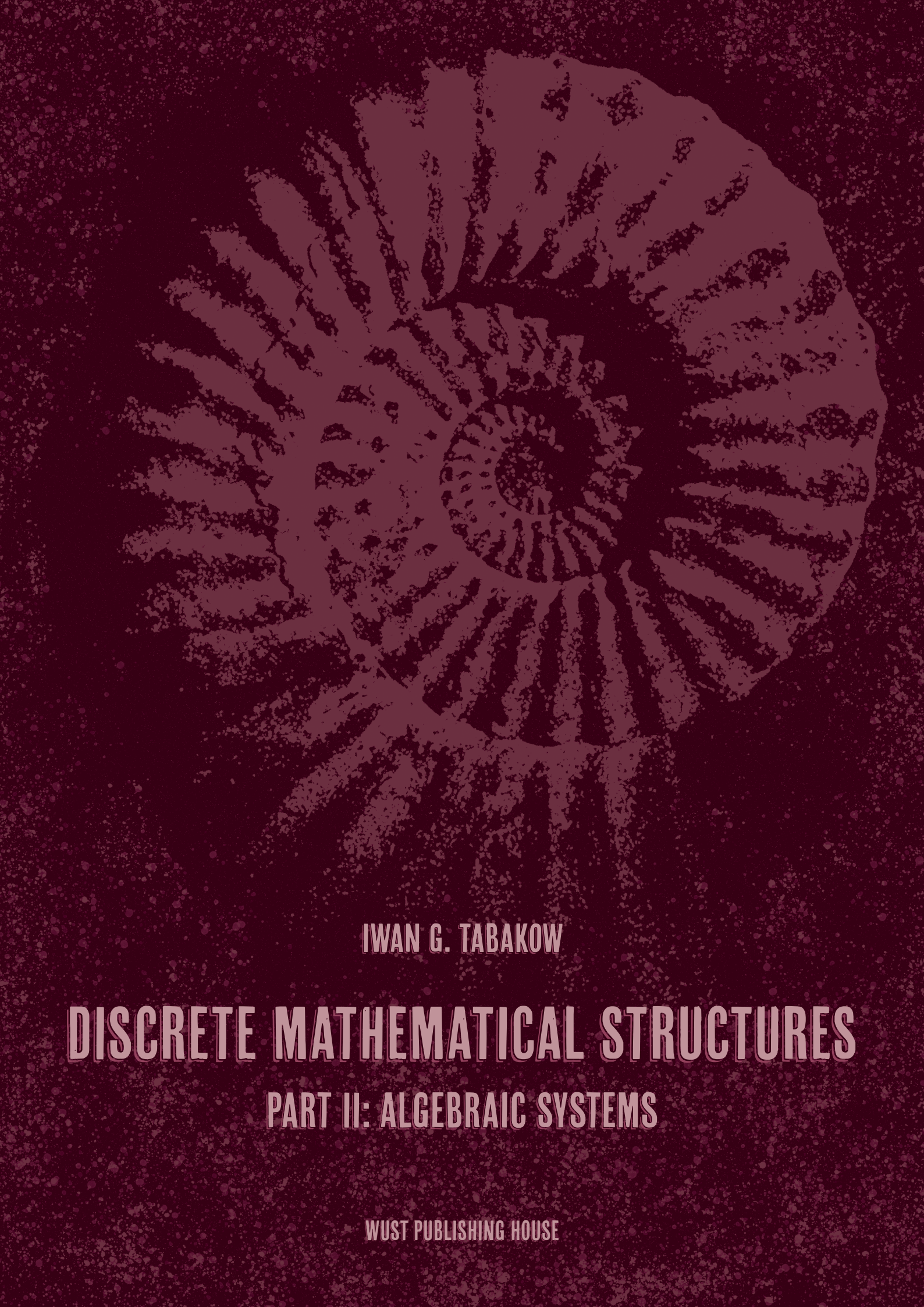


IWAN G. TABAKOW

DISCRETE MATHEMATICAL STRUCTURES

PART II: ALGEBRAIC SYSTEMS

WUST PUBLISHING HOUSE

DISCRETE MATHEMATICAL STRUCTURES

Part II: Algebraic Systems

Iwan G. Tabakow

Wrocław University of Science and Technology
Poland

WUST Publishing House
Wrocław 2022

Cover design
Marcin Zawadzki

Printed in the camera ready form.

WUST PUBLISHING HOUSE
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.edu.pl, ksiegarnia.pwr.edu.pl
e-mail: oficwyd@pwr.edu.pl, zamawianie.ksiazek@pwr.edu.pl

ISBN 978-83-7493-221-9

Print and binding: beta-druk, www.betadruk.pl

In memory of my parents:
Nikolina and Georgi.
In memory of my teachers:
prof. dr. Ludwik Borkowski
and prof. dr. Jerzy Bromirski

Contents

Preface 7

The used designations 9

Chapter I. Operations and algebraic systems 11

1. Operations 11
   - 1.1. Two-argument operations 11
   - 1.2. Boolean, multiple valued and fuzzy-logic operations 14
2. Algebraic systems 18
   - 2.1. Algebraic system definition 18
   - 2.2. Algebraic systems with one binary operation 20
   - 2.3. Deletion rule 21
   - 2.4. Lattices 22
   - 2.5. Multiple valued and fuzzy algebraic systems 29

Chapter II. Homomorphisms, direct products and free algebraic systems 34

3. Homomorphisms 34
   - 3.1. Epi-, mono-, iso-, endo- and automorphisms 34
   - 3.2. Congrujencies and quotient algebraic systems 36
   - 3.3. Finite direct products 39
   - 3.4. Free algebraic systems 41
4. Applications 42
   - 4.1. Grammars and sequential machines 42
   - 4.2. Computability and recursion 43
   - 4.3. Graph theory and Petri nets 43
   - 4.4. Combinatorial analysis and probability theory 44
   - 4.5. Number theory, Markov's chains, coding 44
   - 4.6. Algorithm complexity 45

Conclusions 46

References 46

"There is no any mathematical abstraction which would not be applicable at an earlier time or later time in practice".
Nikolai Ivanovich Lobachevsky (1792 – 1856)

Preface

Discrete mathematical structures (in short: *discrete structures*), in particular such as mathematical logic and set theory, algebraic systems, formal languages, automata theory, graphs, number theory, coding theory, combinatorial analysis, discrete probability theory, Petri nets and so on, underpin a large amount of modern computer science. Discrete structures became a very large and dynamic science. Unfortunately, the speedy developments and knowledge in this area makes impossible the presentation of all notions, definitions and applications used here. This part is an extension of the previous one (i.e. a supplement / a separate work) and it is related to algebraic systems (considered as *discrete mathematical structures*).

Some basic notions concerning: operations and algebraic systems, lattices, multiple valued and fuzzy algebras, homomorphisms of algebraic systems (i.e. epimorphism, monomorphism, isomorphism, endomorphism and automorphism), congruencies, quotient algebraic systems, finite direct products of algebraic systems and free algebraic systems, grammars and sequential machines, algorithms, computability, recursion, graph theory and Petri nets, combinatorial analysis, probability theory, Markov's chains, number theory, information, coding and algorithm complexity, are briefly considered in this part*.

Several parts of this work were presented during my lectures at the Institute for Mechanical and Electrical Engineering in Sofia, now known as TU Sofia, Bulgaria and also at the Wroclaw University of Technology in Wroclaw, Poland. A preliminary version of this study was realised in accordance with some research projects, e.g. such as *Z0802-331557-W0800*, *Z0802-341763-W0800*, etc., during my stay in Wroclaw.

I would like to thank my wife for her countless patience and love during the writing of the manuscript of this book.

This open access work is firstly addresses to the (advanced) computer science students, but may be useful for any researcher who is interested in the above given area. Any suggestions or other comments related to this work are well come. To all such remarks I would be grateful.

Iwan.G.Tabakow (retd. Professor)

* 'The problem-solving emphasis of computer science borrows heavily from the areas of mathematics and logic. Faced with a problem, computer scientists must first formulate a solution. This method of solution, or *algorithm* as it is often called in computer science, must be thoroughly understood before the computer scientists make any attempt'.

The used designations

The used names for the primitive and/or derived rules given below are in accordance with the Łukasiewicz´s symbols of negation, conjunction, disjunction implication, and equivalence denoted by N, K, A, C, and E, respectively (introduced in the *parenthesis-free notation* called also *Polish notation*: Jan Łukasiewicz 1878 – 1956). Some commonly used symbols are given in parentheses. Other designations and/or abbreviations are the same as in (Słupecki J. and Borkowski L. 1967)*.

~	symbol of *negation*, called also *logical inversion* or *logical not* (called also e.g. '*quantum negation*', i.e. '*orthocomplement*' in quantum logic systems or '*linear negation*' in linear logic systems, etc.). Another used designations: $\neg$, ´ , ¯ , $^{\perp}$ (e.g. $p^{\perp}$), *Not*, etc. The used symbol ' ´ ' may be also used as a citation of designations or formulae, e.g. ´∼´ to denote the set *equinumerosity relation* or ´A ∩ B´ etc., depending on the context ;
∧	symbol of *conjunction* (logical multiplication, logical and: & , · , ∩, ⊓, *And*) or *weak conjunction* (many-valued logics) ;
∨	symbol of *disjunction* (logical sum, logical alternative, join: + , ∪ , | , /, ⊔, *Or*) or *weak disjunction* (many-valued logics) ;
⇒	symbol of (the material) implication (→ , ⊃ , I) ;
⇔	symbol of equivalence (co-implication, ≡, iff , ~ , ↔ , E: ' ≡ ' may also denote the congruence modulo m relation on the set of integers or a linear logic equivalence or also *symmetric* (called also *symmetrical*} *set difference operation* , depending on the context ;
∪ , ∩, − , ÷ , ´	the set union (set sum), set intersection, set difference and symmetric (or: symmetrical) set difference operations (another designation for set difference: ' \ ' , e.g. X \ Y, instead of X − Y), provided there is no ambiguity by X´ is denoted the *complement* of X (another designation: $\overline{X}$): in a similar way and for convenience ' ´ ' may be also used to denote some element, e.g. x´ or algebraic operation, e.g. o' (depending on the context) ;
× , X^n	Cartesian product of sets, $X^n =_{df} X \times X \times ... \times X$, n times, $n \geq 2$ (René Descartes 1596 – 1650, Latinised: Renatus Cartesius) ;
$a \in A$, $a \notin A$	the *membership relation*'is *element of* ' : a is element of A (or belongs to A), a is not an element of A ;
∅	*empty* (or *null*) set ;
⊆ , = , ≠	the *set inclusion*, *equality* and *nonequality,* respectively ;
{a, b, … , z}	a finite set of *elements*: a, b, … , z ;
(x,y) , $(x_1, x_2, \ldots . x_n)$	*ordered pai*r, having as a *first element* x and as a *second element* y and ordered system having (more than two but) a finite number of elements ;

* Słupecki Jerzy (1904 – 1987), Borkowski Ludwik (1914 – 1993).

$\{a_i / \ i \in I\}$, $\{A_i / \ i \in I\}$	a *set of elements* a_i and a *family of sets* A_i $(i \in I)$;
$\{ x \ / \ \varphi(x) \}$	the set of all x satisfying the *propositional function* $\varphi(x)$;
$\underset{x}{\exists} \varphi(x)$, $\underset{x}{\forall} \varphi(x)$	the *existential* and the *universal quantifiers* related to $\varphi(x)$;
$\mathbb{N}$, $\mathbb{N}_E$, $\mathbb{N}_O$, $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{R}_{++}$, $\mathbb{C}$	the sets of *natural numbers* $\mathbb{N} =_{df} \{1, 2, ...\}$, the subsets of *even* and *odd natural numbers,* i*nteger numbers*, *nonnegative integer numbers*, *rational numbers*, *real numbers*, *nonnegative real numbers* , *positive real numbers* and *complex numbers* ;
$\lvert X \rvert$	the *cardinality* of set X (denoted also by: $\overline{\overline{X}}$ (Cantor's designation: G.F.L.P. Cantor 1845 – 1918) , card(X), nc(X), #(X) or #X ;
$\aleph_0$	the cardinality of $\mathbb{N}$;
$\circ$, ω	a binary operation and some mapping ;
$\bigcup_i X_i$, $\bigcap_i X_i$	the *generalised set union* and *set intersection* operations, respectively* .
$\sqsubset$	*proper inclusion* between two algebraic systems ;
$\subseteq$, $\subsetneq$, $\nsubseteq$	the *set inclusion* and the *proper set inclusion* (or *strict set inclusion*, denoted also by $\subset^{\dagger}$) binary relations. $X \nsubseteq Y \Leftrightarrow_{df} \sim (X \subseteq Y)$: if $X \subseteq Y$ then X is a *subset* of Y or equivalently, Y is a *superset* of X ;
(x,y)	an *ordered pair* (*2-tuple* in short "*couple*": means "pair"), denoted also by <x,y>, Kuratowski K. (1896 – 1980) ;
□	′end of proof ′ (of an example, algorithm, or another formalised text) ;
wrt, iff	with respect to, if and only if ;
'text'	citation of a text, e.g. 'a variety of problems that can be solved by … ' .

For any other designations, see Part I of this book.

---

* Usually I coincides with the set of natural numbers. And hence, for convenience, e.g. instead of $\bigcup_{i \in I} X_i$, it is also used $\bigcup_i X_i$ (assuming that I is known) . And hence, provided there is no ambiguity and for simplicity, it is assumed that the used index $i \in I \subseteq \mathbb{N}$ (in a similar way: $\bigcap_i X_i$) .

† This set operation may also denote usual set inclusion, e.g. see (Słupecki J. and Borkowski L. 1967).

I. Operations and algebraic systems

The used below basic notions and definitions are mainly under (Kerntopf P.[*] 1967). Some other considerations are also presented, in particuar concerning *Boolean*, *multiple valued* and *fuzzy algebraic systems*. The notion of an operation is first presented.

1. Operations

Any mathematical operation is considered as a function taking zero or more input values (known as operands) to a well-defined output value[†]. The notion of an n-argument operation is first given. Some properties concerning binary operations are next presented.

1.1. Two-argument operations

Let A be a given set. The following definition is introduced (Kerntopf P. 1967).

Definition 1.1 (*n-argument algebraic operation*)

An *n-argument algebraic operation* defined in A (where n is an arbitrary *cardinal number*) is the mapping $\omega : A^n \rightarrow A$[‡].

The following notation was also used: $\omega : < a_1, a_2, \dots, a_n > \rightarrow \omega(a_1, a_2, \dots, a_n)$, where $a_1, a_2, \dots, a_n \in A$ and $\omega(a_1, a_2, \dots, a_n) \in A$ (i.e. A is *closed*[§] under ω). As an example, the set of natural numbers $\mathbb{N}$ is closed wrt the usual operations of addition and multiplication.

Below we shall assume mainly *two*-argument (provided there is no ambiguity and for simplicity called below *binary*) *algebraic operations*.

Definition 1.2 (*commutative binary operation*)

Let ω be a binary operation defined in A (i.e. n = 2). We shall say that ω is a *commutative binary operation* if:

---

[*] Paweł Kerntopf, born 1938.

[†] *The Free Encyclopaedia, The Wikimedia Foundation, Inc*.

[‡] 'Historically, the concepts of *binary* (n = 2) and *unary* (n = 1) were the first to be considered. *Nullary* (n = 0) operations are fixed elements of the set A ; they are also known as distinguished elements or constants. In the 20th century the concept of an infinitary operation appeared, i.e. a mapping $\omega : A^{\alpha} \rightarrow A$, where α is an arbitrary cardinal number.' *Encyclopedia of Mathematics*, Springer: https://encyclopediaofmath.org/wiki/Algebraic_operation

[§] Performing that operation on members of the set A always produces a member of that set.

$$\forall_{a,b \in A} \; (\omega(a,b) = \omega(b,a)).$$

Definition 1.3 (*associative binary operation*)

The binary operation ω defined in A is an *associative binary operation* if:

$$\forall_{a,b,c \in A} \; (\omega(\omega(a,b),c) = \omega(a,\omega(b,c))).$$

Equivalently, the following additive and multiplicative notasions are also used (for any $a,b,c \in A$):

property	multiplicative notion	additive notion
commutativity	$a \cdot b = b \cdot a$	$a + b = b + a$
associativity	$(a \cdot b) \cdot c = a \cdot \{b \cdot c\}$	$(a + b) + c = a + \{b + c\}$

Figure 1.1

As an example, commutative and associative are the following operations: addition and multiplication defined in $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$.

Example 1.1

Let $a + ib$ and $c + id$ be two complex numbers, where $a,b,c,d \in \mathbb{R}$ and i is the *imaginary unit* ($i^2 = -1$)*. We have:

$$(a + ib) + (c + id) = a + c + i(b + d)$$

and

$$(c + id) + (a + ib) = c + a + i(d + b) = a + c + i(b + d).$$

In a similar way, we can obtain:

$$(a + ib) \cdot (c + id) = ac + iad + ibc + i^2bd = ac + iad + ibc - bd$$

and

$$(c + id) \cdot (a + ib) = ca + icb + iad + i^2bd = ca + icb + iad - bd = ac + iad + ibc - bd \; . \; \square$$

It can be observed that the *subtraction operation* defined in $\mathbb{Z}, \mathbb{R}$ and $\mathbb{C}$ is not associative. Moreover, there are operations which are not commutative or associative: the following example operation was given (Kerntopf P. 1967): $h : < n, m > \rightarrow n^m$ $(n, m \in \mathbb{N})$.

Definition 1.4 (*left-sided and right-sided separate operations*)

Let ω_1 and ω_2 be two binary operations defined in A. We shall say that ω_1 *left-sided separate operation* wrt ω_2 if:

$$\forall_{a,b,c \in A} \; (\omega_1(a, \omega_2(b,c)) = \omega_2(\omega_1(a,b), \omega_1(a,c))))$$

And we shall say that ω_1 *right-sided separate operation* wrt ω_2 if:

* As an example: $(x + iy)^2 = x^2 + 2xyi - y^2$.

$$\forall_{a,b,c \in A} \; (\omega_1(\omega_2(b,c),a) = \omega_2(\omega_1(b,a), \omega_1(c,a))).$$

Example 1.2 (left-sided and right-sided separate operations)

Let ω_1 and ω_2 be the two arithmetic operations of multiplication and addition, denoted by: ‘ · ‘ and ‘ + ‘, respectively. And so, the corresponding left-sided and right-sided separate operations are illustrated below (for any a, b, c $\in$ A):

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$(b + c) \cdot a = b \cdot a + c \cdot a$$

Since arithmetical multiplication is commutative, the obtained last two lines are equivalent. □

There were presented in (Kerntopf P. 1967) various examples concerning the above distributivity property. In particular, the following one.

Example 1.3

Consider the following two binary operations defined in $\mathbb{R}$: $a * b =_{df} a + b + 1$ and $a \odot b =_{df} a + b + a \cdot b$. It was shown that the last binary operation is distributive wrt the first one, but not vice versa: left to the reader. □

Definition 1.5 (*operation closed on a subset of A*)*

Let ω be a k-argument operation defined in A. We shall say that ω is *closed* in $B \subseteq A$ if:

$$\forall_{a_1,\ldots,a_k \in B} \; \omega(a_1, \ldots, a_k) \in B .$$

The following example was given.

Example 1.4

Let $\mathbb{N}_E, \mathbb{N}_O \subseteq \mathbb{N}$ be the subsets of *even* and *odd natural numbers*. The arithmetic operation of addition in $\mathbb{N}_E$ is closed one (but not in $\mathbb{N}_O$) whereas the arithmetic multiplication is closed in the above both subsets. □

Definition 1.6 (*left-, right identity elements*)

The elements $e_L, e_R \in A$ are said to be *left* - and *right identity elements* wrt to the binary operation ω defined in A if:

$$\forall_{a \in A} \; ((\omega(e_L, a) = a) \wedge (\omega(a, e_R) = a)) .$$

Theorem 1.1

Let ω be a binary operation defined in A. Assume this operation has at the same time as identity elements e_L and e_R. Then we have: $e_L = e_R$.

Proof:

We have: $\omega(e_L, e_R) = e_R$ and $\omega(e_L, e_R) = e_L$. □

In accordance with the last theorem: $e_R = e_L = e$ (*the identity element*). Moreover, it follows that any binary operation defined in a given set may have at most one identity element.

---

* Equivalently: a subset of A closed under an operation.

Example 1.5

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{C}$ be the sets of *natural numbers*, *integer numbers*, *real numbers* and *complex numbers*. The identity elements wrt addition and multiplication are 0 and 1: for $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$ and $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$, respectively. □

Definition 1.7 (*left-, right zero elements*, *zero element*)

The elements $z_L, z_R \in A$ are said to be *left* - and *right zero elements* wrt to the binary operation ω defined in A if:

$$\forall_{a \in A} \; ((\omega(z_L, a) = z_L) \wedge (\omega(a, z_R) = z_R)) .$$

Definition 1.8 (*idempotent element*)[*]

The element $a \in A$ is said to be an *idempotent* one wrt to the binary operation ω defined in A if: $\omega(a, a) = a$.

1.2. Boolean, multiple valued and fuzzy-logic operations

'There are only two values, 0 and 1, unlike elementary algebra that deals with an infinity of values, the real numbers. Since there are only two values, a *truth table* is very useful tool for working with *Boolean algebra*[†]' (Plantz R.G.[‡] 2021): here *Raspberry Pi* is 'a tiny and affordable computer' that can be used 'to learn programming through fun, practical projects'. Some considerations related to Boolean algebra operations are cited below. A truth table lists all possible combinations of the values of the corresponding elementary Boolean variables. 'The resulting value of the Boolean operation(s) for each variable is shown on the respective row. Elementary algebra has four operations (*addition*, *subtraction*, *multiplication* and *division*), but Boolean algebra has only the following three operations (Plantz R.G. 2021):

AND	A binary operator. The result is 1 iff both operands are 1, othervise the result is 0.
OR	A binary operator. The result is 1 iff at least one of the two operands is 1, othervise the result is 0.
NOT	A unary operator. The result is 1 if the operand is 0, and vice versa.
Precedence rules: NOT, AND, OR	Corresponding symbols: $'$, $\cdot$, $+$ (denoted also e.g. by: $\sim, \wedge, \vee$ respectively).

Figure 1.2 Basic Boolean operations and precedence rules

'There is a *duality* between the AND and OR operators, i.e. in any equality, the interchange of AND and OR along with the constants 0 and 1 do not change this equality' (as an example, in particular, the interchange of AND and OR in De Morgan's laws[§]).

---

[*] There exist some operations such that they can be applied multiple times without changing the result beyond the initial application. The notion of i*dempotence* was first introduced by Benjamin Peirce (1809 – 1880). See: *The Free Encyclopaedia, The Wikimedia Foundation, Inc*.

[†] George Boole (1815 – 1864)

[‡] Robert G. Plantz, born 1939.

[§] Augustus De Morgan (1806 – 1871)

Some basic Boolean algebra properties for manipulating Boolean expressions were also considered (Plantz R.G. 2021). According to the last work, the considered here properties are presented as theorems and easily proved by using truth tables. The associativity properties and commutativity properties related to AND and OR were first presented.

Let $x,y,z \in \{0,1\}$. The associativity and commutativity properties related to the AND and OR operators are presented as follows (for any $x,y,z \in \{0,1\}$)* :

The *associativity*:

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$

$x + (y + z) = (x + y) + z$

The *commutativity*:

$x \cdot y = y \cdot x$

$x + y = y + x$

The used precedence rules are given in Figure 1.3 below.

Operation	Symbol	Precedence
NOT	′	Highest
AND	·	Middle
OR	+	Lowest

Figure 1.3 Precedence rules of Boolean algebra operators

Another properties, also considered in (Plantz R.G. 2021) are resumed in Figure 1.4 given below below.

Logical operations	Property
AND and OR	Each have *identity value*: $x \cdot 1 = x$ and $x + 0 = x$
AND and OR	Each have an *annulment value*: $x \cdot 0 = 0$ and $x + 1 = 1$
AND and OR	Each have a *complement value*: $x \cdot x' = 0$ and $x + x' = 1$
AND and OR	Each of them is *idempoten*t: $x \cdot x = x$ and $x + x = x$
AND and OR	They are *distributive*: $x \cdot (y + z) = x \cdot y + x \cdot z$, $x + y \cdot z = (x + y) \cdot (x + z)$
AND and OR	Duality: $(x \cdot y)' = x' + y'$ and $(x + y)' = x' \cdot y'$ (*DeMorgan's laws*)

* We have three variables each of them assuming two values and hence: 2^3 , i.e. 8 0 -1- combinations to be checked. As an example see the proof of the following De Morgan's law given in the next considerations: $(x + y)' = x' \cdot y'$ (there are now $2^2 = 4$ 0 -1- combinations to be checked).

NOT	Shows *involution*: $(x')' = x$

Figure 1.4 Basic properties of Boolean algebra operators

Example 1.6 (*distributivity of OR wrt AND*)

The proof of the following equality: $x + y \cdot z = (x + y) \cdot (x + z)$ is illustrated in Figure 1.5 below. It can be observed that the left and the right sides of this equality coincides in any of the $2^3 = 8$, 0 -1- combinations related to x,y and z. □

x	y	z	x+y	x+z	y · z	(x+y) · (x+z)	x + y · z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

Figure 1.5

Example 1.7 (*DeMorgan's laws*)

The proof of DeMorgan's law: $(x + y)' = x' \cdot y'$ is given in Figure 1.6 below. The proof of the second DeMorgan's law, i.e. $(x \cdot y)' = x' + y'$ is similar: left to the reader. □

x	y	x′	y′	x+y	(x+y)′	x′· y′
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Figure 1.6

And finally, we have the following values, called *identity* and *annulment* ones, related to the above two AND and OR operators: $x \cdot 1 = x$, $x + 0 = x$ and $x \cdot 0 = 0$, $x + 1 = 1$. It can be observed that only some of above cited properties are similar as in the elementary algebra.

In general, the above two Boolean operations AND and OR can be interpreted as a particular case of the following (more general) two operations: *minimum* and *maximum* respectively. Some considerations related to *multiple valued operations* are given below.

Many-valued logics are non-classical logics. But they are similar to classical logic because they accept the *principle of truth-functionality*, i.e. the truth of a compound proposition is determined by the truth-values of its

component propositions (and so remains unaffected when one of its component propositions is replaced by another having the same truth-value). Here a fundamental fact is that the above logics do not restrict the number of truth-values to only two, The main systems of many-valued logic often come as families, which comprise uniformly defined finite-valued as well as infinite-valued systems such as: Łukasiewicz's logics, Gödel's logics, t-norm related systems, 3-valued systems, product systems, Dunn/Belnap's 4-valued system, etc. (Gottwald S. 2000)*. As an example, Łukasiewicz†'s systems $Ł_m$ and $Ł_\infty$ were introduced by means of the following two truth degree sets: $W_m =_{df} \left\{ \frac{k}{m-1} / k \in \{0,1,...,m-1\}, m \geq 2 \right\}$ and $W_\infty =_{df} [0,1] \subseteq \mathbb{R}_+$ (the set of all nonnegative real numbers),

&	strong conjunction: $p \ \& \ q =_{df} \max\{0, p + q - 1\}$
$\wedge$	weak conjunction: $p \wedge q =_{df} \min\{p,q\}$
$\veebar$	strong disjunction: $p \veebar q =_{df} \min\{1. p + q\}$
$\vee$	weak disjunction: $p \vee q =_{df} \max\{p,q\}$
$\sim$	negation: $\sim p =_{df} 1 - p$
$\Rightarrow$	implication: $p \Rightarrow q =_{df} \min\{1, 1 - p + q\}$

Figure 1.7 Logical connectives

The equality relations between the strong and weak conjunction are illustrated in the next figure below.

$p \ \& \ q = p \wedge q$	for $p = 0$ or $q = 0$
$p \ \& \ q = p \wedge q$	for $p + q > 1$ and $\max\{p,q\} = 1$
$p \ \& \ q \neq p \wedge q$	for $p + q \leq 1$

Figure 1.8 The strong and weak conjunction relations

Example 1.8 (*equality relations*)

Let $L =_{df} \max\{0, p + q - 1\}$ and $R =_{df} \min\{p,q\}$. where $p, q \in [0,1]$. We have:

(1) Assume that: $p = 0$ or $q = 0$. As an example, for $p =_{df} 0$ (since $q \leq 1$) we can obtain: $L = 0 = R$. And hence: $p \ \& \ q = p \wedge q$.

(2) Let $p, q > 0$, $p + q > 1$ and $\max\{p,q\} = 1$. We have: $L = R$. As an example, for $p = 1$ and $q = 0.6$ we can obtain: $L = 0.6 = R$.

(3) Assume now $p + q \leq 1$: e.g. $p = 0.5$, $q = 0.4$. We have: $L = \max\{0, 0.5 + 0.4 - 1\} = \max\{0, -0.1\} = 0$ and $R = 0.4$. And hence: $L \neq R$. □

It is assumed that the symbol of negation binds more strongly than the remaining symbols. As an example, an algebraic interpretation of *De Morgan´s laws* is given below (the corresponding proofs are left to the reader: here the equals sign is used to denote equality between the corresponding left and right sides).

(i) $\sim(p \ \& \ q) = \sim p \veebar \sim q$,

* See Part I of this book.

† Jan Łukasiewicz (1878 – 1956). The first Łukasiewicz's system (a ternary logic system with $W_3 =_{df} \{0, 1/2, 1\}$) was given in 1918: usually 0, 1/2 and 1 are denoted by the logical constants F, U (unknown) and T, respectively.

(ii) $\sim(p \veebar q) = \sim p \ \& \sim q$,

(iii) $\sim(p \wedge q) = \sim p \vee \sim q$ and

(iv) $\sim(p \vee q) = \sim p \wedge \sim q$.

'The b*asic fuzzy propositional logic* is a relatively young discipline, both serving as a foundation for the fuzzy logic in a broad sense and of independent logical interest, since it turns out that strictly logical investigation of this kind of logical calculus can go rather far (Hájek P. 1998). It is broadly accepted that t-norms (dually, t-conorms) are possible truth functions of *fuzzy conjunction* (of *fuzzy disjunction*). The best-known candidate for *fuzzy negation* is the *Łukasiewicz's negation* $x' =_{df} 1 - x$. However, some other notions, e.g. such as *Sugeno's fuzzy negation* or *Yager's fuzzy negation* are also applicable (Bronstein I.N. et al. 2001). The *fuzzy implication* connective is sometimes disregarded but is of fundamental importance for fuzzy logic in the narrow sense. A straightforward but logically less interesting possibility is to define implication from disjunction and negation or conjunction and negation using the corresponding theses of classical logic T 1.15 and T 1.19, respectively (see Subsection 1.3). Such implications are called *S-implications*[*]. In fact, more useful and interesting are the so-called *R-implications* (any such implication is specified as a residuum with respect to the used t-norm)'. A more formal treatment is omitted: see Part I of this book: Fuzzy logic, p.72.

2. Algebraic systems

The notion of an algebraic system is 'one of the basic mathematical concepts and its general theory has been developed in depth. This was done in the 1950s, and the work took place on the interface between algebra and mathematical logic[†].' This notion is first presented. Systems with one binary operation are next considered. The law of deletion is also given. And finally, such algebraic systems as: Boolean, multiple valued and fuzzy algebraic systems are also considered.

2.1. Algebraic system definition

The notion of an algebraic system is defined as follows (Kerntopf P. 1967).

Definition 2.1 (*algebraic system*)

Algebraic system is an ordered system $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$, where: A is an arbitrary set, known as the *set of system elements*; $a_1, a_2, \dots, a_n$ are some distinguished elements of A, i.e. the so-called *constants* of the above algebraic system (the last set of constants may be empty) and $o_j : A^{nj} \rightarrow A$ (j = 1, 2, … , m) are the *system operations*.

The above algebraic system α is said to be *finite* if the set of system elements A is finite[‡].

Definition 2.2 (*algebraic system type*)

---

[*] In accordance with T 1.19 and CE (the law of transposition or contraposition of equivalence) we have: $p \Rightarrow q \Leftrightarrow \sim(p \wedge \sim q)$. Let e.g. $x' =_{df} 1 - x$ be the Łukasiewicz's fuzzy negation and $x \otimes y =_{df} \min\{x,y\}$ be the Zadeh's t-norm. Hence, the following S-implication can be obtained: $x \Rightarrow y =_{df} 1 - \min\{x, 1 - y\} = \max\{1 - x, y\}$ (the logical value of this implication: the proof of the last equality is left to the reader). Obviously, it is possible also the use of other fuzzy negations and / or t-norms. It can be observed that sometimes the above two S- and R-implications may coincide, e.g. in $Ł_\alpha$-BL (and hence in Ł-BL, assuming Yager's fuzzy negation: see Corollary 2.4 of this subsection).

[†] Encyclopedia of Mathematics: https://encyclopediaofmath.org/wiki/Algebraic_system.

[‡] The above distinguished elements are sometime treated as *0 - argument operations*. Moreover, there are sometime introduced also relations defined in A or partial operations: left to the reader;

The *type of* α , in short $T(\alpha)$, is defined as follows: $(0, 0, \dots , 0, n_1, n_2, \dots , n_m)$.

In accordance with the last definition: the number of 0's corresponds to the number of constants and n_j - to the number of arguments of o_j $(j = 1, 2, \dots , m)$.

Let $T(\alpha)$ be *the type of* α and β be another algebraic system. We shall say that α and β are *similar algebraic systems* iff $T(\alpha) = T(\beta)$.

Definition 2.3 (*algebraic subsystem*)

Let $\alpha = (A ; a_1, a_2, \dots ,a_n ; o_1, o_2, \dots, o_m)$ be given and $\beta =_{df} (B : a_1, a_2, \dots ,a_n ; o_1', o_2', \dots, o_m'\}$ be another algebraic system. The system β is said to be a *subsystem* of α , if: $B \subseteq A$, o_j' are *restricted* to (the subset)* $B \subseteq A$ and o_j' are closed in B (for any $j \in \{1, 2, \dots , m\}$).

It can be observed that any subsystem of α is similar to α and having the same constants as in α . By $\Re =_{df} \{\alpha_k\}_{k\in\Delta}$ we shall denote below the family of all algebraic subsystems related to α . The following two definitions were also introduced (Kerntopf P. 1967).

Definition 2.4 (*algebraic system inclusion*)

Let $\alpha_s , \alpha_t \in \Re$. We shall say that the subsystem α_s is *included* in α_t , i.e. $\alpha_s \sqsubset \alpha_t$ if $A_s \subset A_t$.

Definition 2.5 (*algebraic subsystem intersection*)

Let $B \neq \varnothing$ and $\Re_B =_{df} \{\alpha_p\}_{p \in \Gamma \subseteq \Delta} \subset \Re$ be the family of all subsystems of α such that $B \subset A_p$. The *intersection* of the subsystems belonging to $\Re_B$ is defined as follows:

$$(\bigcap_{p \in \Gamma} A_p ; a_1, a_2, \dots , a_n ; \underline{o}_1, \underline{o}_2, \dots , \underline{o}_m) \in \Re .$$

For simplicity, the last intersection was denoted by $\bigcap_{p \in \Gamma} \alpha_p$: is the *smallest subsystem* of α whose set of elements contains B (the *set of generators*).

Example 2.1 (*algebraic subsystems*)

Consider the following algebraic system $\alpha =_{df} (\{a, b, c\}; o)$, where ' o ' is a binary operation defined in accordance with Figure 2.1 given below.

o	a	b	c
a	a	b	a
b	a	b	b
c	a	b	c

Figure 2.1 An example algebraic system

Subsystems of α are the following ones (for $k = 0, 1, \dots , 7\}$: $\alpha_k =_{df} (A_k ; o^{(k)})$, where: $A_0 = \varnothing$, $A_1 = \{a\}$, $A_2 = \{b\}$, $A_3 = \{c\}$, $A_4 = \{a,b\}$, $A_5 = \{a,c\}$, $A_6 = \{b,c\}$ and $A_7 = \{a,b,c\}$: corresponds to the sum $\sum_{k=0}^{3} \binom{3}{k}$. □

* i.e: $dom(o_j') =_{df} dom(o_j) \cap B$

In accordance with Example 2.1, e.g. the algebraic subsystem $\alpha_6 =_{df} (A_6 ; o^{(6)})$ is illustrated in Figure 2.2 below: by $o^{(6)}$ it is denoted the truncation of ' o ' to A_6. Obviously: $\alpha_6 \subset \alpha$.

$o^{(6)}$	b	c
b	b	b
c	b	c

Figure 2.2 The algebraic subsystem α_6.

2.2. Algebraic systems with one binary operation

Definition 2.6 (*groupoid*)

Let α be an algebraic system. This system is said to be a *groupoid* if $T(\alpha) = (2)$[*].

Definition 2.7 (*semigroup*)

A groupoid with an associative binary operation.

Definition 2.8 (*monoid*)

A semigroup with an *identity element*[†].

Definition 2.9 (*inverse element*)

Let o be a binary operation defined in A, having an identity element e and $a \in A$. We shall say that a' is an *inverse element* to a wrt o if:

$$\exists_{a' \in A} \; (a \circ a' = e = a' \circ a) .$$

Theorem 2.1

Let o be a binary operation defined in A, having an identity element e. Assume that o is an associative operation. Then, there is no an element of A having more than one inverse element.

Proof:

The proof is indirect. Let a' and a" ($a' \neq a''$) be two different elements of A. Since o is associative, we can obtain: $(a' \circ a) \circ a'' = e \circ a'' = a''$ and $a' \circ (a \circ a'') = a' \circ e = a'$. And hence: $a' = a''$. This is a contradiction with our assumption. □

Theorem 2.2

Let o be an associative binary operation defined in A, having an identity element e. Assume that $a', b' \in A$ are the corresponding inverse elements to a and b in A wrt o. Then $b' \circ a'$ is the inverse element to $a \circ b$. □

In accordance wich T 2.2, it is necessary to be proved the following implication:

---

[*] As an example, a groupoid is the system $(\mathbb{R} , +)$, where ' + ' denotes the usual operation of addition (in fact, the last grupoid is a *group*: having an *associative binary operation*, *identity* and *inverse elements*).

[†] Called also '*neutral element*' of a binary operation: an element of the set, leaving unchanged any other element of this set under operation with the first one, e.g. $x + 0 = x = 0 + x$ or $x \cdot 1 = x = 1 \cdot x$ (here '+' and '·' denotes *arithmetical addition* and *multiplication*, respectively).

$(a \circ a' = e = a' \circ a) \wedge (b \circ b' = e = b' \circ b) \Rightarrow ((b' \circ a') \circ (a \circ b) = e = (a \circ b) \circ (b' \circ a'))$

The proof of the last implication is based on Definition 2.9: left to the reader.

Definition 2.10 (*group*)*

Monoid having an inverse element a' for any $a \in A$.

Definition 2.11 (*inverse operation*)

$$\forall_{a,\, b \in A} \quad \exists!_{x} \quad \exists!_{y} \ ((a \circ x = b) \wedge (y \circ a = b)) \Rightarrow (\circ \text{ has an } \textit{inverse operation} \text{ in } A).$$

The notion of a group can be equivalently introduced by means of one of the next two definitions (Kerntopf P. 1967).

Definition 2.12 (*group*)

Monoid with a binary operation having an inverse operation.

Definition 2.13 (*group*)

An algebraic system $\alpha =_{df} (A ; e ; o_1 , o_2)$ having a type $T(\alpha) = (0, 1, 2)$ and satisfying the following two conditions:

(i) The system $(A ; e ; o_2)$ is a monoid and

(ii) $\forall_{x \in A} \ (o_2(x , o_1(x)) = e = o_2(o_1(x), x))$.

The above considered algebraic systems (see: Definitions: 2.6, 2.7, 2.8 and 2.10) are said to be *abelian*† ones if the corresponding binary operation used here is commutative.

2.3. Deletion rule

Definition 2.14 (*left-sided deletion rule*)

$$\forall_{a,b,c \in A} \ ((a \circ b = a \circ c) \Rightarrow (b = c))$$

In a similar way there is introduced the notion of a *right-sided deletion rule* and a *two-sided deletion rule* (in short: *deletion rule*). Obviously, if $\circ$ is a commutative binary operation, the one-sided deletion rule becomes two-sided one‡.

Theorem 2.3

Let $\circ$ be an associative binary operation defined in A, having an identity element e. Then, the two-sided deletion rule is satisfied for any $a \in A$ having (as an inverse element) a', i.e:

$$\forall_{b,c \in A} \ ((a \circ b = a \circ c) \vee ((b \circ a = c \circ a)) \Rightarrow (b = c)).$$

---

* The notion of a group was first introduced by Évariste Galois (1811 – 1832).

† Niels Henrik Abel (1802 – 1829): *abelian group*, also called a *commutative group*, is a group in which the result of applying the group operation to two group elements does not depend on the order in which they are written. The concept of an *abelian group* underlies many fundamental algebraic structures, such as fields, rings, vector spaces, and algebras. In general, any group is isomorphic to a *group of permutations* (Arthur Cayley 1821 – 1895).

‡ As an example, this rule is satisfied in $(\mathbb{R} , +)$, but not in $(\mathbb{R} , \cdot)$: zero cannot be abbreviated.

Proof:

Let $a \circ b = a \circ c$. Then: $a' \circ (a \circ b) = a' \circ (a \circ c)$. And hence: $(a' \circ a) \circ b = (a' \circ a) \circ c$. Since $a' \circ a = e$ then: $e \circ b = e \circ c$. And so: $b = c$. The remaining proof (assuming: $b \circ a = c \circ a$) is left to the reader. □

Corollary 2.1

The two-sided deletion rule is satisfied in any group. □ {T 2.3}

2.4. Lattices

Definition 2.15 (*lattice*[*])

Lattice is an algebraic system $\mathcal{L} =_{df} (L ; o_1, o_2)$ having a type $T(\mathcal{L}) = (2,2)$ and satisfying the following three conditions:

(i) o_1 and o_2 are *commutative* binary operations,

(ii) o_1 and o_2 are *associative* and

(iii) o_1 and o_2 satisfy the *laws of absorption*, i.e:

$$\forall_{x,y \in L} \left(o_1(x, o_2(x,y)) = x = o_2(x, o_1(x,y)) \right) .$$

Usually, the above two binary operations are denoted by: $\sqcup =_{df} o_1$ and $\sqcap =_{df} o_2$, respectively. And hence: $\mathcal{L} =_{df} (L ; \sqcup , \sqcap)$. Moreover, the above three conditions: (i), (ii) and (iii) are represented as follows:

$$\forall_{x,y \in L} ((x \sqcup y = y \sqcup x) \wedge (x \sqcap y = y \sqcap x))$$

$$\forall_{x,y,z \in L} ((x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z) \wedge (x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z))$$

$$\forall_{x,y \in L} \left(x \sqcap (x \sqcup y) = x = x \sqcup (x \sqcap y) \right) .$$

Example 2.2 (*powerset diagram*)[†]

Let $A =_{df} \{a,b,c\}$. The power set diagram of A (ordered by '$\subset$') is illustrated in Figure 2.3 below. □

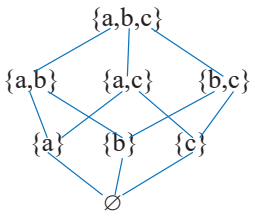


Figure 2.3

[*] Also known as '*structure*': Here, for any pair of elements in L there exist a smallest upper and a largest lower bounds.

[†] Another interesting examples were also presented (Kerntopf P. 1967): left to the reader.

Theorem 2.4

$\underset{x \in L}{\forall}$ (x is an idempotent element wrt $\sqcup$ and $\sqcap$) .

Proof:

Consider the above condition (iii), i.e: $x \sqcap (x \sqcup y) = x = x \sqcup (x \sqcap y)$. Since $x, y \in L$ may be arbitrary, let $y =_{df} x \sqcap y$. In accordance with (iii): $x = x \sqcap (x \sqcup y) = x \sqcap (x \sqcup (x \sqcap y)) = x \sqcap x$. In a similar way by assuming $y =_{df} x \sqcup y$ we can obtain: $x = x \sqcup (x \sqcap y) = x \sqcup (x \sqcap (x \sqcup y)) = x \sqcup x$. □

Theorem 2.5

Let $\mathcal{L} =_{df} (L ; \sqcup , \sqcap)$ be a lattice. For any $x,y \in L$: $x \sqcup y = y \Leftrightarrow x \sqcap y = x$.

Proof:

Assume that $x \sqcup y = y$. Then: $x \sqcap y = x \sqcap (x \sqcup y) = x$. Let now $x \sqcap y = x$. Then: $x \sqcup y = (x \sqcap y) \sqcup y = y$. □

Theorem 2.6

Let $\mathcal{L} =_{df} (L ; \sqcup , \sqcap)$ be a lattice and '$\preccurlyeq$' be a binary relation defined in L as follows:

$\underset{x,y \in L}{\forall}$ $(x \preccurlyeq y \Leftrightarrow_{df} ((x \sqcup y = y) \vee (x \sqcap y = x)))$.

Then '$\preccurlyeq$' is a *partial order* relation defined in L such that: $\sqcup \{x,y\} = x \sqcup y$ and $\sqcap \{x,y\} = x \sqcap y$*. And vice versa, if '$\preccurlyeq$' is a *partial order* relation defined in L such that: any two-element subset {x,y} of L have as a *smallest upper bound* $x \sqcup y$ and a *largest lower bound* $x \sqcap y$, then $\mathcal{L}$ is a lattice.

Proof (part I): †

Let $\mathcal{L} =_{df} (L ; \sqcup , \sqcap)$ be a lattice and '$\preccurlyeq$' be a binary relation defined in L as follows:

$$\underset{x,y \in L}{\forall} (x \preccurlyeq y \Leftrightarrow_{df} ((x \sqcup y = y)) .$$

In accordance with Theorem 2.4, for any $x \in L$: $x \sqcup x = x$. And hence, '$\preccurlyeq$' *reflexive*. i.e: $\underset{x \in L}{\forall} (x \preccurlyeq x)$.

Assume now: $x \preccurlyeq y$ and $y \preccurlyeq x$, where: $x, y \in L$. Then: $x \sqcup y = y$ and $y \sqcup x = x$. Since '$\sqcup$' is commutative then: $x = y$. Hence, '$\preccurlyeq$' is *weak antisymmetric*‡. Let now $x \preccurlyeq y$ and $y \preccurlyeq z$ $(x, y, z \in L)$. Hence: $x \sqcup y = y$ and $y \sqcup z = z$. Since '$\sqcup$' is associative: $z = y \sqcup z = (x \sqcup y) \sqcup z = x \sqcup (y \sqcup z) = x \sqcup z$. Then: $x \preccurlyeq z$. And so, '$\preccurlyeq$' is *transitive*.

In accordance with the last considerations, the above binary relation '$\preccurlyeq$' is at the same time *reflexive*, *weak antisymmetric* and *transitive*. And hence, this relation is a *partial order* on L. Next, it is necessary to show the existence of a *smallest upper bound* and a *largest lower bound* for any pair of elements belonging to L.

We have: $x \sqcup y = x \sqcup (x \sqcup y)$ and $x \sqcup y = y \sqcup (x \sqcup y)$. Hence: $x \preccurlyeq x \sqcup y$, $y \preccurlyeq x \sqcup y$ and $x \sqcup y$ is the *upper bound* of {x,y}. Let now $z \in L$ be another upper bound for {x,y}. Then: $x \preccurlyeq z$ and $y \preccurlyeq$

---

* The used here designations correspond to the notions of *supremum* and *infimum* (from latin: *supremus* and *infimus*, respectively).

† This proof consists of two parts: see (Kerntopf P. 1967).

‡ In general, a binary relation ρ defined in X is *weak antisymmetric* if:
$\underset{x,y \in X}{\forall} ((x \rho y) \wedge (y \rho x) \Rightarrow (x = y))$: See Definition 5.25 of Part I.

z . Hence: $(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z) = x \sqcup z = z$. Hence: $x \sqcup y \preccurlyeq z$ and $x \sqcup y$ is the s*mallest upper bound* for {x,y}.

In a similar way it can be shown that $x \sqcap y$ is the *largest lower bound* for {x,y}. In accordance with T 2.5: $x \sqcup y = y \Leftrightarrow x \sqcap y = x$.

Proof (part II):

Let now L be a given set, '$\preccurlyeq$' be a partial order defined in L. Assume that for any two element subset {x,y} of L there exist $x \sqcup y$ and $x \sqcap y$ (i.e. the smallest upper and the largest lower bounds, respectively) wrt '$\preccurlyeq$'. And hence: $x \sqcup y = y \sqcup x$ and $x \sqcap y = y \sqcap x$.

Assume that $s =_{df} y \sqcup z$ and $t =_{df} x \sqcup s = x \sqcup (y \sqcup z)$. And hence: $s \succcurlyeq y$, $s \succcurlyeq z$, $t \succcurlyeq x$ and $t \succcurlyeq s$. Then: $t \succcurlyeq y$ and $t \succcurlyeq z$. And so, t is an upper bound for {x, y, z}. If r is another upper bound for {x, y, z}, then $r \succcurlyeq x$, $r \succcurlyeq y$, $r \succcurlyeq z$. Hence, r is an upper bound for {y, z}. Since s is the smallest upper bound for {y, z} then $r \succcurlyeq s$. Then r is an upper bound for {x, s}. However t is the smallest upper bound for {x, s} , hence: $r \succcurlyeq t$. And so, we have: $t = x \sqcup (y \sqcup z)$ is the smallest upper bound for {x, y, z}. In a similar way, it can be shown that $(x \sqcup y) \sqcup z$ is the smallest upper bound for {x, y, z}. Since the smallest upper bound is unambiguously determined then: $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$. In a similar way it can be shown that: $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$.

Assume now $p =_{df} x \sqcup y$ and $q =_{df} x \sqcap p = x \sqcap (x \sqcup y)$. Then: $p \succcurlyeq x$, $p \succcurlyeq y$, $q \preccurlyeq x$, $q \preccurlyeq p$. Since $x \succcurlyeq x^*$ then x is a lower bound of {x,p}. However q is the largest lower bound of {x,p}. And hence, $x \preccurlyeq q$. By assumption $q \preccurlyeq x$ and hence: $x = q = x \sqcap (x \sqcup y)$. In a similar way, it can be shown that: $x \sqcup (x \sqcap y) = x$. This way, it was shown that in a partial ordered set L, such that for any two-element subset {x,y} of L there exist a smallest upper and a largest lower bounds, the above three conditions: (i), (ii) and (iii), given in Definition 2.15 are satisfied. And hence, $\mathcal{L}$ is a lattice. □

In accordance with the last theorem, lattices can be presented graphically by Hasse's diagrams† as it is shown in the above Figure 2.3 (or e.g. in Figure 5.5 of Part I of this book).

Definition 2.16 (*sublattice*)

An arbitrary algebraic subsystem‡ of $\mathcal{L}$.

In general, sublattices can be generated only by some subsets of L such that: along with each pair of elements x and y they also include: $x \sqcup_{\mathcal{L}} y$ and $x \sqcap_{\mathcal{L}} y$.

Example 2.3 (*sublattice*)

An example of a *sublattice* (wrt the lattice of Figure 2.3) is given in Figure 2.4 below (shown by read colour). □

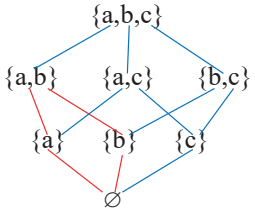


Figure 2.4

---

*The *partial order* relation '$\preccurlyeq$' defined in L is at the same time reflexive, weak antisymmetric and transitive.

† Helmut Hasse (1898 – 1979)

‡ See Definition 2.3.

Theorem 2.7[*]

Let $\mathcal{L} =_{df} (L ; \sqcup , \sqcap ; \succcurlyeq)$ be a lattice[†]. The following *semidistributivity rules* are satisfied:

(i) $\forall_{x,y,z \in L} (x \sqcap (y \sqcup z) \succcurlyeq (x \sqcap y) \sqcup (x \sqcap z))$ and

(ii) $\forall_{x,y,z \in L} (x \sqcup (y \sqcap z) \preccurlyeq (x \sqcup y) \sqcap (x \sqcup z))$.

Proof:

Since $y \sqcup z \succcurlyeq y$ and $y \sqcup z \succcurlyeq z$ then $x \sqcap (y \sqcup z) \succcurlyeq x \sqcap y$ and $x \sqcap (y \sqcup z) \succcurlyeq x \sqcap z$. And hence $x \sqcap (y \sqcup z)$ is an upper bound for $\{x \sqcap y, x \sqcap z\}$. Then: $x \sqcap (y \sqcup z) \succcurlyeq (x \sqcap y) \sqcup (x \sqcap z)$. The proof of (ii) is similar: left to the reader. □

We shall say that $\mathcal{L}$ is *semimodular* if the following condition is satisfied:

$$\forall_{x,y,z \in L} (x \succcurlyeq z \Rightarrow x \sqcap (y \sqcup z) \succcurlyeq (x \sqcap y) \sqcup z).$$

Corollary 2.2

The last condition is satisfied in any lattice $\mathcal{L} = (L ; \sqcup , \sqcap ; \succcurlyeq)$. □

Definition 2.17 (*modular lattice*)

$\mathcal{L} = (L ; \sqcup , \sqcap ; \succcurlyeq)$ is a *modular* (*Dedekind's*[‡]) lattice if:

$$\forall_{x,y,z \in L} (x \succcurlyeq z \Rightarrow x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup z).$$

Example 2.4 (*modular lattice*)

Consider lattice given in the above Figure 2.3. For example, let $x =_{df} \{b,c\}$, $y =_{df} \varnothing$ and $z =_{df} \{b\}$. Assume that $\sqcap$ and $\sqcup$ correspond to the set operations: $\cap$ and $\cup$, respectively. Since $\{b,c\} \succcurlyeq \{b\}$ then $\{b,c\} \cap (\varnothing \cup \{b\}) = \{b\} = (\{b,c\} \cap \varnothing \} \cup \{b\}$. □

Theorem 2.8 (*lattice distributivity*)

Let $\mathcal{L} = (L ; \sqcup , \sqcap)$ be a lattice. The following property is satisfied (for any $x,y,z \in L$):
$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ iff $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$.

Proof:

Assume that: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Then, in accordance with Definition 2.15 (i – iii) we can obtain (for any $x,y,z \in L$):

$$\begin{aligned} x \sqcup (y \sqcap z) &= (x \sqcup (x \sqcap z)) \sqcup (y \sqcap z) \\ &= x \sqcup ((x \sqcap z) \sqcup (y \sqcap z)) \\ &= x \sqcup ((z \sqcap x) \sqcup (z \sqcap y)) \\ &= x \sqcup (z \sqcap (x \sqcup y)) \\ &= (x \sqcap (x \sqcup y)) \sqcup (z \sqcap (x \sqcup y)) \end{aligned}$$

---

[*] The most of the considerations given below are under (Kerntopf P. 1967).

[†] Prowided there is no ambiguity and for simplicity, the partial order relation is some time also introduced in the lattice definition.

[‡] Julius Wilhelm Richard Dedekind (1831 – 1916)

$$= ((x \sqcup y) \sqcap x) \sqcup ((x \sqcup y) \sqcap z)$$
$$= (x \sqcup y) \sqcap (x \sqcup z)\,.\ \square$$

The proof of the opposite implication is similar: left to the reader. In accordance with T 2.8, the following definition is introduced.

Definition 2.18 (*distributive lattice*)

$\mathcal{L}$ is distributive if it is distributive wrt the one of its wo binary operations (i.e. $\sqcup$ or $\sqcap$. respectively).

Corollary 2.3

If $\mathcal{L}$ is distributive then $\mathcal{L}$ is modular.

Proof:

For any $x,y,z \in L$: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. Let $x \geqslant z$. Then $x \sqcap z = z$. And hence, in accordance with Definition 2.17, for any $x,y,z \in L$: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup z\,.\ \square$

Theorem 2.9

$\mathcal{L}$ is distributive $\Leftrightarrow \underset{x,y,z \in L}{\forall} ((x \sqcap y = x \sqcap z) \wedge (x \sqcup y = x \sqcup z) \Rightarrow (y = z))$.

Proof T 2.9a:

Let $\mathcal{L}$ be distributive, $x \sqcap y = x \sqcap z$ and $x \sqcup y = x \sqcup z$ (for any $x,y,z \in L$). We have:

$$\begin{aligned} y &= y \sqcup (y \sqcap x) \\ &= y \sqcup (z \sqcap x) \\ &= (y \sqcup z) \sqcap (y \sqcup x) \\ &= (y \sqcup z) \sqcap (x \sqcup z) \\ &= z \sqcap (x \sqcup y) \\ &= z \sqcap (z \sqcup x) \\ &= z\,.\ \square \end{aligned}$$

Proof T 2.9b:

Assume now $x,y,z \in L$ such that: $x \sqcap y = x \sqcap z$, $x \sqcup y = x \sqcup z$ and $y \neq z$: {aip}*. If $\mathcal{L}$ was distributive then: $(x \sqcap y) \sqcup z = (x \sqcup z) \sqcap (y \sqcup z)$ and $(x \sqcap y) \sqcup z = (x \sqcap z) \sqcup z = z$. On the other hand: $(x \sqcup z) \sqcap (y \sqcup z) = (x \sqcup y) \sqcap (y \sqcup z) = (x \sqcup y) \sqcap (z \sqcup y) = (x \sqcap z) \sqcup y = (x \sqcap y) \sqcup y = y$. And hence: $y = z$. $\square$ {contr†.}

Let $\mathcal{L} = (L\,;\sqcup,\sqcap\,;\geqslant)$ be a lattice. The *smallest upper* and *largest lower bounds* for L (if they exist) we shall denote by $\bigwedge_{\mathcal{L}}$ and $\bigvee_{\mathcal{L}}$ or by $\bigwedge$ and $\bigvee$ (if $\mathcal{L}$ is known) , respectively. The last lattice is then denoted by: $\mathcal{L} = (L\,;\bigwedge,\bigvee\,;\sqcup,\sqcap\,;\geqslant)$. If $\mathcal{L}$ is *finite*, i.e. $|L| \in \mathbb{N}$ then $\bigwedge$ and $\bigvee$ always exist. In general, if $\bigwedge$ and $\bigvee$ exist then for any $x \in L$: $\bigwedge \leqslant x \leqslant \bigvee$. And hence: $x \sqcup \bigwedge = x$, $x \sqcup \bigvee = \bigvee$, $x \sqcap \bigwedge = \bigwedge$ and $x \sqcap \bigvee = x$. Moreover, if $x \sqcup y = \bigwedge$ then $x = y = \bigwedge$. In a similar way, if $x \sqcap y = \bigvee$ then $x = y = \bigvee$.

Definition 2.19 (*complete lattice*)

$\mathcal{L} = (L\,;\sqcup,\sqcap\,;\geqslant)$ is *complete* $\Leftrightarrow \underset{\varnothing \neq l \subset L}{\forall} \underset{a,b \in L}{\exists} ((\sqcup l = a) \wedge (\sqcap l = b))$.

Corollary 2.4

---

* assumption(s) of indirect proof

† contradiction

If $\mathcal{L}$ is finite then $\mathcal{L}$ is complete. If $\mathcal{L}$ is complete then $\wedge$ and $\vee$ always exist. □

It can be shown (proof by induction) that the following two properties are satisfied in any distributive lattice $\mathcal{L}$:

$$x \sqcup \left(\prod_{i=1}^{n} y_i\right) = \prod_{i=1}^{n} (x \sqcup y_i)$$

$$x \sqcap \left(\bigsqcup_{i=1}^{n} y_i\right) = \bigsqcup_{i=1}^{n} (x \sqcap y_i).$$

The last two rules are said to be *nonfinite* ones if instead of $n \in \mathbb{N}$ it is used the symbol '∞'.

Definition 2.20 (*element's complement*)

Some element $c \in L$ is said to be a *complement* of $a \in L$ in $\mathcal{L}$ iff $a \sqcup c = \vee$ and $a \sqcap c = \wedge$.

In general, a given element of L may have not a complement, may have exactly one complement or also may have many complements. Obviously, if c is a complement of a then a is a complement of c. Moreover, $\vee$ is the exactly one complement wrt $\wedge$ and vice versa.

Theorem 2.10

If $\mathcal{L} = (L; \wedge, \vee; \sqcup, \sqcap)$ is distributive then for any $x \in L$ there exists at most one complement x'.

Proof:

Let x and y be two different complements for $a \in L$: $a \sqcup x = \vee$, $a \sqcap x = \wedge$ and $a \sqcup y = \vee$, $a \sqcap y = \wedge$ {aip}. Then we can obtain: $y = y \sqcap \vee = y \sqcap (a \sqcup x) = (y \sqcap a) \sqcup (y \sqcap x) = \wedge \sqcup (y \sqcap x) = (x \sqcap a) \sqcup (x \sqcap y) = x \sqcap (a \sqcup y) = x \sqcap \vee = x$. □ {contr.}

In general, in a given distributive lattice some elements of L may not have complements. According to the last theorem, any $a \in L$ may have no more than one *complement* a'.* Moreover some elements may not have complements.

Let a' be a complement of $a \in L$. Then a jest the complement of a". Since $\mathcal{L}$ is distributive then a is the only one such complement. And hence, the following theorem is satisfied.

Theorem 2.11

If $\mathcal{L} = (L; \wedge, \vee; \sqcup, \sqcap)$ is distributive and a' is complement of a then there exists a" and a" = a. □

Theorem 2.12

Let $\mathcal{L}$ be distributive and $a, b \in L$. If there exist the complements a' and b' then there exist also the complements $(a \sqcup b)'$ and $(a \sqcap b)'$ such that: $(a \sqcup b)' = a' \sqcap b'$ and $(a \sqcap b)' = a' \sqcup b'$.

---

* Provided there is no ambiguity and for simplicity, e.g. the *complement* of a instead of $\bar{a}$ is denoted by a', i.e. $a' =_{df} \bar{a}$. In particular: $\wedge' = \vee$ (and vice versa).

Proof:

We have:

$(a \sqcup b) \sqcap (a' \sqcap b') = (a \sqcap a' \sqcap b') \sqcup (b \sqcap a' \sqcap b') = (\wedge \sqcap b') \sqcup (\wedge \sqcap a') = \wedge \sqcup \wedge = \wedge$,

$(a \sqcup b) \sqcup (a' \sqcap b') = (a \sqcup b \sqcup a') \sqcap (a \sqcup b \sqcup b') = (\vee \sqcup b) \sqcap (\vee \sqcup a) = \vee \sqcap \vee = \vee$.

The proof for: $(a \sqcap b)' = a' \sqcup b'$ is similar (left to the reader). □

According to T 2.12, the above two equalities, i.e. $(a \sqcup b)' = a' \sqcap b'$ and $(a \sqcap b)' = a' \sqcup b'$ are known as *De Morgan's laws*[*].

Corollary 2.5

The subset of all elements of $\mathcal{L} = (L ; \wedge , \vee ; \sqcup , \sqcap)$ having complements together with $\sqcup$ and $\sqcap$ truncated to this subset is a sublattice of $\mathcal{L}$. □

Theorem 2.13

Let $\mathcal{L}$ be distributive and $x,y \in L$ have complements x' and y'. Then:

(i) $x \preccurlyeq y \Leftrightarrow x' \succcurlyeq y'$ and

(ii) $x \preccurlyeq y \Leftrightarrow x \sqcap y' = \wedge \Leftrightarrow x' \sqcup y = \vee$.

Proof:

(i): $x \preccurlyeq y \Leftrightarrow x \sqcup y = y$
$\Leftrightarrow x' \sqcap y' = y'$
$\Leftrightarrow x' \succcurlyeq y'$.

(ii): $x \preccurlyeq y \Rightarrow x \sqcap y' = (x \sqcap y) \sqcap y' = x \sqcap (y \sqcap y') = x \sqcap \wedge = \wedge$.

$x \sqcap y' = \wedge \Rightarrow x = x \sqcap \vee = x \sqcap (y \sqcup y') = (x \sqcap y) \sqcup (x \sqcap y') = (x \sqcap y) \sqcup \wedge = x \sqcap y$.

And hence: $x \preccurlyeq y$. □

Definition 2.21 (*complemented lattice*)

We shall say that $\mathcal{L} = (L ; \wedge , \vee ; \sqcup , \sqcap)$ is complemented if every element of L has a complement.

Definition 2.22 (*Boolean lattice / Boolean algebra*)

A distributive lattice which at the same time is complemented.

In accordance with the last definition: since $\mathcal{L}$ is *distributive* then every element may have at most one complement. Since $\mathcal{L}$ is also *complemented* then every element may have at least one complement. As a consequence, in *Boolean algebra* every element has exactly one complement. And hence, the *complement* in Boolean algebra is accepted as one argument operation. Moreover, by definition, the elements $\wedge$ and $\vee$ are also included in any such algebra. The last algebra is denoted as follows: $\mathcal{B} =_{df} (B ; \wedge , \vee ; \sqcup , \sqcap ; ')$. Another. equivalent definition is the next one.

Definition 2.23 (*Boolean algebra*[†])

An algebraic system $(B ; \wedge , \vee ; \sqcup , \sqcap ; ')$ of type (0, 0, 2, 2, 1) such that:

---

[*] Augustus De Morgan (1806 – 1871)

[†] George Boole (1815 – 1864)

(i) $(B ; \sqcup , \sqcap)$ is a *distributive lattice*,

(ii) $\wedge$ ($\vee$) is the *identity element* for $\sqcup$ ($\sqcap$) and *zero element* for $\sqcap$ ($\sqcup$),

(iii) $\forall_{b \in B} ((b \sqcup b' = \vee) \wedge (b \sqcap b' = \wedge))$.

The following properties are satisfied in any Boolean algebra :

$\wedge' = \vee$	$\vee' = \wedge$	(1)
$a \sqcup \vee = \vee$	$a \sqcap \wedge = \wedge$	(2)
$a \sqcup \wedge = a$	$a \sqcap \vee = a$	(3)
$a \sqcup a' = \vee$	$a \sqcap a' = \wedge$	(4)
$a'' = a$		(5)
$a \sqcup a = a$	$a \sqcap a = a$	(6)
$(a \sqcup b)' = a' \sqcap b'$	$(a \sqcap b)' = a' \sqcup b'$	(7)
$a \sqcup b = b \sqcup a$	$a \sqcap b = b \sqcap a$	(8)
$a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$	$a \sqcap (b \sqcap c) = (a \sqcap b) \sqcap c$	(9)
$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$	$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$	(10)

The above equalities are usually accepted as axioms of a Boolean algebra (described as in Definition 2.23). A complete axiomatic system is also the following one: {(3), (4), (8), (10)}. However, the corresponding proofs should be more difficult. As an example, $(2^A; \varnothing , A ; \cup , \cap , ')$ is a *Boolean algebra*[*] (Kerntopf P. 1967).

2.5. Multiple valued and fuzzy algebraic systems

Multiple valued and fuzzy algebraic systems were introduced as generalisations of the concept of Boolean algebra. Some informations concerning these two systems are given below. Multiple valued systems are first briefly presented.

Multiple valued systems

The notion of a *multiple valued algebra* was first introduced by Chang C.C.[†] (1958): 'So because of my own shortcomings in following proofs in Polish notation during a seminar held by Rosser on the completeness of Łukasiewicz[‡] axioms for infinite valued propositional logic, MV – algebras came to be … It occurred to me that the approach used in the proof of completeness of the two valued propositional logic via the *Boolean* (*Lindenbaum* / cnown also as: *Lindenbaum – Tarski*[§]) *algebra* of equivalence classes of formulas and the Boolean maximal ideal theorem might be another way to do what Rosser did, but avoiding syntactical manipulations of formulae in Polish notation. This was the beginning of MV - algebras. Chang found a proof of the completeness of the $\aleph_0$ - *valued logic* (developed by Łukasiewicz and Tarski). See also: (Rose A. 1956). Since then MV - algebras have found increasing interest. There were also introduced finitely additive measures (called *states*) on MV - algebras with the intent of capturing the notion of 'average degree of truth' of a proposition Mundici D. (1995). In particular, the following definition was introduced in (Barbieri G. and Weber H. 2002).

Definition 2.24 (*MV- algebra*)

---

[*] See Part I of this work.

[†] Chen Chung Chang (1927 – 2014): https://www.ams.org/journals/tran/1958-088-02/S0002-9947-1958-0094302-9/S0002-9947-1958-0094302-9.pdf

[‡] Jan Łukasiewicz (1878 – 1956)

[§] Adolf Lindenbaum (1904 – 1941), Alfred Tarski (1901 – 1983)

An *MV- algebra* $(L , + ,' , 0 , 1)$ is a commutative semigroup with $0,1$ and a unary operation $' : L \to L$ which satisfies the following axioms:

(L1) $x + 1 = 1$

(L2) $(x')' = x$

(L3) $0' = 1$

(L4) $(x' + y') + y = (x + y')' +$ for every $x,y \in L$.

The following definitions were also introduced: $x \leq y$ iff $x' + y = 1$ and $y \Delta x =_{df} (x + y')$. And hence, whenever $x \leq y$ the following axioms were used:

(P1) $x \leq x + y$

(P2) $x \leq y \leq z + x \Rightarrow y \Delta x \leq z$

(P3) $y = y \Delta x + x$ whenever $x \leq y$

(P4) $x' = x$.

In general, any MV - algebra 'is an algebraic structure with a binary operation $\oplus$, a unary operation $\neg$, and the constant 0 , satisfying certain axioms. MV - algebras are the algebraic semantics of Łukasiewicz's logic; the letters MV refer to the many-valued logic of Łukasiewicz. MV - algebras[*] coincide to the class of bounded commutative BCK algebras' [†]. In accordance with the last considerations, the following definition was also introduced.

Definition 2.25 (*MV- algebra*)[‡]

An *MV - algebra* is an algebraic structure $(L, \oplus , \neg , 0)$, where: $L \neq \varnothing$ is a set, $\oplus$ is a binary operation on L, $\neg$ is a unary operation on L and 0 is a constant denoting a fixed element on L , such that the following identities are satisfied (for any $x.y,z \in L$):

(1) $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

(2) $x \oplus 0 = x$

(3) $x \oplus y = y \oplus x$

(4) $\neg \neg x = x$

(5) $x \oplus \neg 0 = \neg 0$

(6) $\neg (\neg x \oplus y) \oplus y = \neg (\neg y \oplus x) \oplus x$.

An *MV – algebra* can be equivalently defined as a 'prelinear commutative bounded integral residuated lattice $(L, \wedge , \vee, \otimes, \Rightarrow , 0, 1)$ satisfying the following additional equivalence: $x \vee y \Leftrightarrow (x \Rightarrow y) \Rightarrow y$[§] (Hájek P[**]. 1998): the last equivalence corresponds to the following thesis: $p \vee q \Leftrightarrow (p \Rightarrow q) \Rightarrow q$ (the corresponding two

---

[*] Known also as Łukasiewicz - Moisil algebras: However, in 1956 Alan Rose discovered that for $n \geq 5$, the Łukasiewicz–Moisil algebra does not model the Łukasiewicz logic

[†] https://en.wikipedia.org/wiki/MV-algebra. BCI and BCK algebras are algebraic structures, introduced by Imai Y., Iséki K. and Tanaka S. in 1966, e.g. see: Iséki K. and Tanaka S., *An introduction to the theory of BCK-algebras*. Mathematica Japonica, Japanese Association of Mathematical Sciences, vol. 23 (1978) 1 – 26.

[‡] https://en.wikipedia.org/wiki/MV-algebra: provided there is no ambiguity instead of A, for convenience, the set of elements is here denoted by L.

[§] The implication binds more strongly that equivalence.

[**] Petr Hájek (1940 – 2016)

proofs are indirect: left to the reader). According to the first three axioms $(L, \oplus, 0)$ is a *commutative monoid* (see Definition 2.8).

Example 2.5 (*two-element MV – algebra*)

The two - element Boolean algebra $\{0,1\}$, with $\oplus$ coinciding with Boolean disjunction and $\neg$ with Boolean negation. By adding (as an axiom): $x \oplus x = x$ we can obtain the axiomatisation of Boolean algebra*. □

Another algebraic system was introduced in the 1940s by Moisil†, now known as *Łukasiewicz-Moisil algebra* ('in the hope of giving algebraic semantics for the n-valued *Łukasiewicz* logic'). However, it was discovered that for $n \geq 5$, the last algebra does not model the Łukasiewicz logic. 'For the axiomatically more complicated (finite) *n*-valued Łukasiewicz logics, suitable algebras were published in 1977 by Revaz Grigolia and called MV_n-algebras. MV_n-algebras are a subclass of LM_n-algebras, and the inclusion is strict for $n \geq 5$. In 1982 Roberto Cignoli published some additional constraints that added to LM_n-algebras to produce proper models for *n*-valued Łukasiewicz logic. Cignoli called his discovery proper Łukasiewicz algebras' (Cignoli R. 1982). There were next introduced 'two chains of unary operations, as a key in establishing many connections between these algebras and n-valued Lukasiewicz-Moisil algebras' (Iorgulescu A. 1999).

Fuzzy algebraic systems

Fuzzy algebra is an important chapter of fuzzy set theory. As an example, some properties of fuzzy algebraic systems, e.g. such as: Lie algebras and superalgebras, interval-valued fuzzy Lie ideals, etc., were studied in (Akram M. 2018). In particular, a *Lie algebra*‡ is 'a vector space g together with an operation called the Lie bracket, an alternating bilinear map $g \times g \to g$, $(x,y) \mapsto [x,y]$ that satisfies the *Jacobi*§ *identity* (a property of a binary operation that describes how the order of evaluation, the placement of parentheses in a multiple product, affects the result of operation: '*map*' is an abbreviation of *mapping*)'.

The *Smarandache function* $\mu(n)$ is 'the function first considered by Lucas E. (1883), Neuberg J. (1887) and Kemper A.J. (1918), and subsequently rediscovered by Smarandache** (1980) that gives the smallest value for a given $n \in \mathbb{N}$ at which $n / \mu(n!)$ (i.e. n divides $\mu(n!)$, e.g. 8 is the smallest natural number that divides 4! $(= 1 \cdot 2 \cdot 3 \cdot 4)$††. *Smarandache fuzzy algebra* is briefly presented below.

'*Smarandache algebra*, like its predecessor, *fuzzy algebra*, arose from the need to define structures which were more compatible with the real world where the grey areas mattered. Lofti A Zadeh‡‡, the father of fuzzy sets, remarked that: "So, this whole thing started because of my perception at that time, that the world of classical mathematics - was a little too much of a black and white world, that the principle of the 'excluded middle' meant that every proposition must be either true or false. There was no allowance for the fact that classes do not have sharply defined boundaries." So, here is this book, which is an amalgamation of alternatives' (Vasantha Kandasamy W.B. 2003). There was studied in the last book 'the subject of Smarandache Fuzzy Algebra. Originally, the revolutionary theory of Smarandache notions was born as a paradoxist movement that challenged the status quo of existing mathematics. The genesis of Smarandache Notions, a field founded by Florentin Smarandache, is alike to that of Fuzzy Theory: both the fields imperatively questioned the dogmas of classical mathematics. Despite the fact that Fuzzy Algebra has been studied for over fifty years, there are only two books on fuzzy algebra. But both the books do not cover topics related to fuzzy semirings, fuzzy near-rings etc. so we have in this book, two parts: In Part 1 we have recalled all the definitions and properties of fuzzy algebra. In Part II we give Smarandache fuzzy

---

* https://en.wikipedia.org/wiki/%C5%81ukasiewicz%E2%80%93Moisil_algebra

† Grigore Constantin Moisil (1906 – 1973)

‡ Marius Sophus Lie (1842 – 1899)

§ Carl Gustav Jakob Jacobi (1804 – 1851)

** Florentin Smarandache, born 1954

†† Wolfram Math World (https://mathworld.wolfram.com/SmarandacheFunction.html)

‡‡ Lotfi Aliasker Zadeh (1921 – 2017)

algebraic notions. This is the first book in fuzzy algebra which covers the notions of fuzzy semirings and fuzzy near-rings though there are several papers on these two concepts'.

Some applications of Smarandache fuzzy algebraic structures were given in the last Chapter 7 of this work (Sections 7.1 and 7.2). And finally, some research problems (i.e. 25 problems) related to the applications of Smarandache fuzzy algebraic structures were also presented (Section 7.3). In particular: 'the applications of Smarandache algebraic structures viz. Smarandache groupoids, Smarandache near-rings and Smarandache semirings in automaton theory, in error correcting codes and in the construction of S-subbiautomaton which are given in about forty definitions'. The notions of '*semi-automaton* and *automaton*' were introduced by using such fundamental algebraic structures as *semigroups* (i.e. groupoids having associative binary operations: see Definition 2.7). The chapter starts with the notion of *Smarandache free groupoid* (in short: *S-free groupoid*), see: (Vasantha Kandasamy W.B. 2003 / Chapter seven: Applications of Smarandache fuzzy algebraic structures).

Definition 2.26 (*S-free groupoid*)

Let S be non empty set. Generate a free groupoid using S and denote it by ⟨S⟩. Clearly the free semigroup generated by the set S is properly contained in ⟨S⟩; as in ⟨S⟩ we may or may not have the associative law to be true.

In accordance with the last definition, the associativity property is not assumed (as e,g. in a *free semigroup*[*]). The following property was also presented.

Theorem 2.14

Every free groupoid is a S-free groupoid. □

The (*classical*) definitions of semi automaton and automaton are given as follows (Vasantha Kandasamy W.B. 2002, Mordeson J. N. and Malik D.S. 2002).

Definition 2.27 (*semi automaton*)

A *semi-automaton* is a triple $Y =_{df} (Z, A, \delta)$ consisting of two non - empty sets Z and A and a function $\delta : Z \times A \to Z$, where Z is called the *set of states*, A the *input alphabet* and δ the *next state function* of Y.

Definition 2.28 (*automaton*)

An *automaton* is a quintuple $K = (Z, A, B, \delta, \Lambda)$ where (Z, A, δ) is a *semi automaton*, B is a non-empty set called the *output alphabet* and $\lambda: Z \times A \to B$ is the *output function*.

Any automaton (or also semi automaton) can be described e.g. using *next state table* or graph. In the case of automaton it is needed also an *output table*. Some applications of fuzzy algebraic structures and Smarandache fuzzy algebraic structures were also given (Vasantha Kandasamy W.B. 2003 / see: Part Two, Chapter 7, Section 7.2, p.426): 'The application of fuzzy algebra is mainly found in finite machines or which we choose to call as Smarandache fuzzy automaton. Apart from this we see that there is no direct application of Fuzzy algebra. When we say Fuzzy algebraic structures we mean only fuzzy groupoids, fuzzy semigroups, fuzzy groups, fuzzy rings, fuzzy nearrings, fuzzy seminear-rings, fuzzy semirings and fuzzy vector spaces. We don't mix up the concepts of fuzzy logic with fuzzy algebraic structures'. Some notions related to the algebraic fuzzy automaton theory were also given ('directly helpful to us in constructing *Smarandache algebraic fuzzy automatons*').

The notion of *Smarandache free groupoid* and its application to automaton or linked automatons was also studied in (Vasantha W.B. and Chetry M,K. 2004): 'The study of free groupoids is very new and we see that the free groupoids have its application to the theory of automaton provided they satisfy a special condition namely

---

[*] *Free monoid* on a set A (usually denoted by A^*) is 'the monoid whose elements are all the finite sequences (or strings) of zero or more elements from that set, with *string concatenation* as the monoid operation and with the unique sequence of zero elements, often called the empty string (denoted by ε or λ as the *identity element*). The free semigroup on A (usually denoted by A^+) is the subsemigroup of A^* containing all elements except the empty string': https://en.wikipedia.org/wiki/Wikipedia

these free groupoids should contain at least a subset which is a free semigroup. Thus motivated by this property we define in this paper the notion of *Smarandache free groupoid*, these Smarandache free groupoids by their very definition contains at least one free semigroup. It may happen that these Smarandache free groupoids may have more than one free semigroup in which case we see still it has more applications, for we can define several automatons which can be easily linked for a free given groupoid. Thus we in this paper define Smarandache free groupoids and give its application to automaton or linked automatons, which we choose to call as New Smarandache Automaton or New Smarandache Linked Automaton'.

There exist many algebraic systems corresponding to multi-valued logic and using various logical, arithmetic, set-theoretical or other operations (Lu and Lee 1985). In particular, (the multi-valued) M-algebra (Lu 1983) was introduced as the system $\mathbf{M} =_{df} (M ; 0, m-1; +, \cdot, -)$, where for any multi-valued variables of $M =_{df} \{0, 1, \ldots, m-1\} \subsetneq \mathbb{N} \cup \{0\}$ the basic (two logical and one arithmetic) operations MAX(x,y), MIN(x.y), and SUB(x,y) (denoted by $+, \cdot$, and $-$,respectively) are specified as follows: $x + y$ = MAX(x,y) = the *larger value* of {x,y}, $x \cdot y = xy$ = MIN(x,y) = the *smaller value* of (x,y), and $x - y$ = SUB(x,y) = *subtract* y from x (only if $x \geq y$). The NOT operation (denoted by ' ′ ') is defined as $x' = SUB((m-1),x) = (m-1) - x$. The operation $x - y$ is undefined. If $x < y$. So, an improved version of the above system, also called M-algebra (and also denoted by **M**) was proposed (Lu and Lee 1985). Instead of (the partial operation) SUB a new arithmetic operation. called *truncated subtraction* and denoted by TSUB, was introduced, i.e. $x \ominus y$ = TSUB(x,y) = if $x \geq y$ then SUB(x,y) else 0. It has been shown that {MAX,TSUB} and {MIN,TSUB} are two minimal functionally complete sets of basic operations. The last two sets can be implemented using only digital circuits.

A multi-valued algebraic system by using the above notion of M-algebra was introduced in (Tabakow I.G. 1993). The system, called *D-algebra*[*], in short **D** ,was represented as an isomorphic image of the direct product of two M-algebras, i.e.the following theorem was sown.

Theorem 2.15

D = φ(**M** × **M**) , i.e. **D** is an isomorphic image of the direct product of two M-algebras wrt φ. □

Next this D-algebra was used to generate tests for m-logic combinational circuits (i.e. circuits which realize m-valued logic functions, $m \geq 2$). The test generation was a fault oriented process (tests are derived for specific faults). This process was illustrated by means of an informal modification of the classical *Roth's D-algorithm*[†] (a more formal treatment is omitted). For simplicity, only the s-a-fault model was considered and several examples were given. An example ternary lattice (related to the corresponding D-algebra) is shown in Figure 2.5 below.

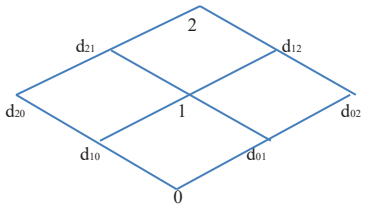


Figure 2.5

[*] The composite multi-valued algebraic system, called *D-algebra*, is the system $\mathbf{D} =_{df} (D ; 0, m-1 ; +_1, \cdot_1, -_1)$, where D is the set of composite value symbols (the elements of the algebra), 0 and $m-1$ (i.e. d_{00} and $d_{(m-1)(m-1)}$) are the minimal and maximal elements of the corresponding lattice and for any $d_{ij}, d_{kl} \in D$ the last three operations are defined as follows: $d_{ij} +_1 d_{kl} = DMAX(d_{ij}, d_{kl}) = d_{(i+k)(j+l)}$, $d_{ij} \cdot_1 d_{kl} = DMIN(d_{ij}, d_{kl}) = d_{(i \cdot k)(j \cdot l)}$ and $d_{ij} -_1 d_{kl} = DTSUB(d_{ij}, d_{kl}) = d_{(i-k)(j-l)}$.

[†] John Paul Roth, born: 1922

II. Homomorphisms, direct products and free algebraic system

Below are first considered various kinds of homomorphisms concerning algebraic systems. Such notions as direct products of two algebraic systems and free algebraic systems are also considered. The most of the used below basic notions and definitions are mainly under (Kerntopf P. 1967)

3. Homomorphisms*

The notion of homomorphism, i.e. considered as a 'structure-preserving map between two algebraic systems of the same type' was early used by Felix Klein†. Some kinds of homomorphisms are considered below‡.

3.1. Epi-, mono- , iso- , endo- and automorphisms

Definition 3.1 (*homomorphism)*

Let $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$ and $\beta =_{df} (B ; b_1, b_2, \dots, b_n ; o_1', o_2', \dots, o_m')$ be two algebraic systems of the same type§. We shall say that $\varphi : A \to B$ is a *homomorphism* of α in β if the following two condisions are satisfied (known as: *preservation of constants and operations*, respectively):

$$(1) \qquad \forall_{i \in \{1,\dots,n\}} (\varphi(a_i) = b_i).$$

$$(2) \qquad \forall_{j \in \{1,\dots,m\}} \; \forall_{x_1, \dots, x_{nj} \in A} (\varphi(o_j(x_1, \dots, x_{nj})) = o_j'(\varphi(x_1), \dots, \varphi(x_{nj})))$$

The system $(\varphi(A) ; b_1, b_2, \dots, b_n ; o_1'/\varphi(A), o_2'/\varphi(A), \dots, o_m'/\varphi(A))$ is said to be a *homomorphic image* of α wrt φ. Moreover, we shall say that α *homomorphically maps*** *to* β if there is a mapping such that it is a *homomorphism* of α in β.

Let $\varphi : A \to B$ be a homomorphism of α in β. In particular, if φ is *surjective mapping* (i.e. *onto*) then the above homomorphism is said to be *epimorphism*. In a similar way, if φ is *injective mapping* (i.e. *one – to – one*) we have *monomorphism*.

---

* This term (gr. ὅμοιος, *homoios* – podobny; μορφή, *morphē* – shape, form) is different from the notion of '*homeomorphism*' used in graph theory (a mapping between two graphs that respects their structure).

† Christian Felix Klein (1849 – 1925)

‡ https://en.wikipedia.org/wiki/Homomorphism

§ And hence, α and β are two *similar algebraic systems*, i.e. of the same type: see Definition 2.2 of Subsection 2.1.

** The used term 'map' is an abbreviation of '*mapping*'.

The above homomorphism is an *isomorphism* if it is at the same time surjective and injective mapping*. The homomorphism of α in α is said to be *endomorphism* and the isomorphism of α in α is defined as *automorphism*.

The following two properties are satisfied (in accordance with Definition 3.1).

(i) Let α, β and γ be three algebraic systems of the same type. Assume that φ is a homomorphism of α in β and ψ is a homomorphism of β in γ, then the *superposition* of homomorphisms $\varphi\psi$ is a homomorphism of α in γ.

(ii) Let φ be a homomorphism of α in β. Then, the *homomorphic image* of α wrt φ is a subsystem of β.

By $\alpha \approx_\varphi \beta$ it is denoted below the fact that: 'α *maps isomorphically to* β *wrt* φ'.The *identity map* is here denoted by 'e'. Obviously, the last binary relation '$\approx$' is an equivalence one, i.e. reflexive, symmetric and transitive . And hence we have: $\alpha \approx_\varphi \alpha$, $\alpha \approx_\varphi \beta \Rightarrow \beta \approx_{\varphi^{-1}} \alpha$ and $(\alpha \approx_\varphi \beta) \wedge (\beta \approx_\psi \gamma) \Rightarrow (\alpha \approx_{\varphi\psi} \gamma)$, i.e:

$$\alpha \underset{\text{hom}}{\overset{e}{\rightarrow}} \alpha,$$

$$\alpha \underset{\text{hom}}{\overset{\varphi}{\rightarrow}} \beta \Rightarrow \beta \underset{\text{hom}}{\overset{\varphi^{-1}}{\rightarrow}} \alpha \quad \text{and}$$

$$\alpha \underset{\text{hom}}{\overset{\varphi}{\rightarrow}} \beta \wedge \beta \underset{\text{hom}}{\overset{\psi}{\rightarrow}} \gamma \Rightarrow \alpha \underset{\text{hom}}{\overset{\varphi\psi}{\rightarrow}} \gamma.$$

Here $\varphi\psi$ (or more formally: $\varphi\cdot\psi$) denotes the *composition* (or equivalently: *superpositiom*) of the last two mappings. The next examples are under (Kerntopf P. 1967).

Example 3.1 (*algebraic system homomorphism*)

Consider the following two algebraic systems: $\alpha =_{df} (\mathbb{Z}; 0; +)$ and $\beta =_{df} (\{-1, 1\}; 1; \cdot)$, where '+' and '$\cdot$' denote the usual *arithmetical addition* and *multiplication*, respectively. Since $T(\alpha) = T(\beta) = (1, 2)$ the last two algebraic systems are similar. And hence, the map $\varphi: \mathbb{Z} \rightarrow \{-1, 1\}$ such that:

$$\forall_{z \in \mathbb{Z}} \; (\textit{if } z \textit{ is even then } 1 \textit{ else } -1)$$

is an *epimorphism* between the last two algebraic systems.†□

Example 3.2 (*algebraic system homomorphism*)

Let $\alpha =_{df} (\mathbb{R}; 0; +)$ and $\beta =_{df} (\mathbb{C} - \{0\}; 1; \cdot)$ be two algebraic systems of type)1,2). Consider the map $\varphi: \mathbb{R} \rightarrow \mathbb{C} - \{0\}$ such that: $\varphi(a) = e^{ia}$‡. We have: $\varphi(0) = e^{i0} = e^0 = 1$ and $\varphi(a + b) = e^{i(a+b)} = e^{ia} \cdot e^{ib} = \varphi(a) \cdot \varphi(b)$. □

Example 3.3 (*algebraic system homomorphism*)

---

* A *bijection* (*bijective function*: in general *bijective map* or *one-to-one correspondence*) is a map $f: X \rightarrow Y$ which is at the same time *one-to-one* (*injective*, i.e. $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$, for any $x_1, x_2 \in X$) and *onto* (i.e. *surjective*): https://en.wikipedia.org/wiki/Bijection

† Let $\varphi: \mathbb{Z} \rightarrow \{-1, 1\}$ such that: $\varphi(z) =_{df}$ if z/2 then 1 else -1. Since '0' is even, i.e. $\varphi(0) = 1$, then it is sufficient to show the following equality: $\varphi(a + b) = \varphi(a) \cdot \varphi(b)$ (for any $2^2 = 4$ even / odd combinations: left to the reader).

‡ $e^{ix} = \cos(x) + i \cdot \sin(x)$ (Leonhard Euler 1707 - 1783)

Consider the following two *Boolean algebras*: $\alpha =_{df} (B ; \wedge , \vee ; \sqcup , \sqcap ; ')$ and $\beta =_{df} (B ; \vee , \wedge ; \sqcap , \sqcup ; ')^{*}$. Let $\varphi : B \rightarrow B$ such that $\varphi(b) =_{df} b'$ (for any $b \in B$). We have:

(i) $\varphi(\wedge) = \vee, \varphi(\vee) = \wedge$

(ii) $\varphi(b_1 \sqcup b_2) = (b_1 \sqcup b_2)' = b_1' \sqcap b_2' = \varphi(b_1) \sqcap \varphi(b_2)$,

(iii) $\varphi(b_1 \sqcap b_2) = (b_1 \sqcap b_2)' = b_1' \sqcup b_2' = \varphi(b_1) \sqcup \varphi(b_2)$,

(iv) $\varphi(b') = b'' = \varphi(b)'$.

And so, we have an *automorphism* (i.e. a bijective homomorphism of an object with itself). □

Obviously, an isomorphism between two algebraic systems can be considered as an equivalence relation. And hence, all isomorphic algebraic systems have the same algebraic properties. In particular, the notion of a homomorphism of two *partial ordered sets* is defined in a similar way (by $\geqslant_A$ and $\geqslant_B$ are denoted below the partial order relations corresponding to the sets A and B, respectively).

Definition 3.2 (*homomorphism: partially ordered sets*)

Let A and B be two partially ordered sets. The map $\varphi : A \rightarrow B$ is said to be a *homomorphism*[†] of A in B iff:

$$\forall_{a,b \in A} \; (a \geqslant_A b \rightarrow \varphi(a) \geqslant_B \varphi(b))$$

Theorem 3.1

Let φ be a homomorphism of α in β such that it maps the set of generators of α onto the set of generators of β. Then φ is an epimorphism. □

Theorem 3.2

Let A_0 be the set of generators of $\alpha =_{df} (A ; a_1, a_2, \ldots, a_n ; o_1, o_2, \ldots, o_m)$ and let φ_1 and φ_2 be two homomorphisms of α in $\beta =_{df} (B ; b_1, b_2, \ldots, b_n ; o_1', o_2', \ldots, o_m')$. Then:

$$\forall_{a \in A_0} (\varphi_1(a) = \varphi_2(a)) \Rightarrow \forall_{a \in A} (\varphi_1(a) = \varphi_2(a)),$$

i.e. if $g : A_0 \rightarrow B$ can be extended to a homomorphism of α in β then any such extension is uniquely defined.

Proof:

In accordance with the notion of a homomorphism (see Definition 3.1), it follows that $\{a \in A / \varphi_1(a) = \varphi_2(a)\}$ is a subsystem of α, containing the same set of generators as in α. □

3.2. Congruences and quotient algebraic systems

The notion of congruence is defined as follows (Kerntopf P. 1967):

Definition 3.3 (*congruence*)

---

[*] Of the same type: (0 , 0 , 2 , 2, 1): the complement of a, instead of $\bar{a}$, is here denoted by a' , i.e. $a' =_{df} \bar{a}$.

[†] The above homomorphism is an *isomorphism* if it is at the same time surjective and injective mapping.

Let $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$ be an algebraic system. The *congruence* of α is an equivalence relation ρ defined in A and satisfying the following condition:

$$\forall_{j \in \{1, \dots, m\}} ((x_1 \rho y_1) \wedge (x_2 \rho y_2) \wedge \dots \wedge (x_{nj} \rho y_{nj}) \Rightarrow o_j(x_1, \dots, x_{nj}) \rho\, o_j(y_1, \dots, y_{nj})) .$$

The last condition is said to be a *substitution property*[*]. The partition generated by the above congruence is called a *regular* one (or *partition with a substitution property*).

Definition 3.4 (*quotient algebraic system*)

Let ρ be a congruence of the above algebraic system α (see: Definition 3.3). The *quotient system* associated with α wrt ρ is defined as follows: $\alpha/\rho =_{df} (A/\rho ; [a_1]_\rho, [a_2]_\rho, \dots, [a_n]_\rho ; o_1', o_2', \dots, o_m')$, where:

$$\forall_{j \in \{1, \dots, m\}} \; \forall_{x_1, \dots . x_{nj} \in A} (o_j'([x_1]_\rho , \dots , [x_{nj}]_\rho) =_{df} [o_j(x_1, \dots, x_{nj})]_\rho .$$

Obviously, the above operations are well defined. In accordance with the last cited work, two examples are illustrated below.

Example 3.4 (*congruence*)

Consider the monoid[†] $\alpha =_{df} (\mathbb{Z} ; 0 ; +)$. Let $\rho \subseteq \mathbb{Z} \times \mathbb{Z}$ be a binary relation defined in $\mathbb{Z}$ as follows: $a \rho b$ iff $(a - b)$ is an even number. The *quotient algebraic system* $\alpha/\rho = (\{E, O\}; E ; \oplus)$, where E and O denote the subsets of *even* and *odd numbers*, respectively. The binary operation ' $\oplus$ ' is defined as it is shown in Figure 3.1 below. □

$\oplus$	E	O
E	E	O
O	O	E

Figure 3.1

Example 3.5 (*congruence*)

Consider the Boolean algebra[‡] $\alpha =_{df} (2^{\mathbb{N}} ; \varnothing , \mathbb{N} ; \cup , \cap , ')$. Let $\rho \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$ be defined as follows: $A \rho B \Leftrightarrow_{df} | A \div B | < \aleph_0$ (*alef-zero*). Then ρ is a congruence.[§] In fact, the last binary relation is at the same time a left and a right invariant ones wrt the above two operations: '$\cup$' and '$\cap$', i.e. for any $A, B, C \in 2^{\mathbb{N}}$:

$$\begin{aligned} (C \cup A) \div (C \cup B) &= (A \cup C) \div (B \cup C) \\ &= ((A \cup C)' \cap (B \cup C)) \cup ((A \cup C) \cap (B \cup C)') \end{aligned}$$

---

[*] In accordance with the last property, congruencies are also said to be *equivalence relations with the substitution property*.

[†] See Definition 2.8.

[‡] See Definition 2.23.

[§] $A \div B =_{df} (A - B) \cup (B - A)$: *symmetric set difference* (known also as *disjunctive union operation*) is here denoted by ' $\div$ ' (instead of '$\oplus$' used in the original work and also ' ρ ' instead of '$\equiv$ ').

$$
\begin{aligned}
&= (A' \cap B \cap C') \cup (A \cap B' \cap C') \\
&= ((A' \cap B) \cup (A \cap B')) \cap C' \\
&= (A \div B) \cap C'.
\end{aligned}
$$

In a similar way we can obtain:

$$
\begin{aligned}
(C \cap A) \div (C \cap B) &= (A \cap C) \div (B \cap C) \\
&= ((A \cap C)' \cap (B \cap C)) \cup ((A \cap C) \cap (B \cap C)') \\
&= ((A' \cup C') \cap (B \cap C)) \cup ((A \cap C) \cap (B' \cup C')) \\
&= A' \cap B \cap C \cup C' \cap B \cap C \cup A \cap C \cap B' \cup A \cap C \cap C' \\
&= A' \cap B \cap C \cup C' \cap B \cap C \cup A \cap C \cap B' \cup A \cap C \cap C' \\
&= A' \cap B \cap C \cup \varnothing \cup A \cap C \cap B' \cup \varnothing \\
&= A' \cap B \cap C \cup A \cap C \cap B' \\
&= (A' \cap B \cup A \cap B') \cap C \\
&= (A \div B) \cap C .
\end{aligned}
$$

And hence, if $A \div B$ is finite then the above two sets $(A \div B) \cap C'$ and $(A \div B) \cap C$ are also finite. And hence, we can obtain:

$$A \rho B \Rightarrow (((C \cup A) \rho (C \cup B)) \wedge ((A \cup C) \rho (B \cup C)) \wedge ((C \cap A) \rho (C \cap B)) \wedge ((A \cap C) \rho (B \cap C))).$$

Since $A \div B = A' \div B'$ then: $A \rho B \Rightarrow A' \rho B'$.

And hence, ρ is a congruence wrt the above considered Boolean algebra. □

The following two theorems were given (Kerntopf P. 1967).

Theorem 3.3

For any algebraic system $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$ and any congruence ρ in α there exists an epimorphism from α to α/ρ.

Proof:

Consider the map $\varphi : A \to A/\rho$ such that: $\varphi : x \to [x]$. This map preserves constants and operations. Moreover $\varphi(A) = A/\rho$. And hence, φ is an epimorphism from α to α/ρ. □

The map φ defined in the last proof is said to be a *natural homomorphism* (*epimorphism*). In general, it can be shown that the quotient systems associated with a given algebraic system (wrt the all its congruences) are exhaustive to isomorphism of all its homomorphic images.

Theorem 3.4

Let $\beta =_{df} (B ; b_1, b_2, \dots, b_n ; o_1', o_2', \dots, o_m')$ be an arbitrary homomorphic image of the algebraic system $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$ and $\varphi : A \to B$ be an epimorphism of α to β. Then there is such a congruence ρ of α, that the system β is isomorphic to $\alpha/\rho = (A/\rho ; [a_1], \dots, [a_n] ; o_1'', \dots, o_m'')$.

Proof:

Let $\rho \subseteq A \times A$ be defined as follows:

$$\forall_{x_1, x_2 \in A} \; (x_1 \rho x_2 \Leftrightarrow_{df} \varphi(x_1) = \varphi(x_2)).$$

In accordance with the last definition and the notion of a *homomorphism* (see Definition 3.1) we can obtain:

$$\forall_{j \in \{1, \dots, m\}} \; (((x_1 \rho x_1') \wedge (x_2 \rho x_2') \wedge \dots \wedge (x_{nj} \rho x_{nj}')) \Rightarrow o_j'(\varphi(x_1), \dots, \varphi(x_{nj})) =$$

$= o_j'(\varphi(x_1'), \dots, \varphi(x_{nj}'))$). Then: $\varphi(o_j(x_1, \dots, x_{nj})) = \varphi(o_j(x_1', \dots, x_{nj}'))$. We have: $o_j(x_1, \dots, x_{nj}) \rho\, o_j(x_1', \dots, x_{nj}')$. And hence, ρ is congruence.

Consider the map ψ assigning to each element $y \in B$ the class of equivalence of the congruence ρ consisting of counter images of y. The last map ψ is one-to-one (any two different $y_1, y_2 \in B$ are related to different elements of A/ρ). Moreover for any $y \in B$ there exists at least one counterimage (φ is an epimorphism). It is shown below that β is isomorphic to α/ρ. We have:

$$\forall_{i \in \{1, \dots, n\}} \; (\varphi(a_1) = b_i \Rightarrow \psi(b_i) = [a_i]_\rho) ,$$

$$\forall_{j \in \{1, \dots, m\}} \; \forall_{y_1, \dots, y_{nj} \in B} \; (((\varphi(x_1) = y_1) \wedge \dots \wedge (\varphi(x_{nj}) = y_{nj}) \wedge (\varphi(o_j(x_1, \dots, x_{nj}))) =$$

$$= o_j'(\varphi(x_1), \dots, \varphi(x_{nj})))) \Rightarrow \psi(o_j'(y_1, \dots, y_{nj})) = (o_j(x_1, \dots, x_{nj}))_\rho \Rightarrow$$

$$\Rightarrow \psi(o_j'(y_1, \dots, y_{nj})) = o_j''([x_1]_\rho, \dots, [x_{nj}]_\rho) \Rightarrow \psi(o_j'(y_1, \dots, y_{nj})) =$$

$$= o_j''(\psi(y_1), \dots, \psi(y_{nj})). \; \square$$

3.3. Finite direct products

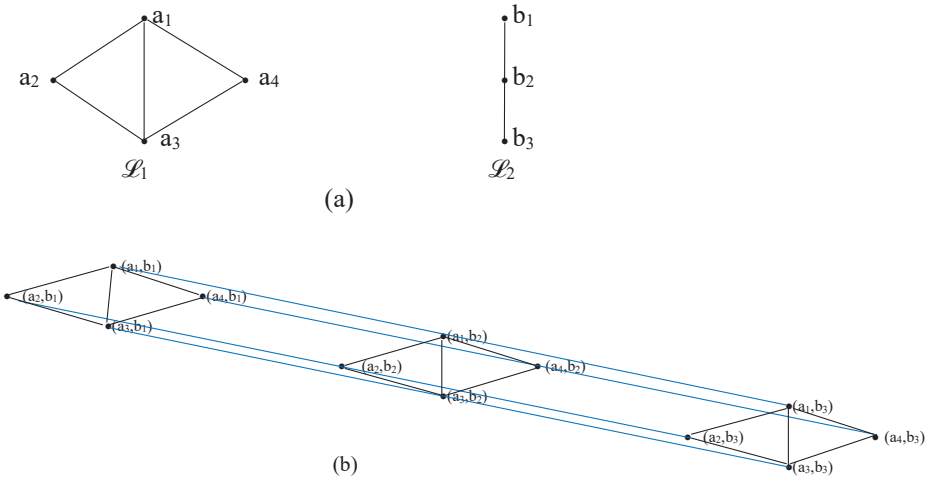
Definition 3.5 (*direct product*)

Let $\alpha =_{df} (A ; a_1, a_2, \dots, a_n ; o_1, o_2, \dots, o_m)$ and $\beta =_{df} (B ; b_1, b_2, \dots, b_n ; o_1', o_2', \dots, o_m')$ be two similar algebraic systems*. The *direct product* (or simply the *product*) of α and β, denoted by $\alpha \times \beta$, is defined as follows: $\alpha \times \beta =_{df} (A \times B ; (a_1,b_1), \dots, (a_n,b_n) ; o_1'', o_2'', \dots, o_m'')$, where:

$$\forall_{j \in (1, \dots, m\}} \; \forall_{x_1, \dots, x_{nj} \in A} \; \forall_{y_1, \dots, y_{nj} \in B} \; (o_j''((x_1,y_1), \dots, (x_{nj},y_{nj})) = (o_j(x_1, \dots, x_{nj}), o_j'(y_1, \dots, y_{nj}))).$$

In accordance with the last definition, the direct product of two similar algebraic systems is a new algebraic system belonging to the same category, e.g. the direct product of two groups is a group.

Example 3.6 (*direct product*)



* i.e. of the same *type*: see Definition 2.2.

Figure 3.2 (a) Two lattices: $\mathscr{L}_1$ and $\mathscr{L}_2$ (b) Hasse* diagram for $\mathscr{L}_1 \times \mathscr{L}_2$

Theorem 3.5

Let $\mathscr{B} =_{df} (B ; \wedge , \vee ; \sqcup , \sqcap ; ')$ be a finite Boolean algebra† such that $|B| \geq 2$. Then $\mathscr{B}$ is isomorphic to some nonegative power of the two-element Boolean algebra. □

Definition 3.6 (*atom*)

Let $\mathscr{L} =_{df} (L ; \sqcup , \sqcap ; \succcurlyeq)$ be a lattice having as an element $\wedge_{\mathscr{L}}$. Any other element of L immediately after $\wedge_{\mathscr{L}}$ is said to be an atom, i.e:

$$a \text{ is an } \textit{atom} \text{ of } \mathscr{L} \text{ iff } (a \neq \wedge_{\mathscr{L}}) \wedge \sim \exists_{x \in L} (\wedge_{\mathscr{L}} \prec x \prec a).$$

The proof of the above Theorem 3.5 is based on the following six lemmas given below (on atoms of *finite Boolean algebras*: the corresponding proofs are omitted. For any $x \in B$, by $A(x) =_{df} \{a_1, a_2, \dots , a_k\}$ it is denoted the set of all atoms 'a' of the Boolean algebra $\mathscr{B} =_{df} (B ; \wedge , \vee ; \sqcup , \sqcap ; ')$ such that: $a \preccurlyeq x$)‡.

Lemma 3.1

$$x \neq \wedge \Rightarrow \exists_{a \in A(x)} (a \preccurlyeq x). \square$$

Lemma 3.2

$$(a \in A(x)) \wedge (x \in B) \Rightarrow (a \preccurlyeq x) \veebar (a \sqcap x = \wedge)^{\S}. \square$$

Lemma 3.3

$$A(x \circ y) = A(x) \circ A(y) \text{ , where: } \circ \in \{ \sqcup , \sqcap \}. \square$$

Lemma 3.4

$$A(x') = A(\vee) - A(x) . \square$$

Lemma 3.5

$$A(x) = A(y) \text{ iff } x = y . \square$$

Lemma 3.6

Let $a_1 , a_2 , \dots , a_k$ be arbitrary and at the same time different atoms. Then: $A(a_1 \sqcup a_2 \sqcup \dots \sqcup a_k) = \{a_1, a_2, \dots ,a_k\}$. □

Proof of theorem 3.5:

Let $A =_{df} \{a_1, a_2, \dots , a_n\}$ be the set of all atoms of the above Boolean algebra $\mathscr{B} =_{df} (B ; \wedge , \vee ; \sqcup , \sqcap ; ')$ and $f : B \rightarrow 2^A$ be a map such that $f : x \rightarrow A(x)$. Since $\mathscr{B}$ is finite, by Lemma 5 it follows that f is *injective* (i.e one-to-one) and by Lemma 6 it follows that f is *surjective* (i.e. onto). Then f is bijective. And hence, f is an

---

* Helmut Hasse (1898 – 1979)

† See Definition 2.23

‡ (Kerntopf P. 1967)

§ By '$\veebar$' it is denoted the *exclusive disjunction* logical operation: is true iff arguments differ: one is true and the other false.

isomorphism of $\mathscr{B}$ to $(2^A ; \varnothing , A ; \cup , \cap , -)$, i.e. the corresponding operations are preserved (Lemmas 3 and 4). Moreover: $f(\wedge) = \varnothing$ and $f(\vee) = A$ (the constant preservation).

But the last system $(2^A ; \varnothing , A ; \cup , \cap , -)$ is isomorphic to the following one: $(\{\wedge , \vee\} ; \wedge , \vee ; \sqcup , \sqcap ; ')$, e.g. wrt the following map (for any $A' \subseteq A$): $\varphi(A') =_{df} (\varphi_1(A'), \dots , \varphi_n(A'))$, where $\varphi_i : 2^A \rightarrow \{\wedge , \vee\}$ (for any $i \in \{1, 2, \dots , n\}$) and $\varphi_i(A') =_{df}$ if $a_i \in A'$ then $\vee$ else $\wedge$. □

The following two corollaries are satisfied (in accordance with Theorem 3.5).

Corollary 3.1

The number of elements in any finite Boolean algebra is a non negative power of 2. □

Corollary 3.2

Any two finite Boolean algebras are isomorphic iff they have the same number of elements. □

3.4. Free algebraic systems

Definition 3.7 (*free algebraic system*)

Let S be a class of algebraic systems of the same type and $\alpha =_{df} (A ; a_1, a_2, \dots ,a_n ; o_1, o_2, \dots, o_m)$. We shall say that α is a *free algebraic system in the* S *class* if:

$$\exists_{A_0 \text{ of } \alpha} \quad \forall_{\beta \in S} \text{ (any map } f : A_0 \rightarrow B \text{ can be extended to a homomorphism of } \alpha \text{ to } \beta \text{),}$$

where: $\beta =_{df} (B ; b_1, b_2, \dots ,b_n ; o_1', o_2', \dots, o_m')$ and A_0 is said to be the set of *S-free generators* for α .

In accordance with Theorem 3.2, the above extention given by Definition 3.7 is uniquely defined. As an example, if a semigroup is free in any class of semigroups then any such semigroup is said to be a *free semigroup* (in a similar way for: groups, lattices, Boolean algebras, etc.). There are given in the next example some basic notions used in *automata theory* and *mathematical linguistics* (Kerntopf P. 1967).

Example 3.7 (*free semigroup*)

Let $\Sigma \neq \varnothing$ be an arbitrary nonempty set, called here: *alphabet*. The elements of Σ are said to be *letters*. Any *word* is defined as an arbitrary sequence of letters belonging to Σ . The *word's length* is the number of the corresponding letters. The set of all finite words in Σ is denoted by $\overline{\Sigma}$. The operation of linking two words $x_1x_2 \dots x_n$ and $y_1y_2 \dots y_m$ written side by side is said to be *concatency* (or *concatenation*) The concatency of words u and v is denoted by $u \cdot v$ (in short: uv). Obviously, the last binary operation is associative but not commutative. The following semigroup* $(\overline{\Sigma} , \cdot)$ is generated by Σ ($\overline{\Sigma}$ is the smallest set containing all words formed from the letters of Σ). It is shown below that $(\overline{\Sigma} , \cdot)$ is a *free semigroup*.

Let now $(A, +)$ be an arbitrary semigroup and $f : \Sigma \rightarrow A$ be an arbitrary map. The last map can be extended as follows:

$$\forall_{u \in \overline{\Sigma}} (u = x_1x_2 \dots x_n \rightarrow f(u) = f(x_1) + f(x_2) + \dots + f(x_n))$$

It can be also shown that each semigroup free in any class of semigroups is isomorphic to $(\overline{\Sigma} , \cdot)$, for a certain alphabet Σ . Usually, the last alphabet is extended by the so-called *empty word*, denoted by λ, where:

$$\forall_{u \in \overline{\Sigma} \cup \{\lambda\}} (u \cdot \lambda = \lambda \cdot u = u)$$

---

* See Definition 2.7

Let $\Sigma^* =_{df} \overline{\Sigma} \cup \{\lambda\}$. The following *free monoid* is obtained: $(\Sigma^* ; \lambda ; \cdot)$.

In accordance with the last considerations, any subset of Σ^* (including the empty set} is said to be an *event* over Σ. The system $(2^{\Sigma^*}; \varnothing , \Sigma^* ; \cup , \cap , ')$ is a *free Boolean algebra*. The following two operations, defined in 2^{Σ^*}, were also introduced.

The *concatenation* of two *events* $P, Q \in 2^{\Sigma^*}$ (denoted by $P \cdot Q$ or PQ) is the set $\{uv / (u \in P) \wedge (v \in Q)\}$. This operation is associative but not commutative. We have: $P^0 = \{ \lambda \}$ and $P^{i+1} = P^i \cdot P$ for $i \geq 0$.

The *iteration* P^* of an event $P \in 2^{\Sigma^*}$ is defined as the following event: $P^* =_{df} \{\lambda\} \cup P \cup P^2 \cup P^3 \cup \ldots = \bigcup_{i=0}^{\infty} P^i$.

Definition 3.8 (*algebra of events*)

The following algebraic system is said to be an *algebra of events*: $(2^{\Sigma^*}; \varnothing , \Sigma^* ; \cup , \cdot , ^*)$.

Definition 3.9 (*set of regular events*)

The smallest subset R of the set 2^{Σ^*} containing Σ^* and such that:

$$\forall_{P,Q \in R} (P \cup Q , PQ , P^* \in R)$$

Any element of R is said to be a *regular event*.

The above notion of a regular event introduced by Kleene* is one of the most important notions used in automata theory.

4. Applications

Some applications , e.g. such as: grammars, sequential machines, computability, etc. are briefly presented below..

4.1. Grammars and sequential machines†

Some introductory notions related to the last work are cited below. Here, any *alphabet* is considered as a finite set of symbols, e.g. {a, b, c, … , z} or also e.g. as a binary alphabet {0, 1}. Any *string* over an alphabet is defined as a finite sequence of symbols drawn from the considered alphabet, e.g. happybirthdaytoyou or e.g. the binary string: 101100101, etc. It is generally omitted '' '' from strings unless doing so would lead to confusion. The set of all possible strings over an alphabet $\sum$ is written by $\sum^*$. The length of a string is defined as the number of symbols in it. As an example, we have: the *length*(101100101) = 9. Some operations on strings are cited below.

Concatenation of two strings x and y: appending the string y to the string x.

---

* Stephen Cole Kleene (1909 – 1994)

†https://docest.com/grammars-languages-and-machines

Replication: for each string w and each natural number i, the string w^i is defined as follows:

$w^0 = \varepsilon$
$w^i = w^{i-1}w$, for any $i \geq 1$.

The replication operator has a high precedence (like exponentiation). The notion of *string reversal* was introduced in accordance with the following inductive definition.

(1) if $|w| = 0$ then $w^R = w = \varepsilon$

(2) if $|w| \geq 1$ then $\exists a \in \sum: w = ua$ (a is the last character of w) and $w^R = au^R$.

The following property was shown.

Theorem

Let w,x be two strings. Then $(wx)^R = x^R w^R$. □

The notion of a *language* is introduced as a '(finite or infinite) set of finite length strings of a finite alphabet $\sum$. The language $\sum^*$ contains an infinite number strings (including: ε, a, b, ab, …). Languages can be considered as sets. And hence, in the next considerations there were presented some techniques for defining languages and also: 'how large are languages, operations on languages, and finally regular expressions, regular grammars and finite state machines': left to the reader.

4.2. Computability and recursiom*

In accordance with the last work, it was considered the informal concept of "*computability*" or "*effective calculability*" and two of the formalisms commonly used to define it: "(*Turing*) *computability*" and "(*general*) *recursiveness*." There were considered their origin, exact technical definition, concepts, history, general English meanings, how they became fixed in their present roles, how they were first and are now used, their impact on nonspecialists, how their use will affect the future content of the subject of computability theory, and its connection to other related areas. After a careful historical and conceptual analysis of computability and recursion there were maked several recommendations about preserving the intensional differences between the concepts of "*computability*" and "*recursion*." In particular, it was recommend that: the term "recursive" should no longer carry the additional meaning of "computable" or "decidable;" functions defined using Turing machines, register machines, or their variants should be called "computable" rather than "recursive". Moreover, it was distinguished the intensional difference between Church's Thesis and Turing's Thesis, and use the latter particularly in dealing with mechanistic questions; the name of the subject should be "Computability Theory" or simply Computability rather than "Recursive Function Theory."†

4.3. Graph theory and Petri nets

"In mathematics, *graph theory* is the study of *graphs*, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of *vertices* (also called *nodes* or *points*) which are connected by *edges* (also called *links* or *lines*). A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. Graphs are one of

---

*Soare R.I., 10th International congress for logic, methodology and philosophy of science. Section3: Recursion theory and constructivism, August (1995) 19 – 25: http://www.people.cs.uchicago.edu/~soare/History/compute.pdf

†"*Computability* is the ability to solve a problem in an effective manner. It is a key topic of the field of computability theory within mathematical logic and the theory of computation within computer science. The computability of a problem is closely linked to the existence of an algorithm to solve the problem. The most widely studied models of computability are the Turing - computable and μ – recursive functions and the lambda calculus (all having computationally equivalent power): Wikipedia.

the principal objects of study in discrete mathematics". Petri net (also known as a *place/transition net*) is one of the several mathematical modelling languages used for description of distributed systems (a class of discrete event dynamic systems). Any Petri net is introduced as a directed bipartite graph having two types of elements: *places* and *transitions* (depicted by circles and rectangles, respectively): depicted as white circles and rectangles, respectively. A place can contain any number of tokens, depicted as black circles. A transition is *enabled* if all places connected to it as inputs contain at least one token. Petri nets were invented in August 1939 by Carl Adam Petri*: for the purpose of describing chemical processes.

4.4. Combinatorial analysis and probability theory

The *combinatorial analysis* is a "branch of mathematics devoted to the solution of problems of choosing and arranging the elements of certain (usually finite) sets in accordance with prescribed rules. Each such rule defines a method of constructing some configuration of elements of the given set, called a *combinatorial configuration*. One can therefore say that the aim of *combinatorial analysis* is the study of combinatorial configurations. This study includes questions of the existence of combinatorial configurations, algorithms and their construction, optimalisation of such algorithms, as well as the solution of problems of enumeration, in particular the determination of the number of configurations of a given class.The simplest examples of combinatorial configurations are permutations, combinations and arrangements." †

*Probability theory i*s the "branch of mathematics concerned with probability. Although there are several different probability interpretations, probability theory treats the concept in a rigorous mathematical manner by expressing it through a set of axioms. Typically these axioms formalise probability in terms of a probability space, which assigns a measure taking values between 0 and 1, termed the probability measure, to a set of outcomes called the sample space. Any specified subset of the sample space is called an event. Central subjects in probability theory include discrete and continuous random variables, probability distributions, and stochastic processes."‡

4.5. Number theory, Markov's chains, coding

Number theory (or arithmetic or higher arithmetic in older usage) is a branch of pure mathematics devoted primarly to the study of the integers and integer-valued functions. Mathematics is the queen of the sciences and number theory is the queen of mathematics.§ Number theorists study prime numbers as well as the properties of mathematical objects made out of integers (e.g. *rational numbers*) or defined as generalisations of integers (e.g. algebraic integers): Wikipedia.

A *Markow's chain*** is 'a mathematical system that "experiences transitions from one state to another are in accordance with certain probabilistic rules. The defining characteristic of a Markov chain is that no matter *how* the process arrived at its present state, the possible future states are fixed. In other words, the probability of transitioning to any particular state is dependent solely on the current state and time elapsed. The state space, or set of all possible states, can be anything: letters, numbers, weather conditions, baseball scores, or stock performances. … Markow's chains may be modelled by finite state machines, and random walks provide a prolific example of their usefulness in mathematics. They arise broadly in statistical and information-theoretical contexts and are widely employed in economics, game theory, queueing (communication) theory, genetics, and finance.

---

* Carl Adam Petri (1926 – 2010)

† https://encyclopediaofmath.org/wiki/Combinatorial_analysis

‡ https://en.wikipedia.org/wiki/Probability_theory

§ Carl Friedrich Gauss (1777 – 1855)

** https://brilliant.org/wiki/markov-chains/: Andrey Andreyevich Markow (1856 – 1922)

While it is possible to discuss Markov's chains with any size of state space, the initial theory and most applications are focused on cases with a finite (or countably infinite number of states. "

Coding theory is 'an important study which attempts to minimize data loss due to errors introduced in transmission from noise, interference or other forces. With a wide range of theoretical and practical applications from digital data transmission to modern medical research, coding theory has helped enable much of the growth in the 20th century. *Data encoding* is accomplished by adding additional information to each transmitted message to enable the message to be decoded even if errors occur. In 1948 the optimization of this redundant data was discussed by Claude Shannon* from Bell Laboratories in the United States, but it wouldn't be until 1950 that Richard Hamming† (also from Bell Labs) would publish his work describing a now famous group of optimized linear codes, the Hamming Codes. It is said he developed this code to help correct errors in punch tape. Around the same time John Leech from Cambridge was describing similar codes in his work on group theory' (Grossman J. 2008).

4.6. Algorithm complexity

'Algorithmic complexity is a measure of how long an algorithm would take to complete given an input of size n. If an algorithm has to scale, it should compute the result within a finite and practical time bound even for large values of n. For this reason, complexity is calculated asymptotically as n approaches infinity. While complexity is usually in terms of time, sometimes complexity is also analyzed in terms of space, which translates to the algorithm's memory requirements. Analysis of an algorithm's complexity is helpful when comparing algorithms or seeking improvements. Algorithmic complexity falls within a branch of theoretical computer science called computational complexity theory. It's important to note that we're concerned about the order of an algorithm's complexity, not the actual execution time in terms of milliseconds. Algorithmic complexity is also called *complexity* or *running time*'‡.

'In algorithmic information theory (a subfield of computer science and mathematics), the *Kolmogorov complexity* of an object, such as a piece of text, is the length of a shortest computer program (in a predetermined programming language) that produces the object as output. It is a measure of the computational resources §needed to specify the object, and is also known as algorithmic complexity, Solomonoff–Kolmogorov–Chaitin complexity, program-size complexity, descriptive complexity, or algorithmic entropy. It is named after Andrey Kolmogorov**, who first published on the subject in 1963.The notion of Kolmogorov complexity can be used to state and prove impossibility results akin to Cantor's diagonal argument, Gödel's incompleteness theorem, and Turing's†† halting problem'‡‡.

---

* Claude Elwood Shannon (1916 – 2001)

† Richard Hamming (1915 – 1998)

‡ https://devopedia.org/algorithmic-complexity

** Andrey Nikolaevich Kolmogorov (1903 – 1987)

†† Alan Turing (1912 – 1954)

‡‡ https://en.wikipedia.org/wiki/Algorithmic_complexity

Conclusions

Discrete mathematical structures underpin a large amount of modern computer science. Unfortunately, the speedy developments and knowledge in this area makes impossible the presentation of all notions, definitions and applications used here. This part is an extension of the previous one (i.e. a supplement / a separate work) and it is related to algebraic systems (considered as *discrete mathematical structures*). The considered here systems may be useful for any researcher who is interested in the above given area.

References

Akram M., *Fuzzy Lie Algebras*, Springer (2018) 302pp.

Barbieri G. and Weber H., *Measures on clans and on MV – algebras.* Handbook of measure theory: Chapter 22, Pad E. ed., Elsevier, (2002) 1632pp.
https://www.sciencedirect.com/topics/mathematics/mv-algebra

Bronstein I.N., Semendjajew K.A., Musiol G. and Mühlig H., Taschenbuch der Mathematik. Verlag Harri Deutsch (2001) 1149pp.

Chang C.C., *Algebraic analysis of many – valued logics*. Transactions of the American Mathematical Society, no. 88 (1958) 476 – 490.

Cignoli R., *Proper n-Valued Łukasiewicz Algebras as S-Algebras of Łukasiewicz n-Valued Propositional Calculi*, Studia Logica, 41, 1982, 3–16

Gottwald S., *Many-valued logic*. The Stanford Encyclopaedia of Philosophy, Zalta E.N. ed., The Metaphysics Research Lab at the Center for the Study of Language and Information, Stanford University, Stanford, CA (2000) 14pp.

Grossman J., *Coding theory: introduction to linear codes and applications*. Rivier Academic Journal, vol. 4, number 2, FALL (2008) 17pp.

Hájek P., *Metamathematics of fuzzy logic*. Kluwer Acad.Publ., Dordrecht (1998) 297pp.

Iorgulescu A., *Connections between MV_n algebras and n-valued Łukasiewicz – Moisil algebras – II*, Discrete Mathematics, Science Direct, North – Holland, vol. 202 (1999) 113 – 134.

Kempner A. J., *Miscellanea*. Amer. Math. Monthly no. 25 (1918) 201 – 210.

Kerntopf P., *Basic mathematical notions in automata theory. Relations, algebraic systems and structures*. CO PAN, PWN Warszawa (1967) 64pp. In Polish (Library of Congress Catalog: QA402.3.K4).

Lu H., *Fault detection in multi-valid digital circuits*. M.S. thesis, Univ. of Oklahoma, U.S.A. (1983).

Lu H. and Lee S,C., *Fault detection in M-logic circuits using the M-difference*. Proc. of the IEEE International Symposium on Multiple-Valued Logic (1985) 62 – 70 / *M-algebra*: 272 – 284.

Lucas E., *Question Nr. 288*. Mathesis no. 3 (1883) 232.

Mordeson J. N. and Malik, D.S., *Fuzzy automata and languages: Theory and Applications*, Chapman & Hall / CRC, New York (2002) 576pp.
https://www.taylorfrancis.com/books/mono/10.1201/9781420035643/fuzzy-automata-languages-john-mordeson-davender-malik

Mundici D., *Averaging the truth-value in Łukasiewicz logic*. Studia Logica, vol. 55, no. 1 (1995) 113 – 127.

Neuberg J., *Solutions de questions proposes*. *Question Nr. 288*, Mathesis **7** (1887) 68-69.

Plantz R.G., *Introduction to computer organization: ARM assembly language using the raspberry pi.* Sonoma State University (2021) CS 252. https://bob.cs.sonoma.edu/IntroCompOrg-RPi/sec-balgebra.html

Rose A., *Formalisation du calcul propositionnel implicatif* $\aleph_0$ *valeurs de Łukasiewicz, C. R.* Acad. Sci. Paris 243 (1956) 1183 – 1185.

Tabakow I.G., *Using D-algebra to generate tests for m-logic combinational circuits*. International Journal of Electronics 75, UK (1993) 897 – 906.

Vasantha Kandasamy W.B. , *Groupoids and Smarandache groupoids*. American Research Press, Rehoboth, NM (2002) 114pp.
http://fs.unm.edu/Vasantha-Book2.pdf

Vasantha Kandasamy W.B., *Smarandache fuzzy algebra*. American research press, Rehoboth (2003) 454pp.
http://fs.unm.edu/Vasantha-Book9.pdf

Vasantha W.B. and Chetry M,K., *Smarandache free groupoids and its application to automaton*.The University Grants Commision sponsored seminar, Jamal Mohammed College Tiruchi, TN India (2004)
http://fs.unm.edu/SN/SmarandacheFreeGroupoids.pdf

WUST Publishing House
mailorder:
zamawianie.ksiazek@pwr.edu.pl

ISBN 978-83-7493-221-9