Eq.(3.43);
6. Is there some model, among those computed at the previous step, having a partial F-test value higher than the value of (3.44) where $\alpha = \alpha_1$? If No, then go to step 11;
7. Update $\mathbb{M}$ as the model, among those of step 5, having the highest value of partial F-test;
8. For each regressor $g^t \in \mathbb{M}$, compute the LSE of Eq. (3.45), where $\mathbb{M} = \mathbb{M}/\{g^t\}$, and then compute the partial F-test of the current model $\mathbb{M}$ with respect to the reduced model $\mathbb{M}/\{g^t\}$, according to Eq.(3.43);
9. Is there some model, among those computed at the previous step, having a F-test value lower than the value of (3.44) where $\alpha = \alpha_2$? If No, then go to step 5;
10. Update $\mathbb{M}$ as the model, among those of step 8, having the lowest value of partial F-test, then go to step 5;
11. Add the current model $\mathbb{M}$ to the set $\mathbb{S}$ of saved models;
12. Is $\mathbb{G}' = \mathbb{G}$? If No, then update $\mathbb{G}'$ as $\mathbb{G}' \cup \{g^t\}$, where g^t is the expanded regressor among those in $\mathbb{G}/\mathbb{G}'$ having the highest log-gain score, and go to step 5;
13. Among the models of $\mathbb{S}$ select the best one, by using the indexes of Eqs. (3.35), (3.36), (3.37), (3.39);
14. Provide as output the model selected at the previous step with its evaluation of parameters and confidence intervals of regressors coefficients.

3.4.3 Problems Related to the Regression with LGSS

The size of the systems of equations solved by LGSS ranges from about a thousand equations to some hundreds of thousands, depending on the number of substances and reactions of the MP system under examination and on its time interval (a smaller time interval requires a longer time series and so a larger system of equations)[3]. The size of regressor dictionary depends on the complexity of the phenomenon under investigation, but usually it comprises no more than one hundred regressors. The total computation time for a regression depends on the size of the regressor dictionary and on the number of equations to be solved. However, the computation usually ends in few minutes (less than five minutes on average, using a common laptop with a single dual core CPU and 4Gbyte of RAM memory), but it can increase to hours when the system is very big (i.e. a system with many hundreds of thousands of equations, and a regression dictionary of hundreds of regressors).

As explained in Sect. 7.7.1, the correctness of a multiple regression model is based on some assumptions about the independent variables, and about the probability distribution of the errors associated to observations. When one or more of these assumptions are not completely satisfied, some mistakes may occur in the definition of the regression model. There are three main problems that we need to be aware of in the context of multiple regression: (i) the problem of *heteroscedasticity*, (ii) the problem of *residual autocorrelation*, and, the most common in LGSS, (iii) the problem of *multicollinearity* [76, 216].

The problem of **heteroscedasticity** occurs when the variance of the residuals of the regression is not constant and is (directly or indirectly) proportional to the value of one or more independent variables of the model. This is in contrast with the assumption of uniform error variance, carried out at the beginning of Sect. 7.7.1. When hereroscedasticity is present, our regression coefficient estimators are not efficient. This violation of the regression assumptions may sometimes be corrected by the use of a transformation for the dependent variable Y or by substituting the ordinary least squares estimation method with the method of *weighted least squares*.

The problem of **residual autocorrelation** arises when the error ε depends on the observation points. It is also called residual autocorrelation, because it occurs when the time series of the error values is highly correlated with the values of the series at certain previous steps [229].

As in the case of heteroscedasticity, in this case the ordinary least squares may fail, therefore it can be useful to adopt another procedure called *generalized least squares*.

The most common problem related to regression in LGSS is the problem of **multicollinearity**. In multiple regression, we hope to have a strong correlation between each independent variable and the dependent variable Y, but we do not want to have independent variables correlated among them. In case of perfect collinearity, the regression algorithm breaks down completely. Since in LGSS regulators usually are

[3] An implementation of LGSS, as a set of MATLAB functions, has been developed by Luca Marchetti in 2012 [108].

assumed to be linear combinations of polynomial regressors, it is very common to meet multicollinearity problems.

In order to overcome the problem, the stepwise algorithm implemented in LGSS was extended by filtering the expanded regressors, during the forward selection phase, in order to avoid the insertion of a regressor which is highly correlated with others already inserted in the regression model. To this end, LGSS computes the *variance inflation factor* (VIF) for each regressor, which gives an idea of the degree of multicollinearity it introduces with respect to the other regressors in the regression equation. To calculate the VIF for a regressor g, we need to run a multiple regression by considering g as the dependent variable and the set of already inserted regressors as the set of independent variables. The variance inflation factor associated with g is:

$$VIF(g) = \frac{1}{1 - R_g^2} \tag{3.46}$$

where R_g^2 is the coefficient of determination for the multiple regression of g with respect to the other regressors.

A VIF of 6, for example, means that the variance of the regression coefficient estimator for the considered regressor is 6 times what it should be when no collinearity exists. In LGSS the user can select a threshold value for the variance inflation factor in order to avoid the insertion of collinear regressors. This solution, however, may affect the performance of the algorithm since it requires many additional computations.

Of course, a way to overcome the problem of multicollinearity is to drop collinear variables before launching LGSS. We extended LGSS by including a procedure, based on a hierarchical clustering technique [86, 104], which allows us to cluster the time series of the regressors associated to the same reaction and to select those which are less correlated and that best satisfy the log-gain principle. In many cases this allows us to significantly reduce the set of regressors and provides a better approximation.

3.4.4 Models of Mitosis

LGSS represents a solution, in terms of MP systems, of the inverse dynamics problem, that is, of the identification of (discrete) mathematical models exhibiting an observed dynamics and satisfying all the constraints required by the specific knowledge about the modeled phenomenon.

In Table 3.14 a short list of models obtained by LGSS is presented. In this section we outline an application of LGSS to Golbeter's oscillator given in Fig. 3.8 [81, 82, 83]. LGSS provided 700 different models of this oscillator, which, for the most part, provide the same order of approximation of Golbeter's model. Moreover, by considering the phenomenon at different timescales, we obtained different models and in many cases the analytical form of these models is simpler than Golbeter's model [101].

Table 3.14 MP models obtained by LGSS

Belousov-Zhabotinsky, Prigogine's Brusselator (BZ)	Brusselatus ([84, 68], Table 3.9)
Lotka-Volterra, Predator-Prey dynamics (LV)	Volteranus ([89, 116, 78], Table 3.8)
Susceptible-Infected-Recovered Epidemics (SIR)	Epidemicus ([88, 68])
Early Amphybian Mitotic Cycle (AMC)	Mitoticus ([81, 82, 101], Table 3.16)
Drosophila Circadian Rythms (DCR)	Drosophilus ([68])
Non Photochemical Quenching in Photosynthesis (NPQ)	Photochemicus ([110, 107])
Minimal Diabetes Mellitus (MDM)	Mellitus ([105])
Bi-catalytic Synthetic Oscillator	Sirius ([92], Table 3.5)
Circular Synthetic Oscillator	Goniometricus ([100], Table 3.11)

The fundamental mechanism of mitotic oscillations concerns the periodic change in the activation state of a protein produced by *cdc2* gene in fission yeast or by homologous genes in other eukaryotes. The simplest form of this mechanism is found in early amphibian embryos (see [83]). Here cyclin (C) is synthesized at a constant rate and triggers the transformation of inactive (M^+) into active (M) *cdc2* protein, which leads to the formation of a complex known as M-phase promoting factor (*MPF*). MPF triggers mitosis, but at the same time M elicits the activation of a protease from state X^+ to X. The active protease degrades cyclin resulting in the inactivation of *cdc*2. This brings the cell back to initial conditions and a new division cycle can take place. The ODE presented on the right of Fig. 3.8 is the differential model of dynamics described on the left of Fig. 3.7, where C, M, X are the concentrations of C, M, X, respectively, and $1-M, 1-X$ are the concentrations of M^+, X^+ respectively (the definitions of the parameters of the ODE model of Fig. 3.8 are not simple and are not relevant for our further discussion, however they can be found in [81]). The regulation maps calculated by LGSS are obtained starting from a dictionary of 20 possible *regressors*, that is monomials of C, M, and X with degree less than or equal to 3 (i.e. constants, C, M, X, C^2, M^2, X^2, CM, CX, MX, C^3, M^3, X^3, C^2M, CM^2, C^2X, CX^2, M^2X, MX^2, and CMX)[4].

The 700 MP models that LGSS found for Goldbeter's mitotic oscillator are distributed into 40 different grammatical schemata. In Table 3.15 other descriptional indexes of models are given for the first 14 grammatical schemata which define 621 models from a total of 700 (89%). These indexes are useful for discriminating interesting aspects of the MP grammars and they comprehend:

1. the *number of regressors*;
2. the *total number of monomials*;
3. the *temporal grain* of dynamics observation, which is expressed by the values of time interval τ;

[4] Substances M^+ and X^+ are not considered because they depend on M and X respectively.

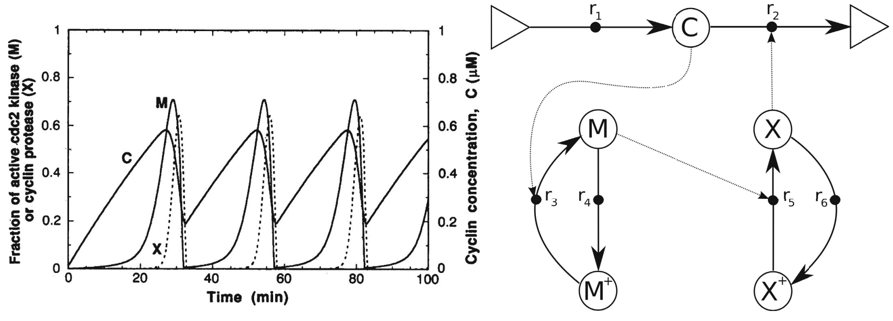
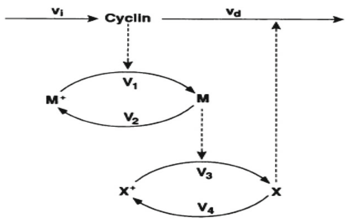


Fig. 3.7 On the left: a numerical solution of the set of differential equations (right part of Fig. 3.8) comprising the model introduced by A. Goldbeter (figure taken from [81]). On the right: an MP graph which represents Golbeter's oscillator



$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d+C} - k_d C$$
$$\frac{dM}{dt} = V_{M1} \frac{C}{K_c+C} \frac{(1-M)}{K_1+(1-M)} - V_2 \frac{M}{K_2+M}$$
$$\frac{dX}{dt} = M V_{M3} \frac{(1-X)}{K_3+(1-X)} - V_4 \frac{X}{K_4+X}$$

Fig. 3.8 Goldbeter's oscillator, which has a cycle of about 25 min [81]

4. the *best value of* τ which is relative to the model, which provides the best dynamical approximation of the mitotic phenomenon;
5. the best *RMSE* which is the average value of the RMSE relative to the substances curves corresponding to the best τ.

It is worthwhile remarking that the grammatical schemata occurring at the first positions (with high frequency) are also the grammatical schemata having a small number of regressors and monomials. In Table 3.16 a parsimonious MP model of mitotic oscillator is given.

3.5 Reactivity and Inertia

The notion of MP grammar is based on regulators, which define the value of reaction fluxes at a given state. However, a different approach for defining fluxes can be followed, where fluxes are obtained as results of a competition among the reactions. In order to formalize this intuition, a **reactivity parameter** is associated to every reaction, as a score in the competition for getting the reactants necessary to its realization. This competition, among reactions consuming a given substance, concerns a part of matter available in a given state, therefore another parameter is necessary for each substance, which provides the amount of substance that, in a given state,

Table 3.15 Descriptional indices of models given for the first 14 grammatical schemata

Grammatical schemata	Number of models	Number of regressors	Total n. of monomials	τ interval (10^{-3} *min*)	Best τ (10^{-3} *min*)	RMSE
1	135	6	16	151 – 345	315	$1.61 \cdot 10^{-2}$
2	128	6	17	343 – 477	401	$1.62 \cdot 10^{-2}$
3	49	6	17	43 – 93	43	$1.84 \cdot 10^{-2}$
4	46	6	16	138 – 232	219	$1.95 \cdot 10^{-2}$
5	44	8	24	1 – 71	40	$1.48 \cdot 10^{-2}$
6	38	6	16	525 – 699	683	$1.78 \cdot 10^{-2}$
7	33	6	16	473 – 563	556	$1.79 \cdot 10^{-2}$
8	32	5	15	514 – 694	602	$2.78 \cdot 10^{-2}$
9	28	7	16	570 – 696	671	$1.09 \cdot 10^{-2}$
10	26	6	16	493 – 684	684	$1.8 \cdot 10^{-2}$
11	20	8	23	118 – 137	137	$5.86 \cdot 10^{-2}$
12	15	9	25	103 – 117	103	$9.6 \cdot 10^{-3}$
13	15	6	17	474 – 499	474	$1.62 \cdot 10^{-2}$
14	12	7	21	191 – 212	212	$1.97 \cdot 10^{-2}$

cannot be transformed. We call it the *inertia* of the substance (in a given state). The difference between the whole quantity of a substance and its inertia is the quantity of substance that can be partitioned among all reactions competing for it. MP systems with reactivity parameters were the first kind of MP systems formally defined [92]. They are a special class of MP systems with regulators, which we call *reactive* MP systems or MPR systems. When we want to refer to the MP systems defined in Definition 3.1, we refer to them as MPF systems for distinguishing them from MPR systems (MP will mean one of the two systems, MPF or MPR, according to the context). In an MPR system, we denote by ψ_x the parameter providing the *inertia* of substance x, and by f_r $(r \in R)$ which provides the *reactivity* of the reaction r. Let us set by $R^-(x)$ the set of reactions consuming the substance x, by $S^-(r)$ the reactants of reaction r, by $|r^-(x)|$ the multiplicity of the reactant x in the rule r, and by s any state of the system. Then, another parameter $p_{r,x}$ of (metabolic) *reactance* of the rule r, with respect to the substance x, can be associated to any substance $x \in S^-(r)$:

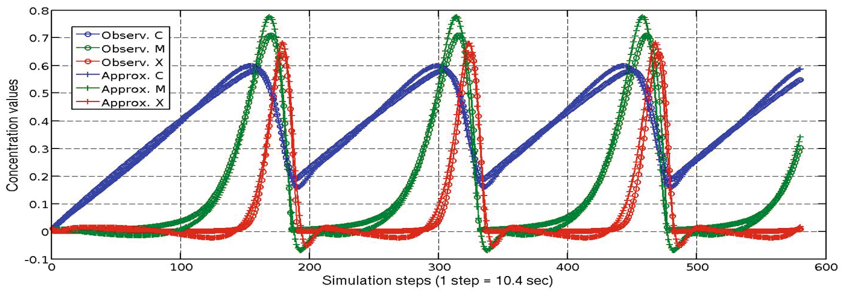
$$p_{r,x}(s) = \frac{x/|r^-(x)|}{\psi_x(s) + \sum_{r' \in R^-(x)} f_{r'}(s)}. \tag{3.47}$$

Let *min A* be the minimum over a finite set A of numbers, conventionally extended to the empty set by $\min \emptyset = 1$. With these notations, the flux regulation maps of a reactive MP system are given by:

$$\varphi_r(s) = \min\{p_{r,x}(s) \mid x \in S^-(r)\} \cdot f_r(s). \tag{3.48}$$

Table 3.16 The MP mitotic oscillator with the minimum total number of monomials ($\tau = 173 \cdot 10^{-3}$ min, $RMSE \approx 2.67 \cdot 10^{-2}$). Constants and initial values: $v_i = 0.025$, $k_1 = 0.0209$, $k_2 = 0.0149329$, $k_3 = 0.0351323$, $k_4 = 0.0200062$, $k_5 = 0.000662743$, $k_6 = 0.215816$, $k_7 = 0.0696881$, $k_8 = 0.0911799$, $k_9 = 0.166106$, $k_{10} = 0.569463$, $k_{11} = 0.00823672$, $k_{12} = 0.252676$, $k_{13} = 0.404647$, $k_{14} = 0.668527$, $C[0] = M[0] = X[0] = 0.01$, $M^+[0] = X^+[0] = 0.99$.

$r_1 : \emptyset \to C$	$\varphi_1 = v_i$
$r_2 : C \to \emptyset$	$\varphi_2 = k_1 + k_2\, M + k_3\, X - k_4\, CM$
$r_3 : M^+ \to M$	$\varphi_3 = k_5 + k_6\, CM$
$r_4 : M \to M^+$	$\varphi_4 = k_7\, M + k_8\, X$
$r_5 : X^+ \to X$	$\varphi_5 = k_9\, C + k_{10}\, M$
$r_6 : X \to X^+$	$\varphi_6 = k_{11} + k_{12}\, X + k_{13}\, C^2 + k_{14}\, CM$



Two MP systems are **dynamically equivalent** when they have the same substances and parameters, the same parameter evolution functions, the same scale factors ν, μ, τ, and, starting from the same initial state, provide the same dynamics.

Any MPR system is a special case of an MPF system, but we will show that, for some MPF systems, equivalent MPR systems exist. Lemma 3.3 and Theorem 3.4 explain the equivalence between MP systems based on flux functions and MP system based on reactivity and inertia parameters. In this lemma, and in the related theorem, we consider the notion of **non-cooperative** MP rule. Table 3.17 considers this and other properties of general interest for an MP rule r given by the multiset rewriting $\alpha \to \beta$.

An MP system is **non-cooperative** if each rule r defined in the system is non-cooperative:

$$|S^-(r)| \leq 1.$$

Lemma 3.3. *For any MPF system there exists a non-cooperative MPF system which is dynamically equivalent to it.*

Proof. If a rule, for example $r : a + c \to b$, has more than one reactant, then we can split it into two rules: $r_1 : a \to b$, $r_2 : c \to \emptyset$ by requiring that the fluxes φ_1, φ_2 of the two rules are equal to the flux φ_r of r. We can proceed in this way for all the rules which are cooperative. The non-cooperative MP system which we get in this manner is dynamically equivalent to the original one. □

Table 3.17 Types of an MP rule r of form $\alpha \to \beta$

monic	$\lvert\alpha\rvert = 1$
monogenic	$\lvert S^-(r)\rvert = 1$
non-cooperative	$\lvert S^-(r)\rvert \leq 1$
cooperative	$\lvert S^-(r)\rvert > 1$
synthetic	$\lvert S^-(r)\rvert > \lvert S^+(r)\rvert$
transformative	$\lvert S^-(r)\rvert = \lvert S^+(r)\rvert$
dissociative	$\lvert S^-(r)\rvert < \lvert S^+(r)\rvert$
assimilative	$\alpha = \emptyset$
dispersive	$\beta = \emptyset$
catalytic	$S^-(r) \cap S^+(r) \neq \emptyset$

An MP system is **positive** if, in any state s, reaction fluxes do not consume more matter than the amount available (for any reaction r and any substance x):

$$\sum_{r \in R^-(x)} \varphi_r(s) \leq x$$

Theorem 3.4. *For any positive MPF system there exists a dynamically equivalent MPR system.*

Proof. According to Lemma 3.3, we can start by considering a non-cooperative MPF system $\mathbb{M}$. Now we transform $\mathbb{M}$ into an MPR system $\mathbb{M}'$ which will result dynamically equivalent to $\mathbb{M}$. The system $\mathbb{M}'$ is given by the same substances, rules, parameters, ν, μ, τ of M. Moreover, for any flux φ_r of $\mathbb{M}$, we define the corresponding reaction map f_r of $\mathbb{M}'$ as

$$f_r(s) = \frac{|r^-(x)|}{x} \cdot \varphi_r(s)$$

where $|r^-(x)|$ gives the multiplicity of the reactant x of r (when $S^-(r) = \emptyset$, we consider $\frac{|r^-(x)|}{x} = 1$). Finally, for each substance x, we define its inertia function ψ_x as

$$\psi_x(s) = 1 - \sum_{r \in R^-(x)} f_r(s).$$

The MPR system $\mathbb{M}'$ equipped with the reaction maps and with the inertia functions defined above is dynamically equivalent to the MPF system $\mathbb{M}$ we started from. In fact, for any rule r of $\mathbb{M}'$, the reactance $p_{r,x}(s)$ is given by:

$$p_{r,x}(s) = \frac{x/|r^-(x)|}{\psi_x(s) + \sum_{r' \in R^-(x)} f_{r'}(s)}$$

and so the corresponding flux $\varphi_r'(s)$ of $\mathbb{M}'$ is

$$\varphi_r'(s) = \min\{p_{r,y}(s) \mid y \in S^-(r)\} \cdot f_r(s).$$

Now, if $S^-(r) = \emptyset$, then

$$\begin{aligned}\varphi_r'(s) &= \min\{p_{r,y}(s) \mid y \in S^-(r)\} \cdot f_r(s)\\ &= \min\, \emptyset \cdot f_r(s) = f_r(s) = \varphi_r(s).\end{aligned}$$

Otherwise, if $S^-(r) \neq \emptyset$, since the system $\mathbb{M}$ is supposed to be non-cooperative, then $|S^-(r)| = 1$ and, having $S^-(r) = \{x\}$, we can set:

$$\begin{aligned}\varphi_r'(s) &= \min\{p_{r,y}(s) \mid y \in S^-(r)\} \cdot f_r(s)\\ &= \frac{x/|r^-(x)|}{\psi_x(s) + \sum_{r' \in R^-(x)} f_{r'}(s)} \cdot f_r(s)\\ &= \frac{x/|r^-(x)|}{\left(1 - \sum_{r' \in R^-(x)} f_{r'}(s)\right) + \sum_{r' \in R^-(x)} f_{r'}(s)} \cdot f_r(s)\\ &= \frac{x}{|r^-(x)|} \cdot f_r(s) = \frac{x}{|r^-(x)|} \cdot \frac{|r^-(x)|}{x} \cdot \varphi_r(s) = \varphi_r(s).\end{aligned}$$

Therefore, in any case we have

$$\varphi_r'(s) = \varphi_r(s).$$

In conclusion, the fluxes of the MPR system $\mathbb{M}'$ are the same as the MPF system $\mathbb{M}$, whence the two systems result equivalent. □

The MP graph of Fig. 3.9 with its corresponding MP grammars (MPF and MPR) of Table 3.18 provides, in N, the famous Fibonacci's sequence when we start with one unit of A (adult) and zero unit of N (newborn).

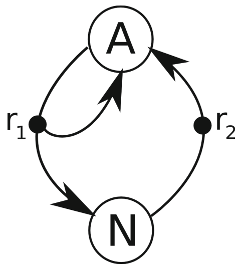


Fig. 3.9 The MP graph related to Fibonacci's MP grammars of Table 3.18

Other examples of MPR and equivalent MPF systems are given in Tables 3.19, 3.20, and 3.21.

Table 3.18 MPF and MPR grammars providing Fibonacci's sequence

Rules	Fluxes	Reaction maps	Inertia
$r_1 : A \to A+N$	$\varphi_1 = A$	$f_1 = 1$	$\psi_A = 0$
$r_2 : N \to A$	$\varphi_2 = N$	$f_2 = 1$	$\psi_N = 0$

Table 3.19 An MPR system providing the dynamics given in Fig. 3.3. All inertias are 100, the initial values are 100 for A, B, and 0.02 for C.

$r_1 : A \to 2A$	$f_1 = 10$
$r_2 : A \to B$	$f_2 = 0.02C$
$r_3 : A \to C$	$f_3 = 0.02B$
$r_4 : B \to \emptyset$	$f_4 = 4$
$r_5 : C \to \emptyset$	$f_5 = 4$

Table 3.20 An MPR dynamically equivalent to the system given in Table 3.19 (the first reaction of duplication becomes here an input reaction). The inertia of A is 110, while inertias of B and C are 100, the initial values are 100 for A, B, and 0.02 for C.

$r_1 : \emptyset \to A$	$f_1 = 10A/(0.02B + 0.02C + 110)$
$r_2 : A \to B$	$f_2 = 0.02C$
$r_3 : A \to C$	$f_3 = 0.02B$
$r_4 : B \to \emptyset$	$f_4 = 4$
$r_5 : C \to \emptyset$	$f_5 = 4$

Table 3.21 The MPF of the oscillator associated to the MPR of Table 3.20

$r_1 : \emptyset \to A$	$\varphi_1 = 10A/(0.02B + 0.02C + 110)$
$r_2 : A \to B$	$\varphi_2 = 0.02AC/(0.02B + 0.02C + 110)$
$r_3 : A \to C$	$\varphi_3 = 0.02AB/(0.02B + 0.02C + 110)$
$r_4 : B \to \emptyset$	$\varphi_4 = 4B/(4B + 100)$
$r_5 : C \to \emptyset$	$\varphi_5 = 4C/(4C + 100)$

3.6 Metabolic Patterns

In this section we present some basic cases of metabolic phenomena. Their combinations of reactions and regulations provide the majority of typical mechanisms arising in metabolic networks, by means of **cascades**, **antagonisms**, **feedbacks**, and **delays**. The simplest mechanism of composing reactions is their sequential composition, according to which the product of a first reaction is the reactant of a second one. A cascade is essentially given by a sequential composition of many reactions (see Fig. 3.10). The **source** of a cascade is the reactant of the first reaction, while its **destination** is the product of the last reaction of its sequential composition. A cascade can be either perfectly linear, or sharing substances (reactants or products) with other reactions.

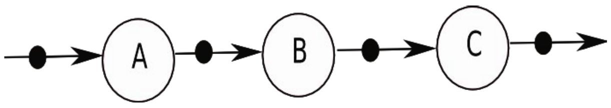


Fig. 3.10 Sequencing

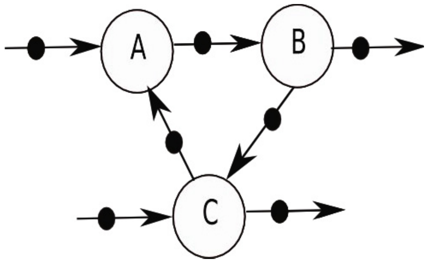


Fig. 3.11 Cycling

A (transformation) cycle is a sequential composition where the destination coincides with the source of the cascade.

Coupling reactions means that their fluxes are correlated. This does not mean that the two reactions can be considered as a single reaction. In fact, in general their fluxes can be different, but for example, the flux of one reaction could be the square of the flux of another one.

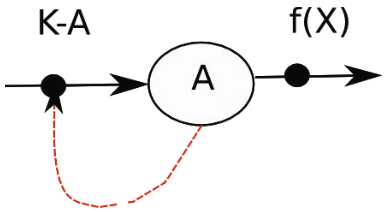


Fig. 3.12 A kind of homeostasis

Homeostasis can occur in many forms. The example given in Fig. 3.12 shows a simple case of homeostasis where the amount of a given substance is automatically kept to a fixed value K (if a quantity $f(X)$ is consumed, then the difference with respect to the fixed value K is introduced from outside).

The next examples of metabolic patterns are related to the regulation aspect of reactions. The simplest form of regulation composition is shown in Fig. 3.13, which can be seen as a regulation cascade. In the case of a regulation cascade we can speak of a **source reaction** and of a **destination reaction**.

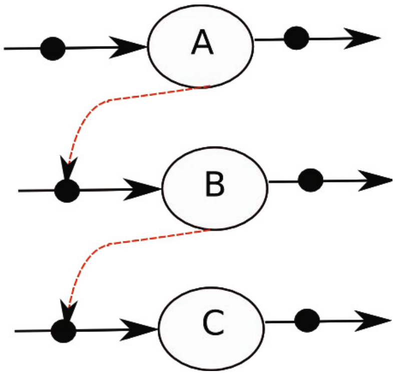


Fig. 3.13 A regulation chain

A regulation cascade where the source reaction coincides with the destination reaction is a regulation cycle (see Fig. 3.14). In this case we may have a lot of possibilities in dependence of the kind of substances (reactants or products) regulating the reactions. An important aspect of a regulator is given by its direct or inverse character. In the first case, the flux is directly dependent on the value of the regulating substance. In the second case, it is inversely dependent on it. Two cases of regulation cycles are represented in Fig. 3.14. Biregulation, represented in Fig. 3.15, is another recurrent schema of regulation mechanism.

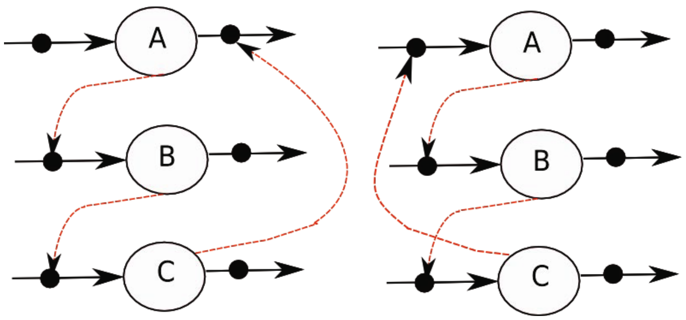


Fig. 3.14 Two regulation cycles

In the case of **antiregulation**, given in Fig. 3.16, one substance regulates the production of another substance, which in turn, regulated the consumption of the first one.

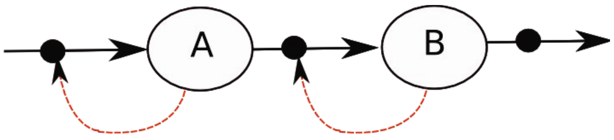


Fig. 3.15 Biregulation

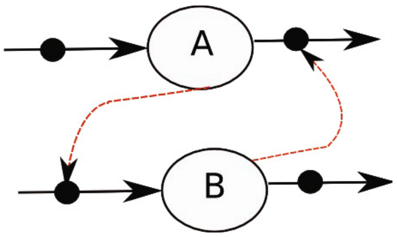


Fig. 3.16 Antiregulation

Mechanisms of **cisregulation** and **transregulation** (Figs. 3.17, 3.18) concern backward and forward relationships, respectively, between regulation and reaction composition.

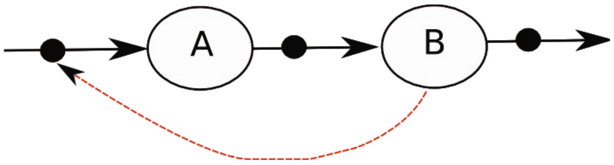


Fig. 3.17 Cisregulation

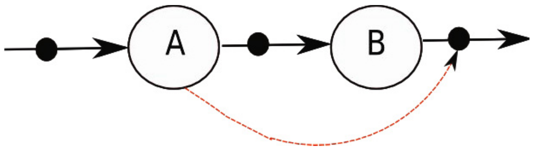


Fig. 3.18 Transregulation

Self-regulation is a basic mechanism, similar but different from autocatalysis, where a substance promotes (or inhibits) its production, or its consumption. For example, in almost all growth phenomena the bigger is the population, the faster is its growth rate. For example, a commercial product is sold in a bigger quantity when there is a greater production of it (making it cheaper), and analogously, the more it is sold, the more it is produced. Negative self-regulation occurs when a substance provides an inhibition of its production or an inhibition of its consumption.

3.7 Metabolic Oscillators

Metabolic oscillators are ubiquitous in life phenomena, and life in itself is an oscillatory phenomenon reproducing and propagating in space. Many of the metabolic patterns we analyzed in the previous section provide, in a simple manner, oscillatory behaviors. In this section we present some examples of basic metabolic phenomena. In almost all cases the possibility of dynamical curves exhibiting values which range in a given interval, without diverging and without terminating, is correlated to the structure of the MP graphs involving cycles of reactions and regulations. The oscillators which we present are expressed as MPR (systems with reactivity maps). In this formulation they assume a simpler form and are also easier to be discovered. For any oscillator, its MPR graph (where reaction maps are indicated) is given together with its dynamics (where inertias and initial values are indicated). For the sake of simplicity, we maintain the terminology of the previous section (cis-regulation, trans-regulation, and so on), even if regulators are replaced by reaction maps, therefore their meanings are somewhat different from the regulation mechanisms analyzed in the previous section. Dynamics are computed by means of software simulating MPF systems (Psim, specifically developed in years 2007-2009 [67, 69, 71]). In many cases, oscillations are very robust in dependence of the initial values and of the algebraic form of regulations, while in other cases systems prove to be very sensitive and only small changes in the third or fourth decimal digit alters dramatically the dynamical pattern. Figures 3.19, 3.20, 3.21, and 3.22 provide some first examples of interesting metabolic oscillators with their dynamics.

The names of many oscillators derive from names of stars. This fact is due to many reasons. Namely, stars were the first objects of systematical dynamical investigation (and mathematical formalization). Moreover, the dynamical system Sirius, the first defined by an MP grammar, exhibits a behavior resembling the double star Sirius, where the movement of one star is hidden by the other, because in its original formulation [92], one substance is transformed in the other two, but the curve of values (along time) of one substance is quite almost completely hidden by the curve of the other substance (see Fig. 3.3).

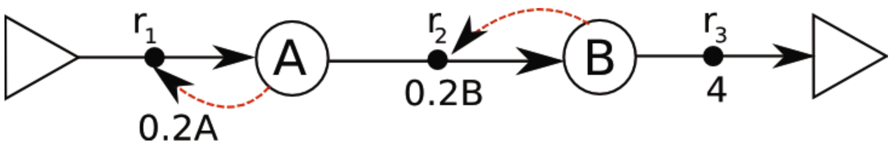


Fig. 3.19 *Bicatalyticus*, a very simple MPF oscillator

The metabolic oscillator of Fig. 3.23 is based on a transformation cycle. Its initial behavior follows a very irregular pattern, however, on a long run, it seems to reach a regular dynamical pattern. On the contrary, the metabolic oscillator of Fig. 3.24

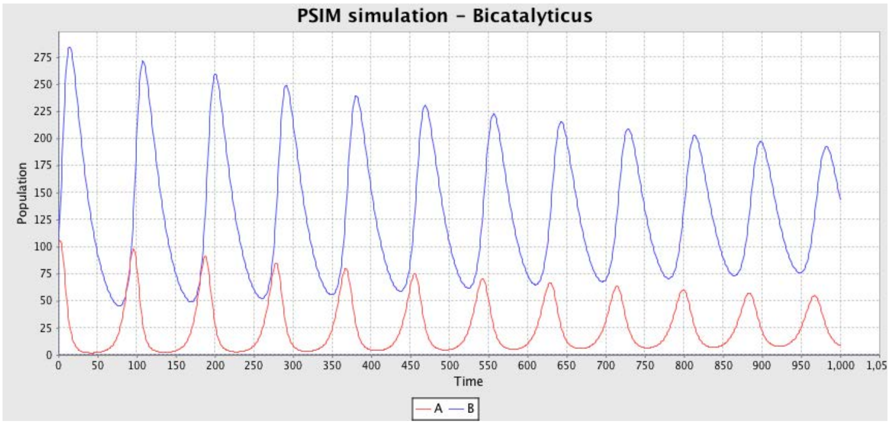


Fig. 3.20 The dynamics of *Bicatalyticus* in 1000 steps, with initial values A=100, B = 100, and all inertia values 100

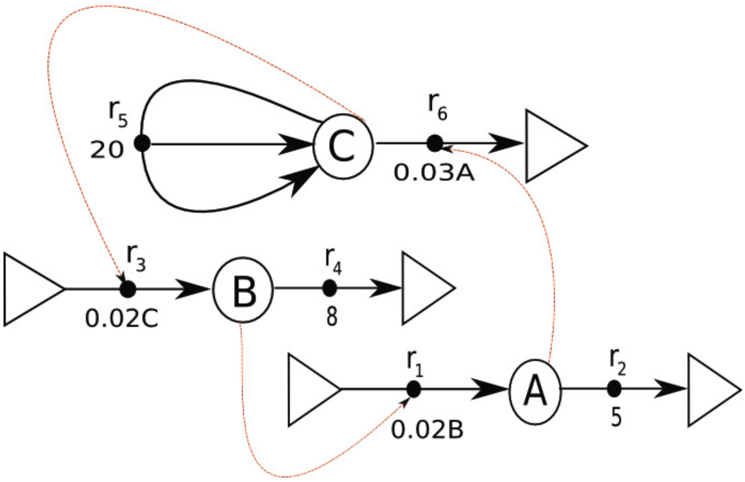


Fig. 3.21 *Paramitoticus*, an oscillator based on a kind or regulation mechanism analogous to that of the mitotic cycle

shows an opposite behavior, because it seems to tend toward a regular dynamics, but in the long run, it does not reach a complete regularity (see Figures 20.16, 20.17, 20.19, and 20.20 of [95]).

Other examples of oscillating MPF with their dynamics are given in Figs. 3.25, 3.26, 3.27, 3.28, 3.29, and 3.30.

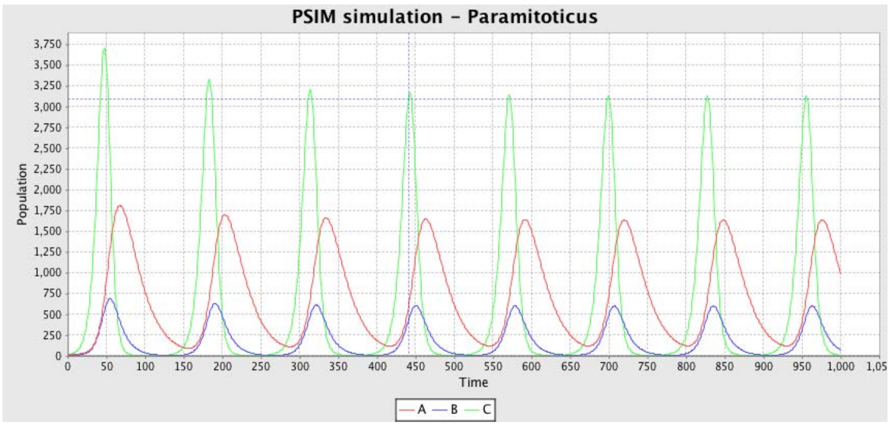


Fig. 3.22 The dynamics of the oscillator *Paramitoticus* in 1000 steps with initial values A = B = C = 10, and all inertias 100

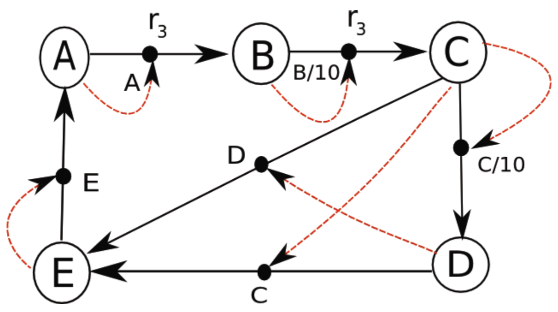


Fig. 3.23 *Vega*, an oscillator based on a transformation cycle

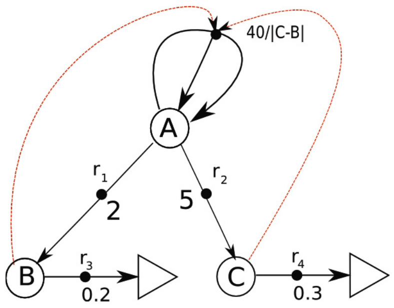


Fig. 3.24 *Mizar*, an oscillator based on two joint cis-regulations

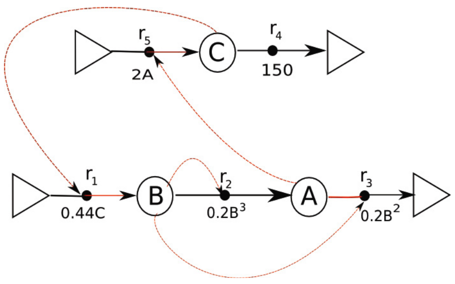


Fig. 3.25 *Deneb*, an oscillator based on a cycle of regulations

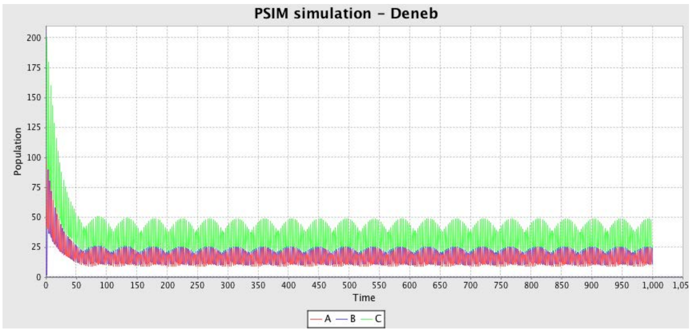


Fig. 3.26 The dynamics of 1000 steps of the oscillator *Deneb* with initial values A = 1, B = 100, C = 200, A-inertia = 50, B-inertia = 200, C-inertia = 1

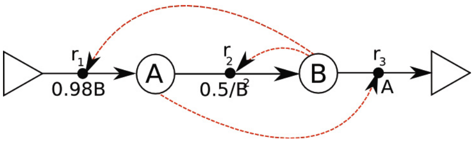


Fig. 3.27 *Orcincus*, an oscillator based on a composition of two cis regulations with a trans regulation

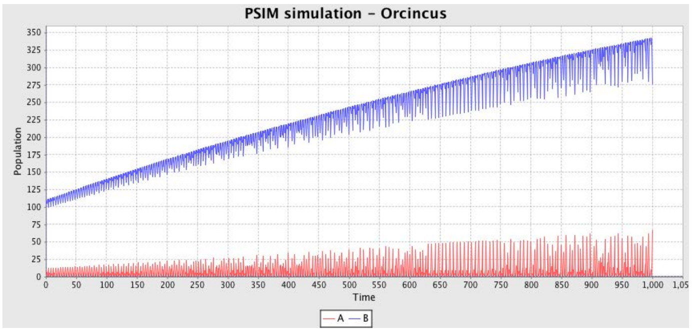


Fig. 3.28 The dynamics of 1000 steps of the oscillator *Orcincus* with initial values A = 100, B = 10, and inertias 10

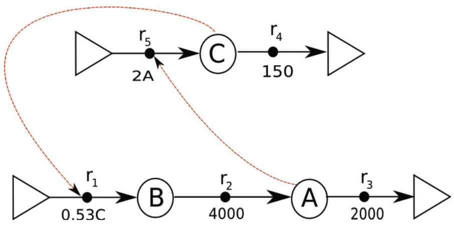


Fig. 3.29 *Gemini*, an oscillator based on an indirect self-regulation

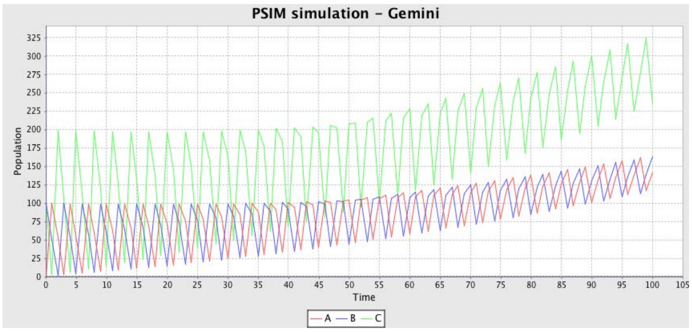


Fig. 3.30 The dynamics of 100 steps of the oscillator *Gemini* with initial values A=1, B = 100, C = 100, and A-inertia = 50, B-inertia = 200, C-inertia = 1

The MPR oscillators represented in Figs. 3.32 and 3.33 are different forms of the same metabolic structure, providing the same dynamics described in Fig. 3.31, which are based on the oscillator Sirius considered at the beginning of the chapter (see 3.5 and Fig. 3.3). In comparing the two MP graphs it is interesting to realize how duplication can be replaced by an input rule (in the following MP graphs, provided by PSIM software, ”Environment measures” refer to the parameters, which in these cases are absent).

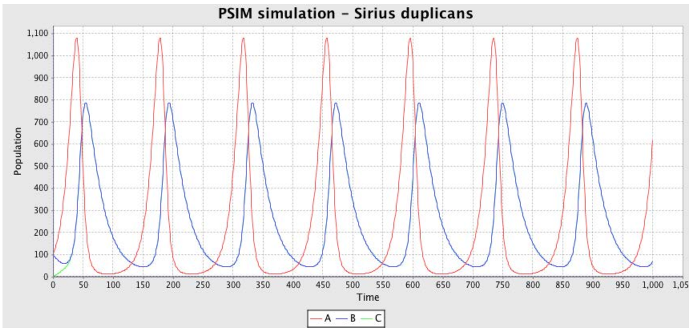
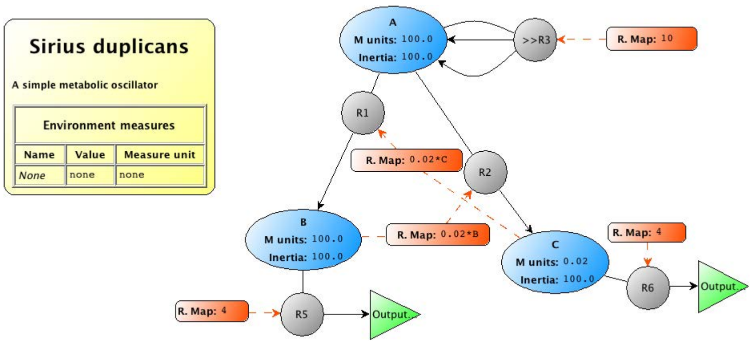


Fig. 3.31 The dynamics of *Sirius duplicans* and of *Sirius assimilans*, given in Figs. 3.32 and 3.33



Sirius duplicans

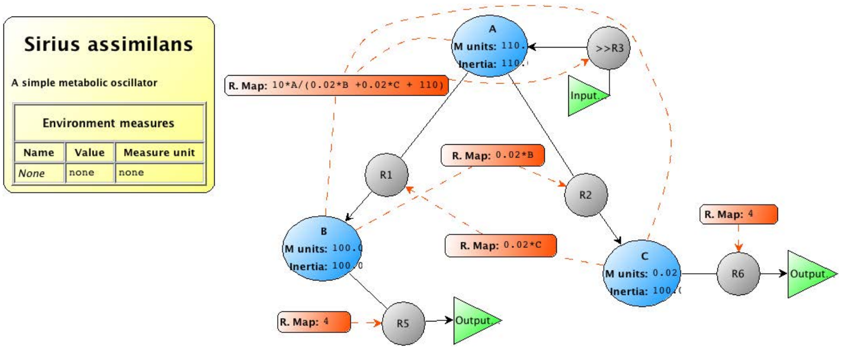
A simple metabolic oscillator

Environment measures

Name	Value	Measure unit
None	none	none

Fig. 3.32 The MPR graph of *Sirius duplicans*

The basic oscillatory mechanism of Sirius can be obtained with only two substances (see Figs. 3.34 and 3.35). This oscillator is essentially similar to Bicatalyticus (see 3.19). When the regulators of the reactions that transform the substance A are different, then substances B and C of Sirius exhibit different shapes (see Fig. 3.31 *versus* Fig. 3.37 with the corresponding dynamics of Fig. 3.37).

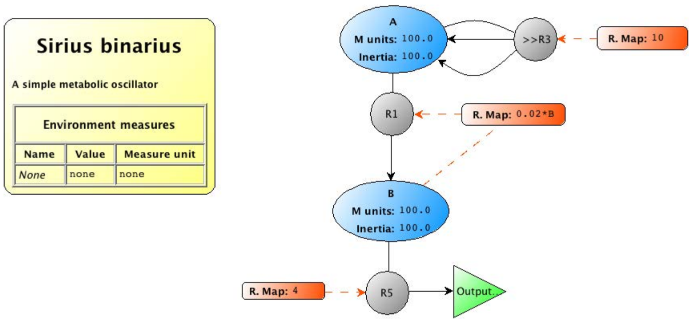


Sirius assimilans

A simple metabolic oscillator

Environment measures		
Name	Value	Measure unit
None	none	none

Fig. 3.33 The MPR graph of *Sirius assimilans*, dynamically equivalent to *Sirius duplicans*



Sirius binarius

A simple metabolic oscillator

Environment measures		
Name	Value	Measure unit
None	none	none

Fig. 3.34 The MPR graph of *Sirius binarius*

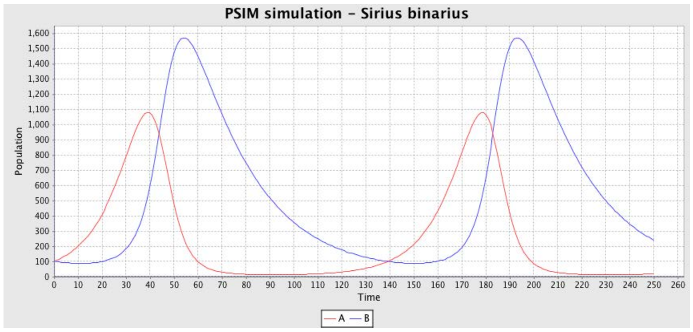
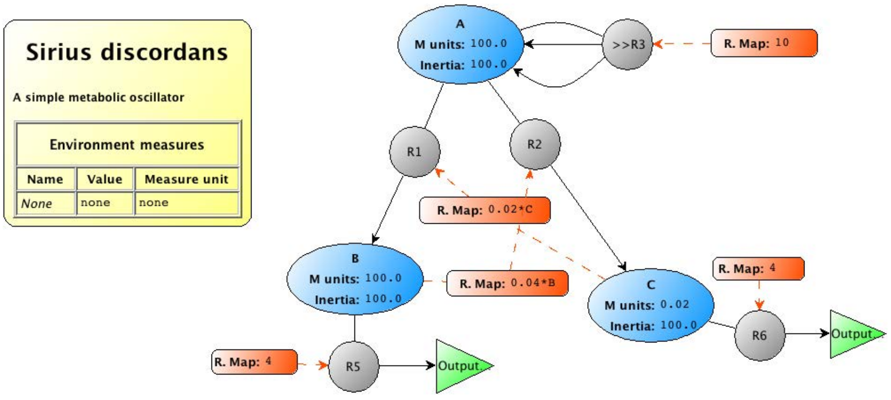


Fig. 3.35 The dynamics of *Sirius binarius*



Sirius discordans

A simple metabolic oscillator

Environment measures		
Name	Value	Measure unit
None	none	none

Fig. 3.36 The MPR graph of *Sirius discordans*

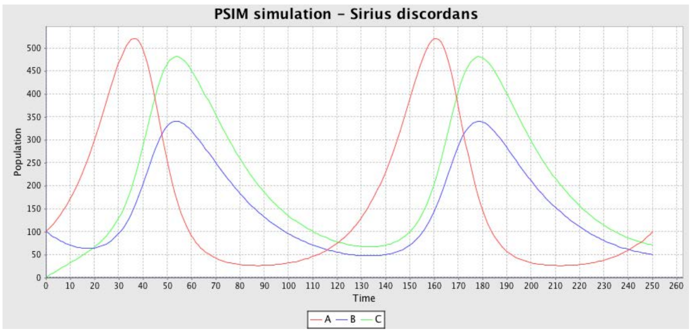


Fig. 3.37 The dynamics of *Sirius discordans*

3.7.1 Anabolism and Catabolism

A deeper analysis of metabolic phenomena distinguishes two different types of metabolism: **anabolism** and **catabolism**. This distinction is related to the energetic aspect of biochemical reactions. Anabolism is the part of metabolism related to the synthesis of substances that are energetically rich, or play a structural role in life processes. Catabolism is the metabolism of degradation of substances in their chemical components less energetically rich. In this section we introduce some concepts, related to anabolism and catabolism that are crucial in the analysis of metabolic systems. Two important consequences of distinction anabolism/catabolism are: i) the intrinsic verse of life processes which require input substances, coming from outside, and output substances, expelled to the outside; ii) the deep relationship between life and oscillations.

Let us consider **conservative** MP systems, which are close to the real metabolic system, because they satisfy two essential requirements of metabolic phenomena: *matter conservation* and *energy orientation* of metabolic transformations. Let us give some notation and definitions for expressing these requirements.

Given a reaction $\alpha \to \beta$ where multisets α, β are not empty, the matter conservation principle requires that the whole matter of reactants has to equate the whole matter of products, that is, neither loss nor gain of matter can occur in transformation reactions. This implies that in such a system a gain of matter can be realized only with input rules of kind $\emptyset \to x$, and a loss of matter can occur only with output rules of kind $x \to \emptyset$. In symbols, according to notation of MP systems, we have ($\mu(x)$ as the mass of a mole of x):

$$\sum_{x \in \alpha} \mu(x) = \sum_{y \in \beta} \mu(y).$$

A *reaction chain* of substances $x_1, x_2, \ldots, x_n$ is given by reactions $r_1, r_2, \ldots, r_{n-1}$ such that $x_1 \in S^-(r_1)$ (x_1 is a reactant of r_1) $x_n \in S^+(r_{n-1})$ (x_n is a product of r_{n-1}) and, for $1 < i < n$, $x_i \in S^+(r_{i-1})$ and $x_i \in S^-(r_i)$. A *reaction cycle* is a reaction chain where $x_n = x_1$.

An MP system is **time-bounded** if there is a bound to the maximum number of times its reactions can be applied. This means that, after a number of steps, its reactions do not have the amounts of substances necessary to them. A system is **time-unbounded** if it is not time-bounded. The system is **matter-bounded** if all the matter available in the system is not greater than a given quantity. A system is **matter-unbounded** if it is not matter-bounded. Of course, no physical system can last an infinite time, or can have an infinite quantity of matter, therefore, when speaking of time or matter unboundedness, we have to think of the term *unbounded* as equivalent to having *extreme* values (which we avoid to fix now) of duration or size (in relation to a certain time and matter availability, respectively).

An MP system is **assimilative** when there is at least one input rule. It is **dispersive** if there is at least one output rule. It is **dissipative** if it is simultaneously **assimilative** and **dispersive**, and it is **open** if it is either **assimilative** or **dispersive**, while it is **closed** if it is not open.

A simple way for providing oscillations in a closed MP system is given by only two substances A, B and two reactions: $r_1 : A \to B$ and $r_2 : B \to A$ with flux maps A and B respectively. However, no real chemical or biochemical system of this kind can exist. This impossibility is connected to a crucial aspect of chemical transformations which have to obey a principle based on Gibbs' variation of free energy. In simpler terms, the natural orientation of a chemical transformation is from reactants that are globally *energetically richer* to correspondent products *energetically poorer*. This verse can be inverted only by providing an additional energy, for example, by coupling the reaction with another one releasing the necessary energy for this inversion.

For MP systems where matter conservation holds, the energy orientation requirement can be stated by means of the following *cycle dissipation* principle (no free reaction cycle may exist).

Cycle dissipation principle
When an MP system is time-unbounded and has a reaction cycle, then surely a substance has to be present in the system that is consumed during the application of the reactions of the cycle.

Figure 3.38 shows the abstract idea underlying the cycle dissipation principle. The intuition of such a phenomenon can be understood with the metaphor of a watermill, where the wheel activity of this device needs an external water flow supporting the necessary mechanical energy providing the dynamics of the system. If matter-boundedness is assumed in a metabolic system with a reaction cycle, then according to the cycle dissipation principle, the input matter feeding the internal cycle needs to be coupled with output rules expelling matter.

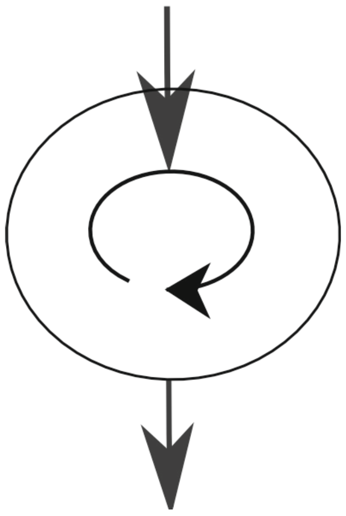


Fig. 3.38 The principle of cycle dissipation for metabolic systems, in the form of an (abstract) watermill

In the following discussion we consider MP systems which are assumed to satisfy the conservation principle and the cycle dissipation principle. The results we derive show the importance of open systems and their relationship with oscillatory dynamics.

Lemma 3.5. *Any closed MP system has to be time-bounded.*

Proof. If the system is closed, then it is matter-bounded. From the cycle dissipation principle no reaction cycles are present in the system. Therefore, after a number of steps reactants disappear. Therefore, reactions in the system cannot continue to be applied. □

Lemma 3.6. *Any open but non-dispersive MP system which is matter-bounded cannot be time-unbounded.*

Lemma 3.7. *Any open, but non-assimilative MP system which is matter-bounded cannot be time-unbounded.*

In the context of MP systems we say that a system oscillates if its substance quantities vary in time, but their values are always within some fixed ranges.

Proposition 3.8. *An MP system having a reaction cycle and which is matter-bounded but time-unbounded, has to be assimilative, dissipative, and oscillating.*

Proof. (*outline*) An MP system with a reaction cycle needs to feed some substance from outside, therefore it is assimilative. But if it is also matter-bounded and time-unbounded, then it has to be dissipative. These requirements imply that surely its substance quantities have to vary within some intervals, therefore the system is oscillating. □

Cells host complex metabolic processes and are bounded in matter and unbounded in time (when considered with their genealogical lineages). In fact, their sizes cannot be greater than some values and their reactions need to last as long as possible, because reproduction provides the reiteration of their internal metabolic processes in time. Of course, they are special conservative metabolic systems. This means that what was abstractly deduced for conservative MP systems definitely applies to cells, which need to realize oscillatory phenomena. In conclusion, *life is oscillation* and the analysis of metabolic oscillations is crucial to understanding life. In this regard, it is interesting to quote the words of the Nobel laureate Ilia Prigogine in his foreword to Goldbeter's book [82] (the oscillatory phenomenon to which Prigogine refers is the Belousov-Zhabotinsky phenomenon, which we considered in Sect. 3.1.2, about time asymmetry see also [147], Sect. 7 of Chap. 4, and Sect. 8 of Chap. 5):

> *I remember my astonishment when I was shown for the first time a chemical oscillatory reaction, more than 20 years ago. I still think that this astonishment was justified, as the existence of chemical oscillations illustrates a quite unexpected behavior. We are used to thinking of molecules as traveling in a disordered way through space and colliding with each other according to the laws of chance. It is clear, however, that this molecular disorder may not give rise by itself to supramolecular coherent phenomena in which millions and millions of molecules are correlated over macroscopic dimensions. ... It is this synchronization that breaks temporal symmetry.*

3.8 The Ubiquity of Metabolism

The French mathematician René Thom quotes in his book **Structural Stability and Morphogenesis** [155] a statement by the Greek philosopher Eraclitus saying that "fire rests by changing" (*Metabollon anapauetai*). Existence is transformation, and metabolism is the basis of biochemical transformation. However, metabolism is not only an important kind of transformation, but can be considered a paradigm within which many kinds of dynamical systems can be represented. We have already seen that real functions can be approximated by means of suitable MP systems (see Sect. 3.3), and we discussed in many points the prominent role of metabolism in cell functionalities and its link with DNA replication. Here, we want to point out some other fields, where the metabolic perspective could be naturally applied to analyze phenomena from observed behaviors.

The solution of many problems consists in the ability to discover discrete dynamical systems that generate time series of an observed phenomenon. This yields the identification of **an internal state transition logic** *from* **an external phenomenalistic evolution**. The algorithm LGSS, which we presented in Sect. 3.4.2 provides a method for solving dynamical inverse problems by means of MP grammars. Its resolution schema follows a pattern illustrated in Fig. 3.39. Therefore, if we are able to describe a phenomenon in "metabolic terms", within a suitable class of MP grammars, the discovery of specific MP grammars modeling it can be developed by means of LGSS. In Sect. 3.8.2, a very general notion of MP grammar will be presented, which suggests a large applicability of this concept for a wide class of dynamical systems. Of course, this enlarges the fields where MP theory and LGSS methodology could be applied for solving dynamical inverse problems: from the time series of meteorological variables, to time series of economic, or of ecological nature. What are the evolution rules that are responsible for the series we observe? Sometimes, some basic principles are known, but many detailed aspects are missing or are difficult to be deduced in a reliable manner.

In this section we outline an example, where a crucial phenomenon of gene expression is analyzed by means of MP grammars. In the next section, examples are given where distributed computations are expressed in terms of suitable MP grammars.

Two relevant cases of biological interest are represented by **signal transduction** and **regulation networks**. The first case is a sort of metabolism where no matter conservation principles are applicable. Namely, signals (for their intrinsic nature) propagate by changing places and forms of appearance, but they can also be amplified or reduced, and this is a peculiarity very important in their transformation and communication.

The second case, which we report shortly, concerns a research in progress on gene expression analysis in a cancer cell [109, 140, 141]. Assume that a given perturbation, of a pharmacological nature, is activated on this cell, and the gene-expressions of around 20,000 genes are given along a sequence of time points. An important problem to solve is the effect of this perturbation on the gene-

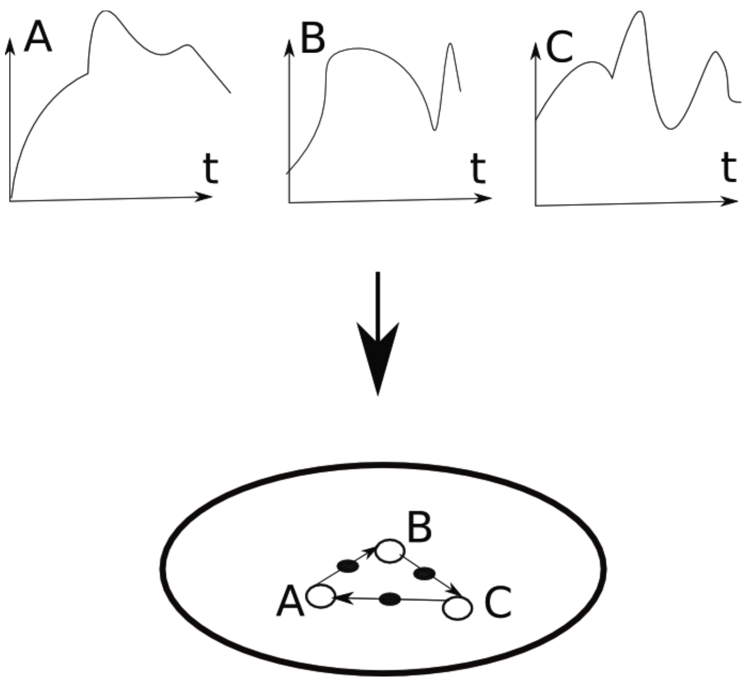


Fig. 3.39 The general schema of metabolic paradigm in dynamical inverse problems

expressions. Given the huge number of different genes, a clustering phase is necessary to collect all the genes that provide the same response, in time, to the perturbation. When this is done, after some statistical elaboration of data and a normalization of the measurements, it is important to discover the logic of interactions of these clusters of genes during the effect of this perturbation. This can be performed by using an MP grammar, and then by reading it in terms of classical gene regulation networks. This is an example of the utility of the MP paradigm in a field that apparently does not present metabolic phenomena (in classical sense). The following Figs. 3.40, 3.41, 3.42 provide seven interpretations of basic gene regulatory phenomena where the representation of these gene regulation mechanisms are accompanied by the corresponding MP graph representation. From the results we got, in a specific analysis of medical interest, this approach provided new clues in the explanation of the regulatory mechanisms involved with a specific cancer pathology (see [109] for more details and for other applications to medical problems).

In a first approximation MP systems can be viewed as discrete approximations of metabolic models based on ordinary differential equations (ODE), by means of recurrent difference equations. Namely, for very small temporal interval τ, fluxes

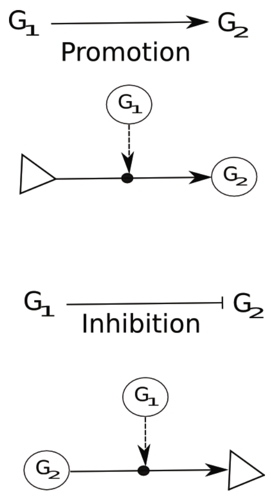


Fig. 3.40 Gene-expression networks and MP graphs: promotion and inhibition

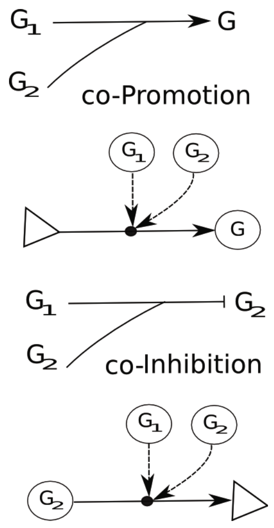


Fig. 3.41 Gene-expression networks and MP graphs: co-promotion and co-inhibition

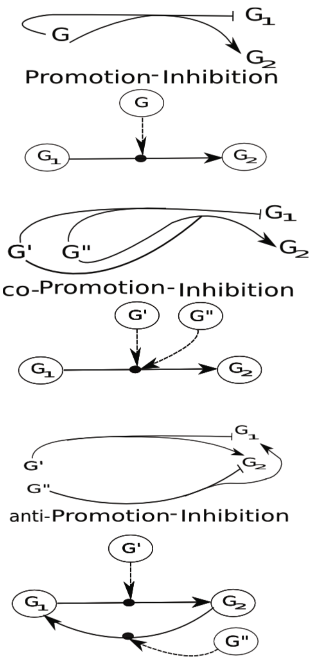


Fig. 3.42 Gene-expression networks and MP graphs: co-promotion-inhibition and anti-promotion-inhibition

of an MP system approximate to differentials and EMA reduces to a numerical integration performed according to the Euler method [77]. However, when the time interval cannot be approximated to an infinitesimal time, MP models introduce a perspective which is radically different from the ODE perspective [83, 115], more related to the transformations of the systems along observation steps, rather than to the real kinetic of reactions, and to the chemo-physical mechanisms underlying biochemical processes. However, the different perspective of MP systems has computational and modeling advantages, by opening the possibility of algorithmic and algebraic procedures for disclosing regulation logics of metabolic systems. Namely, dynamical inverse problems can be solved (via the LGSS algorithm) and the mathematical form of regulators can be discovered in a systematical way.

In ODE models, and in stochastic models [80, 74, 51, 79] a critical point is constituted by the evaluation of kinetic rates or stochastic parameters. This is a hard and unreliable task, because very often the exact microscopic dynamics of reactions is not completely known, or even when it can be completely studied in vitro, its in vivo behavior could significantly differ from its experimental evidence.

An MP model assumes that some species are related by means of some transformations, and that the variations are due to the global action of these transformations. Of course, their execution could involve very complex underlying subtransformations, but this is outside the objective of the model. It only tries to explain what is observed in terms of the chosen species and transformations. If the choices are not the right one, this means that the model was not adequately defined, but this is independent from the methodology, it is only a matter of modeling design. Therefore, MP modeling is deliberately at a different, more abstract, level with respect to ODE models. This does not means that it is less adherent to the reality, but simply that it is focused on a different level of reality. Of course, in this way many important dynamical details can be lost, but in many cases what is relevant is the global pattern of a behavior and the main correlations among the involved quantities. Moreover, in many cases, a higher level of analysis is the only possible approach, because we know only some time series related to a given phenomenon, and no idea is available about the internal laws ruling the observed phenomenon. This is the case that we consider above, concerning a situation of gene expressions network, where thousands of variables are involved.

3.8.1 Metabolic Computing

We have already shown (see Sect. 3.3) that MP systems can be used for approximating real functions. But they can also generate time series which correspond to exact values of real functions along sequences of values. In the following, we give some examples of MP grammars computing (or generating) the values of linear (Table 3.22), square (Table 3.23), root square (Table 3.25), and exponential functions (Fig. 5.13). It is remarkable that in all these cases fluxes are computed by linear regulators.

In these computations, we assume a criterion of **lack of reactant block**, that is, the flux of a rule becomes null when the regulator assumes a value greater than the current quantity of one of its reactants (there is an insufficient amount of matter for applying the rule). In this case, also all the rules which share that reactant are forced to have a null flux.

Elaborating on the idea of fluxes that become null when they exceed the availability of reactants, a computation model on natural numbers can be developed which is computationally universal (equivalent to register machines [99]).

The square root of positive integers is easily computed by the algorithm given in Table 3.24, where the input value $n > 0$ is decreased at any step i by odd number $2i+1$, thus by reducing the initial value of increasing squares. This reduction is continued while the reduced value remains positive (or null) and, at same time, a counter (B) memorizes the number of steps. When the reduction process stops, the counter provides the approximate value of the square root of n. This algorithm is expressed by the MP grammar of Table 3.25. In fact, the number n is put as value of A, and at any step A is decreased by increasing odd numbers, according to rules

r_1 and r_2. When A cannot be reduced anymore, because its quantity is less than the flux of rule r_1 (lack of reactant block), then $D = 0$, therefore, in the following step the value of B is assumed by Y (rule r_4), and $C = 0$ (rule r_5). At that moment, the successor of the rounded square root of n is the value of Y. Therefore, in another step rule r_6 provides the final result in Y and puts $Z = 1$, that is, the computation ends (no rule can be applied). The obtained result in Y is exact if $A = 0$, otherwise A contains the value exceeding the rounded root of n.

In Fig. 3.24 the MP graph of grammar of Table 3.25 is given, where an external oval envelops *internal* substances, while those that host input and output values are externally visible. In this kind of metabolic computational device, the control of computation is naturally distributed, because it emerges from the way reactions and regulators are connected (the correctness of the grammar, with respect to expected computations of the square roots, can be proved by induction). The device of Fig. 3.24 can easily be represented in a pure visual form (see Fig. 3.45), as regulators are sums of substances, thus the formulae defining them can be replaced by arrows connecting the substances that additively contribute to the same regulation. Moreover, Fig. 3.24 suggests a notion of *multi-membrane* as a structure collecting many membranes, but where only some of them play the role of interfaces (accessible internally and externally).

We refer to [99] for a discussion on some topological and biological intuitions related to these structures. Of course, there are many similarities between this approach and the well-known model of Petri nets [113, 114, 72], but here regulators play a specific role that gives special peculiarities to MP computations.

MP grammars generating/computing linear, quadratic, and exponential functions are given in Tables 3.22, 3.23, and Fig. 5.13. By combining and extending such kind of grammars, also MP grammars for polynomials, logarithms, sigmoids, and asymptotic functions can be defined.

Table 3.22 An MP grammar generating the sequence of natural numbers $mn + k$ for $n \in \mathbb{N}$

$r_1 : \emptyset \rightarrow A$	$\varphi_1 = m$
$A[0] = k$	

Table 3.23 An MP grammar generating the sequence of square natural numbers in the values of substance A

$r_1 : \emptyset \rightarrow A$	$\varphi_1 = 2B + 1$
$r_2 : \emptyset \rightarrow B$	$\varphi_2 = 1$
$A[0] = 0$	
$B[0] = 0$	

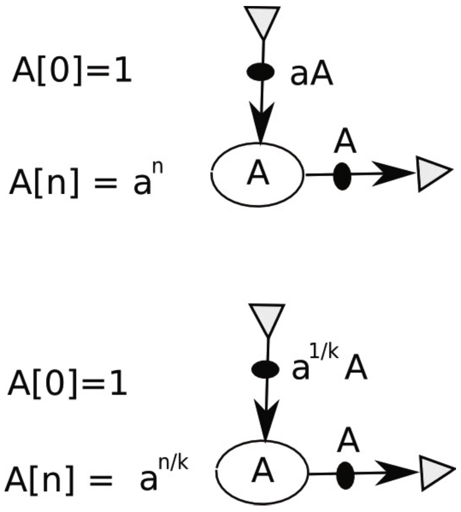


Fig. 3.43 MP grammars computing exponential functions

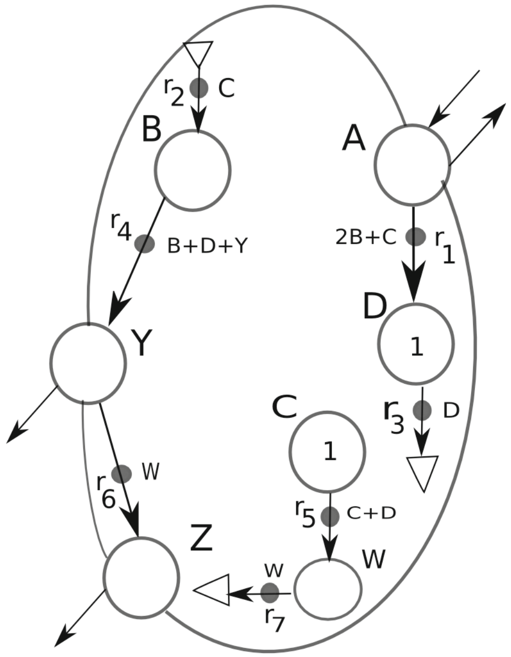


Fig. 3.44 The MP graph of the MP grammar of Table 3.25

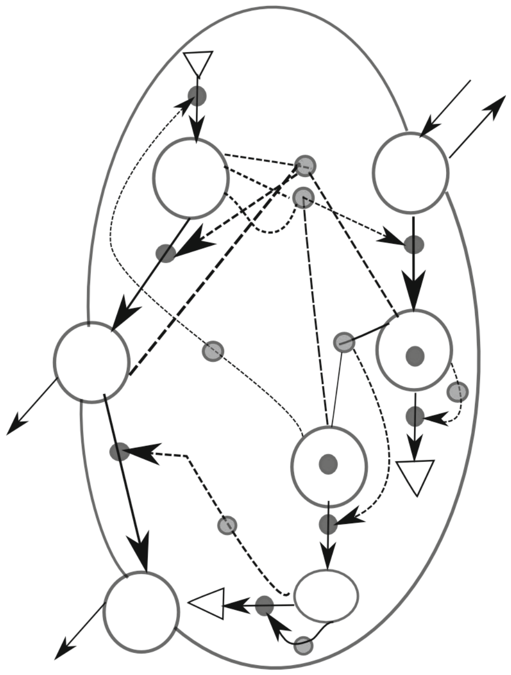


Fig. 3.45 A pure visual representation of the MP grammar of Fig. 3.24

Table 3.24 An algorithm for computing the square root of positive integers

```
begin
input $A$ (positive integer);
$B := 0$;
$C := 1$;
while $A \geq 2B + C$
do
    A := A - (2B + C);
    B := B + 1;
od;
output B, A
```

Table 3.25 An MP grammar computing the square root $\sqrt{n}$ of a natural number n. The value n is the initial value of substance A, while the (rounded) result can be found in Y when $Z = 1$, and at the end, the difference between the initial value and result is put in A.

$r_1 : A \to D$	$\varphi_1 = 2B + C$
$r_2 : \emptyset \to B$	$\varphi_2 = C$
$r_3 : D \to \emptyset$	$\varphi_3 = D$
$r_4 : B \to Y$	$\varphi_4 = B + D + Y$
$r_5 : C \to W$	$\varphi_5 = C + D$
$r_6 : Y \to Z$	$\varphi_6 = W$
$r_7 : W \to \emptyset$	$\varphi_7 = W$
$A[0] = n$	
$B[0] = 0$	
$C[0] = 1$	
$D[0] = 1$	
$Y[0] = 0$	
$W[0] = 0$	
$Z[0] = 0$	

3.8.2 *Revisiting MP Grammars*

We conclude this chapter with a general concept of MP grammar, which extends the initial metabolic inspiration toward a general dynamical paradigm which can be applied to any system of variables.

First, let us recall that a dynamical system is given by a set of real **variables** changing in time and a set of **invariants**, that is, conditions or constraints that are satisfied by the variables during their change.

Now, let us assume a generalized notion of reaction among variables such as $X \to Y$, meaning that X decreases by a certain amount, while Y increases by the same amount (with respect to some measurement unit). The decreasing/increasing amount, called flux of the rule, is determined by a regulator (regulation map) depending on some variables of the system.

An MP rule is a **regulated reaction among variables**, that is, a pair constituted by a reaction and a regulator. A reaction is constituted by left variables which decrease, while some corresponding right variables increase (each variable is equipped with a multiplicity). A **regulator** is a function providing the flux of the rule, in dependence of the values of some regulation variables, called **tuners** of the rule (possibly including parameters). Every left variable decreases of a quantity which is the product of its multiplicity with the flux of the rule, and every right variable increases analogously.

An MP grammar is a set of MP rules. Given an MP grammar, when we start from an initial state (the values of its variables at an initial time) by applying all the rules of the grammar, we obtain the next state and so on, for all the subsequent steps. An MP grammar becomes an MP system when some numerical values are fixed for the physical interpretation of the time series: the time interval between two consecutive applications of rules, and other values related to the quantity units (depending on the physical nature of variables).

In mathematical terms an MP grammar is specified by: **variables, reactions, regulators, initial values**. Variables that do not occur in reactions, but occur in regulators are called **parameters**, while variables that are not parameters are called **(proper) variables**. Given some initial values of its proper variables, an MP grammar deterministically generates a time series for each of its proper variables. If parameters are present in the grammar, then also the time series of the parameters take part in the generation of dynamics. It is easy to realize, and it will be evident in the following definition, that this notion of MP grammar defines a system of finite difference recurrence equations, which represent the invariant of the dynamical system generated from it. In the following generalization of MP grammar, the letters MP may be related to **Minus-Plus** rules, which express the dynamics by means of (regulated) decreasing-increasing of (proper) variables.

Definition 3.9 (Generalized MP Grammar). A (generalized) MP grammar is a structure:

$$(X, Y, R, X[0])$$

where:

- X is a set of real **variables**;
- Y is a (possibly empty) subset of variables, called **parameters**. The elements of set difference $X - Y$ are also called **proper variables**;
- R is a set of MP **rules**, that is, pairs of type (**reaction, regulator**):

$$\alpha_r \rightarrow \beta_r \quad ; \quad \varphi_r.$$

  For any $r \in R$, a reaction $\alpha_r \rightarrow \beta_r$ is a pair of multisets over the set X/Y of proper variables, and a regulator φ_r is a function from X-states to $\mathbb{R}$ (an X-state is a function from X to $\mathbb{R}$). Variables that are arguments of φ_r are the **tuners** of φ_r;
- $X[0] = (x[0] \mid x \in X)$ is an **initial state** constituted by the values of variables at an initial time 0.

For any X-state s, and for any proper variable $x \in X/Y$, the rules determine a decrease-increase $\Delta_x(s)$ of x, according to the following formula (each reaction decreases any left occurrence of x and increases any right occurrence of x):

$$\Delta_x(s) = \sum_{r \in R} (\beta_r(x) - \alpha_r(x))\varphi_r(s). \tag{3.49}$$

($\alpha_r(x) = \beta_r(x) = 0$ if x does not occur in the reaction of r). Let us set, for every $i \in \mathbb{N}$:

$$X[i] = (x[i] \mid x \in X)$$

then, the (proper) variable variation (column) vector $\Delta[i]$ is given by (superscript T denotes matrix transposition):

$$\Delta[i] = (\Delta_x[i] \mid x \in X/Y)^T$$

Therefore, if $\mathbb{A}$ is the **stoichiometric reaction matrix** (with proper variables as row indexes, and rules as column indexes):

$$\mathbb{A} = ((\beta_r(x) - \alpha_r(x)) \mid x \in X/Y, r \in R)$$

and the **flux** (column) **vector** is given by:

$$U[i] = (\varphi_r(X[i]) \mid r \in R)^T$$

then, assuming to know at every step $i > 0$ the values of parameters, Eq. (3.49) can be expressed in vector form by ($\times$ is matrix product):

$$\Delta[i] = \mathbb{A} \times U[i]$$

that is, the finite difference recurrence equation EMA of Definition 3.1.

The passage from an MP grammar to an MP system is obtained by fixing some aspects of physical nature: the time interval τ and other physical measurement units related to the specific system under investigation. Of course the same idea can be expressed in other ways, according to the aspects one wants to stress and to the conceptual organization of the structural and physical elements defining a discrete dynamical system.

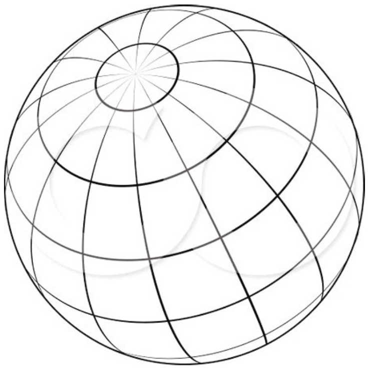
For example, we could define an MP system M by a structure, where the grammatical component G of the system is focused completely on the rules, that is:

$$M = (X, Y, F, X[0], \tau, G)$$

where:

- X are the variables and $Y \subset X$ are the parameters ($X - Y$ the proper variables);
- F is the set of physical measurement units of the system;
- $X[0]$ is the vector of the initial values of variables;
- τ is the step interval time (the values of parameters are supposed to be known in all the points at time multiple distances τ from the initial time 0);
- G is the MP grammar of the systems, that is, a set of MP rules over variables X (consisting of reaction-regulator pairs).

In this perspective, an MP grammar is the discrete counterpart of an ODE system, as it identifies the difference recurrence (vector) equation EMA representing the dynamical invariant of the system, while the other elements $X, Y, F, X[0], \tau$, refer to the variables and to the other notions giving physical meaning to the numerical values of the time series associated to them.


Sphere

4 Life Strategies

Abstract. Life is based on complex systems interacting with environments. In order to guarantee their existence, these systems developed sophisticated strategies that solve problems by introducing new problems. This is a peculiar aspect of life complexity that sounds paradoxical in many aspects. In this chapter we outline some basic strategies of life.

4.1 The Enzymatic Paradox

The kernel of Eigen's paradox [128] points out a crucial aspect of life and its fundamental relationship with the basilar mechanism of molecule replication. In fact, assume that at some point of molecular evolution a polymer α was produced with the ability of self-replication. Let n be the length of this polymer, of course this replication ability was characterized by an error rate ε which, after a while, with a great probability provides polymers which lose the very rare and remarkable property of self-replication. For improving its auto-replicative capacity, this molecule α surely had to increase its complexity, that is its length. In fact, a longer replicator can perform a better replication. But, at the same time, a longer polymer implies a greater error rate in the replication process. Therefore, self-replication implies two conflicting tendencies which have to meet at an equilibrium point $(\alpha_0, n_0, \varepsilon_0)$ with an error rate which is impossible to diminish in terms of polymer self-replication. This argument shows, in qualitative terms, that self-replication can be improved along a wider strategy. This was the strategy which life followed in its evolution from the biochemical level to the cell level. Rather than single molecule self-replication,

V. Manca: *Infobiotics*, ECC 3, pp. 169–211.
DOI: 10.1007/978-3-642-36223-1_4 © Springer-Verlag Berlin Heidelberg 2013

life was surely forced to explore self-replicating molecular systems. A good self-replicating system is a system where self-replication is realized in the context of a wider system of reactions. In other words, replication is coupled with metabolism by means of a reaction system able to keep stability and robustness with respect to the external variations. Membranes are the essential devices for the realization of this coupling. Therefore, cells could have emerged, from the prebiotic attempts, when cell replication became a synchronized process of polymer and membrane replication together with all the underlying reactions also connecting the two processes. In this case, replication is not only a replication of molecules, but a more general process of **reproduction**, where not only molecules, but the entire processes of their intertwined reactions are replicated. In conclusion, a good replication of molecules can be reached within a more complex process that includes the replication of molecules. In this sense, life followed a strategy where a solution is searched for by framing the original problem in a more general context. This is a resolution pattern recurrent in many life phenomena, that is, *life thinks at large*. It is also interesting to remark that, it corresponds to a well-known strategy, in the solution of mathematical problems, termed by Pólya [187] **Inventor's paradox**: "The more ambitious plan may have more chances of success".

We do not know, at present, the details which explain why molecule replication within cells is able to reach a more reliable and efficient level, by escaping the limitation of Eigen's paradox. However, surely replication, in the context of cell reproduction, benefits from the evolutive mechanism, in the sense that natural selection pushes toward the best reproductive processes, which also selects the best molecule replication processes. Now we present another life paradox, the **enzymatic paradox**, where we will consider metabolism and we show that an intrinsic contradiction of this phenomenon can be solved if metabolism is joined with replication in a reproductive system. Therefore, if replication postulates metabolism, according to Eigen's paradox, vice versa metabolism postulates replication and the deep interplay between these processes in reproductive systems follows as an intrinsic necessity of their basilar mechanisms. This means that the long debate about a priority between them seems meaningless, on the basis of their mutual logics. The cell is essentially a reproductive unit, and in reproduction phenomena replication and metabolism are the two faces of the same coin.

Another life paradox is the **evolutive paradox**. Is evolution driven by the search toward the best fitness/dominance in a given environment, or more precisely its dynamics has an internal nature which is not caused by the environment, but is only selected by it? In other words, life realizes its fitness for evolving or evolves for reaching the best fitness. The first case seems to be more appropriate, according to the error/chance strategy of life, which freely explores possibilities, subsequently selected by the environment. This means that life evolves just for evolving, and fitness is a by-product of this tautological, intrinsic necessity.

Analogously, the **morphogenetic paradox** arises when we consider that differentiation postulates a previously determined form to be realized, but how and when this form realization does emerge in cells which are not predetermined to express forms?

Life is a paradoxical phenomenon which evolves for solving paradoxes. In fact, the solutions it finds raise other paradoxes, which move the intrinsic mystery of its nature forward.

Almost the totality of biochemical reactions is performed by means of specific catalysts. They are bio-molecules which allow reactants to meet and to react in the best way by decreasing the intensity of *activation energy*. These molecules are called **enzymes**. Usually, each reaction needs its specific enzyme, which is often a protein. However, this notion of enzyme immediately raises a very general problem of logical nature. In fact, also enzymes are bio-molecules, therefore i) either they need biochemical reactions producing them; therefore enzymes which in turn need enzymes along a tautological regression, ii) or they need to be introduced into the cell from outside. In the second case, either they are introduced in the cell in a free way, or they need some specific molecules which select the kinds of molecules that are allowed to enter. The case of nonselective introduction of enzymes is not reasonable from many points of view, but basically it is biologically unreasonable that in a cell any enzyme floating in the environment could enter.

In the case of a regulated introduction of enzymes, the problem of enzyme production is only shifted to the problem of enzyme introduction. In fact, enzymes should need specific biomolecules that are able to recognize them, by allowing them to enter in the cell. But where do these introduction enzymes come from? In conclusion, enzymes are supposedly produced internally in the cell, but this assumption implies a *petitio principii*. In fact, any reaction which synthesizes an enzyme postulates an enzyme too, therefore any chain of enzymes (A making B making C, and so on, for a finite number of steps) needs a starting enzyme which cannot be produced internally by the cell. In conclusion, some enzymes need to come from outside, but this seems only to change the terms of the problem without solving it. Therefore, where and how are enzymes produced/introduced in the cell? We call this contradictory situation the **enzymatic paradox**.

A possible solution of the enzymatic paradox is the existence of molecules of a kind that allows the synthesis of enzymes, but which are not enzymes and which came from the outside. Where do they come from? Let us call these bio-molecules **I-molecules** for their intrinsic informational role. They came into the cell from outside, but their externality is in a temporal sense rather than in a spatial sense. They are passed to the cell by its mother cell. Simple arguments of reliability imply that a unique I-molecule collecting the whole information for the synthesis of all enzymes is the best solution of the enzyme puzzle. But this unique I-molecule can be passed to the daughter cell, after a duplication of it, otherwise the mother cell has to die as soon as its unique copy of the I-molecule is released outside it.

The enzymatic paradox, and its solution outlined above, constitutes a sort of a *logical proof* of the necessity of the two levels in the organization cells: the M-level of metabolism and the I-level for storing the information of enzyme construction. According to this perspective, it turns out that metabolism and replication are two intertwined phenomena, and the argument here outlined represents a *more geometrico* deduction of the dualism genes and proteins as a logical necessity of the autocatalytic nature of the biochemical reactions hosted in the cells. The following list of facts shows in a more evident way the logical nature of the enzymatic paradox and of its solution:

1. Fact 1. Every biochemical reaction needs an enzyme catalyzing it.
2. Fact 2. Every enzyme is either internally produced by the cell, or introduced into the cell from the outside.
3. Fact 3. The introduction of an enzyme in the cell has to be selective (not any enzyme floating in the external environment can be internally introduced).
4. Fact 4. A selective introduction of an enzyme in the cell needs a specific molecule which needs a specific enzyme producing it.
5. Fact 5. The previous facts imply either a *regressio ad infinitum*, or a contradiction in both cases of internal production or external introduction of enzymes.
6. Fact 6. A different kind of external introduction can avoid the consequences of the previous facts: an introduction of special molecules in the cell from the mother cell. They are I-molecules which are not enzymes, but provide the necessary information for internally producing the enzymes of a cell.
7. Fact 7. I-molecules need to be replicated in the cell by enzymes already present inside it, otherwise the passage from the mother to the daughter cell should imply the death of the mother cell.
8. Fact 8. According to the previous fact, life is based on heredity and I-molecules are the molecular basis of this mechanism.
9. Fact 9. A more reliable and efficient mechanism for molecular heredity is reached by a unique I-molecule collecting all the information for the synthesis of enzymes.
10. Fact 10. The solution of the enzymatic paradox, outlined by the previous facts, is based on reproduction and on heredity. In reproduction, the replication of molecules is realized in the context of metabolism and vice versa metabolism is realized by means of mechanisms of molecule replication.

4.1.1 Genes and Proteins

So far we have concentrated on enzymes. They provide the basic metabolic activity of cells, but bio-molecules performing the basic functions of cells constitute the larger class of **proteins**. In fact reactions need to be catalyzed, but also regulated. Moreover, bio-molecules for recognizing, transporting molecules, and for performing signals and movements are also essential in a cell as well as bio-molecules for assembling specific structural components (gates, tunnels, reticula, bases, histones, cilia, and so on). Table 4.1 summarizes the basic functions of proteins.

Amino acids are the monomers of polypeptide sequences. They are encoded by triples of RNA nucleotides. However, this encoding is partial and redundant. This means that some of all possible triples over the alphabets $\{A,U,C,G\}$, which do not encode amino acids, as is the case of triples UAA, UAG, UGA are not encoding, while many triples may denote the same amino acid. Table 4.2 gives this correspondence, called **genetic code**, where a letter of the English alphabet is associated to any amino acid, and sets of codons encoding the same amino acid are synthetically denoted by

Table 4.1 Protein functions

Transformation
Regulation
Recognition
Signal
Transport
Movement
Structure

regular expressions where $P = U + C$, $Q = A + G$, $N = P + Q$, $\neg G = U + A + C$. The 20 amino acids are encoded by 61 codons while the remaining codons (UAG, UGA, UAA) encode the *stop* character. The codon AUG both codes for methionine and serves as an initiation signal for translation. In fact, the first AUG in an mRNA's coding region is where translation into protein begins. A circular representation of genetic code is given in Fig. 4.3.

Table 4.2 The amino acids with their codons and regular expressions

Amino acid	Name	Letter	Codons	Regular expression
Arg	Arginine	R	$\{CGU, CGC, CGA, CGG, AGA, AGG\}$	$CGN + AGQ$
Leu	Leucine	L	$\{UUA, UUG, CUU, CUC, CUA, CUG\}$	$CUN + UUQ$
Ser	Serine	S	$\{UCC, UCU, UCA, UCG, AGU, AGC\}$	$UCN + AGP$
Ala	Alanine	A	$\{GCU, GCC, GCA, GCG\}$	GCN
Gly	Glycine	G	$\{GGU, GGC, GGA, GGG\}$	GGN
Pro	Proline	P	$\{CCU, CCC, CCA, CCG\}$	CCN
Thr	Threonine	T	$\{ACU, ACC, ACA, ACG\}$	ACN
Val	Valine	V	$\{GUU, GUC, GUA, GUG\}$	GUN
Ile	Isoleucine	I	$\{AUU, AUC, AUA\}$	$AU\neg G$
Asn	Aspargine	N	$\{AAU, AAC\}$	AAP
Asp	Aspartate	D	$\{GAU, GAC\}$	GAP
Cys	Cysteine	C	$\{UGU, UGC\}$	UGP
His	Hystidine	H	$\{CAU, CAC\}$	CAP
Gln	Glutamine	Q	$\{GAA, GAG\}$	GAQ
Glu	Glutamate	E	$\{CAA, CAG\}$	CAQ
Lys	Lysine	K	$\{AAA, AAG\}$	AAQ
Phe	Phenilananine	F	$\{UUU, UUC\}$	UUP
Tyr	Tyrosine	Y	$\{UAU, UAC\}$	UAP
Met	Methionine	M	$\{AUG\}$	AUG
Trp	Tryptophan	W	$\{UGG\}$	UGG

RNA molecules are obtained by transcription from DNA and after that they are translated by specific complexes of proteins and RNA which are called ribosomes. They are particular translation machines, called transducers, in the sense of formal language theory. A transducer is an extension of a finite automaton which reads one letter per transition, possibly changing its internal state, but also replacing the letter by a suitable string (in dependence from its current state). For example, the *iterated transducer* given in Fig. 4.1 generates the tri-somatic language. It translates the string in input (in the initial state on the top) in the string of the same language that immediately follows it in length. Then, transduction is iterated, because the result of the translation (when it reaches the final state, on the bottom) is given as input of another translation. The strings obtained in this way, when it reaches the final state, are those of the tri-somatic language.

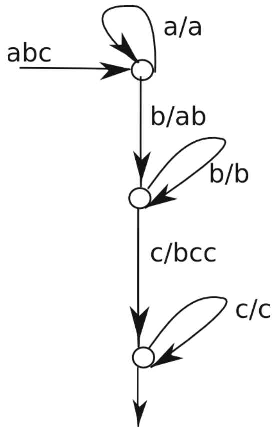


Fig. 4.1 An iterated transducer performing a tri-somatic growth

Figure 4.2 represents a transducer which realizes the translation of RNA transcripts, according to the genetic code. This transducer has 21 states, which is exactly the number of data encoded by the genetic code (20 aminoacids + the stop character). In Fig. 4.2, the region constituted by the full circle in the center, together with the bold line and the bold external circle, represents the initial state of the transducer. The states having transition to the external circle are the final states (where the transducer is after reading three consecutive nucleotides). All the transitions from non-final states translate the symbol into the empty character (indicated in the figure with "-"). There are 25 transitions to 16 final states. In fact, in four cases there are two different transitions providing the same output (R, L, S , and stop). Therefore, the transducer translates all the 64 codons into 21 outputs (20 aminoacids and stop). Apart from one case, all the final states have one or two transitions reaching again the initial state. This means that the transitions cover in an almost uniform way all the possible outputs. In principle, 5 transitions for each group of 4 final states, plus an additional transition, for example, a schema of transitions 5+5+5+6 could encode the 20 amino acids plus the stop character.

In the transducer of Fig. 4.2, the schema of final transitions 5+5+7+8 is near to the minimal schema, apart from the redundancy of the 4 extra transitions (related to R, L, S, and stop). We remark that a code of length 3 over an alphabet of four symbols, where the third symbol is always significant, has to encode al least 20 elements. In fact, the 4 groups of final states (each for each alphabet symbol) cannot have only 4 transitions, otherwise the translation can be done only with strings of length 2. Therefore, at least 5 transitions for each group provide at least 20 transitions. It would be interesting to establish if the codon length 3 is a consequence of having at least 20 encoded elements, or otherwise, if 20 is a consequence of having codons of length 3. Moreover, is the little discordance from the minimality related to other important aspects of the genetic code, or is it a matter of contingency in the evolutive process of definition of the code? Finally, we remark the importance of having an asymmetric treatment of start and stop signals. The begin signal is given by an AUG codon which encodes also Methionine (a very sharp encoding with only one codon), while the stop signal is external to the code and uses 3 different codons. We can easily realize that stop cannot be encoded by using a codon for one amino acid. In fact, such a choice would reduce the number of effective amino acids. But, why not employ specific codons only for marking the beginning of translation?

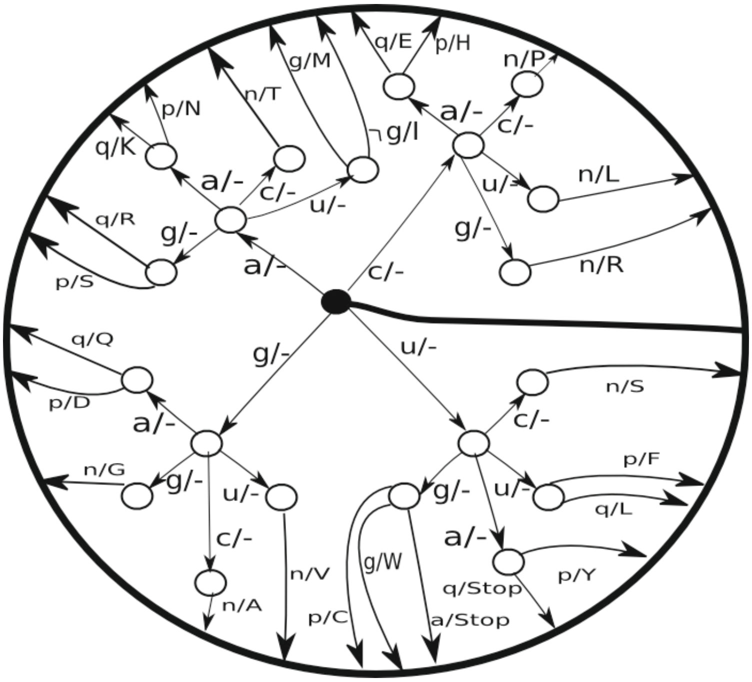


Fig. 4.2 The transducer realized by ribosomes

What abstractly we indicated I-molecules correspond to genes, while M-molecules correspond to proteins. Genes and proteins interact in the interplay between the processes of translation from I-molecules to M-molecules and the regulation role of M-molecules in the translation of I-molecules. In fact, a specific enzyme (in principle could be many enzymes) performs the transcription from I-molecules into an intermediate form of molecules, which in real cells are RNA molecules, that we can abstractly indicate as IM-molecules. RNA molecules surely, in the first stages of cell evolution, were able to play both the roles of I-molecules and M-molecules. This

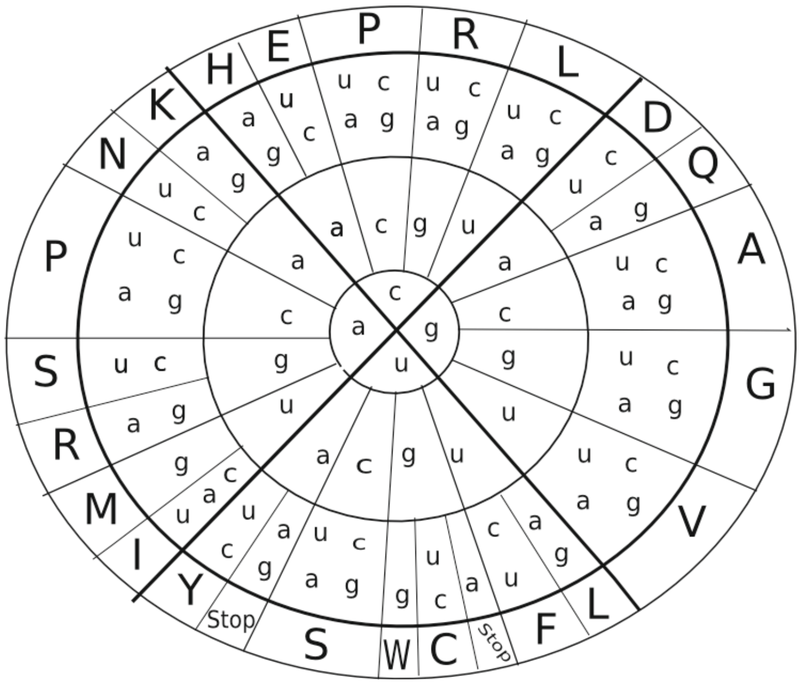


Fig. 4.3 A (circle) tree representing the genetic code

intermediate role is crucial for a clear distinction between information and action. The passage from an informative level to the active level cannot be direct and automatic, it has to be modulated according to the specific context and the global state of the cell, because no any kind of information has to be important/relevant/useful in any situation. The IM-molecules realize the need of making active, in a given context, the information stored in DNA. When IM-molecules are produced, some of them are R-molecules having specific regulation functions, while others are converted into M-molecules. Their conversion into M-molecules is automatically performed by specific transducers, or T-molecules (which in real cells are ribosomes, molecular machine constituted by RNA and proteins). Therefore, some M-molecules play specific roles among the seven functions of proteins, but a crucial role is the regulation role activating the transcription phenomenon. This means that some M-molecules determine the active state of I-molecules and the speed/intensity of the transcriptional mechanism according to a complex network of interferences (possibly at many levels) which define a complex cycle of transcription regulation: the activation of I-molecules depends on the kinds and the amounts of M-molecules translated in the current context. But vice versa, the types and the amounts of M-molecules which are translated depend on the kinds of I-molecules activated and on the amounts of molecules (which are R-molecules or M-molecules) regulating their transcription. Another subtle point emerges from this abstract analysis. In fact, activation is a yes/no phenomenon, but the regulation of speed and efficiency of the transcription mechanism provides a quantitative modulation of this activation by means of the amounts of translated M-molecules.

Figure 4.4 shows the three steps of the passage from a gene to a protein generated from it. The first step, articulated in several phases (starting from DNA unpacking), is the transcription where *factors, activators, enhancers* make active the promoter region associated to the gene in order to start the *transcription* of one contiguous region of DNA into an RNA string, called pre-mRNA, by means of an enzyme, which in eukaryotic cells is RNA polymerase II and works inside the nucleus (where the genome of an eukaryote cell is). The starting point of the RNA polymerase transcription process, called TSS (*Transcription Start Site*), is characterized by many specific strings in its upstream proximal region (TATA box and analogous strings). Specific proteins (around 3,000 in the Human genome), together with RNA complexes, direct the entire process by means of a network of positive, negative, cooperative, or antagonistic interactions (promotion, repression, inhibition, induction, and activation) with many specific sites of regulation, distributed in a wide area that may vary from many thousands to around hundred thousand base pairs.

The transcribed pre-mRNA string is processed in a second phase, called *maturation*. In this phase, through a *splicing* operation, performed by *spliceosomes*, some substrings, called *exons* are selected and the remaining, called *introns* are removed. Then, the resulting string, called mRNA, is sent outside the nucleus in the cytoplasm. Here, the final step consists in *translating* a region between a prefix (5-Untranslated Terminal Region) and a suffix (3-Untranslated Terminal Region). Whenever the translation is performed, according to the genetic code, the resulting linear sequence of amino acids assumes a specific shape which determines the function of the obtained protein.

The complex interaction between genes and proteins is based on an intrinsic circularity. In fact, proteins or protein complexes that regulate gene transcription are coded by some specific genes. The overall networks of gene transcription regulation are based on a multiset of proteins, DNA and RNA strings with a composition that is specific for each gene. This multiset is responsible not only for the activation of the process, but establishes the main parameters of the process (speed, duration, etc.).

In this regulative mechanism, fundamental problems are still open about the mechanisms that determine the links between the coding regions and the related specific promoter regions (outside the transcribed region, and surprisingly enough, often, very far from the protein-coding region). We could suppose that a sort of **genomic code** has to be present, which so far is unknown, in order to direct the correspondence of promoting regions with related coding regions. Differently from the genetic code, a genomic code is specific to each genome (though general principles of organization can be reasonably assumed). In passing, probably many "organic codes" are waiting to be discovered in many biological contexts [118, 154, 151].

RNA molecules (in the many different forms, such as pre-mRNA, mRNA, snRNA, snoRNA, miRNA, lincRNA, siRNA, tRNR, rRNA, shRNA, and riboswitches) play crucial roles (often in molecular complexes with proteins) in the all the phases of the whole process of transcription/translation, which consists of a cycle (DNA $\rightarrow$ RNA $\rightarrow$ Proteins $\rightarrow$ DNA).

In the phase preceding the translation, other mechanisms of *post-transcriptional* modification may intervene that alter the mRNA in order to modulate or even block the normal translation (see, for example, Sect. 4.5). Analogously, even when the translation is realized, *post-translational* modification may further modulate the overall result of the long way from genes to proteins. The richness and flexibility of these collateral processes is probably the key to the role of **epigenetics**, concerning the processes that have no direct record in the genes, but modulate gene expression in correspondence to the different contexts of their realization, and moreover can acquire certain forms of biologic memory, with possible short-term hereditable characters.

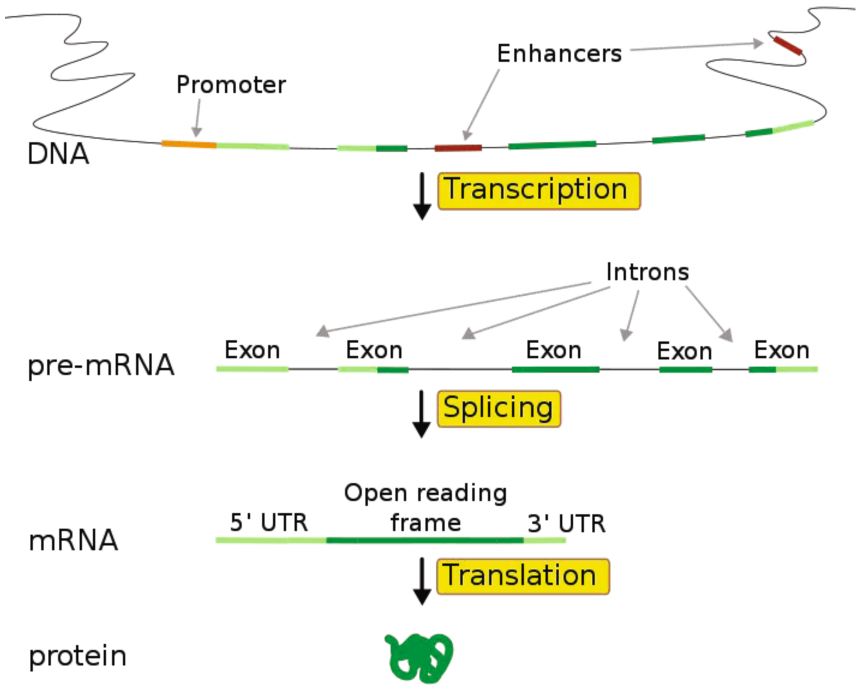


Fig. 4.4 From a gene to a protein via transcription, splicing, and translation

4.2 Bio-inspired Algorithms

Nature computes when it elaborates the biological information involved in complex processes by means of specific sophisticated mechanisms of control, coordination, organization, and finalization of particular objectives. As we have already noticed, natural computing can be seen as a perspective intrinsically related to computer science. If natural systems are the most complex systems we know, and if any complex system can work on the basis of a related system of information elaboration, the analysis of living phenomena from a computational perspective is appropriate in the same manner as it is appropriate to observe birds for designing flying objects.

In this section, we present some relevant algorithmic paradigms directly suggested by nature. The main stress of this analysis is on the algorithms rather than on the systems for realizing them. The latter pertain to "hardware", for example, DNA computing or Membrane computing, but also Neural Computing [9, 8], or Cellular automata (introduced by von Neumann) [161], while the former pertain to the methods and the paradigms for designing resolutive strategies for specific classes of problems.

The unifying aspects of all the methods we present briefly are the **population-based strategies**. In a sense, all the algorithmic methods suggested by nature have in common, even if at different levels, the notion of populations of individuals and the solutions are found by means of suitable processes pushing an initial population (of solutions or of solvers) to evolve toward a final state providing the solutions we search for.

These algorithms apply usually to problems of a combinatorial or optimization nature. In fact, their strategies are typical of situations which nature has to cope with, for finding the best cases satisfying the constraints imposed by life.

Evolutionary algorithms [127] follow a general (abstract) schema of resolution that is suggested by the basic mechanism of Darwin's natural selection.

Table 4.3 The general schema of an evolutionary algorithm

```
Initialize the population $S$ of solutions with a set $S_0$ of possible solutions
Evaluate a fitness level $F$ over $S$, and while $F$ is under a given threshold
  do
      Select a subset $S'$ of $S$
      Expand $S'$ into a population $S$ according to a generation mechanism
  done
Output the population $S$ reaching the fitness threshold.
```

The fitness function and the threshold value depend on the specific problem under investigation. However, crucial and particular features of evolutionary algorithms are the selective and the evolutive mechanisms. For example, each solution can provide new solutions by using casual changes or by specializing certain aspects, and then the solutions which solve, at least a given fraction of test cases, are selected.

Genetic algorithms are an important class of evolutionary algorithms, where the evolutive mechanism is realized by two steps: i) a **recombination** step where two (or more) individual solutions are mixed in some way for producing new solutions, and ii) a **mutation** step, where some individuals are casually changed in a certain small percentage. These two ingredients reflect the two main forces used by nature in genetic evolution. Of course, the manners and the extent of these two evolutive steps greatly influence the kind of results, and greatly depend on the context of application of the genetic algorithms we consider. Moreover, the representation of solutions determines many different forms of recombinations and mutations. Usually solutions can be represented by strings; therefore recombination corresponds to the ways of exchanging substrings, while mutations are casual replacement of single symbols (a certain number of them).

Memetic algorithms are a generalization of genetic algorithms, where the more abstract notion of **meme** replaces the notion of gene. Many possible interpretations of this concept, very often informally defined, have been developed. However, generally, a meme has a recombinant character and a mobility greater than those exhibited by a gene. In fact, a meme can be spread around in many copies and in many places, and can take part in mutation and recombinations with other memes where the number of copies, and the mutation and recombination extent are context dependent. For example, the number and the sizes of components it exchanges and/or the number of partners with which it interacts result from the "place" where the meme lives. The metaphor underlying memes is that of cultural units. They spread and evolve with the interactions among social groups which come in contact, by exchanging ideas, habits, words, mental attitudes, and so on. The composition of these groups constantly changes at a very quick rate. Formally, these features can be expressed in many ways, but their overall effect is to make the evolutive, selective, and mutational aspects more articulated and complex than the corresponding genetic mechanisms. This fact relates to the complexity and fluidity of cultural evolution with respect to the genetic evolution. A possible way of formalizing memetic mechanisms is provided by membrane structures, where we can assume genetic algorithms operating on internal membranes (genetic pools), and additional rules exchanging individuals and even sub-membranes, among different membranes, and where the levels of nested membranes could represent different levels of recombination.

Immunological algorithms are another type of evolutionary algorithms. Their specificity is based on the notions of **clonal selection** and **negative selection**, which are inspired by the corresponding theories developed since the mid 20th century in immunology. Clonal selection is realized when selection is performed by some recognition mechanism. For example, only solutions which match, to a certain degree, some target elements are produced in many copies and this confers to them more chance of becoming chosen in the further evolution process (this is the kind of selection B cells undergo by means of **antibody-antigen** matching). Negative selection occurs when the target elements which solutions have to satisfy constitute an *unknown space*, viewed as a set complement of some *known space* (this is the kind of selection T cells undergo, when apoptosis is induced in all the cells while the other cells survive to recognize the non-self elements as enemies.

Neural algorithms share with evolutionary algorithms the notion of fitness. In fact, they search for the best neural network connecting a given input/output correspondence. Any neuron is an element which is stimulated by inputs and reacts to them by producing outputs, generated by means of certain *reaction functions*, usually of sigmoid nature. Their outputs are propagated to neurons of a next level. The solution that a neural network reaches is the function that provides the outputs corresponding to a given class of inputs. The reaction function assigned to each neuron is chosen within a class of functions, but some parameters, usually called *weights*, express the specific kind of reactivity for the given neuron. Formally, if we assume n neurons for each of m levels, the neural network is represented by a matrix $n \times m$ of **weights** for the neural connection of the neuron to the neurons of the next level (zero weight means no connection).

In order to find the best neural network providing the best approximation to a given input/output behavior, some trainings with input/output cases are selected and the measure of the efficacy of the solution consists of the ability to provide the right correspondence for all the cases covered by the problem. A different class of bio-inspired algorithms are the **population-based cooperative algorithms** which are designed in a perspective where individuals are agents searching for the best solution to a given problem. In this case, solutions are searched through **cooperative and competitive search** or **cooperative and distributed generation** processes. In the first case, some agents move in the solution space by evaluating, at any step, their current solutions, by comparing them with the current solutions of other agents, and by using this comparison for developing a better current solution. The different classes of bio-inspired algorithms, realizing cooperative-competitive procedures, are characterized by different methods of cooperation-competition. The following are the general phases of these algorithms:

1. **Try**
2. **Compare**
3. **Modify**
4. **Communicate.**

The *communicate* module refers to the way agents share their knowledge in order to improve their current solution. Many schemata can be used for realizing this crucial aspect of cooperation. One possibility is based on some common space where periodically agents put their current solutions, and some specific mechanism, in this space, for assigning a score to all the solutions. In this case, any agent puts his solution in the common space if it is better than those which are published there. Another possibility is realized by means of a network, possibly dynamically defined, which ensures a communication between agents who ensure they update their knowledge and the consequent evaluation of their solutions. The agent who first reaches a solution with a given level of satisfaction publishes it and the process terminates. Therefore, in this case the publication of the solution coincides with the end of the process. Of course, many other intermediate possibilities can be conceived.

However, in both the previous cases, each agent searches for the global solution to the problem in question, therefore in this sense the cooperation is complementary to a competition, because each agent wants to win the "solver cup".

Algorithms based on strategies of cooperation and distribution try to assemble solutions by composing some partial solutions devolved to the single agents. In some cases the agent population is articulated in many levels or roles, and even the structure of a solution strategy can reflect this articulation, in such a way that any agent is required only for some steps and in different phases of construction of the solution. All these cases are strongly suggested by mechanisms found in nature (bacteria, ants, and bees).

A paradigm very useful in optimization problems is the **swarm intelligence**. It refers to situations where a collective solution emerges from many coordinated individual behaviors that follow simple rules of action and communication, usually driven by common finalities (reaching a place, searching for food, escape from a danger, and attack an enemy). Special cases of swarm intelligence are: ACO (Ant Colony Optimization), PSO (Particle Swarm Optimization), and FSO (Flock of Starlings Optimization). A similar idea inspired a computational model, called **Boids** that was elaborated in 1987 [149], where the flocking behavior of birds (in a stereotypical New York pronunciation boids corresponds to birds) is simulated by using very simple rules aimed at avoiding trajectory conflicts, at keeping spatial vicinity, and at sharing move direction.

Many aspects of animal intelligence, even if based on specific phenomena, are easily representable in computational terms. For example, the method of ACO (Ant Colony Optimization) is used for the search for a minimum path between two points. Each ant going from a point A to a point B follows, at first casually, an itinerary. However, during the journey each ant releases on the ground a substance, called a *pheromone*. If we assume that only some paths can be followed; of course the paths which are shorter are also the paths traversed by more ants and consequently having more pheromone along them. In this way, paths richer in this substance have a major score, and in this way the solution *emerges* automatically as a consequence of the population-based strategy the ants follow.

Apart from the application to optimization problems [148], swarm intelligence and similar paradigms could play a crucial role in the definition of algorithmic models of biological behaviors. In fact, if rules and computations can be established that are able to reproduce some behaviors, this can shed light on the possible internal mechanisms that are responsible of emergent functionalities in biological populations (of particles, cells, and animals).

In concluding the section, we mention the **self-assembly algorithms**, which apply a natural mechanism that is based on DNA computing algorithms and other algorithms which are inspired by DNA recombinant power. In this case, a solution is constructed by connecting some basic pieces which have an intrinsic tendency to assemble with other pieces, according to forces attracting them toward complementary pieces. In other words, any piece follows a local satisfaction principle which is reached when it combines with its correct companion. In this way, structures are generated spontaneously which provide some possible solutions for a given problem. Usually, this solution generation phase, due to the distributed nature and to the autonomous character of the elementary pieces, is very efficient and parallel. When the possible solutions are formed, a phase of solution selection starts for filtering the true solutions of the problem in question. This schema corresponds to the Adleman–Lipton extract model of DNA computing; however, it generally suggests possible applications which can be extended to many other situations. These methods become effective when the self-assembly generation phase is able to provide the whole (or a great part of the) solution space for a given problem. In this case, a trade-off between time and matter has to be evaluated. In fact, if we start with 20 elementary pieces where each piece has two different companions, the possible solutions are 2^{10}. This means that any elementary piece has to be provided with this multiplicity if we want to generate the complete space of the possible solutions.

4.3 Replication and Autopoiesis

As we have often remarked, metabolism and replication are the basis of life and the synchronization and cooperation of their activities is the key of life's birth. In this section, we want to stress again this point by using a metaphor which could illuminate the main aspects of their crucial relationship. Let us consider an airport and let us assume that according to some strategy of an economic, industrial, and social nature it was decided to replicate the same structure within a given time, say of some years, in the same region, but in a given place at a given distance from the existing airport. Firstly, this is a problem of a system replication, but not only, because it is also required that no service interruption can be tolerated, and moreover that the newborn airport has to work almost instantaneously, even with a gradual organization process, in such a way that, in a specified period, it has to be completely comparable with the airport from which it derives. Let us consider carefully how we will plan all this phenomenon, or better, how the original airport has to be organized in order to provide for the realization of its replication.

Let us analyze the main airport structures and functionalities, without aspiring to any completeness and adequacy regarding real cases, because we are mostly interested in the elements which are relevant to the chosen metaphor. In an airport we can distinguish a number of basic components and services going from the terminals, control towers, ramps, runways, taxiway, maneuvering and parking areas, passenger areas, cargo areas, passengers and personnel hosting, internal transport services, security, surveillance, maintenance, communication, etc.

All the services are controlled and directed by a central management which is at the top of a hierarchical structure with a complex network of interactions, and organized at different coordination and subordination levels. Moreover, a system of evaluation of the activities has to be defined. All the services are performed by operative units with an internal organization which depends on the type of service. The structures have a sort of operative kernel devoted to their maintenance, which is related to suitable stores.

In order to realize a well-functioning replication mechanism, each operative unit has to follow a strategy of acquisition of components over time, with a precise increment rate. The members which join a unit, since their employment, are linked to other members in order to learn from them specific tasks and skills, with the aim of reaching a skill level similar to that of their teachers. The components of groups are aware of their split, but nobody will know neither the exact time of this event, nor who will leave for the new airport and who will remain in the original one. This kind of ignorance, of the specific details about the separation, is important for avoiding bias in the way individuals are related to their groups. Of course, this ignorance is difficult to realize in the central management unit; therefore a casual mechanism has to be followed on which all the members have to agree. Moreover, a mirror of all the operative units has to be available to the management unit, in order to appropriately direct the splitting operation.

In conclusion, if we limit our analysis only to the human resources of an airport, we clearly distinguish: i) a personnel acquisition office, ii) a distributed system of personnel formation (transforming people in operative members), iii) an office for monitoring the state of affairs of all operative units, iv) an office controlling the right relationships among all the operative units, and v) an office providing solutions of conflict and emergency, which require a very quick redistribution of tasks and responsibilities. These offices are a sort of "meta operative unit" directly related to vi) the central management office and vii) the maintenance office, which has in charge the migration plan and the installation phase of the newborn airport.

In each group their oversized structure, due to the doubling of abilities, is in many aspects a way for having a more reliable and efficient realization of the specific tasks, but at the same time, it opens the possibility of conflicts due to situations of competition. This means that when the structure is big enough, it is at the right moment for a separation which provides a reset for a new deal in the life of the system. Therefore, when the split order is given by the central management unit, all the groups are split and move, in a specified number of days, to the place where it was decided to place the new airport.

Firstly, the structure builders are moved, with the suitable materials they need, which belong to the operative units for the maintenance of buildings and airport areas. Then, the groups move that are needed for activating the main structures. When they are in the new reality the basic functions are organized, by following the emergency protocols, which were used in the original system. An important aspect deserves particular attention, which is related to the specificity of the activities. At beginning, basic and generic tasks are required, therefore a sort of lack of specialization is required to individuals, in order to be able to carry out generic tasks. For example, a member operating surveillance of luggage, at beginning, when only a few flights are regularly established, probably has to be willing to perform other kinds of surveillance activities, when they are required by the superordinate units on which he/she depends.

As results from the previous metaphor, replication is related with two distinct, even if related, aspects: the replication of resources which are needed for assembling the new structure and its processes, but also an aspect which is crucial in another sense, different from that of replication of resources. In fact, the ability required by replication is essentially the same ability of a system which is aware of itself, of its structure and of the processes it hosts. This awareness is based on a sort of mirror of the global system (structures and activities) which has to be available to the management office of the system. This awareness is the basis of a self knowledge, and it is the same kind of awareness responsible for the ability of self maintenance. For this reason, any function has to be associated with an information which is an identification and description of it, in the context of the overall structure of the system. Self knowledge is the basis for **autopoiesis**, a concept originally introduced by Chilean biologists Humberto Maturana and Francisco Varela [142], as the main peculiarity of living organisms. A system which is capable of self maintenance, self restoration, and self construction, has to possess some kind of self knowledge, and for this reason the replication ability is based on the same mechanism which drives self control, by means of the double register information/function which life realizes by means of genes and proteins.

4.4 Main Informational Steps from LUCA to OVUM

In this section we will try to outline briefly the systemic logic driving some main passages from protocells to complex forms of life.

Let us consider a prototypal cell able to perform the main function of life. For it everything is minimal and this cell is able to survive and to produce offspring very similar to it. This process requires matter and energy, which are taken from the environment, and requires an incorporated knowledge, which is taken from the heredity, represented by molecules coming from the mother cell along the reproductive line to which the cell belongs. This is not only an individual process, but it is a process which has to be considered in the context of a population to which the cell belongs, and more precisely in the context of other populations which play concurrently the same game. This aspect is based on an intrinsic conflict. In fact, a population or a multi-population of individuals is a better guarantee that some individual could survive, but is also the reason for a competition among individuals competing for the same resources, for maintaining their life. If a perfect cell could exist (whatever "perfect" would mean), this cell would have a better life without any conflict, but cells, as any biological entity, are arrival points in an endless process of improvement, and a logic of populations seems to be necessary for implementing evolutionary strategies. In a sense, individuals need to be organized in populations because they are non "perfect", but their belonging to populations can improve their finalities. Of course, populations do not make sense without individuals, but at the same time, individuals are products of a population, and even their finalities emerge within the populations they belong to. It seems appropriate, in this context, the quotation of a maxim, taken from an African tradition, which claims that "a child needs a village for growing".

The duality individuals/population is probably the deepest mechanism underlying life phenomena, and is an internal force driving the aggregation/selection forces, from the subatomic particles to the universe galaxies. This duality is conjugated by the tendency toward the complexity of forms. In fact, the main seven functions of Table 4.4, which refer to cells, occur recurrently, and the same game of life starts again for complex organisms, plants, animals, and animal populations. At the end of a level of biological organization, new entities emerge which are the starting point of a further application of the same paradigm. This schema explains the logic underlying a lot of phenomena, going from morphogenesis, speciation, and sexual mating. These seven steps are basilar in the construction of living individualities, but they are always immersed in contexts of a population dynamics, where individualities define specific roles of interaction, cooperation, and competition. For any individual, the acquisition of the abilities of Table 4.4 is realized with conflicts, and individual death is an existential necessity. Paradoxically, death which is the antithesis of individual lives, is the key of the natural selection applied to populations.

Table 4.4 Seven basic functions of living organisms

Nutrition (environment resource acquisition)
Ingestion/digestion (acquired resource transformation)
Movement/autonomy (territory navigation and control)
Respiration (energy acquisition and transformation)
Differentiation (function specialization)
Morphogenesis (form acquisition)
Reproduction (heredity and replication)

Let us explain, in more details, the seven musts of Table 4.4 from an evolutive point of view. Of course, without taking matter from outside, no cell can survive. The first form of nutrient acquisition was possible in the surroundings of such nutrients. However, very often they were available where also enzymes were present which were able to produce them from some substances. Therefore, a natural passage to a more efficient form of nutrient acquisition was the ability of cells to acquire these enzymes in order to catch directly substances and to transform them internally.

This ability surely provided a great advantage, by removing the necessity of remaining near to nutrients and enzymes (in the case of heterotrophs), or of directly depending on the fluctuations of nutrient concentrations (in the case of autotrophs). In this way, (heterotroph) cells became open to a space exploration capacity; therefore this was the prerequisite for developing movement functionalities. From a "technical" point of view, the ingestion and digestion abilities require complex topological skills of membrane manipulations. In fact, the predation of small molecules can be obtained by channel mechanisms. But, for catching big molecules (or other cells), more dramatic methods need to be elaborated. This happens when a membrane realizes **invagination** and **phagocytosis**. In invagination, a part of a cell surface forms a concavity where an external body can be wrapped in order to pass through the cell skin within a vesicle (Figs. 4.5, 4.6, and 4.7). As depicted in Fig. 4.5, after wrapping the external object and after closing the concavity mouth, a separation of two membranes is realized: the original one from the newly formed internal membrane. In experiencing such a kind of mechanism, the cell acquired an ability which became essential in developing its symbiont activity, when it was able to catch other cells for devolving to them, under its control, specific tasks and converting its main activity to more directive roles rather than to operative roles. This step was essential in the passage from prokaryote to eukaryote cells, and follows the general principle, continuously used by life, of **reusing acquired competencies** for going on in the construction of more complex abilities.

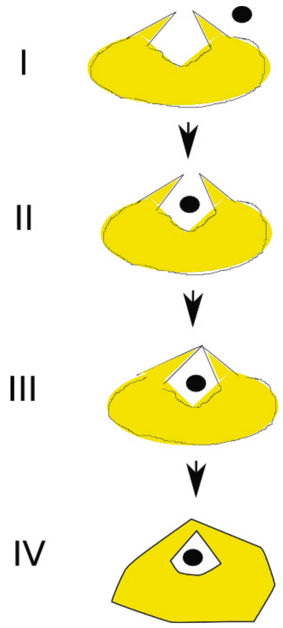


Fig. 4.5 Phagocytosis: Invagination (I), Predation (II), Endocytosis (III), Encapsulation (IV)

Phagocytosis is a crucial point in passing from the acquisition from the environment of small organic molecules, to the ability of acquiring bigger objects which do not pass through **ports** realized by **protein channels**. To this end, it is easy to realize that a basic mechanism of mitosis is reused for other purposes. In fact, as is shown in Fig. 4.6, when a cell separates in two distinct cells (mother and daughter cells) we have a situation depicted at the top of Fig. 4.6, which we call **devesiculation**, because the new cell is released in the environment as a separated entity. When, after invagination and the closure of the invagination mouth, the situation depicted at the bottom of Fig. 4.6 is realized, then we have a situation analogous to mitosis, with the difference that now the new formed vesicle can be released internally to the old one. For this reason, we call it **invesiculation**. In this case, apart from the realization of an invagination, some additional tools have to be used for performing the separation of vesicles. In fact, in the first case, as we know, by general physical laws about surface tension and minimal surface constraints, when the size of a vesicle grows over a certain limit, automatically the separation happens. For invesiculation, some cutting agents are required (and probably, other agents for closing the invagination mouth need to be considered). In Fig. 4.7, a model of invagination driven by receptor adherence is illustrated, which suggests the molecular basis of this phenomenon.

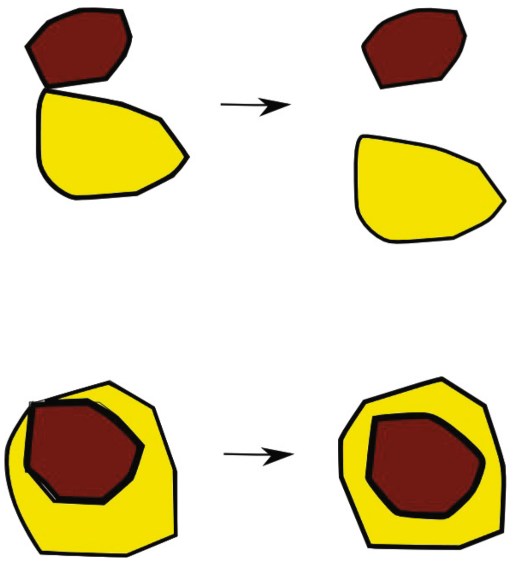


Fig. 4.6 Vesiculation: Devesiculation (top), Invesiculation (bottom)

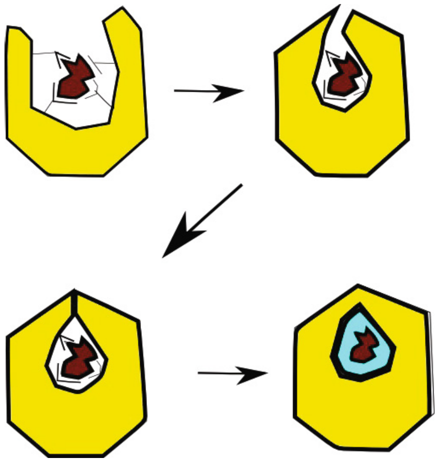


Fig. 4.7 Receptor driven invagination

As we already mentioned, in the next evolutive step, respiration was realized by extending the phagocytosis in a more elaborated way. In fact, the more probable event, allowing the passage from a **prokaryote** to an **eukaryote** cell, was the **symbiont** phenomenon where a cell captured an **aerobic** cell with a more efficient ATP production system based on **Oxygen metabolism** by devolving to the slave cell its energetic metabolism. In this case, the topological transformation of Fig. 4.5 was very appropriate, because the captured cell needs to be also compartmentalized in a way that its reproductive activity could be controlled by the capturing cell. For this reason, probably, a previous compartmentalization of the replicative DNA mechanisms was realized, that is, the separation between DNA replication, in the nucleus, and metabolism in the cytoplasm.

Differentiation has its deep roots in asymmetry. Even at molecule levels asymmetric forms naturally arise. Nucleotides are chiral objects and this is a fundamental feature for the realization of polymers conveying information. In terms of membrane structures, asymmetry is represented by **concentration gradients**. They arise naturally by consequence of general numeric properties of reaction fluxes. This result was pointed out in a seminal paper by Turing devoted to **morphogenesis**. However, two important factors pushing toward differentiation are the **pumps** and the **gradients**. Both are surely related to the input/output direction which drives the mechanisms of acquisition and expulsion of molecules, and to the organization of the internal space of cells.

When different cells are integrated in a population articulated in parts, but with a functional identity, we have a multicellular organism. The problem of this organization is how to reproduce this identity in a reliable and efficient manner. In abstract terms, this means: *how to memorize the knowledge underlying the biological structure of a multicellular organism, in order to be able to replicate it?* This is, first at all, an enormous informational problem. The solution which nature found is the discovery of a sort of **supercell**. This supercell is the **ovum**. It is a supercell because in its genetic memory there is a written record of how to reproduce the whole organism it belongs to.

Multicellularity could not be realized without this formidable biological discovery. From embryogenesis [163] we know that the construction of the organism from ovum (here there is no commitment with the sexual or asexual reproduction) follows a progressive differentiation, and this is the basis of the so-called **evo-devo** paradigm (the developmental itinerary resembles the evolutive one); therefore the ovum logic is strongly related to the differentiation logic and is the same kind of phenomenon underlying *stem cells*, **totipotent**, and cellular type reversibility and irreversibility, all concepts which are crucial in the new perspectives of many therapeutic approaches. The human body has around 250 main cellular types. They have the same genome, but only a part of it is active in each type. This is a great redundancy from the point of view of any system designer. But the motivation of this lack of economy depends on the evolutive logic of the differentiation process.

The **reusing of acquired competencies** principle, which we already noticed, is a general device driving the evolutive steps of life. However, another principle is more specifically involved in the differentiation and organization steps along the

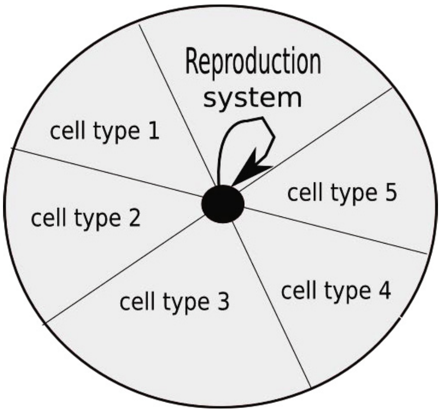


Fig. 4.8 The logical structure of multicellular organisms and the implicit tautology of the ovum's role

processes pushing toward more complex individual organisms. This principle, we could call it the **Recurrent Functionalities** principle, consists in the extension to cells of the functions of proteins. For example, the functions reported in Table 4.1 for proteins are essentially the same we may find in different types of cells of a multicellular organism. Thus, the same principle acts when different tissues and organs are assembled in complex living organisms, and again the same principle acts in the population of individuals, when societies are organized.

Table 4.5 Recurrent functionalities in life differentiation and organization phenomena

Transformation
Regulation
Recognition
Signal
Transport
Movement
Structure

In an evolutive perspective, the role of each actor is not established a priori, roles emerge when the players are ready to play them. In metaphorical terms, we could say that the plot of the play is written during its development, from the interaction with the actors when they are put together in situations which stimulate actions. Another metaphor of differentiation could be suggested by the computers which we use everyday. In a travel agency they are used for booking tickets, in an office for preparing documents, in a laboratory for elaborating data coming from experiments, and so on. However, the majority of them have an operating system and standard software packages, which are used in a very small percentage. This depends on the fact that it is easier to distribute computers with standard software packages, by leaving users to choose what they need. Passing to the biological level, why do genes include many portions which are transcripts, but are removed when they are translated, according to the well-known mechanism of the alternative splicing? Let us suggest a possible reason, which of course would need a biological validation. Namely, assume that in the course of bio-molecular evolution some DNA sequences were selected having some specific biological meaning. The aggregation principle toward complexity could have pushed these pieces to link in bigger sequences including them, and this mechanism could have used sequences having the function of connectors. When these pieces were joined, some new functions emerged, resulting from their combinations, but the connectors which had the merit of putting them together, in this new scenario, were useless or even disturbing, therefore an inverse process was activated for their removal (during the translation phase).

Gametes are the sexual discovery, which is another big step toward biological complexity. They are recombination biological machines, based on duality, which is a very recurrent biological paradigm.

Mind, consciousness, language, and society, belong to other levels, but are rooted in the protocell-multicell evolutive path.

4.5 Life Analysis and Synthesis

The short account of the previous section seeks only to outline a possible, informationally based, interpretation key of biological evolution. Many sub-steps could/should be considered, but the aim of these short remarks is to argue that logically based analyses could help in disclosing principles of life strategies. In any case, we can discover the key of life when we know how replication and metabolism interact. In other words, how is it logically possible to design functional units where these two processes interact by producing autonomous systems? How do they successfully replicate with the entirety of the processes they host, including replication?

The next step toward comprehension should concern a logical model of multicellular reproduction based on the ovum mechanism, which adds a further complexity level to the single cell miracle. Maybe, the logic which explains the replication/metabolism integration is the same which explains ovum/multicellular interaction, and maybe we need a new kind of logic for dealing adequately with these two problems. However, we want to stress that we need such a kind of investigation. In fact, it is impossible to reach a rational understanding, only by cumulating partial and detailed knowledge about single phenomena and specific aspects. Such marvelous devices, as cells and multicellular organisms, cannot exist without a unifying logic and general principles. Even if this logic results from an emergence phenomenon, this only changes the terms of the investigation, because the elements which underly it need to be *preformed* for allowing such a result.

Time and chance surely play a crucial role, and even if a very sophisticated and evolutionary process of self assembly is involved, some kinds of behavior attitudes and frames are built in the elements which participate in the life game, in order to make effective the role of time and chance. In his famous book *Science et hypothése*, the mathematician Henri Poincaré develops a beautiful reflection about chance. Very often we say that some events happen by chance when there is a gap between the causes of them and the effects that we observe. In fact, we "see" clearly the latter ones, but are not able to identify the former ones, which are usually a great number and on different scales of observability. The discovery of deterministic chaos, around the middle of the 20th century, is based on the phenomenon, called **initial condition sensitivity**, that occurs in certain dynamics. In such cases, although the evolution of a system is completely deterministic, it reacts, in an exponential way, to any perturbation affecting the state of the system. This implies that, even if we determine a certain dynamical parameter with a very high precision, we eventually cumulate such a big error that whatever we predict is completely useless. The only possibility of overcoming this limitation would be an infinite precision of that parameter determination, that, apart from its practical impossibility, would imply a computational power which cannot be reached by any computational system. This profound discovery is the last of a long series of limitative results which mathematics and mathematical logic discovered, such as the existence of unsolvable problems and of non-axiomatizable theories. The lesson of deterministic chaos points out the need for a new scientific attitude regarding predictability and scientific models. The so-called Laplacian paradigm of scientific explanation, was surprisingly applicable to classical physics. According to it, if the initial state of a system is given, by using a deterministic law ruling the dynamics of the system, we should be able to predict the behavior of the system at any future time. This powerful predictability cannot continue to hold for complex phenomena. Therefore, the conceptual analyses of complex phenomena, such as those occurring in living organisms, cannot expect, in general, to exhibit some kind of Laplacian predictability. This means that rational comprehension of biological phenomena does not imply the discovery of a machine which tells us what will happen in the future, but will disclose, when it is good enough, some internal relations which explain how a given system works, and could help us in the evaluation of parameters adding new levels of knowledge about life phenomena. A cell is a system where many coordinated molecular events produce macroscopic effects. Therefore, the logic of cells is the logic which underlies this coordination. Discovering it has to be surely based on the biological and experimental evidence, but there is the need of logical, mathematical, and rational speculation which cannot be disregarded if we hope that such an enterprise could be successful.

An important point concerning biological analyses, concerns a clear distinction between description and explanation. The exact knowledge of the parts of a system and of the processes they participate in does not imply an understanding of the phenomenon. Paradoxically, too many details could hide its real logic. Abstraction is not common in biological investigations, while the descriptive approach has dominated the life sciences. Finally, the possibility of discovering principles, independently from the biological level, is an important issue to take into account. For example, the logic of the human immune system, which is a very complex informational device, is probably based on security mechanisms and dynamical memories, therefore a rigorous approach, in terms of abstract concepts developed in this field, could provide fundamental clues for a new understanding of their components and their connections with other biological levels.

Formal analyses of basic life mechanisms are strictly related to the computational approaches. In fact, the discrete nature of basic molecular dynamics and their complexity naturally requires algorithmic formulations of phenomena. Moreover, computational simulations and analyses are essential tools when they are hard or even impossible to reproduce in laboratories. In some cases the outcomes of computational experiments can suggest biological experiments, or could evaluate, exclude, or confirm some possibilities which are expected in a given conceptual explicative framework.

Synthetic biology, which is often advantageously associated to the computational approaches, is a recent perspective of biological investigation which seems to be very promising in the near future. The first objective of this discipline is the synthesis of simplified cell structures, which can be components of **minimal cells**, that is, living organisms, according to some minimal definition of life. In this context, the notion of **ribocell** was elaborated [143, 152], which is constituted by a lipid vesicle containing: i) nucleotides, ii) a ribozyme which can catalyze its own polymerization, by means of an RNA template, and iii) another enzyme able to synthesize the membrane lipids from some suitable precursor molecules. This cell could exhibit the capacity of self reproduction, in the moment that their synthesis reactions are synchronized in the right way with their ribozymes replication. Some computational experiments show that this possibility is coherent with the data coming from the known mathematical models of synthesis reactions (differential models). The membrane is essentially regulated by a mechanism which, in a simple way, can be expressed in terms of **relative surface**. If V is the volume of a cell and S is its surface, then its **relative surface** $\boldsymbol{\Phi}$ is the ratio between S and the surface of a sphere having volume V. In fact, the sphere of radius $\sqrt[3]{\frac{V}{\pi}\frac{3}{4}}$ is the solid with the minimal surface comprising the volume V, and the tension/stretch profile in the lipid bilayer, originated by the membrane curvature plays a fundamental role in the mechanosensitivity of the cell. When the relative surface $\boldsymbol{\Phi}$ is equal to 1, then the cell is a perfect sphere, while if $\boldsymbol{\Phi} < 1$ the membrane is subjected to a tension, and if $\boldsymbol{\Phi} > 1$, then the membrane surface is relaxed and can be split. The tolerance for the membrane tension can be expressed by the value ε such that under $1 - \varepsilon$ a crisis occurs and the surface breaks. It depends on the kind of lipid structure. For example in oleic acid, which is used in ribocell experiments, $\varepsilon = 0.21$. A sphere of radius R has the following volume:

$$V = \frac{4}{3}\pi R^3$$

therefore:

$$R = \sqrt[3]{\frac{V}{\pi}\frac{3}{4}}$$

and its surface, expressed by means of V is:

$$S = 4\pi R^2 = 4\pi \sqrt[3]{\frac{V^2}{\pi^2}\frac{9}{16}} = \sqrt[3]{36\pi V^2}.$$

A reasonable value for having a split is when $\Phi = \sqrt[3]{2}$, in fact in this case:

$$\sqrt[3]{2} = S/\sqrt[3]{36\pi V^2}$$

that is:

$$\begin{aligned} S &= \sqrt[3]{2 \times 36\pi V^2} = \\ &= \sqrt[3]{8 \times 36\pi (V/2)^2} = \\ &= 2\sqrt[3]{36\pi (V/2)^2} \end{aligned}$$

which means that the surface of the cell corresponds to that of two spheres having volume $V/2$; therefore there is enough surface for splitting the volume in two equal spheres.

In the attempt to realize minimal cells having autopoietic ability, the starting point is the capacity to produce lipid vesicles including the basic ingredients of biochemical dynamics: water, DNA and RNA nucleotides and oligo-polymers, enzymes, ribozymes, peptides, PNA, catalysts, and so on. Of course, interesting byproducts, of biomedical interest, of these researches are related to crucial mechanisms in innovative therapies and drug development. The main objectives in this field concern the realization within artificial vesicles of the following functionalities.

1. Realization of basic metabolic functionalities, and specific resources acquisition expulsion;
2. Biochemical energy production;
3. Specific protein expressions mechanisms with some given context dependencies;
4. Complex vesicle internal articulations;
5. Realization of sensorial abilities with activation of environment response mechanisms;
6. Communication among synthetic cells;
7. Movement and mechanical activities.

In a wide sense, the essence of this synthetic trend is the realization of supramolecular biochemical machines with specific functions, similar to those performed by natural cells, capable of self assembly, and self organization.

Another recent trend in synthetic biology was inaugurated on May 20th 2010 by J. Craig Venter, the same researcher who first completed the human genome sequencing. The experiment reported by Venter consists first in the synthetic assembly of the entire genome of a bacterium, Mycoplasma mycoides. For security reasons, some genes were removed, for controlling possible unexpected problems, and other genes were introduced for expressing proteins which confer a blue color to the bacterium colonies. Then, this synthetic genome was replaced in another bacterium, Mycoplasma capricolum, of the same family but with a genetic distance from the M. mycoides, comparable to the distance between human and mouse genomes (this bacterium was made DNA-free, by using a technique already available). The resulting artificial cell was named Mycoplasma mycoides JCVI-syn1.0. The new cell was capable of self reproduction (a colony with blue color was obtained) and, as expected, it expressed the same proteins of the M. mycoides. In conclusion, the main relevant facts were that: i) the genome of M. mycoides JCVI-syn1.0 was synthetically assembled (with small changes), ii) M. mycoides JCVI-syn1.0 was a living organism, iii) M. mycoides JCVI-syn1.0's life was driven by the replaced genome which proved to be the real "operating system" of the cell. The crucial point of the experiment was the M. mycoides JCVI-syn1.0's start-up. In fact, as we know from computer operating systems, in order to make a system really operative for a machine, a small set of instructions has to be executed which passes control to the operating system, by activating its execution. This is a key point of the potency/act passage, and it is not clear, as far as we know, what is the biomolecular basis of this delicate passage. Of course, other steps need to be realized for a complete acquisition of the conceptual framework underlying this experiment of synthetic biology, but in any case, this seems to be a great achievement toward a comprehension of life by means of synthetic biology.

Cell genomes strongly resemble computer operating systems. Even in terms of digital size, there is a remarkable similarity between the number of bytes of genomes of complex organisms and the number of bytes of modern computer operating systems. If it is true that genomes establish how cells work, it is also true that the ways they work are not completely determined by them. Therefore, the problem remains of a better understanding of the relationship between these programs and their execution. In fact, the same genomes (genotypes) can produce different observable aspects (phenotypes). Surely, the genetic determinism is not so strict as the analogous determinism between computer programs and their executions. This suggests the research of models which could disclose a more complex dialectics relating genotypes with phenotypes. This is a key point for another aspect that is fundamental in life strategy: the relationship between **programmability and evolvability**. The expressions of genes (the programs executed in a given situation) result from a complex network of interactions where external inputs play a crucial role. This means that, analogously with complex computer programs, many conditions on control variables influence the kinds of executions which can be performed. The simplest way of imagining such a phenomenon is based on instructions which, at runtime, instantiate the value of certain variables by values taken from external input channels, therefore, if conditional instructions such as "if X= ... then do ... else ... " are present in the program, then the contingent value of such control variables can dramatically change the observable program executions. Everything which influences the realization of genetic programs, but is not directly related to genes, is defined as **epigenetics**. However, this is only a negative definition, which does not say anything about the internal mechanism which underlies the phenomenon of gene expression and regulation and the way phenotypical patterns emerge from them [130, 153, 144, 119, 121, 132]. The model based on the external variable checkpoints is probably too simplified.

The essential point here is probably related to some forms of biological memories which are not genetically driven. Of course, different phenotypes determine different individuals, and different individuals determine different ways which the natural selection can act on them. However, are there some kind of execution memories which add kinds of articulation to this basic evolutive dynamics?

These questions raise, in biological terms, a problematic which relates to well-known philosophical debates going back to Aristotle, full of implications with many contexts and disciplines. In this sense, the dichotomy genetics/epigenetics relates to Aristotle's potency/act or potentiality/actuality distinction, and more recently to René Thom's salience/pregnance in the form emergence and development [156]. A form emerges in a context which is necessary to its identification. An object in space is a form of discontinuity, as different from what is external to it, but at the same time, a form becomes a center of pertinent relations which determine its function giving sense to its existence. If genes encode forms, the functions they activate are related to the forms they provide, after transcription and translation, and to the forms which are involved in these processes. Here an essential difference with programs is due to the conformational, geometric, and physical character of the forms these programs realize. In this sense the dialectics genetic/epigenetic links naturally to **morphogenesis**, **embryogenesis**, and developmental mechanisms of form emergence in biological organisms. Perhaps, a unified logic underlies these phenomena, and discovering the main rules of this logic will be a great gain in knowledge of life.

An important field of application of synthetic biology is the so-called **gene therapy**. This means synthetic methods that transform portions of genomes, aimed at contrasting or blocking pathological phenomena. In particular, oncolytic adenoviruses are an emerging therapeutic approach for cancer [124, 123, 120, 159, 157, 135, 136]. The general schema of this method is based on the specific relationship between a gene and its promoter region. An activation *regulative configuration* is realized in the promotion region of a gene, in terms of protein and regulative RNA fragments binding to sites (at the end of a complex network of intermediate levels, where for example, A promotes B that inhibits C, and so on, through a chain of effects and counter-effects). When this happens, then the transcription process of the corresponding gene starts, in several phases, from the DNA unpacking to the gene transcription into pre-mRNA (performed by RNA-polymerase enzyme of a specific type).

Let us consider an adenovirus $\mathbb{A}$ having a gene X controlling the virus proliferation in the infected cells (causing their destruction). The cancer cells are the selective target of the therapy, by avoiding, at same time, the viral effect against the normal cells. Let us assume that some cancer cells express a specific protein, for example a marker of cancer proliferation. Let us suppose also, for the sake of simplification, that this protein occurs only in these cancer cells. Let Y be the gene that expresses this marker protein. Now, let us replace the promoter region P(X) of the gene X of $\mathbb{A}$ with the promoter region P(Y) of the gene Y.

In this synthetic virus $\mathbb{A}'$, the gene X is promoted by P(Y). Therefore, if a patient with cancer is infected by the engineered virus $\mathbb{A}'$, then it remains silent in the normal cells, because its promoter is absent and the regulative configuration

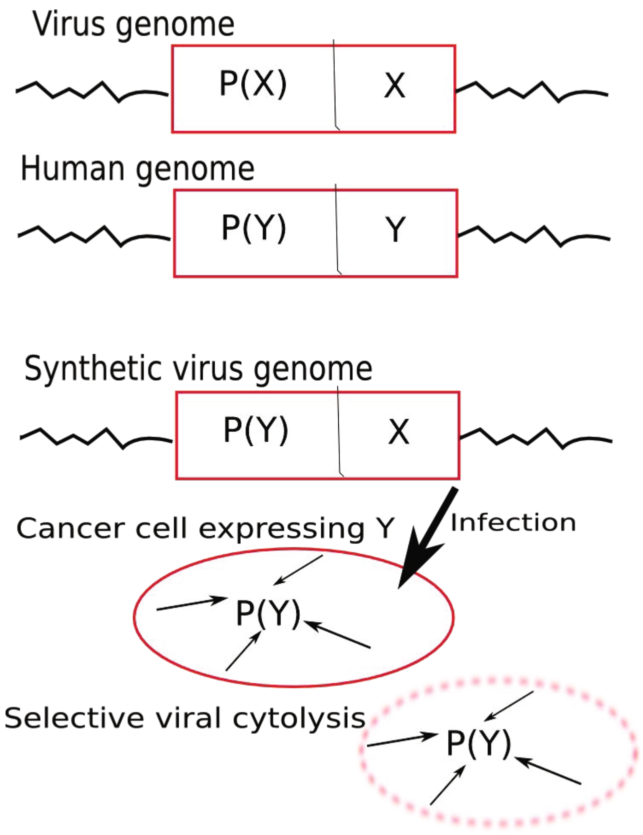


Fig. 4.9 The mechanism of oncolytic gene therapy

promoting *Y* is not active in these cells. But, when the engineered virus $\mathbb{A}'$ infects cancer cells, then the biomolecular setting of these cells makes active P(Y) that promotes X. This means that virus $\mathbb{A}'$ become aggressive and its destructive activity kills the cancer cells hosting it (see Fig. 4.9). Of course, this is only an ideal behavior, but the logic of this mechanism, apart from its enormous clinical interest, discloses general principles of a crucial process of life.

In the case of clinical applications, unfortunately the expression of gene Y is not exclusive of cancer cells. A way for improving the selectivity of viral attack [136] is based on post-transcriptional modification, by using the mechanism of RNA silencing [119, 153, 130] of small fragments of RNA that interfere with transcripts (by pairing with portions of them) by contrasting or blocking the translation process of mRNA into proteins. In fact, let us choose some micro RNAs (miRNA) that are specific to some cells, say liver cells, which we want to protect from the virus attack, but where Y is produced, thus P(Y) is active. If many copies complementary to these RNA strands are inserted in the 3′-UTR region (Untranslated Terminal Region) of the gene X of $\mathbb{A}'$, even if the transcript of X is translated when $\mathbb{A}'$ is in the liver cells (where the P(Y) region is active), nevertheless when their terminal region is recognized by the liver specific miRNA, they activate a silencing process that prevents the translation of this transcript, therefore the gene X cannot be expressed. Silencing is a very complex process, studied some time ago in plants, which is very important in gene regulation and defence against viral infections (often based on double-stranded RNA vectors). Many biomolecular elements and complexes are involved in RNA silencing (RNAase, Dicer, Risc-complex, and argonaute protein), which are activated by the double RNA strands of miRNA paired to complementary target regions, and eventually cleaved in the process.

4.6 Life Evolution

The following are Darwin's main arguments at the basis of his evolution theory [125]. On putting them together, there follows a confutation of the fixity of biological species over time (a mathematical analysis of natural selection was developed by Fisher [131] who was also one of the founders of mathematical statistics).

- **Geology**
  Geology tells us that earth changes and consequently life environments can differ dramatically in different epochs.
- **Fitness**
  Malthusian growth of biological populations shows that, when there is a competition for resources, individuals with characters which better fit the environment have more chance of surviving.
- **Heredity**
  Genealogical evidence and breeders' experience show that the characters of parents are transmitted to the offspring (at that time, Darwin was not aware of Mendel's experiments).
- **Chance**
  The appearance of characters which are selected is not forced by their increased fitness, but they occur freely by chance, and then they are kept by natural selection.

The *Geology* argument was an important start of Darwin's reflection and its main value was the correct temporal scale on which the dynamics of biological characters should be considered. Arguments *Geology+Fitting* imply **that** species change, while *Heredity+Chance* imply **how** they do it. Darwin developed a theory where the four statements above prove to be integrated in a coherent conceptual framework, which he supported by means of the findings he collected during his famous scientific journey. But, it is important to stress that the motivations for data supporting his claims followed the logical analysis he started from. In this sense, we can compare the intellectual strength of Darwinian theory of evolution, to Galieo's mental experiment about falling bodies, which marked the birth of modern mechanics.

It is surprising that in the *Geology+Fitting+Heredity+Chance* argument, heredity, which is the mechanism for transmitting biological information, represents also the way in which biological innovation is realized by means of chance. Maybe, as

Einstein said, God does not plays dice, but surely Nature does it. Evolutionary dice are Mutation and Recombination, they are the tools for exploring biological possibilities. These two golden rules can be implemented in many ways, for example, by replication errors and by sexual mating. However, what is revolutionary in Darwin's approach is that the free creative role of chance transforms a feature of individual continuity into a source of innovation for species. By reversing Lamark's perspective, Darwin disregards any passage from the individual biological experience, to the species. The biological memory gained by the species passes to the individuals (by means of genomes, in modern terms), but the reverse does not apply. This means that if one animal acquires a long neck by trying to eat the best leaves, which reside on the top of some high tree, then this character does not pass to his offspring, unless it is written in his biological memory since his birth. In fact, the giraffe got a long neck not because of the attempts he or his parents made, but on the contrary, the long neck occurred, by chance, in his genetic characters, and for this reason this animal had a better chance of surviving in places where high trees (with good leaves on the top) are more available than other kinds of vegetable foods.

Species are abstract entities which extract biological population characters present in reproductive lineages. They are archetypal forms of life, which are subjected to their own dynamical laws, Species originate when a sufficient number of characters is cumulated (a clear understanding of what sufficient means, is matter of investigation), which determines a reproductive lineage which separates it from the lineage where it occurs. It is very difficult to find rigorous definitions and models of these intuitions. From Linnaeus's approach to the classification of living organisms, to the modern molecular approach based on genome sequencing, and to the recent proposals of genetic barcoding, enormous problems remain for a clear understanding of the relationship between individuals and species. Perhaps, a better solution to this problem has to cope with more complex relations of membership between instances and genera, within a gamma of levels, where sharp and univocal classifications need to be abandoned. This is, in many aspects, a mathematical and computational problem, where phylogenetic distances, clustering of strings, sequence fingerprints, and genomic informational indexes could be applied in very specific and sophisticated ways.

4.7 Time's Arrow and Complexity

Time is experienced directly by living organisms as the dimension along which life dynamics develops with an evident irreversible character. A life starts with a birth and ends with a death. Maybe humans are the only living organisms with a clear awareness of this, but no living organism can escape this law. Life flows along a direction going from birth to death and all life events are arranged in this order. This notion of time is postulated by Darwin's evolution theory and, along the course of time, life follows a process going from the appearance of a prebiotic stage to the complex multicellular organisms, and to the complex societies of human civilizations. This scenario is so familiar to all of us, that it can hardly be surprising for the layman.

However, from a scientific point of view the same scenario raises a paradox so difficult to analyze and explain, that the most well versed minds of all times were very often lost in the enormous logical difficulties related to **time's arrow**. This paradox links two scientists who enormously influenced their disciplines, biology and physics, between the mid 19th and 20th centuries: Charles Darwin and Ludwig Boltzmann. The first chapter of Wiener's famous book, *Cybernetics*, is entitled *Newtonian and Bergsonian time* and a problematic conciliation is discussed between the two opposite notions of time in biology and physics. A conciliation which surely exists, because biology cannot falsify physics on which it is surely based.

The time of classical physics, which successfully explained star and planet motions, and which is the basis of the modern science inaugurated by Galileo, is a time where laws are expressed by equations, and therefore where no privileged verse can be chosen. But people are getting older, and no reliable method has been discovered, so far, for stopping or inverting the aging process. A notion of irreversibility appeared in physical phenomena concerning thermodynamics. The second principle of thermodynamics establishes an energy degradation principle pushing isolated systems, toward states where thermic energy is uniformly distributed. This raised the problem of relating this principle with classical Newtonian physics, where no time arrow is postulated. Boltzmann, surely influenced by Darwin, dedicated his life to the explanation of the thermodynamical arrow of time. Boltzmann's research was greatly opposed by the leading scientific culture of his time, and the mathematical proof he presented was criticized in a very aggressive way, surely because of some technical inadequacies, but essentially because of its revolutionary scientific meaning. The intrinsic logical difficulty of Boltzmann's theory was put in evidence by Henry Poincaré (*we cannot accept an argument where the conclusion is in a clear contradiction with its premises*). Surely, Boltzmann was on the edge of the paradox, but he was right, as the following science confirmed his intuition. Moreover, the physical theory that he founded, statistical mechanics, is a masterpiece of all the physics of the 20th century.

A more subtle paradox is related to the two kinds of time of classical mechanics and of biology. In fact, even if the Boltzmann theory is accepted and thermodynamics is conciliated within Newtonian mechanics, even in this case, how can we explain the dichotomy between the thermodynamic arrow and the time of biological evolution, where living organisms evolve, against the order degradation required by the second principle of thermodynamics? In other words, even if time's arrow is explained, two opposite arrows rule inorganic and organic phenomena, respectively. Prigogine's theory of non-equilibrium thermodynamics tries to explain this apparent contradiction, but no definitive explanation seems to be clearly settled. In the following, we will try to explain the relationship between time and complexity. We will develop arguments and (numerical) experiments showing that time's arrow is a consequence of statistical complexity. In the analysis that we develop some crucial interactions appear among time, complexity, chance, and information.

The H-theorem was proved by Boltzmann in 1872. It introduces the function H, which appears to predict an irreversible increase in entropy, despite the microscopically reversible dynamics of thermodynamical systems. This conclusion was considered paradoxical by many authoritative thinkers of that time. But many criticisms missed some deep aspects of the question. The main point is that in a population dynamics, resulting from a huge number of individual dynamics, some properties emerge which are new with respect to the simple additions of the individual effects. In other words, in molecule populations a temporal asymmetry emerges which breaks the time symmetry. Today, many proofs are available of the H theorem. In this section we will define and interpret some numerical experiments which confirm Boltzmann's conclusion and suggest the statistical and informational character of the thermodynamical arrow of time.

Given a partition of an interval in the real line, into subintervals of size τ (the discretization approximation), we call (discrete) **distribution** any finite population of (real) values, which are considered indiscernible when they belong to the same subinterval of the partition. A distribution, can be considered a (discrete) random variable where the *internal multiplicities* (how many values fall in each interval) provide the probability distribution of the random variable. In fact, a distribution of interval values $v_1, v_2, \ldots, v_m$ with multiplicities $k_1, k_2, \ldots, k_m$ respectively, where $k = k_1 + k_2 + \ldots + k_m$, determines the probability distribution $p_1 = k_1/k, p_2 = k_2/k, \ldots, p_m = k_m/k$.

Given a value distribution, then (discrete) Boltzmann's function H for this population has the following definition, where m is the number of subintervals which partition the distribution interval, and n_i is the number of values belonging to the $i-th$ subinterval.

$$H = \sum_{i=1}^{m} n_i \lg n_i.$$

The H function is related to Shannon's entropy S (already introduced by Gibbs for thermodynamical systems), and apart from additive and multiplicative constants they are functions of the same kind, but with opposite signs. Therefore if H decreases, then S increases. Both H and S easily extend to continuous variables and to continuous probability distributions by replacing sums with integrals. Very often symbol H denotes Shannon entropy. To avoid confusion, in this section, we reserve symbol H to Boltzmann's function and S to Shannon's entropy.

Let us consider an ideal gas where molecules can be assimilated to balls and their collisions are elastic collisions where the impulse and energy conservation laws hold. We can describe the evolution of such a gas in a simplified abstract two-dimensional setting. In fact, let us start from a distribution of values representing molecule velocities. Then, apply the following evolution rules which abstractly correspond to molecule collisions.

It is easy to realize that the game described in Table 4.6 represents, in a population of molecules, the collision of two molecules (two-dimensional balls), along a collision line (passing through the centers of the two balls), which, after the

Table 4.6 The Pythagorean recombination game

Randomly choose two numbers a, b of the given number population;
Randomly choose a number a_1, such $a_1 \leq a$, and split a into a_1 and $a_2 = \sqrt{a^2 - a_1^2}$;
Randomly choose a number b_1, such $b_1 \leq b$, and split b into b_1 and $b_2 = \sqrt{b^2 - b_1^2}$;
Replace the pair a, b with the pair $a' = \sqrt{a_1^2 + b_2^2}$, $b' = \sqrt{b_1^2 + a_2^2}$.

collision, exchange the components of their velocities along the collision line, by leaving unchanged the components orthogonal to that line.

Now if we apply the Pythagorean recombination game to a given distribution, along a number of steps we observe two facts: 1) the H function decreases (does not increase), and 2) the distribution approximates to a χ^2 distribution, which is typical of sum of squares of stochastic variables following normal distributions.

By using a simple MATLAB function, a series of numerical experiments were performed, some of which are reported in Figs. 4.10, 4.11, 4.12, 4.13, and 4.14.

We claim the validity of the following statement which corresponds to an abstract formulation of the H theorem.

Boltzmann's function H decreases, or does not increase, during the Pythagorean recombination game.

In particular, the H theorem follows from the Maxwell's velocity distribution theorem, by means of statistical and informational arguments. In fact, the normal distribution is the distribution holding for casual phenomena where a great number of independent causes influence their evolution. Therefore, it is reasonable to expect that after a great number of collisions the velocity distribution of each component follows a normal distribution. Moreover, given the nature of the collisions which follow the conservation principles of elastic collisions, we can assume that the variance of the distributions remains constant during the evolution. In fact, the sum of squares of two colliding "values" does not change after the application of a Pythagorean recombination (the velocities they represent exchange one component, see Table 4.6).

Therefore, starting from an initial value distribution, by applying the Pythagorean recombination game, we get distributions which keep the same variance of the initial probability distribution. Moreover, these distributions are the sum of the distributions of the two Pythagorean components, which for reasons of symmetry have the same variance.

From information theory, we can deduce that, given a random variable Z such that:

$$Z = \sqrt{X^2 + Y^2}$$

if X and Y are independent random variables, from the definitions of entropy, joint entropy, conditional entropy, and random variable independence, it follows that the entropy of Z is the sum of the entropies of its components X and Y.

According to Shannon's information theory, the entropy of probability distributions with a given variance reaches the maximum value in the normal distribution having that variance (see Table 4.8 for a proof in the continuous case). Therefore, the joint probability distribution of $Z = \sqrt{X^2 + Y^2}$ reaches its maximum, when both components X and Y reach their maxima. As we have shown above, the Pythagorean recombination game transforms a distribution in another one having the same variance, therefore if the distributions of the two Pythagorean components of these distributions evolve toward normal distributions, then, along the game, the H function evolves toward its minimum value (and S toward its maximum value).

The entropy of a normal distribution is $\frac{1}{2}\ln(2\pi e\sigma^2)$, as deduced in Table 4.7.

Table 4.7 The entropy of the Gaussian distribution N of mean 0 and variance σ^2

$$
\begin{aligned}
S(N) &= -\int_{-\infty}^{+\infty} N(x)\ln N(x)dx \\
&= \int_{-\infty}^{+\infty} -N(x)\ln \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}dx \\
&= \int_{-\infty}^{+\infty} -N(x)[-\frac{x^2}{2\sigma^2} - \ln\sqrt{2\pi\sigma^2}]dx \\
&= \int_{-\infty}^{+\infty} \frac{N(x)x^2}{2\sigma^2}dx + \ln\sqrt{2\pi\sigma^2}\int_{-\infty}^{+\infty} N(x)dx \\
&= \frac{E(x^2)}{2\sigma^2} + \ln\sqrt{2\pi\sigma^2}\cdot 1 \\
&= \tfrac{1}{2} + \tfrac{1}{2}\ln 2\pi\sigma^2 \\
&= \tfrac{1}{2}(1 + \ln 2\pi\sigma^2) \\
&= \tfrac{1}{2}(\ln e + \ln(2\pi\sigma^2)) \\
&= \tfrac{1}{2}(\ln(2\pi e\sigma^2)
\end{aligned}
$$

The **entropic divergence** (or *Kullback-Leibler divergence*) between two probability distributions p, q, denoted by $D(p \parallel q)$ is given by:

$$D(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

We know from information theory [198] that entropic divergence D between any two probability distributions is never negative. Moreover, it can be extended to continuous probability distributions in a natural way by setting (while keeping its non-negativity property):

$$D(p \parallel q) = \int_{-\infty}^{+\infty} p(x) \ln \frac{p(x)}{q(x)} dx.$$

On the basis of non-negativity of D, Table 4.8 shows that the normal distribution of variance σ^2 has the maximum entropy in the class of the probability distributions with the same variance.

Table 4.8 The continuous entropy of distributions with variance σ^2 reaches the maximum value for the normal distribution of variance σ^2 (f denotes any probability distribution of variance σ^2)

$D(f \parallel N)$	$= \int_{-\infty}^{+\infty} f(x) \ln \frac{f(x)}{N(x)} dx$	$N(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$
	$= \int_{-\infty}^{+\infty} f(x) \ln f(x) dx - \int_{-\infty}^{+\infty} f(x) \ln N(x) dx$	
	$= \int_{-\infty}^{+\infty} f(x) \ln f(x) dx - \int_{-\infty}^{+\infty} f(x) \ln \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} dx$	
	$= -S(f) - \int_{-\infty}^{+\infty} f(x) \ln e^{-\frac{x^2}{2\sigma^2}} dx +$ $\ln \sqrt{2\pi\sigma^2} \int_{-\infty}^{+\infty} f(x) dx$	
	$= -S(f) + \frac{1}{2\sigma^2} \int_{-\infty}^{+\infty} f(x) x^2 dx + \frac{1}{2} \ln 2\pi\sigma^2 \cdot 1$	
	$= -S(f) + \frac{1}{2\sigma^2} Var(f) + \frac{1}{2} \ln(2\pi\sigma^2)$	$Var(f) \leq \sigma^2$
	$\leq -S(f) + \frac{1}{2} + \frac{1}{2} \ln(2\pi\sigma^2)$	
	$= -S(f) + \frac{1}{2}(\ln e + \ln(2\pi\sigma^2))$	
	$D(f \Vert n) \leq -S(f) + \frac{1}{2} \ln(2\pi e\sigma^2)$	but $D(f \Vert N) \geq 0$
	therefore $S(f) \leq \frac{1}{2} \ln(2\pi e\sigma^2)$	

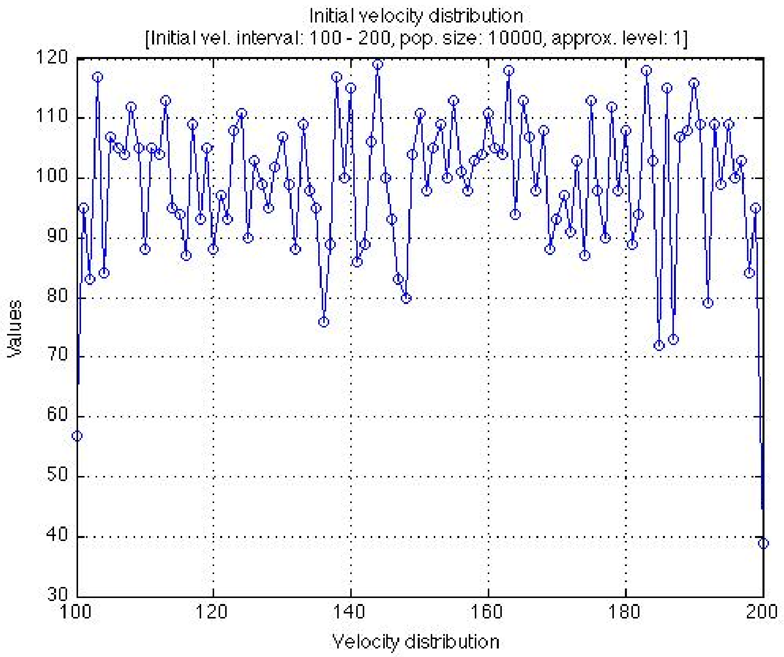
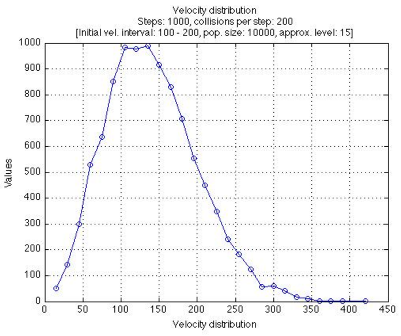


Fig. 4.10 Initial distribution, Interval 100–200, population size 10000



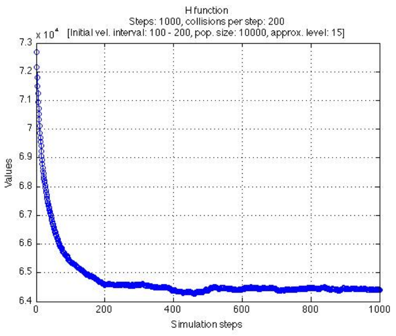


Fig. 4.11 1000 steps with 200 collisions per step, applied to the distribution of Fig. 4.10 with approximation level of 15

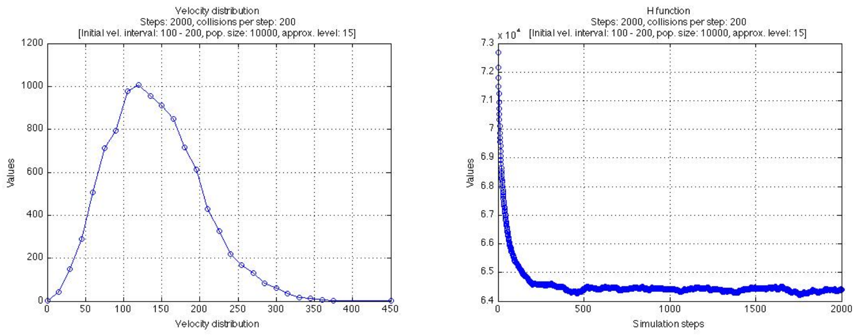


Fig. 4.12 2000 steps with 200 collisions per step, applied to the distribution of Fig. 4.10 with approximation level of 15

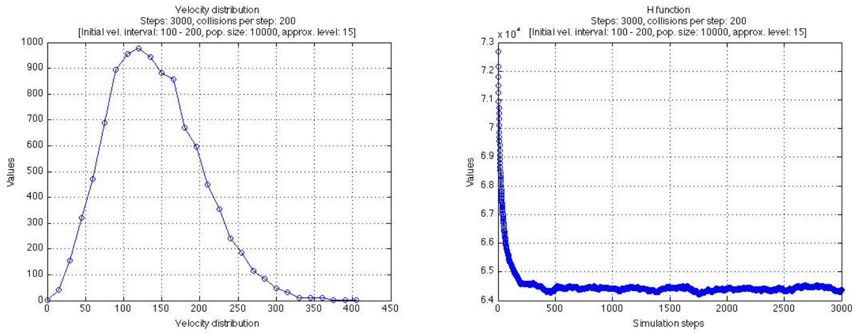


Fig. 4.13 3000 steps with 200 collisions per step, applied to the distribution of Fig. 4.10 with approximation level of 15

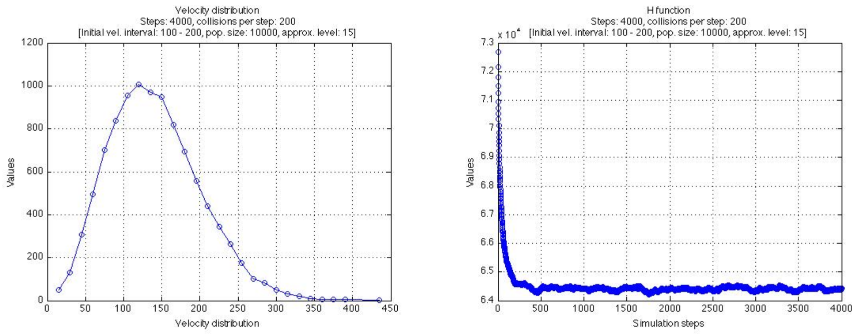


Fig. 4.14 4000 steps with 200 collisions per step, applied to the distribution of Fig. 4.10 with approximation level of 15

In conclusion, the Pythagorean recombination game pushes the system toward the maximum of entropy, because the more the game goes on, the more velocities approximate to normal distributions, which maximize the entropy, by keeping the initial variance (interactions casually increase and velocities depend on an increasing number of small independent effects). In fact, Figs. 4.10, 4.11, 4.12, 4.13, and 4.14 show that our pool of numbers (sums of squares of velocities) distribute according tho a χ^2 distribution.

This analysis shows that time's arrow is a population phenomenon strictly related to the complexity of systems, where the statistical and informational perspective transforms the reversibility of individual events into an irreversible population process. If we realize that also life is a population phenomenon based on bio-molecular dynamics, then we easily can deduce that life postulates a time arrow. However, despite this point of contact between thermodynamics and life, an essential difference is immediately evident. The thermodynamical complexity implies an irreversibility pushing an isolated system to an equilibrium state of maximum entropy, while life evolution generates an increasing complexity of living organisms pushing species to evolve toward increasing biological complexity, far from entropic maxima. Therefore, the two arrows follow opposite directions. The research for answering this kind of questions is crucial for a clear definition of biological complexity and for understanding the deep reasons on which the origin of life is based [122]. However, two important remarks are appropriate in this context: i) living organisms cannot be isolated systems, ii) in order to keep their dissipative character (assimilating from the environment and dispersing to it) they need to be far from thermodynamical equilibrium states. This non-equilibrium situation is the great discrimination between organic and nonorganic organizations, and introduces the time arrow opposite to the thermodynamical arrow, which however prevails when individual deaths occur, by restoring matter to the organic course.

Two giants of science, Galileo and Darwin are crucially involved in the scientific analysis of time. The former discovered the basis for a reliable measurement of time. The latter introduced a "historical" perspective in the scientific analysis of life. In fact, starting from considering living organisms according to suitable time scales, Darwin found an interpretation key of their internal organization and development in the framework of the evolution processes. Biological evolutions are population-based dynamics driven by environment, chance (mutation+recombination), and heredity. Heredity is, of course, an oriented transmission of information from a past to a future.

As we have shown in this section, Boltzmann introduced an orientation in the time of physics, by making the first step in the conciliation of Galileo's and Darwin's times. Certainly, other steps are necessary in this direction for a better understanding of time and life [126].

4.8 Life and Computation

Life and computation have a very long history of complex interactions. Many fundamental steps in the discovery of important principles and methods of computing machines are related to deep speculations about the typical natural processes involved in the elaboration of complex information. The main characteristics of theories of computation, developed in the last century, were the discrete character of dynamics underlying computation systems [139], different from the continuous notions dominating classical mathematical calculi, and their strong connections with biological systems. Probably, it is not an overestimation to claim that natural computing was the basis of modern theories of computations on which the computer science revolution of the last century was performed. In the nature/computation dialectics we may distinguish the following perspectives:

1. Computing by means of natural objects/phenomena (as in DNA Computing);
2. Computing by means of calculi inspired by natural objects/phenomena (as in Membrane Computing or Neural Computing);
3. Computing for reproducing typical phenomena of life, with no obligation of biological adherence (like self-replication phenomena by means of cellular automata);
4. Computing for analyzing or predicting biological phenomena (like metabolic behaviors by means of MP models or growth processes by means of L systems).

About the last point above, let us remarks some important aspects. A model is either good or bad only to the extent it helps us in predicting and explaining what we can observe. No other criterion can be discriminant, and it is ingenuous to adopt a mirror analogy of an absolute character. In fact, reality is different when it is considered at different levels of observation. A priori it is very hard to chose the "pertinent aspects" of a phenomenon and to disregard what is not relevant.

What is the adherence to reality in the physical theories at quantum levels, or at cosmological levels? What is the reality of the probability wave in Schrödinger equation? We trust them because they work. No mirror principle can assist us for their evaluation. Models are creations of human invention. Modeling is an art, and it cannot follow easily prefixed procedures. This art is based on the right guess of what has to be observed, what relationships are relevant between the observed features, how translate them in a chosen conceptual universe, and how to interpret the findings which result from this translation.

Let us consider a "logical" link between life and computation. The following is an equational version of a famous result of lambda-calculus, a formalism elaborated by Alonzo Church in the context of a foundational theory in mathematical logic. Consider a set S of *operators* where an *application* operation is defined, which will be denoted by concatenation. Then assume that three operators F, G, Y exist which satisfy the following equations for every $x \in$ S:

$$Gx = F(xx)$$

$$Y = GG.$$

From the previous equations the following equation holds:

$$Y = F(Y)$$

and consequently, for every $n \in \mathbb{N}$:

$$Y = F^n(Y).$$

This means that Y, also called Curry-Feys **paradoxical operator**, provides an infinite process of self-generation. The paradoxical nature of Y consists in its ability to exhibit the property of producing a computation with the only result of producing computations making the same task, along a non-terminating process which does not return any definite result.

This argument shows, in abstract, algebraic terms, that application and replication are the essential ingredients of an endless phenomenon of self-generation. However, an important issue follows directly from this analysis. In fact, a difference between life computations and artificial problem solving computations clearly arises. On the one hand, the computations performed for solving problems are terminating processes providing results when they stop. On the other hand, almost all computations performed in life processes are aimed at keeping their livening nature over time, by fulfilling some specific characteristics, but without terminating. Life in itself can be seen as an endless phenomenon propagating in space and in time and evolving into forms able to make this propagation ability more efficient and diversified.

In the development of modern computer science, four giants had a dominant role: Alan Turing, Nobert Wiener, Claude Shannon, and John von Neumann. Their major contributions were related to natural computing issues, along a network of ideas and formalisms developed by other scholars at the borders of many disciplines: mathematics, physics, logic, linguistics, and biology.

Alan Turing was the first scientist who elaborated in 1936 a general mathematical model of a computational device performing computations [214]. His model was inspired by a deep behavioral analysis of mind activity of a human agent performing calculations. A less known work of Turing was devoted to a mathematical analysis of morphogenesis, as a consequence of numerical phenomena ruling the threshold passage between different dynamical regimes described in terms of differential equations [158].

A joint work of Wiener with Rosenbluth, a Mexican physiologist, and Bigelow, an American electrical engineer [150] (Wiener studied mathematics under Russell's guidance) was entitled "Behavior, purpose and teleology" and was a philosophical starting point of his Cybernetics [160], a discipline he founded for studying, in a unified perspective, mechanisms of information processing in animal and artificial systems (see also [117]).

Claude Shannon founded the modern information theory in his famous booklet published in 1948 [212], "A mathematical theory of communication", where the mathematical analysis of quantitative principles of digital information was defined. It is not well-known that Shannon studied in his PhD thesis some methods for representing genetic information, while in his master's thesis he defined the electrical circuit representing boolean functions.

John von Neumann, surely influenced by Turing's model, elaborated in 1945 the project of the first modern computer, realizing the standard von Neumann's paradigm: EDVAC (Electronic Digital Variables Automatic Computer) [207]. The circuits performing the basic arithmetical operations were based on the McCulloch and Pitts model of neuron [9]. In this perspective, the electronic valve was seen as the basic device for representing the neuron. In von Neumann's terminology a neural network is a graph of E-elements. These elements are connected by edges of two different types: excitation and inhibition edges (with a small terminal circle) and are labeled with delays (expressed by angles in the middle of lines). A neural computation is performed by a network of E-elements which transform some boolean signals entering in input E-elements into signals exiting from output E-elements.

In Figs. 4.15 and 4.15 simple graphs of E-elements are given, from the initial section of von Neumann's report about EDVAC.

The thread connecting biological systems with computation can be found not only in the first stages of computer science, but remains a constant throughout its evolution. (see Kleene's famous paper "Representation of events in nerve nets and finite automata"). L systems were suggested by plant growth, and cellular automataare related to self-replicating systems (L systems are particular cellular automata). Splicing systems or H systems introduced by Head in 1987 represent genomic operations in terms of string rewriting, DNA computing extends the framework of formal language theory to double strings, and membrane computing extends string manipulations to multisets (strings with commutative concatenation).

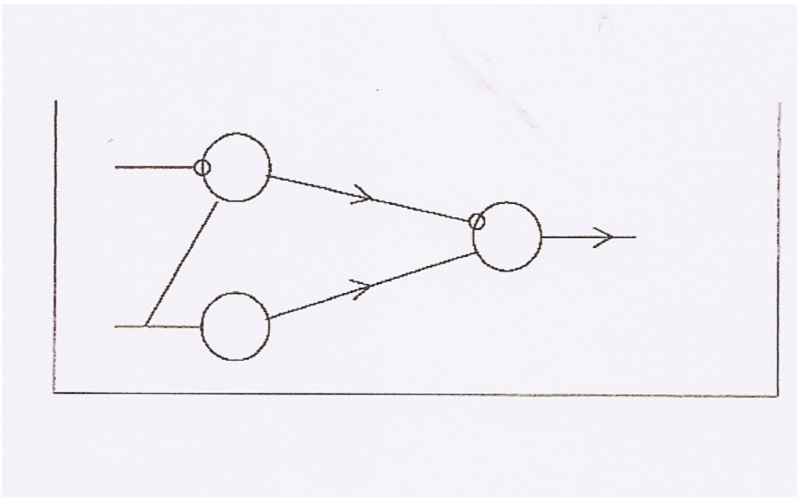


Fig. 4.15 A simple connections of three E-elements, taken from von Neumann's EDVAC report

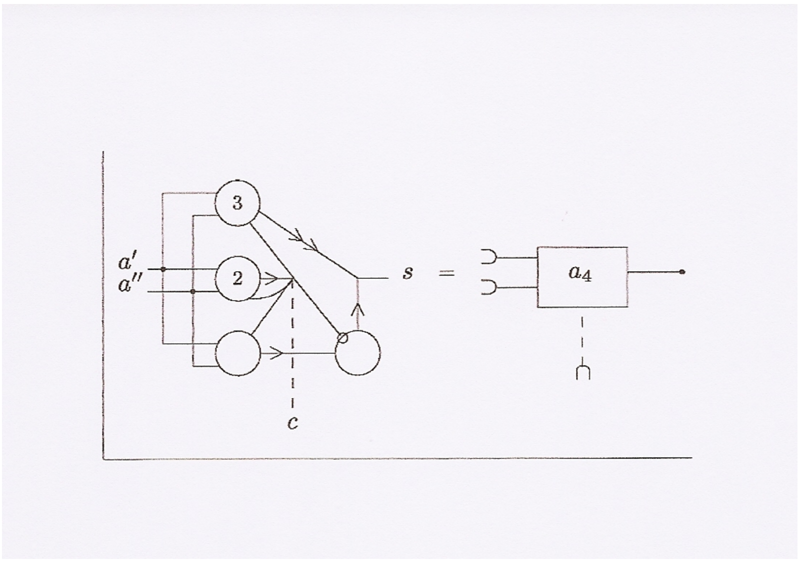
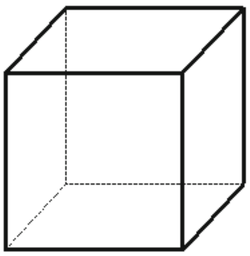


Fig. 4.16 The one-digit addition circuit, taken from the EDVAC report (its complex explanation is not reported here)

Part II
Discrete Mathematical Backgrounds



Platonic Esahedron

5 Numbers and Measures

Abstract. Numbers are the essence of any mathematization process. They measure entities, but also provide rigorous frameworks for analyzing the notion of infinity; the set of natural numbers is the first example of infinity. Finiteness and infinity, which very often appear in dialectic aporias, underly the essence of mathematics. Numbers are essential to the notion of population, and are intrinsically related to computation processes. In this chapter, after a brief section on sets and functions, the essentials of numerical systems will be outlined, by emphasizing the concepts that are most relevant for discrete structures. Then, the principle of induction, and basic topics of arithmetic and logic will be presented. Two sections conclude the chapter: one on series and growths (with numbers related to time, space, and matter aggregation), the other one on basic dynamical concepts.

5.1 Sets and Functions

Sets, numbers, sequences, relations, operations, functions, and variables are the basic concepts of mathematics, intrinsically intertwined and related to the infinite, a notion that is the essence of mathematics ("The Art of the Infinite" [179]). Here we give a brief presentation of these concepts and some standard notation (see [193, 174, 177, 187, 186, 169] for more extended or advanced presentations).

Sets or **classes** (in this context we use these terms as synonyms) are collections of distinct objects. The specification of all the elements which belong to a set completely identifies it. If X is a set, $a \in X$ means that a belongs to X ($a \notin X$ if a does not belong to X). If a set has a finite number of elements, it is completely described by a list, where the appearance order of the elements is not relevant, and multiple appearances are redundant, that is, equivalent to single appearances. A set $\{a,b\}$

DOI: 10.1007/978-3-642-36223-1_5 © Springer-Verlag Berlin Heidelberg 2013

of two elements is an unordered pair. There are many ways for introducing an order between the elements a, b of an unordered pair. A standard way to do it is by means of the set $\{a, \{a, b\}\}$, which is usually denoted by (a, b). A triple (a, b, c) can be identified by $((a, b), c)$, that is, an ordered pair where the first element is a pair too. In this way, any sequence of finite length can be obtained by iterating the set construction of (ordered) pairing.

Table 5.1 collects the standard set-theoretic concepts (sometimes set difference is also denoted by $/$).

Table 5.1 Fundamental set-theoretic notation

$\in$	Membership	$a \in B$	a is an element of A (a belongs to A)
$\subseteq$	Inclusion	$A \subseteq B$	All the elements of set A belong to set B
$\emptyset$	Empty set	$\emptyset \subseteq A$	$\emptyset$ is included in any set A
$\cup$	Union	$A \cup B$	The set of elements belonging to A or B
$\cap$	Intersection	$A \cap B$	The set of elements belonging to both sets A, B
$-$	Difference	$A - B$	The set of elements of A which do not belong to B
$\times$	Cartesian product	$A \times B$	The set of pairs (a, b) with $a \in A, b \in B$
$()^k$	k-power	A^k	The set of all k-sequences over A
$\mathbb{P}$	Powerset	$\mathbb{P}(A)$	The set of all the subsets of A
$\mathbb{N}$	Naturals	$\{0, 1, 2 \ldots\}$	The set of null and positive integers
$\mathbb{Z}$	Integers	$\{0, 1, 2. -1, -2 \ldots\}$	The set of integers
$\mathbb{Q}$	Rationals	$\{0, 1/2, 2/3. -1/2, -2/3 \ldots\}$	The set of rationals
$\mathbb{R}$	Reals	$\{0, 1, 1/2, \sqrt{2}, \ldots\}$	The set of reals
$\mathbb{C}$	$\mathbb{R} \times i\mathbb{R}$	$i\mathbb{R} = \{ix \mid x \in \mathbb{R}\}$	(The imaginary unit $i = \sqrt{-1}$)

5.1.1 Relations and Operations

The notion of sequence is connected with the mathematical concept of **relation**. For $k \in \mathbb{N}$, a k-sequence is a sequence of k elements. A k-relation (or a relation of k arguments) over a set X is a condition which either holds or does not hold for any given k-sequence of elements in X.

Given two sets A, B, the set of pairs (x, y), where $x \in A$ and $y \in B$, is the *cartesian product* of the two sets, denoted by $A \times B$. A subset of $A \times B$ is also called a **correspondence** between the sets A and B. The set of all k-sequences over a set A is denoted by A^k. Therefore, mathematically a k-relation R over A is a subset of A^k, in symbols, $R \subseteq A^k$. For example, the relation $\leq$ on natural numbers is identified by the set of pairs:

$$\{(x, x+y) \mid x, y \in \mathbb{N}\}.$$

Binary relations are an important type of relations. They are usually indicated by a symbol put between the two objects which are related. For example, the equality symbol $=$ expresses a binary relation between expressions such that $E_1 = E_2$ holds when E_1 and E_2 denote the same object. The order symbol $\leq$, when put between two numbers, expresses a relation which holds if the number on its left is less than or equal to the number on its right. Very often, when a binary relation does not hold its symbol is barred, for example, $a \not< b$ means that $a < b$ does not hold.

A binary relation R is said to be **reflexive** in a set A if aRa holds for all $a \in A$. R is **symmetric** on A if for every pair of elements $a, b \in A$, aRb implies that also bRa holds. R is **transitive** if for every triple of elements $a, b, c \in A$, if aRb and bRc hold, then also aRc holds. A binary relation on a set A which is reflexive, symmetric, and transitive is called an **equivalence relation** on A. If $\equiv$ is an equivalence on A, then for any element $a \in A$ the **equivalence class** of a is the set $[a]_\equiv$ defined by:

$$[a]_\equiv = \{b \in A \mid b \equiv a\};$$

the **quotient set** $A/\equiv$ of A with respect to $\equiv$ is the set of equivalence classes of A:

$$A/\equiv = \{[a]_\equiv \mid a \in A\}.$$

A binary relation R on a set A is an **order relation** if it is reflexive, transitive and **antisymmetric**, that is for every $a \neq b \in A$, aRb and bRa cannot both hold. An order relation is **linear** or total on A if, for any every $a, b \in A$ either aRb or bRa holds (otherwise the ordering is said to be partial). Many important concepts based on order relations can be defined in general (minimum, maximum, minimal, maximal, upper bound, lower bound, least upper bound, and greatest lower bound).

The number of sequences over an alphabet of symbols grows exponentially with their length. For example the number of possible different sequences of length ten, over an alphabet of twenty symbols is 20^{10}, that is more than ten trillions. In fact in any of the ten positions we can put one of twenty possible symbols. This simple observation explains why polymers are the natural way for producing an enormous chemical variety.

An **operation** of k arguments over a set X is a rule that, when applied to a k-sequence of *arguments* over the set X, may yield one element of X as a result. If it does not provide any result, then it is *undefined* on that sequence of arguments.

If we order the elements of a set X, say $X = \{x_1, x_2, \ldots, , x_n\}$, then any subset of X can be expressed by a sequence of 0s and 1s where, at position i of the sequence, the value 1 occurs in the sequence if the element x_i belongs to the subset, otherwise the value 0 occurs. From this representation it follows that the number of possible subsets of X is 2^n. Table 5.2 provides a few examples of relations and operations.

5.1.2 Functions and Variables

A (total) function from a set A to a set B is an operation, that is, a rule that applies to all elements in A and for each of them produces exactly one result in B. The underlying idea of the function is that of *imaging*, as a representation device associating images to given objects. The term “function” goes back to theater representations, and its etymology is common to Latin words like *fungere* and *fingere*, which roughly refer to the action of *playing a role*. The standard notation for a function is:

$$f : A \to B$$

Table 5.2 Examples of relations and operations (*iff* means if and only if)

Father-Son relation	$F(x,y)$ *iff* x is father of y
Son-Father-Grandfather relation	$G(x,y,z)$ *iff* x is son of y who is son of z
Arithmetical Order	the usual enumeration order $\leq$ over the natural numbers
Number Divisibility	$D(x,y)$ *iff* x can be exactly divided by y
Linear betweeness	$B(x,y,z)$ *iff* point y is internal to the segment x,z
Arithmetical operations	$+\,,\,-\,,\cdot\,,/$
Area measure	$\sigma(P)$ is the area of a polygon P
Number Factorization	$factor(n)$ is the set of prime numbers dividing n
Sequence occurrence	$\alpha(i)$ is the element at position i in sequence α
Sequence Length	$\lvert\alpha\rvert$ is the length of α

and if $x \in A$, then $f(x)$ is the *image* of x, which belongs to B (A is called *domain* of f, while B is called *codomain* of f). The properties of *injectivity*, and *surjectivity* are expressed by the following formulae (where $\Rightarrow$ is the logical implication):

$$x \neq y \in A \Rightarrow f(x) \neq f(y) \text{ (\textbf{injectivity})}$$

$$y \in B \Rightarrow y = f(x) \text{ for some } x \in A \text{ (\textbf{surjectivity})}$$

bijectivity of f means both injectivity and surjectivity, that is, a correspondence one-to-one between the set A and the set B. The imaging principle realized by functions is one of the most powerful principles in mathematics, which can be found in a wide variety of situations. In fact, a typical approach in mathematics is the representation of certain objects (for example, physical states) by means of other objects, very often more abstract (for example, numbers), in order to derive useful information about the former ones by working with the latter ones. In some contexts it is useful to consider **partial functions**, that is, functions that can be undefined on some elements of their domains. However, unless explicitly stated, functions are implicitly assumed to be total, that is, always defined on their domain.

Variables are a crucial device of mathematical language. Any general statement in mathematics uses variables implicitly or explicitly. For example, a geometrical proof about triangles, begins with a preamble of such a type: "Given three points P, Q, R, and a line passing through P ...". In fact, in order to show any relationship of general nature, it is essential to deal with "generic" elements. Generality means variability, because a point P can be *instantiated* with any specific point when a definite context is selected. Variables were essential for the development of algebra. In fact, when we establish some conditions holding for a number, but we do not know the exact identity of this number, we use a variable to denote it, and in this sense that variable corresponds to a an *unknown* value. A *solution* of an equation in one variable corresponds to the determination of a value of the variable that satisfies the equality between the left hand side and the right hand side of the equation.

Very often, different names for indicating variables relate to different levels of variability, for example: *parameter, indeterminate, unknown*.

The essence of the notion of variable is that of an entity taking values within a **range** of variability, which can be specified by a set. The simplest kind of variables are *boolean variables*, ranging over a set of two values (usually denoted by $0, 1$). An important issue concerning variables, attains to the relationship between variables. For example, let X and Y represent the measures of two quantities related to some bacterial cells (say, the quantity of a nutrient in the environment and the quantity of a protein inside the cell, respectively). Let us assume we discover that, in certain conditions, the value of Y depends on the value of X. In this case we say that Y is a dependent variable with respect to X, and this dependence defines a function $f : A \to B$, from the range A of X to the range B of Y: for any $a \in A$, $f(a) = b$ when b is the value taken by Y in correspondence to the value a taken by X.

5.2 Numbers and Digits

Numbers appear as quantity ratios when measures are considered. Natural numbers measure sizes of populations; rational numbers measure, with approximations, the sizes of continuous quantities.

In counting populations, a relevant aspect is the distinguishability of elements. In fact, the possibility of counting many objects does not imply the ability of recognizing them as single objects distinguishable from each other. This situation is typical with molecules. One may be able to find 1000 molecules of a protein, but one can seldom recognize each of them from another of the same group. If we measure populations by using a standard population size of 1000 elements, say it a Kilo K (of elements), as a unit, then we obtain fractions. For example, the size of a population of 3200 elements is 3 K + 2/10 K, that is, 3.2 K.

Greeks had different notions of numbers. Natural numbers were “arithmoi”, fractions were “megethe”, connected to the geometrical intuition of segments. The four classical operations on numbers were defined in geometrical terms (by compass and ruler constructions over segments). “Logoi” were related to the incommensurable ratios (e. g., between the edge and the diagonal of a square) and to the notion of infinite.

A definite, complete understanding of number systems was developed in the 19th century, at the end of a long process of unification of the distinct Greek notions and of the geometric Greek perspective with the algebraic notion of number [190]. This process evolved along the new algorithmic perspective of the positional notation, of Hindu-Arabic origins, that was introduced by Leonardo Fibonacci in his *Liber Abaci* (1202), whereby numbers are represented by sequences of *digits*.

Sequences over an alphabet can always represent numbers. In fact, as we will explain later on, if the pairwise distinct elements which may occur in a sequence are fewer than k, then a sequence may be taken as a base-k representation of a number.

5.2.1 Natural Numbers

The set $\mathbb{N} = \{0,1,2,3,\ldots\}$ of natural numbers can be constructed by starting from 0, by applying iteratively the successor function, which always yields a new number:

$$0 \mapsto s(0) \mapsto s(s(0)) \mapsto s(s(s(0)))\ldots .$$

If we represent numbers by sequences of only one symbol, say 1, then zero corresponds to symbol 1 and the successor operation becomes the juxtaposition of a symbol 1 to its argument:

$$1 \mapsto 11 \mapsto 111 \mapsto 111 \mapsto \ldots .$$

The **sum** of two numbers n, m can be obtained by iterating the successor operation n times, starting from m (s occurring n times):

$$n + m = s(\ldots s(m)\ldots).$$

The **difference** is the inverse operation of the sum, that is, $n - m$ is the number k, if it exists, such that $m + k = n$.

The **product** of n, m is defined by iterating the m-sum (sum of m) n times, starting from 0:

$$n * m = (((0 + m) + m) + \ldots m))).$$

The **division** is the inverse operation of the product, that is, $\frac{n}{m} = k$ if $m * k = n$.

Sum and product are commutative operations, that is, their result does not depend on the order of their arguments. Difference and division are not commutative. The product is distributive with respect to the sum: $a * (b + c) = a * b + a * c$.

The number x raised to **power** of n, is defined by iterating the x-product n times, starting from 1: $1 \times x \times x \ldots \times x$; it is denoted by x^n. Number x is called the *base*, and n the *exponent*.

When a base b is fixed, then the operation yielding b^x in correspondence of x is called **exponential** (of base b).

The **root** $\sqrt[m]{n}$ is the inverse of the m-power, that is, $\sqrt[m]{n^m} = n$, while the **logarithm** (of base b) is the inverse of the exponential, so that $m^{\lg_m k} = k$.

Inverse operations are not always defined on natural numbers. For example, difference and division are not defined when the first number is smaller than the second one. The extension of naturals to the set $\mathbb{Z}$ of integers guarantees that difference is always defined. The extension of integers to the set $\mathbb{Q}$ of rationals (fractions) guarantees that division is always defined for any pair of rationals (apart from any division by zero, which is always undefined). Square root is not always defined on positive rationals, as mathematicians of Pythagoras' school discovered (no rational number can equal $\sqrt{2}$). The extension of rationals to the set $\mathbb{R}$ of reals guarantees that square root is always defined on positive reals. However, square roots (in general, even roots) of negative reals cannot be real numbers. The extension of reals to the set $\mathbb{C}$ of complex numbers guarantees that roots are always defined.

Iteration and inversion are the two basic mechanisms to define operations on numbers, starting from the basic operation of successor which generates the infinite sequence of natural numbers. As we have just seen, sums are iterations of successor, products are iterations of sum, and powers and exponentials are iterations of product. Differences are inverse operations of sums, divisions are inverse operations of products, and root extractions and logarithms are inverse operations of powers. The extensions of the number sets: $\mathbb{N} \Rightarrow \mathbb{Z} \Rightarrow \mathbb{Q} \Rightarrow \mathbb{R} \Rightarrow \mathbb{C}$ guarantee that inverse operations of difference, division, and root extraction are always defined.

It is interesting that the inverse operations can be also obtained by a chain of iterations, using the operation *pred*, for predecessor, which is the inverse of successor. In fact, difference can be obtained by iterating predecessor, division by iterating difference, and logarithm by iterating division.

Usual symbols for numbers came from Magreb. In fact, Leonardo Fibonacci (1180–1250), who introduced them in Europe in his famous book *Liber Abaci*, was son of a merchant from Pisa and studied near Tunis. These digits are compact forms, drawn by a continuous line, where the number of angles corresponds to the numerical value of each digit (zero is a circle without angles), see Fig. 5.1. Hindu-Arabic positional number representation had an enormous impact on all the western cultures: it can be compared with the discovery of phonetic alphabets (the birth of efficient writing systems). Algorithms for numerical elaboration based on positional representation were the beginning of a mathematical attitude from which modern algebra stems, with systematic methods for solving equations.

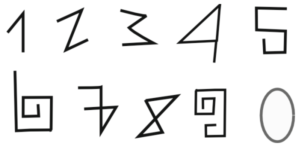


Fig. 5.1 The angles of Arabic digits

The usual representation of numbers with decimal digits expresses a number, say 357, as a sum of powers of 10: $357 = 3 * 10^2 + 5 * 10^1 + 7 * 10^0$. It is possible to show that this representation is univocal for any natural number, provided the leftmost digit is nonzero. Moreover, the method may be applied with any base greater than 1. For example, the same number, with respect to base 8, becomes 545, because $357 = 5 * 8^2 + 4 * 8^1 + 5 * 8^0 = 357$. A more general and compact way for expressing this kind of notation is given in the next subsection.

5.2.2 *Sums and Positional Representations*

The sum operation can be extended to any sequence of numbers, by using the summation symbol $\sum$:

$$\sum_{i=1}^{n} a_i$$

standing for $a_1 + a_2, + \ldots + a_n$. The extreme values can be indicated in many ways, for example:

$$\sum_{i=1,n} a_i$$

or similar, clearly understandable, notations. The following equation expresses the *appearance* of the sum variable (its identity is not relevant).

$$\sum_{i=1}^{n} a_i = \sum_{j=1}^{n} a_j$$

other properties of $\sum$ follow from the properties of sum operation:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{k} a_i + \sum_{i=k+1}^{n} a_i$$

$$\sum_{i=1}^{n} (a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i$$

$$\sum_{i=1}^{n} b a_i = b \sum_{i=1}^{n} a_i$$

$$\sum_{j=1}^{m} b_j \left(\sum_{i=1}^{n} a_i \right) = \sum_{i=1}^{n} a_i \left(\sum_{j=1}^{m} b_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j = \sum_{j=1}^{m} \sum_{i=1}^{n} a_i b_j .$$

Given a natural number $b > 1$, called *base*, for any natural number n, there exists a natural k such that the following univocal representation of base b holds:

$$n = \sum_{i=0,k} c_i b^i$$

where $c_i < b$ for $i \leq k$ and $c_k > 0$. Therefore, sequence $(c_0, c_1, \ldots, c_k)$ (usually written in the reverse order) univocally identifies, with respect to base b, the natural number n. When b different symbols, called **digits**, are chosen to represent the numbers smaller than b, then any sequence of digits, with $c_k \neq c_0$, is a representation of a natural number.

All classical algorithms to compute the four arithmetical operations can be generalized to any base, when the tables for products and carries are given for any pair of values smaller than b (Pythagorean tables).

Other positional representations different from the decimal one are: the binary representation (introduced by Leibniz), the octal and hexadecimal representations (basis 8 and 16 respectively), which, besides the binary representation, are mostly used in computer representations of numbers.

Representations of base 60 and 20 are found in ancient civilizations, the Assyro-Babilonyan and the Maya, respectively. The latter is a positional representation proper, for it also had a symbol for zero. This is an essential aspect of positional representations.

The word **zero** comes from an Arabic root for *zephyr* (a gentle wind, or something ineffable). Zero is the first great abstraction of modern algebra (the empty set is its set-theoretic equivalent). It is essential in positional notation and is the basis for computations with big numbers, a problem that Greek mathematics addressed with Archimedes' *Arenarius*, but without a complete and definite solution. However, the informational power of zero and of positional notation relies on the reduction of numbers to sequences of digits. It seems noteworthy that the etymology of the word algorithm is rooted in the positional representation. It is used to mean the computational procedures (on representation of numbers) after al-Khwarizmi's seminal work, which was translated into Latin (several versions in the 12th and 13th centuries) and published with title "Algorismi de numero Indorum" (in the 19th century) [170, 180, 167]. This was the beginning of the algebraic methods for elaborating numerical information.

5.2.3 Integer Numbers

The set $\mathbb{Z}$ of integer numbers was obtained from $\mathbb{N}$ to express negative quantities. An integer number is a pair of a sign and of a natural number providing an absolute value (usually, the sign is omitted when positive). The extension of the four fundamental operations to the integers is straightforward, except for product, which is obtained by the product of the absolute values equipped with a sign according to the rules given in Table 5.3, which, at a first glance, may appear somewhat arbitrary. The rules of product sign are necessary for having an extension of the product operation to integers that is coherent with the product defined on naturals. It is easy to accept the first two rules of Table 5.3, what about the third rule?

Let us evaluate the product $(-k)(-h)$. We have $(-k) = [n-(n+k)]$ and $(-h) = [m-(m+h)]$ for any pair of natural numbers n, m, therefore:

$$(-k)(-h) = [n-(n+k)][m-(m+h)] =$$

$$nm - n(m+h) - m(n+k) \,?(n+k)(m+h) =$$

$$nm - nm - nh - mn - mk \,?(nm + nh + km + hk).$$

Table 5.3 The sign multiplication rules

$+ * + = +$
$+ * - = -$
$- * - = +$

The symbol ? stands for the sign to assign to the product between $-(n+k)$ and $-(m+h)$. But, in order to get a result which does not depend on n, m, the sign ? has got to be $+$, thereby yielding $(-k)(-h) = kh$.

5.3 Rational and Real Numbers: Approximation and Infinite

Rational numbers are fractions of integers. When we add a “part of the unit” to a natural number, we get a rational number. For example, 3 plus $1/7$ is equal to $21/7 + 1/7$, that is $22/7$. All positive fractions can be viewed as the sum of a natural number, possibly zero, plus a fraction of the unit. Within fractions, with positive or negative sign, all the four arithmetical operations are always defined (apart from division by zero, always undefined). Fractions of the unit can be represented in decimal notation by sums of negative powers of ten. For example, $1/8$ is equal to $1 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$, that in the usual decimal notation is written as $1/8 = 0.125$. However, in some cases these sums are infinite. For example, $1/6 = 1 \times 10^{-1} + 6 \times 10^{-2} + 6 \times 10^{-3} + \cdots$, that is, in the usual decimal notation it is written as $1/6 = 0.\bar{6}$, meaning that digit 6 is infinitely repeated. Therefore fractions provide either finite sequences of decimal digits, separated by a dot (or a comma or some symbol different from digits) to distinguish the integer part from the fractional part, or periodical sequences ending with an infinitely repeated subsequence of digits.

It is possible to show that finite and periodical sequences are the only two kinds of decimal representations associated to fractions. In fact, the decimal notation of a fraction is obtained by applying the usual division algorithm learned at primary school (with many variants, but with the same essential procedure). According to this algorithm, at each step, a difference is calculated between two decimal numbers, and then a digit of the greater number is appended to this difference, by providing a number of at most $n+1$ digits, if n is the number of digits of the divisor. The comparison of this number with the divisor provides a new digit of the result. But, the possible decimal sequences of $n+1$ digits are 10^{n+1}, therefore surely before than $10^{n+1} + 1$ times, the same case must occur twice. The digits of the result between these two occurrences are exactly the digits of the periodical part in the decimal representation of fractions (finite decimal representations are a special case of periodical representations with periodical part consisting of zeros).

The above argument raises a natural question. What kind of numbers are infinite non-periodical decimal sequences of type $0.- - - \cdots$? It is easy to realize that the four arithmetical operations can be naturally defined for them. Moreover, these sequences have a direct reading as points of the unitary segment. If a segment is partitioned into 10 contiguous intervals (for example, including the extreme left point, but excluding the right extreme point), then each digit can be associated to each interval. For example, a sequence of type 0.23 individuates the points internal to the third part (associated to digit 2) of the unitary segment, and then internal to the fourth part of it (associated to digit 3). In conclusion, infinite non-periodical sequences correspond to infinite processes generating subintervals of the unitary segment. These numbers are the **real numbers**. Here are two examples of irrational numbers obtained by defining infinite sequences of digits that are not periodical. The first one is an infinite sequence which cannot be periodic because 1 occurs once, 2 twice, ..., 0 ten times, 10 eleven times, and so on.

$$0.122333444455555666666777777\ldots$$

Another infinite sequence is due to Champernowne, a mathematician of the beginning of the 20th century; it has the digits of all natural numbers arranged after the dot in their enumeration order (of course this prevents any possibility of a periodical repetition of the same group of digits).

$$0.1234567891011121314151617\ldots$$

5.3.1 Incommensurability, Divisibility, and Distinguishability

The way of introducing irrational numbers by means of infinite decimal sequences is related to the positional representation of numbers (see [179]). However, it does not correspond to the historical development of the concept of number. In fact, real numbers were discovered by the Greeks, when no positional representation of numbers was available. The Greek discovery of the necessity of real numbers was entirely based on geometrical arguments that we want to present briefly.

Consider a square having side of length a. Let d denote the length of its diagonal, as indicated in Fig. 5.10. If the side and the diagonal lengths are in a rational ratio, then $a/d = n/m$, where n, m are natural numbers ($n < m$). Therefore, the discovery that this ratio is not a rational number implies that, if c is the length of a portion of the side such that $a = nc$, even assuming the length of c as small as we want, no natural number m can exist such that m times this portion could equal the diagonal, that is, $d = mc$. For this reason, we say that the lengths of these two segments are **incommensurable**. This argument seems completely counterintuitive, because it is reasonable that at some, even very small level, a common part that is submultiple of any two segments must exist. As we show, the mathematical power of this argument is in its logical nature. Of course, our resolution ability has to stop at some level, therefore the claim of incommensurability may appear meaningless. However, if we assume a pure geometrical notion of segment where any two internal points define a part of it, and any two points can be separated by a point between them, then the argument holds perfectly. This argument has a pure logical nature; this is the reason for the name ”logoi” given to numbers defined by such a kind of existential argument (”logos” means reasoning). However, despite their purely logical existence, these numbers are fundamental for the development of the whole mathematics.

From a historical viewpoint, we remark that a lot of Greek philosophy was familiar with processes of time and space divisibility. The famous Zeno’s paradoxes were examples where specific processes of infinite divisibility are taken into account. The deep understanding of “infinite” as a key feature of mathematical entities is surely one of the most original and fruitful aspects of Greek thought (developed in several directions by Eudoxus of Cnidus, Euclid of Alexandria, and Archimedes of Syracuse).

The following argument is a modern elaboration of a proof of the incommensurability between the side and the diagonal of a square, developed by the Pythagorean school (6th century B.C.).

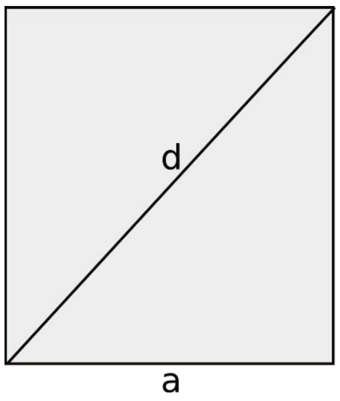


Fig. 5.2 The incommensurability between the side and the diagonal of a square

A proof of geometrical incommensurability. Consider a unitary square (with side of length one). According to Pythagoras' theorem the length of its diagonal is $\sqrt{2}$, therefore if the ratio between the diagonal and the side were a rational number, then two natural numbers p, q should exist such that $\sqrt{2} = p/q$. We show that this hypothesis implies a contradiction. In fact, without loss of generality we can assume that no common factor exists between p and q. If $\sqrt{2} = p/q$, then $2 = p^2/q^2$, that is, $p^2 = 2q^2$. The last equation implies that p^2 has number 2 as a factor, but this requires that also p has 2 as a factor. Therefore $p = 2r$ for some r, that is, $4r^2 = 2q^2$, thus $2r^2 = q^2$. But, by the same argument used earlier, this fact implies that q has 2 as a factor. In conclusion, both p and q have 2 as a factor, and this contradicts our initial assumption that no common factor exists between p and q. Whence, the initial hypothesis $\sqrt{2} = p/q$ implies a contradiction, thus proving the **irrationality** of $\sqrt{2}$, because $\sqrt{2} \neq p/q$ for any pair of natural numbers p, q. In conclusion, this proof shows the incommensurability between the side and the diagonal of any square. In an analogous way it can be shown that $\sqrt{p}$ is irrational for any prime number p.

Real numbers completely express the continuous nature of the straight line and of segments. If we consider a segment as a set of points arranged in a linear structure, then we realize that the elements of this set include points that we cannot singularly individuate. This is a subtle argument implied by the analysis developed by Georg Cantor in the 19th century. We do not enter in the details of this discussion, we only want to remark that a continuous structure implies infinite divisibility and undistinguishability of its components. This aspect is a crucial discrimination between continuous and discrete structures. Compare a segment, which is continuous with a set of golf balls, which is discrete. A subset of balls can be partitioned into smaller subsets, but if the initial set is finite, then after a finite number of partitions you cannot go on because one half of a golf ball is not a golf ball. But if you start from a segment, which has a finite size, the partition process can be infinitely performed, because one half of a segment is still a segment. The difference is in the composition of the two structures.

However, distinguishability of objects needs a further subtle analysis. In fact, two balls can be distinguished from three balls, but this does not mean that they can be individually distinguished. For example, if they have the same shape and their positions may change when you are not observing them, are you able to tell whether they exchanged their positions, when you observe them again? Surely, you can tell whether their number increases or decreases, but in many cases their single individuality cannot be determined.

Let us mention that arithmetical operations on numbers, when numbers are represented by segments, are realized by using compass and ruler. Sum and difference may be performed by putting segments on the same line, in straightforward manners, while product and division are performed by using similar triangles, at it is shown in Fig. 5.3. An algorithm is a procedure solving a given problem. Many famous geometrical constructions are essentially algorithms manipulating geometrical entities to solve specific problems.

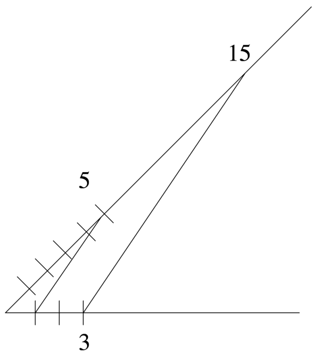


Fig. 5.3 Segment multiplication by similarity: the lengths of two parallel segments are in the ratio of their intercepts with the same incident lines

For our further discussion we recall the famous Pythagoras' theorem about right-angled triangles. Figure 5.4 is essentially a proof of this theorem (due to Chinese mathematicians).

Pythagoras' theorem is the basis of the well-known goniometric equation: $\sin^2 x + \cos^2 x = 1$. In the goniometric unitary circle, a radius which forms an angle α with the x-axis has projections, over the two axes, $\sin\alpha$ (y-projection) and $\cos\alpha$ (x-projection), which together with the radius form a right triangle. From this circle all the important goniometric properties can be derived. Figure 5.5 is essentially a

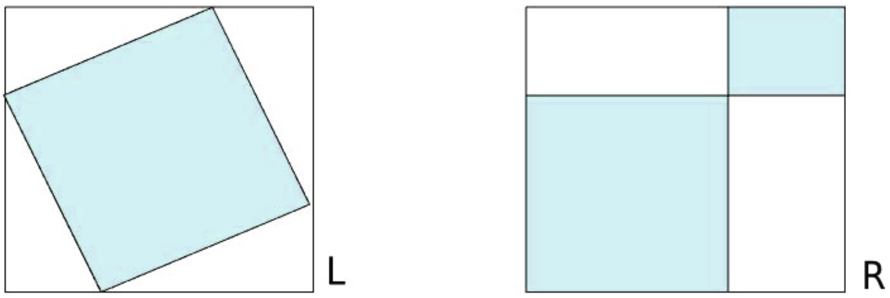


Fig. 5.4 Chinese proof of Pytagoras' theorem. The area of the left and right squares L, R is the same. The internal square of L is obtained by removing four triangles from L. These four triangles are equivalent to the two rectangles removed from the right rectangle R. Therefore, the areas of the internal figures of L and R are equal

proof of the formula providing the sine of the angle $\alpha+\beta$. Angles are measured by *radiants*, where a radiant is the angle corresponding to an arc with the same length of the radius (so that 2π is the measure of a whole rotation, of 360 degrees). The segment between double circles is $sin(\alpha+\beta)$. It is the sum of two projections (bold lines). In fact, the segment between the small and big full circles is $\cos\beta$, and its vertical projection (the smaller bold segment) is $\sin\alpha\cos\beta$, while the segment between the double circle and the big full circle is $\sin\beta$ and its vertical projection (the bigger bold segment) is $\sin\beta\cos\alpha$. The sum of the two projections (bold lines) is $\sin(\alpha+\beta)$.

The number $\sqrt{2}$ is irrational, but it is the root of the equation $x^2=2$, that is, it is **algebraic**, while the number π cannot be obtained as root of some equation with rational coefficients. Irrational numbers which are not algebraic are called **transcendental**. However, there are many formulae and algorithms for generating all the digits of $\sqrt{2}$, as well as all the digits of π. Real numbers for which procedures are known which generate the sequence of their digits are called **computable real numbers**. When the notion of computable number was definitely clarified, after a seminal paper of Turing (1936), it was completely clear that computable real numbers form a very small (in terms which can be mathematically formalized) subset of all real numbers. In conclusion, we can define sets having elements which we can collectively identify, albeit most of them have no individual identification process.

All the sets of numbers $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ are infinite. However, there is a crucial difference between the sets of natural, integer, and rational numbers, from one side, and the sets of real and complex numbers, on the other side. In fact, elements of $\mathbb{N},\mathbb{Z},\mathbb{Q}$ are represented by suitable finite sequences of symbols, while those of $\mathbb{R}$ require, in general, infinite sequences of symbols. It can be shown that any infinite set of finite sequences is **enumerable**, that is, it can be put in one-to-one correspondence with the natural numbers. This kind of correspondence cannot be established between the natural numbers and the set of all infinite sequences over a finite alphabet. This means that real numbers are *many more* than the rationals. We can express

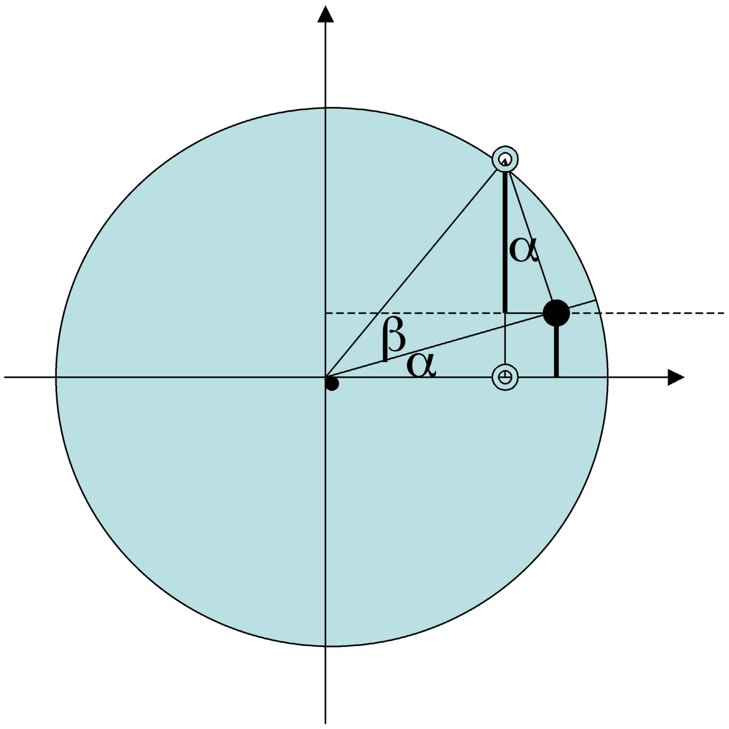


Fig. 5.5 $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$ (analogously $\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$ holds)

this situation with the following statement (which we do not explain formally): If you choose randomly a point in the interval $[0,1]$ the probability of getting a rational number is almost zero, while that of getting a real irrational number is almost one.

Let S_k the set of finite sequences (over a given finite (ordered) alphabet) of length k. For any $k \in \mathbb{N}$, S_k is a finite set. Enumerate in the alphabetic order the elements of $S_1, S_2, \ldots$ In this way any finite sequence occurs at some step of the enumeration.

The set S of all the infinite sequences over a binary alphabet $\{0,1\}$ cannot be put in one-to-one correspondence with natural numbers (the same argument can be easily generalized to the set of sequences over any finite alphabet). Table 5.4 explains the original intuition of the *diagonal* argument by Georg Cantor.

Table 5.4 Cantor's square of binary sequences

1	0	1	0	0	1	1	...
1	**0**	0	0	0	1	1	...
1	0	**1**	0	0	0	1	...
1	1	1	**0**	0	1	1	...
1	0	1	0	**1**	1	1	...
1	0	1	1	0	**1**	1	...
1	0	1	0	1	1	**0**	...
...	...	...	...	...	...	...	...

No infinite list of binary sequences can contain all the infinite binary sequences. The *anti-diagonal sequence*, where each symbol of the diagonal is changed, must differ from any sequence of an infinite list. For example, **0 1 0 1 0 0 1 ...** differs from the seven sequences explicitly indicated in the list of Table 5.4: from the first one in its first digit, from the second one in its second digit, and so on.

An argument similar to Cantor's diagonal argument, due to Bertrand Russell, shows that no set A can be put in one-to-one correspondence with its power set $\mathbb{P}(A)$. In fact, assume, that a bijective function $f(x)$ could provide such a correspondence between A and $\mathbb{P}(A)$. Let

$$D = \{x \mid x \notin f(x)\}$$

then, some $y \in A$ must exist such that $D = f(y)$. But, for such a y either of the following statements leads to a contradiction:

$$i)\ y \in D$$

$$ii)\ y \ \notin D.$$

In fact, $i)$ implies $y \notin f(y)$, because y has to satisfy the property, holding for all the elements of D, $x \notin f(x)$, and by the equation $f(y) = D$ defining y, we deduce $y \notin D$, therefore $i)$ implies $ii)$. Analogously, $ii)$ implies, by the equation $D = f(y)$, that $y \notin f(y)$, thus y satisfies the property of the elements of D, that is, we deduce $y \in D$, therefore $ii)$ implies $i)$.

According to Russell's argument, the powerset $\mathbb{P}(\mathbb{N})$ cannot be put in one-to-one correspondence with $\mathbb{N}$. Moreover, any subset of the natural numbers identifies a binary infinite sequence s (and vice versa) when we consider the numbers in this subset as the ordinal positions in the sequence where s takes the value 1. But, these sequences are in one-to-one correspondence with $\mathbb{R}$, therefore by Russel's argument we obtain another proof that the set of infinite binary sequences (bijective with the reals) cannot be put in bijective correspondence with the set of the natural numbers.

5.4 Complex Numbers and Real Vectors

Complex numbers are sums of a real number with an **imaginary** number. Imaginary numbers were discovered by the Italian mathematicians of the Renaissance, during their study of formulas to solve algebraic equations. In fact, the two solutions of a second degree equation $ax^2 + bx + c = 0$ are given by the following formula (a solution with $+$, the other with $-$):

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

When $(b^2 - 4ac)$ is negative we get square roots of negative numbers, or numbers multiplied by $\sqrt{-1}$. No real number exists having a negative square, therefore $\sqrt{-1}$ is meaningless in the set of real numbers. However, in formulae giving solutions of third and fourth degree equations, in some cases, real solutions are obtained by using formulae which manipulate imaginary numbers. This discovery was shocking for Italian mathematicians, in the same way as that of incommensurability was shocking for Greek mathematicians. In fact, meaningful solutions to a problem could be obtained by manipulating meaningless objects. Moreover, if i denotes $\sqrt{-1}$, then in the set of numbers $a + ib$ with $a, b \in \mathbb{R}$ the arithmetical operations can be extended in a way which is coherent with their definition on the reals. One may just deal with these numbers as if they were algebraic expressions (sums of two reals) including an indeterminate i satisfying the condition (impossible for every real number) that $i^2 = -1$. Therefore $(a_1 + ib_1)(a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + b_1a_2)$ (analogously for the other operations).

5.4.1 Euler's Identity

The imaginary unit i is related to two numbers which are crucial in mathematics. One of them is Archimedes' constant π, the ratio between the length of a circle and its diameter. The second one is the constant e, introduced by John Napier, and further defined and fully investigated by Leonhard Euler. The Scottish mathematician Napier in 1614 published a work where the problem is addressed of reducing the computation of products to that of suitable sums. An arithmetical progression is a sequence of numbers where each element (apart the first one) is obtained by the previous one by adding a constant value to it (the common difference of the progression). Analogously, a geometric progression is a sequence of numbers where each element (apart from the first one) is obtained from the previous one by multiplying it by a constant value (the common ratio of the progression). The initial idea of Napier (suggested by mechanical analogies) was to determine geometrical progressions coinciding, with a good approximation, with an arithmetical progression of very small common difference, for example 0.001. In this manner, a product of two numbers $a \leq b$, with an approximation to the third decimal, can be obtained by locating them in the geometrical progression and then by locating, a greater number c at a distance (number of elements) from a equal to the position of b. In this way, summing distances in the geometrical progression corresponds to multiplying elements of the arithmetical progression. Napier realized that numbers such as 1.01^{100}, or 1.001^{1000}, that is, having the form $(1+1/n)^n$ with large n were useful for realizing his intuition. All these numbers are greater than 2 and smaller than 3, and for increasing values of n they approximate a real number e, later formally defined by Euler. We will present, in Sect. 5.6.5, an argument showing the importance of e in many developmental processes. Here, we report an analysis of Euler which links this number to the imaginary unit and to Archimedes' constant π. The first important result of Euler, about e, was its representation as an infinite sum. The starting point of this result is Newton's formula:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

where $\binom{n}{k}$ denotes the **binomial coefficient**, that is, the number of different k-subsets of n elements, which is given by the following formula, where $n!$ is the factorial of n, that is the product $1 \times 2 \times \ldots \times (n-1) \times n$:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

The argument of Euler is informal, but extremely fascinating. He expresses e by the following formula, where ω stands for an infinite number (*numerus infinite magnus*):

$$e = (1+1/\omega)^{\omega}$$

but, being, for any number x, also ω/x an infinite number, we can put:

$$e = (1+x/\omega)^{\omega/x}$$

therefore

$$e^x = ((1+x/\omega)^{\omega/x})^x = (1+x/\omega)^{\omega}$$

and by Newton's formula, dealing with ω as if it were a natural number, this can be written as:

$$e^x = (1+x/\omega)^{\omega} = \sum_{k=0}^{\omega} \binom{\omega}{k} x^k/\omega^k$$

but, by definition of binomial coefficients and factorials, we get:

$$e^x = \sum_{k=0}^{\omega} \binom{\omega}{k} x^k/\omega^k = 1 + \omega x/1!\ \omega + \omega(\omega-1)x^2/2!\ \omega^2 + \omega(\omega-1)(\omega-2)x^3/3!\ \omega^3 \ldots$$

and for the infinity of ω we can assume $\omega = (\omega-1) = (\omega-2)\ldots$ therefore we can conclude that:

$$e^x = 1 + x/1! + x^2/2! + x^3/3! \ldots$$

With similar arguments Euler proved the following identities involving the circular functions sine and cosine:

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! \dots$$
$$\cos x = 1 - x^2/2! + x^4/4! - x^6/6! \dots$$

Therefore by comparing the series for the exponential function e^x and of the circular functions, it is easy to obtain the famous identity of Euler connecting the three numbers π, i, e:

Table 5.5 Euler's trigonometric identity

$$e^{ix} = \cos x + i \sin x.$$

If we put $x = \pi$, from $\cos(\pi) = -1$ and $\sin(\pi) = 0$, we easily get the following celebrated formula, called **Euler's identity**.

Table 5.6 Euler's identity

$$e^{i\pi} + 1 = 0.$$

According to Napier's logarithms, Euler's constant e is called the **natural base of logarithms**. The operation $\log_a x$ of a number x, with respect to any base $a \in \mathbb{R}, a > 1$, yields the value such that $a^{\log_a x} = x$. From this definition, the main properties of Table 5.7 easily follow.

Table 5.7 Basic properties of logarithms

$\log_a xy = \log_a x + \log_a y$
$\log_a x^y = y \log_a x$
$\log_a b = -\log_b a$
$\log_a 1/x = -\log_a x$
$\log_b x = \log_b a \log_a x$

All the definitions of basic arithmetical operations, plus those of exponentiation, and logarithm, with respect to real bases, easily extend to rational and real numbers. We conclude by remarking that logarithms, and circular functions, had an enormous impact on the technological and social development of the 17th century. In fact, reliable navigation is based on reliable and efficient methods of trigonometric computations (for the angles to follow along a navigation trajectory). Therefore, logarithmic tables of sine, and cosine values were essential computational tools for the geographic revolution of that time.

5.4.2 Polar Representation of Complex Numbers

Euler's identity suggests a geometrical interpretation of complex numbers in the plane. Firstly the French mathematician Argand [166], and then Gauss represented a number $a+ib$ as the point of the Cartesian plane with abscissa a (on the real line) and ordinate b (on the imaginary line). In this way, the complex number $\cos x + i \sin x$ is a point on the unitary circle centered in the origin of the Cartesian plane where the radius connecting it to the origin forms an angle x with the real line. This means that, according to Euler's identity, e^{ix} corresponds to this point, and any point of the complex plane has the **exponential** or **polar** representation ρe^{ix}, where ρ is the distance between the point and the origin, called the **module** of the number, while x is the angle that the radius connecting the point to the origin forms with the real axis. The following identity is a consequence of the previous analysis, where arcsin is the inverse function of sin, that is $\arcsin x$ is the angle α such that $\sin(\alpha) = x$:

$$a+ib = \sqrt{a^2+b^2}\, e^{i \arcsin(b/\sqrt{a^2+b^2})}.$$

This interpretation of $\mathbb{C}$ gives a (geometrical) meaning to the complex numbers based on the *imaginary* unit. Moreover, it opens enormous possibilities of applications, ranging from electromagnetism to hydrodynamics, or quantum physics, which show, in almost literal terms, the power of mathematical imagination. The sum of two complex numbers is easily obtained by separately summing their real and imaginary parts, while the exponential representation implies that their multiplication can be carried out by multiplying their modules and by summing their angles.

An important consequence of the polar representation of complex numbers is the proof of the fundamental theorem of algebra: *Every algebraic equation of degree n with coefficients in $\mathbb{C}$ has n solutions in $\mathbb{C}$ (by counting each solution with its multiplicity).*

Firstly, we observe that, given an equation $P(x) = 0$ where $P(x)$ is a polynomial of degree n, then α is a solution of it iff $P(x) = (x-\alpha)Q(x)$, where $Q(x)$ has degree $n-1$. In fact, if $P(x) = (x-\alpha)Q(x)$, then obviously $P(\alpha) = 0$. Vice versa, If $P(\alpha) = 0$, then assume that $P(x) = (x-\alpha)Q(x) + k$, with $k \neq 0$, then $0 = P(\alpha) = (\alpha - \alpha)Q(x) + k$, that is, $0 = k$, which is a contradiction. This means that if $P(x)$ has n solutions $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{C}$ then $P(x) = a(x-\alpha_1)(x-\alpha_2)\cdots(x-\alpha_n)$ (for some constant a). Therefore, if any equation has at least one solution, then any equation of degree n has exactly n solutions (counted with their multiplicities). In fact, if $P(x)$ has degree n and it has at least one solution, say it α_1, then $P(x) = (x-\alpha_1)Q(x)$ with $Q(x)$ of degree $n-1$, and all solutions of $Q(x) = 0$ are also solutions of $P(x) = 0$. But, also $Q(x)$ has at least one solution, therefore we can apply iteratively the same argument, and so we eventually reach a polynomial of degree 1, having exactly one solution.

In conclusion, the essential part of the fundamental theorem is the proof of the existence of at least one solution for any algebraic equation $P(z) = 0$, with $z \in \mathbb{C}$.

After some partial proofs of Euler, a first complete attempt of proof of the fundamental theorem is that given by d'Alembert in 1746, assuming a lemma, which was stated without a rigorous proof (Argand proved it in 1806 [166]). During his lifetime, Carl Friedrich Gauss offered at least four different proofs of this theorem, covering the timespan of his entire adult life. His first proof was published in his doctoral dissertation in 1799, at age of 22. In 1849, just a few years before his death, Gauss gave a fourth proof that bore similarity to his first one. These proofs (the first and the fourth one) were grounded on subtle and deep properties of continuous functions and topological concepts, which at that time were missing of any rigorous clarification (the masterpieces of Dedekind and Weierstrass about reals and continuity were completed after 1870, and topological properties of curves, implicitly assumed in the first and fourth proofs of Gauss, reach a completely rigorous mathematical elaboration only in the 20th century). Quoting from [192], Gauss claims that: "It seems to be well demonstrated that an algebraic curve neither ends abruptly (as it happens in the transcendental curve $y = 1/\log x$), nor loses itself after an infinite number of windings in a point (like a logarithmic spiral)".

The informal proof that we outline uses arguments different from those of Gauss (see [171] for a topological proof). Let us assume the concept of transformation of the complex plane; for any polynomial P, any point z of the complex plane transforms into point $P(z)$ of the same plane. Let us fix a polynomial $P(z)$ (with complex coefficients) having degree $k > 2$ and where the monomial z^k has coefficient 1 (this does not imply any loss of generality). Let us transform the points of a disk D of the complex plane having as boundary a circle C, the center in the origin, and radius R. Of course, $P(0)$ belongs to D' ($P(0) \neq 0$, otherwise the proof would be concluded, being the origin a solution of the equation). The set of points that are P-images of the points in D form a region D', with boundary C'. The polynomial P transforms interior points of D into interior points of D' and points on the boundary C into points on the boundary C'. This intuitive property has a rigorous explanation, based on analytical aspects related to the "open mapping theorem". Let us define the radius R' of D' with respect to $P(0)$ as the minimum distance between $P(0)$ and C'. For large enough and increasing values of the radius R of C, the monomial z^k of P becomes dominant (on the points of C), thus the corresponding radius R' of D' (with respect to $P(0)$) increases consequently. Therefore, for a sufficiently large radius R of C the radius R' of D', is greater than the distance between $P(0)$ and the origin. In conclusion, when the radius R of C reaches this value, the origin belongs to D', so that a point z_0 in D has to exist which is transformed into 0, that is, $P(z_0) = 0$.

The representation of complex numbers in the plane is an example of geometrical representation of pairs of numbers as points in a two-dimensional space. This idea easily generalizes to triples and, in general, to any sequence of k components. In other words, if $\mathbb{R}^2$ corresponds to the plane and $\mathbb{R}^3$ corresponds to the three-dimensional space, then $\mathbb{R}^k$ can be seen as a k-dimensional space, where basic

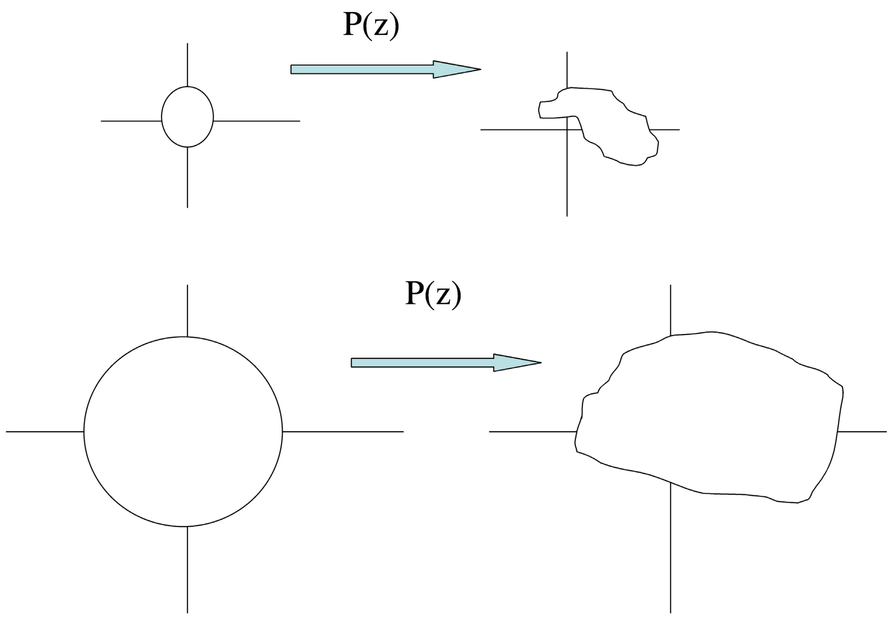


Fig. 5.6 Transformation of the circles by means of algebraic transformations

geometric concepts can be developed in an abstract setting. In general, a space where points are identified by a number (the dimension of the space) of variables ranging over the real numbers forms a (real) **vector space**. A more biologically oriented example of vectors is constituted by a cell culture plate, divided into k compartments where for each compartment different nutrients or drugs were placed. The number of cells for each compartment provides a vector in $\mathbb{N}^k$ and the growth process provides a time series for each compartment. As another example, a DNA microarray is a small chip where many thousand of spots are arranged in a matrix (any spot has a row and a column position). Each spot contains DNA probes of the same type (single strands of length around 20 bases), which are specific of genes, in such a way that each column refers to one gene. When RNA coming from a cell is uniformly distributed on the microarray, neglecting the details of the process, a quantity can be assigned to each spot, which determines the overall expression of the genes associated to the columns for the cell under observation. In conclusion, a DNA microarray yields a vector in $\mathbb{R}^k$, if k are the columns in the chip (the vector of the overall expression relative to each of k genes). A population of objects of many different types, that is a multiset over a given set, where an order is given, provides a vector formed by the multiplicities of the different types of objects. This means that the state of a biochemical system where some reactions happen is naturally represented by a point of a suitable space, and the evolution of the system, starting from this state, is represented by a motion of this point in this space.

In general, a vector space over an algebraic field C (for example real or complex numbers) is defined when a set of elements, called **vectors**, are given with an internal sum operation $\oplus$ which is commutative and associative, and for which a zero vector $\mathbf{0}$ exists with respect to this sum, and an opposite vector $-v$ exists for any vector v such that $v \oplus -v = \mathbf{0}$ (for two vectors v, w, then $v \oplus -w$ is often abbreviated by $v - w$). Moreover, a scalar multiplication is defined between any element $c \in C$ of the field and any vector v such that $c \cdot v$ is a vector too, and some natural axioms hold for this multiplication (distributivity of scalar multiplication with respect to vector addition $a(v \oplus w) = av \oplus aw$, and distributivity of scalar multiplication with respect to field addition $(a+b) \cdot v = a \cdot v \oplus b \cdot v$). Finally for two scalars a, b, if ab is their product in the field $a \cdot (b \cdot v) = (ab) \cdot v$ and $1 \cdot v = v$ if 1 is the multiplicative identity of the field C.

In conclusion, we can realize that measures allow us to locate objects and populations in spaces. In the simplest case this space is the unidimensional real line, but in general, it is a vector space of many dimensions.

5.4.3 Some of Euler's Jewels

In this book we encountered some spectacular results due to Leonhard Euler. For example, the famous identity $e^{i\pi} + 1 = 0$ and the infinite sum representations of e^x, $\sin x$, and $\cos x$. In this section we shortly present one of Euler's most sensational discoveries. It is the solution of a problem, called Basel problem, which was raised by the Italian mathematician Pietro Mengoli and was investigated also by Jakob Bernoulli. The question concerns the exact determination of the limit of the *p-harmonic series* for $p = 2$:

$$\sum_{1}^{\infty} \frac{1}{n^p}$$

It was proved that for $p = 1$ the harmonic series is divergent, but for $p > 1$ these series are convergent.

The following is the solution found by Euler in 1735 (since 1731, at the age of 24, he was hardly working on this problem).

According to Euler's determination of the series for sine we have:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \cdots$$

therefore:

$$\frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \frac{x^8}{9!} \cdots$$

Now the main intuition of the argument can be introduced. We know that given an algebraic equation $P(x) = 0$ of degree n with non-zero roots $a_1, a_2, \ldots, a_n$ and $P(0) = 1$ we can express $P(x)$ by means of the following product (which is easily proved to have the same roots and $P(0) = 1$):

$$P(x) = \left(1 - \frac{x}{a_1}\right)\left(1 - \frac{x}{a_2}\right)\left(1 - \frac{x}{a_3}\right)\dots\left(1 - \frac{x}{a_n}\right)$$

Euler assumed that this property extends to "infinite polynomials" and applied it to $\sin x/x$. Namely, $\sin x/x$ is equal to zero in correspondence to the following values of x:

$$\pi, -\pi, 2\pi, -2\pi, 3\pi, -3\pi\dots$$

whence:

$$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \frac{x^8}{9!}\dots = \left(1 - \frac{x}{\pi}\right)\left(1 - \frac{x}{-\pi}\right)\left(1 - \frac{x}{2\pi}\right)\left(1 - \frac{x}{-2\pi}\right)\dots$$

that is:

$$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \frac{x^8}{9!}\dots = \left(1 - \frac{x^2}{\pi^2}\right)\left(1 - \frac{x^2}{4\pi^2}\right)\left(1 - \frac{x^2}{9\pi^2}\right)\dots$$

or equivalently:

$$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \frac{x^8}{9!}\dots = 1 - \left(\frac{1}{\pi^2} + \frac{1}{4\pi^2} + \frac{1}{9\pi^2} + \dots\right)x^2\dots$$

therefore, by equating the coefficients of x^2 of the two members (the other powers do not matter, in this context), we get:

$$\frac{1}{3!} = -\frac{1}{\pi^2}\left(1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \dots\right)$$

thereby concluding that:

$$1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \dots = \frac{\pi^2}{6}.$$

The $\sin x/x$ representation, in terms of its infinite roots, provides another important result due to Wallis, which we use to prove Stirling's approximation (see Sect. 7.5).

In fact:

$$\frac{\sin(\pi/2)}{\pi/2} = \left(1 - \frac{\pi^2/2^2}{\pi^2}\right)\left(1 - \frac{\pi^2/2^2}{4\pi^2}\right)\left(1 - \frac{\pi^2/2^2}{9\pi^2}\right)\left(1 - \frac{\pi^2/2^2}{16\pi^2}\right)\dots$$

that is:

$$\frac{2}{\pi} = \frac{(4-1)}{4}\frac{(16-1)}{16}\frac{(36-1)}{36}\dots$$

which is exactly Wallis' identity, usually given by the following equality:

$$\frac{2}{\pi} = \frac{1\cdot 3\cdot 3\cdot 5\cdot 5\cdot 7\cdot 7\cdot 9\dots}{2\cdot 2\cdot 4\cdot 4\cdot 6\cdot 6\cdot 8\cdot 8\dots}.$$

After the presentation of this argument, William Dunham [178] concludes by saying: ”Surely this established that Euler’s train of thought had not derailed. If this argument could recover previously known results such as this, there seemed all the more reason to embrace his initial conclusion.” (the infinite product representation of $\sin x/x$, *Euler, Introduction to Analysis of the Infinite, Book I, pp. 154–155*)[1].

5.5 Induction and Recurrence

Natural numbers are generated by means of the basic counting process. In fact, we start from an initial number, called zero, and then by applying the successor operation, at each step, we get a new number from the last one previously generated. Apart from the specific manners to identify the elements of this sequence, the essence of this (infinite) process is the combination of **zero** and **successor**. A definition of natural number which expresses this perspective is due to the Italian mathematician Giuseppe Peano at the beginning of the last century: *zero is a number, and the successor of any number is a number too*. We remark that the successor operation is always defined and its result applied to a number n is always different from all the numbers from zero to n in the generation process. This structure characterizes the set $\mathbb{N}$ of the natural numbers and is the essence of the **mathematical induction**, a very powerful method for proving and defining properties of natural numbers. The explicit and formal identification of the principle of induction was given by Peano, but it was used, in an implicit or intuitive manner, by many mathematicians before Peano (primitive forms of induction are present in Greek mathematics) such as: Maurolico, Tartaglia, Pascal, and Dedekind. The name induction seems to be a little misleading, because it suggests an analogy to the process of inferring general properties from particular cases. However, this connection is not completely wrong. In fact, the primitive forms of induction emerge when some properties are verified in some specific numerical cases, but a pattern occurs which is not specific to single cases, because it follows a common general schema. The principle of induction claims that in order to prove or to define a property P over $\mathbb{N}$ it is sufficient to provide:

Table 5.8 The **initial step** and the **induction step** of an induction schema

i) The proof or definition of $P(0)$ (validity or definition of P for zero)
ii) The proof or definition of $P(n+1)$ by assuming that $P(n)$ holds or is defined (n generic).

[1] Another famous Euler’s jewel, related to Riemann’s zeta function on the distribution of prime numbers, is the so called sum-product formula $\sum_{k\in\mathbb{N}} \frac{1}{k} = \prod_{p\in P} \frac{p}{p-1}$, where P is the set of prime numbers.

This schema can be easily changed and extended in many ways. For example, if we want to prove P only for all the numbers greater than or equal to k, it applies by just taking k instead of zero (the property in question is not exactly $P(n)$, but $P(n+k)$ for all values of n). In other cases it is more convenient to prove or define $P(n+1)$ by assuming that $P(j)$ holds or is defined for all $j \leq n$. Or, more generally, what we want to prove or to define is not directly related to natural numbers, but to a sequence $S_0, S_1, \ldots$ of structures indexed by natural numbers. In this case the initial step applies to S_0 and the induction step proves or defines something for S_{n+1} by assuming it holds or is defined for the structure S_n (or for all structures with indexes smaller than $n+1$). In the literature there are many methods indicating different schemas of induction; however, they are all consequences of the basic form considered above. Often, the numerical variable used in the schema of induction (n in our formulation) is called the induction variable, and a proof or definition by induction is said to be developed *over the variable n*.

Given a quite obvious validity of the induction, it is surprising that its generality was discovered in relatively recent times. Moreover, despite its simplicity, very often induction proofs and definitions are not so easy to be correctly developed and their construction and analysis requires a good mathematical maturity.

We provide an initial example of proof by induction, due to Maurolico, of the fact that n^2 is the sum of the first n odd numbers:

$$n^2 = \sum_{i=1}^{n}(2i-1). \tag{5.1}$$

The initial step trivially holds. For the induction step, we need to prove the following equation, by assuming the previous one:

$$(n+1)^2 = \sum_{i=1}^{n+1}(2i-1).$$

In fact,

$$(n+1)^2 = n^2 + 2n + 1$$

but, for the induction hypothesis, we can replace n^2 by the right member of Eq. (5.1), thereby obtaining the same equation instantiated at the value $n+1$, as required by the induction step:

$$(n+1)^2 = \sum_{i=1}^{n}(2i-1) + (2n+1) = \sum_{i=1}^{n+1}(2i-1).$$

The above example shows a peculiarity of induction, that is, a sort of tautological flavor, which is intrinsic to the induction schema. In fact, it seems that we use in the proof the same claim we want to prove. This impression is related to the basic mechanism of the induction step, where essentially an hypothetical statement is transformed into a universal statement. This phenomenon is more clearly apparent if we reformulate the previous schema in a logical form where an arrow denotes the proof consequence and $\forall$ is the universal quantification asserting the validity of a proposition for all the values of the variable it refers to.

Table 5.9 Proof by induction of $\forall n \in \mathbb{N}\, P(n)$

i) $P(0)$
ii) $\forall n \in \mathbb{N} \left(P(n) \Rightarrow P(n+1)\right)$

The above induction proof is of a completely new kind with respect to the well-known traditional geometric proof of Greek mathematics, developed in the context of *figurate numbers*. For the sake of completeness, we want to mention it. Consider an arrangement of n^2 balls forming a square of side n. This arrangement can be constructed by putting an initial ball and then by surrounding it by three other balls. In general, for passing from a square to the next square, that is from n^2 to $(n+1)^2$, you need to add a row of n balls, a column of n balls and a further common ball extending the added row and column. Therefore, $2n+1$ is the whole number of balls for passing from n^2 to $(n+1)^2$. This means that in general a square of n^2 balls is obtained by increasingly arranging n odd numbers of balls which extend the square sides. Table 5.10 shows the construction of a square of 9 objects, according to the outlined procedure. The following interesting proposition, due to the Greek mathematician Nichomachus, has a very simple proof by induction.

Table 5.10 The construction of a square of 9 objects by arranging $1+3+5$ objects

$\diamond$	$\ast$ o		$\diamond \ast$ o
$\diamond$	$\ast\ast$	$\longrightarrow$	$\diamond\ast\ \ast$
$\diamond\ \diamond\ \diamond$			$\diamond\ \diamond\ \diamond$

Proposition 5.1 (Nichomachus' Theorem)

$$\sum_{i=1}^{n} i^3 = \left(\sum_{i=1}^{n} i\right)^2$$

Proof. (Proof by induction) The initial step is evident. The induction step is established by the following identities (where the passage from the last but one equation follows by induction hypothesis and by the identity $1+2+\ldots n = n(n+1)/2$, which will be proved in a next section (Sect. 5.6.1, Eq. (5.3)):

$$(1+2+\dots n+(n+1))^2 = (1+2+\dots n)^2+(n+1)^2+2(1+2+\dots n)(n+1) =$$
$$1^3+2^3+\dots n^3+(n+1)^2+n(n+1)(n+1) =$$
$$1^3+2^3+\dots n^3+(n+1)^2(n+1) = 1^3+2^3+\dots(n+1)^3. \qquad \square$$

Tables 5.11, 5.12, 5.13, and 5.14 respectively display classical definitions by induction (on the variable n) of the arithmetical sum, product, exponential, and factorial operations. These definitions have two equations, the first one for the initial case, while in the second one the left member provides the definition of the operation by assuming, by induction hypothesis, that the right member is already defined for smaller values of the variable. In the definition of product, the definition of sum is assumed, and, in the definition of exponentiation, the definition of product is assumed. They can be proved to be correct (they define the expected operations) by using the induction principle.

Table 5.11 The inductive definition of sum

$n+0=n$
$m+(n+1)=(m+n)+1$

Table 5.12 The inductive definition of product

$n*0=0$
$m*(n+1)=(m*n)+m$

Table 5.13 The inductive definition of exponentiation

$m^0=1$
$m^{n+1}=m^n*m$

Table 5.14 The inductive definition of factorial $n!$ (the product of all the numbers equal to or lower than n)

$0!=1$
$(n+1)!=n!*(n+1)$

The definition in Table 5.15 characterizes the property of prime numbers by induction (a prime number is greater than 1 and has no divisor different from itself).

Table 5.15 The sequence of prime numbers

$p(1) = 2$
$p(n+1) = min\{j \mid j > p(n)$ and $p(i)$ is not a divisor of j for $i \leq n\}$

The induction principle is a powerful tool in the analysis of discrete structures. Tables 5.16, 5.17, and 5.18 provide inductive definitions of tree and graph over a set A of labels (see Figs. 5.7 and 5.8). In particular, the definition of Table 5.17 is a special form of inductive definition, usually referred as **recursive definition** (a tree is defined in terms of smaller trees).

Table 5.16 The inductive definition of rooted ordered trees over the set A

Basis step: A single vertex is a rooted tree.

Induction step: If $T_1, T_2, \ldots, T_k$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_k \in A$, respectively, then, starting with an element $r \in A$ such that $r \neq r_1, r \neq r_2, \ldots, r \neq r_k$, and adding an edge from r to each of roots $r_1, r_2, \ldots, r_k$ a new tree is obtained having r as its root and trees $T_1, T_2, \ldots, T_k$ as its *subtrees*.

Table 5.17 The recursive definition of rooted ordered trees over the set A

A finite non-empty set $T \subseteq A$ of *nodes* is a tree if the following conditions hold:

i) there is a node $r \in T$ called the *root* of T
ii) the remaining nodes of $T/\{r\}$ are partitioned into $m \geq 0$ sets, and each of them is a tree over A, called a *subtree* of T.

An infinite tree is a tree having an infinite set of nodes N. An infinite tree is said to be **finitary** if every node of the tree has a finite number of children. An important result about infinite trees, based on induction, is the following proposition known as **König lemma**.

Proposition 5.2. *An infinite finitary rooted tree has an infinite path.*

Proof. In fact, if r is the root of the tree, then an infinite path $(p_n | n \in \mathbb{N})$ can be defined by induction by putting $p_0 = r$, which is the root of an infinite tree, and for any $n \in \mathbb{N}$, if p_n has been already defined, as the root of an infinite tree, then surely it has a child q which is the root of an infinite tree, therefore we define $p_{n+1} = q$. $\square$

Table 5.18 The inductive definition of simple oriented graph over the set A

If $a \in A$ then $a \in G(A)$.

If $G \in G(A)$, $a \in A$ and a is not a node of G ($a \notin G$), then the graph denoted by $(G+a)$ belongs to $G(A)$, where the node a is added as new node disconnected from the nodes of G. Moreover, if $G \in G(A)$, $a, b \in G$ and a is not connected to b in G, then the graph denoted by $(G+(a,b))$ belongs to $G(A)$, where the edge connecting a to b is added to G.

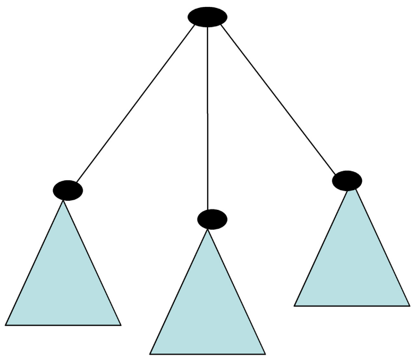


Fig. 5.7 Construction of a new tree from three already given trees

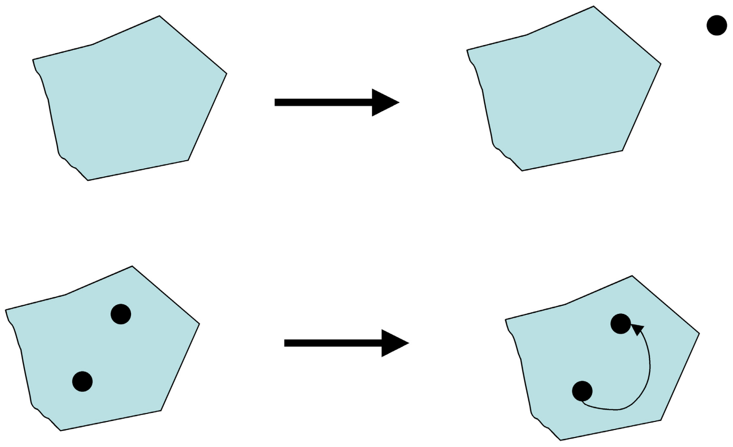


Fig. 5.8 Extending a graph with a new node (top) and with a new arc (down)

Induction is strictly related to two other notions: **iteration** and **recurrence**. Iteration is a special case of induction obtained by applying an operation a number of times, and at each step to the result of its previous application. This notion can be formally defined by means of a sort of exponentiation, as it is expressed in Table 5.19.

Table 5.19 The inductive definition of iteration

$f^0(x) = x$
$f^{n+1}(x) = f(f^n(x))$

For example, the iteration of the successor function s of exponent 3 and argument 4 is:

$$s^3(4) = s(s^2(4)) = s(s(s^1(4))) = s(s(s(s^0(4)))) = s(s(s(4))) = 7.$$

5.5.1 Fibonacci Sequence

Recurrence is a form of induction where a function is defined on natural numbers by starting from some initial values, and then by defining the values of the function in terms of some of the values it takes on smaller arguments.

One of the most famous examples of recurrent definition is the following, due to Leonardo Fibonacci (Leonardo Pisano) and presented in his famous *Liber Abaci* (1202 A.D.), whereby the Indo-Arabic positional notation for numbers was introduced in Europe:

Table 5.20 The recurrent definition of Fibonacci sequence

$F(1) = F(2) = 1$
$F(n+2) = F(n+1) + F(n)$

Table 5.21 The first eight values of Fibonacci sequence

$F(1) = 1, F(2) = 1, F(3) = 2, F(4) = 3, F(5) = 5, F(6) = 8, F(7) = 13, F(8) = 21$

The Fibonacci sequence is strictly related to the *golden ratio* $\phi = (1+\sqrt{5})/2$. In fact, for increasing values of the sequence, the ratio between a Fibonacci number and its predecessor approximates to ϕ.

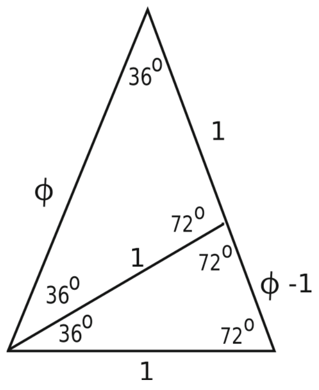


Fig. 5.9 The "golden triangle"

Let T be a "golden" isosceles triangle with vertex angle equal to one-tenth of the full circle angle, that is $\alpha = \pi/5$.

The internal triangle of Fig. 5.9 is isosceles too and similar to T because it has the same angles as T: $\pi/5, 2\pi/5$, and $2\pi/5$ (equal angles at the basis). In the golden triangle T, if 1 is the length of the basis and ϕ is the length of the oblique sides, then it is easy to find the following proportion:

$$\phi : 1 = 1 : (\phi - 1).$$

Therefore ϕ is solution of the equation:

$$\phi^2 = \phi + 1.$$

This equation has only one positive solution given by $\phi = (1+\sqrt{5})/2$, that is, the golden ratio, approximately equal to 1.618. Moreover, from the equation above, it follows that:

$$\phi = 1 + \frac{1}{\phi}$$

therefore by applying iteratively its second member to replace ϕ, we get a *continuous fraction* approximating to ϕ:

$$\phi = 1 + \cfrac{1}{1 + \frac{1}{1+\frac{1}{1+\frac{1}{1+\dots}}}}$$

It can be showed, by induction, that:

$$F(n) = \frac{\phi^n - \varphi^n}{\phi - \varphi}$$

where ϕ is the positive root and φ the negative root of the equation $x^2 = x + 1$ (*square* = *successor*) that is:

$$F(n) = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}}$$

being φ smaller than 1, in absolute value, for n sufficiently large, $F(n)$ approximates to $\phi^n/\sqrt{5}$ (whence $F(n+1)/F(n)$ approximates to ϕ).

It is easy to verify that $-\varphi = \frac{1}{\phi}$ and that $1 + \varphi = 1 - \frac{1}{\phi} = \frac{1}{\phi^2}$. The value $1/\phi^2$ is also called *circle golden ratio*, and the angle $2\pi/\phi^2$ is frequently occurring in nature structures.

The problem, which suggested to Fibonacci his sequence, was the growth of a population of rabbits. The (ideal) rule of their reproduction establishes that each rabbit generates one offspring (a couple of rabbits, an offspring couple), but a newborn at some generation time, say *i*, can generate rabbits only when it becomes adult, that is, at generation $i+2$ (one step is necessary to a newborn to become an adult). Fig. 5.10 exemplifies this kind of development by means of a tree.

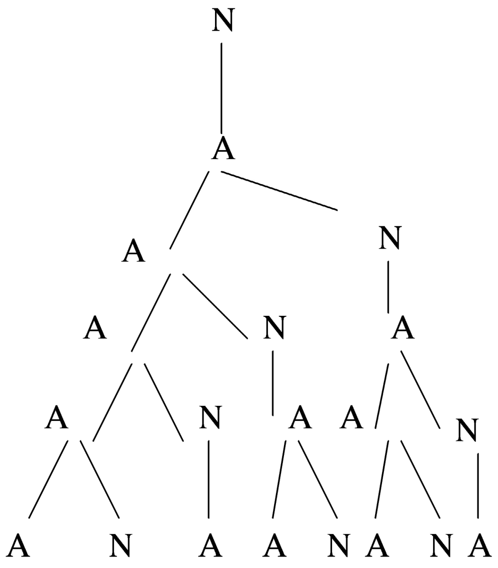


Fig. 5.10 Five generations of Fibonacci development

Fibonacci sequence is surprisingly ubiquitous in processes of biological morphogenesis and development. In DNA double strands, the angle formed by two consecutive nucleotides is approximately $\pi/5$, and the ratio between the two grooves of DNA helix is near to ϕ. Moreover, Fibonacci sequence and golden ratio are found in many patterns formed by leaves and flowers during plant developments. In the case of phyllotaxis, Fibonacci spirals and "golden angles" frequently occur. A Fibonacci spiral, based on Fibonacci sequence, is given in Fig. 5.11. The most frequent golden angle in phyllotaxis (the arrangement of leaves in stems) is $2\pi(1/\phi^2)$, approximately equal to 137.52^o.

The reason of such geometrical phenomena seems to be due to the optimal balance, realized by these patterns, between two opposite forces: i) a repulsive force between nearby individuals (leaves which need to have enough space around to receive enough light without interference from other leaves), and ii) an attractive force due to a whole population requirement (a tree tending to have the maximum number of leaves on its branches).

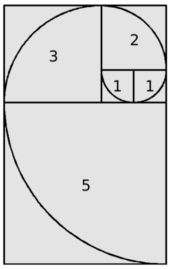


Fig. 5.11 Fibonacci's spiral

5.5.2 Arithmetic and Logic

Arithmetic and logic are deeply related. Both have a foundational nature for the whole mathematics. Here we present some basic concepts of First-Order Logic (FOL), mainly developed in the last two centuries (books [172, 185, 189] are some classical references, [184] is a succinct presentation).

A **model**, or a (relational) structure, $\mathscr{M}$ is given by: i) a set D, called *domain* of $\mathscr{M}$, ii) some elements $a, b, \ldots \in D$, called *individual constants* of $\mathscr{M}$, and iii) some operations $f, g, \ldots$ and relations $R, Q, \ldots$ over D (to each operation and relation an *arity* is associated that specifies the number of the arguments). Relations are considered equivalent to predicates, that is, functions that map their arguments to a truth value (1,0) in correspondence to the fact that the relation holds or does not hold between them. Usually $\mathscr{M}$ is indicated by:

$$\mathscr{M} = (D, a, b, \ldots f, g, \ldots, R, Q, \ldots).$$

The set $Term(\mathscr{M})$ of the terms over $\mathscr{M}$ is given by all the expressions that can be constructed, in the usual algebraic sense, by induction, by applying operations and relations of $\mathscr{M}$ to the individual constants of $\mathscr{M}$, or to already constructed terms. For example, if f has arity 1 and g has arity 2, then the following are terms over the model $\mathscr{M}$:

$$f(a),\ g(a,b),\ f(g(a,b)),\ g(a,f(a)),\ f(g(a,f(a))).$$

An equation such as $g(a,f(a)) = b$ means that by applying the operation f to the constant a, we get an element of D, say c, and by applying g to the pair (a,c) we get b. This means that $g(a,f(a))$ is considered as the denotation of the element of D obtained by applying the operations according to the usual way algebraic expressions are evaluated (in the order specified by parentheses). However, we can also consider a term as an uninterpreted expression (a tree) constructed from constants, operations, commas, and parentheses, disregarding its denotation. If we want to be precise we write $g(a,f(a))$ to mean the element of D (if it exists) denoted by the term, while we write $\lceil g(a,f(a)) \rceil$ (or something analogous) to denote the uninterpreted term. In formal logic this distinction is fundamental and a lot of different terminologies are used to tell this intrinsic difference. Very often a denoting symbol is said to be **used** while it is said to be **mentioned** when it refers to itself.

A **first-order logic signature** (FOL signature) Σ is a set of symbols for objects, operations, and relations. A Σ-model is a relational structure where objects, operation and relations are denoted by the symbols of Σ.

We call $Term(\Sigma)$ the uninterpreted terms over the signature Σ, and $Term_V(\Sigma)$ the terms over the signature Σ and a set V of variables.

Terms over a signature Σ can be transformed into terms over a model when the symbols of Σ are interpreted in the model.

We call $Term_V(\mathscr{M})$ the terms over the model $\mathscr{M}$ where variables V range over the domain of $\mathscr{M}$, and $Term(\mathscr{M})$ the terms of $\mathscr{M}$, where variables do not occur.

Atomic formulae (over a signature) are expressions $R(t_1,t_2,\ldots,t_k)$ where R is a k-ary relation symbol and $t_1,t_2,\ldots,t_k$ are terms.

First-Order Logic or FOL formulae are constructed by starting from atomic formulae by using the First-Order Logical operators (FOL), which are:

1. **connectives** denoted by: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$; respectively for **negation** (not), **conjunction** (and), **disjunction** (or), **implication** (if), **equivalence** (iff), and
2. **quantifiers** denoted by $\forall$ (for all), the **universal quantifier**, and $\exists$ (there exists), the **existential quantifier**.

Let us associate connectives with the **boolean functions** described by their **truth tables** defined by the following boolean equations:

- $\neg 1 = 0$, $\neg 0 = 1$
- $(1 \wedge 0) = (0 \wedge 1) = (0 \wedge 0) = 0$, $(1 \wedge 1) = 1$
- $(1 \vee 0) = (0 \vee 1) = (1 \vee 1) = 1$, $(0 \vee 0) = 0$
- $(0 \rightarrow 1) = (1 \rightarrow 1) = (0 \rightarrow 0) = 1$, $(1 \rightarrow 0) = 0$
- $(1 \leftrightarrow 1) = (0 \leftrightarrow 0) = 1$, $(0 \leftrightarrow 1) = (1 \leftrightarrow 0) = 0.$

Let us define **universal** and **existential quantifications** in the following way, where P stands for any unary predicate, D for the domain of a given model, and x for a variable ranging over D (1 stands for *true*, 0 stands for *false* and assume that $0 < 1$):

$$\forall_D P(x) = min\{P(x) \mid x \in D\}$$

$$\exists_D P(x) = max\{P(x) \mid x \in D\}.$$

If we apply connectives and quantifiers to atomic formulae over a model $\mathscr{M}$ and variables V, then we generate the **formulae** over $\mathscr{M}$ and over a set of variables V. A variable occurrence is free if it is not quantified. Formulae denote k-ary predicates if k different free variables occur in them, while they denote a truth value if no free variable occurs in them. Formulae without free variables are called **sentences** over $\mathscr{M}$ (over its signature). A FOL **theory** is a set of sentences over a FOL signature. We say that a sentence φ over $\mathscr{M}$ holds in $\mathscr{M}$ if it denotes the *true* value (say 1) and we write:

$$\mathscr{M} \models \varphi.$$

We can obtain a definition of logical validity by means of the following notion of **interpretation** of FOL formulae over FOL models.

Let $\mathscr{M}$ be a model of signature Σ and domain D, and $F(\Sigma, V)$ the FOL formulae over the signature Σ and the set V of variables. An **interpretation** of $F(\Sigma, V)$ in $\mathscr{M}$ is defined by giving to the constant, operation, and relation symbols of Σ their meaning in the model $\mathscr{M}$, that is, by giving to the terms their denotation in D with the variables in V ranging over D. Then, for an atomic formula $\varphi = R(t_1, t_2, \ldots, t_k)$ where no variable occurs, the interpretation τ is defined by stating $\tau(\varphi) = 1$ if and only if the relation R holds on the terms $t_1, t_2, \ldots, t_k$ of $\mathscr{M}$. The interpretation τ is extended to all the sentences of $F(\Sigma, V)$, by requiring the following conditions for the truth values of all non-atomic sentences of $F(\Sigma, V)$:

- $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$
- $\tau(\varphi \vee \psi) = \tau(\varphi) \vee \tau(\psi)$
- $\tau(\neg\varphi) = \neg\tau(\varphi)$
- $\tau(\varphi \to \psi) = \tau(\varphi) \to \tau(\psi)$
- $\tau(\varphi \leftrightarrow \psi) = \tau(\varphi) \leftrightarrow \tau(\psi)$
- $\tau(\forall P(x)) = \forall_D \tau(P(x))$
- $\tau(\exists P(x)) = \exists_D \tau(P(x))$.

A sentence of $F(\Sigma, V)$ is **logically valid** iff it is true in every FOL model of signature Σ. It is **satisfiable** if it is has a FOL model.

Given a FOL theory Φ and a FOL sentence φ we say that φ is a **logical consequence** of Φ, by writing:

$$\Phi \models \varphi$$

when φ is true in all models where all the sentences of Φ are true.

The discovery of the first (informal) counter-model proof, showing the independence of the **fifth Euclidean axiom** (on the unicity of the parallel line) from the remaining Euclidean axioms was essentially based on the notion of interpretation.

In fact, an interpretation of the Euclidean axioms was found where all the Euclidean axioms were true apart from the fifth axiom which was false. This discovery was the first *meta-theorem* of geometry, proving the mathematical impossibility of deducing the fifth axiom from the other ones.

Let us consider an example of FOL theory. We use: i) three variables x, y, z ranging over the individuals of some biological population with sexual reproduction; ii) four *predicate symbols* G, A, M, F, that is, symbols of relations over the domain consisting of our individuals, such that:

$G(x,y)$ means *x generated y*,
$A(x,y)$ means *x is an ancestor of y*,
$M(x)$ means *x is a male*,
$F(x)$ means *x is a female*.

By using logical symbols, predicate symbols, variables, and parentheses we can put the following sentences in formulae expressing usual facts about sexual reproduction:

- $\forall x(\exists y(G(y,x) \wedge M(y)))$ (Everybody has a father)
- $\forall x(\exists y(G(y,x) \wedge F(y)))$ (Everybody has a mother)
- $\forall x(M(x) \vee F(x))$ (Everybody is male or female)
- $\forall x(\forall y(G(x,y) \rightarrow A(x,y)))$ (Parents are ancestors)
- $\forall x(\forall y(\forall z(A(x,y) \wedge G(y,z)) \rightarrow A(x,z)))$ (The ancestors of parents are ancestors)
- $\forall x(\neg(A(x,x)))$ (Nobody is self-ancestor).

These sentences constitute the **axioms** of a theory, consisting of all the logical consequences of them. Can we interpret them in a different domain, with different meanings for predicate symbols, in such a way that they turn to be true also in this new interpretation? With a detailed analysis, we can discover that these axioms cannot be fulfilled by a real biological population, consisting of a finite number of individuals. On the other hand, we can interpret coherently these formulae on natural numbers. However, in which sense can we prove that the father's unicity is not a logical consequence of the given axioms? And, in which sense the non-existence of a common ancestor for all individuals, $\neg\exists x(\forall y A(x,y))$, is a logical consequence of them? Could we find an algorithm generating all the logical consequences of these axioms? What does it mean $\exists y \forall x(G(y,x) \wedge M(y))$? Can this axiom be added to the theory, while preserving its consistency (avoiding a contradiction)?

Peano's axioms of natural numbers are given by the following sentences (P is any predicate).

0 is a number.
Each number has a successor.
0 is not successor of any number.
Distinct numbers have distinct successors.
If $P(0)$ and, for every number n, the implication $P(n) \rightarrow P(n+1)$ holds, then, for every n, proposition $P(n)$ holds.

Theory *PA* is the following theory which expresses Peano's axioms as FOL formulae (on symbols $0,1$ for distinct constants and $+$ for a binary operation):

$\neg\exists x(0 = x+1)$
$\forall x(\forall y((x+1 = y+1) \rightarrow (x = y)))$
$P(0) \wedge \forall x(P(x) \rightarrow P(x+1)) \rightarrow \forall x(P(x))$.

One of the most important results in formal logic, originally due to Kurt Gödel establishes the existence of algorithmically definable logical calculi that can express completely the FOL logical validity. In fact, let $\vdash$ be a derivation relation over uninterpreted formulae defined in some algorithmic way (many of such calculi were found in the last century), such that we write:

$$\Phi \vdash \varphi$$

to denote that formula φ is **derivable** from a set of formulae Φ, then Gödel's **completeness theorem** asserts the following equivalence:

$$\Phi \models \varphi \quad \text{if and only if} \quad \Phi \vdash \varphi.$$

Some important properties of FOL are briefly reported:

1. The completeness of FOL is related to another important property of this logic called **compactness**. It asserts that a theory Φ has a model (there is a model where all the sentences of Φ hold) iff every finite subset of Φ has a model.
2. Any sentence of a FOL theory can be transformed into an equivalent formula, which is called Skolem normal form, where quantifications are all at the beginning of a propositional formula.
3. By introducing suitable operation symbols, any formula can be transformed into an equisatisfiable formula φ, called Herbrand expansion of φ, where only universal quantifications are, at the beginning of a propositional formula.
4. Herbrand expansions imply that any FOL theory is equisatisfiable with a propositional theory (which in general may be infinite even if the original FOL theory is finite).
5. From the propositional representation of FOL theories, it can be derived that any satisfiable FOL theory has a model with numerable domain. This phenomenon shows a deep relationship between FOL and natural numbers, and it is related to the existence of non-standard models of real numbers (Skolem's paradox: Real numbers that are a non-denumerable set have a denumerable FOL model).

FOL can be considered the formal language of mathematics because any mathematical theory can be represented as a FOL theory by using a suitable representation of its axioms as FOL sentences.

5.5.3 A Glimpse of the History of Formal Logic

Formal Logic originates with Aristotle and concerns the human activity of drawing inferences. Of course, since the time when language developed, man has deduced conclusions from premises, but Aristotle inaugurated the systematic study of the rules involved in the construction of valid reasoning. The first important discovery of this approach was that the logical structure of sentences and deductions is given by relations between signs in abstraction from their meaning. This aspect motivates the attribute *formal*. Modern logic, as it has been developed since the middle of 19th century, emphasizes this aspect by developing notational systems able to represent and analyze the logical form of sentences and deductions. In this sense formal logic is also referred as *Symbolic Logic*, or also *Mathematical Logic*, because the emergence of the symbolic perspective was stimulated by certain trends within mathematics, namely, the generalization of algebra, the development of the axiomatic method, especially in geometry, and the *reductionism*, that is, the tendency, especially in analysis, to find basic concepts for a foundation of the whole mathematics. The elaboration of the formal method in modern logic was chiefly founded by the works of: Gottfried Wilhelm von Leibniz (1646–1716), De Morgan (1806–1871), Boole (1815–1864), Peirce (1839–1914), Schröeder (1841–1902), Frege (1848–1925), Peano (1858–1932), Hilbert (1862–1943), Russell (1872–1970), Löwenheim (1878–1957), Skolem (1887–1963), Tarski (1901–1983), Gödel (1906–1978), Herbrand (1908–1931), Gentzen (1909–1945), and Turing (1912–1954). The essential aspect of formal methods, in the representation of sentences and deductions, consists of a clear distinction between *syntax* and *semantics*. This is an intrinsic feature of any *formal language*, as opposed to natural language. Syntax establishes which (linear) arrangements of symbols of a specified alphabet have to be considered well-formed expressions, the categories in which they are classified, and symbolic rules according to which some relations between expressions are defined. Semantics establishes how to define the general concepts of interpretation, satisfiability, truth, compatibility, independence, and entailment. This distinction does not mean that syntax and semantics are opposed, but rather, complementary. Syntactic categories, and syntactic rules, are given in such a way that important semantic concepts can be adequately expressed in syntactic terms. When this is possible those concepts can be *calculated* or *mechanized*. One of the most important questions investigated in modern logic is that of the completeness of given logical systems. A way of expressing completeness, as in celebrated Gödel's completeness theorem (1930), is that any sentence that is provable, by means of the syntactic rules of a logical system (e.g. Russell's system of *Principia Mathematica*), is universally valid, that is, true in all of its possible interpretations. Therefore, *form* and *content* of sentences are distinguished in order to achieve *effective*, general methods in elaborating and understanding sentences. Syntax defined apart from semantics, can individuate processes which elaborate formulae by using only symbolic structures, which by virtue of their finite nature, can be encoded as physical states of a machine (the famous Gödel's incompleteness theorem was based on a suitable encoding of FOL syntax in arithmetical terms). Semantics, which deal with no particular interpretation of symbols, can define general semantic concepts which are the basis of any notion of logical validity (truth in all possible worlds, according to a definition due to Leibnitz).

5.6 Series and Growths

Fibonacci sequence is related to a growth process. Growth processes occur in any life phenomenon and at each level from the cell to the most complex organisms, and to biological populations. The mathematical form of a growing process is very often given by a series, that is, a sequence where at each step an additive term is added to the term which is the outcome of the preceding step.

5.6.1 Arithmetical Progressions

The simplest form of growth is represented by a constant additive increment. It is typical of a population where the number of individuals increases at each step by a constant number of k individuals. If the initial quantity is $F(0)$ then the formula which provides the population size at step $n+1$ is given by:

$$F(n+1) = F(n) + k$$

that, by a trivial induction argument, gives for $n > 0$:

$$F(n) = F(0) + nk.$$

If the population, at each step i increments of ki units, then, the formula providing the size $G(n)$ of such a population, by assuming that $G(0) = 0$, is:

$$G(n) = \sum_{i=1}^{n} F(i) = \sum_{i=1}^{n} ik$$

The value $G(n)$ is thus:

$$G(n) = k\frac{n \cdot (n+1)}{2}. \tag{5.2}$$

This equation can be easily proved by induction. Without loss of generality we consider the case $k = 1$, by denoting $T(n)$ the corresponding sum. Of course, $T(0) = 0$, therefore the formula holds for the value 0 of n. Let us assume that the formula holds for n, then the following chain of equations shows its validity for $n+1$:

$$T(n+1) = \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+1)(n+2)}{2} \tag{5.3}$$

(the reader is advised to search for a very beautiful geometrical proof of this formula, due to Gauss, when he was a child).

5.6.2 Figurate Numbers

The study of figurate numbers was an old subject of Greek mathematics. However, its investigation in the 17th century was the beginning of important discoveries in discrete mathematics. Here, we will show a phenomenon somewhat related to Nichomachus' theorem.

A triangular configuration of objects has the following structure:

o
o o
o o o
o o o o
o o o o o

The total number of objects of a triangular configuration T_n of *basis* n is given, according to formula 5.2:

$$T_n = n(n+1)/2 \tag{5.4}$$

Of course, $T_n + T_{n-1} = n^2$ therefore $T_n + T_n - n = n^2$, that is, $2T_n = n^2 + n$, which provides another way of proving the equation above.

A square number n^2 can be represented by arranging n^2 objects in a configuration Q_n of n rows of objects each of them with n objects. Now let us evaluate the sum of the squares Q_i for i ranging from 1 to n (see Fig. 5.12).

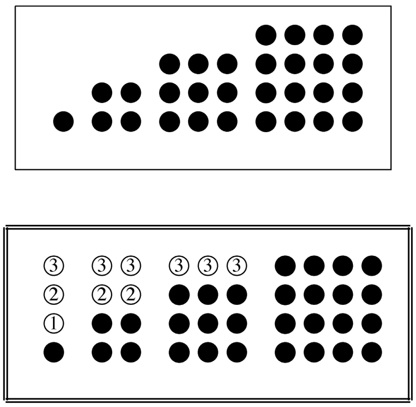


Fig. 5.12 Square configurations of increasing side (top) and their extension with added (numbered) objects (bottom)

The following equation expresses the way of obtaining the top configuration of Fig. 5.12 from the bottom configuration of the same figure, by removing three sums of triangular numbers (the number of circles with number 1, with number 2, and with number 3):

$$\sum_{i=1,n} i^2 = nT_n - \sum_{i=1,n-1} T_i \tag{5.5}$$

but by formula 5.4 we have:

$$\sum_{i=1,n} T_i = \sum_{i=1,n} i(i+1)/2 = 1/2 \left(\sum_{i=1,n} i^2 + \sum_{i=1,n} i \right)$$

that is:

$$\sum_{i=1,n} i^2 = 2 \sum_{i=1,n} T_i - \sum_{i=1,n} i \tag{5.6}$$

therefore, by equating the right members of Eqs. (5.5) and (5.6), we have:

$$nT_n - \sum_{i=1,n-1} T_i = 2\sum_{i=1,n} T_i - \sum_{i=1,n} i$$

that is:

$$nT_n - \sum_{i=1,n-1} T_i = 2 \sum_{i=1,n-1} T_i + 2T_n - T_n$$

$$3 \sum_{i=1,n-1} T_i = (n-1)T_n$$

$$\sum_{i=1,n-1} T_i = \frac{(n-1)T_n}{3}.$$

Finally, from Eq. (5.5) it follows:

$$\begin{aligned}\sum_{i=1,n} i^2 &= nT_n - \frac{(n-1)T_n}{3} \\ &= \frac{3nT_n - (n-1)T_n}{3} \\ &= \frac{T_n(3n-n+1)}{3} \\ &= \frac{n(n+1)(2n+1)}{6}.\end{aligned}$$

Formulae for the sums of powers (up to the 17th power) were discovered by Faulhaber, a mathematician of the 17th century. Faulhaber's formulae follow a general pattern which can be proved by induction.

Theorem 5.3 (Faulhaber). *The sum of powers of degree k, for $k > 0$, has the following general form, where $P_k(n)$ is a polynomial of degree k:*

$$\sum_{i=1}^{n} i^k = \frac{n^{k+1}}{k+1} + P_k(n)$$

Proof. (by induction). Let

$$S_k(n) = \sum_{i=1}^{n} i^k$$

If $k = 1$, then $S_k(n) = T(n) = n(n+1)/2$ and the statement of the theorem holds. Assume that a polynomial of degree $(k-1)$ was determined such that:

$$S_{k-1}(i) = i^k/k + P_{k-1}(i) \tag{5.7}$$

then, we can put:

$$\sum_{i=1}^{n-1} S_{k-1}(i) = \sum_{i=1}^{n-1} i^k/k + \sum_{i=1}^{n-1} P_{k-1}(i)$$

therefore:

$$\sum_{i=1}^{n-1} S_{k-1}(i) = \frac{S_k(n)}{k} + Q_k(n-1) \tag{5.8}$$

where $Q_k(n-1) = P_{k-1}(n-1) + P_{k-1}(n-2) + \ldots P_{k-1}(1)$ is, by induction hypothesis, a polynomial of variable n having degree k. On the other hand, the sum $\sum_{i=1}^{n-1} S_{k-1}(i)$ can be put in the following way:

$$\begin{array}{llllll}
1 & & & & & \\
+1 & +2^{k-1} & & & & \\
+1 & +2^{k-1} & +3^{k-1} & +\ldots+ & & \\
\vdots & \vdots & \vdots & \vdots & & \\
+1 & +2^{k-1} & +3^{k-1} & +\ldots+ & (n-2)^{k-1} & \\
+1 & +2^{k-1} & +3^{k-1} & +\ldots+ & (n-2)^{k-1} & +(n-1)^{k-1}
\end{array}$$

or equivalently:

$$(n-1)1^{k-1} + (n-2)2^{k-1} + (n-3)3^{k-1} + \ldots + [n-(n-1)](n-1)^{k-1}$$

which is the difference of two terms:

$$\begin{cases} n1^{k-1} + n2^{k-1} + n3^{k-1} + \ldots = nS_{k-1}(n-1) \\[2ex] 1^k + 2^k + 3^k + \ldots \qquad\qquad = S_k(n-1) \end{cases}$$

In conclusion:

$$\sum_{i=1}^{n-1} S_{k-1}(i) = nS_{k-1}(n-1) - S_k(n-1) \tag{5.9}$$

and if we equate the right members of Eqs. (5.8) and (5.9), we get:

$$nS_{k-1}(n-1) - S_k(n-1) = \frac{S_k(n)}{k} + Q_k(n-1)$$

that is:

$$nS_{k-1}(n-1) - Q_k(n-1) = \frac{S_k(n)}{k} + S_k(n-1)$$

which, adding n^k to the both members, becomes:

$$nS_{k-1}(n-1) - Q_k(n-1) + n^k = \frac{S_k(n)}{k} + S_k(n-1) + n^k$$

that is

$$\frac{k+1}{k} S_k(n) = nS_{k-1}(n-1) - Q_k(n-1) + n^k$$

$$S_k(n) = \frac{k}{k+1}\left[nS_{k-1}(n-1) - Q_k(n-1) + n^k\right]$$

and, by Eq. (5.7):

$$S_k(n) = \frac{k}{k+1}\left[n[(n-1)^k/k + P_{k-1}(n-1)] - Q_k(n-1) + n^k\right]$$

and finally:

$$S_k(n) = \frac{1}{k+1}\left[n(n-1)^k + kP_{k-1}(n-1) - kQ_k(n-1) + kn^k\right]$$

which corresponds to the statement of the theorem. □

5.6.3 Geometrical Progressions

A second case of growth is that of a population where at each step each individual generates k offspring, therefore at each step the population size increases of its size multiplied by k (no external effect and no death is considered). In this case, assuming an initial size equal to 1, the formula providing the population size at generation step n is:

$$G(n) = \sum_{i=0}^{n} k^i.$$

We prove that

$$G(n) = \frac{k^n - 1}{k-1}.$$

In fact, the proof of this equation is the following:

$$kG(n) - G(n) = \sum_{i=1}^{n+1} k^i - \sum_{i=0}^{n} k^i = k^{n+1} - 1$$

that is,

$$G(n)(k-1) = k^{n+1} - 1$$

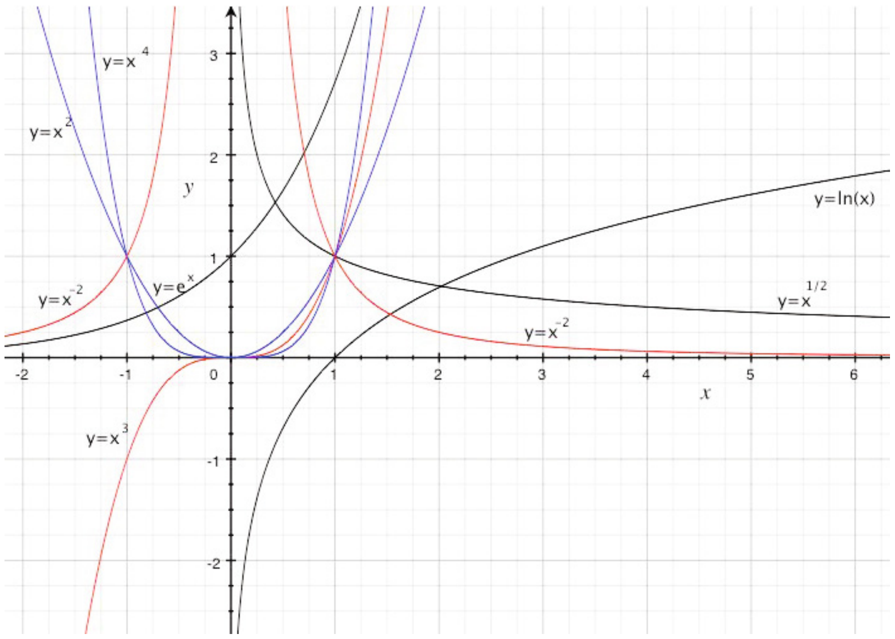


Fig. 5.13 The curves of some simple growths

therefore, the final formula is obtained:

$$G(n) = \frac{k^{n+1} - 1}{k - 1}.$$

Figure 5.13 shows many kinds of growth functions.

Sigmoid functions occur when growth processes are between two thresholds and the growth rate is inversely proportional to the growth level. This happens, for example, in the exponential sigmoid $2/(1+e^{-t})$ displayed in Fig. 5.14.

Hill functions are another kind of **sigmoid functions** with shapes such as that one displayed in Fig. 5.15:

$$H(x) = \frac{Ax^n}{B^n + x^n}.$$

5.6.4 Logistic Maps

The *logistic map* is a a growth rule which provides the fraction of living individuals, in a population having a maximum size denoted by 1. If x denotes the fraction of living individuals, then $f(x)$ represents the fraction of them at the next generation (like in the previous cases, the process evolves along a discrete sequence of generation steps):

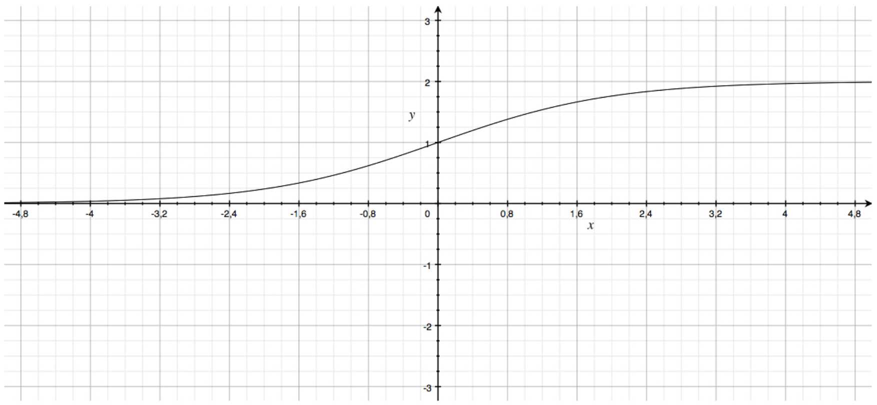


Fig. 5.14 An exponential sigmoid function

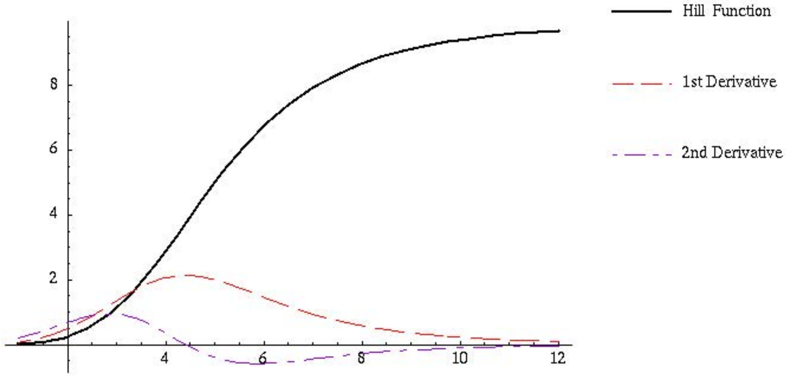


Fig. 5.15 A Hill function where $A = 10$, $B = 5$, $n = 4$

$$f(x) = ax(1-x).$$

This form expresses the size at the next step, as depending at the same time on the number of living individuals x and on the number of potential living individuals which can be added for reaching the maximum size. The parameter a is specific of any particular kind of logistic rule. If we want to move from the function f to the time series $G_f(n)$, along the generation steps, we need to consider the iteration of f, therefore:

$$G_f(n) = f^n(x_0)$$

where x_0 is the size of the population at the initial time.

Despite the simplicity of their form, logistic maps provide very rich behavioral forms. When the parameter a is close to a value around 3.57, logistic growths become chaotic. This implies many typical phenomena: *sensitivity* (starting from very close values, then very different dynamics could be generated), *transitivity* (the dynamics takes, in an erratic way, almost all of its possible values), and *dense periodicity* (any value is close, at any distance, to periodical states). A detailed analysis of chaotic phenomena, and of the chaotic aspect of logistic maps, is outside the scope of our discussion (see [183]), the literature about logistic map and related aspects is very rich and full of spectacular phenomena (very interesting images can easily found on the web), we refer to [162] for a general analysis of iterated maps and historical notes. In a very intuitive manner, chaos means impossibility of reliable predictions. The surprising fact disclosed by phenomena such as logistic maps is *deterministic chaos*. The discovery of the existence of this possibility is one of the deepest mathematical results of the last century, and can be found not only in complex systems, but in very simple cases, as logistic growth shows. Then, in a deterministic chaotic phenomenon, unpredictability is not due to a lack of rule, but to the impossibility of applying it with the exactness required for a reliable adequacy to the reality modeled by it.

5.6.5 Natural Exponential Growth

Geometrical progressions are growths of exponential type. A pure exponential law is of kind a^n. This happens in a population where, starting from one individual, at each generation step every individual generates a individuals, but no individual at a generation step can survive at the next generation step. Between arithmetical and exponential growths there are intermediate levels, for example *power laws* such as $an^2, an^3, \ldots$. Of course, growths can be modeled with continuous time, and the base of an exponential law can be lower than one, or equivalently, the exponents can take negative values. However, here we want to mention the relationship between growth and the Euler constant e, by following an analysis due to the Italian mathematician of the last century Bruno De Finetti.

Let us consider a simple process where in a unitary time an initial quantity a_0 becomes $2a_0$. We can assume that this process is due to a uniform increment along the considered time interval. This means that if we partition the interval into n subintervals, then during each of them a quantity a_0/n is added to the initial quantity. In fact, $a_0 + n(a_0/n) = 2a_0$. Now, let us change this increment rule, by assuming that at each step the increment by the $1/n$ factor is not applied to the initial quantity, but to the global quantity cumulated along the preceding steps. We can express this *natural growth rule* by the following inductive definition ($i > 0$):

$$a_{i+1} = a_i + a_i/n$$

this means that:

$$a_{i+1} = a_i(1 + 1/n)$$

therefore

$$a_n = a_0(1 + 1/n)^n.$$

It can be shown that, for any n, $2 < (1 + 1/n)^n < 3$ and, for increasing values of n, the value $(1 + 1/n)^n$ approaches the limit value $e = 2,71828\ldots$.

In conclusion, the hypothesis of infinitesimally distributed increment (in time and in quantity) implies that after a unitary time, an initial quantity a_0 become ea_0, and the growth law, along time t, is a_0e^t.

5.6.6 Asymptotic Orders and Infinitesimals

Growth processes can be compared according to their speeds. An exponential growth with base greater than 1 seems to be faster than a quadratic growth law. This is not completely right, because comparing $100t^2$ with 2^t for $t = 2$ yields the values 400 and 4 respectively, although the first expression is quadratic and the second one is exponential. However, for all the values $t \geq 15$ the exponential function overtakes the quadratic one by a gap increasing with the value of t. This means that a comparison of growths is correct when it refers to the overall growth behavior, possibly by forgetting a finite number of values. This intuition can be formalized by the notion of **almost everywhere** comparison of positive non-decreasing functions.

A positive non-decreasing function f defined on the set $\mathbb{N}$ is less than (or equal to) a function g of the same type when, for some $n_0 \in \mathbb{N}$, $f(n) \leq g(n)$ for every value $n > n_0$. We write $f(n) \leq_\infty g(n)$ to denote this fact.

$O(g)$ denotes the class of positive non-decreasing functions $\{f \mid f(n) \leq_\infty g(n)\}$. Often, with an abuse of notation, $f \in O(g)$ is indicated by writing $f = O(g)$ (by saying that f **is a O of** g). Symmetrically, $\Omega(g)$ denotes the class of functions $\{f \mid g(n) \leq_\infty f(n)\}$.

Any positive real number ε can be seen as a constant function yielding the value ε everywhere.

An **infinitesimal** is a function f defined on the set $\mathbb{N}$ and taking values in the set of real numbers such that their absolute values (the function yielding the absolute values of $f(n)$) belong to $O(\varepsilon)$ for any $\varepsilon > 0 \in \mathbb{R}$. The function $f(n)$ is an **infinite** if $1/f(n)$ is an infinitesimal. Two functions $f(n), g(n)$ are **asymptotically equivalent** if $1 - f(n)/g(n)$ is an infinitesimal. The class of functions $f(n)$ such that, for some positive constant $k \in \mathbb{R}, k \neq 0$, $kf(n)$ and $g(n)$ are asymptotically equivalent is denoted by $\theta(g)$.

All the concepts of classical infinitesimal calculus can be developed in terms of infinitesimals. For example, the classical condition defining the number z as the limit of a real function $f(x)$, for x approaching a value x_0, usually denoted by the following notation:

$$\lim_{x \to x_0} f(x) = z$$

is equivalent to the requirement that:

$$f(x_0 + 1/n) - z$$

and:

$$f(x_0 - 1/n) - z$$

are two asymptotically equivalent infinitesimals.

The function $f(n)$ is a **zero** of $g(n)$ if $f(n)/g(n)$ is an infinitesimal, and $o(g)$ denotes the class of the zeros of g. If $f \in \theta(g)$, then f and g are said to be of the same **asymptotic order**, written $f =_o g$. If $f \in o(g)$, then f is said to be of a smaller asymptotic order with respect to g, and we also write $f \leq_o g$.

The growth of functions from reals to reals can be compared by considering their restrictions to the set of the natural numbers. It can be easily shown that polynomials of the same degree have the same asymptotic order, while those of smaller degrees have smaller asymptotic orders. Moreover, $log_b(x) \leq_o x$ (for any $b \geq 2$), and $x \leq_o a^x$ (for any $a > 1$). The following rules can be also proved. If $f(x) \leq_o g(x)$, then $f(x) + g(x) =_o g(x)$. If $f(x) \leq_o g(x)$ and $h(x)$ is an increasing real function, then $f(x)h(x) \leq_o g(x)h(x)$. If $f(x) \leq_o g(x)$ and $h(x)$ is a non-decreasing real function, then $f(h(x)) \leq_o g(h(x))$ and $h(f(x)) \leq_o h(g(x))$. By means of these rules (about polynomials, logarithms, exponentials, sums, products, and compositions) it is easy to asymptotically compare a large class of real functions.

However, manipulating infinitesimals and infinites requires one to be very careful, because very often the intuition may lead to wrong consequences. Let us mention, for example, a surprising phenomenon related to the following **harmonic sequence**:

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \dots$$

of course it is an infinitesimal, but it can be easily shown that the following **harmonic series** H_n is an infinite:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}.$$

Table 5.22 The infinity of harmonic series (Oresme, 1323 –1382)

$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$	=
$1 + \frac{1}{2} + (\frac{1}{3} + \frac{1}{4}) + (\frac{1}{5} + \dots + \frac{1}{8}) + \dots$	>
$1 + \frac{1}{2} + (\frac{1}{4} + \frac{1}{4}) + (\frac{1}{8} + \dots + \frac{1}{8}) + \dots$	=
$1 + \frac{1}{2} + \frac{2}{4} + \frac{4}{8} + \dots$	=
$1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \dots$	= an infinite

There exists a constant $\gamma = 0.577215\dots$ (it is unknown whether γ is irrational), called Euler-Mascheroni constant such that:

$$H_n = \ln(n) + \gamma.$$

It can be shown that the function

$$\zeta(k) = \sum_{i=1}^{\infty} \frac{1}{i^k}$$

converges for any $k > 1$. The function ζ is extremely important in number theory. Here we want to recall, from Sect. 5.4.3, the relationship, discovered by Euler, between $\zeta(2)$ and π:

$$\zeta(2) = \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}.$$

We remark that the name of Euler is strictly connected to almost all important special numbers of mathematics. His crucial role in the definition of e (called Euler number or Napier constant) and his famous formula connecting e with i were already explained. However, apart from the constant γ mentioned above, which frequently occurs in many contexts, he was also the mathematician who popularized the symbol π for the famous Archimedes' number 3.1415926535... which, maybe, is the irrational (trascendental) number of which we know the largest number of decimal digits (about 10^{12} digits). This possibility is due to the BaileyBorweinPlouffe formula (BBP formula) discovered in 1995, allowing for the computation of the nth digit of π without calculating the first $n-1$ digits.

5.7 Scales of Time, Space, and Matter

Any mathematization of a natural phenomenon is based on the measurement of quantities and on the evaluation of their orders, that is, the powers, with respect to a given base, that better approximate their values. Table 5.23 provides the decimal scale of powers with their corresponding symbols and prefixes (it is interesting to realize that the tenth power of 2 is 1024 which is close to the third power of 10).

Let us evaluate some measure orders. *What is the magnitude order of a solar year?* A year is consists of 365 days of 24 hours of 3600 seconds. Therefore, one year is $365 \times 24 \times 3,6 \times 10^3 < 3,2 \times 10^7$ seconds, that is only a few tens of millions of seconds.

What is the magnitude order of universe life? Assuming the universe started 15 billions of years ago, the number of seconds from the so-called "big-bang" are less than 5×10^{17}.

How many corn grains can be put on a chessboard where its 64 squares are numbered and where one grain is put on the first square, two on the second one, and in general the double of the number of grains placed in a square is placed on the next square (chessboard puzzle)? It is easy to evaluate this number as $2^{64} - 1$. If we approximate 2^{10} with 10^3, and 2^4 with 15, then the overall evaluation gives $2^{64} \approx (2^{10})^6 \times 15 = 1.5 \times 10^{19}$. This means that if a grain is one gram, we need about 15000 billions of tons of corn to cover the chessboard according to this procedure. You can never find such a corn quantity on Earth.

Table 5.23 Decimal orders

Prefix	Symbol	Power
yotta	Y	10^{24}
zetta	Z	10^{21}
exa	E	10^{18}
peta	P	10^{15}
tera	T	10^{12}
giga	G	10^{9}
mega	M	10^{6}
kilo	K	10^{3}
hecto	h	10^{2}
deka	da	10^{1}
deci	d	10^{-1}
centi	c	10^{-2}
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}
femto	f	10^{-15}
atto	a	10^{-18}
zepto	z	10^{-21}
yocto	y	10^{-24}

Tables 5.24, 5.25, 5.26, 5.27, and 5.28 summarize some of the most important measures on which universe, life and evolution are based on. The web site `http://bionumbers.hms.harvard.edu` hosts a huge collection of numbers corresponding to measures of quantities of biological organisms. The numbers of regarding measurements that relate to bacterium Escherichia coli constitute a file of size near to one megabyte.

Usually, the macroscopic quantities of time, space, and mass are around three or four orders above second, meter, and gram, respectively. The microscopic magnitudes have the same range but six orders below these orders, while three orders below the microscopic world we have the nanoscale. The visual resolution of our eyes can distinguish at the level of one-tenth of a millimeter. The microscope can increase that by around 2000 times, while more powerful instruments can increase our sight by millions of times. Telescopes have visibility ranges around 10^{20}, and by means of radio telescopes we can receive signals from the extreme part of the universe. But structures of small molecules and atoms are placed at nano scales. Therefore, we can see cells, but we cannot see inside molecules and atoms, even if we can indirectly inspect their structure by using suitable physical phenomena.

Table 5.24 Matter aggregation levels (distances in meters)

Level	Distances	Aggregation Forces
Subatomic particles	10^{-15}	Subnuclear
Atoms	10^{-12}	Nuclear
Inorganic Molecules	10^{-10}	Atomic-Electrical
Organic Molecules	10^{-8}	Electrical-Chemical
Cells	10^{-6}	Electrical-Chemical-Mechanical
Microorganisms	10^{-5}	Electrical-Chemical-Mechanical
Organisms	10^{1}	Electrical-Chemical-Mechanical
Ecosystems	10^{5}	Electrical-Chemical-Mechanical
Planets	10^{7}	Gravitational
Stars	10^{14}	Gravitational
Constellations	10^{17}	Gravitational
Nebulosas	10^{21}	Gravitational
Galaxies	10^{24}	Gravitational
Universe	10^{27}	Gravitational

Time-Space, Matter-Energy, Life-Evolution, Mind-Consciousness are the great problems of science. They are always revisited to gain better understandings of their nature, but since Plato's allegory of the cave, presented in his *Republic*, our pictures of reality are partial, inadequate, and continuously changing. The scientist, like the philosopher, cannot directly see the reality because he cannot stand the truth dazzle. But the truth is under this dazzling light, everything else are only shadows, indirect and defective realities.

Table 5.25 Mass numbers in grams

Electron	10^{-28}
Hydrogen	$1.67 \cdot 10^{-24}$
Nucleotide	10^{-21}
Human genome	$6 \cdot 10^{-12}$
Cell	10^{-6}
Human brain	$1.5 \cdot 10^{3}$
Human body	$6 \cdot 10^{4}$
Earth	10^{27}
Sun	10^{33}

Table 5.26 Ages in years

Universe	$15 \cdot 10^9$
Earth	$4.5 \cdot 10^9$
Life	$3.8 \cdot 10^9$
Eukariotic Life	$1.8 \cdot 10^9$
Multicellular life	$6 \cdot 10^8$
Mammals	$2 \cdot 10^6$
Neanderthal Homo	10^5

Table 5.27 Physical and Biological numbers (space in meters, time in seconds)

Plank scale time	10^{-43}
Big-bang time interval	10^{-40}
Big-bang temperature	10^{28} C^o
Light speed	$3 \cdot 10^8$ m/s
Light Year	$3 \cdot 10^{23}$
Electromagnetic wave lengths	$10^{-6} - -10^{-10}$
Gravitational/Electromagnetic force ratio	10^{36}
One gram mass	$3 \cdot 10^{16}$ Joule
Atom types	10^2
Cell molecules	10^{10}
Human body cells	10^{14}
Human Brain neurons	10^{11}
Human Brain synapses	10^{14}
Human Brain elementary operation energy	10^{-16} Joule
Human genes	$3 \cdot 10^5$
Human genome	$3 \cdot 10^9$ base pairs
Mycoplasma G. genome	$5.8 \cdot 10^4$ bp
Escherichia C. genome	$4 \cdot 10^6$ bp
Gene lengths	$10^3 - -10^4$ bp
Human cell types	254
Human protein types	10^5
Human cell divisions per minute	$2 \cdot 10^7$
Human immunological space	10^8 shapes

Table 5.28 Big numbers

Avogadro number	$6.2 \cdot 10^{23}$
Universe Atoms	10^{78}
Universe Particles	10^{87}
Gogol	10^{100}
Chessboard Games	10^{120}
Gogolplex	$10^{10^{100}}$

5.8 Discrete Dynamical Systems

The mathematical formalizations of dynamical systems, in spite of their elementary character, reveal many subtle points related to the puzzling nature of time. Here we give a brief outline of some basic concepts (see [168, 181, 165, 182] for more advanced concepts).

Given a function $\delta : S \to S$ and $i \in \mathbb{N}$, the ith-iterates of δ are inductively defined by the following equations, for every $s \in S$:

$$\delta^0(s) = s,$$
$$\delta^{i+1}(s) = \delta(\delta^i(s)).$$

Definition 5.4. An **autonomous dynamical system** is a pair (S, δ) where the set of states S is also called the phase space and $\delta : S \to S$.

In an autonomous dynamical system, the dynamics δ is a *next-state-function* from the phase space S to itself. According to the definition above, an autonomous dynamical system with an initial state s_0 "represents" $\mathbb{N}$ if $\delta^n(s_0) \neq \delta^m(s_0)$ for every $n, m \in \mathbb{N}$, with $n \neq m$. In this perspective, the infinite enumeration process is based on the possibility of starting from s_0 and generating, with the *next-state-function*, a state that is new at every application of this function. If the set of states generated by the *next-state-function* is finite, the counting process cannot be developed beyond a certain bound, because surely the dynamics has a cycle (reaching an already generated state). In this case, it is possible to go beyond the bound by restarting the cycle after its end, and by counting, in some way, the number of times the cycle is iterated (this is the principle of calendars and chronologies, based on astronomical cycles).

In autonomous dynamical systems, time is a consequence of their internal dynamics. Instead, in the following definition of timed dynamical system, an external notion of time is assumed.

Definition 5.5. A **timed dynamical system** D is a triple $D = (S, T, \delta)$ given by a set S of states, called **phase space**, $T \subseteq \mathbb{R}$ is a set of **time instants**, and $\delta : T \to S$ is a **dynamics**, which assigns to any time instant the state which the system takes at that instant.

Any function $f : \mathbb{R} \to \mathbb{R}$ determines a dynamical system, in a trivial way. Motions, developments, growth processes, and morphogenesis are representable as dynamical systems. In planetary motions phase space is $\mathbb{R}^6$ because the position and velocity of the planet with respect to three cartesian axes identify the state of the system, and the dynamics is the motion law giving the planet positions and velocities at every instant (a system of two planets needs a phase state included in $\mathbb{R}^{12}$). A dynamical system $D = (S, T, \delta)$ is **discrete** if T is a subset of $\mathbb{Z}$.

Proposition 5.6. *An* **autonomous** *dynamical system* (S, δ) *determines a family of timed dynamical systems* $D_s = \{(S, \mathbb{N}, \delta_s) \mid s \in S\}$, *where dynamics* δ_s *is called the* **timing** *of* δ *based on the* **initial state** $s \in S$, *and it is defined by* $\delta_s(i) = \delta^i(s)$.

The following definition of *flow* generalizes the concept of autonomous system, by using a next-state-function equipped with a real number as further parameter.

Definition 5.7. A **flow** D is a triple $D = (S, T, \delta)$ given by i) a set S of states, called **phase space**, ii) a set $T \subseteq \mathbb{R}$ of **time intervals** such that $0 \in T$ and when $t, t' \in T$, then also $t + t' \in T$, and iii) a **dynamics** δ, which is a function $\delta : S \times T \to S$ providing the state $\delta(s,t)$ which the system assumes after time t, starting from s. It is required that $\delta(s,0) = s$, and for any $s \in S$ and $t, t' \in T$ we have $\delta(\delta(s,t),t') = \delta(s,t+t')$. When T is a subset of $\mathbb{Z}$, then D is a **discrete** flow.

A timed dynamical system (S, T, δ) is **totally discrete** when both sets S and T are finite or enumerable sets. Even though it sounds a little strange, a real number decimal representation can be seen as a totally discrete dynamical system, whenever we identify it with a sequence providing its decimal digits. In fact in this case $S = \{0, 1, 2, \ldots 9\}$ and $T = \mathbb{N}$. Infinite words, trees, or graphs are totally discrete dynamical systems on suitable discrete phase spaces, where the dynamics is the function giving the structure at some step of its generation. Examples of totally discrete dynamical systems are: deductive and rewriting systems, Chomsky grammars, automata, L systems, cellular automata (see Chap. 6).

Any computation is a kind of dynamics. However, the two concepts are based on different perspectives. In a computation an input is provided to the system (in an initial state) and an output is expected at the end of the process if the system reaches a state which is considered as final, according to some termination criterion. This means that the dynamics underlying the move of a computation from an initial to a final state is functional to the desired relation between input and output, and the sooner a final state is obtained, the better the computation is realized. On the contrary, in a dynamical system the focus of the process is on the dynamics itself. Input and output can be even disposed of, but when they are present, they are used in order to guarantee that dynamics could proceed by satisfying some behavioral requirements (e.g., acquiring or expelling substances). A major example of this dynamical perspective are living organisms. They are open systems, that is, they need inputs and outputs, to ensure that a dynamics, with specific features, can be maintained in time, as long as possible. In this perspective, it is very important to individuate dynamical properties which are relevant in life phenomena, and to discover principles according to which some artificial dynamics exhibits behaviors with these properties.

The definition of autonomous systems implicitly assumes that the application of δ can be considered uniform, that is, the same temporal interval is assumed for any application of δ. In the late 16th century, Galileo discovered the pendulum, as a physical phenomenon for which this uniformity feature may be assumed. This discovery is the beginning of the experimental science, which is intrinsically based on the measure of time.

Does time come before any dynamics, or vice versa is time an abstraction originated by dynamical phenomena? In fact, time apart from dynamics seems to be a metaphysical abstraction. From an epistemological point of view, we only have clocks, and any physical notion of time is always related to a **clock**, that is, a system

(S,δ,s_0) where (S,δ) is an autonomous dynamical system, and $s_0 \in S$ is an *initial state*, conventionally associated to time 0. With a clock, it is intrinsically assumed that the application of its next-state-function can be considered as a uniform process, in such a way that the transition from a state to the next one happens according to the same law, with no appreciable difference in the quantity of time (whatever this could mean) necessary to perform that transition. In other words, clocks are physical entities where we can observe *regular* events of transition from a state to another state. If k is a bound such that no physical clock is available at a level inferior to 10^{-k} seconds (the time measurement unit), what reason can be used to assume the existence of time at smaller temporal scales?

One of the most important concepts in dynamical systems is that of **attractor**. There are many ways to formalize this notion. Intuitively, an attractor is a quasi state (a set of states) such that when a trajectory reaches it (a state in it), then this trajectory remains inside it. Moreover, this quasi state is included in a bigger quasi state, called its **basin**, such that any trajectory running through the basin, after a while (or according to a limit process), tends to "fall" into the attractor. A basin is a case of **dynamically invariant** set B, that is, in a dynamical system (S,δ), if $x \in B$, then $\delta(x) \in B$. An attractor of basin B is a sort of minimal or limit subset of B where dynamics could remains eternally confined. This means that attractors can be viewed as "dynamical states", or even, as *second level states*. For example, a living organism in a stable situation, performing life functions, moves along a trajectory, which macroscopically seems just a state, but corresponds essentially to an attractor. In mammals there are around 200 cell types. The stable states of these cells can be seen as different attractors where cell dynamics may fall, in order to satisfy some specific conditions corresponding to their biological role.

Example 5.8. The Collatz dynamical system [191] is the autonomous system on positive natural numbers where dynamics is given, for any $x \in \mathbb{N}$, by $3x+1$ if x is odd, and $x/2$ if x is even. . The Collatz conjecture claims that $\{1,2,4\}$ is the attractor of this system (starting from any number you fall into the $1-4-2-1$ loop, and then you remain there). It has been proved to be true for numbers until around 2^{60} (a web computational project on this topic is actively testing the Collatz conjecture, see `http://www.ericr.nl/wondrous/`).

A simple notion of attractor can be defined in the following way.

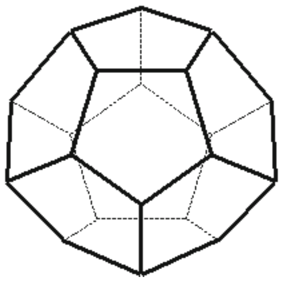
Definition 5.9. Given a dynamical system of dynamics δ, an attractor A of basin B, with $A \subseteq B$, is a set of states such that, for any $x \in A$, $\delta(x) \in A$, and for any $x \in B$ there exist a natural number n such that $\delta^n(x) \in A$.

An attractor A of basin B is called **minimal** if $A \subseteq C$ for any attractor C of basin B. Usually, attractors are intended as minimal attractors. It can be easily proved that a minimal attractor is consisting of only one state or of a cycle.

Time measurement is present as soon as the first human civilizations are developed. In any organized society, especially when it reaches a certain level of complexity, humans need to dominate the flow of time, in order to plan, project and organize events and activities. The common experience of the day/night alternation, the seasons, and the natural cycles are the primitive points of reference in the flow of time. The discovery of astronomical regularities represents a more sophisticated way of dominating time. Stars give the spatial orientation and the temporal rhythms to human activities. But, in a sense the time of stars, in the primitive societies on the earth, depends on the sky, and a reliable and precise synchronization of events is possible only at local level.

Clocks were available even before Galileo's discovery. In the middle ages, the improvement of some technical devices is the basis of clocks, usually put in public buildings, which have a precision acceptable for everyday life. However, the precision necessary for scientific measurements was possible after Galileo's discovery of the *isochrony* of small oscillations. Galileo observed that the oscillation period of a pendulum is constant, regardless of the angle of the swing, if it is around 3 degrees. On the basis of this phenomenon, further important technical improvements allowed the construction of reliable clocks (for example, Christiaan Huygens' and Robert Hooke's *balance spring*).

This was the passage from an astronomical time to a physical time, and was the basis of modern physics where motion laws are expressed by equations relating space measurements with the measurements of times necessary to cover them. Physical measurable time was also the basis for dominating the flow of time at nonlocal ranges, for example, for the needs of reliable and precise methods of sea navigation.


Platonic Dodecahedron

6 Languages and Grammars

Abstract. Strings are sequences on which some specific operations are defined. The most important operation over strings is the **concatenation**. If α and β are two strings, their concatenation, usually denoted by $\alpha\beta$, is the sequence where the last element of α is followed by the elements of β, in the order they have in β. **Formal languages** are sets of strings. In this chapter we present some basic concepts of formal language theory: languages, grammars, automata, expressions, and patterns. We conclude by outlining the notions of decidability and undecidability, and with two famous kinds of computation automata: Turing machines and register machines.

6.1 Strings and Languages

Formal language theory (FLT) started at the beginning of the 20th century with some seminal papers by Thue [195], where strings are considered as mathematical objects defining the algebraic structure of **monoid**, based on the binary operation of concatenation (juxtaposition of strings) that is associative. When the theory of computation (and computability) started and symbolic processes were investigated in mathematical terms [214, 208, 209, 196, 197, 203], then formal languages, as sets of strings, and classes of formal languages proved to be crucial for a wide class of concepts that became of fundamental importance for computer science [201, 199, 206, 210, 211, 213].

Given an alphabet A, we denote by A^* the set of finite sequences of symbols of A equipped by the operation of **concatenation** such that, if:

$$\alpha = \alpha_1, \alpha_2, \ldots, \alpha_n$$

$$\beta = \beta_1, \beta_2, \ldots, \beta_m$$

DOI: 10.1007/978-3-642-36223-1_6

then their concatenation $\alpha \cdot \beta$ is defined as:

$$\alpha \cdot \beta = \alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_m.$$

This means that $\alpha \cdot \beta$ is constituted by the symbols of α in the order they have in α followed by the symbols of β (in the order they have in β). In this case, sequences are more properly called **strings** (over the alphabet A) and very often $\alpha \cdot \beta$ is abbreviated as $\alpha\beta$, by omitting the symbol of concatenation (juxtaposition notation). Moreover, very often, comma between symbols is omitted. Strings will be usually denoted by Greek letters. We recall that when we say symbols of a string, we mean symbol occurrences, because the same symbol may occur many times. The length of a string α, denoted by $|\alpha|$, is the number of symbols occurring in α (the sum of their multiplicities).

The concatenation of strings is an associative operation:

$$\alpha(\beta\gamma) = (\alpha\beta)\gamma.$$

It is also useful to introduce a special string, called the **empty string** and denoted by λ with no symbol occurring in it, such that:

$$\alpha\lambda = \lambda\alpha = \alpha.$$

If $n \in \mathbb{N}$, the **iteration** α^n of a string α is defined by concatenating n copies of the string α, by assuming that $\alpha^0 = \lambda$. If for some (possibly empty) strings β, γ, δ it holds that:

$$\alpha = \beta\gamma\delta$$

then β, γ, and δ are **substrings** of α and in particular β is a **prefix** of α, δ is a **suffix** of α, and γ is an **infix** of α.

Notation $\gamma \subseteq \alpha$ means that γ is a substring of α, and $\gamma \subset \alpha$ means that γ is a substring of α which is different from α.

The reverse of a string $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ is the string where symbols occur in the reverse order, with respect to α:

$$rev(\alpha) = \alpha_n\alpha_{n-1} \ldots \alpha_1.$$

Given a string α, then, if $i \leq j \leq |\alpha|$, then $\alpha[i, j]$ denotes the **substring selection** of α, which is the string constituted by the symbols occurring in α from i to j, in the same order they occur in α. We denote by $\alpha(i) = \alpha[i, i]$ the symbol of α occurring in its position i.

Many kinds of operations of **insertion**, **deletion**, and **permutation** can be defined in terms of concatenation, length, and substring selection.

Given a string α, its **left derivation** ∂_α is an operation such that:

$$\partial_\alpha(\alpha\beta) = \beta$$

and $\partial_\alpha(\beta) = \lambda$ if α is not a prefix of β (an analogous definition can be given for the **right derivation**, by replacing prefixes by suffixes).

A (formal) **language** L over an alphabet A is a set of strings over this alphabet, that is, $L \in \mathbb{P}(A^*)$. Of course, the alphabet A is a language over A, A^* is also a language over A, and any finite (or empty) subset of A^* is a language over A.

For example, all the strings over an alphabet A of length equal to or smaller than n define a (finite) language over the alphabet A:

$$\{\alpha \in A^* \mid |\alpha| \leq n\}$$

The following are languages over an alphabet A:

$$\{\alpha\alpha \mid \alpha \in A^*\}$$

$$\{\alpha \in A^* \mid \alpha \neq \beta^n \text{ for any } \beta \in A^*, n \in \mathbb{N}\}$$

Any string $\alpha \in A^*$ provides the following languages:

$$Pref(\alpha) = \{\beta \mid \beta\gamma = \alpha, \gamma \in A^*\}$$

$$Suf(\alpha) = \{\beta \mid \gamma\beta = \alpha, \gamma \in A^*\}$$

$$Sub(\alpha) = \{\beta \mid \beta \subseteq \alpha\}$$

If ω is an operation defined on strings over an alphabet A, then we can extend it to any language L over A by putting:

$$\omega(L) = \{\omega(\alpha) \mid \alpha \in L\}.$$

The definition of languages as particular sets, allows us to apply to them all the usual boolean operation on set (union, intersection, and difference). However, the catenative nature of strings provides other natural operations. Table 6.1 reports the main operations on languages, where $L_1, L_2 \in \mathbb{P}(A^*)$. In the context of formal languages, for reasons of similarity with the notation that will be introduced, set-theoretic union will be also called *sum* and will be denoted by $+$.

Abbreviation αL stands for $\{\alpha\} \cdot L$ and $\alpha + L$ stands for $\{\alpha\} + L$.

A **language expression** E is constructed by applying language operations on some initially given languages. A **grammar** G for a language L is an algorithm

Table 6.1 Operations on languages

$L_1 + L_2 = \{\alpha \mid \alpha \in L_1, \text{ or } \alpha \in L_2\}$	**language sum**
$L_1 - L_2 = \{\alpha \mid \alpha \in L_1, \alpha \notin L_2\}$	**language difference**
$L_1 \cap L_2 = \{\alpha \mid \alpha \in L_1, \alpha \in L_2\}$	**language intersection**
$L_1 \cdot L_2 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$	**language concatenatiion**
$L^n = \{\alpha_1\alpha_2\ldots\alpha_n \mid \alpha_i \in L, \text{ for } i = 1, 2, \ldots, n\}$	**language iteration** ($n \geq 1$)
$L^* = \{\alpha_1\alpha_2\ldots\alpha_n \mid \alpha_i \in L, \text{ for } i = 1, 2, \ldots, n, \text{and } n \in \mathbb{N}\}$	**Kleene star**

which generates all and only the strings of L. An **automaton** M for L is an algorithm which recognizes all and only the strings of L. A string **pattern** P is a linear structure (represented by an algebraic expression, as we see later on) which is common to all the strings of L.

We will denote by $\mathbb{L}(E), \mathbb{L}(G), \mathbb{L}(M), \mathbb{L}(P)$ the language specified by the grammar G, by the automaton M, and by the pattern P respectively. A typical problem of formal language theory is the passage from one method of specification of a given language, to another method which specifies the same language.

The intuition behind a grammar is that of a set of rules for writing words (syntactic forms). An automation can be seen as a device which, when it takes a symbolic form as an input, it answers yes/not in correspondence of a recognized/unrecognized structure. The term *pattern* comes from Latin *paternitas* and expresses an origin which is common to all the strings with a specific *imprinting*. These strings are said to be **instances** of the pattern. In formal terms, a pattern (of strings) can be defined as an expression built by means of string operations, from some strings, and variables ranging in some specific sets, and/or satisfying some conditions. When values are given to the variables of a pattern P, the corresponding values that P assume are the strings of $\mathbb{L}(P)$.

An analogy can be useful to clarify the notion of (string) pattern. In fact, a polynomial with positive integer coefficients and positive integer variables can be considered a number pattern. For example,

$$3x + 4y$$

is a pattern such that number 10 is an instance of it, in correspondence to the values $x = 2, y = 1$, but 5 is not an instance of this pattern because the equation $3x + 4y = 5$ does not hold for any pair of positive integer values of x, y. Analogously, the string pattern axx with $x \in A^*$ has the string abb as its instance, but the string aab cannot be an instance of it.

The following language is constructed by starting from languages $\{a\}$ and $\{b\}$ by Kleene star and language concatenation:

$$\{a\}^* \cdot \{b\}^*.$$

The same language is specified by the pattern $a^n b^m$ with $n, m \in \mathbb{N}$.

Some important patterns defining languages (over alphabets $\{a, b\}$ and $\{a, b, c\}$ respectively) are the following, where term *soma* (meaning body in Greek) refers to their interpretations, as growing structures (of one, two, or three components):

$$\textbf{Mono-somatic Pattern} = a^n \quad \text{with } n \in \mathbb{N}, n > 0$$

$$\textbf{Bi-somatic Pattern} = a^n b^n \quad \text{with } n \in \mathbb{N}, n > 0$$

$$\textbf{Tri-somatic Pattern} = a^n b^n c^n \quad \text{with } n \in \mathbb{N}, n > 0.$$

The first one expresses a simple homogeneous linear form. The second one expresses a symmetric linear form constituted by two parts having the same size, but of two different types. The third one has an analogous structure with three parts. This kind of form corresponds to an idealized form of development which is typical of many animal embryos (ectoderm, mesoderm, endoderm). In the next section, we will present some grammars generating the languages corresponding to these patterns.

6.2 Grammars and Chomsky Hierarchy

In Formal Language Theory a grammar is a formal device generating strings. In this sense, a (formal) grammar is similar to a logical calculus equipped with axioms from which it derives theorems. In fact, the first kinds of formal systems generating strings, introduced by Post [208, 209], were inspired from the deduction systems of formal logic. The following definition, due to Chomsky [196] is a particular case of Post system.

Definition 6.1. A (Chomsky) grammar G is a *generative* device. It is specified by four components:

$$G = (A, T, S, R)$$

- A finite **alphabet** A;
- A subset T of A of **terminal** symbols;
- an **initial symbol** S belonging to **nonterminal** symbols in $A - T$;
- a finite set R of **rules** $\alpha \to \beta$ with $\alpha, \beta \in A^*$ and $\alpha \neq \lambda$, that is, pairs of strings over A.

A grammar $G = (A, T, S, R)$ defines a language by means of two relations: a **direct rewriting** relation $\Rightarrow_G$, and a **transitive rewriting** relation $\Rightarrow_G^*$:

$$\varphi \Rightarrow_G \psi$$

iff

$\varphi = \gamma\alpha\delta$
$\psi = \gamma\beta\delta$
$\alpha \to \beta \in R.$

while

$$\varphi \Rightarrow_G^* \psi$$

iff for some sequence $\varphi_1, \varphi_2, \dots, \varphi_n$ of strings:

$$\varphi = \varphi_1 \Rightarrow_G \varphi_2 \Rightarrow_G \dots \Rightarrow_G \varphi_n = \psi.$$

The language $\mathbb{L}(G)$ generated by the grammar G is the set of strings of terminal symbols obtained by *transitive rewriting* from the initial symbol S of G.
The language defined by the **bi-somatic pattern** is generated by the grammar of Table 6.3 (S is the initial symbol, and terminal symbols are lower case letters).

Table 6.2 The language $\mathbb{L}(G)$ generated by the grammar G

$$\mathbb{L}(G) = \{\alpha \in T^* \mid S \Rightarrow_G^* \alpha\}$$

Table 6.3 A grammar for the bi-somatic language

$S \rightarrow ab$
$S \rightarrow aSb$

In fact, by applying the second rule a certain number n of times, n symbols a are generated as prefix of S followed by n symbols b as suffix. With the first rule S is replaced by ab, therefore instances of the bi-somatic pattern are always produced. The language defined by the **tri-somatic** pattern is generated by the grammar of Table 6.4.

Table 6.4 A grammar for the tri-somatic language

$S \rightarrow abc$
$S \rightarrow aSBc$
$cB \rightarrow Bc$
$bB \rightarrow bb$

In fact, by applying the first two rules we get a string with a number n of symbols a as a prefix, followed by a string abc and by the same number n of pairs Bc. The last two rules of the grammar move symbols B in the middle and symbols c at the end of the string. When a symbol B comes on the right of a symbol b it transforms into b, according to the last rule. In this way all the B which reach the middle of the string become b, but the number of B is equal to the number of a and of c. Therefore, instances of the tri-somatic pattern are always produced by the grammar of Table 6.4. Moreover, it is easy to verify that, for whatever is the order of application of the last two rules, the same terminal string is generated.

The mono-somatic language defined by the pattern a^n $(n > 0)$ is generated by the grammar of Table 6.5.

Table 6.5 A grammar for the mono-somatic language

$S \rightarrow aS$
$S \rightarrow a$

Table 6.6 A grammar for the bipartite language

$S \rightarrow aS$
$S \rightarrow Sb$
$S \rightarrow a$
$S \rightarrow b$

Table 6.7 Another grammar for the tri-somatic language

$S \rightarrow abc$
$S \rightarrow S_a abc$
$S_a \rightarrow aS_b$
$S_b a \rightarrow aS_b$
$S_b b \rightarrow bbS_c$
$S_c b \rightarrow bS_c$
$S_c c \rightarrow S'cc$
$S_c c \rightarrow cc$
$bS' \rightarrow S'b$
$aS' \rightarrow aaS_b$

The language defined by the **bipartite pattern** $a^n b^m$ $(n, m \in \mathbb{N}, n+m \neq 0)$ is generated by the grammar of Table 6.6.

Another grammar generating the tri-somatic language is given in Table 6.7. The idea behind its construction is that a special symbol "writes" a new terminal symbol a and transforms in a symbol moving on the right and searching for a symbol b. When a b is reached it writes a new symbol b and transforms in a symbol searching (on the right) for c. When c is reached a new symbol c is written and either rewriting concludes, or a search (on the left) for a begins with another analogous rewriting cycle. In this way, the synchronized rewriting of the three symbol, in the right positions, is performed which guarantees the generation of tri-somatic forms. The same strategy can be adapted for generating a lot of interesting linear patterns. For example, the pattern of duplicated strings $\mathbb{L}(\alpha)$ can be generated with a similar kind of strategy. In this kind of generative mechanism two possibilities are essential: i) moving a symbol along different parts of a string, which are possibly far and at unbounded distances, and ii) transform a non-terminal symbol according to the symbols occurring around it. These kinds of mechanisms can be realized only with rules which rewrite more than one symbol.

The grammars given above correspond to different models of development of linear structures. For example, in the bipartite pattern the only requirement is the distinction of two parts an a-part on the left and a b-part on the right. In the case of bi-somatic pattern, the two distinct parts are required to have the same size. This means that a-part and b-part have to grow in a synchronized way. Finally, the tri-somatic pattern extends, from two to three, the requirement of equal size and distinct parts. This implies that the structure growth needs to use long distance relationships or symbol transport mechanisms which guarantee the required symmetry.

The rules of a grammar can be classified according to four basic types individuated by Chomsky and given in Table 6.8.

Table 6.8 Types of grammatical rules

Rule Types	$G = (A, T, S, R)$	$\alpha \to \beta \in R$
Type 0	$\alpha \in A^* - T^*$	General
Type 1	Type 0 + condition: $\lvert\alpha\rvert \leq \lvert\beta\rvert$	Monotone (Context-sensitive)
Type 2	Type 1 + condition: $\alpha \in A - T$	Context-free
Type 3	Type 2 + condition: $\beta \in T + T(A - T)$	Right-linear

According to the previous classification, types are inclusive, in the sense that a rule of type $i+1$ satisfies the conditions of type i, but the *proper type* of a rule is the maximum type which we can assign to it. A grammar G is of type i with $i \in \{0, 1, 2, 3\}$ if i is the minimum of the proper types which can be assigned to the rules of G. A language is of type i if it can be generated by a grammar of type i. It is *properly* of type i ($i < 3$) if it is generated by a grammar of type i and cannot be generated by a grammar of type $i+1$.

Let us indicate by $\mathscr{L}_i$ the class of languages generated by grammars of type i.

A fundamental result about these grammatical types is the following strict inclusion of these classes of languages, constituting the well known **Chomsky Hierarchy**.

$$\mathscr{L}_3 \subset \mathscr{L}_2 \subset \mathscr{L}_1 \subset \mathscr{L}_0$$

In fact, all the finite languages are of type 3, but there are infinite languages of type 3, for example, the language $\mathbb{L}(a^n)$ ($n > 0$) is of type 3, but is not finite. Moreover it can be proved that: i) the bi-somatic language is properly of type 2; ii) the tri-somatic language is properly of type 1, and a class of languages there exists (the class *REC* will mention below), which is properly included in $\mathscr{L}_0$, and which properly includes $\mathscr{L}_1$.

Let *RE* be the class of **recursively enumerable languages**, for which an algorithm exists which enumerate all its strings and only them. An important result about Chomsky grammars states that:

$$\mathscr{L}_0 = RE.$$

A classical result in computation theory, which was one of the most important achievements of Turing analysis in his fundamental paper of 1936 (see Sect. 6.5), establishes that, if *REC* are the **recursive languages**, for which the membership of a string can be algorithmically decided (in finite number of computation steps), then:

$$REC \subset RE.$$

The inclusion of the class $CS = \mathscr{L}_1$ in the class RE is a consequence of the monotony condition for the grammars of type 1. Moreover, the existence of a language of REC which does not belong to CS can be obtained by a kind of reasoning going back to Cantor and Russell (*the diagonal argument*).

It can be shown that any rule of type 1 is equivalent, from the generative point of view, to a set of rules having the following **context-sensitive** form, where X is a non-terminal symbol and α, β, γ are strings of the alphabet A with $\gamma \neq \lambda$:

$$\alpha X \beta \to \alpha \gamma \beta.$$

Languages generated by grammars with context-sensitive rules are also called **context-sensitive** and their class is denoted by CS. Therefore, $CS = \mathscr{L}_1$. Rules of type 2 are called **context-free** and languages generated by grammars of type 2 are also denoted by CF. Therefore, $CF = \mathscr{L}_2$.

In the next section we show that $\mathscr{L}_3$ coincides with the class REG of *regular languages* obtainable from finite languages by applying the language operations of concatenation, sum, and Kleene star.

If X, Y are non-terminal symbols and a is terminal symbol, then $X \to aY$ is a **right-linear rule**, while $X \to Ya$ is a **left-linear rule**, and $X \to a$ is at same time right-linear and left-linear.

Grammars with both types of rules right-linear and left-linear generate the class LIN of linear languages which is strictly included in CF and which strictly includes REG.

In conclusion, Chomsky hierarchy can be reformulated and extended in the following way:

$$FIN \subset REG \subset LIN \subset CF \subset CS \subset REC \subset RE.$$

Two important aspects of grammatical generation which are interrelated are crucial: i) rules are applied without no strategy, ii) terminal symbols are the only mechanism for discriminating, among the strings generated from S, those which belong to the language $\mathbb{L}(G)$. It can be shown that if we do not use terminal symbols, then the generation power of grammars is reduced. In this way, the terminal/non-terminal distinction plays the role of integrating the rules toward the realization of forms which are specific of a given language. In other words, only string generations where rules are applied in a "correct" way are able to "terminalize". The other generations can be viewed as unsuccessful trials in a specific process of string generation.

Natural languages, programming languages, and strings representing linear forms that occur in natural processes require grammars having a complexity between context-free and context-sensitive grammars.

The theory of formal languages studies four main phenomena concerning formal languages:

1. **equivalence** results, showing that different methods and formal devices define the same languages. An example of such a kind of results is given in the next section where an equivalence will be given which implies an equivalence between regular grammars and the class of finite state automata;
2. **normalization** results, showing that some formal devices defining languages can be put in some specific formats which allow to simplify the analysis of specific aspects. For example, any grammar with monotone rules generates the same language of a grammar where rules have the **context sensitive form**.
3. **decidability** results, showing that some class of problems about languages can be solved in an algorithmic way, by reaching the solution in a finite number of steps. For example, given a grammar *CF* it is possible to decide if it generates a finite or infinite language;
4. **closure** results showing that some classes of languages are closed with respect to some language operations. For example, the sum of two *CF* languages is a *CF* language too, but *CF* is not closed with respect to complementation (with respect to the set of strings over its terminal alphabet), while *REG* and *CS* are closed with respect to all boolean operations (sum, intersection, and complementation).

6.3 Regular Expressions and Finite Automata

Consider singleton languages, that is, languages constituted by only one string. For example $\{a\}, \{b\}, \{c\}$. Then, let us construct languages by applying to them the language operation of sum, concatenation, and Kleene star, with an implicit priority of star with respect to concatenation, and of concatenation, with respect to sum. For example:

$$\{a\}^* + \{b\}\{c\}^*.$$

Languages built in this manner are called **regular languages**. If we adopt the abbreviation of eliminating set parentheses, by assuming that strings specify singleton languages, then we get **regular expressions** which naturally identify regular languages. For example, the language above is simply denoted by the following regular expression (parentheses can be introduced for a better reading of expressions):

$$a^* + b\,c^*.$$

Regular expressions are related to finite state automata.

A finite state automaton M is a device equipped with: i) an *alphabet*, ii) a finite set of *internal states*, one of which is the *initial state*, and some of which are the *final states* (see Fig. 6.1), iii) plus a set of *transition rules* which determine the following behavior. When a string α of an alphabet of the automaton is put on its tape, then M reads it by starting from the first symbol on the left of α and in its initial state. After reading a symbol, in dependence of its transition rules, M changes its state and moves to read the next symbol on the right. When the whole string α was

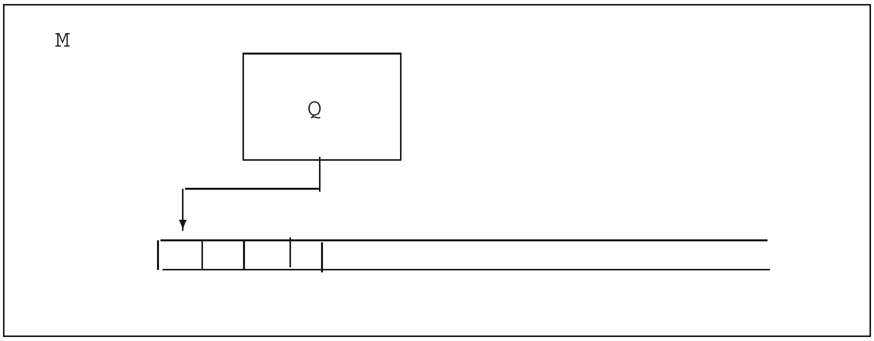


Fig. 6.1 A finite state automaton at beginning of its functioning

read, M "accepts" α, if the state assumed by M belongs to the set of its *final* states. Otherwise, M does not accept α. We denote by $\mathbb{L}(M)$ the **language recognized** by M, that is, the set of strings which are accepted by the automaton M.

Figure 6.2 describes, by means of a graph, an automaton which recognizes the bipartite language $\mathbb{L}(a^n b^m)$. Initial and final states are indicated by entering and exiting arrows respectively that are not linked with other states.

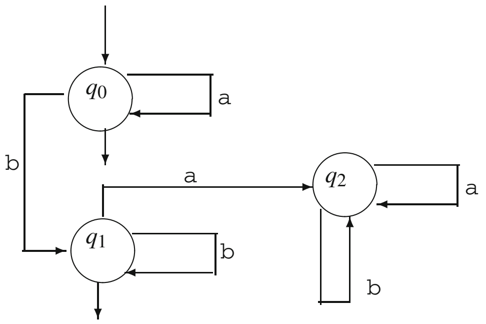


Fig. 6.2 A finite state automaton recognizing the bipartite language

The formal definition of finite state automaton follows.

Definition 6.2. A (deterministic) finite state automaton M is given by the following structure:

$$M = (A, Q, \tau, F, q_0)$$

where A is the *alphabet*, Q the set of *states*, F the set of *final states*, q_0 is the *initial state*, and τ is the *state transition* function, from pairs of symbols and states to states:

$$\tau : A \times Q \to Q.$$

M reads an input string α, starting from the first symbol (from the left) of α and in the state q_0 as current state. The automaton M applies the transition function τ to the symbol x that it reads and to its current state q, by taking $\tau(x,q)$ as next current state, and by passing to read the symbol on the right of the symbol x previously read. When the whole input string is read, then M halts. If the state of M, when it halts, belongs to F, then α is accepted by M, otherwise α is not accepted by M.

We can express the transition function of an automaton by a rewriting relation. For example the transition of the automaton given in Fig. 6.2 could be given by (q_0 initial state and q_0, q_1 final states):

Table 6.9 The automaton recognizing the bipartite language expressed by state transition rules

$q_0 a \to q_0$
$q_0 b \to q_1$
$q_1 b \to q_1$
$q_1 a \to q_2$
$q_2 a \to q_2$
$q_2 b \to q_2$

In terms of rewriting relation, the language recognized by a finite automaton M can be defined in the following way:

$$\mathbb{L}(M) = \{\alpha \in A^* \mid q_0\alpha \to_M^* q_f, q_f \in F\}.$$

The transition rules representation of a finite state automaton can be easily transformed into a grammar if we start from the initial state q_0 and in passing from a state to another one we generate a symbol instead of reading it. For example, in Table 6.10, the rules of automaton of Table 6.9 are transformed into generative rules R which provide the type 3 grammar $G = (\{a,b,q_0,q_1,q_2\},\{a,b\},q_0,R)$ generating the bi-partite language. In this case, the initial state of the automaton becomes the start symbol of the grammar G, and rules $q_0 \to a$, $q_0 \to b$, and $q_1 \to b$ are added for deleting the final states q_0, q_1. Transition rules of the automaton which are relative to state q_2 do not need to be translated into generative rules. This method can be easily generalized, by proving that any language recognized by a finite automaton is also generated by a type 3 grammar. In an analogous way it is simple to define the inverse process of constructing a finite automaton which recognizes the language generated by a type 3 grammar. In conclusion, the two kinds of formal devices determine the same class of languages.

Table 6.10 A type 3 grammar generating the bi-partite language

$q_0 \to aq_0$
$q_0 \to bq_1$
$q_1 \to bq_1$
$q_0 \to a$
$q_0 \to b$
$q_1 \to b$

A finite state automaton M is **deterministic** when, for every state, only one possible transition can be applied. This means that the codomain of τ is Q:

$$\tau : A \times Q \to Q.$$

A deterministic automaton M recognizes a string α (over its alphabet) when starting in the initial state and reading α on its tape (from the first symbol on the left to the last on the right), by applying the transition function τ, M ends in a final state.

A finite state automaton M is **nondeterministic** when, for every state, one among a set of possible transitions can be applied. This means that the codomain of τ is $\mathbb{P}(Q)$:

$$\tau : A \times Q \to \mathbb{P}(Q).$$

A nondeterministic automaton M recognizes a string α (over its alphabet) when starting in the initial state and reading α on its tape, if M chooses at each step one of the possible transitions, for some sequence of choices, M ends in a final state.

Proposition 6.3. *The language bi-somatic $\mathbb{L}(a^n b^n)$ cannot be recognized by any finite state deterministic automaton.*

Proof. Any finite state automaton M determines an equivalence relation $\equiv_M$ between the strings over the alphabet of M such that $\alpha \equiv_M \beta$ if M finishes reading both the strings in the same state. Now, let $\{a,b\}$ the alphabet of M, then having M a finite number of states, there exist two distinct natural numbers n,m such that $a^n \equiv_M a^m$. If we concatenate both a^n and a^m by the same string b^n, then we get $a^n b^n$ and $a^m b^n$. Assume that M could recognize $\mathbb{L}(a^n b^n)$, then, when M reads a^n and a^m, it ends in the same state, according to the equivalence $\equiv_M$, therefore, it ends in the same state also when it reads both strings $a^n b^n$ and $a^m b^n$. But this is a contradiction because we assumed that M is able to recognize $\mathbb{L}(a^n b^n)$. □

For the sake of brevity, usually we will avoid to provide all the transition rules of a finite automaton. In fact, we adopt the convention that when a transition is not explicitly given, it is assumed that the state specified by the missing transition is a special non-final state, say it a *failure* state, which does not belong the the final states, where the automaton remains for any other symbol read in that state.

In some cases it is useful to use λ-transition, that is, the automaton can change its state without reading any symbol. However, it can be shown that any automaton with λ-transition has an equivalent automaton without λ-transitions.

Let us consider the pattern *abxy* where $x \in \{a,b,c\}^*, y \in \{a,c\}$. The corresponding regular expression is

$$ab(a+b+c)^*(a+c)$$

Figure 6.4 gives two automata recognizing the language $\mathbb{L}(ab(a+b+c)^*(a+c))$. The automaton on the top is nondeterministic because reading the symbols a,c in the top rightmost state different edges can be followed, by reaching different states. On the bottom, an equivalent deterministic automaton is given. Usually, the nondeterministic automaton equivalent to a deterministic one has a simpler structure, but its functioning is more complex (all the possible behaviors have to be considered for recognizing a given string).

A result due to Kleene is expressed by the following proposition which we report without proof.

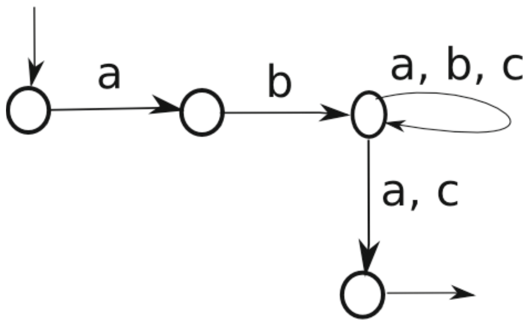
Proposition 6.4. *Any language which is recognized by a nondeterministic finite state automaton is also recognized by a suitable deterministic finite state automaton.*

Finite state automata were inspired by studies in modeling of neurological activity. The original paper of S. C. Kleene, in 1956, where these automata were introduced, was entitled **Representation of events in nerve nets and finite automata**, which was based of the fundamental paper of McCulloch and Pitts of 1943 (A logical calculus of the ideas immanent in nervous activity).

Another result, proved by Kleene in the paper mentioned above, shows the equivalence between FSA (Finite State Automata) and REG. In fact the following proposition holds.

Proposition 6.5. *$REG \equiv FSA$, that is, the class of languages generated by finite languages by applying, a finite number of times, the operations of concatenation, sum, and Kleene star, coincides with the class of languages recognized by finite state automata.*

Proof. Given a regular language L, a (nondeterministic) finite automaton recognizing it can be easily constructed. In fact, if a regular language L_1 is recognized by M_1 and a regular language L_2 is recognized by M_2, then putting together the graphs representing the transition rules of the automata M_1 and M_2, and connecting the final



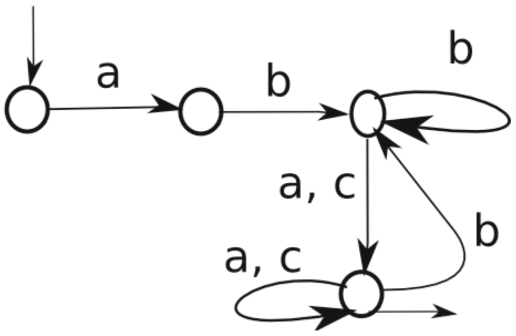


Fig. 6.3 A nondeterministic (top) automaton and a deterministic (bottom) automaton recognizing $\mathbb{L}(ab(a+b+c)^*(a+c))$

states if M_1 with the initial state of M_2, we get an automaton recognizing $L_1 \cdot L_2$. If the two initial states of M_1 and M_2 are "fused" in a unique initial states connecting the remaining parts of the two graphs, then we get an automaton recognizing the language $L_1 + L_2$. Finally, if a graph describes the transitions of an automaton M recognizing L, when in the graph of M, any transition edge entering in a final state, is replicated by a new transition edge which connects that final state with the initial of M, then a new automaton is obtained which recognizes the language corresponding to L^*. These three constructions define a general procedure for constructing an automaton equivalent to a given regular expression.

The more complex part of the proof concerns the identification of the regular language recognized by a given finite state automaton. We will show that it can be obtained by applying the operation of sum, iteration, and Kleene star by starting from finite languages. The proof we report is the original one of Kleene. It is based on the definition of the following language, relative to a given automaton $M = (A, Q, \tau, F, q_1)$ with $Q = \{q_1, q_2, \ldots, , q_n\}$

$$L_{i,j}{}^{n}$$

of the strings which produces the state passage of M from q_i to q_j possibly by passing through $q_1, q_2, \ldots, q_n$.

Of course, we have:

$$L_{i,j}{}^{0} \subseteq A$$

moreover, for all $i, j \in \mathbb{N}$, the following equation holds:

$$L_{i,j}{}^{k+1} = L_{i,j}{}^{k} + L_{i,k+1}{}^{k} \cdot (L_{k+1,k+1}{}^{k})^{*} \cdot L_{k+1,j}{}^{k}$$

The validity of this equation can be explained if we consider the following example of chain of state transitions in the automaton M, where dots between states denotes intermediate states among $\{q_1, q_2, \ldots, q_k\}$, and the state q_{k+1} occurs three times (clearly any number of times could be considered):

$$q_i \cdots\cdots\cdots q_{k+1} \cdots\cdots\cdots q_{k+1} \cdots\cdots\cdots q_{k+1} \cdots\cdots\cdots q_j.$$

Now, for passing from q_i to q_j, possibly by passing through q_{k+1}, there are two possibilities: or M does not pass through q_{k+1} (term $L_{i,j}{}^{k}$ in the equation above) or M goes from q_i to q_{k+1} (term $L_{i,k+1}{}^{k}$ in the equation above) by passing again from q_{k+1} to q_{k+1} (two times in the example above) through the other k states (term $(L_{k+1,k+1}{}^{k})^{*}$ in the equation above), and finally, from q_{k+1} to q_j through the other k states (term $L_{k+1,j}{}^{k}$ in the equation above). Therefore, if the final states are $F = q_m, q_{m+1}, \ldots q_n$, the language recognized by M is given by:

$$\mathbb{L}(M) = L_{1,m}{}^{n} + L_{1,m+1}{}^{n} + \ldots + L_{1,n}{}^{n}.$$

In conclusion, all the equations above show that the language recognized by M is obtained by applying regular operations (sum, concatenation, and Kleene star) starting from finite languages (the proof uses implicitly the principle of induction given in Sect. 5.8). □

Very often grammars and automata corresponding to a given regular expression can be found in a simple way. However, an interesting problem, which occurs in many contexts, is the search of "minimal" grammars or automata equivalent to some given (complex) regular expression (according to some specified notion of size of these devices). The automaton of Fig. 6.4 provides two equivalent automata recognizing the (language of) the first regular expression of Table 6.11.

Table 6.11 shows a list of regular expressions, while Table 6.12 shows the patterns (with $n, m > 0$) of languages of types 1 or 2. First four languages are of type 2, while the last three are of type 1. Classes of languages of interest in natural phenomena (forms of developments, complex communication languages, natural languages) belong usually to classes between the Chomsky types 1 and 2. All Chomsky classes

Table 6.11 Regular expressions

$a(b+c)^*$
$a^*(c+b)$
$(ab)^*$
$(ab)^* + (ac)^*$
$(a+ab)^*c$
$((a+ab)^*c+bb)^*$
$aabcbb$

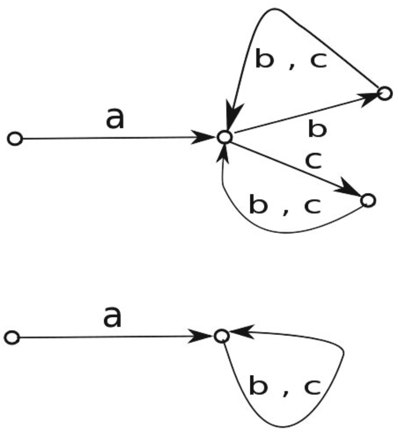


Fig. 6.4 Two equivalent automata recognizing $\mathbb{L}(a(b+c)^*)$

Table 6.12 Patterns of languages of types 1 or 2

$a^n b^{2n}$
$a^n b^{n+3}$
$(aba)^n$
$a^n b^n a^m$
$a^n b^n a^n$
$a^n b^n a^n b^n$
$a^n b^m a^{n+m}$

are closed with respect to the sum. In fact. given a grammar G_1 for a language L_1 and a grammar G_2 for a language L_2 it is easy to provide a grammar, generating $L_1 + L_2$, with the lowest type among those of G_1 and G_2. The class *REG* is boolean (closed with respect to sum, intersection, and complementation). An important result of late 1980s has proved that also *CS* is boolean. However, *CF* is not boolean because it is not closed with respect to intersection. In fact, $a^n b^n c^m$ and $a^n b^m c^m$ are patterns of CF languages, but their intersection, having pattern $a^n b^n c^n$, is not CF. Many important properties of languages can be algorithmically established. For example, a language is infinite iff any grammar generating it has an auto-rewriting symbol, that is, a non-terminal symbol X such that $\alpha \Rightarrow_G^* \beta$ where X occurs in both strings α, β, and such that $X \Rightarrow_G^* \gamma$ with $\gamma \in T^*$.

However, given two grammars G_1, G_2 of type 2, it can be shown that it is impossible to have an algorithm which, in general, is able to decide whether $\mathbb{L}(G_1) \subseteq \mathbb{L}(G_2)$. We mention other two important results about Chomsky hierarchy. The intersection of any language L with a regular language preserves the Chomsky type of L. Moreover any language L of type 0 can be obtained by a suitable CS language L' by deleting a special symbol from all the strings of L'.

6.4 Patterns and Rules

The notion of string pattern was already introduced in informal way. Here its formal definition follows.

Definition 6.6. A string pattern over an alphabet A is an algebraic expression P built by means of symbols (of the alphabet), variables (with specified ranges of variability), and operations. When variables of P are instantiated by values in the respective ranges, then P denotes a string over the alphabet A. If $P = P(x_1, x_2, \ldots, , x_k)$ is a pattern of variables $x_1, x_2, \ldots, , x_k$, and $S_1, S_2, \ldots, , S_k$ are the respective ranges of its variables, then the language $\mathbb{L}(P)$ defined by P is given by:

$$\mathbb{L}(P) = \{P(x_1 := \alpha_1, x_2 := \alpha_2, \ldots, , x_k := \alpha_k) \in A^* \mid \alpha_1 \in S_1, \alpha_2 \in S_2, \ldots, , \alpha_k \in S_k\}$$

where $P(x_1 := \alpha_1, x_2 := \alpha_2, \ldots, , x_k := \alpha_k)$ is the evaluation of the pattern when the values $\alpha_1, \alpha_2, \ldots, \alpha_k$ are assigned to the corresponding variables (and the operations are applied according to the structure of the expression P).

For example, let us consider the pattern:

$$x^n \, a \, y^m \, b \, x^{n+m}$$

with $x, y \in A^*$, $n, m \in \mathbb{N}$. Here we have strings a, b and variables x, y, n, m (with their ranges of variability). When variables are instantiated and concatenation, iteration, and arithmetical sum are applied, we get a string of A^*.
The elements of $\mathbb{L}(P)$ are also called *instances* of the pattern. In more complex cases, specific constraints can be required to a pattern (for example, the number of its variables). Patterns can be combined by using the language operations $+, \cdot, /, \cap$. In fact, if P_1 and P_2 are two patterns that identify languages, then, in a completely natural way, $P_1 + P_2$ means that:

$$\mathbb{L}(P_1 + P_2) = \mathbb{L}(P_1) + \mathbb{L}(P_2)$$

and analogously for concatenation, difference, intersection, and Kleene star.

Some regular expressions can be represented by patterns having only variables ranging on natural numbers. For example $a^* + b(c^*)$ can be represented by $a^n + b(c^m)$ with $n, m \in \mathbb{N}$ (different occurrences of Kleene star deserve different variables). Patterns built with strings, string operations, variables, and language operations provide a very powerful way of expressing forms of strings. It can be proved that suitable combinations of patterns with logical conditions on them provide an expressive power equivalent to the Chomsky grammars of type 0 [204].

The notion of pattern is useful for giving a general definition of **string rewriting rule**. The general idea of such a rule is that of a combinatorial rearrangement of pieces of some input strings for producing some output strings. This can be represented by a sequence of patterns, called **premises**, and a sequence of patterns called **conclusions**, such that, in the conclusions variables occur that appear also in the premises:

$$\frac{\textbf{Premises}}{\textbf{Conclusions}}.$$

According to this format, the grammatical rewriting relation has only one premise and one conclusion, that is, $\alpha \Rightarrow_G \beta$ can be expressed in the following way, where x, y are variable over A^* (A the alphabet of G):

$$\frac{x\alpha y}{x\beta y}.$$

This kind of rule is also called a **replacement**. In a string rewriting rule all the variables occurring in the conclusions have to occur also in the premises, and have to verify the same constraints.

For example, the following rule has two premises, one conclusion and two variables ($x, y \in A^*$):

$$\frac{xb, by}{bxyb}.$$

A string rewriting rule can be *applied* to a sequence of strings when these strings are instances of the premises, for suitable values of their variables. In this case, the application of the rule yields the evaluation of the conclusions, according to the same values of variables assigned to the variables of premises.

The rule given above can be applied to the strings:

$$ababab, bcc$$

according to the following values of premise variables:

$x = ababa$
$y = cc$

and provides as a conclusion the following instance:

$$bababaccb.$$

If R is a set of string rewriting rules, over some alphabet, and L a language over the same alphabet, then the language $R^*(L)$, generated by R from L, is defined as the minimum language including L and *closed* with respect to the rules of R. This means that, when $R^*(L)$ contains some strings, then it contains also the strings obtained from them by means of an application of a rule of R.

It can be shown that the language $R^*(L)$ can be generated by applying the rules of R to strings of L by means of **string derivations**. A string derivation is a sequence of steps, where in the initial step some strings of L are chosen, and in every following step strings are generated that are either strings of L, or strings obtained by an application of a rule of R to strings generated at preceding steps.

Figure 6.5 represents a string derivation by means of rules with two premises and two conclusions, by means of a suitable tree (with leaves on the top and the root on the bottom).

Consider the following rule ($x, y \in \{a,b,c\}^*$):

$$r = \frac{xaby}{xaabbyc}$$

it is easy to verify that:

$$\mathbb{L}(a^n b^n c^n) = \{r\}^*(\{abc\}).$$

In fact, when the rule is applied to a string of the tri-somatic language, then it provides a string which follows the same pattern. Therefore, starting from the shortest string of this type (we do not consider string λ), we are able to provide any possible

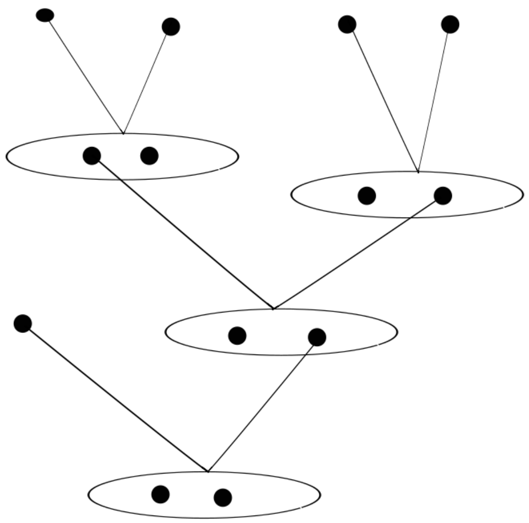


Fig. 6.5 A string derivation represented as a tree

string of the language. The above generation of $\mathbb{L}(a^n b^n c^n)$ shows as powerful are rewriting rules including variables.

In the case of replacement variables remain “fixed”, but when they are rearranged in the conclusions, then a global transformation is applied to the input strings in order to generate the output strings. This means that particular kinds of rewriting rules can be very powerful, and few rules can generate languages which require Chomsky grammars with greater numbers of rules. For example, consider a sort of multiple rewriting expressed by the rule:

$$a,\, b,\, c \rightarrow aa,\, bb,\, cc$$

that simultaneously rewrites any three occurrences of the three symbols by their duplicates, according to the following formulation by means of patterns ($x, y \in \{a, b, c\}^*$):

$$\frac{axbyc}{aaxbbycc}.$$

In this way, starting from the string *abc* the tri-somatic language is generated by means of string derivations which use only this multiple rewriting.

6.5 Turing Machine

From a very general point of view, a computation system is defined by seven components:

1. A set D of **data**;
2. a set S of **states**;
3. an **input function** (encoding data into states) $in : D \rightarrow S$;
4. an **output function** (decoding states into data) $out : S \rightarrow D$;
5. a set Ω of **operations** on the set S;
6. a set Φ of **rules** which are functions $\rho : S \rightarrow \mathbb{P}(\Omega)$;
7. a set of **computations** Γ which are sequences of states with an initial state encoding the initial data. For any state s a state s' follows s in a computation if $s' = \omega(s)$ for some $\omega \in \rho(s)$. A computation halts with a **terminal** state s_f when $\rho(s_f) = \emptyset$.

The previous structure describes a system where data are encoded by states and a dynamics is defined on the states that ends when the system reaches final states encoding results. In this sense, a computation is a special kind of dynamics where the passage from a state to another is performed by applying one operation among those which are applicable in a given state.

This general schema can be realized and extended in many ways. Namely, the rule of applying operations can be external to the system, and at each step an operator can intervene in the system for its evolution. Otherwise, a description of this rule, usually called **program**, can be internal to the system, by making it able to evolve in a specific way. In this case the system is a **programmed** computation system, also called **automaton**, which in its etymological sense means just “autonomously evolving system”. If the system can input a program, as part of its data, by behaving according to it, we say that the system is **universal** with respect to the class of programs it can input. If at each step the computation rule provides only one applicable operation, then the resulting computation is **deterministic** as opposed to the more general case of a **nondeterministic** computation. More general kinds of computations may assume the presence of **interactions**, that is, of exchanges of data during the computation. In this case, the behavior of a system depends on the sequence of data exchanged during these interactions. For example, in some state the system can require a value from outside. This means that the system stops its computation, by waiting for a value, and when it is provided (by means of an input device), the computation continues, possibly providing output values during the computation.

A fundamental model of computation was elaborated by Alan Mathison Turing in a famous paper published in 1936. It describes a general kind of computation processes based on a very intuitive basis. This kind of computation develops by transforming the contents of cells linearly arranged on a tape unbounded in the two directions (see Fig. 6.6). One symbol is contained in each cell which belongs to an alphabet A including a special symbol, say it B, called the blank symbol. A system can assume internal states of a finite set Q and in each state can read the symbol put in one cell (the current cell). A program, consisting of a finite sequence of **instructions**, defines the action that the system can perform at any step of its evolution. Any instruction has a pre-condition and a post-condition. A pre-condition is given by a pair (q,a). If the precondition of an instruction corresponds to the current situation of the system, that is, q is the internal state of the system and a is the symbol in the current cell, then the instruction can be applied according to its post-condition, which is a tern (b,q',m) where b is the symbol which replaces the symbol a in the current cell. The state q' is the state which replaces q and m indicates the current cell at the next step of computation. If $m = L$ the (new) current cell is located on the left of previous one, while if $m = R$ on the right.

Any computation starts with an initial state (usually indicated by q_0), with a sequence of symbols over the alphabet A arranged in a contiguous portion of the tape, and with the blank symbol B in all the cells outside this portion. At the beginning, the system reads on the first cell on the left where a symbol different from B is written (this is the initial current cell). When the machine stops, because no instructions are applicable, the output is the string put on the tape between the two most extreme symbols on the tape which are different from B.

The formal definition of A Turing machine follows.

Definition 6.7. A Turing machine M is given by the following structure:

$$M = (A, Q, q_0, I)$$

where A is the alphabet of its *symbols*, including the special *blank symbol* B, Q is the finite set of its *states*, $q_0 \in Q$ is its *initial state*, and I the set of its *instructions*, that is, quintuples (q,a,b,p,m) where $q,p \in Q$, $a,b \in A$ and $m \in \{L,R\}$. The set I of

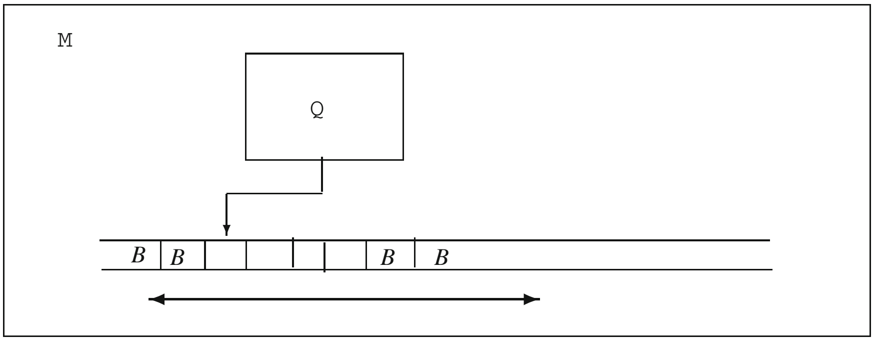


Fig. 6.6 The Turing machine

instructions defines the *computations* of M, as sequences of *configurations* generated in the following way.

A configuration is a string $\varphi q x \psi$ (representing the tape content, with an unbounded number of blank symbol occurrences implicitly assumed on the left of φ and on the right of ψ), where: $\varphi, \psi \in A^*$, $q \in Q$ is the *current state* of the configuration, and $x \in A$ is the *current symbol* of the configuration. In the configuration $\varphi q x \psi$ any instruction having q, x as initial symbols can be applied. M is *deterministic* when at most one instruction is applicable in every configuration, otherwise it is *nondeterministic*. The application of an instruction (q, x, y, p, R) in the configuration:

$$\varphi q x \psi$$

yields a next configuration:

$$\varphi y p \psi,$$

while an instruction (q, x, y, p, L) in the configuration:

$$\varphi z q x \psi,$$

where $z \in A$ is the symbol on the tape before the position of the current symbol x, yields a next configuration:

$$\varphi p z y \psi.$$

The machine M starts a computation in an *initial configuration* $q_0\alpha$, where $\alpha \in A^*$ is the *input string* of the computation.

M halts when no instruction is applicable. In this case, the configuration, say $\xi q x \chi$, is a *final configuration*, and the *output* of the corresponding computation of M is the string $\xi x \chi$ (where the symbol of the state is removed from the final configuration).

The Turing machine of Table 6.13 when starts with a string of $\{B, 1\}^*$ of type $1^{n+1}B1^{m+1}$ provides as result 1^{n+m+1} (1^{n+1} is a representation of number n).

Table 6.13 A Turing machine performing the sum of numbers in unary notation

$q_0, 1, 1, q_0, R$
$q_0, B, 1, q_1, R$
$q_1, 1, 1, q_1, R$
q_1, B, B, q_2, L
$q_2, 1, B, q_3, L$
$q_3, 1, B, q_4, L$

A Turing machine can be used as a computing device providing an input/output behavior, but it can be also used as a recognizing device when given an input it starts a computation and ends in some chosen states (a class o final states has to be defined). A Turing machine can be also viewed as generating the language of all the strings obtained as outputs in correspondence of all possible inputs strings.

We remark that, in principle, computing functions or recognizing and generating languages are equivalent tasks. In fact, a function $f : A \to B$ is univocally identified by its graphic, that is, the set $\{(x, f(x)) \in A \times B \mid x \in A\}$. Therefore, generating or recognizing the graphic of f is equivalent to computing f and vice versa.

It can be shown that the class of languages recognized by Turing machines is equal to the class of languages generated by Turing machines, and that the languages recognized/generated by nondeterministic Turing machines are equal to the class of languages recognized/generated by deterministic Turing machines. This class coincides with the class generated by Chomsky grammars of type 0.

Table 6.14 A non-terminating Turing machine on $\{0, 1\}$

$q_0, 1, 1, q_0, R$
$q_0, 0, 0, q_0, R$
q_0, B, B, q_0, R

The Turing machine given in Table 6.14 describes a computation that does not ends. The Turing machine given in Table 6.15 recognizes the bi-somatic language (stops in the state q_{yes} when the input is a bi-somatic string, while stops in the state q_{no} otherwise). It is easy to generalize the same strategy for recognizing the tri-somatic language.

We remark that what makes Turing machines more powerful than finite state automata is the possibility of movement in the two directions of the tape and the possibility of reading and writing on it.

Table 6.15 A Turing machine on $\{a,b\}$ recognizing the bi-somatic language

q_0,a,B,q_1,R
q_1,a,a,q_1,R
q_1,b,b,q_2,R
q_2,b,b,q_2,R
q_2,B,B,q_3,L
q_3,b,B,q_4,L
q_4,a,a,q_4,L
q_4,b,b,q_4,L
q_4,B,B,q_0,R
q_0,B,B,q_{yes},R
q_0,b,b,q_{no},R
q_2,a,a,q_{no},R

6.6 Decidability and Undecidability

The original Turing's paper of 1936 presents a basilar result of the theory of computation: the undecidability of the general halting problem for Turing machines. As a consequence of it, it is possible to deduce the existence of **recursive enumerable** sets which are not **decidable**. Namely, no Turing machine exists which can, in general, establish wether a given Turing machine stops yielding a result in correspondence to a given input. Therefore, we can collect in a set all the outputs provided by a Turing machine, but there is no way to know whether a given Turing machine (encoded by its program, which is a string) halts on not. Of course, if it halts, we know it, but if it does not halt, we cannot exclude that a halting configuration will be reached in the future. This means that, in general, we can decide positively the termination if it occurs, but we cannot decide negatively when it does not occur.

A recursively enumerable language is also called **semi-decidable**. In fact we can always correctly say yes, when a string belongs to it, but we cannot, in general, say no when a string does not belong to it. In other words, membership can be asserted in a finite amount of time, but non-membership cannot, in general, definitively ascertained in a finite amount of time.

A semi-decidable language can be generated as outputs of a Turing machine, in correspondence to all input strings over its alphabet. It is easy to show that a language can be generated by a Turing machine iff it is also recognized by some Turing machine. The class *RE* coincides with the class of semi-decidable languages, while the class *REC* coincides with the class of **decidable** languages (a language is decidable if a Turing machine exists answering "yes" for the (input) strings that belong to the language and "no" for those that do not belong to it.

A simple result about decidability (often mentioned as Post theorem) says that if a language L and its complementary $\bar{L}$ are both recursively enumerable, then they are both recursive. In fact, let us start, in parallel, the enumerations of the two languages, then any string will be generated by one of the two enumerations, therefore, in a finite time it will appear, and according to which enumeration will provide it, we can decide if it belongs to L or not. The converse implication is obvious. In fact if L is decidable, then we easily can enumerate both L and $\bar{L}$.

The existence of languages in *RE* which are not in *REC* was an important achievement of the computation theory founded by Turing. Such a language was defined by Turing, by using a technique resembling the diagonal argument of Cantor showing that natural numbers cannot be put in a bijective correspondence with the set of real numbers. In fact, if we enumerate all the strings over an alphabet $\alpha_1, \alpha_2, \ldots$ and all the Turing machines over the same alphabet $M_1, M_2, \ldots$ (by enumerating strings representing their programs), then the language:

$$K = \{\alpha_i \mid \alpha_i \in \mathbb{L}(M_i)\}$$

can be easily recognized as recursively enumerable, but its complementary $\bar{K}$ cannot be recursively enumerable because no Turing machine M_j can recognize it. In fact if this machine could exist, then either $\alpha_j \in \mathbb{L}(M_j)$ or $\alpha_j \notin \mathbb{L}(M_j)$, but both possibilities lead to a contradiction. In conclusion K is recursively enumerable, but $\bar{K}$ is not recursively enumerable, therefore K cannot be decidable, otherwise both K and $\bar{K}$ should be decidable.

The other crucial achievement of Turing's paper was the notion of **universal Turing machine**. In fact, a Turing M is univocally identified by its program, and it can be easily encoded with a string of a suitable alphabet. Turing showed that a Turing machine $\mathbb{U}$ can be constructed which, taken as an input an encoding $< M, \alpha >$ of M with the input α, can provide a computation which corresponds univocally to the computation of M with the input α. In particular, the computation of $\mathbb{U}$ with this input stops if and only if M stops on α, and this computation yields the same output produced by M with α as input.

Turing machines resulted to be *equivalent* to other formalism introduced for defining general classes of algorithmically effective computation processes. In fact, reasonable processes of mutual translations were found such that computations performed by Turing machines were encoded by computations performed in these other formalisms, and vice versa. Just for mentioning some of them: λ-calculus, Post systems, Herbrand-Gödel general recursive functions, Kleene-Gödel partial recursive functions, and Markov Algorithms are all *Turing-equivalent.*

In 1936, Alonzo Church published a paper where was formulated the famous **Turing-Church Thesis**:

Every computation can be realized, by means of some suitable encoding of data, by means of a Turing machine.

According to Turing-Church Thesis, the informal notion of computable function, can be adequately reduced to the rigorous concept of Turing computable function. Any formalism which results to be equivalent to Turing machines is said to be **computationally universal**.

We conclude by giving the schema of a sketch of Gödel's famous incompleteness theorem [175, 173, 188], where logical incompleteness is strictly related to the notion undecidability (formulae over a signature can be seen as strings of a particular formal languages). Basic logical notions of first-order logic are outlined in Chap. 5.

Given a model $\mathscr{M}$ over a domain D, a theory Φ **represents** a subset A of D, by means of a formula $\varphi(x)$ with a free variable x, if $a \in A \Leftrightarrow \mathscr{M} \models \varphi(a)$.

A theory is **gödelian** if it can represent any language of RE. A theory is **axiomatic** if it is constituted by all the logical consequences, called theorems, deducible, by means of a logical calculus (a deduction algorithm), from a finite set of sentences, called axioms. A theory is **complete** if for any sentence φ of its signature either φ or $\neg\varphi$ belongs to the theory. A theory is **consistent** if there is no sentence φ such that both φ and $\neg\varphi$ belong to the theory.

The following lemmas are the basis of Gödel's **incompleteness theorem**.

Lemma 6.8. *The Peano Arithmetic PA is gödelian.*

The previous lemma was proved by Gödel via a method, called *syntax arithmetization*, able to translate in arithmetical terms the logical notions of first-order theories.

Lemma 6.9 (Gödel). *Any consistent axiomatic theory that is complete is decidable.*

An informal argument proving the previous lemma is the following. In order to decide if φ is a theorem of a theory Φ, generate all the theorems you can deduce from the axioms, by the completeness of Φ either φ or $\neg\varphi$ will be generated. In the first case φ is a theorem, in the other case φ is not a theorem; both cases cannot occur because Φ is assumed to be consistent.

We know that there is a language K that is a RE language, but it is not decidable. This fact implies the following proposition.

Lemma 6.10. *If a consistent theory Φ is decidable, then it cannot be gödelian.*

Proof. If Φ were gödelian then Φ should represent the set K in RE, which is not decidable, that is, $a \in K \Leftrightarrow \Phi \models \varphi(a)$, for some formula $\varphi(x)$ with a free variable. But, being Φ decidable, the theorems of Φ are a decidable set, therefore also the subset of sentences $\varphi(a)$, which are theorems of Φ, are a decidable set. Therefore, K should be decidable, against the hypothesis that K is not decidable. □

No gödelian axiomatic theory can deduce all the arithmetical true sentences (that, of course, are a complete theory), as it is stated by the following theorem.

Theorem 6.11. *PA is incomplete.*

In fact, *PA* is axiomatic and gödelian. If it would be complete, then it should be decidable for Lemma 6.9, but this should contradict, by Lemma 6.10, its property of being gödelian.

Gödel Incompleteness says that there are true arithmetical propositions that cannot be deduced from *PA* axioms. In general, arithmetics cannot be axiomatized in FOL. This incompleteness, in the original proof of Gödel, was proved by using an auto-referential sentence, that is a form of a diagonal construction, related to Cantor's proof of non-enumerability of real numbers (and also to Russell's paradox). In the proof sketched above the diagonal argument is implicit in the Turing's construction of the language K (directly suggested by Cantor's diagonal argument).

First-order logic connects arithmetic, computability, and formal language theory [188, 204]. Many logical limitations are due to computational aspects, and at same time, many limitations of calculi are of logical nature. Analogously, deduction of consequences by means of a logical calculus is a special case of computation, but at same time any computation can be represented in terms of suitable logical deductions. Finally, any FOL (First-Order Logic) theory can be interpreted in models having natural numbers as domain. These facts show that arithmetic, logic, and computation are fields strongly connected.

6.7 Register Machines

Register machines are computation systems strictly related to Turing machines. They have a structure similar to the central unit of von Neumann's architecture, representing the general schema of a stored-program electronic computers, designed in the EDVAC (Electronic Digital Variables Automatic Computer) draft of 1945 [207, 194]. The constitutive element of EDVAC project is the electronic valve and its analogy with the neuron, according to McCulloch and Pitts' formalization [9]. The document by von Neumann provides the constitutive principles of a universal electronic computer; however, a systematic analysis of electronic circuits, in terms of boolean operations, was developed afterwards, on the basis on researches of Emil Post and Claude Shannon, in the years between 1930 and 1945. Register machines are computationally universal: for any Turing machine M there exists a register machine performing the same computations of M (by a suitable encoding of data of M as natural numbers).

Consider the cells of a Turing machine tape and, instead of arrange them linearly, associate to each of them an **address** expressed by a number or by a label. Moreover, instead of put in these cells symbols, assume to put in them numbers (if any number can occur, then the size of cells is unbounded). In a register machine, instead of moving along the cells, it is possible to apply a set of operations to the numbers contained in a cell, called **register**, by specifying the cell address. A register machine performs a computation according an internal **program** which is constituted by a sequence of **instructions**. Each instruction has three components: i) an order number, ii) an operation referred to registers, and iii) the number (or two numbers where only one is chosen) which has to be put in the (instruction) counter C. This number establishes the next instruction to execute. In fact, at each computation step, the machine executes the instruction having the order number contained in the counter. At the beginning of the computation, the counter contains the number of the first instruction to be executed. When the counter C contains a number which is not the number of an instruction of the program, then the computation stops. The instructions, with the related operations, are given in Table 6.16, where R_i, R_j denote two registers of addresses i, j respectively. Three special registers are assumed: an input register R_{in} providing data to the machine, an output register R_{out} providing results, plus the register C, with the role of **counter**. Any machine is assumed to have, apart the three special registers, all the registers necessary for executing the instructions of its program.

Table 6.16 The instructions of a register machine (applicable when the counter C contains h

$h, input_i, k$	the value of register R_{in} is put into the register R_{in} and k in C.
$h, output_i, k$	the value of register R_i is put into the register R_{out} and k in C.
h, inc_i, k	the value put in the register R_i is incremented of one unit and k in C.
h, dec_i, k	the value put in the register R_i is decremented of one unit and k in C.
$h, zero_i, k$	the value 0 is put in the register R_i and k in C.
$h, test_{i,j}, k, l$	if the values of R_i and R_j are equal then k, otherwise l, is put in C.

Table 6.17 A register machine computing the sum of two numbers

1. $input_1$ 2
2. $input_2$ 3
3. $zero_3$ 4
4. $test_{2,3}$ 7, 5
5. dec_2 6
6. inc_1 4
7. $output_1$ 8

It can be shown that the same computational power of register machines defined above continues to hold by substituting the two operations of test and decrement with a unique operation of conditional decrement $cdec_i\ k,\ l$ such that if the content of R_i is not zero decrements this register and put k in C, otherwise only puts l in C.

Register machines have the same computational power of structured **register while-programs**, which are realized by using only the operations of increment and decrement, but by structuring operations in **blocks** of type $while_j$. This means, all the operations in a $while_j$ block are iterated while the content of register R_j is not zero. For example, the program of Table 6.17 could be expressed by the register while-program given in Table 6.18. In this case, instructions do not need to have order numbers and next instruction numbers, because operations are executed in the order they are listed in the program, and the program execution ends when the last instruction is executed. When a while-block is entered, the exit from it happens when its condition does not continue to hold.

This notion of program is the basis of structured programming, which was a key feature of high level programming languages developed in 1970s. In a structured program, a correspondence holds between the order of instructions in the program

Table 6.18 A structured register while-programs for the sum of two numbers

```
$input_1$
$input_2$
while  $R_2 > 0$
       $dec_2$
       $inc_1$
end
$output_1$
```

and the order of their execution. In this way, program analysis results to be easier and may be developed in a more reliable manner.

The algorithm underlying the register program of Table 6.18 is the natural way to perform a sum, according to the finger-counting rule.

6.8 Information, Codes, and Entropy

In 1948 Claude Shannon published a booklet entitled "The mathematical theory of communication" [212], which is a scientific milestone and the first systematic mathematical investigation about a quantitative perspective in the analysis of information. The starting point of Shannon's approach is a probabilistic perspective based on the concept of **information source**, which is constituted by a set D of data with a function assigning a probability of emission to each element of D. The (probabilistic) measure of a datum d (with respect to logarithm basis b) is the logarithm of $1/p_d$ that is $-\log_b p_d$. The motivation of this definition is based on the idea that the information conveyed by an event is proportional to its rarity and that the information conveyed by a pair of independent events is the sum of the information conveyed by each of them. It is interesting to remark that the same intuition was the basis of a method elaborated by the Arabian crypto-analysts Al-Kindi (Ninth century A.D.) in deciphering the codes based on mechanisms of letter substitution. In fact, if in a text letter "a" is replaced by letter "p", then you can guess this encoding by counting the frequency of "p" in a sufficiently long text of the language to which this encoding is applied.

6.8.1 Shannon's Entropy

The crucial concept of Shannon's theory is the (information) entropy of an (information) source defined by:

$$-\sum_{d \in D} p_d \log_b p_d \tag{6.1}$$

Shannon's entropy is the center of a theoretical framework where codes and digital information can be rigorously analyzed and applied to an enormous spectrum of scientific fields [198, 200].

The notion of **code** is easily defined by a function from **codewords** to data. Codewords are strings over a prefixed finite alphabet. A code C is **proper** if there is only one codeword for every datum, otherwise the code is **redundant**. Here, we consider only proper codes. In the case of a proper code, the **digital size** of the information of a datum, with respect to the code, is given by the length of the codeword associated to the datum.

Given a code C we define $digit(C)$ by the following equation, where $|\alpha|$ denotes the length of the string α:

$$digit(C) = \sum_{\alpha \in C} |\alpha|$$

Let us define $digit(n)$ as the minimum value of $digit(C)$ for C varying in the proper codes of n codewords. The following result can be shown, which again provides a strict link between logarithms and information:

$$n\lceil log_k n \rceil \geq digit(n) \geq n(\lfloor log_k n \rfloor - 2).$$

6.8.2 Optimal Codes and Compression

Codes are **univocal** when do not exist two different sequences of codewords which can be concatenated by providing the same string. These codes have important properties which are essential for providing efficient decoding processes when data are transmitted along a transmission channel. Kraft norm $||C||$ of a code C is defined by:

$$||C|| = \sum_{\alpha \in C} |A|^{-|\alpha|}$$

where A is the alphabet of the code, and $|A|$ denotes the cardinality of A. Kraft norm of a univocal code is always equal to or less than 1. A special kind of univocal code are the **instantaneous** codes, having the property that no pair of their codewords α, β can exist such that α is a prefix of β. Any univocal code C can be always transformed into an instantaneous code C' such that $||C|| = ||C'||$. Instantaneous codes can be defined by means of an encoding tree (an example is given n in Fig. 6.7), where codewords correspond to the leaves of the tree (analogous trees can be constructed for alphabets with more than two symbols).

An **information source** is any device emitting data $d \in D$ with probability $p(d)$, the **mean length** L_C of a proper code C is given by:

$$L_C = \sum_{\alpha \in C} p_\alpha |\alpha|.$$

A code C is **optimal**, with respect to an information source, if no other code exists having a shorter mean length with respect to that information source. A very simple algorithm due to Huffman exist for defining an optimal code with respect to an

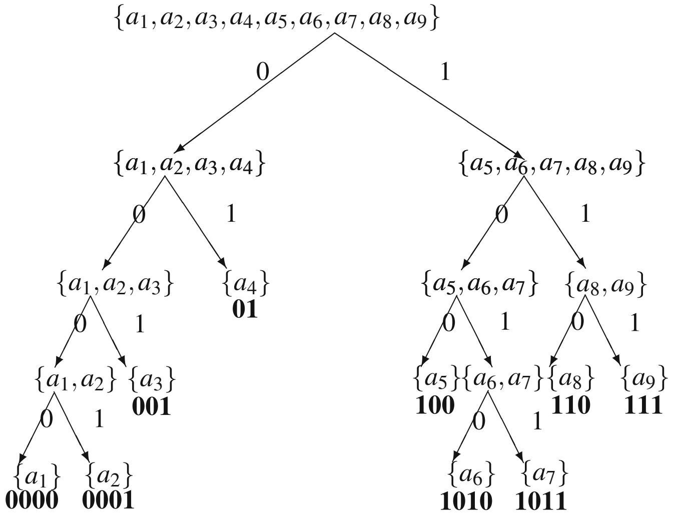
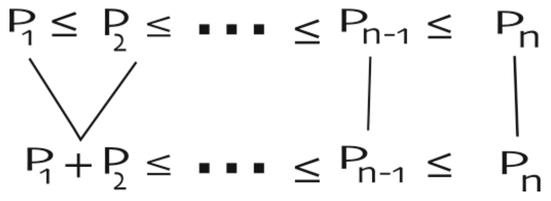


Fig. 6.7 An encoding tree

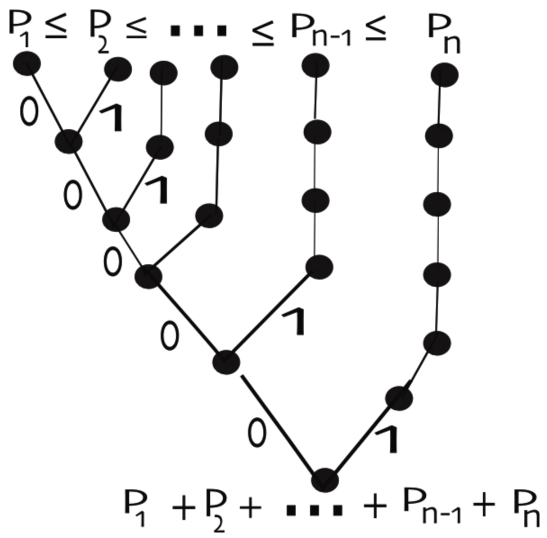
information source. Let us recall briefly the Huffman method. Given k data with $k > 1$ and with probabilities $p_1, p_2, \ldots, p_k$ respectively, let us define the following process of tree construction.

1. Consider two probabilities p_1, p_2 such that no probability is smaller than them, then introduce a new probability $p_1 + p_2$ and connect this new probability with their summands and label these edges with 0 and, 1 respectively.
2. Apply again the procedure of the previous step to a set of probabilities where p_1, p_2 are replaced by $p_1 + p_2$ (see Fig. 6.8), and iterate this step until only a unique probability is obtained (at each step the set of probabilities decreases by one unit, see Fig. 6.9).
3. Associate to the datum of probability p_j the sequence of labels of the edges connecting the probability p_j with the final unique probability.



$P_1 \leq P_2 \leq \ldots \leq P_{n-1} \leq P_n$
$P_1 + P_2 \leq \ldots \leq P_{n-1} \leq P_n$

Fig. 6.8 The initial step of an Huffmann code



$P_1 \leq P_2 \leq \dots \leq P_{n-1} \leq P_n$
0
1
0
1
0
0
1
0
1
$P_1 + P_2 + \dots + P_{n-1} + P_n$

Fig. 6.9 A Huffmann code tree

Shannon's first theorem shows that entropy is a lower bound to the code optimality. In fact, if H is the entropy of an information source and L_C is the mean length of C with respect to this information source, then:

$$L_C \geq H.$$

Let T be a text, that is a sequence $\alpha_1\alpha_2\dots\alpha_m$ of codewords, and let us denote by $|\alpha|$ the length of a codeword α. The digital size of T (with respect to the code of its codewords) is given by:

$$digit_C(T) = \sum_{j=1,m} |\alpha_j|.$$

A compression of T is a text T' such that, for some code C': i) $digit_{C'}(T') < digit_C(T)$ and ii) text T can be univocally recovered from T'. The **compression ratio** is $digit_{C'}(T')/digit_C(T)$. This ratio together with the space and time complexity of a compression algorithm are the basic elements to evaluate a compression method.

It is easy to realize that no universal compression algorithm can exist. In fact, no compression method can satisfy the two conditions above for every text. If this universal method would exist, then by applying a compression, every text of a certain digital size q could be reduced to a text of digital size $q' < q$. But of course the number of texts of size q' are less than the number of texts of size q, therefore it is impossible to recover, univocally, every q-text from some q'-text. The impossibility of universal compression methods implies that we can only try to discover compression methods which work for specified classes of texts.

In compression theory there are three main principles: i) encoding more probable codewords in a shorter way, ii) trying to determine a dictionary of the codewords occurring in a given text, and iii) changing the order of codewords in such a way that similar codewords become contiguous. In fact, contiguity of similar codewords allow for replacing a subtext α^n by the pair (α, n), which can be encoded more shortly than α^n.

6.8.3 Typicality and Transmission

Shannon's second theorem concerns with the transmission error. It shows that, even if transmission is affected by an error probability, when data are encoded in a suitable way, this probability can be made indefinitely small.

After the theoretical result of the second theorem, transmission codes were introduced where the reliability of transmission is obtained by adding extra symbols which can recover the lost information when some noise corrupts the invoiced messages. These extra symbols are called control digits. For this reason, an encoding with m digits plus k control digits encodes, in the binary case, 2^{m+k} messages which send only 2^m different messages. The ratio $m/m+k$ is the **transmission rate**. The **capacity** of a channel transmitting between a sender information source and a receiver information source is the maximum amount of mutual information (see later) which can pass between the two sources (the transmitter and the receiver). In more precise terms, the second theorem claims that when the transmission rate is less than the capacity of the channel, then the more we reduce the transmission rate, the more the error probability decreases.

The second theorem is a masterpiece of Shannon's approach and was the beginning of the theory of the self-correcting transmission codes. Its proof relies on the notion of **typical** sequence generated by an information source. When an information source emits a sequence of data, the longer is the sequence, the more it approximates to a typical sequence of the source. In fact, a sequence is typical when the frequency of any datum occurring in it is the same as the emission probability of the source. This means that sufficiently long sequences can assumed to be typical, with a small error probability. The deep connection between typicality and entropy is given by the following asymptotic results: i) the probability of any typical sequence of length n approximates, for increasing n, to 2^{-Hn}, where H is the entropy of the information source, ii) the number of typical sequences of length n approximates, for increasing n, to 2^{Hn}.

The theoretical framework necessary to the proof of the second theorem includes many important notions related to the entropy. In this context, it is useful to consider an information source as a casual variable, that is, a variable with an associated probability of assuming their values. If $H(X)$ is the entropy of X:

$$H(X) = -\sum_{X=x} p(x) \log p(x)$$

then the **joint entropy** of two casual variables X, Y is given by:

$$H(X,Y) = -\sum_{X=x,Y=y} p(x,y)\log p(x,y)$$

the **entropy** of X **conditioned** to Y is given by the following equation where $p(x|y)$ is the conditional probability of having $X = x$ when $Y = y$:

$$H(X|Y) = -\sum_{X=x,Y=y} p(x,y)\log p(x|y)$$

it can be shown that:

$$H(X|Y) = -\sum_{X=x,Y=y} p(y)H(X|Y=y)$$

and

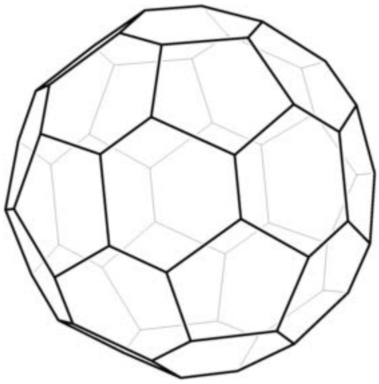
$$H(X,Y) = H(X) + H(Y|X).$$

Other two important and related notions of information theory are **mutual information** $I(X|Y)$ between X and Y and the **entropic divergence** between them.The mutual information (usually defined by means of the entropic divergence) satisfies the following important equation:

$$I(X|Y) = H(X) - H(X|Y)$$

the equation above explains the name used for denoting $I(X|Y)$. In fact, let us identify the entropy of an information source with the average (probabilistic) information of the data that it emits. if we consider X and Y as two processes, where the first one is an information source sending data through a channel, and the second is the information source of the data as they are received, then $I(X|Y)$ is a (probabilistic) measure of the information passing from the sender to the receiver: the average information of X (the transmitted information) minus the average information of X when Y (the received information) is given.

The entropic concepts can be easily extended from discrete to continuous variables. This extension is necessary to the analysis of transmission by means of signals. They are functions, periodical in time, realized by electromagnetic waves. In this case, data are encoded by altering a wave in a suitable way (modulating it), and when the wave is received its alterations are decoded (demodulated) for recovering the encoded messages. This method mixes discrete methods for representing information with continuous signals playing the role of channels. Shannon shows the *sampling theorem*, extending an already known result, according to which the capacity of a periodical function is related to its maximum frequency (in its composition as sum of circular functions, in Fourier representation). By using the sampling theorem Shannon shows his celebrated third theorem giving the capacity of a continuous signal affected by a noise of power N. This capacity is given by the maximum frequency of the signal multiplied by a factor which is the logarithm of the ratio $P+N/N$, where P is the power of the signal.


Archimedes' Truncated icosahedron

7 Combinations and Chances

Abstract. Combinatorics is the field of mathematics which provides methods for counting discrete structures of a given type. The exact evaluation of these numbers has very often a great importance in many theoretical and applicative contexts. In this chapter the basic formulae of combinatorics will be presented, by emphasizing the links among combinatorics, probability, statistics, and biological applications. The Least Square Evaluation method is outlined, and basic notions about trees and graphs are also provided with the fundamental enumeration formulae of these structures.

7.1 Factorials and Binomial Coefficients

The main aim of combinatorics is the evaluation of the numbers of finite mathematical structures of a given type. In the following sections, we present the basic combinatorial schemata which occur frequently in the problems of counting finite structures. A simple and unifying way of describing combinatorial schemata is the notion of **allocation**. An allocation is defined by a set of objects and a set of cells, and a way of putting objects into cells. Many kinds of allocations can be considered in correspondence to the following features: i) whether or not objects are distinct, ii) whether or not cells are distinct, iii) whether or not objects can be repeated, and iv) whether or not cells can be repeated.

An important aspect in this regard is the explanation of what exactly means "distinct". In fact, two objects are of course distinct, otherwise they is only one object. However, in many contexts it is necessary to discern between *distinct* and *distinguishable*. Two balls having exactly the same shape are distinct when you put them together on a table, but they may be not easily distinguishable if you put one of them alone, and then the other one, or the same, at subsequent points in time. In the

V. Manca: *Infobiotics*, ECC 3, pp. 309–368.
DOI: 10.1007/978-3-642-36223-1_7 © Springer-Verlag Berlin Heidelberg 2013

context of combinatorial schemata, objects or cells are always distinct when you count them, but they are undistinguishable when some of them can occur many times and these multiple occurrences are counted as the occurrence of the same object (distinguishability implies distinctness, but the converse implication does not hold). In the following, when we speak of objects or cells, we assume that they are distinguishable, unless the contrary is stated. The combinatorial schemata that we will analyze will clarify these aspects.

7.1.1 Permutations and Arrangements

Let us consider n **objects**. How many possibilities we have of arranging them in a sequence? Such an arrangement is usually called a **permutation** of n objects. In set-theoretic terms a permutation is a bijective function. In fact, we determine this number by using an allocation schema associating the n objects to n **cells** or locations numbered from 1 to n (this terminology is used for a better visualization of the arrangement). In the first cell we can choose one of n objects, in the second one of the remaining $n-1$ objects, and so on. In the last cell only one possibility of choice remains, because the other $n-1$ objects are already allocated. In conclusion, the number of n-permutations is

$$n(n-1)(n-2)\dots 2\cdot 1 = n!$$

In this sense, factorial numbers count permutations.

The argument for counting permutations can be easily extended to the case of **arrangements**, that is, injective functions from n objects to k places, where $k < n$ (permutations are arrangements where $n = k$). In this case, we get the following product, which we call k-**subfactorial** of n:

$$n(n-1)(n-2)\dots(n-k+1) = (n)_k.$$

Of course:

$$(n)_k = n!/(n-k)!$$

Let us consider allocations of objects in cells where any object can occur any number of times, but only one object can be allocated to each cell. How many possible allocations of this kind, of n objects to k cells are there? Any allocation of this type corresponds to a function from the cells to the objects. In this case, at any step of the allocation process, n choices are possible, whence we have n^k different arrangements with repetition.

The number of permutations and arrangements are useful for counting words. How many words of length 10 can be written with 20 letters? They are 20^{10}. How many words of length 10 where no letter occurs more than once? They are $(20)_{10}$. How many different words can be obtained by rearranging the order of letters in the word *number*? $6! = 720$.

It is interesting to remark that the ratio between the number of injective functions and all possible functions becomes very small when n increases. This phenomenon, referred to as the *rarity of injections* has very surprising consequences. For example, in a town where seven accidents occur each week, it is very improbable that these accidents are distributed one per day. In fact, $7!/7^7 < 0.00613$, therefore unlucky days and lucky days, most probably, can be experienced. A famous example in this regard is the so-called birthday phenomenon. If you consider a class with m students, with $m > 23$, you have a big probability of finding two of them who were born on the same day of the year. In general, the probability that m birthdays coincide is given by:

$$p = 1 - \frac{(365)_m}{365^m} = \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\cdots\left(1 - \frac{m-1}{365}\right).$$

If we discard the products between fractions (which are very small) the expression can be approximated by:

$$p \approx 1 - \frac{1+2+3\ldots+(m-1)}{365} = 1 - \frac{m(m-1)}{730}.$$

The previous examples are close to an important combinatorial principle known as **Pigeonhole Principle**.

Table 7.1 The Pigeonhole Principle

For any allocation of n objects to m cells, where $m < n$, at least one cell has to contain more than one object.

In fact, in a class with more than 366 students, at least two of them were born in the same day (the date February 29 is included). This principle has very important consequences. For example, if a dynamical system has a finite number of possible states, with no terminal state (where it may remain forever), then it has to be *eternally recurrent*, that is, some of its states occurs an unbounded number of times.

7.1.2 Combinations and Binomial Coefficients

How many possibilities are there of choosing k objects among n given objects? That is, how many k-subset does a set of cardinality n have?

A k-subset of n objects is also called a k-combination over n objects. Also in this case we can use an allocation schema. In fact, a combination is obtained by allocating k of the n objects to k undistinguishable cells, with exactly an object per cell. Therefore, the number of k-combinations of n objects can be deduced by allocating the n objects to k distinguishable cells, in $(n)_k$ ways, and then by dividing this number for the number of possible different ways these cells can be ordered, that is $k!$ ways. In conclusions the combinations we are searching are:

$$\frac{(n)_k}{k!} = \frac{n!}{k!(n-k)!}$$

the standard notation for this number is:

$$\binom{n}{k}.$$

These numbers are called **binomial coefficients**, because (as Newton showed) they are the coefficients occurring in the expansion of the powers of binomials:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k\, b^{n-k}$$

By using the factorial representation of subfactorials we get the following equation:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

The formula above is simple, but it is very complex to be computed. In fact, when numbers are big it is computationally prohibitive. In the next section we will provide a recurrent formula for computing $\binom{n}{k}$ without using factorials.

7.1.3 Inductive Derivation of Binomial Coefficients

There is only one possibility of choosing n objects among n given objects, and there are n possibilities of choosing one object among them. Of course, zero objects among n corresponds to the choice of the empty set, therefore we can put the following equalities:

$$\binom{n}{n} = 1$$

$$\binom{n}{1} = n$$

$$\binom{n}{0} = 1.$$

Theorem 7.1. *For any $n \geq k > 0$:*

$$\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$$

Proof. Let us consider $n+1$ objects ($n \geq 0$). We can select $k+1$ of them ($k \leq n$) by using the following procedure. Let us fix one object among the given $n+1$ objects, call it a. The subsets of the $k+1$ chosen elements can belong to one of two classes: the sets including a and the sets that do not include a. The sets of the first type are in one-to-one correspondence with the choices of k objects among n, because a can be added to any choice of k. The sets of the second type are in one-to-one correspondence with the choices of $k+1$ objects among n, because a is never chosen. In conclusion, if binomial coefficients provide the number of j-subsets for sets of size smaller than $n+1$ (with $j \leq n$), then: the number of combinations with a is $\binom{n}{k}$, and the number of combinations without a is $\binom{n}{k+1}$. This concludes the proof. □

The number of combinations is easily computed by the following triangle due to Tartaglia and Pascal, but already known by Chinese mathematicians at least 1000 years earlier. It is based on the previous theorem. In fact, the n-th row provides the values of $\binom{n}{k}$ for k going from 0 (left) to n (right). In the triangle, the elements of the $(n+1)$-th row are obtained by summing the two elements over it in the n-th row (the top left and the top right). The first seven rows of this triangle are the following:

$$\begin{array}{ccccccccccccc}
 & & & & & & 1 & & & & & & \\
 & & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 2 & & 1 & & & & \\
 & & & 1 & & 3 & & 3 & & 1 & & & \\
 & & 1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \\
1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1
\end{array}$$

Tartaglia's (or Pascal's) triangle is symmetric. In fact, choosing k objects among n is equivalent to chose $n-k$ among n, which correspond to those which are not chosen, that is:

$$\binom{n}{k} = \binom{n}{n-k}.$$

Of course, the definition by induction has to coincide with the definition by means of factorials, that is:

$$\frac{(n)_k}{k!} + \frac{(n)_{k+1}}{(k+1)!} = \frac{(n+1)_{k+1}}{(k+1)!}.$$

Newton's formula, for computing the power of a binomial, is based on binomial coefficients. In fact, it is easy to realize that the coefficient of a power $a^k b^{n-k}$, of degree n, coincides with the number of ways the factor a can be chosen in the product $(a+b)(a+b)\dots(a+b)$ of n binomials (so that b is chosen $n-k$ times), therefore:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

When in this formula $a = b = 1$ we get the number 2^n of all possible subsets of n elements. In fact, k-subsets, with k going from 0 to n, are all possible subsets of n elements:

$$2^n = \sum_{k=0}^{n} \binom{n}{k}.$$

Newton's formula is the starting point of a lot of important mathematical theorems. An example is the following **Fermat's little theorem** which follows easily by induction from Newton's formula. It is an important property of modular arithmetic, based on congruences. Notation $n \equiv m$ mod k denotes the **congruence** of n and m **modulo** k, holding when the divisions of n and m by k provides the same remainder.

Proposition 7.2 (Fermat's little Theorem). *If p is a prime number then, for any $x \in \mathbb{N}$:*

$$x^p \equiv x \; mod \; p$$

Proof

$$(x+1)^p = \sum_{k=0}^{n} \binom{p}{k} x^k$$

in the right side of the equation, apart 1 and x^p, other terms have coefficients with the following forms with $1 < j < p$:

$$p!/(j!(p-j)!)$$

and, due to the primality of p, the factor p cannot be removed from the numerator, therefore they are congruent to zero modulo p. In conclusion, for $x = 1$ the equation of the proposition trivially holds, then if we assume that it holds for a value x, then we have:

$$(x+1)^p \equiv x^p + 1 \text{ mod } p$$

but, by induction:

$$x^p \equiv x \text{ mod } p$$

therefore:

$$x^p + 1 \equiv x + 1 \text{ mod } p$$

and, by the transitive property of equality, this implies:

$$(x+1)^p \equiv x + 1 \text{ mod } p.$$ □

7.2 Distributions and Discrete Probability

Diagrams 7.1 e 7.2 are a kind of discrete version of Gaussian distribution shown in Fig. 7.3. This corresponds to the fact that by using Stirling approximation, explained in the next section, it is possible to obtain a formula, due to De Moivre and Laplace, which approximates the binomial coefficients with a bell-shaped curve, usually called *gaussian* because Gauss showed its deep probabilistic meaning. In fact, if we consider the Bernoulli's law of two possible events, say $\{1,0\}$, of probabilities p and $q = 1 - p$, respectively, then the probability $p(n,k)$ of having k times 1 in n events is given by:

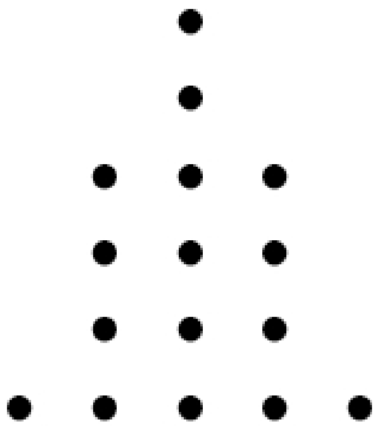


Fig. 7.1 The 16 subsets of a set of 4 elements, grouped in columns of equinumerous subsets

$$p(n,k) = \binom{n}{k} p^k q^{n-k}. \tag{7.1}$$

Equation 7.1 is easily explained by following the model of balls extraction from a urn where p is the percentage of white balls (1 value) and q is the percentage of black balls (0 value, apart the color, the balls are undistinguishable). Assuming a large number of extractions, if we replace factorials by their Stirling's approximations (we avoid to give here the details), then the probability of extracting n balls where k are white is given by the De Moivre–Laplace formula:

$$p(n,k) = \frac{1}{\sqrt{npq}\sqrt{2\pi}}\ e^{-\left(\frac{1}{2}\left[\frac{(k-np)}{\sqrt{npq}}\right]^2\right)}.$$

The mean value is np and the mean value of the deviation from it is $\sqrt{npq}$. If these two values are replaced by μ and σ, respectively, we get the famous Gaussian distribution of mean μ and standard deviation σ:

$$n_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\left(\frac{1}{2}\left[\frac{(x-\mu)}{\sigma}\right]^2\right)}.$$

The area under this curve, between two extremes, gives the probability of having values comprised between them. This law can be found in many random phenomena, going from the distribution of the height in a population of individuals, to the distribution of the errors in the measures of a physical quantity.

Another fundamental law of aleatory nature is the Poisson law (see Fig. 7.4), which can be derived from Bernoulli's law under suitable hypotheses, typical of

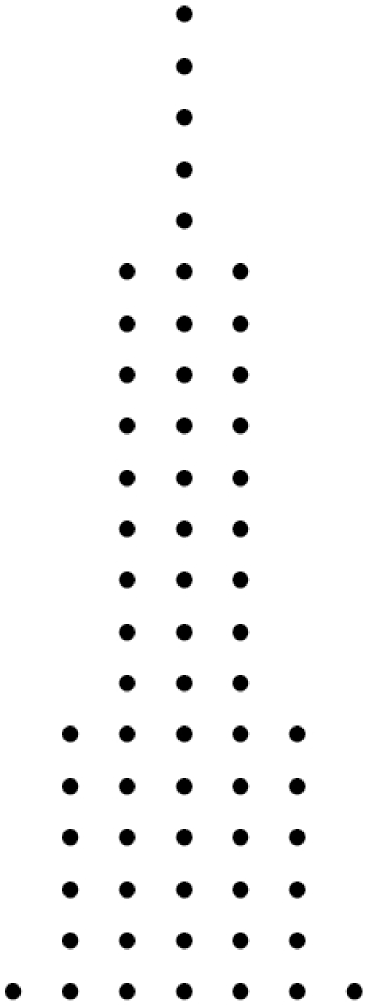


Fig. 7.2 The 64 subsets of a set of 6 elements, grouped in columns of equinumerous subsets

rare events: a number n very big of possible events and a probability p very small of having k successful events . In this case, if we call λ the product np (λ real), the probability is given by the following Poisson distribution (of parameter λ):

$$p(k,\lambda) = e^{-\lambda}\,\frac{\lambda^k}{k!}.$$

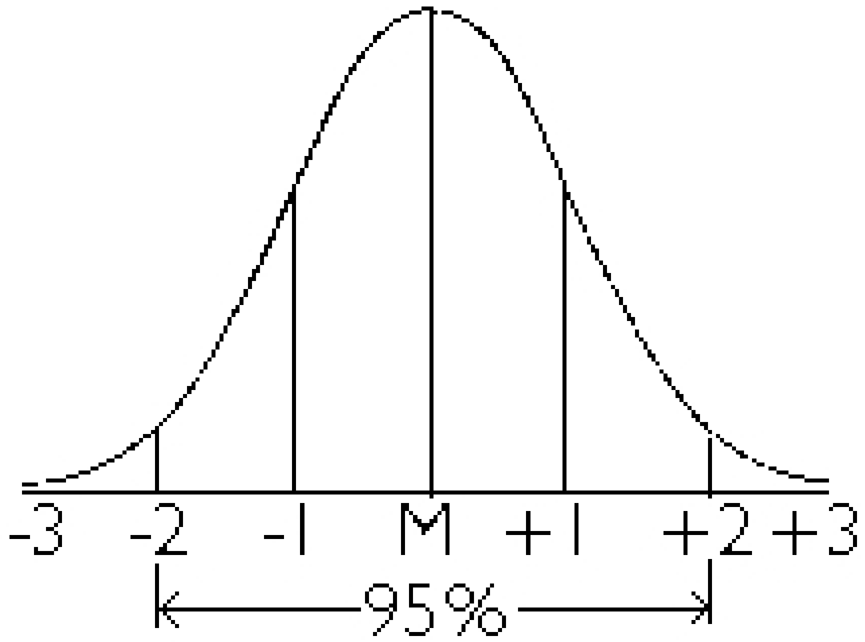


Fig. 7.3 Gauss's distribution. M denotes the mean. More than 68% of the values is within a radius σ centered on M; more than 95% is within 2σ, and more than 99% within 3σ

7.3 Allocations and Partitions

In the following we consider other important kinds of allocations and some kinds of partitions, which are special cases of allocations where cells are assumed undistinguishable and no cell is empty.

7.3.1 Multinomial Coefficients

If in the formula of binomial coefficients by means of factorials we put at denominator the product of m factorials such that $k_1 + k_2 + \ldots k_m = n$, we get the following formula for **multinomial coefficients**, related to the Boltzmann distribution in physics (of the fractional numbers of particles in a gas having some energy levels):

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\ldots k_m!}.$$

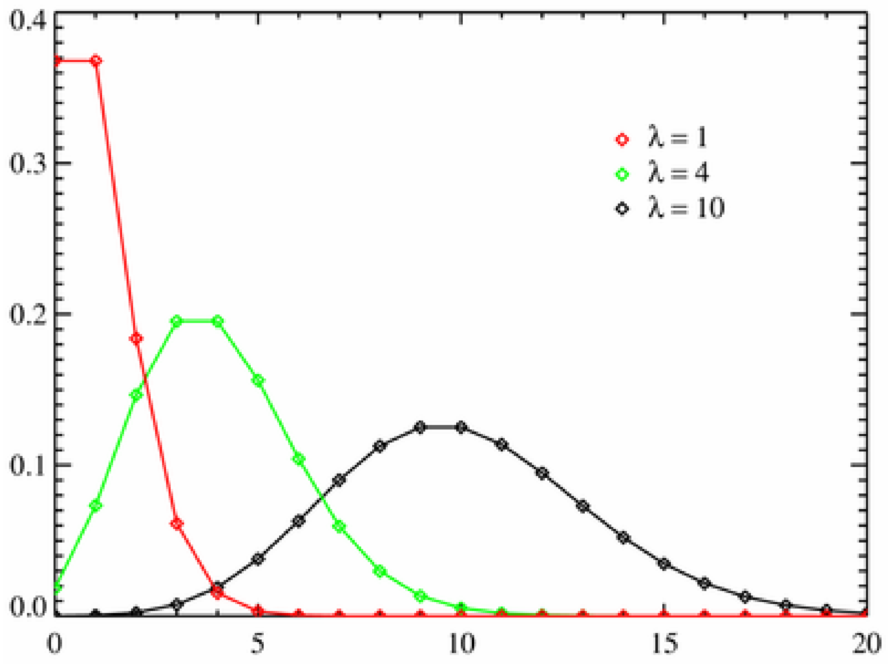


Fig. 7.4 Poisson distribution for different values of λ (if $\lambda > 10$ it approximates to a gaussian)

This formula has a simple combinatorial meaning. It corresponds to the number of different ways of allocating n distinguishable objects into m distinguishable cells, where $k_1, k_2, \ldots k_m$ objects are allocated, respectively in the m cells.

It gives also the number of **permutations** with repetitions, that is, the different ways we can change the positions to the elements of a sequence over an alphabet, say $\{a,b,c,d\}$, with n_a occurrences of a, n_b of b, n_c of c, and n_d of d. It is easy to show that:

$$\binom{n}{k_1, k_2, \ldots, k_m} = \binom{n}{k_1}\binom{n-k_1}{k_2}\binom{n-k_1-k_2}{k_3}\cdots\binom{n-k_1-k_2\ldots-k_{m-2}}{k_{m-1}}.$$

7.3.2 *Partitions and Multisets*

Binomial coefficients provide also the formula for computing the number of partitions of n undistinguishable objects into k distinguishable cells possibly empty.

In fact, for obtaining this number we can add $k-1$ undistinguishable tokens, that is, elements which are different from the given n undistinguishable objects. Then, we consider all permutations of a sequence of $n+k-1$ objects of two types. Each permutation provides a partition into different cells. In fact, the $k-1$ tokens determine k different locations in a sequence: before the first occurrence, between the first and the second occurrence, and so on. Therefore, by choosing the positions where to put the n objects of the first type, we get the following value corresponding to this kind of partitions:

$$\binom{n+k-1}{n}.$$

If we want the number of partitions of n undistinguishable objects into exactly k non-empty distinct cells ($n \geq k$), then we apply the same method given above, but we force the presence of one object before each of the k separators. Therefore, in this case we get the following number of possibilities:

$$\binom{n-1}{k-1}.$$

With an analogous argument we can determine the number of k-multisets over a set of n elements. In fact, consider now $n-1$ undistinguishable separators and k undistinguishable tokens. Each permutation of $n+k-1$ objects with $n-1$ of one type and k of a different type (tokens and separators), represents a multiset where the number of tokens before the $i-1$ separator corresponds to the multiplicity of the i-th object of the set. For example, the multiset $3a+2b+4c$ of size 9 over 3 different kinds of objects, a,b,c, would be represented by (1 as separator and 0 as token):

$$00010010000.$$

Therefore, the number of k-multisets over a set of n elements is given by:

$$\binom{n+k-1}{k}.$$

7.3.3 Integer Partitions

A partition of a positive integers n is a sum of positive integers equal to n. For example, $10 = 2+3+5$ is a partition of 10 in three parts (the summands $2,3,5$). This kind of partition corresponds to allocate n undistinguishable objects to a number of undistinguishable, non empty, cells. We denote by $P(n)$ the number of partitions of the integer n. Euler found a generating function for $P(n)$, that is a function that, when represented as infinite series of monomials $a_n x^n$, with $n \in \mathbb{N}$, satisfies the equation $a_n = P(n)$. In fact, Euler has proven that:

$$\sum_{n=0}^{\infty} P(n)x^n = \prod_{k=1}^{\infty}\left(\frac{1}{1-x^k}\right).$$

The following asymptotic expression for $P(n)$ was first obtained by Hardy and Ramanujan in 1918 [224]:

$$P(n) \sim \frac{\exp\left(\pi\sqrt{2n/3}\right)}{4n\sqrt{3}} \text{ as } n \to \infty.$$

Many other recurrent enumerative formulae are also known. Let us mention only one of them, due to Euler:

$$P(n) = P(n-1) + P(n-2) - P(n-5) - P(n-7) + P(n-12) + P(n-15) - \dots$$

where $P(n)$ is recurrently computed by using the values $P(n-q)$ with $q \in \{(3k^2 + k)/2 \mid k \in \mathbb{Z}\}$, according to the following formula ($P(1) = 1, P(i) = 0$ for $i \leq 0$):

$$P(n) = \sum_{k \in \mathbb{Z}} (-1)^{k+1} P(n - (3k^2 + k)/2).$$

Numbers q above are called **pentagonal numbers** because in the case of k negative they correspond to figurate numbers already known by Greek mathematicians, which can be arranged in pentagonal shapes (differences of two suitable triangular numbers). This formula is one of Euler's jewels, known as **Euler's Pentagonal Number Theorem**.

The following tables show the first 30 values of $P(n)$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(n)	1	2	3	5	7	11	15	22	30	42	56	77	101	135	176

n	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
P(n)	231	297	385	490	627	792	1002	1255	1575	1958	2436	3010	3718	4565	5604

Now we present a new perspective were partitions are classified according to their minimum summand. In this way, we discover a recurrent formula computing $P(n)$ by using elementary combinatorial arguments (see Eq. (7.4)). However, computational experiments show that the recurrent pentagonal formula of Eulero is computationally better than our formula. Nevertheless, we briefly present this elementary method for computing integer partitions, as an example of derivation of recurrent relations in a classical combinatorial context.

Let us denote by $P_{min(k)}(n)$ the number of partitions of n having k as minimum summand.

Lemma 7.3
$P_{min(n)}(n) = 1$
$P_{min(k)}(n) = 0 \quad \textit{for} \quad \lfloor n/2 \rfloor < k < n$
$P_{min(k)}(n) = 1 \quad \textit{for} \quad \lfloor n/3 \rfloor < k \leq \lfloor n/2 \rfloor.$

Proof. If $k = n$ there is obviously only one partition of n with k as minimum summand. If k is smaller than n, but greater than $\lfloor n/2 \rfloor$, no partition of n can exist with minimum part k. If k is comprised between the two extreme values $\lfloor n/2 \rfloor$ and $\lfloor n/3 \rfloor$, then there is exactly one partition of $n = k + (n-k)$ having k as minimum part. □

Lemma 7.4. *For $n > 1$,*

$$P(n) = \sum_{i=1}^{\lfloor n/3 \rfloor} P_{min(i)}(n) + \lfloor n/2 \rfloor - \lfloor n/3 \rfloor + 1. \tag{7.2}$$

Proof. Of course, $P(n) = \sum_{i=1}^{n} P_{min(i)}(n)$, therefore the equation above follows from the previous lemma. □

Let us denote by $P_k(n)$ the number of partitions of n with at most k summands, which is computable by means of the following well-known inductive equation:

$$P_k(n) = P_{k-1}(n) + P_k(n-k)$$

with $P_1(n) = P_n(n) = P_k(0) = 1$.

The following lemma expands the integer partitions enumeration formula (7.2) of Lemma 7.4.

Lemma 7.5. $P_{min(1)}(n) = P(n-1)$. *For $k > 1$ and $n > k$,*

$$P_{min(k)}(n) = \sum_{i=2}^{\lfloor n/k \rfloor} P_{i-1}(n-ik).$$

Proof. In general, $P(n-m)$ is the number of partitions of n where the summand m occurs. Therefore, $P_{min(1)}(n) = P(n-1)$. Given an arbitrary partition, say it π, of $n-ik$ with at most $i-1$ summands, we can transform it (in a 1-to-1 way) in a partition of n with i summands, say it π', having k as its minimum summand. Namely, consider the ik units of n which are exceeding $n-ik$, then add k of them to each of the $i-1$ (possibly null) terms of π, and use the remaining k units for the i-th summand of π' (so that in π' the presence of a minimum summand k is guaranteed). In this manner, if i varies from 2 to $\lfloor n/k \rfloor$ ($n > k$), then we get $P_{min(k)}(n)$ by means of the formula given above. □

However, the previous enumeration, based on minimum summands, can be substantially improved by using the following lemma.

Lemma 7.6. *For $k < \lfloor n/3 \rfloor$ and $1 \leq n$,*

$$P_{min(k+1)}(n) = P(n-k-1) - \sum_{i=1}^{k} P_{min(i)}(n-k-1)$$

Proof. Let us consider the partitions of n having $k+1$ as a part, which equals $P(n-k-1)$. Now remove from these the partitions that have 1 as a minimum part, those having 2 as a minimum part, and so on, up those having k as a minimum part. In this way, we obtain the partitions of n having $k+1$ as minimum summand. □

Let us define by induction the following **partition difference** operator D^k, of degree k, for $n > 1$ and $k \leq n$ (see Example 7.9 for a computation of D):

$$\begin{cases} D^1(n) = P(n-1) \\ D^{k+1}(n) = P(n-k-1) - \sum_{i=1}^{k} D^i(n-k-1) \end{cases} \tag{7.3}$$

Lemma 7.7. *For $k < \lfloor n/3 \rfloor$,*

$$P_{min(k)}(n) = D^k(n).$$

Proof. This lemma follows easily by induction on k, from the definition of D^k and the previous lemma. In fact, $P_{min(1)}(n) = P(n-1) = D^1(n)$. Moreover, from the previous lemma, $P_{min(k+1)}(n) = P(n-k-1) - \sum_{i=1,k} P_{min(k)}(n-k-1)$. Therefore, by induction hypothesis, we can replace the terms of the right side by obtaining $P_{min(k+1)}(n) = P(n-k-1) - \sum_{i=1,k} D^k(n-k-1)$, from which, according to the definition of D^{k+1}, we get $P_{min(k+1)}(n) = D^{k+1}(n)$. □

Finally, from lemmas 7.4, 7.6, and 7.7 a theorem follows which provides a simple recurrent formula enumerating integer partitions.

Theorem 7.8. $P(1) = 1$ *and* $P(2) = 2$. *For $n > 2$,*

$$P(n) = \sum_{i=1}^{\lfloor n/3 \rfloor} D^i(n) + \lfloor n/2 \rfloor - \lfloor n/3 \rfloor + 1. \tag{7.4}$$

Example 7.9. The computation of $P(27) = 3010$, according to formula (7.4).

$P(27) = D^1(27) + D^2(27) + D^3(27) + D^4(27) + D^5(27) + D^6(27) + D^7(27)$
$+D^8(27) + D^9(27) + \lfloor 27/2 \rfloor - \lfloor 27/3 \rfloor + 1$
$D^1(27) = P(26) = 2436$
$D^2(27) = 383$
$D^3(27) = 110$
$D^4(27) = 39$
$D^5(27) = 18$
$D^6(27) = 9$
$D^7(27) = 5$
$D^8(27) = 3$
$D^9(27) = 2$
$P(27) = 2436 + 383 + 110 + 39 + 18 + 9 + 5 + 3 + 2 + 13 - 9 + 1 = 3010.$

The detailed computation of $D^4(27) = 39$

$D^4(27) = P(23) - D^1(23) - D^2(23) - D^3(23)$
$P(23) = 1255$
$D^1(23) = P(22) = 1002$

$D^2(23) = P(21) - D^1(21)$
$P(21) = 792$
$D^1(21) = P(20) = 627$
$D^2(23) = 165$
$D^3(23) = P(20) - D^1(20) - D^2(20)$
$D^1(20) = P(19) = 490$
$D^2(20) = P(18) - D^1(18)$
$P(18) = 385$
$D^1(18) = P(17) = 297$
$D^2(20) = 88$
$D^3(23) = 627 - 490 - 88 = 49$
$D^4(27) = 1255 - 1002 - 165 - 49 = 39.$

7.4 Stirling, Bell, Catalan, and Bernoulli Numbers

In this section we briefly present some important classes of numbers related to fundamental enumeration formulae (see [218] for more details).

7.4.1 Stirling and Bell Numbers

Now, let us consider the number of allocations of n distinguishable objects to k undistinguishable non-empty cells. For counting these combinatorial schemata, we introduce this definition, by induction, of Stirling numbers of the second kind, denoted by:

$$\left\{ \begin{matrix} n \\ k \end{matrix} \right\}.$$

Stirling numbers of the first kind, we only mention here, are denoted by:

$$\left[\begin{matrix} n \\ k \end{matrix} \right]$$

and count the number of permutations of n objects having exactly k cycles (a cycle $(abc \ldots m)$ is a permutation where a goes to b, b goes to c …and so on, and ends with m going to a).

Stirling numbers (of the second kind) satisfy the following equations:

$$\left\{ \begin{matrix} n \\ n \end{matrix} \right\} = \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} = 1$$

$$\left\{ \begin{matrix} n+1 \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n \\ k-1 \end{matrix} \right\}.$$

In fact, when allocating $n+1$ objects into k undistinguishable cells, firstly we can remove one object from the given object. The remaining objects can be allocated to k cells in two possible manners: i) either allocating the remaining objects to k cells in $\left\{ \begin{matrix} n \\ k \end{matrix} \right\}$ ways, and then adding the removed object to one of the k cells, in k different ways, or ii) allocating the remaining objects to $k-1$ cells, leaving empty one cell, and putting the removed object by itself in the empty cell.

By using Stirling numbers, we can also obtain the number of surjective functions $S(n,k)$ from a set of n elements to a set of k elements ($k \leq n$). In fact, these functions correspond to the allocations of n objects to k distinguishable cells. Therefore, all ways of distinguishing k cells, correspond to giving them $k!$ different orderings, which yield the following cardinality of $S(n,k)$:

$$|S(n,k)| = k! \left\{ \begin{matrix} n \\ k \end{matrix} \right\}.$$

The number of all possible partitions of n objects into any number of non-empty indiscernible cells is given by the numbers b_n, called Bell numbers, after the American mathematician who studied them:

$$b_n = \sum_{i=1,n} \left\{ \begin{matrix} n \\ i \end{matrix} \right\}.$$

These numbers correspond to the different ways of arranging the first n non-zero numbers into *monotonic* sequences, that is, sequences where the first occurrence of a number cannot follow that of a number greater than it. For example 1 1 2 3 3 4 5 is monotonic, while 1 1 3 2 3 4 5 is not monotonic. In this way we get a partition of the positions of the sequence into undistinguishable non-empty cells (the different numbers occurring in the sequence).

7.4.2 Catalan Numbers

Catalan numbers count the possible ways of writing correct expressions of n pairs of parentheses.

Expression () is correct. If E is a correct expression, then also (E) is correct, and if E_1 and E_2 are correct, then E_1E_2 is correct too. This number corresponds to the different ways $2n$ persons at a table can shake hands by avoiding crossing, or the ways we can connect the two extremes of a diagonal in a square grid, by means of paths consisting of horizontal and vertical segments on the grid. These numbers also equal to the numbers of sequences of n occurrences of 1 and n occurrences of -1, such that the partial sums are never negative (from the beginning to any other position). Let us call these sequences *positive sequences* over $\{1,-1\}$.

Theorem 7.10. *The number of positive sequences over* $\{1,-1\}$ *of length* $2n$ *is given by the n-th Catalan number:*

$$\frac{1}{n+1}\binom{2n}{n}$$

Proof. The sequences S over $\{1,-1\}$ of length $2n$ are $\binom{2n}{n}$. Let us partition the sequences of S in the classes $S_0, S_1, S_2, \ldots, S_n$, where S_i, for $0 \leq i \leq n$, consists of the sequences having i as maximum negative value of their partial sums. The sequences that we want to count coincide with those of class S_0. These classes have the same number of elements. In fact, we can transform, by means of a one-to-one correspondence, the class S_{i+1} in the class S_i, for $0 \leq i \leq n$. This transformation, given a sequence of S_{i+1}, interchanges the value -1 of the position where the partial sum $-(i+1)$ is reached with the value 1 of its next position (this value is necessarily 1). This means that $|S_{i+1}| = |S_i|$, for $0 \leq i \leq n$. Therefore, $|S_i| = |S|/(n+1)$, for $0 \leq i \leq n$, and in particular, for the set S_0 whose cardinality we want to count, we have that $|S_0| = |S|/(n+1)$, that is, the number of our sequences is that one asserted by the theorem. □

7.4.3 Bernoulli Numbers

Jakob Bernoulli developed in his *Ars Conjectandi* a generalization of Faulhaber's method for computing the sums of powers. He introduced an interesting sequence of numbers, $B^{(i)}$ with $i > 0$, which now are called Bernoulli numbers. He reported proudly of having computed the sum of the first thousand tenth powers "intra semi-quadrantem horae" (in seven minutes and a half).

Bernoulli numbers can be defined, by induction, in the following way.

Initial Step. Let us consider the following equation:

$$B^2 - 2B^1 + 1 = B^2$$

that becomes:

$$2B^1 = 1$$

by obtaining $B^1 = 1/2$. This value is the first Bernoulli number $B^{(1)}$.

Induction Step. For $i > 1$, consider the equation:

$$(B-1)^{i+1} = B^{i+1}$$

replace in it, for $j < i$, powers B^j with the Bernoulli numbers $B^{(j)}$, computed at the previous steps (power B^{i+1} disappears). The value of B^i resulting from the obtained equation is the Bernoulli number $B^{(i)}$.

Bernoulli proved the following theorem (see [218]).

Theorem 7.11 (Bernoulli's formula for the sums of powers). *The sum of powers $\sum_{i=1,n} i^{k-1}$ is given by the following equation, where double curly brackets mean that, after applying Newton's binomial formula, the powers B^i have to be replaced with the Bernoulli numbers $B^{(i)}$.*

$$\sum_{i=1,n} i^{k-1} = \frac{\{\{(n+B)^k - B^k\}\}}{k}.$$

7.5 Stirling Approximation

The approximation of $n!$ which we prove in this section, due to Stirling, is a key point in many contexts of discrete mathematics.

Theorem 7.12 (Stirling Approximation).

$$n! \to_{n\to\infty} \sqrt{2\pi n}(n/e)^n$$

Proof. Let us consider the ratio $n^n/n!$, then

$$n^n/n! = \frac{n}{n} \cdot \frac{n}{n-1} \cdot \frac{n}{n-2} \cdot \ldots \frac{n}{1}.$$

Of course:

$$\frac{n}{n-i} = \frac{n}{n-1} \cdot \frac{n-1}{n-2} \cdot \frac{n-2}{n-3} \cdot \frac{n-i+1}{n-i}.$$

If we replace its left member with the right one, in the initial equation, then we get:

$$n^n/n! = \left(\frac{n}{n-1}\right)\left[\left(\frac{n}{n-1}\right)\left(\frac{n-1}{n-2}\right)\right]\left[\left(\frac{n}{n-1}\right)\left(\frac{n-1}{n-2}\right)\left(\frac{n-2}{n-3}\right)\right]\ldots$$

whence, by grouping in decreasing order:

$$n^n/n! = \left(\frac{n}{n-1}\right)^{n-1}\left(\frac{n-1}{n-2}\right)^{n-2}\left(\frac{n-2}{n-3}\right)^{n-3}\ldots$$

that is:

$$n^n/n! = \prod_{i=1}^{n-1}\left(1+\frac{1}{n-i}\right)^{n-i}$$

where each factor seems to go to Euler constant e when n goes to infinity, by suggesting this (wrong) equation:

$$n^n \approx n! e^{n-1} \quad \textbf{(wrong)}$$

This evaluation is wrong because these factors do not tend to e simultaneously. In fact, when the first one is near to e, the last one has to start its approximation. In order to get a better evaluation we define the following term $b(n)$ by trying to evaluate its asymptotic behavior:

$$e^n n!/n^n = b(n).$$

The evaluation of $b(n)$ is strictly related to the following Wallis' product for $\pi/2$ (a consequence of *Euler's jewel* given in Sect. 5.4.3):

$$\pi/2 = \prod_{n>0} \frac{4n^2}{4n^2-1} = \lim_{n\to\infty} \prod_{i=1,n} \frac{4i^2}{4i^2-1} \tag{7.5}$$

$$\frac{\pi}{2} = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7} \;\cdots\; \cdot\frac{2n}{2n-1}\cdot\frac{2n}{2n+1}\;\cdots \tag{7.6}$$

if we denote by $(2n)!!$ the product of the even numbers equal to or less than $2n$, we get:

$$(2n)!! = 2^n \cdot n! \tag{7.7}$$

moreover, if we denote by $(2n+1)!!$ and by $(2n-1)!!$ the product of the odd numbers equal to or less than $2n+1$ and $2n-1$ respectively, we get:

$$(2n-1)!! = (2n)!/(2^n \cdot n!) \tag{7.8}$$

$$(2n+1)!! = (2n+1)\cdot(2n)!/(2^n \cdot n!) \tag{7.9}$$

therefore, Eq. (7.6) becomes:

$$\pi/2 \rightarrow_{n\to\infty} \frac{(2n)!!\cdot(2n)!!}{(2n+1)!!\cdot(2n-1)!!}$$

whence, according to Eqs. (7.7), (7.8), and (7.9), it follows that:

$$\pi/2 \rightarrow_{n\to\infty} \frac{2^n n!\cdot 2^n n!}{(2n+1)\cdot(2n)!/(2^n\cdot n!)\cdot(2n)!/(2^n\cdot n!)}$$

that is:

$$\frac{\pi}{2} \rightarrow_{n\to\infty} = \frac{2^{4n}(n!)^4}{((2n)!)^2(2n+1)}.$$

Now, let us consider the terms $(b(n))^2$, $b(2n)$ and evaluate the ratio $(b(n))^2/b(2n)$. We have:

$$(b(n))^2 = \frac{(n!)^2 e^{2n}}{n^{2n}}$$

$$b(2n) = \frac{(2n)!e^{2n}}{(2n)^{2n}}$$

$$\frac{(b(n))^2}{b(2n)} = \frac{(n!)^2 e^{2n}}{n^{2n}}\frac{(2n)^{2n}}{(2n)!e^{2n}} = \frac{(n!)^2}{n^{2n}}\frac{2^{2n}n^{2n}}{(2n)!} = \frac{2^{2n}(n!)^2}{(2n)!}$$

which, by approximating $(2n+1)^{1/2}$ by $(2n)^{1/2}$, becomes:

$$\frac{(b(n))^2}{b(2n)} = (2n+1)^{1/2}(\pi/2)^{1/2} \rightarrow_{n\to\infty} \sqrt{\pi n}.$$

The evaluation of this ratio allows us the evaluation of $b(n)$ as n goes to infinity. In fact, $b(n)$ has to be a sub-linear function of n, because otherwise the ratio $\frac{(b(n))^2}{b(2n)}$ could not be of type $c\sqrt{n}$. Therefore, $b(2n) = qb(n)$, with $q < 2$, and the following equation holds:

$$\frac{(b(n))^2}{b(2n)} = \frac{b(n)}{q}.$$

Now, if the above ratios have to equal $\sqrt{\pi n}$, then $b(n)$ has to be asymptotically $k\sqrt{n}$ for some suitable k. If we replace this value in the above equation, then we get:

$$\frac{(b(n))^2}{b(2n)} = \frac{k^2 n}{k\sqrt{2n}} \rightarrow_{n\to\infty} \sqrt{\pi n}$$

therefore

$$\frac{k\sqrt{n}}{\sqrt{2}} \rightarrow_{n\to\infty} \sqrt{\pi n}$$

which means that $k = \sqrt{2\pi}$ and $b(n) \rightarrow_{n\to\infty} \sqrt{2\pi n}$. This concludes the proof of Stirling's approximation (more precise forms of this approximation can be found in Feller's book [221]). $\square$

7.6 "Ars Conjectandi" and Statistical Tests

A **statistical distribution** on a population of objects is obtained by associating to each of them a numerical value. These values over the population naturally provide a population of numbers, for which many important concepts can be defined.

For example, let us consider a text seen as a multiset of words (ignoring their order), if we assign to each word its length, we get a statistical distribution (of word lengths) taking values in the positive integers.

Given a statistical distribution X, the **frequency** $\nu(x)$ of an element $x \in X$ is the ratio between the multiplicity of x in X and the size of X. The **range** of X is the interval between the **minimum** and the **maximum** values occurring in X. The **majority** and the **minority** of X are the maximum multiplicity and minimum multiplicity of X, respectively. The **mode** is the value (or the values) having the majority as multiplicity, while the **median** is the value in the middle position, when values of X are arranged in increasing order (if the elements of X are an even number, either the last value of the first half, or the first one of the second half is chosen as median).

A **discrete statistical distribution** or *interval statistical distribution*, is a statistical distribution where the range of the distribution is partitioned into disjoint intervals (usually having the same length) and only one value for each interval occurs in the distribution, for example, the middle point of each interval.

Given a statistical distribution $X = (x_1, x_2, \ldots, x_n)$, its **mean** $\mu(X)$ is given by:

$$\mu(X) = \frac{x_1 + x_2 + \ldots + x_n}{n} = \sum_{x \in X} \nu(x)x. \tag{7.10}$$

The **mean square deviation** $\sigma^2(X,x_0)$ of X with respect to a *reference value* $x_0 \in \mathbb{R}$ is a sum, for x varying in X, given by:

$$\sigma^2(X,x_0) = \frac{1}{n}\sum_{x\in X}(x-x_0)^2 = \sum_{x\in X} \nu(x)(x-x_0)^2. \tag{7.11}$$

The **variance** of X, $\sigma^2(X)$ is the **mean square deviation** of X with respect to the mean $\mu(X)$:

$$\sigma^2(X) = \sigma^2(X,\mu(X)). \tag{7.12}$$

It can be shown that variance reaches the minimum value among all possible mean square deviations.

The **standard deviation** $\sigma(X)$ of a statistical distribution X is the square root of variance:

$$\sigma(X) = \sqrt{\sigma^2(X)}. \tag{7.13}$$

A fundamental (elementary) result, due to the Russian mathematician Chebichev, states that in any statistical distribution X of mean μ and standard deviation σ, for any positive real k, the fraction of the values of X which, in absolute value, differ from μ more than or exactly equal to $k\sigma$ is a fraction less than or equal to $1/k^2$. Formally, setting $(x \geq y) = 1$ if the $\geq$ relation holds between x, y and zero otherwise, we have:

$$\frac{\sum_{x\in X}(x-\mu \geq k\sigma)}{|X|} \leq 1/k^2. \tag{7.14}$$

This means that, using $k = \sqrt{2}$, at least half of the values lie in the interval $\mu - 2\sigma$ and $\mu + 2\sigma$.

The **entropy** $H(X)$ of a statistical distribution X is a sum, for x varying in X, given by:

$$H(X) = -\sum_{x\in X} \nu(x)\log \nu(x). \tag{7.15}$$

This value, which is the basis of Shannon’s information theory, corresponds to a measure of the *disorder* of a population X.

7.6.1 Statistics and Probability

A **sample** of a population X is a subpopulation of X, that is, a multiset where any element occurs with a multiplicity less or equal to its multiplicity in X. A sample of X is **typical** if the frequency of any element of the sample is the same frequency that it has in X. **Statistics** is the science of discovering properties of populations from its samples.

In statistical distribution the notion of multiplicity, which is the crucial concept of multisets, provides naturally the notion of frequency, which can be seen as a sort of relative multiplicity. Frequencies measure the occurrence of values by means of fractions, and the sum of all frequencies of a statistical distribution is equal to 1.

This notion is naturally extended to that of **probability measure**. Given a set of elements, usually called **events** (and the set is a *space of events*) we may assign to each of them a probability which determines the possibility of their *appearance* with respect to a given (observation) context. An axiomatic approach, establishing natural conditions which a probability measure has to satisfy, has been developed in the last century by Kolmogorov. However, it is easy to realize that probability theory is the natural mathematical setting for studying statistical phenomena. Many important statistical parameters obey specific laws expressed by means of concepts developed in the theory of probability. The basic concepts of probability theory are those of **random variable** and **probability distribution**. A real random variable is a variable ranging on real numbers that assigns to each real interval I a measure, less than or equal to 1, to the event that the variable takes values inside that interval. The theory of probability is a discipline which emerged in the modern age, with some anticipations by the Italian mathematician Girolamo Cardano (1501–1576) (*Liber de ludo aleae*, that is, The book about the dice game), Galileo (1564–1642) (*Sopra le scoperte dei dadi*, that is, About discoveries on dice), and Christian Huygens (1629–1695) (*De ratiociniis in ludo aleae*, that is, The rules of dice game). The first main intuitions were developed by Pascal (1623–1662) and Fermat (1601–1665), but a systematic treatise on this subject was the book on *Ars Conjectandi*, written by Jacob Bernoulli (1654 –1705), who used binomial coefficients in the distribution of boolean variables, and found the law of large numbers (frequencies approximate to probabilities for large populations), called *Theorema Aureus*. The idea of assigning degrees of possibility to events, is an aspect of great innovation with respect to the classical mathematics, and implicitly, it replaces facts *in suppositione objecti* (farts as they are) with that of facts in *suppositione subjecti* (facts as they are judged). Namely, reality is not what it is, but what an observer judges it could be, according to a percentage of "actuality", in the context of a potential space of events.

The most important discoveries in probability theory, after Bernoulli's work, were developed since the 17th and18th centuries, starting with De Moivre (1667–1754), who found the normal distribution as a curve for approximating large binomial coefficients. Bayes (1702 –1752) discovered the theorem named by his name (independently discovered also by Laplace), concerning the *inversion of conditional probabilities*. Laplace (1749 –1827) extended the De Moivre's result about normal distribution. Finally, Gauss (1777–1855) recognized the normal distribution as the law of casual error distribution, and Poisson (1781–1840) introduced the distribution named by his name as the law of rare events.

A space of events is a special boolean algebra (with 0, 1, sum, product, and negation, analogous to the operations $\neg, \wedge, \vee$ over propositions). Moreover, a probability measure can be easily assigned to a proposition, by measuring in some way, with numbers in [0,1], the class of models where the proposition holds. However, an important remark about probability is the distinction of two different aspects: i) the definition of the space of events which is more appropriate in a given context, and, ii) the application of probabilistic methods for analyzing and deducing information within a given event space. The methods of the theory of probability constitute a conceptual framework which is, for many aspects, independent from the specific space of events. A comparison may help to distinguish these two levels. The calculus and the theory of differential equations provide rules and methods for solving and analyzing differential equations, but the choice of the right equation which fits the best description of a physical phenomenon is a different thing, which pertains to the ability of modeling correctly phenomena of a certain type.

The axiomatic approach in probability theory was initiated by Kolmogorov and was developed within the Russian mathematical tradition. It plays a role which is comparable to the axiomatic approach in geometry, and it is important for understanding the logical basis of probability. However, it cannot replace the intuition and the analysis of adequacy of probabilistic models with respect to the realities they apply to. Logical analyses of probability are related to inductive logics and to many non-classical methods of logical inference (Carnap and Hintikka are two logicians of the last century who developed theories in this direction). From the mathematical point of view, probability theory is part of a general field of mathematics, referred to as *Measure theory*, initiated by French mathematicians of the last century. An important exponent of this school, Emil Borel, proved the following result about real numbers, which is related to the intrinsic random character of real numbers: *Almost all numbers, when expressed in any base, contain every possible digit or possible string of digits*. Here, *almost all* means that the set of numbers without this property are a set having null measure (a non-empty set may have null measure).

An event can be expressed by means of a proposition asserting that the value of a variable X belongs to a subset of its range, denoted by range(X). This means that the usual propositional operations $\neg, \wedge, \vee$ are defined on events. The **conditional probability** of an event A, given an event B, is denoted by $P(A|B)$. It expresses a sort of *implicational* probability, or the probability of A, under the assumption that event B has occurred. Formally:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}.$$

Events A and B are said to be **independent**, and we write $A||B$, if $P(A|B) = P(A)$. Events A and B are **disjoint** if $P(A \wedge B) = 0$. The following rules connect propositional operations to probabilities . Proposition $\neg A$ has to be considered in terms of complementary set, that is, if $A = (X \in S)$, then $\neg A = (X \in \text{range}(X) - S)$.

1. $P(A) \geq 0$
2. $P(A \vee \neg A) = 1$
3. $P(A \wedge \neg A) = 0$
4. $P(\neg A) = 1 - P(A)$
5. $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
6. $P(A|B) = P(A \wedge B)/P(B)$
7. $A||B \Leftrightarrow P(A \wedge B) = P(A)P(B)$.

The theory of probability is the field were even professional mathematicians can be easily wrong, and very often reasoning under probabilistic hypotheses is very slippery. Let us report some examples, from [215], which can help to grasp main subtle points about probability. Assume that a pilot has a 2% chance of being shot down in a military mission. It seems quite natural to conclude that in 50 missions we can guess that he is almost sure to die. But this is completely wrong. In fact, let us apply the rules above, then the probability of surviving is 0.98^{50}, assuming that each mission is independent from the others. Therefore the probability of dying is $1 - 0.98^{50} = 0.64$, which is markedly different from 1. The same argument shows that the probability of obtaining one head, by tossing a coin twice, is not 1.

This kind of error was in the problem posed by the *Chevalier de Méré* to Pascal, about a popular dice game: *Why the probability of one ace, rolling one die 4 times, is greater than that of both aces, rolling two dice 24 times?* In fact, one could led to reason in the following way. The probability of one ace is $1/6$, and $4/6 = 2/3$, analogously the probability of 2 aces is $1/36$, and $24/36 = 2/3$, by concluding that in principle the two events: "1 ace in 4 rolls", and "2 aces in 24 double rolls" are equiprobable. But the empirical evidence reported by Chevalier de Méré was against this wrong conclusion. Namely, Pascal (interacting with Fermat on this puzzle) solved the apparent paradox by the following computation. In the first game, P(no-ace-in-4-tolls) = $(5/6)^4$, therefore P(ace-in-4-rolls)= $1 - (5/6)^4 = 0.5177$. In the second game, P(no-2-aces-in-24-double-rolls)= $(35/36)^{24}$, therefore P(2-aces-in-24-double-rolls) = $1 - (35/36)^{24} = 0.4914$. A detailed analysis shows that the simple mistake in the argument suggesting the same probability for the two events is due to the violation of rule 5 given above.

Sometimes difficulties arise in choosing the right event space. The problem which was posed to Galileo by the archduke of Tuscany was the following: *Why rolling three dice the value 10 is more frequent than 9?*. His question was motivated by the fact that there are 6 ways for decomposing 10 into three parts: (6+3+1), (6+2+2), ((5+4+1), (5+3+2), (4+4+2), (4+3+3), but even 6 ways of decomposing 9: (6+2+1), (5+3+1), ((5+2+2), (4+4+1), (4+3+2), (3+3+3). The answer Galileo found (in modern language) is that the space of events is not given by the partitions of integers in three parts, but by the possible outcomes of the three dice, and it is easy to realize that there are 27 ways of obtaining 10 with three dice, while the 25 ways of obtaining 9 with three tolls (for example (3+3+3) is realized in only one way, but (6+3+1) can be realized in a number of ways corresponding to the 6 permutations of three objects.

Another interesting case of confusion is related with conditional probability. Bayes' theorem, in a simplified form, asserts the following equation:

$$P(A|B) = P(A)P(B|A)/P(B).$$

Its proof is very simple. In fact, by the definition of conditional probability we have:

$$P(A|B) = P(A \wedge B)/P(B)$$

but

$$P(A \wedge B) = P(B \wedge A) = P(B|A)P(A)$$

therefore, the statement claimed by the theorem follows easily from the above two equations.

Despite the simplicity of this proof, what the theorem asserts is an inversion of conditions. In fact, $P(A|B)$ can be computed by means of the inverse conditional probability $P(B|A)$. This is a very subtle point. Assume that a test T for a disease D is wrong only in one out to 1000 cases. Assume that a person is positive to this test. What is the probability of being affected by D? Is it 0.999? Fortunately it is not the case. In fact, this deduction confuses $P(D|T)$ (the probability of the disease D, when the test T is positive) with $P(T|D)$ (the probability test T positivity, when the disease D is present). The right way to reason is the application of Bayes' theorem. For this, assume to know the simple probabilities of D and T. D affects one out of 10000 persons, and T provides positivity one out of 1000 cases. Let us assume also that T is very reliable, with $P(T|D) = 1$, that is, if D is present, then T is positive with probability 1. This means that in 10000 persons we have 11 positive cases: 10 because T is wrong 1 to 1000) plus 1, because 1 to 10000 has the disease D and T discovers it. In conclusion, the probability of having D is less than 10% . In fact:

$$P(D|T) = P(D)P(T|D)/P(T) = 1/10000 \times 1/(11/10000) = 1/11.$$

A famous debate about probability is the *three doors quiz* also known as the *Monty Hall problem*, by the name of the TV conductor in a popular American show. There are three doors A, B, and C. A treasure is behind only one of them, and you are asked to guess which is the lucky door. Let us assume that you choose the door C. Then, another chance is given to you. In fact, they open the door B, showing that it is not the fortunate door, and you are free of confirming your preceding choice C, or changing by guessing A. The question is: *What is the more rational choice? Confirming C or changing it with A?* Many famous mathematicians concluded that there is no reason for changing (Paul Erdös, the number one of 20th mathematicians was among them). However, by using Bayes' theorem it can be shown that changing the door is the better choice, because it provides a passage from a probability $1/3$ of success to a probability of $2/3$. This fact was even confirmed by computer simulations of the game. We do not present the detailed analysis based on Bayes' theorem, but we give an intuitive, very convincing reason. If the doors are 100: $D_1, D_2, \dots D_{100}$ and after you guess D_{100}, the doors $D_2, D_3, \dots$ are shown to be unlucky, are you sure to confirm D_{100}, or rather, are you inclined to believe that is wiser change your initial choice? In fact, what is crucial in this game is that opening doors change the the space of events.

7.6.2 *Statistical Inference*

The simplest probabilistic schema is an urn of n balls, where $m < n$ are white and $n - m$ are black. This model, due to Jakob Bernoulli, assumes a boolean random variable (the color of the extracted ball), but it is of crucial importance in analyzing and developing general probabilistic concepts. Moreover, it is surprising that

Laplace, the mathematician who is the symbol of deterministic perspective in science, is also the same scientist who grasped the importance and generality of the normal distribution, as the most extraordinary law of chance. At a first glance, this seems to be a paradox. In fact, if chance is the absence of rules, how is it possible to discover a law for phenomena which are outside of any law? Maybe, just this apparent paradox can provide a sort of reconciliation between determinism and chance. In fact, a way of considering the laws of chance, is that of rules for passing from the nondeterminism, at level of individuals, to a collective determinism. Namely, chance is not the pure absence of causes, but rather it is a population of (small) causes such that we cannot distinguish them singularly [230], but in their collective combination. Probability enlarged the power of mathematical analyses and explanations, by replacing the rigid Newtonian model of an exact predictability with a conceptual framework where complex phenomena going beyond any kind of Newtonian explanation can be mathematically framed for discovering mutual influence and causal relations. For this reason, presently, probabilistic and statistic methods are applied everywhere. The discovery, in the second part of the last century, of deterministic chaos reinforces the limitations of a rigid determinism in science. In fact, even in simple systems, under suitable conditions, it happens that there is no way of predicting the behavior of the system even if it is completely deterministic, because the system is so sensible to the determination of its initial state that only an (impossible) infinite precision could avoid an error of its future states which will exponentially amplify with time.

Since the 19th century, statistics begins to be recognized as a discipline, firstly related to the quantitative analysis of population phenomena interesting for sociological and economical analyses (birth and death rates, richness distribution, migration fluxes, and so on). Adolphe Quételet (1796–1874) played a crucial role in the connection between statistics and probability. He tried to apply the Laplace discovery about the central role of normal distribution in the context of sociological disciplines. His program was very ambitious (defining a quantitative analysis of sociology based on normal distribution), but his ideas had a strong influence through the publication of a volume of Henry Thomas Buckle, who reported, in detailed terms, the theories of Quételet. This volume had great success, and it was certainly read by Darwin, Maxwell, and Boltzmann, suggesting to them the power of statistics in the analysis of natural phenomena. If the theory of probability, starting from a given space of events, provides methods for assigning probabilities to complex events, statistics plays the inverse game, because it provides methods for inferring the right probabilities which underlie some statistical observation. This idea had been emerging since Laplace, but became a crucial aspect of the statistical studies of 20th century when mathematically advanced methods were developed, in two complementary directions: i) inferring probabilities from statistical data, ii) discovering the probabilistic laws of statistical parameters. This second trend provided one of the most powerful tools in the causal analysis of population phenomena, namely, the methods for testing hypotheses about cause/effect relations. Galton (Darwin's cousin), who tried to apply Quételet's approach to biology, introduced the important notion of *correlation*, better formalized by Karl Pearson [229], who discovered the

χ^2 test, Gosset discovered the Student’s test (Student is Gosset’s pseudonym), and Fisher the F test. Fisher, who was a great mathematical statistician [222], was also a great geneticist, he gave a statistical interpretation to natural selection, by means of the *Natural selection theorem* showing that, under suitable hypotheses, the force driving evolution pushes a biological population toward the maximum of possible fitness (the maximum possible demographic rate reachable in a given context).

The use of correlation indexes and of tests of significance, according to specific methods, provide effective tools in **statistical regression** techniques, which, in very general terms, are aimed at finding the best analytical forms which underly time series of observed data, an important field of statistical application in the same line of Gauss’ methods of minimum least squares (by means of which Gauss solved an astronomical puzzle by determining the orbit of the asteroid Ceres).

It is reported [228] that Henry Poincaré proved the dishonesty of a baker, by discovering that the weights of the daily loaves of bread fluctuated without following the gaussian distribution (clearly, a guilty defect with respect to the declared weight). This is a simple example of using a law of chance for finding the existence of a non-accidental motivation providing an observed effect. In other words, Poincaré’s experiment answered the question: *Is there some cause which is not due only to chance, affecting the weights of loaves of bread?*.

Hypothesis testing is largely the product of Ronald Fisher, Jerzy Neyman, Karl Pearson, and Egon Pearson (Karl’s son), developed in the early 20th century. Fisher coined the term **tests of significance** for denoting the essence of hypothesis testing. In general, such tests provide tools for evaluating data coming from sampling populations or from analytical models for comparing them, in order to discover whether data of distinct sources significantly differ with respect to casual differences. The distinction between casual and causal differences is the basis for: i) recognizing the existence of cause providing some observed effects; ii) establishing that some observed effects can be associated to a given cause acting on individuals of a population, or iii) evaluating if a model explains some observed data, apart from some approximation errors of casual nature.

For example, in a population of people affected by a disease, does the treatment with a drug have a positive effect against some pathology? Let us assume that we want to observe the possible effects of a specific drug on the blood pressure in persons affected by blood hypertension. Comparing two samples, one of which was exposed to the drug treatment, we would deduce whether the effects we observe could be associated to the treatment or are simply casual fluctuations.

The basic approach of a statistical test consists in the comparison of the casual hypothesis H_0, claiming that the observed effects are casual, against the opposite hypothesis H_1, according to which the observed effects are due to a phenomenon acting on the population. In this way then we can conclude, on a statistical basis, in favor of a proved effect caused by the phenomenon in question. If the test concerns only a parameter of a population, and no comparison is performed with a population of reference, we can only conclude that a cause is acting on the parameter, apart from the influence of chance (the nature of this cause has to be deduced by using other kinds of information, as in the case of Poincaré’s experiment).

The general schema of performing such a test is given by the following procedure:

1. The measure of a statistical parameter Q in two populations (for example the mean blood pressure);
2. The determination of the probabilistic law F which the chosen parameter follows (F gives, for each possible value of the parameter Q, the probability of having samples where the parameter assumes that value);
3. The calculation of the value V of the chosen parameter, in the population where the cause under investigation is acting;
4. Definition of the confidence interval of the test, that is, the level of error probability E which is tolerated in the choice of the hypothesis (for example, 3% or 5%);
5. The localization of the observed value V in the distribution probability F;
6. The evaluation of the probability P_V which F assigns to the event $\{Q \geq V\}$ (or $\{|Q| \geq |V|\}$, in the case of a symmetric distribution);
7. The rejection of the null hypothesis if $P_V \leq E$, with the confidence E, or acceptance in the other case (with confidence E).

In the previous procedure, P_V is also called the **p-value** of V, while the value V_0 such that the probability of $\{Q \geq V_0\}$ is less than E, is called the **critical value** for

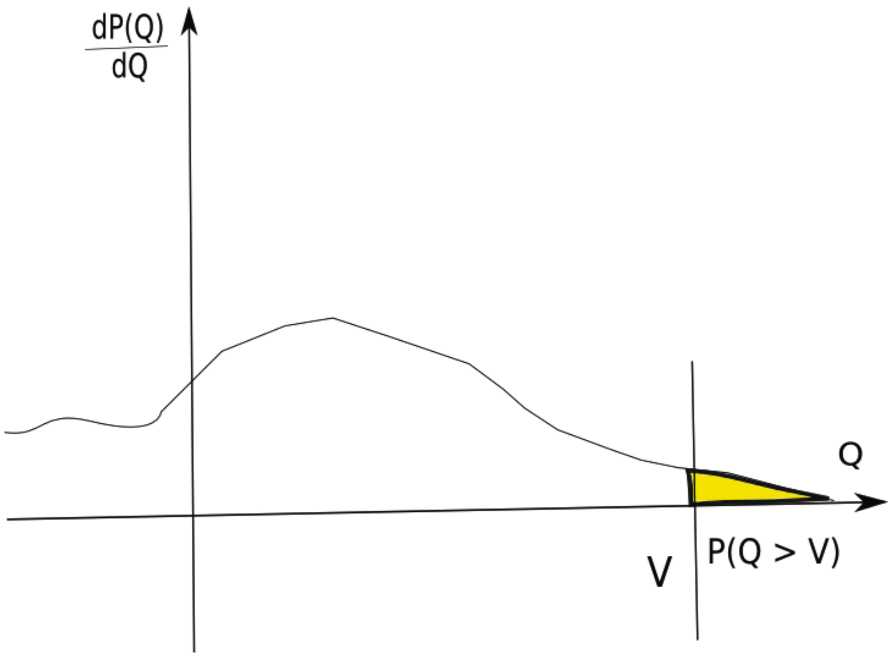


Fig. 7.5 Statistical inferential value of test parameters

the test (with respect to the confidence E). Equivalently, instead of considering the p-value, the null hypothesis can be rejected when $V \geq V_0$.

7.6.3 Sample Significance Indexes

Analyzing a population can be aimed at selecting individuals of the population having some property of interest. In the following we express some sampling situations in terms of clinical tests [223]. A population D is partitioned in two sub-populations D^+ and D^- of people having a disease, and people without the disease, respectively. The same population is selected in other two sub-populations T^+ and T^- of people resulting positive to a clinical test T and people resulting negative to the test, respectively. Some indexes can be defined for the test T, which characterize its capacity in discriminating correctly, a priori and a posteriori, with respect to disease D.

Table 7.2 defines four important indexes of a clinical test T for a disease D.

Table 7.2 Indexes of adequacy for clinical tests

Sensitivity	$\lvert T^+ \cap D^+ \rvert / \lvert T^+ \rvert$
Specificity	$\lvert T^- \cap D^- \rvert / \lvert T^- \rvert$
Positive predictability	$\lvert T^+ \cap D^+ \rvert / \lvert D^+ \rvert$
Negative predictability	$\lvert T^- \cap D^- \rvert / \lvert D^- \rvert$

We can interpret a test as a measure of reliability of statistical inference with respect to a property of having a pathology. Table 7.3 explains the meaning of statistical inferences.

Table 7.3 Statistical inferential value of test indexes

High value	Reliability
Sensitivity	$Positivity \Rightarrow Disease$
Specificity	$Negativity \Rightarrow Health$
Positive predictability	$Disease \Rightarrow Positivity$
Negative predictability	$Health \Rightarrow Negativity$

Given a population X, the population of the samples of X is a *second level* population which is very important in the statistical analysis of X. In fact, it provides some statistical distributions, called **sample distributions** of the samples taken from X. The power of statistics is based on the evidence that sample distributions follow precise mathematical laws. These are the basis for important procedures inferring properties of populations from their samples.

The **h-index** of a population X over a set A (also called *Hirsch index*, introduced in the context of bibliometrics), is the maximum number n of elements of A having in X a multiplicity greater than or equal to n. For example, the h-index of the population $2a+3b+4c+5d$ is 3, because the multiplicities of 3 of them, b,c,d, are equal to or greater than 3.

Let us consider the statistical distribution of values provided by a variable Z assuming values in correspondence to the individuals of a population. Let us order the values of this distribution in a decreasing order. In many real cases there is a law relating the ordering position of a value v with the percentage of elements x of the population for which $Z(x) \geq v$. For example, in the statistical distribution of the richness (expressed in money) of a population, it is very common that around the 80% of the total richness of the population is covered by around the 20% richest people (in the richness order). This is the famous **Pareto law** of 20/80 (from the Italian economist who discovered this regularity). Similar laws occur in many situations of resource distributions, and provide patterns of general relevance.

7.7 Least Square Approximation

The method of least squares was introduced by Carl Friedrich Gauss around 1794 as an approach to the approximate solution of overdetermined systems, that is, sets of equations with more equations than unknowns. The term “least squares” is used, because the overall solution minimizes the sum of the squares of the errors made with respect to an exact solution of the system.

A very elegant analysis of this method can be developed in terms of vector spaces [225]. In a vector space over the field $\mathbb{R}$ of real numbers an internal scalar product $(\ |\ )$ of vector pairs is defined such that $(v|w) \in \mathbb{R}$. In this case we say that the vector space is a **pre-Hilbert** real space. The scalar product of a pre-Hilbert real space defines a **norm** by $||v|| = \sqrt{(v|v)}$ and a notion of convergence with respect to this norm can be given which generalizes the convergence of real functions. A pre-Hilbert real space is **complete** if any Cauchy sequence of vectors converges to a vector in the space. A pre-Hilbert space which is complete is said to be a **Hilbert space**.

A very general and crucial result about pre-Hilbert spaces is the **projection theorem** given in Table 7.4. This theorem is the basis for the *Least Square Estimation* given in Table 7.5.

Table 7.4 Existence and unicity of the approximating vector with orthogonal error

The projection theorem. Let H a Hilbert space and M a closed subspace of H. There exists a unique vector $m_0 \in M$ such that $||x-m_0|| \leq ||x-m||$ for any pair $x \in H, m \in M$, and the error vector $e = x - m_0$ is orthogonal to M, that is, $(e \mid m) = 0$ for every $m \in M$.

Given n vectors $x_1, x_2, \ldots, x_n$ of a real Hilbert space H, we know that their linear combinations with real coefficients provide a closed subspace K (a subset of H closed under the vector operations) of H. Therefore the vector $v \in H$ closest to K has to minimize the norm

$$||v - c_1x_1 - c_2x_2 - \ldots - c_nx_n||.$$

According to the projection theorem, the unique minimizing vector $x_0 \in K$ has to be orthogonal to K. Therefore, for $i = 1, 2, \ldots, n$:

$$((v - c_1x_1 - c_2x_2 - \ldots - c_nx_n) \mid x_i) = 0.$$

This means:

$$\begin{array}{rcl} c_1(x_1 \mid x_1) + c_2(x_2 \mid x_1) + \ldots + c_n(x_n \mid x_1) & = & (v \mid x_1) \\ c_1(x_1 \mid x_2) + c_2(x_2 \mid x_2) + \ldots + c_n(x_n \mid x_2) & = & (v \mid x_2) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots & \ldots\ldots \\ c_1(x_1 \mid x_n) + c_2(x_2 \mid x_n) + \ldots + c_n(x_n \mid x_n) & = & (v \mid x_n) \end{array} \tag{7.16}$$

The matrix G which is transpose to the matrix of system 7.16 is called Gram matrix of the n vectors generating the vector space K.

Given a set of m equations with n unknowns, which expresses a linear combination of n linearly independent vectors of $\mathbb{R}^m$ for some (unknown) coefficients, there exists a unique n-vector of these coefficients providing the best approximation to a given m-vector v. The unicity of this n-vector, and its algebraic form, is determined by the theorem of Table 7.5, as a natural consequence of the projection theorem for Hilbert spaces, and the solution is expressed in terms of the Gram matrix of the vectors.

Table 7.5 The Least Square Evaluation of matrix W

Least-Square-Estimate. Let W a $m \times n$ matrix ($m > n$) with linearly independent column vectors and let Wz be the matrix product of W by $z \in \mathbb{R}^n$. Then, for any m-vector $v \in \mathbb{R}^m$, there exists a unique vector $z_0 \in \mathbb{R}^n$ minimizing $||v - Wz||$ (the Euclidean m-dimensional norm) over all $z \in \mathbb{R}^n$. Moreover, if W^T denotes the transpose of W, this vector z_0 is given by $z_0 = (W^TW)^{-1}W^Tv$.

The existence and unicity of vector z_0, as the Least-Square-Estimate claims (see 7.5), follows directly from the projection theorem. Moreover, the Gram matrix corresponding to the column vectors of matrix W is easily seen to be W^TW (see left sides of Eqs. (7.16)), and W^Tv corresponds to the right hand vector of equation system (7.16). Therefore, given the linear independence of the column vectors of W, the matrix W^TW is invertible, and the asserted value for z_0 can be computed.

7.7.1 The *k*-Variable Multiple Regression Model

In this section we will recall the classical *k-variable multiple regression*. The reader can find more details and statistical motivations in Aczel and Sounderpandian's book [216], from which we adopt the notation.

In statistics, the regression analysis provides techniques for finding the relationship between a *dependent variable* Y and one or more *independent variables* X_1, X_2, ..., X_k. The following equation is the general form of a linear regression:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon. \tag{7.17}$$

When $k > 1$ the regression equation above is called a **multiple regression model**. The correctness of the regression model is subjected to the following assumptions:

1. For each observation, the observation error ε is normally distributed with mean zero and standard deviation σ and is independent from the errors associated with all other observations;
2. The variables X_i are independent from the error ε.

Table 7.6 The three deviations associated with the data points (see also Fig. 7.6). We indicate with Y the value of the dependent variable, with $\widehat{Y}$ its predicted value by means of the multiple regression model, and with $\bar{Y}$ the average of the values of Y. Finally, we indicate with n the total number of data points used during the regression

$Y - \bar{Y}$	=	$Y - \widehat{Y}$	+	$\widehat{Y} - \bar{Y}$
Total deviation		Unexplained deviation (error)		Explained deviation (regression)
$\Sigma_{j=1}^{n}(y[j] - \bar{y})^2$	=	$\Sigma_{j=1}^{n}(y[j] - \widehat{y}[j])^2$	+	$\Sigma_{j=1}^{n}(\widehat{y}[j] - \bar{y})^2$
SST		SSE		SSR
Sum of Squared Total deviations		Sum of Squared Errors		Sum of Squared Regression deviations

If the assumptions given above are satisfied, then we can compute the coefficients $c_i, i = 0, \ldots, k$ in terms of least squares estimations of the regression parameters β_i, as the values providing the best approximation $\widehat{Y}$ to the value Y:

$$\widehat{Y} = c_0 + c_1 X_1 + c_2 X_2 + \ldots + c_k X_k \tag{7.18}$$

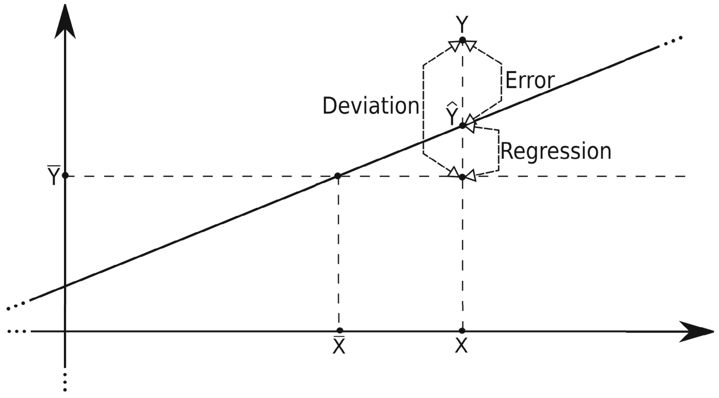


Fig. 7.6 Plot representing the three deviations introduced in Table 7.6 for a linear regression model

which are given by the Least Square approximation:

$$\begin{pmatrix} c_0 \\ c_1 \\ \dots \\ c_k \end{pmatrix} = \left(M^T \times M\right)^{-1} \times M^T \times \begin{pmatrix} Y[1] \\ Y[2] \\ \dots \\ Y[n] \end{pmatrix} \tag{7.19}$$

where M is the following matrix:

$$M = \begin{pmatrix} 1 & X_1[1] & X_2[1] & \dots & X_k[1] \\ 1 & X_1[2] & X_2[2] & \dots & X_k[2] \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_1[n] & X_2[n] & \dots & X_k[n] \end{pmatrix}.$$

The correctness of the estimated values, with respect to the real ones, depends on the amount of the unexplained deviation (i.e. the regression error) as indicated in Table 7.6 and displayed in Fig. 7.6.

The *Mean Square Error* is an unbiased estimator of the variance of the errors. It is given by the ratio of the sum of squared errors (SSE) with the number of the degrees of freedom associated to the regression model, that is, the number of data points minus the number of the regression coefficients used in the model (n is the number of data points used for the regression):

$$MSE = \frac{SSE}{n-(k+1)} = \frac{\sum_{j=1}^{n}(y[j]-\widehat{y}[j])^2}{n-(k+1)}. \tag{7.20}$$

We remark here that when we use n data points, a regression model which uses $n-1$ independent variables always reaches a perfect fit. However, when we do this we are *overfitting* our data, leaving no degree of freedom for errors. In this case, we will fit

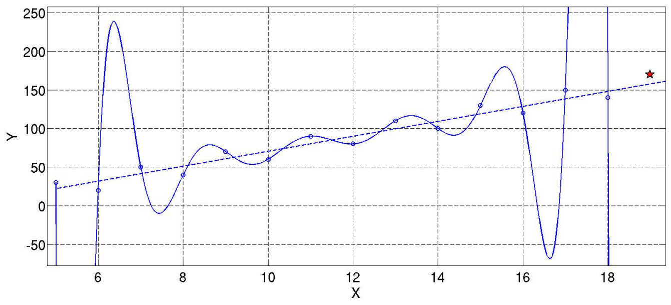


Fig. 7.7 Comparison between the predicting power of two regression models: a 13-degree polynomial $\widehat{Y} = c_0 + c_1X + c_2X^2 + \ldots + c_{13}X^{13}$ (depicted by the continuous line) and a least-square line (depicted by the dotted line). The dataset used to calculate the models is the 14 points depicted as circles, while the last point represented by a star is the value of Y we predict with our models. The 13-degree polynomial is a perfect example of model which overfits the data. Namely, it fails completely the prediction of Y in the 15th data point

perfectly Y in the n data points, but it will give a very poor prediction of Y in other points (see Fig. 7.7 for an example of overfitting).

The mean square error is a measure of the size of regression error, but does not give any indication about the explained component of the regression. The *multiple coefficient of determination* provides a measure of the capacity of a regression model in explaining the dependent variable (SSR, SST, SSE as in Table 7.6):

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}. \tag{7.21}$$

The value R^2 measures the part of the dependent variable variation which is explained by the combination of the independent variables in the multiple regression model. In other words, R^2 measures the percentage of Y explained by the regression model in terms of X_i variables (even if it is not relevant here, the notation R^2 is due to the fact that it corresponds to the square of another statistical index). However, R^2 has also a disadvantage, because it does not take into account the number of independent variables used in the regression model. For this reason, the *adjusted multiple coefficient of determination* is introduced ($MST = SST/(n-1)$):

$$\bar{R}^2 = 1 - \frac{SSE/[n-(k+1)]}{SST/(n-1)} = 1 - \frac{MSE}{MST} \tag{7.22}$$

which accounts the degrees of freedom of SSE and SST, by giving a sort of penalty to those models which fit the data well, but that are not parsimonious, as they use too many independent variables.

When we define a new regression model, we need to avoid the use of independent variables that are not related with the dependent variable Y. In other words, when we try to define a regression model, we have to answer to this basic question: *is* *there a linear regression relationship between the dependent variable Y and any of the independent variables X_i used by the regression equation?* The statistical test F will answer the question.

The following value:

$$F = \frac{SSR/k}{SSE/[n-(k+1)]} = \frac{MSR}{MSE} = \frac{R^2}{1-R^2} \cdot \frac{n-(k+1)}{k} \qquad (7.23)$$

provides the ratio between the variance explained by the regression model (given by the MSR) and the unexplained variance (given by the MSE). It is well-known in the literature that, under the hypothesis that F is a ratio of unrelated variances, its value follows an $F_{[k,n-k+1]}$-distribution (with k and $n-(k+1)$ degrees of freedom), a probability density distribution which takes the name of the English statistician Sir Ronald A. Fisher and is widely used in statistics to compare the distribution of two populations and to carry out the analysis of variance. In Fig. 7.8, the plot of an F-distribution with 5 and 14 degrees of freedom is depicted, the one which should be used, for example, when we want to test a regression model which fits $n = 20$ data points by means of $k = 5$ independent variables. After having fixed a **significance level** α for the test, which specifies the probability of error of the test, we can reject the null hypothesis H_0 when the value of the F-ratio of a given regression model is greater than a threshold value $F_{[\alpha;k,n-(k+1)]}$. This value divides the area of the F-distribution in two different parts of cumulative probabilities $1-\alpha$ and α, respectively. If we fix $\alpha = 0.05$, then we can conclude, with 5% probability of being wrong, that there exists a linear relationship between Y and any of the independent variables occurring in the regression equation, when the value of the F-ratio is greater than $F_{[\alpha;k,n-(k+1)]}$. In the case of considering the distribution $F_{(5,14)}$, this value is equal to 2.9582. The value of $F_{[\alpha;k,n-(k+1)]}$ is called the **critical value** of $F_{[k,n-(k+1)]}$ for α. In the literature, some tables are given which provide the right values of $F_{[\alpha;k,n-(k+1)]}$ for different values of α and of the degrees of freedom of the F-distribution.

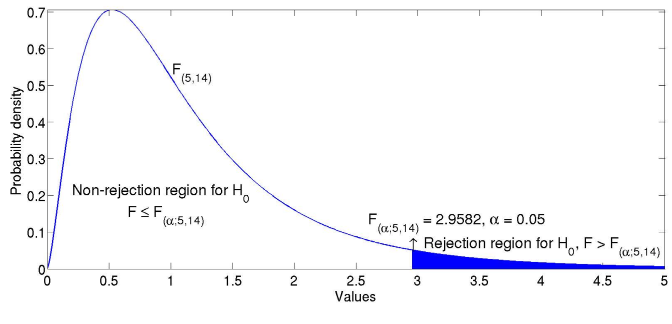


Fig. 7.8 The plot of the F-distribution $F_{(5,14)}$ with 5 and 14 degrees of freedom. The full right-tailed area of 0.05 is the one which represents the rejection region for the null hypothesis H_0 defined in Table 7.7

Table 7.7 A statistical hypothesis test for the existence of a linear relationship between Y and any of the X_i in the regression model of Eq. 7.17

H_0: $\beta_1 = \beta_2 = \beta_3 = \ldots = \beta_k = 0$
H_1: not all β_i $(i = 1, \ldots, k)$ are zero.

In conclusion, if the null hypothesis H_0 is true, no linear relationship exists between Y and any of the independent variables proposed in the regression equation. On the other hand, if we reject the null hypothesis, there is statistical evidence to conclude that a regression relationship exists between Y and at least one of the independent variables X_i, $i = 1, \ldots, k$ (see 7.7). The F test provides what is also called the **analysis of variance**, based on the localization of the F-ratio with respect to the F-distribution (with respect to a significance level).

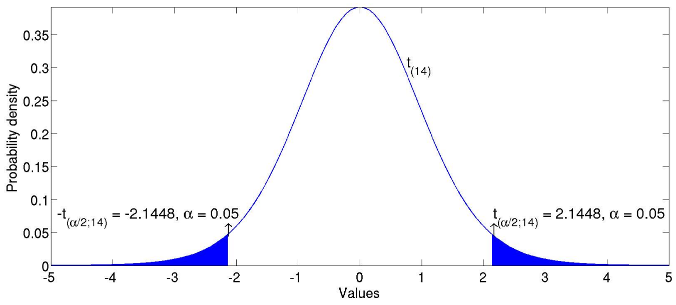


Fig. 7.9 The plot of a t-distribution with 14 degrees of freedom

Another probability distribution which is useful in the context of multiple regression is the *Student's t-distribution* (or simply the t-distribution), a continuous probability distribution that is the standard method for evaluating the confidence intervals for a mean when variance is unknown (assuming that population is normally distributed). Such a kind of distribution is used for calculating the **confidence intervals** for the least squares estimations of the regression coefficients calculated in (7.18). In statistics, a confidence interval provides the evaluation of the range, of an estimated parameter, where it is likely to find the correct value of the parameter. This evaluation is relative to a significance value α which gives the probability of being wrong in the confidence estimation. The $(1-\alpha)\%$ confidence interval for each β_i, $i = 0, \ldots, k$ in Eq. (7.17) is given by:

$$\beta_i = c_i \pm t_{[\alpha/2;n-(k+1)]}\sqrt{e_{ii} \cdot MSE}, \tag{7.24}$$

where $t_{[\alpha/2;n-(k+1)]}$ is the critical value of a t-distribution with $n-(k+1)$ degrees of freedom for $\alpha/2$, and e_{ii} is the element in position (i,i) of the matrix $(M^T \times M)^{-1}$ used in Eq. (7.19), which is the sum of the squares of X_i (the term $\sqrt{e_{ii} \cdot MSE}$ is the "scale factor" in the specific estimation of the confidence interval of c_i). In Fig. 7.9 the t-distribution $t_{(14)}$, with 14 degrees of freedom, is depicted with significance level $\alpha = 0.05$.

7.7.2 The Classical Stepwise Regression

In the previous section we have defined main concepts about multiple regression, by indicating some statistical tests which are used to understand the correctness of a given regression model. In this section, we address the problem of **variable selection**, that is, the problem of deciding which independent variables have to enter into a multiple regression model, among a given set of possible variables. The simplest method we can define consists of running all possible regressions, for all possible choices of independent variables, and then choosing the best model by selecting the one having the highest $\bar{R}^2$ or the lowest MSE. This brute-force method has the problem that it considers a number of models which increases exponentially with respect to the number of possible variables. In fact, the number of different models that we can define by means of k independent variables is 2^k.

The *stepwise regression* algorithm provides a method for variable selection which allows us to obtain good regression models with a lower complexity in time. This algorithm does not necessarily find the best model among all possible 2^k models, but it allows us to find a good model in a feasible time even when we need to consider a high number of independent variables. The method uses a statistical test, again based on F-distribution, which is called **partial F-test**, as it evaluates the relative significance of a subset of all possible variables.

Suppose that a regression model of Y with k independent variables is postulated:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon. \tag{7.25}$$

We will call this model the **full model** in the sense that it includes the maximal set of independent variables. Now, suppose that we want to test the relative significance of a subset of r of the k independent variables in the full model. The partial F-test provides a statistical criterion for evaluating if the full model given in Eq. (7.25) is better than the **reduced model** with only $k - r$ variables:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_{k-r} X_{k-r} + \varepsilon. \tag{7.26}$$

This corresponds to comparing the two hypotheses given in Table 7.8.

Table 7.8 Hypothesis of the partial F-test

H_0: $\beta_{k-r+1} = \beta_{k-r+2} = \ldots = \beta_k = 0$
H_1: $\beta_{k-r+1}, \beta_{k-r+2}, \ldots, \beta_k$ are not all zero.

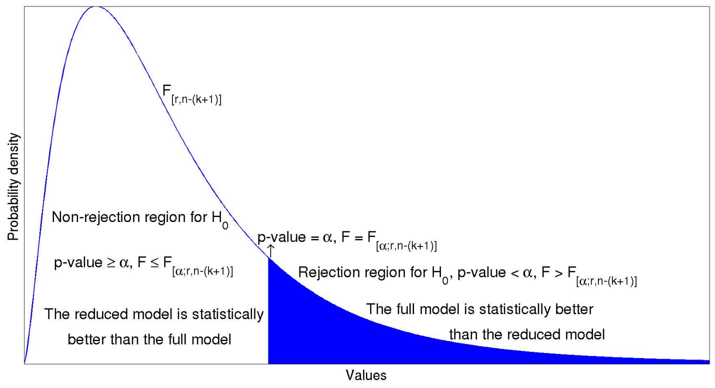


Fig. 7.10 The plot of the F-distribution $F_{[r,n-(k+1)]}$. The full right-tailed area is the one which represents the rejection region for the null hypothesis H_0 defined in Table 7.8.

The value which is considered in the partial F-test is given by:

$$F_{[r,n-(k+1)]} = \frac{(SSE_R - SSE_F)/r}{MSE_F} \tag{7.27}$$

where SSR_R is the sum of error squares of the reduced model, SSE_F is the sum of error squares of the full model, MSE_F is the mean square error of the full model, k is the number of independent variables in the full regression model, and r is the number of variables removed from the full model when it is reduced. The difference $SSE_R - SSE_F$ is called the extra sum of squares associated to the reduced model. Since this sum of squares refers to r variables, it has r degrees of freedom. Also the partial F-test is based on an F-distribution (like the test defined in Table 7.7 in the previous section). After having fixed a significance level α for the test, we can reject the null hypothesis H_0 (concluding that the full model is statistically better than the reduced one) when the value of the partial F-statistic is greater than the threshold value $F_{[\alpha;r,n-(k+1)]}$ (see Fig. 7.10).

The algorithm of stepwise regression carries out partial F-tests by using the notion of **p-value**. The p-value is the probability that the null hypothesis H_0 of Table 7.8 is correct, and is given by the size of the right-tailed area, in Fig. 7.10, of the considered F-distribution beyond the critical value given by the partial F-test. Using of p-values to carry out the test is perfectly equivalent to use critical values already considered in F-statistics.

The regression algorithm is made of two different parts. The first one, called *forward selection* (steps 2 and 3, below), is devoted to increasing the set of independent variables used by the regression model, by adding the most significant ones among those which do not occur in the model. The second phase is called *backward elimination* (steps 4 and 5, below) and is devoted to the elimination of variables that have been previously inserted in the regression model, but which are become statistically less significant after the insertion of some variables. Finally, the algorithm ends at step 3, when it is not possible to further improve the model by adding or removing variables. The minimum values of significance for introducing and for dropping variables in the model are called P_{in} and P_{out}, respectively. They correspond to significance levels of the partial F-test. They do not need to be equal, but they must be set very carefully because if $P_{in} > P_{out}$, then a loop may occur where a variable enters the model, then leaves it, then reenters, and so on.

The stepwise regression algorithm is based on the following steps:

1. Start from a user defined k-variable multiple regression model (possibly, $k = 0$).
2. Compute the partial F-test for each variable not yet included in the regression model.
3. Check if there is at least one variable whose p-value is less than the user defined threshold value P_{in} (this is the same of checking for those variable whose partial F-statistic is greater than the threshold value $F_{[P_{in};1,n-(k+1)]}$). If there are some variables which fulfill the requirement, add to the model the most significant one (i.e. the one with smaller p-value). If there is no variable which can be added, the regression algorithm stops.
4. Compute the partial F-test for all variables currently in the regression model.
5. Check if there is at least one variable whose p-value is greater than the user defined threshold value P_{out}. If there are some variables which fulfill the requirement, remove from the model the less significant one (i.e. the one with greater p-value).
6. Restart from step 2.

7.8 Trees and Graphs

Trees and graphs are everywhere in discrete mathematics and in computer science. Moreover, a huge number of discrete models in biology are based on trees and/or graphs. Here we present some basic concepts about them (see [224, 217, 220] for more advanced concepts).

A **(rooted) tree** T is specified by a structure $T = (N, r, f)$, where N is a set of $n$odes, (or *vertices*), including a node r, called the **root** of T, and f is a function, assigning a *father* node to any node of N different from r, such that, for any node x different from r, $f^n(x) = r$ for some natural number n ($f^n(x)$ denotes the n-fold application $f(\dots f(f(x))\dots)$ of f to x, $f^0(x) = x$, and $f^1(x) = f(x)$). The terminology about trees is inspired by the many different contexts where they occur, especially genealogy and botanic. For example: *root, node, leaf, parent, father, child, son, brother, ancestor, descendant, branch, generation, path, and depth.*

A simple (unoriented) **graph** G is specified by a pair $G = (N,E)$, where N is a set *n*odes, (or *vertices*), and E is a set of *e*dges (or *arcs*), such that, for any $e \in E$, an unordered pair of nodes of N is specified that are *c*onnected trough e (the graph is *oriented* if the pairs of nodes associated to edges are ordered pairs, it is a *multi-graph* if different edges connect a same pair of notes). Rooted trees can naturally be expressed as graphs where an edge is set between every node and its father.

Basic terminology and example of trees and graphs are given in Chap. 1 (here we recall that "parent, child, sibling" are also used instead of "father, son, brother", or "mother, daughter, sister"; $f(y) = x$ means that x is father of y, and y is son of x, and when also $f(z) = x$ that x and y are brothers). In particular, a **path** of a graph is a sequence of nodes where two successive nodes are connected by an edge of the graph. A **cycle** is a path where the last node is connected with the first node. A graph is **connected** if there is a path between any pair of nodes in the graph. Moreover, it can be shown that the previous characterization of rooted tree is equivalent to that of acyclic (without cycles) and connected graph with one particular node, chosen as its root (if no root is chosen, the structure is an *unrooted tree*). Definitions by induction of trees and graphs can be found in Sect. 5.5.

Rooted trees and graphs can be easily represented by means of diagrams, where a node is a point (or a closed region of space) and an edge is a curve connecting points (regions). In terms of diagrams, a rooted tree has a root point, which branches with other son points which possibly branch again. The terminal nodes which do not continue this process are the **leaves** of the tree. Tags can be added to nodes and edges of trees and graphs, which represent specific kinds of information relevant to particular cases of application.

7.8.1 Types of Trees

The study of trees started with the research of Arthur Cayley who developed an enumeration method in 1875 for calculating the number of isomers of branched *alkanes*.

There are many subtle aspects which identify different notions of trees. Usually, in the case of trees the attribute "labeled" without further specification means that nodes are distinguishable elements, while "unlabeled" means that nodes are not distinguishable for themselves, but only for their positions in the tree. In other words, the individual identity of each node is not relevant in the definition of the tree (however this terminology is not uniform, because very often labels are used as tags, allowing different nodes to have the same label/tag). When specific values are assigned to nodes (and edges) of a tree, for avoiding any confusion with the attributes labeled and unlabeled, we say that the tree is *d*ecorated.(unfortunately this notation is not uniform in the literature).

The tree given in Fig. 7.11 is a decorated tree (by numbers and operations).

If among the sons of a node an order is assumed, then the tree is called an **ordered tree**. Genealogical trees are ordered (this was an important issue in royal families, for deciding the crown rights).

An ordered unlabeled tree is representable by an expression built only by parentheses, for example:

$$((()())(()(()()))).$$

In this case, the root correspond to the external parentheses. This root has two sons, and they have two sons too. The second son of the second son branches again with two nodes.

Every algebraic expression can be represented by a rooted tree. For example, the following expression:

$$(((2+3)\times 5)+(2^3\times 5))$$

can be represented by the tree given in Fig. 7.11.

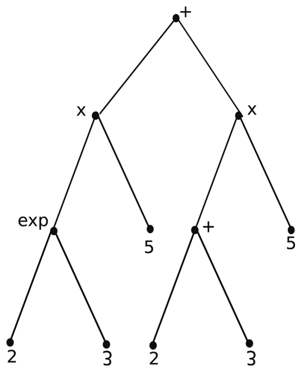


Fig. 7.11 The tree diagram of the expression $(((2+3)\times 5)+(2^3\times 5))$

We can construct any tree by means of a **derivation**, developed in consecutive *steps*. At any *step*, an element occurs which is either a leaf of the tree, or the father of elements occurring at previous steps (#*i* is the element occurring at step *i*). The following derivation provides, at its final step, the expression $(((2+3)\times 5)+(2^3\times 5))$ (for easier reading, numbers and operations decorating the nodes are indicated, with the partial expressions corresponding to the steps):

1. 2
2. 3
3. 5
4. #1 $+$ #2 = (2 + 3)
5. #4 $\times$ #3 $= ((2+3)\times 5)$
6. #1 *exp* #2 $= 2^3$
7. #6 $\times$ #3 $= (2^3\times 5)$
8. #5 $+$ #7 $= (((2+3)\times 5)+(2^3\times 5))$.

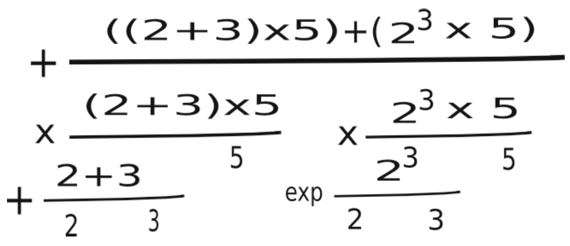


Fig. 7.12 The bar diagram of the expression $(((2+3)\times 5)+(2^3\times 5))$

The diagram of Fig. 7.12 is another way of expressing the hierarchical structure of the the expression $(((2+3)\times 5)+(2^3\times 5))$.

Rooted trees, expressions (built with parentheses), branched diagrams, are equivalent concepts, expressing some kind of hierarchy among components.

7.8.2 Types of Graphs

Edges of graphs can be oriented or not (oriented edges express arrows). The **degree** of a node in a graph is the number of edges connected to the node (in the oriented case it is possible to distinguish entering and exiting edges, consequently, **in** and **out** degree). A graph is complete if every node is directly connected by an edge to every other node. Figure 7.13 shows the graph $K_{3,3}$, investigated by Kuratowski, consisting of six nodes where vertices can be partitioned into two disjoint sets of three nodes, and where every node of one set is connected to every node of the other set. Figure 7.14 shows the complete graph K_5 on five nodes.

We call **hypergraph** is a graph where nodes are graphs. This terminology is not standard, because usually the term refers to a generalization of a graph in which an edge can connect any number of vertices. Here, the term is used by analogy with *hyperset, hypersequence, and hypermultiset* (that is, structures consisting of

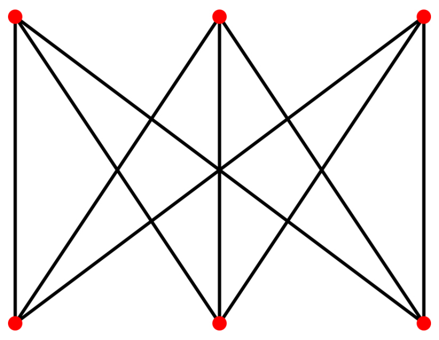


Fig. 7.13 Kuratowski's graph

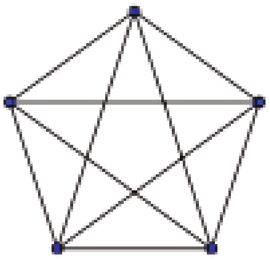


Fig. 7.14 The complete graph on five nodes

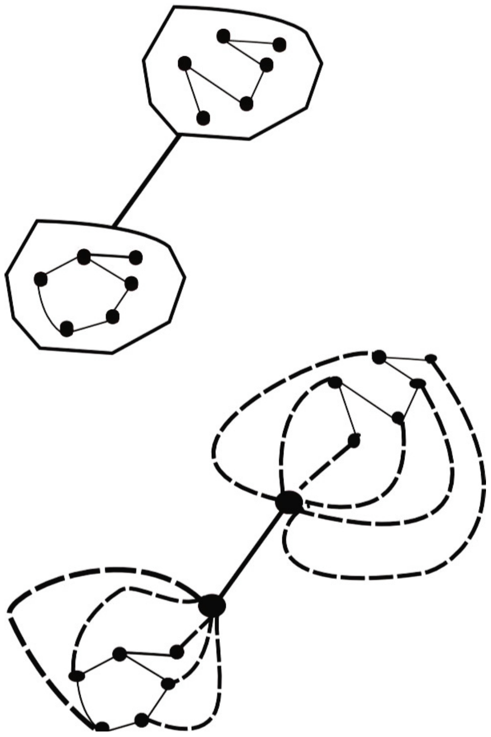


Fig. 7.15 A hypergraph and its representation with a graph with different types of nodes and edges

components that are structure of the same kind). Fig. 7.15 shows an hypergraph with two hyper-nodes and one hyper-edge and its representation as a graph. It is easy to realize that the different levels of the first structure are realized by different types of nodes and edges in the second structure.

7.8.3 Graph Planarity and Colorability

The theory of graphs started with Euler (1707–1793) who posed a problem inspired from the river bridges in the city of Könisberg. The question was: "Is it possible to cross all seven bridges of Könisberg in such a way that all of them are crossed one and only one time?" In general terms, given a (non-oriented) graph we say that a path along the nodes of the graph is *Eulerian* if every edge of the graph occurs exactly once in the path. Euler found a sufficient and necessary condition that a graph has to fulfill for the existence of such a path. Namely, if every node has even degree, that is, an even number of edges connected to it, then when we enter into a node we can exit from it by passing through another edge. But if some node has an odd degree, this manner of visiting a node cannot be realized, and necessarily some edge has to be crossed more than once.

A graph is **planar** when it can be drawn in the plane in such a way that its edges do not intersect. For example, it is easy to realize that the graph of Fig. 7.13, consisting of two sets A and B having both three nodes and where each node of A is connected, by one edge, to every node of B is not a planar graph. Any tree is a planar graph.

A **polyhedron** is a convex region of space bounded by polygons (for every pair of points in the region, every point on the straight line segment that joins them is also within the region). A polyhedron is "equivalent" to a planar graph. The following construction provides the graph $G(P)$ naturally associated to a polyhedron P: remove one face f from P (without removing its edges and vertices), then all vertices and edges of P can be placed on the plane of f (the faces of P correspond to minimal cycles (no subset of nodes of a minimal cycle is a cycle too). Now, consider the portion of plane external to $G(P)$ as a face too, then P and $G(P)$ have the same numbers of vertices, edges, and faces.

Euler proved that convex polyhedra satisfy a relation among the number V of its vertices, the number E of its edges, and the number F of its faces. According to the equivalence between polyhedra and planar graphs, this relation, also known as Euler's **polyhedral formula**, can be proved for connected planar graphs.

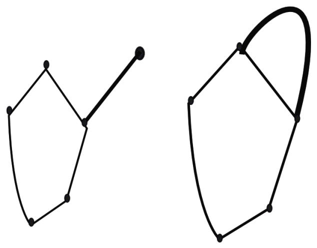


Fig. 7.16 Adding to a graph a new node and an edge (left) or only one edge (right)

Theorem 7.13. *In any connected (finite) planar graph the following equation relating the numbers of its vertices, edges, and faces holds:*

$$V - E + F = 2$$

Proof. This formula trivially holds for the *atomic* connected graph, consisting of only one node, where no edge is present, but one node and one face (the external face) are present. Any (finite) planar connected graph can be constructed starting from an atomic graph by applying, a certain number of times, two different possible operations (see Fig. 7.16): i) adding one node and one edge which connects the new node to a node already in the graph, or ii) adding one edge which connects two nodes already in the graph (which are connected by some path, but not already directly connected by an edge). Of course, in both cases, the value $V - E + F$ does not change, because when we add one node we add also one edge, and when we add only one edge we add also a face (because a new cycle is added to the graph). In conclusion, the value $V - E + F$ is equal to 2 for the atomic graph and remains the same after any of the two operations extending graphs. Therefore, the formula holds for all connected planar graphs. □

The **dual graph** G^* of a given planar graph G has as nodes the faces of G, and two nodes of G^* are connected by an edge if the corresponding faces of G share an edge. An obvious property of planar graphs is that the dual graph of a planar graph is planar too.

A simple relation concerning connected acyclic graphs (unrooted trees) is that the number of vertices equals the number of edges plus 1. In fact, when a connected acyclic graph is constructed, starting from the atomic graph, only the operation on the left of Fig. 7.16 can be applied, because when a new node is added also a new edge is added. Therefore, the node put at beginning of the construction is the only one which does not introduce an edge. Hence, $V = E + 1$.

A **regular polyhedron** is a convex polyhedron bounded by a number of equal regular polygons (convex regions of plane bounded by segments (edges) with equal lengths and forming equal angles). A famous geometrical proposition asserts that only five different kinds of regular polyhedra exist (*hedron* in Greek means face). These polyhedra have been investigated by Plato (in the dialogue entitled Timaeus), whereby they are referred to as the **Platonic solids**. The existence of only five Platonic solids can be proved as a consequence of Euler's polyhedral formula (a proof, based on elementary geometry, is present in Euclid's books). We present this proof as a paradigmatic example of an argument developed in purely mathematical discrete terms, essentially based on the notion of planar graph.

Theorem 7.14 (Platonic solids). *There are only five Platonic solids.*

Proof. In a planar graph associated to regular polyhedron, two properties hold: the **degree** g is the same for all nodes, and the number k of edges around a face is the same for all faces ($k > 2$). These two facts imply the following equations:

$$gV = 2E \tag{7.28}$$

$$2E = kF \tag{7.29}$$

In fact, gV counts edges twice because each edge is common to two nodes, and also kF counts edges twice because each edge is common to two faces.

From Euler's formula $V - E + F = 2$ we have $F = E - V + 2$, therefore we have:

$$2E = k(E - V + 2)$$

that is:

$$(k-2)E = kV - 2k$$

whence:

$$2(k-2)E = 2kV - 4k$$

but from Eq. (7.28) we can replace $2E$ by gV, therefore:

$$(k-2)gV = 2kV - 4k$$

and finally:

$$[2(k+g) - kg]V = 4k. \tag{7.30}$$

In order to have a positive value $4k$, we need to have:

$$2k + 2g > kg$$

that is:

$$g < \frac{2k}{k-2}$$

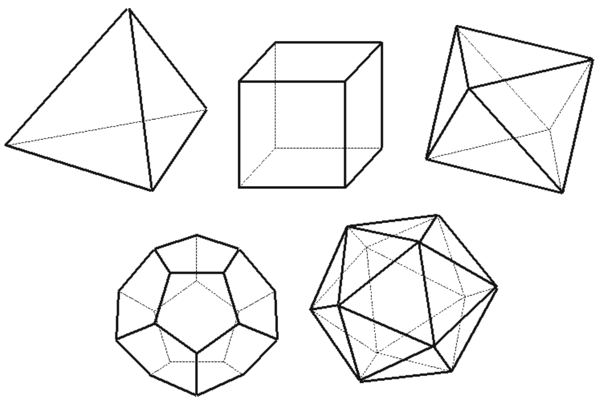


Fig. 7.17 Platonic solids

This implies that $k < 6$. In fact, for $k \geq 6$ the above inequality implies $g < 3$, but in the graphs coming from regular polyhedra k and g are at least 3, In fact, a face needs at least three edges (triangle) and a graph with all nodes with degree 2 is a cycle (only one face).

For $k = 3$ Eq. (7.30) gives:

$$(6-g)V = 12$$

therefore, g can assume only the values 3, 4, and 5, which correspond to the first three lines of following Table 7.9, where the values of F correspond to the number of regular polygons of a polyhedron.

From Eq. (7.30), for $k = 4$ and $k = 5$ the only possible value of g is 3, and we get the last two lines of Table 7.9. We remark that the tetrahedron is self-dual and that the hexahedron is dual of the octahedron, while the dodecahedron is dual of the Icosahedron. No other possibility is allowed. □

Table 7.9 The five Platonic solids

g=3	V = 4	F= 4 (3F = 3V=12)	k=3	Tetrahedron
g=4	V = 6	F = 8 (3F = 4V=24)	k=3	Octahedron
g=5	V = 12	F = 20 (3F = 5V=60)	k=3	Icosahedron
g=3	V = 8	F =6 (dual of Octahedron)	k=4	Hexahedron
g=3	V = 20	F = 12 (dual of Icosahedron)	k=5	Dodecahedron

It is interesting to consider the constructions of the two more complex Platonic polyhedra, the dodecahedron and the icosahedron. For dodecahedron we can start from a pentagon surrounded by other five pentagons (each of them shares an edge with the central one). If the peripheral pentagons share two edges with their neighbors, then they form a sort of cup, a *6-pentagon cup*. By joining two 6-pentagon cups, we get a dodecahedron.

For an icosahedron we consider a 5-triangle cup with a vertex common to all triangles and a pentagon as basis. If at each edge of this pentagon a triangle is put (equal to the triangles of the cup), then between two triangles put at consecutive edges an equal triangle is comprised, sharing an edge with each of them (and the vertex common to these edges). In this way 15 triangles are connected providing a pentagonal border. Then, by adding to this border a reversed 5-triangle cup the icosahedron is formed having 20 triangles. A planar representation of icosahedron is given in Fig. 7.20. Platonic solids occur in natural forms, for example, in capsids of virus. The capsid of Adenovirus, Reovirus, Papovavirus, and Picornavirus is an icosahedron (the regular polyhedron more similar to a sphere, where the 12 vertexes are equipped with extruding fibers used to bind target cells).

From Eq. (7.30), it follows that no regular solid can be constructed with hexagons. However, if 12 pentagons are mixed with hexagons, then convex polyhedra can be constructed. The minimum of such polyhedron, having 20 hexagons and 12 pentagons (60 vertices), is one of 13 Archimedes' solids (realized by two types of regular polygons). Its shape corresponds to the standard football sphere. In 1985, a molecule consisting of 60 atoms of carbon, called **Fullerene**, was discovered which corresponds exactly to Archimedes' polyhedron of 60 vertices, 90 edges, and 32 faces, called *truncated icosahedron*, which is obtained by cutting all 12 vertices of the icosahedron, and thus replacing them with pentagons as new faces (and contemporarily, doubling the edges of the other faces, which pass from triangles to hexagons). Fullerene molecules and similar polyhedron molecules, with different number of vertices, have very important chemical properties and the scientists who discovered C_{60} (Robert F. Curl Jr., Sir Harold Kroto, Richard E. Smalley) won the Nobel prize in 1996.

Colorability of graphs is another important subject of investigation. A graph is k-colorable if it is possible to assign one of k different colors to each of its node, in such a way that nodes with a common edge have different colors (if a graph is k-colorable, then it is also $k+1$-colorable). An important result proves that any planar graph is 4-colorable. Kenneth Appel and Wolfgang Haken proved this fact in 1976 by means of a computer program for analyzing a huge number of possibilities (1936 cases, and 633 cases in a more recent proof by other authors. Previous proofs, since the end of 19th century, turned out to be wrong). The reader is advised to check that the graph of Fig. 7.13 is 2-colorable, while the graph of Fig. 7.14 is not 4-colorable (it is 5-colorable).

It was proved by Kuratowski in 1930 that a graph is planar if and only if it does not "include" graph $K_{3,3}$ of Fig. 7.13 or five-complete graph K_5 of Figure 7.14. In more formal terms, here the forbidden inclusion means that the given graph does not include any expansion of $K_{3,3}$ or K_5, where an expansion of a graph is realized by replacing in it an edge between two nodes with a path between them. Fig. 7.18 shows that non-planar graphs (3D, or three-dimensional) strictly include 4-colorable graphs (4C), which strictly include planar graphs (2D, or 2-dimensional).

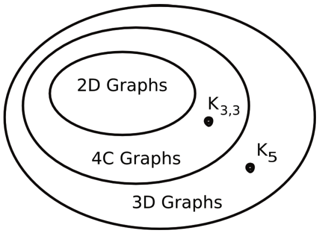


Fig. 7.18 Inclusions among graph planarity, 4-colorability, and non-planarity

We conclude the section by noting that the notion of duality of graphs is a case of a general phenomenon of mathematics, which occurs in many forms and provides very often an important key of comprehension of basic concepts. The simplest example of duality is between elements and classes. A class contains elements, but an element can be characterized by the classes to which it belongs. Analogously, a text

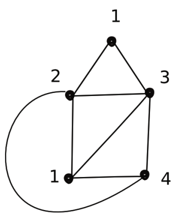


Fig. 7.19 A 4-coloration

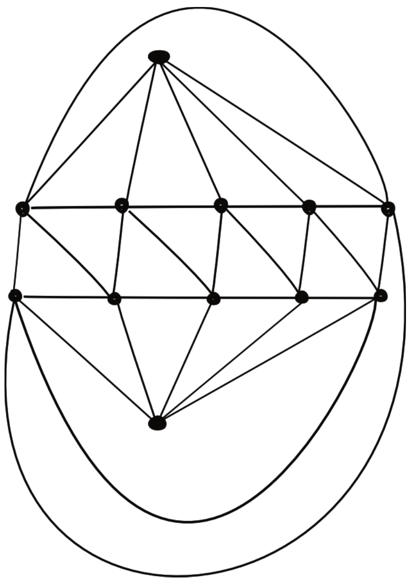


Fig. 7.20 A planar representation of an icosahedron

consists of words which determine its meaning, but conversely the semantic function of a word is determined by the class of texts where it occurs. More technical examples of duality can be found in linear algebra (vectors and linear transformations) and in geometry (points and lines), but cases of duality can be found in biochemistry between metabolic states and reactions (both representable as suitable vectors), or in cell biology where genes and proteins can be considered as dual entities playing the roles of informational and functional biomolecules. The common aspect of all forms of duality is the intrinsic necessity of concepts which are paired, and which can be fully described and analyzed only when they are related, because each of them postulates the other one in order to be completely defined, as it happens in the bilateral symmetry of organisms or in the double-stranded DNA molecules.

7.8.4 *Pointers and Graphs*

The notion of **pointer** (address, link, according to the contexts) allows the definition of another mechanism of representation of (finite) graphs by means of sequences, usually called **hypertexts**. A crucial aspect of hypertexts is the presence of two symbolic levels: i) the *basic level* consisting of sequences over a basic alphabet, and ii) a *structural level*, where some special symbols or strings, also called *metasymbols*, express structural relations (parentheses are structural symbols expressing symbol aggregations).

A **pointer** is a value denoting an address, in an addressed space consisting of **locations**. Each location contains a value (datum), belonging to a set of possible values. Moreover, each location can be partitioned into several fields containing different kinds of values. The example given in the right part of Fig. 7.21 shows the main intuition of this idea. An arrow denotes that the content of the location corresponds to the address of the location from where the arrow originates. In other words, arrows visualize pointers. This mechanism can be realized, because locations can also contain addresses of locations. A structure of this type can be completely determined by the address of the first location (in the figure, the address of location containing value a). This kind of representation can be modified and extended in many ways and corresponds to the way graphs are stored in computer memories (the term **linked list** is often used for these structures). Of course, in correspondence to the different kinds of memories and methods for accessing to the locations, many important concepts and algorithms have been developed, which are specific topics of computer data structures.

7.8.5 *Parentheses and Tags*

Mathematical expressions are usually built by specific symbols, variables, and parentheses. Parentheses are symbols occurring in pairs: opening and closing parenthesis. Any open parenthesis has to be paired to a corresponding closed parenthesis.

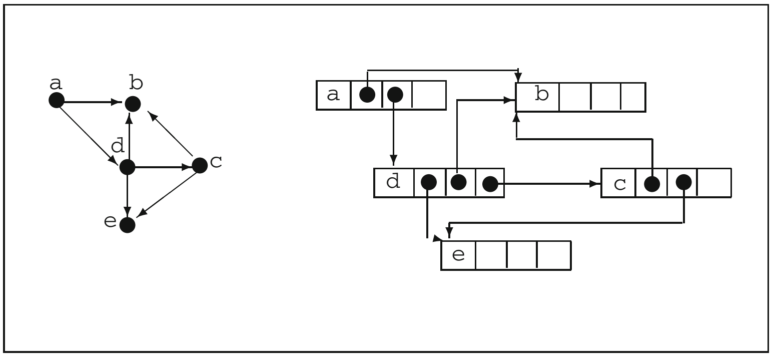


Fig. 7.21 Pointer representation of a graph

Table 7.10 Formula for solving the second degree equation $ax^2+bx+c=0$

$x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$

In mathematical expressions, two, or three, or sometime more types of parentheses are used. Let us consider the formula of Table 7.10 giving the solutions of a second degree equation.

The same formula can be represented by the Latex text [176], reported in Table 7.11. This language developed by Leslie Lamport, is based on the language TEX, which was introduced by Donald Knuth in 1978. It was essentially a formalism providing *abstract formulae* expressing structures of scientific texts and of scientific formulae. In this context, the attribute "abstract" means that such formulae are built by textual characters (including some special characters), but they intend to represent the logical structure of documents and of mathematical formulae, and leave to different levels the specific stylistic choices of a concrete syntactical realization (see [42, 41] for abstract syntax representations of linguistic texts).

Table 7.11 Latex text which produces the formula of Table 7.10

$\$\$x = \backslash frac\{-b \backslash pm \backslash sqrt\{b^\wedge 2-4ac\}\}\{2a\}\$\$$

One of the basic ideas of TEX is the use of a textual formalism for expressing parentheses and operation symbols. This idea, developed by many people, in different contexts of programming language formats, led to the definition of **Mark-up languages**, based on the general notion of *tag*. **Tags** are a generalization of parenthesized expressions. An open **marker** is a symbol put between two angles, and it uniquely corresponds to its closing marker, according the following structure:

$$< marker > \quad content \quad < /marker > .$$

Mark-up languages are the basis of data transfer protocols (for example, HTML, XML), where information at different levels is enclosed in different types of parenthesized expression (XML stands for "eXtensible Markup Language"). When a notion of link is added to tags, by means of **keys**, which univocally identify tag occurrences, then any graph can be represented by a suitable tag expression. In fact, let us, for example, insert inside the left marker an additional information, after the special symbol #, which denotes a key of the tag. In this manner, we can easily express the pointer structure of Fig. 7.13 in the XML style of Table 7.13.

Trees are special graphs, sequences are special trees (where any node different from a leaf has only one son), and a multiset is a special sequence (the sequence of multiplicities, with respect to an ordering of the elements). Moreover, membranes which realize hypermultisets, that is, iterated multiset aggregations, are naturally

Table 7.12 The pointer structure of Fig. 7.21 in XML format

```
< list >
    < element #p1 >
        < content >a< /content >
        < pointer − 1 >p2< /pointer >
        < pointer − 2 >p3< /pointer >
    < /element >

    < element #p2 >
        < content >b< /content >
    < /element >

    < element #p3 >
        < content >d< /content >
        < pointer − 1 >p5< /pointer >
        < pointer − 2 >p2< /pointer >
        < pointer − 3 >p4< /pointer >
    < /element >

    < element #p4 >
        < content >c< /content >
        < pointer − 1 >p2< /pointer >
        < pointer − 2 >p5< /pointer >
    < /element >

    < element #p5 >
        < content >e< /content >
    < /element >
</list>
```

represented by expressions of parentheses. But if we extend parentheses with tags possibly including links (or keys), then we can easily represent graphs.

Parentheses were introduced in mathematical notation relatively late (Girard, 17th century). Before them, other methods for aggregating symbols were based on vincula and dots. Vinculum (used by Viète, the same mathematician who introduced letters in algebraic manipulations, in 15th century) is a line over/below the symbols it connects. Table 7.10 expresses the resolution formula of second degree equations in terms of vincula notations (for root square and fraction).

A dotted notation was systematically used by the Italian mathematician Giuseppe Peano in his *Formulario Mathematico* (1908), and this notation was also inherited by Bertrand Russell and Alfred Whitehead in their *Principia Mathematica*. When dots are put next to operation symbols, they can denote a *priority* of their application (a smaller number of dots means a greater priority). In specific contexts, operation

Table 7.13 A 4-node complete graph expressed in XML format

```
< graph >
    < node #1 >
        < edges > 2, 3, 4 < /edges >
    < /node >

    < node #2 >
        < edges > 1, 3, 4 < /edges >
    < /node >

    < node #3 >
        < edges > 1, 2, 4 < /edges >
    < /node >

    < node #4 >
        < edges > 1, 2, 3 < /edges >
    < /node >
< /graph >
```

symbols have an implicit priority, therefore some dots can be avoided (for example, multiplication and division are applied before sum and subtraction).

Dotted notation can be generalized as an alternative device with respect to parentheses (or used jointly with them). In general, dots can be used for breaking a sequence into pieces. Firstly, the pieces without dots are aggregated, then pieces between one dot are aggregated, and so on. In other words, dots indicate the order of aggregation of components in a sequence. This provides a linear representation of hypermultisets, with dots, that is, extra symbols which can be iterated any number of times. For example, the sequence below is the dot representation of membrane/parenthesis structure: $(((a,c)(a,c)(a,c)),(c,c),((a,a,b)(a,a,b)(a,a,b)))$.

$$(ac \bullet ac \bullet ac \bullet\bullet cc \bullet\bullet aab \bullet aab \bullet aab).$$

In the expression below:

$$(3+5)\times((12\times 10)+(7\times 13))$$

parentheses are removed by putting near to the operation symbols a number of dots representing their application priority (fewer dots for greater priorities). Its dotted representation is the following:

$$3+5 \ \times\bullet\bullet \ 12\times 10 \ +\bullet \ 7\times 13.$$

In considering discrete structures, we discussed very often the passage from a representation to another representation of the same structure. This is a crucial point in their mathematical and computational analyses. A good representation has to preserve the information. This means that it has to be possible to pass from a representation of an object to another one of the same object and vice versa. However, the equivalence of information content does not mean that two representations are completely equivalent. In fact, very often, the information can be processed in a better manner when an object is represented in one format rather than in another. A typical example of this phenomenon is provided by the elementary arithmetical algorithms, which can be easily defined within the positional representation of numbers, but become sometimes very difficult when numbers are represented in terms of geometrical entities. In general, the computational evaluation of a representation is an important issue in the algorithmic analysis of data structures and depends on aspects related to the finality of their specific context of use. For example, the advantage of tag representations is given by the possibility of expressing, in textual formats, complex information with graphical, and even multi-medial, features by using only symbols of the standard 7-bit ASCII alphabet of 128 characters (American Standard Code for Information Interchange).

7.8.6 Tree Enumerations

Tree enumeration formulae are an old subject first investigated in connection to chemical structures. In Knuth's book [224] many classical results are reported for different kinds of trees. Different notions of trees require different enumeration methods. We recall briefly the aspects which are crucial in this regard. Trees can be rooted or unrooted, ordered or unordered, or labeled or unlabeled. An unrooted tree is a connected acyclic graph, while a rooted tree is an unrooted tree where a node is chosen as an ancestor of all nodes of the tree. This means that a genealogical path with respect to the root can be assigned to every node of the tree. In ordered trees, a generation order is established among the sons of a node, in unordered trees no order is given among the sons of a node. In labeled trees all nodes are distinct, while in unlabeled trees nodes are undistinguishable.

A **forest** is a sequence of trees (a multiset of trees, if order is not relevant). Of course, there is a one-to-one correspondence between forests on n nodes and rooted trees of $n+1$ nodes. In fact, any tree becomes a forest after removing its root, and conversely any forest become a tree if we add a node as father of the roots of the trees in the forest. Forests of unlabeled ordered rooted trees on n nodes correspond to the parenthesized expressions of n parenthesis pairs enumerated by Catalan numbers. From this enumeration, the number $(2n-2)!/(n-1)!$ of labeled ordered rooted trees of n nodes easily follows.

For unlabeled unordered rooted trees no exact analytical formula is available, but a recurrent formula, obtained by means of generating functions, was given by

Otter in the middle of last century, together with a complex asymptotic approximate formula [232]. Recurrent formulae, deduced by means of elementary combinatorial arguments, but computationally less efficient than Otter's formula, are given in [226, 227].

The number of labeled unordered rooted trees of n nodes is n^{n-1}. This formula was discovered by Cayley. Here we present a proof of this result, which is based on an interesting encoding of these trees as sequences of length $n-1$ constructed over the n nodes, due to Prüfer [231], as reported in [224].

Encoding trees into sequences.
Let T be a labeled rooted tree of n nodes numbered by $1,2,\ldots n$. The following procedure provides a sequence encoding T. For $i = 1,\ldots n-1$, take the leaf x of T having the minimum index; set $x_i = \mathit{father}(x)$ and update T, by deleting from T the node x and its edge. The sequence $(x_1, x_2, \ldots, x_{n-1})$ is the encoding of T.

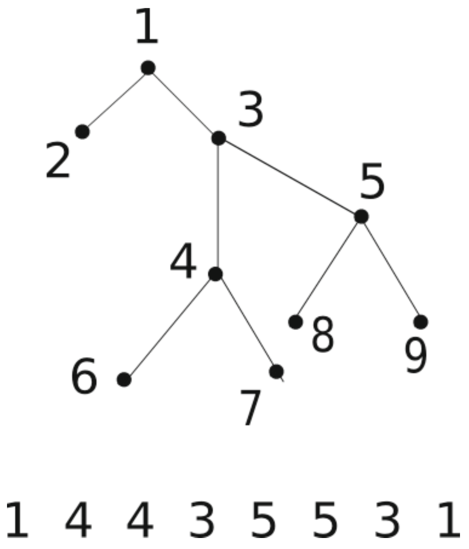


Fig. 7.22 Representation of a tree of n nodes by means of sequence of length $n-1$ over the nodes

We can reverse the encoding process of trees given above. In fact, any sequence of length $n-1$ over n nodes uniquely individuates a labeled rooted tree over n nodes. The following procedure allows us to reconstruct the tree of n nodes encoded by a sequence of $n-1$ nodes.

Decoding sequences into trees

Given a sequence α over a set X of n nodes numbered by $1, 2, \ldots n$, the following procedure provides another sequence β of nodes over X. Let j be the smallest index of the nodes of X which does not occur in α, then put $\beta = (x_j)$ and update α, by removing its first element.

```
For $i = 2, \ldots, n-1$ do:
    let $j$ be the smallest index of the nodes which do not occur in $\alpha$ and do not already occur in $\beta$, then update $\beta$ by adding to it $x_j$ as last element, and update $\alpha$ by removing its first element.
done.
```

The set of pairs of elements having the same positions in α and β, respectively, provide the edges of the tree encoded by the sequence α.

Figure 7.23 shows a tree and the sequence $(1,4,4,3,5,5,3,1)$ encoding the tree. The sequence $(2,6,7,4,8,9,5,3)$ is generated by the above procedure applied to $(1,4,4,3,5,5,3,1)$. In fact, the pairs of elements having the same positions in $(1,4,4,3,5,5,3,1)$ and $(2,6,7,4,8,9,5,3)$ provide the edges of the tree encoded by $(1,4,4,3,5,5,3,1)$.

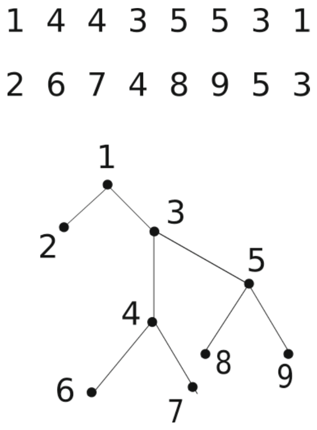


Fig. 7.23 The tree over nodes $\{1,2,3,4,5,6,7,8,9\}$ is encoded by the sequence $(1,4,4,3,5,5,3,1)$. This sequence with the sequence $(2,6,7,4,8,9,5,3)$ returns the tree

Cayley's formula easily follows from the previous encoding. In fact, given n nodes, there are n^{n-1} distinct sequences of length $n-1$. It could be considered counterintuitive that a two-dimensional structure as a tree could be encoded by a sequence. However, at a more careful analysis, we realize that the encoding sequence of a tree is based on a previous ordering of its nodes, therefore, in more precise terms, the sequence encoding of trees over n nodes is given by two sequences: the sequence ordering the nodes, and another sequence, of length $n-1$, over the nodes.

7.8.7 Binary Trees

A natural correspondence can be easily established between any forest and a binary tree (each node which is not a leaf has at most two sons). The correspondence is depicted by Fig. 7.24. The idea is very simple: the roots of the forest are linked in the order of the corresponding trees of the forest; moreover, every father is linked only to the first son and every son to the following son. The root of the first tree of the forest is the root of the resulting tree. In this way, any forest provides a binary tree.

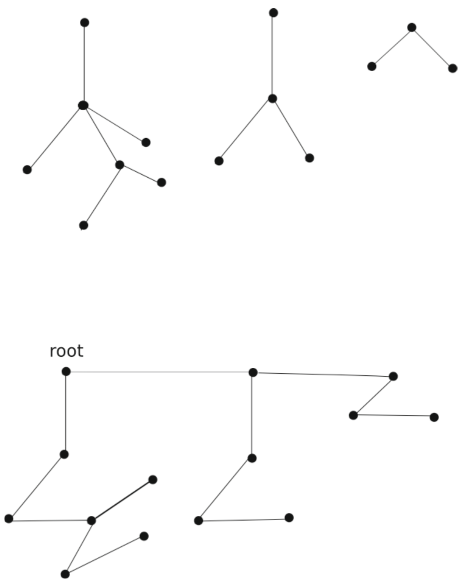


Fig. 7.24 Representation of a forest of trees by means of a binary tree

7.8.8 Phylogenetic Trees

Given n organisms how many different phylogenetic trees can be built from them? Let us assume that we are interested only to the phylogenetic branching processes. A binary tree is *complete* if any node different from the root has exactly two sons (a tree is k-ary if any node different from the root has at most k sons, and it is complete if any node different from the root has exactly k sons).

The problem of (branching) phylogenetic reconstruction reduces to the enumeration of the set $BT(n)$ of rooted complete binary trees on n leaves [219].

Firstly, we recall the following proposition, which can be easily shown by induction on the number of nodes.

Proposition 7.15. *In a complete binary tree with n leaves there are* $n-1$ *internal nodes (that is,* $2n-1$ *nodes).*

The following proposition is an easy consequence of the the previous one, because any tree has a number edges equal to the number of its nodes minus one.

Proposition 7.16. *In a complete binary tree with n leaves there are* $2n-2$ *edges.*

A *Unrooted Branches* (UB) is a unrooted trees where any internal node has degree 3. Given a BT, if we remove its root, by collapsing the two edges exiting from it in one edge, we get a UB (see 7.25). Therefore, the following proposition holds.

Proposition 7.17. *Any BT can be transformed into a UB with the same number of leaves and a number of edges decreased by one.*

In order to obtain the enumeration of the set $BT(n)$ of BT with n leaves, we consider the set $UB(n)$ of UB with n leaves. For simplicity we continue to denote by $BT(n)$ and $UB(n)$ the cardinalities of these sets, respectively.

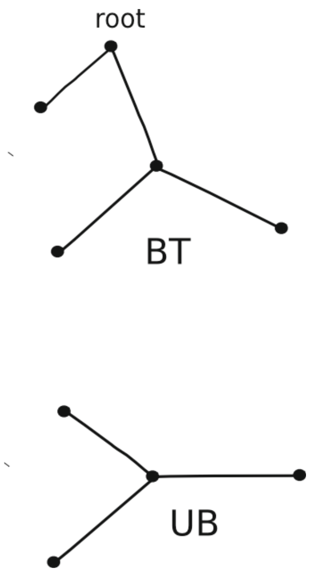


Fig. 7.25 Transforming a binary tree (top) in an unrooted branch (bottom) with the same leaves

Proposition 7.18. *The number* $BT(n)$ *of rooted complete binary trees with n leaves is given by* $(2n-3)UB(n)$ *where* $UB(n)$ *is the number of unrooted branchings with n leaves.*

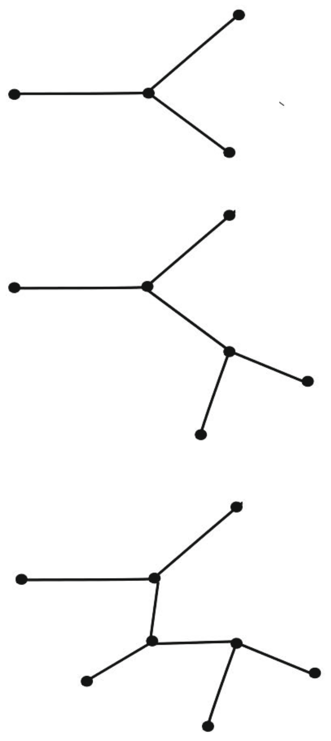


Fig. 7.26 Increasing UB leaves. Top: a minimal UB; Middle: an UB where the left leaf edge of the top UB is replaced by a minimal UB; Bottom: an UB where the internal edge of the middle UB is replaced by a minimal UB

Proof. From the previous propositions it follows that any BT with n leaves has $(2n-2)$ nodes and $(2n-3)$ edges. Moreover, from each UB a BT, with the same number of leaves can be obtained by de-collapsing one of its edges. □

Proposition 7.19. *The number of rooted binary trees of n leaves is*

$$BT(n) = (2n-3)!!$$

Proof. The minimal UB has one internal node connected with 3 edges to 3 leaves, therefore $UB(3) = 1$. We pass from a UB to another UB having a number of leaves increased by one, by replacing one edge with a minimal UB (see, Fig. 7.26). For example, $UB(4) = 3UB(3)$, and $UB(5) = 5UB(4)$, because an UB with 3 leaves has 3 edges, and an UB with 4 leaves has 5 edges. From Propositions 7.16 and 7.17, we know that a UB of $n-1$ leaves has $2(n-1)-3$ edges.

Therefore, the following equation holds ($n > 3$):

$$UB(n) = UB(n-1)(2(n-1)-3)$$

whence, for $n > 3$:

$$UB(n) = 3 \cdot 5 \cdot \ldots \cdot (2(n-1)-3) = 3 \cdot 5 \cdot \ldots \cdot (2n-5).$$

Thus, the number $UB(n)$ given above, for $n > 2$, can be concisely expressed by (denoting $n \cdot (n-2) \cdot (n-4) \ldots \cdot 3$ by $n!!$):

$$UB(n) = (2n-5)!!$$

Therefore, from Proposition 7.18, we get $BT(n) = (2n-3)!!$, which concludes the proof. $\square$

References

References for Chapter 1

1. Alberts, B., et al.: Molecular Biology of the Cell. Garland Science, New York (1994)
2. von Baeyer, H.C.: Information: The New Language of Science. Harvard University Press (2004)
3. Ball, P.: The Elements: A Very Short Introduction. Oxford University Press (2002)
4. Ball, P.: Molecules: A Very Short Introduction. Oxford University Press (2003)
5. Davis, P.: The Origin of Life. Penguin Books (1999)
6. de Duve, C.: Life Evolving. Molecules, Mind, and Meaning. The Christian René de Duve Trust (2002)
7. Doolittle, W.F.: Uprooting the tree of life. Scientific American 6, 90–95 (2000)
8. Hoppfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Nat. Acad. Sci. USA, Biophysics 79, 2554–2558 (1982)
9. McCulloch, W., Pitts, W.: A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)
10. Schrödinger, E.: What is Life? The physical Aspect of the Living Cell. Cambridge University Press (1944)

References for Chapter 2

11. Adleman, L.: Molecular computation of solutions to combinatorial problems 286, 1021–1024 (1994)
12. Adleman, L.: Computing with DNA. Scientific American 279(2), 54–61 (1998)
13. Amos, M.: Theoretical and Experimental DNA Computation. Springer (2005)
14. Bath, J., Turberfield, A.J.: DNA nanomachines. Nature Nanotechnology 2(5), 275–284 (2007)
15. Benenson, K., Paz-Elitzur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434 (2001)
16. Baker, M.C.: The atoms of language. Basic Books (2001)
17. Braich, R.S., Johnson, C., Rothemund, P.W.K., Hwang, D., Chelyapov, N., Adleman, L.M.: Solution of a Satisfiability Problem on a Gel-Based DNA Computer. In: Condon, A., Rozenberg, G. (eds.) DNA 2000. LNCS, vol. 2054, pp. 27–42. Springer, Heidelberg (2001)
18. Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W.K., Adleman, L.: Solution of a 20-variable 3-SAT problem on a DNA computer. Science 296, 417–604 (2002)

19. Brendel, V., Busse, H.G.: Genome structure described by formal languages. Nucleic Acids Research 12(5), 2561–2568 (1984)
20. Castellini, A., Franco, G., Manca, V.: A dictionary based informational genome analysis. BMC Genomics 13, 485 (2012)
21. Cristianini, N., Hahn, M.W.: Computational genomics. Cambridge University Press (2007)
22. Deonier, R.C., Tavaré, S., Waterman, M.S.: Computational Genome Analysis: An Introduction. Springer Science + Business Media, Inc. (2005)
23. The Encode Project Consortium: An integrated encyclopedia of DNA elements in the human genome. Nature 489, 57–72 (2012)
24. Fici, G., Mignosi, F., Restivo, A., Sciortino, M.: Word Assembly through minimal forbidden words. Theoretical Computer Science 359, 214–230 (2006)
25. Fofanov, Y., et al.: How independent are the appearances of n-mers in different genomes? Bioinformatics 20(15), 2421–2428 (2008)
26. Franco, G.: Biomolecular computing: combinatorial algorithms and laboratory experiments. PhD Thesis, University of Verona (2006)
27. Franco, G.: Perspectives in computational genome analysis. Discrete and topological models in molecular biology. Springer (2013)
28. Franco, G., Giagulli, C., Laudanna, C., Manca, V.: DNA Extraction by XPCR. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 104–112. Springer, Heidelberg (2005)
29. Franco, G., Manca, V.: Algorithmic applications of XPCR. Natural Computing 10, 805–819 (2011)
30. Franco, G., Manca, V., Giagulli, C., Laudanna, C.: DNA Recombination by XPCR. In: Carbone, A., Pierce, N.A. (eds.) DNA 2005. LNCS, vol. 3892, pp. 55–66. Springer, Heidelberg (2006)
31. Giancarlo, R., Scaturro, D., Utro, F.: Textual Data Compression in Computational Biology: A synopsis. Bioinformatics 25, 1575–1586 (2009)
32. Gibson, G., Muse, S.V.: A Primer of Genome Science. Sinauer Associates Inc. (2009)
33. Gimona, M.: Protein Linguistics — a grammar for modular protein assembly? Nature 7, 68–73 (2006)
34. Hampikian, G., Andersen, T.: Absent sequences: nullomers and primes. In: Pac. Symp. Biocomputing, vol. 12, pp. 355–366 (2007)
35. Hao, B., Qi, J.: Prokaryote phylogeny without sequence alignment: from avoidance signature to composition distance. J. Bioinf. and Comput. Biol. 2, 1–19 (2004)
36. Haubold, B., Pierstorff, N., Möller, F., Wiehe, T.: Genome comparison without alignment using shortest unique substrings. BCM Bioinf. 6, 123 (2005)
37. Head, T.: Formal language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bulletin of Mathematical Biology 9(6), 737–759 (1987)
38. Lindenmayer, A.: Mathematical models for cellular interaction in development. J. Theoret. Biology 18, 280–315 (1968)
39. Lipton, R.J.: DNA solutions of hard computational problems. Science 268, 542–544 (1995)
40. Lombardo, R.: Unconventional computations and genome representations. PhD Thesis, University of Verona (2013)
41. Manca, V., Jiménez-López, M.D.: GNS: Abstract Syntax for Natural Languages. Languages: mathematical approaches. Triangle 8, 55–80 (2012); Jiménez-López M.D. (ed.)
42. Manca, V., Suzuki, J., Suzuki, Y.: An XML Representation of Basic Japanese Grammar. In: Bel-Enguix, G., Jiménez-López, M.D. (eds.) Language as a Complex System: Interdisciplinary Approaches, pp. 215–244. Cambridge Scholars Publishing (2010)

43. Manca, V.: A Proof of Regularity for Finite Splicing. In: Jonoska, N., Păun, G., Rozenberg, G. (eds.) Aspects of Molecular Computing. LNCS, vol. 2950, pp. 309–317. Springer, Heidelberg (2003)
44. Manca, V.: On the Logic and Geometry of Bilinear Forms. Fundamenta Informaticae 64, 261–273 (2005)
45. Manca, V., Franco, G.: Computing by polymerase chain reaction. Mathematical Biosciences 211, 282–298 (2008)
46. Manca, V., Zandron, C.: A Clause String DNA Algorithm for SAT. In: Jonoska, N., Seeman, N.C. (eds.) DNA 2001. LNCS, vol. 2340, pp. 172–181. Springer, Heidelberg (2002)
47. Păun, G., Rozenberg, G., Salomaa, A. (eds.): DNA Computing. Springer (2000)
48. Rozenberg, G., Salomaa, A.: Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology. Spinger (1992)
49. Păun, G.: Computing with Membranes. Journal of Computer and Systems Science 61(1), 108–143 (2000)
50. Păun, G.: Membrane Computing. An Introduction. Springer (2002)
51. Păun, G., Rozenberg, G., Salomaa, A.: Oxford Handbook of Membrane Computing. Oxford University Press (2010)
52. Percus, J.K.: Mathematics of Genome Analysis. Cambridge University Press (2002)
53. Prusinkiewicz, P., Lindenmayer, A., Hanan, J.S., Fracchia, F.D., Fowler, D., de Boer, M.J.M., Mercer, L.: The Algorithmic Beauty of Plants. Springer (1990)
54. Puglisi, A., Baronchelli, A., Loreto, V.: Cultural route to the emergence of linguistic categories. PNAS 105(23), 7936–7940 (2008)
55. Rose, J.A., Hagiya, M., Deaton, R.J., Suyama, A.: A DNA-based in vitro genetic program. Journal of Biological Physics 28, 493–498 (2002)
56. Saiki, R.K., Scharf, S., Faloona, F., Mullis, K.B., Horn, G.T., Erlich, H.A., Arnheim, N.: Enzymatic Amplification of β-Globin Genomic Sequences and Restriction Site Analysis for Diagnosis of Sickle Cell Anemia. Science 230, 1350–1354 (1985)
57. Searls, D.B.: String variable grammar: A logic grammar formalism for the biological language of DNA. Journal of Logic Programming 24, 73–102 (1995)
58. Park, S.H., Yan, H., Reif, J.H., LaBean, T.H., Finkelstein, G.: Electronic Nanostructures Templated on Self-Assembled DNA Scaffolds. Nanotechnology 15, S525–S527 (2004)
59. Searls, D.B.: The language of genes. Nature 420, 211–217 (2002)
60. Searls, D.B.: Molecules, Languages and Automata. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS (LNAI), vol. 6339, pp. 5–10. Springer, Heidelberg (2010)
61. Seeman, N.C.: DNA in a material worldblocking. Nature 421, 427–431 (2003)
62. Vinga, S., Almeida, J.: Alignment-free sequence comparison - a review. Bioinformatic 19(4), 513–523 (2003)
63. Watson, J.D., Crick, F.H.G.: Molecular structure of nucleic acids: a structure for deoxiribose nucleic acid. Nature 171, 737–738 (1953)
64. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: compressing and indexing documents and images, 2nd edn. Academic Press (1999)

References for Chapter 3

65. Bernardini, F., Manca, V.: Dynamical aspects of P systems. Biosystems 70(2), 85–93 (2003)
66. von Bertalanffy, L.: General Systems Theory: Foundations, Developments, Applications. George Braziller Inc., New York (1967)

67. Bianco, L.: Membrane models of biological systems. PhD Thesis, University of Verona (2007)
68. Bianco, L., Fontana, F., Franco, G., Manca, V.: P systems for biological dynamics. In: [74] (2006)
69. Bianco, L., Manca, V., Marchetti, M., Petterlini, M.: Psim: a simulator for biochemical dynamics based on P systems. In: CEC 2007 - IEEE Congress on Evolutionary Computation, Singapore, September 25-28 (2007)
70. Castellini, A., Zucchelli, M., Busato, M., Manca, V.: From time series to biological network regulations. Molecular Biosystems (2012), doi:10.1039/C2MB25191D
71. Castellini, A.: Algorithms and software for biological MP modeling by statistical optimization techniques. PhD Thesis, University of Verona (2010)
72. Castellini, A., Franco, G., Manca, V.: Hybrid Functional Petri Nets as MP systems. Natural Computing 9, 61–81 (2010)
73. Castellini, A., Franco, G., Pagliarini, R.: Data Analysis pipeline from laboratory to MP models. Natural Computing 10, 55–76 (2011)
74. Ciobanu, G., Păun, G., Perez-Jimenez, M.J.(eds.): Applications of Membrane Computing. Springer (2006)
75. Condon, A., Harel, D., Kok, J.N., Salomaa, A., Winfree, E. (eds.): Algorithmic Bioprocesses. Springer (2009)
76. Draper, N., Smith, H.: Applied Regression Analysis, 2nd edn. John Wiley & Sons, New York (1981)
77. Fontana, F., Manca, V.: Discrete solutions to differential equations by metabolic P systems. Theoretical Computer Science 372, 165–182 (2007)
78. Fontana, F., Manca, V.: Predator-prey dynamics in P systems ruled by metabolic algorithm. BioSystems 91, 545–557 (2008)
79. Gheorghe, M., Manca, V., Romero-Campero, F.J.: Deterministic and stochastic P systems for modeling cellular processes. Natural Computing 9(2), 457–473 (2010)
80. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comp. Phys. 22, 403–434 (1976)
81. Goldbeter, A.: A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. PNAS 88(20)
82. Goldbeter, A.: Biochemical Oscillations and Cellular Rhythms: The molecular bases of periodic and chaotic behaviour. Cambridge University Press (1996)
83. Goldbeter, A.: Computational approaches to cellular rhythms. Nature 420, 238–245 (2002)
84. Hilborn, R.C.: Chaos and Nonlinear Dynamics, 2nd edn. Oxford University Press (2000)
85. Hocking, R.R.: The Analysis and Selection of Variables in Linear Regression. Biometrics 32 (1976)
86. Johnson, S.C.: Hierarchical Clustering Schemes. Psychometrika 2, 241–254 (1967)
87. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice Hall (1989)
88. Lambert, J.D.: Computational Methods in Ordinary Differential Equations. Wiley & Sons (1973)
89. Lotka, A.J.: Analytical Note on Certain Rhythmic Relations in Organic Systems. Proc. Natl. Acad. Sci. U.S. 6, 410–415 (1920)
90. Manca, V.: String rewriting and metabolism. A logical perspective. In: [111], pp. 36–60 (1998)
91. Manca, V., Bianco, L.: Biological networks in metabolic P systems. BioSystems 91, 489–498 (2008)

92. Manca, V.: The metabolic algorithm: principles and applications. Theoretical Computer Science 404, 142–157 (2008)
93. Manca, V.: Log-Gain Principles for Metabolic P Systems. In: [75], ch. 28, pp. 585–605. Springer (2009)
94. Manca, V.: Fundamentals of Metabolic P Systems. In: [51], ch. 19 (2010)
95. Manca, V.: Metabolic P Dynamics. In: [51], ch. 20 (2010)
96. Manca, V.: From P to MP Systems. In: Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) WMC 2009. LNCS, vol. 5957, pp. 74–94. Springer, Heidelberg (2010)
97. Manca, V.: Metabolic P systems. Scholarpedia 5(3), 9273 (2010)
98. Manca, V., Bianco, L., Fontana, F.: Evolution and Oscillation in P Systems: Applications to Biological Phenomena. In: Mauri, G., Păun, G., Jesús Pérez-Jímenez, M., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 63–84. Springer, Heidelberg (2005)
99. Manca, V., Lombardo, R.: Computing with Multi-membranes. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (eds.) CMC 2011. LNCS, vol. 7184, pp. 282–299. Springer, Heidelberg (2012)
100. Manca, V., Marchetti, L.: Metabolic approximation of real periodical functions. The Journal of Logic and Algebraic Programming 79, 363–373 (2010)
101. Manca, V., Marchetti, L.: Goldbeter's Mitotic Oscillator Entirely Modeled by MP Systems. In: Gheorghe, M., Hinze, T., Păun, G., Rozenberg, G., Salomaa, A. (eds.) CMC 2010. LNCS, vol. 6501, pp. 273–284. Springer, Heidelberg (2010)
102. Manca, V., Marchetti, L.: Log-Gain Stoichiometic Stepwise regression for MP systems. Int. J. of Foundations of Computer Science 22(1), 97–106 (2011)
103. Manca, V., Marchetti, L.: Solving inverse dynamics problems by means of MP systems. BioSystems 109, 78–86 (2012)
104. Manca, V., Marchetti, L.: An algebraic formulation of inverse problems in MP dynamics. International Journal of Computer Mathematics (2012), doi:10.1080/00207160.2012.735362
105. Manca, V., Marchetti, L., Pagliarini, R.: MP modelling of glucose-insulin interactions in the Intravenous Glucose Tolerance Test. International Journal of Natural Computing Research 2(3), 13–24 (2011)
106. Manca, V., Martino, M.D.: From String Rewriting to Logical Metabolic Systems. In: Păun, G., Salomaa, A. (eds.) Grammatical Models of Multiagent Systems, pp. 297–315. Gordon and Breach (1999)
107. Manca, V., Pagliarini, R., Zorzan, S.: A photosynthetic process modelled by a metabolic P system. Natural Computing 8, 847–864 (2009)
108. Marchetti, L.: MP representations of biological structures and dynamics. PhD Thesis, University of Verona (2012)
109. Marchetti, L., Manca, V.: A Methodology Based on MP Theory for Gene Expression Analysis. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (eds.) CMC 2011. LNCS, vol. 7184, pp. 300–313. Springer, Heidelberg (2012)
110. Pagliarini, R.: Modeling and reverse engineering of biological phenomena by means of metabolic P systems. PhD Thesis, University of Verona (2011)
111. Păun, G. (ed.): Computing with biomolecules. Theory and Experiments, pp. 36–60. Spinger, Singapore (1998)
112. Prigogine, I., Lefever, R.: Symmetry Breaking Instabilities in Dissipative Systems II. J. Chem. Phys. 48, 1695–1700 (1968)
113. Reising, W., Rozenberg, G. (eds.): Lectures on Petri Nets. Springer (1998)

114. Szallasi, Z., Stelling, J., Periwal, V. (eds.): System Modeling in Cellular Biology Lectures on Petri Nets. The MIT Press (2006)
115. Voit, E.O.: Computational Analysis of Biochemical Systems. Cambridge University Press (2000)
116. Volterra, V.: Fluctuations in the abundance of a species considered mathematically. Nature 118, 558–560 (1926)

References for Chapter 4

117. Ashby, W.R.: An Introduction to Cybernetics. Chapman & hall Ltd., London (1956)
118. Barbieri, M.: The organic codes: The birth of semantic biology. peQuod (2001)
119. Hamilton, A.J., Baulcombe, D.: A Species of Small Antisense RNA in Post-transcriptional Gene Silencing in Plants. Science 286(5441), 950–952 (1999)
120. Campos, S.K., Barry, M.A.: Current advances and future challenges in adenoviral vector biology and targeting. Curr. Gene. Ther. 2, 189–204 (2007)
121. Winkler, W.C., Cohen-Chalamish, S., Breaker, R.R.: An mRNA structure that controls gene expression by binding FMN. Proc. Natl. Acad. Sci. USA 99(25), 15908–15913 (2002)
122. Casti, J.L.: Complexification. Harper Perennial (1994)
123. Cheng, W.S., Dzojic, H., Nilsson, B., Totterman, E.M.: An oncolytic conditionally replicating adenovirus for hormone dependent and hormone-independent prostate cancer. Cancer Gene. Ther. 13, 13–20 (2006)
124. Chu, R.L., Post, D.E., Khuri, F.R., Van Meir, E.G.: Use of replicating oncolytic adenoviruses in combination therapy for cancer. Clin. Cancer Res. 10(5), 299–312 (2004)
125. Darwin, C.: The Origin of Species. John Murray, London (1859)
126. Davies, P.: About Time. Orion Productions (1995)
127. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2007)
128. Eigen, M., Schuster, P.: The Hypercycle: A Principle of Natural Self-Organization. Springer (1979)
129. Gheorghe, M. (ed.): Molecular Computational Models: Unconventional Approachers. Idea Group Publishing (2005)
130. Fire, A., Xu, S.Q., Montgomery, M.K., Kostas, S.A., Driver, S.E., Mello, C.C.: Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans. Nature 391, 806–811 (1998)
131. Fisher, R.A.: The Genetical Theory of Natural Selection. Oxford University Press (1930)
132. Gelfand, M.S., Mironov, A.A., Jomantas, J., Kozlov, Y.I., Perumov, D.A.: A conserved RNA structure element involved in the regulation of bacterial riboflavin synthesis genes. Trends Genet. 15(11), 439–442 (1999)
133. Gibson, D.G., et al.: Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. Science 329(5987), 52–56 (2010)
134. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley (1989)
135. Leja, J., Dzojic, H., Gustafson, E.: A novel chromogranine-A promoter-driven oncolytic adenovirus for midgut carcinoid therapy. Clin. Cancer Res. 13, 2455–2462 (2007)
136. Leja, J., Nilsson, B., Gustafson, E., Akerström, G., Oberg, K., Giandomenico, V., Essand, M.: Double detargeted oncolytic adenovirus shows replication arrest in liver cells and retains neuroendocrine cell killing ability. PLOS ONE 5(1), 1–9 (2010)
137. Kaneko, K.: Life: An Introduction to Complex Systems Biology. Springer (2006)

138. Kauffman, L.H.: The Mathematics of Charles Sanders Peirce. Cybernetics & Human Knowing 8(1-2), 79–110 (2001)
139. Manca, V., Franco, G., Scollo, G.: State transition dynamics: basic concepts and molecular computing perspectives. In: [129] ch. 2, pp. 32–55 (2005)
140. Manca, V.: An Outline of MP Modeling Framework. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) CMC 2012. LNCS, vol. 7762, pp. 47–55. Springer, Heidelberg (2013)
141. Marchetti, L., Mitrea, A., Kruger, H., Draghici, S., Manca, V., Bollig-Fisher, A.: Modeling the transcription effects of HER2 oncogene signaling and a role for E2F2 in breast cancer (submitted for publication, 2013)
142. Maturana, H., Varela, F.J.: Autopoiesis and Cognition: The Realization of the Living. Reidel, Boston (1980)
143. Mavelli, F.: Ribocell Modeling. In: Proc. of the Artificial life XII Conference (2010)
144. Mironov, A.S., Gusarov, I., Rafikov, R., Lopez, L.E., Shatalin, K., Kreneva, R.A., Perumov, D.A., Nudler, E.: Sensing small molecules by nascent RNA: a mechanism to control transcription in bacteria. Cell 111(5), 747–756 (2002)
145. von Neumann, J.: The Computer and the Brain, 2nd edn. Yale University Press (2000)
146. Ohler, U.: Computational promoter recognition in eukaryotic genomic DNA. PhD Thesis, Technische Facultät der Universität Erlagen-Nürnberg (2001)
147. Prigogine, I., Stengers, I.: Entre Temps et l'éternité. Librairie Arthème Fayard, Paris (1988)
148. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer (2006)
149. Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH 1987, Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34. ACM (1987)
150. Rosenblueth, A., Wiener, N., Bigelow, J.: Behavior, purpose, and teleology. Philosophy of Science 10, 18–24 (1943)
151. Salmena, L., Poliseno, L., Tay, Y., Kats, L., Pandolfi, P.P.: A ceRNA Hypothesis: The Rosetta Stone of a Hidden RNA Language? Cell 146, 353–358 (2011)
152. Stano, P., Rampioni, G., Carrara, P., Damiano, L., Leoni, L., Luisi, P.L.: Semi-synthetic minimal cells as a tool for biochemical ICT. BioSystems 109, 24–34 (2012)
153. Stormo, G.D., Ji, Y.: Do mRNAs act as direct sensors of small molecules to control their expression? Proc. Natl. Acad. Sci. U.S.A. 98(17), 9465–9467 (2001)
154. Tay, Y., Kats, L., Salmena, L., Weiss, D., Tan, S.M., Ala, U., Karreth, F., Poliseno, L., Provero, P., Di Cunto, F., Lieberman, J., Rigoutsos, I., Pandolfi, P.P.: Coding-Independent Regulation of the Tumor Suppressor PTEN by Competing Endogenous mRNAs. Cell 147, 344–357 (2011)
155. Thom, R.: Stabilité Structurelle et morphogénèse. Benjamin (1972)
156. Thom, R.: Semiophysics: a sketch. Addison-Wesley (1990)
157. Toth, K., Dhar, O., Wold, W.S.: Oncolytic (replication-competent) adenoviruses as anticancer agents. Expert Opin. Biol. Ther. 10, 353–368 (2010)
158. Turing, A.M.: The Chemical Basis of Morphogenesis. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237(641), 37–72 (1952)
159. Wechler, R., Russel, S.J., Curiel, D.T.: Engineering targeted viral vectors for gene therapy. Nat. Rev. Genet. 8, 573–587 (2007)
160. Wiener, N.: Cybernetics: or Control and Communication in the Animal and the Machine. The MIT Press, Cambridge (1948)
161. Wolfram, S.: Theory and Application of Cellular Automata. Addison-Wesley (1986)
162. Wolfram, S.: A new Kind of science. Wolfram Media (2002)

163. Wolpert, L.: Developmental Biology: A Very Short Introduction. Oxford University Press (2011)
164. Gold, L., Brown, D., He, Y., Shtatland, T., Singer, B.S., Wu, Y.: From oligonucleotide shapes to genomic SELEX: Novel biological regulatory loops. Proc. Natl. Acad. Sci. U.S.A. 94(1), 59–64 (1997)

References for Chapter 5

165. Acheson, D.: From Calculus to Chaos. Oxford University Press (1997)
166. Argand, J.R.: Essai sur une manière de représenter les quantités imaginaires dans les constructions gréomrétriques. Annales de Mathématiques, Paris (1806)
167. Boyer, C., Merzbach, U.: A History of Mathematics, 2nd edn. Wiley (1989)
168. Brin, M., Stuck, G.: Introduction to Dynamical Systems. Cambridge University Press (2002)
169. Balakrishnan, V.K.: Introductory Discrete Mathematics. Dover Publications (1991)
170. Chabert, J.L.: A History of Algorithms. Springer (1999)
171. Courant, R., Robbins, H.: What is Mathematics, 2nd edn. Oxford University Press (1996)
172. Crossley, J.N., et al.: What is mathematical logic. Oxford University Press (1972)
173. Martin Davis, M. (ed.): The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable problems and Computable Functions. Raven Press (1965)
174. Fränkel, A.A.: Abstract Set Theory. North-Holland (1976)
175. Gödel, K.: On Formally Undecidable Propositions of Principia Mathematica and Related Systems (based on the original paper of 1931, in German). Dover (1962)
176. Grätzer: First Step In Latex. Springer (1999)
177. Halmos, P.R.: Naive set theory. Springer (1960)
178. Dunham, W.: Euler: The Master of Us All. The Mathematical Association of America (1999)
179. Kaplan, R., Kaplan, E.: The Art of the Infinite. Oxford University Press (2003)
180. Kline, M.: Mathematical Thought from ancient to modern times, vol. 2. Addison Wesley (1972)
181. Kurka, P.: Topological and Symbolic Dynamics. Société Mathématique de France (2003)
182. Luenberger, D.G.: Introduction to Dynamic Systems. Theory, Models, and Applications. John Wiley & Sons (1979)
183. May, R.B.: Simple mathematical models with very complicated dynamics. Nature 261, 459–467 (1976)
184. Manca, V.: Formal Logic. In: Webster, J.G. (ed.) Encyclopedia of Electrical and Electronics Eng., vol. 7, pp. 675–687. John Wiley & Sons (1999)
185. Kalish, D., Montague, R.: Logic: Techiniques of formal reasoning. Harcourt, Brace & World. Inc. (1964)
186. Péter, R.: Playing with Infinity (Hungarian edition 1957). Dover (2010)
187. Pólya, G.: How to solve it, 2nd edn. Penguin Books (2010)
188. Smoryński, C.: Logical Number Theory. Springer (1991)
189. Smullyan, R.M.: First-Order Logic. Dover Publications (1995)
190. Toeplitz, O.: The Calculus: A Genetic Approach. German edition (1949) edited by Kothe, G. The University of Chicago Press (2007)
191. Wirsching, G.J.: The Dynamical System Generated by the 3n + 1 Function. Lecture Notes in Mathematics, vol. 1681 (1998)

192. van der Waerden, B.L.: A History of Algebra. Springer, Berlin (1985)
193. Whitehead, A.N.: An Introduction to Mathematics. Oxford University Press (1948) (1st edn. 1911)

References for Chapter 6

194. Aspray, W. (ed.): Computing before computers. Iowa State University Press, Ames (1990)
195. Berstel, J.: Axel Thue's papers on repetitions in words: a translation. Institut Blaise Pascal, Université Pierre et Marie Curie, Paris (1994)
196. Chomsky, N.: Three models for the description of language. IRE Transactions on Information Theory 2, 113–124 (1956)
197. Chomsky, N.: On certain formal properties of grammars. Information and Control 2(2), 137–167 (1959)
198. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons (1991)
199. Davis, M.: Computability and Unsolvability. Dover Publications (1982)
200. Luenberger, D.G.: Information Science. Princeton University Press (2006)
201. Seymour Ginsburg, S.: The mathematical theory of context free languages. McGraw-Hill (1966)
202. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison-Wesley (2006)
203. Kleene, S.C.: Representation of Events in Nerve Nets and Finite Automata. In: Shannon, C., John McCarthy, J. (eds.) Automata Studies. Princeton University Press (1956)
204. Manca, V.: Logical String Rewriting. Theoretical Computer Science 264, 25–51 (2001)
205. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants (The Virtual Laboratory). Springer (1990)
206. Minsky, M.L.: Computation: finite and infinite machines. Prentice Hall (1967)
207. von Neumann, J.: First Draft of a Report on the EDVAC. Moore School of Electrical Eng. Univ. of Pennsylvania (1945); IEEE Annals of the History of Computing 15(4), 27–75 (1993)
208. Post, E.L.: Finite Combinatory Processes - Formulation 1. Journal of Symbolic Logic 1, 103–105 (1936)
209. Post, E.L.: Formal Reductions of the General Combinatorial Decision Problem. American Journal of Mathematics 65, 197–215 (1943)
210. Salomaa, A.: Formal Languages. Academic Press (1973)
211. Salomaa, A.: Jewels of Formal Language Theory. Computer Science Press (1981)
212. Shannon, C.E.: A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379–423, 623–656 (1948)
213. Rozenberg, G., Salomaa, A. (eds.): Handbook of formal languages. Springer (1997)
214. Turing, A.M.: On computable numbers with application to the entscheidungsproblem. Proc. London Math. Soc. 2(42), 230–265 (1936)

References for Chapter 7

215. Actzel, A.: Chance. High Stakes (2006)
216. Aczel, A.D., Sounderpandian, J.: Complete Business Statistics. McGraw Hill, International Edition (2006)
217. Aigner, M.: Discrete Mathematics. American Mathematical Society (2007)

218. Conway, J.H., Guy, R.K.: The Book of Numbers. Springer (1996)
219. Durbin, R., et al.: Biological Sequence Analysis. Cambridge University Press (1998)
220. Even, S.: Graph Algorithms. Computer Science Press (1979)
221. Feller, W.: An Introduction to Probability Theory and Its Applications, 3rd edn., vol. 1. John Wiley and Sons (1968)
222. Fisher, R.A.: On The Mathematical Foundation of Theoretical Statistics. Transactions of the Royal Society of London 222, 309–368 (1922)
223. Kirkwood, B., Sterne, J.: Essential Medical Statistics. Wiley-Blackwell (2003)
224. Knuth, D.E.: The Art of Computer Programming: Fundamental Algorithms. Addison Wesley (1968)
225. Luenberger, D.G.: Optimization by Vector Space Methods. John Wiley & Sons (1969)
226. Manca, V.: Enumerating Membrane Structures. In: Corne, D.W., Frisco, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2008. LNCS, vol. 5391, pp. 292–298. Springer, Heidelberg (2009)
227. Manca, V.: A Recurrent Enumeration of Free Hypermultisets. In: Kelemen, J., Kelemenová, A. (eds.) Computation, Cooperation, and Life. LNCS, vol. 6610, pp. 16–23. Springer, Heidelberg (2011)
228. Mlodinow, L.: The Drunkard's Walk. Knopf Doubleday Publishing Group (2009)
229. Pearson, K.: Notes on the History of Correlation. Biometrika 13(1), 25–45 (1920)
230. Poincaré, J.H.: La science et l'hypothèse. Flammarion, Paris (1968)
231. Prüfer, H.: Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. 27, 742–744 (1918)
232. Otter, R.: The number of trees. The Annals of Mathematics, 2nd Ser. 49(3), 583–599 (1948)

Index