*Open Journal of Discrete Mathematics*, **2013, 3, 21-24**
http://dx.doi.org/10.4236/ojdm.2013.31005 Published Online January 2013 (http://www.scirp.org/journal/ojdm)

# Formularless Logic Function

**Alexey Nikolayevitch Teterin**
Izhevsk, Russia
Email: ant2000ic@gmail.com

Received August 18, 2012; revised September 18, 2012; accepted September 28, 2012

## ABSTRACT

The concept of computability is defined more exactly and illustrated as an example of Boolean functions and cryptanalysis. To define a Boolean function is not necessary to record its formula. To do that the reduced (compact) description of values is determined in the truth table or in the statement of the problem. We obtain estimates of computation time, the volume of a compact descriptions and the range of variables under which it takes the value 0 or 1, depending polynomially on the number of arguments.

**Keywords:** Complexity of Computation (03D15); Models of Computation (68Q05); Analysis of Algorithms and Problem Complexity (68Q25); Computational Learning Theory (68Q32); Separability; Satisfiability; Fuzziness; Dynamical Systems; Computability

## 1. Introduction

The universal model of computation will be proposed in the article, which essence firstly is explained at first on the example of Boolean functions and the calculation of the range of variables values, under which it takes the value 0 or 1 (specifying the area satisfiability—SAS). Then conclusions about the possibility of the solution of the problem of cryptanalysis are done.

The values of a Boolean function can be considered as 0 or 1 with a convex neighborhood, painted in different colors (0—is white, 1—is black—two sets of convex disconnected components). Definition of a Boolean function can be reduced to the separation of values equal to 0, the values are equal to 1 in the Boolean *n*-dimensional space (this requires the construction of the table containing the $2^n$ rows or a logical formula which length is proportional to a polynomial $2^n$). We can assume that the Boolean space is a subset of real *n*-dimensional unit hypercube. In the research [1] we obtained polynomial appraisals for the separability of infinite bounded sets that are valid for our particular case too.

All necessary theorems were formulated in [2].

## 2. Boolean Functions

Boolean functions are given by the truth table on which the formula is built. Instead of the formula a compact description of the binary feature space (**Table 1**) is proposed to use. As in the case of the formulas, the truth table is unnecessary.

A formal proof of the theorem is based on estimates obtained in [1], and involves finding a Boolean function that separates the two classes of white and black points. Unlike the standard approach the ordering of space with the lexicographic order, or a binary tree are used.

The sequence of relationships can be defined by a binary tree, the number of coordinates and its value is situated in its nodes. The true value corresponds to one branch, while the false correspond to the other. The leaves have the function value (the number of class 0 or 1). This is so-called compact description (**Figure 1**) of the functions specified. Function identically equal to 0 or 1, determines the root of the tree.

In essence, a compact description of the function specifies a set of domains of different functions (similar to each other) in which they take the values 0 or 1.

**Table 1. The truth table on which the formula is built.**

| $x_1$ | $x_2$ | $x_3$ | $f$ |
|---|---|---|---|
| 00001111 | 00110011 | 01010101 | 11100000 |

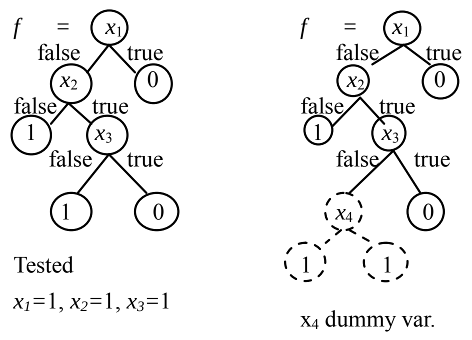

**Figure 1. Compact description.**

Similar to each other functions differ only in the truth table rows with identical values of the function, and they are described by a single tree (domain).

The advantage of a compact description is the lack of dummy variables. Dummy variable if the function does not depend on it, means the presence of the values 0, 0 or 1, 1 in the leaves of a node. You can remove this node to leaf 0 or 1, respectively, or vice versa add to perform logical operations and bring to a tree topology. Identical functions with dummy variables, defined in such tree, are included in a class of similar functions.

The value of one coordinate *n*-dimensional Boolean space divides the Boolean space or a truth table into two parts (each of them, if possible, is divided into two, etc.).

**Theorem 1**. *To define a class of similar Boolean functions of the n-arguments it is sufficient to determine the values of these functions on a sequence consisting of the* $\lceil n\sqrt{n}\rceil$ *argument values which are located in internal nodes (non-leaf nodes) of a binary tree.*

*Proof.* Any Boolean function is uniquely defined in the *n*-dimensional Boolean space with the corresponding coloring. For the separability of the two bounded sets in the real *n*-space the following maximum estimate of the number of partitions were obtained [1]:

$$-np\Big/\log_2\left(1-(1-\varepsilon)\delta_0\Big/\sqrt{n}\right)$$
$$\le n\sqrt{n}p\Big/\left(\delta_0(1-\varepsilon)\right) \le M \le n^2\Big/\left(\delta_0^2(1-\varepsilon)\right)$$

Separability is possible with a positive distance between two bounded sets A and B:

$$\delta_0 \overset{def}{=} \inf\left\{\|x-y\| : x\in A, y\in B\right\}$$

$0<\varepsilon<1$ is chosen arbitrarily. Changes in the $\varepsilon$ in the neighborhood of 0 have little effect on the number *M*. For a Boolean space with a finite set of elements we choose $\varepsilon \approx 0$.

$p \ll \sqrt{n}/\delta_0$ —the maximum number of surfaces for the separation of the two sets. The upper limit was obtained for a "overlapping sets" (water and sand does not exist in the Boolean space), with a value $p=\sqrt{n}/\delta_0$. For a Boolean space the value is $p=1$, so the maximum should be divided into $\sqrt{n}/\delta_0$, $\delta_0=1$. Finally, we obtain

$$-n\Big/\log_2\left(1-1\Big/\sqrt{n}\right) \le n\sqrt{n} \le M \le n\sqrt{n}$$

$$-n\Big/\log_2\left(1-1\Big/\sqrt{n}\right) \le M \le n\sqrt{n} \quad \text{QED}$$

**Assertion 1.** *To define a class of similar Boolean functions of the n-arguments is sufficient to determine the values of these functions on a full tree with a single relation, all n of these internal nodes, except the last, have a leaf.*

In support of this assertion the fact is evidenced that for *n* divisions of the truth table, we clearly identify a single line of it. The principle of dividing the feature space is maintained at each step of the algorithm of the second type, which is uniquely defined by at least one class (Theorem 3 [1,2]).

Any function supposes the free order of the arguments and is proposed by topology of the tree. All others keep some order of the arguments, and to work with such trees is not easy.

As the relationship, the truth that you want to check—the equality of the argument, the value is 1. Such tree can be leveled at the introduction of the branch with true relationship inequality to 1, which increases the number of used relations.

An alternative way of defining relationships—is equality or inequality of the arguments to each other. We can add or not add the ratio equals 0 or 1. In such case, the four relations are used. For example, the ratio $x_1=x_2=x_3=1$ can be rewritten in any option in the binary and/or unary, including this one $x_1=x_2, x_2=x_3$, $x_3=1$, which allows not to indicate the number of the first argument (determined by the place).

Logical formulas are written in the truth table. Before it appears, the problem must be determined on a content level. It is possible to bypass the step of creating a truth table and go to the task of the logical functions of the meaningful statement of problem. You can define as many coordinate relations as necessary in the statement of problem, in which the hierarchy may be present, and you can choose a tree that will fit it. The transition from one representation to another (rebuilding the tree) will not cause algorithmic difficulties.

The theorem and the assertion exist because of the truth of Theorem 3 in [1,2]. We choose the best result—an assertion 1.

The total length of the stored data to a Boolean function in bits (*L*) consists of values, the number of leafs of a binary tree $n+1$ (you cannot assume the last leaf, its value is opposite to the next leaf, if the tree has no dummy variables $-n$, the number of required number of coordinates (the length of the coordinate numbers—$\log_2 n$). $L \le n+\lceil n\log_2 n\rceil+1=\lceil n\log_2 2n\rceil+1$.

There is also the first and the third type of algorithms.

For the first type of algorithms:

$$M \ge \left(\delta_0\sqrt{\varepsilon(2-\varepsilon)}(1-\varepsilon)\right)^{1-n}(1-\varepsilon)$$

The function $f=\sqrt{\varepsilon(2-\varepsilon)}(1-\varepsilon)$ reaches a maximum $f=0.5$ at a value $\varepsilon=0.3$. Then to the Boolean function it is necessary to determine the values of the following number: $0.35\times 2^n \le M \le 0.5\times 2^n$.

We can expect that with the transition from finite to infinite sets the lower bound will be too high, as in the case with the assessment $\lceil n\sqrt{n}\rceil$.

The upper limit is determined by the principle of its operation in the Boolean space for the worst case. One of the coordinates is divided by half-and-half, for the remaining two halves the bisection of the other coordinate takes place, etc. (In accordance with this algorithm and the second approach, we can talk about the balanced tree with the depth *n*).

You can also define the upper limit for the third type of algorithms. After the division of space in $2^n$ the cell is uniquely determined the membership of each cube 0 or 1. But we keep a minimal description of the two. The worst case is obtained in case of equality, and the best one—for one cube $1 \le M \le 0.5 \times 2^n$.

In contrast to the algorithms of the second type, the first and third do not need to store the number of coordinates (determined by the place), so a series of relationships are defined by the ratio of the lexicographic order. The total length of the bit sequence is $-n \cdot M + 1$.

## 3. Area of the Satisfiability and the Computation Time

**Theorem 2**. *For one class of similar Boolean functions of the n arguments there is an area of satisfiability* (*SAS*) *or its negation*, *defined on nothing more than the* $O(n^2)$ *argument values.*

*Proof.* Formally, to solve the satisfiability problem it is necessary to index binary tree (back to the reduced table description), which contains $2n$ values, to define belonging for each index to a class 0 or 1, and calculate the number for each class. In the future we use the minimal description. Example: Suppose class 1 (0) has the smallest number of indices (we have defined its minimal description), then any point of the Boolean space, either gets into this area, and then the value will be equal to 1 (0) or does not get, and the value will be 0 (1). For each index, you must extend the definition of $n-1$ value. It becomes apparent total number of $O\left(n^2\right)$ QED.

In the area of the satisfiability there are two approaches to the above listed algorithms or without them.

In the first approach for each of the new values $n-1$, if possible, the first algorithm should be used to the already calculated $2n$ values (like the second, it allows us to solve optimization problems). And then the third algorithm determines a minimal description of the area of satisfiability or its negation. Its lack is using of the truth table as initial data. The advantage is the index is sorted, by using the algorithms of the first type—the computation time of the logical functions $O\left(\log_2 n\right)$.

The second approach does not have this lack, the tree, built on the meaningful statement of problem, and does not require the mentioned types of algorithms. We have a complete binary tree with *n* leaves and *n* internal nodes. They define the complexity of the problem while writing out the coordinates on all paths from the root to 0 or 1, depending on what is less. After determining a way, all non-listed coordinates are varying from 0 to 1. The complexity of the problem is $O\left(n^2\right)$ (the second proof of Theorem 2). After sorting such table computing time of logical functions is $-O\left(\log_2 n\right)$.

Without specifying the area of satisfiability the computation time of the logical function is $O(n)$.

## 4. Model of Computation

From the separability of bounded sets the obvious mathematical statement implies.

**Assertion [2, p. 1800]**. *Any finite many-valued real piecewise-continuous function* $f$ *can be approximated by a step function at least to accuracy* $\varepsilon$.

A more precise formulation describing the approximating function is given in [2].

Finite function is defined for a bounded area of space, and hence has a finite number of jump discontinuities, infinite discontinuities (only one of the two one-sided limits exists or is infinite), there it has finite values. It can be normalized to a unit hypercube, which allows not only to give a comparative estimate of the number of steps (for which the normalization is not important), but also $\varepsilon$ for different functions.

Thus, we know exactly *what* to do when solving any problem, namely, to build $f$ from the assertion, in the future simply $f$ separating the two classes, and the solutions and non-solutions, possibly consisting of a single point and we can estimate its complexity by the number of steps (or move it the next phase "how").

The question *how* to do that is being discussed [1]. Alternative approaches—[3,4]. In the first approach we prove the existence of three types of computational learning algorithms, one of which requires a memory polynomially depending on the dimension of space. The memory is filled once with a fixed number of operations, and then the time is also polynomially. All this proves the following theorem.

**The theorem of computability**. *Every* $f$ *is computable by a universal computing device*, *with a given accuracy* $\delta_0/2$ *in n-dimensional infinite bounded real space. The complexity of computation (memory and time*) *depends on its dimensions polynomially.*

This statement is well suited for human-robot (like a computer), analog computations (analog computers, quantum computers), analog signal processing, analog electronics, neural networks, control theory built on differential equations and continuous dynamical systems [5, 6].

The real space is not a limitation. An ordered alphabet can be viewed as a number line. The words in the alphabet—are the *n*-dimensional space. Moreover, we can calculate the distance to the character, not only on the real line, but also in the *n*-dimensional space for the written, spoken…

This approach effectively overrides the prior use of the term fuzziness, replacing it by the difference between the actual position and the boundary described in the memory (between objects or the object itself), which is continuously changing in time, on the real line or in a *n*-dimensional space.

For cryptosystems, after normalization of the space to the unit hypercube (interval), $\delta_0 = 2^{-K}$ where the $K$ is the length of the key (one-dimensional space).

The revised approach to the computability was illustrated by the logic functions and cryptosystems. Regarding the latter, the futility of any development of algorithms became clear.

## 5. Conclusions

1) a) It is no place for intuitions in more precise theorem of computability.

b) The process of finding a suitable algorithm is quite formal and can be performed automatically. Its ease of use for Boolean functions and cryptanalysis justify its use in other subject areas.

2) For Boolean functions is to choose the one of two variants.

a) With the area of satisfiability storing the value $O(n^2)$, with the computation time calculating the logical function $O(\log_2 n)$. SAS problem is trivial, since it is already solved.

b) Keeping only the binary tree with $2n$ nodes which is able to participate in logical operations. The computation time is $t \approx O(n)$. The problem of SAS is $t \approx O(n^2)$.

3) If you wish you can generate a truth table by compact description but it is not necessary.

4) Lukasiewicz three-valued logic does not change anything in the proposed theory, making the description more natural and understandable. For the cybernetic systems the four-value logic with the addition of uncertain value can be used. You can increase the *n*-value with the degree of similarity to 0, 1, indefinitely, or go to the real interval [0, 1] (blurring makes it possible to use a 5th point of conclusion).

5) Using the proposed theory, we can estimate the speed, acceleration, the degree of approximation to the solution, non-solution, or a state of uncertainty, and make the appropriate changes to the space features.

## REFERENCES

[1] A. N. Teterin, "Classification on Bounded Sets," *Pattern Recognition and Image Analysis*, Vol. 20, No. 4, 2010, pp. 564-572. doi:10.1134/S1054661810040188

[2] A. N. Teterin, "A Geometric Approach to Classification: New Model of the Operation of a Neuron," *Computational Mathematics and Mathematical Physics*, Vol. 32, No. 12, 1992, pp. 1797-1805.

[3] E. M. Gold, "Language Identification in the Limit," *Information and Control*, Vol. 10, No. 5, 1967, pp. 447-474. doi:10.1016/S0019-9958(67)91165-5

[4] M. Burgin and A. Klinger, "Experience, Generations, and Limits in Machine Learning," *Theoretical Computer Science*, Vol. 317, No. 1-3, 2004, pp. 71-91. doi:10.1016/j.tcs.2003.12.005

[5] P. Orponen, "A Survey of Continuous-Time Computation Theory," In: D.-Z. Du and K.-I. Ko, Eds., *Advances in Algorithms*, *Languages*, *and Complexity*, Springer, Berlin, 1997, pp. 209-224.

[6] C. Moore, "Recursion Theory on the Reals and Continuous-Time Computation," *Theoretical Computer Science*, Vol. 162, No. 1, 1996, pp. 23-44. doi:10.1016/0304-3975(95)00248-0