Far East Journal of Mathematical Sciences (FJMS)
Volume 34, Issue 3, 2009, Pages 289-302

FUNDAMENTALS OF TERNARY LOGIC

JORGE PEDRAZA ARPASI

Departamento de Ciências e Engenharias
Universidade Regional Integrada - URI
Frederico Westphalen, RS, Brazil
e-mail: arpasi@gmail.com

Abstract

In this article, we show that ternary logic is a generalization of the binary (classic) case in the sense of each proposition that is true under the rules of the binary logic will be true under the rules of the trivalent case. Also, we show that the tautologies and contradictions are relatively few when compared with the binary case. Finally, it is shown that ternary logic cannot have a Boolean structure.

1. Introduction

A ternary logic system was first proposed by the polish mathematician Jan Łukasiewicz, in 1920, in his celebrated paper *O Logice Trójwarkoscioewj*, [8], as a generalization of the classical binary logic. After that, Łukasiewicz proposed a more generalized logical system: the multivalued logic. In 1965, Lotfi Zadeh introduced the Fuzzy set theory, in [9], from which was derived the Fuzzy logic. Both, multivalued and Fuzzy logic have the same rules, the only difference is in their respective aims [3].

In connection with ternary logic, Donald Knuth assumed that the replacement of “flip-flop” for “flip-flap-flop” one a “good” day will nevertheless happen [7]. Hence, to move beyond binary logic to circuits based on multivalued logic in which the information density and processing efficiency of a circuit could theoretically be

---

2000 Mathematics Subject Classification: 03Bxx.

Keywords and phrases: ternary logic, fuzzy logic, binary logic, VLSI circuits.

Received April 28, 2009

increased substantially without any further expensive "improvements" to the underlying fabrication technology, is an idea that must be taken into account. Elementary calculations allow us to say that a 16-bit microcomputer with on-board memory has access to no more than 65 Kbytes of directly accessible memory, while that same microcomputer with memory based on ternary logic would have direct access to 43 Mbytes of memory, that means, one gains more than 656 times of memory capacity.

In 1956, in the academy of sciences of the former USSR was initiated the construction of the first ternary digital machine, the *Setun*. But the officials of the USSR did not believe in this project and it was stopped by 1965 after producing about 30 Setun machines. These ternary computers were replaced by binary ones of the same performance, but more than 2.5 times more expensive [1, 2].

In this work, we will describe some basic facts about binary logic and Boolean algebra in spite of both issues being widely known, but this description will be a good basement to introduce and to show the elementary properties of ternary logic. These properties can be summarized as follows: (a) Ternary logic is a generalization of binary logic, (b) it has not a structure to be a Boolean algebra, (c) it is based on more than three basic operations, and (d) its tautologies and contradictions are more hard-to-find propositions with respect to the binary case. The generalization is in the sense that if one proposition p is true(false) under the rules of binary logic, then it is also true(false) under the ternary logic. The lack of Boolean structure, in ternary logic, is compensated by more powerful tools for inferential analysis [6]. Recently some researchers are working in the building of one structure called *trilean*, which could have some analogous properties from Boolean structure [5].

For the binary case, we will use the values $\{0, 1\}$ which means $\text{true} = 1$ and $\text{false} = 0,$ whereas that for ternary case we will use the values $\left\{0, \frac{1}{2}, 1\right\}$ which means $\text{true} = 1,$ $\text{false} = 0,$ and $\text{“unknown”} = \frac{1}{2}.$ By x we will mean a simple statement or proposition, whereas by either p or f we will mean a composed proposition which depends on other simple propositions, hence we also refer them as functions.

2. Binary Logic (Classical Logic)

We can see the binary logic as a system $\mathcal{L}$ whose elements called *propositions* or *statements* are valued on the set $\{0, 1\}$ [4]. This set $\{0, 1\}$ we denote as being $\mathbb{Z}_2$, thus, if *x* is a proposition, the value of *x* is a mapping $\nu : \mathcal{L} \to \{0, 1\}$ such that

$$\nu(x) = \begin{cases} 1; & \text{if } x \text{ is true,} \\ 0; & \text{if } x \text{ is false.} \end{cases}$$

It is standard, in almost all the logical literature, to ignore the mapping ν. Therefore, for the sake of practical purposes, it is assumed that $\nu(x) = x$ and from this, $x = 1$ means *x* is true and $x = 0$ means *x* is false. Over $\mathcal{L}$ are defined the following basic operations:

- The negation $\neg$ (unary operation "not").
- The disjunction $\vee$ (binary operation "or").
- The conjunction $\wedge$ (binary operation "and").

Table 1. Most known operations or functions in binary logic

x	y	Neg(x) $\neg x$	Disj(x, y) $x \vee y$	Conj(x, y) $x \wedge y$	Imp(x, y) $x \to y$	Equiv(x, y) $x \leftrightarrow y$
1	1	0	1	1	1	1
1	0		1	0	0	0
0	1	1	1	0	1	0
0	0		0	0	0	1

The system $\mathcal{L}$ is closed under any of these three operations, in the sense that if $x, y \in \mathcal{L}$, then both $\neg x \in \mathcal{L}$, $x \vee y \in \mathcal{L}$, and $x \wedge y \in \mathcal{L}$. From these three operations, we can derive 16 binary operations, among them the implication $\text{Imp}(x, y) = x \to y$ and the equivalence $\text{Equiv}(x, y) = x \leftrightarrow y$. The value of these basic operations and any other composed propositions depend on the value of each component *x* and *y* as can be seen in the *truth table* for these operations shown in Table 1.

One practical advantage of the use of the identity $\nu(x) = x$, is that the above basic operations can be considered functions. In this way, the unary operator "negation" is a function $f : \mathbb{Z}_2 \to \mathbb{Z}_2$, and a binary operation such as the "disjunction" is a function $f : \mathbb{Z}_2^2 \to \mathbb{Z}_2$. In general, by combining the three basic operations we can define *binary logic functions* as mappings $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$.

When $n = 1$, we have one-variable functions $f(x)$, and there are $2^{2^1} = 4$ of this kind of functions which are: the Identity or Affirmation $id(x)$, the Negation $N(x)$, the Tautology $\tau(x)$ and the Contradiction $\gamma(x)$. All these four functions are also called *modal* functions of *x* and they are shown in Table 2.

Table 2. All the one-variable binary functions

x	$id(x)$	$N(x)$	$\tau(x)$	$\gamma(x)$
1	1	0	1	0
0	0	1	1	0

When $n = 2$, we have two-variable functions $f(x, y)$, and there are $2^{2^2} = 16$ of this kind of functions which are shown in Table 3. It is possible to show that all the 16 two-variable functions can be derived from that basic functions "and" $(\wedge)$, "or" $(\vee)$ and "not" $(\neg)$. Notice, in that table, that $f_2(x, y)$ is the disjunction, $f_8(x, y)$ is the conjunction, $f_5(x, y)$ is the implication, f_7 is the equivalence, whereas f_1 and f_{16} are the tautology and the contradiction functions, respectively.

Table 3. All the two-variable binary functions

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

For three variables we will have $2^{2^3} = 256$ different functions $f(x, y, z)$. In general, there exist 2^{2^n} different binary logical functions $f(x_1, x_2, ..., x_n)$ of *n* variables.

Example 1. Functions or propositions of one variable.

Let *x* be the statement "*Peter is tall*". Then we can construct all the four one-variable functions:

- $id(x) =$ *Peter is tall.*
- $N(x) = \neg x =$ *Peter is not tall.*

- $\tau(x) = x \vee \neg x =$ *Peter is tall or he is not tall.*
- $\gamma(x) = x \wedge \neg x =$ *Peter is tall and he is not tall.*

Example 2. Some functions or propositions of two variables.

Let x and y be the statements "*Peter is tall*" and "*Peter is thin*", respectively. Then we can construct the following functions, among all the 16 possible ones (Table 3):

- $f_2(x, y) = x \vee y =$ *Peter is tall or thin* (disjunction).
- $f_8(x, y) = x \wedge y =$ *Peter is tall and thin* (conjunction).
- $f_5(x, y) = x \rightarrow y =$ *If Peter is tall then he is thin* (implication).
- $f_7(x, y) = x \leftrightarrow y =$ *Peter is tall if and only if he is thin* (equivalence).
- $f_{10}(x, y) = XOR(x, y) =$ *Either Peter is tall or Peter is thin* (Exclusive OR).
- $f_{15}(x, y) = NAND(x, y) =$ *Peter is not tall or Peter is not thin* (Negation of Conjunction).
- $f_1(x, y) = \tau(x, y) = (x \wedge y) \vee (\neg x \vee \neg y) =$ *Peter is tall and thin*, *or he is not tall or he is not thin* (tautology).
- $f_{16}(x, y) = \gamma(x, y) = (x \vee y) \wedge (\neg x \wedge \neg y) =$ *Peter is tall or thin*, *and he is not tall and he is not thin* (contradiction).

Table 4. Boolean properties of binary logic, where τ = tautology and γ = contradiction

1	$x \vee y = y \vee x$	5	$x \wedge y = y \wedge x$
2	$(x \vee y) \vee z = x \vee (y \vee z)$	6	$(x \wedge y) \wedge z = x \wedge (y \wedge z)$
3	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	7	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
4	$x \vee \gamma = x$	8	$x \wedge \tau = x$
9	$x \vee \neg x = \tau$	10	$x \wedge \neg x = \gamma$

3. Boolean Algebra

Definition 1. Let S be a set. Then the *power set* of S, denoted by $\mathcal{P}(S)$, is the set of all subsets of S.

If the set S is finite, with n elements, then the power set $\mathcal{P}(S)$ has 2^n elements. In other words, each set with n elements has 2^n different subsets. Obviously, an infinite set S has a power set which is also infinite.

Example 3. Consider the finite set $S = \{a, b, c\}$, a set with three elements.

The power set $\mathcal{P}(S)$ has $2^3 = 8$ elements(subsets). In effect,

$$\mathcal{P}(S) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\},$$

where $\varnothing$ is the empty set and $\{a, b, c\}$ is the full set S itself.

Definition 2. An Algebra of Boole is a nonempty set $\mathcal{B}$ with two binary operations, the sum $(+)$, and the product $(\cdot)$, and one unary operation, the complement $(')$; satisfying the following conditions:

(1) The sum is commutative, that is, $x + y = y + x$, for all $x, y \in \mathcal{B}$.

(2) The sum is associative, that is, $(x + y) + z = x + (y + z)$, for all $x, y, z \in \mathcal{B}$.

(3) The sum is distributive with respect to the product, that is, $x + (y \cdot z) = (x + y) \cdot (x + z)$, for all $x, y, z \in \mathcal{B}$.

(4) There exists a neutral element for the sum, $0 \in \mathcal{B}$, such that $x + 0 = x$, for all $x \in \mathcal{B}$.

(5) The product is commutative, that is, $x \cdot y = y \cdot x$, for all $x, y \in \mathcal{B}$.

(6) The product is associative, that is, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, for all $x, y, z \in \mathcal{B}$.

(7) The product is distributive with respect to the sum, that is, $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$, for all $x, y, z \in \mathcal{B}$.

(8) There exists a neutral element for the product, $1 \in \mathcal{B}$, such that $x \cdot 1 = x$, for all $x \in \mathcal{B}$.

(9) $x + x' = 1$, for all $x \in \mathcal{B}$.

(10) $x \cdot x' = 0$, for all $x \in \mathcal{B}$.

Table 5. Operations and identities of binary logic and Boolean algebra

Disjunction	$\vee$	$\mapsto$	$+$	Boolean sum
Conjunction	$\wedge$	$\mapsto$	$\cdot$	Boolean product
Negation	$\neg$	$\mapsto$	$'$	Boolean complement
Tautology	τ	$\mapsto$	1	Product identity
Contradiction	γ	$\mapsto$	0	Sum identity

3.1. Examples

Example 4. Consider a class $\mathcal{L}$ of binary logical propositions, that is, with values on $\mathbb{Z}_2 = \{0, 1\}$, with the operations negation $(\neg)$, disjunction $(\vee)$, and conjunction $(\wedge)$, then $\mathcal{L}$ with these operations is an infinite Boolean algebra.

By making the correspondences of Table 5, we can verify that the properties of $\mathcal{L}$ shown in Table 4 satisfy the definition of Boolean algebra.

Example 5. Consider a finite set S, then the power set $\mathcal{P}(S)$; with the set operations: union of sets $\bigcup$, the intersection of sets $\bigcap$, and the complement of a set $'$; is a finite Boolean algebra.

Table 6. The Boolean structure for sets with the operations $'$, $\bigcup$, and $\bigcap$

1	$A \bigcup B = B \bigcup A$	5	$A \bigcap B = B \bigcap A$
2	$(A \bigcup B) \bigcup C = A \bigcup (B \bigcup C)$	6	$(A \bigcap B) \bigcap C = A \bigcap (B \bigcap C)$
3	$A \bigcup (B \bigcap C) = (A \bigcup B) \bigcap (A \bigcup C)$	7	$A \bigcap (B \bigcup C) = (A \bigcap B) \bigcup (A \bigcap C)$
4	$A \bigcup \varnothing = A$	8	$A \bigcap \mathcal{P}(S) = A$
9	$A \bigcup A' = \mathcal{P}(S)$	10	$A \bigcap A' = \varnothing$

In effect, any subsets A, B and C of S hold the properties of Table 6 and therefore $\mathcal{P}(S)$ is a finite Boolean Algebra.

4. Ternary Logic

If binary logic is valued in $\mathbb{Z}_2 = \{0, 1\}$, where $1 =$ true is the best case and $0 =$ false is the worst case, then the set of values of ternary could be the set $\mathbb{Z}_3 = \{0, 1, 2\}$. These values could be made in such a way that $2 =$ false (worst),

$1 =$ true (best), and $0 =$ unknown (intermediate). Some advantages of this representation, of ternary logic, are that the values are integers, and $\mathbb{Z}_3$ is an algebraic ring where $2 = -1$. Thus the set of values $\{0, 1, 2\}$ commutes easily with $\{-1, 0, 1\}$. These last values are more convenient for the people who implement this logic in electrical circuits where would be used alternating current which varies in direction cyclically. One direction would be -1, the opposite direction would be $+1$ whereas the current off would be 0 [2, 6]. In this work, we will use the set $\left\{0, \frac{1}{2}, 1\right\}$, where $0 =$ false, $\frac{1}{2} =$ unknown and $1 =$ true. We think that the choice of these values leads in a more natural way the adaptation of the ideas from the binary logic. Moreover, Łukasiewicz himself proposed this notation in his celebrated work [3, 8].

Definition 3. As ternary logic we will mean a system $\mathcal{L}$ whose elements called *propositions* or *statements* are valued in the set $\left\{0, \frac{1}{2}, 1\right\}$. If x is a proposition, then the value of x is a mapping $\nu : \mathcal{L} \to \left\{0, \frac{1}{2}, 1\right\}$ such that

$$\nu(x) = \begin{cases} 1; & \text{if } x \text{ is true,} \\ \frac{1}{2}; & \text{if } x \text{ is unknown,} \\ 0; & \text{if } x \text{ is false.} \end{cases}$$

From this, we have that if $\nu(x) = 1$ (true) under the rules of binary logic, then also $\nu(x) = 1$ (true) under the ternary logic laws. Analogously for the false value. On the other hand, for the same considerations made for binary logic case, we can avoid ν by making $\nu(x) = x$. Then over $\mathcal{L}$ are defined the following basic operations [6, 3]:

- The negation $\neg$ (unary operation “not”).
- The disjunction $\vee$ (binary operation “or”).
- The conjunction $\wedge$ (binary operation “and”).
- The implication $\to$ (binary operation “if...then”).

Table 7. Basic operations of the ternary logic

x	y	Neg(x) $\neg x$	Conj(x, y) $x \wedge y$	Disj(x, y) $x \vee y$	Imp(x, y) $x \to y$	Equiv(x, y) $x \leftrightarrow y$
1	1	0	1	1	1	1
1	$\frac{1}{2}$		$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$
1	0		0	1	0	0
$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	1	1	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$		$\frac{1}{2}$	$\frac{1}{2}$	1	1
$\frac{1}{2}$	0		0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
0	1	1	0	1	1	0
0	$\frac{1}{2}$		0	$\frac{1}{2}$	1	$\frac{1}{2}$
0	0		0	0	1	1

The system $\mathcal{L}$ is closed under any of these four operations, in the sense that if $x, y \in \mathcal{L}$, then $\neg x \in \mathcal{L}$, $x \vee y \in \mathcal{L}$, $x \wedge y \in \mathcal{L}$, and $x \to y \in \mathcal{L}$. Notice that the implication, in this case, is not derived from the three basic operations as it happens in the binary logic.

The values of $\neg x$, $x \vee y$, $x \wedge y$, and $x \to y$ and other composed operations depend on the value of each component x and y. These values can be obtained by using the *truth table* as Table 7 shows. In such Table 7 also it is shown the equivalence operation which can be derived from the conjunction and implication.

Remembering binary logic, notice that for $\text{Disj}(x, y)$ be true it is enough that only one of the two variables be true. From this, we can say that $\text{Disj}(x, y) = \max\{x, y\}$, whereas for the conjunction we have $\text{Conj}(x, y) = \min\{x, y\}$.

Since ternary logic is an extension of binary logic in the sense that each proposition x that is true(false) in binary logic also must be true(false) in ternary logic, the disjunction $\text{Disj}(x, y)$ and conjunction $\text{Conj}(x, y)$, for the ternary case, must have similar behavior. From this $\text{Disj}(x, y) = \max\{x, y\}$ and $\text{Conj}(x, y) = \min\{x, y\}$. These facts are shown in Table 7.

As in the binary case the operators can be considered functions. The unary operator negation is a function $f : \mathbb{Z}_3 \to \mathbb{Z}_3$, and a binary operator such as the disjunction is a function $f : \mathbb{Z}_3^2 \to \mathbb{Z}_3$. In general, we can define *ternary logical functions* with *n* variables as mappings $f : \mathbb{Z}_3^n \to \mathbb{Z}_3$.

When $n = 1$, we have one-variable functions $f(x)$, and there are $3^{3^1} = 27$ different functions, among them are, the Identity or Affirmation $id(x)$, the Negation $N(x)$, the Tautology $\tau(x)$ and the contradiction $\gamma(x)$. All these 27 functions are also called *modal functions* of *x* and they are shown in Table 8.

Table 8. All the one-variable ternary functions

x	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1	1	1	1	1	1	1	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
0	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$

x	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	0	0	0	0	0	0	0	0	0
1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	1	1	1
0	1	$\frac{1}{2}$	0	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$

When $n = 2$, we have two-variable functions $f(x, y)$, and there are $3^{3^2} =$ 19683 of them. It is impossible, in a single page, to show the truth table for each one.

In the same way, we can compute that there are $3^{3^3} = 7625597484987$ three-variable different functions. In general, there exist 3^{3^n} different ternary logical functions $f(x_1, x_2, ..., x_n)$ of *n* variables.

Example 6. Some functions or propositions of one variable.

Let *x* be the simple statement "*it is raining*". Then we show 6 of the 27 one-variable ternary functions $f(x)$:

- $f_1(x) = id(x) =$ *It is raining* (affirmation).
- $f_{20}(x) = N(x) = \neg x =$ *It is not raining* (negation).
- $f_8(x) = \tau(x) = x \to x =$ *If it is raining then it is raining* (tautology).
- $f_{22}(x) = \gamma(x) = \neg(x \to x) =$ *It is not true that if it is raining then it is raining* (contradiction).
- $f_2(x) = x \vee \neg x =$ *It is raining, or it is not raining.*

  This function is not a tautology as it happens in the binary logic case. Also, this example shows that condition (9) of the definition of Algebra of Boole cannot be held by the ternary logic system. Therefore, the ternary logic system cannot have a Boolean Algebra structure.

- $f_{19}(x) = x \wedge \neg x =$ *It is raining, and it is not raining.*

  This function is not a contradiction as it happens in the binary logic case. Also, this example shows that condition (10) of the definition of Algebra of Boole cannot be held by the ternary logic system. Therefore, the ternary logic system cannot have a Boolean Algebra structure.

Example 7. Some functions or propositions of two variables.

Let *x*, *y* be the propositions "*it is raining*" and "*the sun is shining*", respectively. Then we write some of the 19683 two-variable functions:

- $\tau(x, y) = \neg(x \vee y) \leftrightarrow (\neg x \wedge \neg y) =$ *It is not true that it is raining or the sun is shining, if and only if it is not raining and the sun is not shining.*

  By making $m_{11}(x, y) = \neg(x \vee y)$ and $m_{12}(x, y) = \neg x \wedge \neg y$, we can verify in Table 9 that this is a tautology called the *first law of De Morgan*. Analogously for $m_{21}(x, y) = \neg(x \wedge y)$, and $m_{22}(x, y) = \neg x \vee \neg y$, we can see that the second law of De Morgan holds in ternary logic.

- $\alpha(x, y) = \neg x \vee y =$ *It is not raining or the sun is shining.*

  The truth table of this function is shown in Table 10 together with the truth table of the implication and we can verify over there, that $\alpha(x, y)$ and $\mathrm{Imp}(x, y)$ are not equivalent as it happens in binary logic.

- $\beta(x,\ y) = \neg y \to \neg x =$ *If the sun is not shining then it is not raining.*

  We can verify in Table 10 that it is equivalent to the implication as the binary case.

- $\delta(x,\ y) = (x \to y) \wedge (y \to x) =$ *If it is not raining then the sun is shining, and if the sun is shining then it is raining.*

  We can see in Table 11 that it is equivalent to the "equivalent" function.

Table 9. The De Morgan laws in ternary logic

x	y	$\neg x$	$\neg y$	$x \vee y$	$x \wedge y$	m_{11}	m_{12}	m_{21}	m_{22}
1	1	0	0	1	1	0	0	0	0
1	$\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	1	1	0	0	0	1	1
$\frac{1}{2}$	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	1	1
0	1	1	0	1	0	0	0	1	1
0	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	1	1
0	0	1	1	0	0	1	1	1	1

Table 10. $(x \to y) \leftrightarrow (\neg x \vee y)$ fails and $(\neg y \to \neg x) \leftrightarrow (x \to y)$ holds in ternary logic

x	y	$\neg y$	$\neg x$	β	α	imp(x, y)
1	1	0	0	1	1	1
1	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	0	1	0	0	0	0
$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	1	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	$\frac{1}{2}$	1

$\frac{1}{2}$	0	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
0	1	0	1	1	1	1
0	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	1
0	0	1	1	1	1	1

Table 11. The derivation of the equivalence law in ternary logic

x	y	$x \to y$	$y \to x$	$\delta(x, y)$	$\mathrm{Equiv}(x, y)$
1	1	1	1	1	1
1	$\frac{1}{2}$	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	1	0	0
$\frac{1}{2}$	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	1
$\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$
0	1	1	0	0	0
0	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
0	0	1	1	1	1

5. Conclusions

We have shown, at least, at this elementary level that there exist four main properties of ternary logic:

1. The ternary logic is one generalization of the binary(classic) case. Such generalization is in the sense of each proposition that is true under the rules of the binary logic will be true under the rules of the trivalent case. Analogously for the false propositions.

2. In ternary logic the construction of tautologies is more difficult than in binary logic. Worse for the construction of contradictions.

3. For the ternary logic, the implication $(x \to y)$ operation of two propositions

x, y cannot be derived from the basic operations $(\neg)$, $(\vee)$, and $(\wedge)$ as it happens in the binary case.

4. The ternary logic cannot have a Boolean algebra structure whereas the binary logic can have. The proof of this conclusion is given by the functions f_2 and f_{19} of Table 8.

References

[1] N. P. Brousentsov, Computing machine Setun of Moscow State University, New Developments on Computer Technology, Kiev, 1960, pp. 226-234.

[2] N. P. Brousentsov et al., Development of ternary computers at Moscow State University, www.computer-museum.ru/english/setun.htm, online accessed, 2008-05-29.

[3] R. Cignoli, I. D'Ottaviano and D. Mundici, Álgebra das lógicas de Łukasiewicz, Universidade Estadual de Campinas, UNICAMP, 1995.

[4] Judith L. Gersting, Mathematical Structures for Computer Science, 4th ed., W. H. Freeman, New York, 1998.

[5] Suresh Golconda, Genetic Algorithm for tic-tac-toe,
http://www.cacs.louisiana.edu/sxg3148/~files/Genetic_docu.doc.

[6] Ivan Guzmán de Rojas, Logical and Linguistic Problems of Social Communication with Aymara People, International Development Research Centre (IDRC), Ottawa, Canada, 1984.

[7] D. E. Knuth, The Art of Computer Programming, Vol. 2. Seminumerical Algorithms, Addison-Wesley, 1969.

[8] Jan Łukasiewicz, O Logice Trójwarkoscioewj. English translation: On Three-valued Logic, Selected Works by Jan Łukasiewicz, L. Borkowski, ed., North-Holland, Amsterdam, 1970, pp. 87-88.

[9] L. A. Zadeh, Fuzzy sets, Information and Control 8(3) (1965), 338-353.