Universal Journal of Mathematics and Applications, 4 (1) (2021) 41-49
Research paper

UJMA

Universal Journal of Mathematics and Applications

Journal Homepage: www.dergipark.org.tr/en/pub/ujma
ISSN 2619-9653
DOI: https://doi.org/10.32323/ujma.738463

Boolean Hypercubes: The Origin of a Tagged Recursive Logic and the Limits of Artificial Intelligence†

Ramon Carbó-Dorca[1]

[1]*Institut de Química Computacional i Catalisi Universitat de Girona, Girona 17005, Catalonia, Spain*

†*This paper is dedicated to the memory of Professor Mario Bunge (1919-2020), who enlightened the mind of many, in such a manner that the bright beam of his thought will never fade away.*

Article Info

Keywords: *Fuzzy sets, Fuzzy logic, Tagged sets, Boolean tags, Logical tags, Boolean tagged sets, Boolean hypercubes, Concatenation of Boolean hypercubes, Natural number recursion, Mersenne numbers, Multivalued logic, Gödel like properties of Boolean sets, Artificial intelligence.*
2010 AMS: *03-11, 68T01.*
Received: *16 May 2020*
Accepted: *12 March 2021*
Available online: *23 March 2021*

Abstract

Boolean and logical hypercubes are discussed as providers of tags to logical object sets, transforming them into logical tagged sets, a generalization of fuzzy sets. The equivalence of Boolean and logical sets permits to consider natural tags as an equivalent basis of logical tagged sets. Boolean hypercube concatenation easily allows studying how Boolean information is transmitted. From there a Gödel-like behavior of Boolean hypercubes and thus of logical object sets can be unveiled. Later, it is discussed the iterative building of natural numbers, considering Mersenne numbers as upper bounds of this kind of recursive construction. From there information acquisition, recursive logic, and artificial intelligence are also examined.

1. Introduction

This paper can be considered as another application of the versatile structure of Boolean hypercubes, which has been steadily developing in this laboratory since the first paper on a generalized form of fuzzy sets [1]. Reference [2] can be thought of as an application example of Boolean vertex structures to molecular similarity and design. Since these two works had been published, many developments in mathematical structures definition and applications in chemistry, biology, and physics were produced, as a condensed list see references [3]- [11]. On the other hand, the final prospection of a recent Bunge's book [13], has motivated the background where the present study is inspired and built. Bunge presented in an appendix of the previous reference an extremely interesting prospect to formulate the structure of logic.
The present paper will be developed using Boolean hypercubes as a tool but following essentially Bunge's directives. In this sense, the following working organization will be used. First, the basic ideas about fuzzy sets will be detailed. Then, the concept of Boolean tagged sets will be established. Boolean hypercubes and their transcription into natural numbers will be described next. Concatenation of Boolean hypercubes and the possibility to design a recursive construction of natural numbers will follow. Next, Mersenne numbers will be shown that play a leading role in this recursive construction. In a new section, a simple distance definition will permit in the continuing development of the paper to classify the truth and falsehood contents of available logical tags described at some Boolean hypercube dimension level. The Gödel-like structure of Boolean hypercubes tags associated with their concatenation will be later discussed. Finally, an analysis of the limits of artificial intelligence will close this presentation.

2. Binary Functions Over Object Sets: Basic Fuzzy Sets

One can classically define in general a set containing some sort of objects, an object set Ω. Then according to the ideas of fuzzy set definition by Zadeh [14], see also for extended information on fuzzy sets the reference [15], one can also describe a function f, the so-called membership function, having as domain the elements of such an object set. When applying the function to any object belonging to Ω, one can suppose that if such a function yields a binary outcome, then the resulting value of the function codomain may be associated with a

***Email address and ORCID number*: ramoncarbodorca@gmail.com, 0000-0002-9219-0686, (R. Carbó-Dorca)**

Boolean pair of digits: $B=\{0,1\}$, which alternatively can be also taken as a logical pair: $L=\{F,T\}$, of false and true, values.
Initially, this scheme can be formally written as a classical set theory definition, providing information about some objects and some set, through:

$$\forall \varpi \notin \Omega \to f(\varpi)=0 \wedge \forall \omega \in \Omega \to f(\omega)=1. \tag{2.1}$$

So, one can use the Boolean digits $\mathbf{B}$ as the function codomain leading to the primary definition of a fuzzy set [1], [14] or substitute them with the logical values $\mathbf{L}$ and thus leading to a fuzzy logical set.
This previous fuzzy set definition is equivalent to use the results of the application of the membership function f over the objects' domain as the vertices of a one-dimensional Boolean hypercube: $\mathbf{H}_1=\{(0),(1)\}$, where the parenthesis means one can take the two binary digits as one-dimensional Boolean vectors, the two vertices of the one-dimensional Boolean hypercube.
To attach fuzziness to the membership function codomain, one might proceed one step further and suppose that some objects cannot be considered as sharply belonging to the set Ω or as clearly being in the outside of the object set, but in kind of no-man's-land between the outside and inside of the object set. In this case, the simplest way is to use as function codomain the elements of a two-dimensional Boolean hypercube as follows:

$$\mathbf{H}_2=\{(0,0),(0,1),(1,0),(1,1)\}=\{\langle \mathbf{h}_0|;\langle \mathbf{h}_1|;\langle \mathbf{h}_2|;\langle \mathbf{h}_3|\}. \tag{2.2}$$

A membership function outcome like the zero vertex: $\langle \mathbf{h}_0|$ might mean that the domain object does not belong to the set. However, any vertex of the pair: $\{\langle \mathbf{h}_1|;\langle \mathbf{h}_2|\}$ obtained as a membership function value within the codomain, will represent a fuzzy answer to the question about the object belongs or not to the object set. Finally, the outcome $\langle \mathbf{h}_3|$ can be interpreted as positively belonging to the object set. The whole number of possibilities can be described using as the object set a fuzzy set Φ instead defined as:

$$\begin{array}{l}\forall \omega \in \Phi : f(\omega)=\langle \mathbf{h}_3|,\\ \forall \varpi \notin \Phi : f(\varpi)=\langle \mathbf{h}_0|,\\ \exists \varphi : f(\varphi)=\langle \mathbf{h}_2| \to \varphi \tilde{\in} \Phi,\\ \exists \phi : f(\phi)=\langle \mathbf{h}_1| \to \phi \tilde{\notin} \Phi.\end{array}$$

where the symbols $\tilde{\in}$ and $\tilde{\notin}$ mean an incomplete knowledge of the objects: $\{\varphi;\phi\}$ about belonging or not to the set Φ.
The use of the four vertices of the two-dimensional Boolean hypercube in the equation 2.2 as membership function outcome codomain permits, besides the positive and negative belonging issues, to add two nuances to the fact that the objects are located or not into the set.
A simple fuzzy set Φ can be defined in this manner, constituting an example of the most basic one, a part of the monodimensional case defined in the equation 2.1, which can be defined based on Boolean hypercubes.

3. Basic Boolean Tagged Sets

Nevertheless, the possibility of defining a fuzzy logic structure along with an object set can be easily generalized. Precisely taking as start up the two-dimensional Boolean hypercube as a platform to generalize fuzzy logical sets. From this initial scaffold, then using the properties of Boolean hypercubes of higher dimensions it can be in general created a fuzzy logical framework of arbitrary dimensions.
At the same time, to avoid the fuzzy set idea of defining a membership function, whose variable values in its domain are objects of some set, one can be aware of the fact that the same idea of fuzziness attached to an object set might be also generalized, using instead of the membership function the definition of a tagged set, see for instance references [1], [2].
In a primitive way, similar to the previous fuzzy set description, to the objects of some set one can connect tags, which for this study can be essentially characterized as the elements of the Boolean hypercube as defined in the equation 2.2.
Suppose an object set Ω and the two-dimensional Boolean hypercube $\mathbf{H}_2$ as defined in the equation 2.2. A Cartesian product can be described between the elements of the object set and the two-dimensional Boolean hypercube vertices: $T=\Omega \times \mathbf{H}_2$, such that the elements of the tagged set $\mathbf{T}$ can be defined by the ordered pairs:

$$\forall t \in \mathbf{T} : t=(\omega;\langle h_I|) \leftarrow \omega \in \Omega \wedge \langle h_I| \in \mathbf{H}_2. \tag{3.1}$$

The elements of the tagged set $\mathbf{T}$ permit to consider that every element ω of the object set Ω can be associated with a tag $\{\langle h_I \| I=0,1,2,3\}$ which is one of the Boolean vertices belonging to $\mathbf{H}_2$. The elements of the object set can be identified with the vertices of the Boolean hypercube and thus classified into four kinds, coincident with the fuzzy set structure already discussed.
The object tags might be also replaced by the four natural numbers which coincide with the digital transcriptions of the four Boolean vertices of $\mathbf{H}_2$. In one way or another, the four tags will classify in the same manner the elements of the object set. Of course, instead of Boolean tags, constructed by the set $B=\{0;1\}$ of bits, when looking at the construction of a logic system, one can use the logical values contained in the set: $\mathbf{L}=\{F,T\}$. Then, four logical classes associated with the object set can be also easily described. That is, the logical equivalent of the two-dimensional Boolean hypercube in the equation 2.2 can be written like:

$$\mathbf{L}_2=\{(F,F);(F,T);(T,F);(T,T)\}=\{\langle \mathbf{I}_0|;\langle \mathbf{I}_1|;\langle \mathbf{I}_2|;\langle \mathbf{I}_3|\}$$

and every logical vertex is used as a logical tag.
From now on, the Boolean framework will be used as a collection of Boolean vertex sets, which can be subjected to a trivial transformation, leading to a logical framework simply corresponding to a substitution of the Boolean bits: (0)bit–(1)bit, by logical (F)alse-(T)rue values.

4. Boolean Hypercubes

Because everything from now on will be based on the Boolean hypercubes structure, the building of such mathematical elements will be succinctly discussed next.
Any $N-$ dimensional Boolean hypercube $\mathbf{H}_N$ possess 2^N vertices, which can be expressed as binary row vectors or Boolean strings. [1]

$$\begin{array}{l}\mathbf{H}_N = \{\langle \mathbf{h}_I \| I = 0, 2^N - 1\} \\ \rightarrow \langle \mathbf{h}_I | = (\beta_{I,N}; \beta_{I,N-1}; ...\beta_{I,1}) \leftarrow \forall J = 1, N : \beta_{I,J} \in \{0,1\}.\end{array} \tag{4.1}$$

This notation obeys the fact that the binary strings $\langle \mathbf{h}_I|$ are supposed here to bear the most significant bits on the leftmost side.

5. Transcription of a Boolean Hypercube into a Natural Number Sequence

Every Boolean vertex corresponds to a decimal number within a well-defined natural number sequence. Constituting a relationship which can be defined as:

$$\forall I = 0, (2^N - 1) : \delta(\langle \mathbf{h}_I|) = \sum_{J=1}^{N} \beta_{IJ} 2^{(J-1)} = \upsilon_I \in \mathbf{S}_N \subset \mathbb{N} \tag{5.1}$$

where using as variables the Boolean hypercube vertices, then the multivariate Boolean function $\delta(\langle \mathbf{h}_I|) \in \mathbb{N}$ yields a natural number, including zero.
Thus, the natural number set $\mathbf{S}_N$ in equation 5.1 contains a sequence of 2^N natural numbers:

$$\mathbf{S}_N = \{0, 1, 2, ..., (2^N - 1)\} \tag{5.2}$$

where the last value corresponds to the so-called Mersenne number, which can be represented by:

$$\mu_2(N) = 2^N - 1.$$

6. Equivalence of Boolean and Logical Hypercube Vertices with a Natural Number Sequence

One can see without a problem that the elements of the natural sequence $\mathbf{S}_N$, described in the equation 5.2, can be used as a set of tags as such, accompanying any object set element. While the tagged set in the equation 3.1 can be described as a Boolean tagged set, substituting the Boolean hypercube vertices by elements of the natural number sequence of equation 5.2 results in a natural tagged set.
Natural tagged sets are even more identifiable with fuzzy sets because one can easily define the connection between the values of some membership function as elements of some natural number sequence, like $\mathbf{S}_N$, which can instead be employed as tags in an equivalent tagged set.
However, the connection of Boolean-logical hypercubes and natural numbers can be seen as an iterative way to build the hypercube vertices and the equivalent natural elements, but subject to the binary increase of elements both in $\mathbf{H}_N$ and in $\mathbf{S}_N$.
The natural elements of the sequence $\mathbf{S}_N$ are equivalent to the vertices of some Boolean hypercube $\mathbf{H}_N$. It has also been seen that these Boolean vertices can be transformed into logical ones. Thus, all the possible Boolean tags are transformed into the vertices of a logical hypercube: $\mathbf{L}_N$ and those into a natural number set: $\mathbf{S}_N$.
Next, one can admit that natural sequences, Boolean and logical hypercubes are the same structure described in three equivalent forms and therefore interchangeable.
In this manner the natural number sequence $\mathbf{S}_N$ in equation 5.2 might be well used as a set of tags instead of a set of logical hypercube vertices. Fuzzy logical sets and natural tagged sets are the same mathematical construct.

7. Specific and Complementary Vertices of a Boolean Hypercube

To deepen in the previously commented equivalence of the three possible kinds of tagged sets, the specific nature of some Boolean vertices will be presented now.
In any Boolean hypercube there are two well-defined vertices: The zero vertex:

$$\langle \mathbf{h}_0| \equiv \langle \mathbf{0}| = (0, 0, 0, ..., 0)$$

and the unity or Mersenne vertex:

$$\langle \mathbf{h}_{(2^N-1)}| \equiv \langle \mathbf{1}_N| = (1, 1, 1, ..., 1)$$

which for any dimension the unity vector corresponds to the binary representation of the corresponding so-called Mersenne number:

$$\delta(\langle \mathbf{1}_N|) = \mu_2(N).$$

Another well-defined set of vertices corresponds to a set containing N vertices, which can be called the canonical basis set. Such a specific Boolean vertex set may be constructed associated with the vertices, which starting from the zero vector, then one 0-bit at a time is substituted by a 1-bit only. The canonical basis set is made by the Boolean vertices, which also correspond to the decimal sequence of the powers of two:

$$\mathbf{K}_N = \{\langle \mathbf{h}_{2^0}|; \langle \mathbf{h}_{2^1}|; \langle \mathbf{h}_{2^2}|; ...; \langle \mathbf{h}_{2^{N-1}}|\} \Rightarrow \delta(\mathbf{K}_N) = \{2^0; 2^1; 2^2; ...; 2^{N-1}\}.$$

[1] In the present paper, the Dirac notation is used to represent vectors. Thus, an $N-$dimensional row vector is written as: $\langle \mathbf{a}| = (a_1; a_2; ,, ,; a_N)$ and its transpose which corresponds to a column vector is represented as: $|\mathbf{a}\rangle$.

The canonical basis set vertices can also be expressed with the usual symbols employed to describe the canonical basis set of any vector space:

$$\mathbf{E}_N = \{\langle \mathbf{e}_1|;\langle \mathbf{e}_2|;\langle \mathbf{e}_3|;...;\langle \mathbf{e}_N|\} \rightarrow$$
$$\{|\mathbf{e}_1\rangle;|\mathbf{e}_2\rangle;|\mathbf{e}_3\rangle;...;|\mathbf{e}_N\rangle\} \equiv \mathbf{I}_N = \{I_{IJ} = \delta_{IJ}|I,J = 1,N\}$$

which when considering them as columns and their zero-one elements as natural numbers, the final structure is the $(N \times N)$ unit matrix $\mathbf{I}_N = \{I_{IJ} = \delta_{IJ}\}$, being δ_{IJ} a Kronecker delta. However, turning again into the Boolean meaning side provides 1-bits in the whole diagonal and 0-bits elsewhere. The unity vertex is the complete sum of the canonical basis vertices:

$$\langle \mathbf{1}| = \sum_{I=1}^{N} \langle \mathbf{e}_I| \rightarrow \mu_2(N) = \sum_{I=0}^{N-1} 2^I = 2^N - 1.$$

Every Boolean hypercube vertex has a complementary vertex, where the 0-bits are substituted by 1-bits and vice versa, every 1-bit is substituted by a 0-bit. In particular, a conversion of this kind occurs when the zero and unity vector both are considered as complementary vertices.
The canonical basis vectors possess a complementary basis set made by substitution one of a time of the 1-bit of the unity vector by a 0-bit. As the canonical basis set can be arranged to construct the unit matrix, the complementary canonical basis set can be arranged into a matrix $\mathbf{C}_N$, defined with the aid of the unity matrix: $1_N = \{1_{IJ} = 1\}$, a matrix with all the elements equal to the 1-bit, minus the unit matrix, that is: $\mathbf{C}_N = \mathbf{1}_N - \mathbf{I}_N = \{C_{IJ} = \delta(I \neq J)\}$, where $\delta(I \neq J)$ is a logical Kronecker delta [2], which in this case yields 0-bits at the diagonal and 1-bits at every one of the off-diagonal elements.

8. Concatenation of Boolean Hypercubes

Moreover, Boolean hypercubes can be concatenated [9]. The symbol $\cup$ will be employed to signal the operation of vertex concatenation. For instance, one can simply write the concatenation:

$$\mathbf{H}_2 = \mathbf{H}_1 \cup \mathbf{H}_1 \rightarrow \mathbf{H}_2 = \{(0 \quad 0),(0 \quad 1),(1 \quad 0),(1 \quad 1)\} \tag{8.1}$$

which can be interpreted as concatenating the vertices of the right Boolean hypercube by each vertex of the one on the left. If necessary, one can construct higher dimensional Boolean hypercubes, see reference [9] for more details, just concatenating two or more Boolean hypercubes of lower-dimensional structures, for instance:

$$\mathbf{H}_{M+N} = \mathbf{H}_M \cup \mathbf{H}_N.$$

Following this hypercube building up, concatenation of the mono-dimensional Boolean hypercube $\mathbf{H}_1$ with any arbitrary $N-$ dimensional one: $\mathbf{H}_N$ produces:

$$\mathbf{H}_{N+1} = \mathbf{H}_1 \cup \mathbf{H}_N \tag{8.2}$$

meaning that the resultant Boolean hypercube $\mathbf{H}_{N+1}$ possesses a double number of vertices than the original Boolean hypercube $\mathbf{H}_N$. This is so because with the concatenation 8.2 the original Boolean hypercube vertices, as defined in the equation 5.1, transform into two kinds of vertices belonging to $\mathbf{H}_{N+1}$:

$$\begin{array}{lll} \forall I & = & 0,2^N : \langle \mathbf{h}_{N;I}| \in \mathbf{H}_N : \\ & \rightarrow & (0) \cup \langle \mathbf{h}_{N;I}| = \langle \mathbf{h}_{(N+1);I}| = \langle 0;\langle \mathbf{h}_{N;I}|| \in \mathbf{A}_{N+1} \\ & \rightarrow & (1) \cup \langle \mathbf{h}_{N;I}| = \langle \mathbf{h}_{(N+1);(2^N+I)}| = \langle 1;\langle \mathbf{h}_{N;I}|| \in \mathbf{B}_{N+1} \end{array}$$

with the additional property, involving the two newly defined sets:

$$\mathbf{H}_{N+1} = \mathbf{A}_{N+1} \cup \mathbf{B}_{N+1} \wedge \mathbf{A}_{N+1} \cap \mathbf{B}_{N+1} = \varnothing.$$

Thus, in such a concatenation operation two new Boolean vertex sets are well-defined. The set $\mathbf{A}_{N+1}$ is equivalently made by the old Boolean vertices of $\mathbf{H}_N$, considering that they are one dimension larger, but possessing the most significative bit as the 0-bit; while the set $\mathbf{B}_{N+1}$ contains all the new vertices which conform to the Boolean hypercube $\mathbf{A}_{N+1}$ and cannot belong to the initial $\mathbf{H}_N$.

9. Recursive Construction of Boolean Hypercubes and Natural Numbers

As explained above, this construction indicates that starting from any Boolean hypercube one can obtain another one with the dimension augmented in one unit. An in a deep application can be found in reference [10].
Half of the vertices in $\mathbf{H}_{N+1}$ are those of $\mathbf{H}_N$, because whenever one writes the decimal representations of the involved vertices it is obtained: $\delta(\mathbf{H}_N) = \mathbf{S}_N$, but also it can be written: $\delta(\mathbf{A}_{N+1}) = \mathbf{S}_N$. The Boolean vertex set $\mathbf{B}_{N+1}$ corresponds to a decimal representation which corresponds to the very new nature of the extended dimension and can be easily written as:

$$\delta(\mathbf{B}_{N+1}) = 2^N \oplus \mathbf{S}_N = \{2^N;(2^N+1);(2^N+2);...;(2^{N+1}-1)\} \tag{9.1}$$

meaning that the power of two constant 2^N is summed to every element of the natural set $\mathbf{S}_N$. The final element of this sequence 9.1 is the Mersenne number $\mu_2(N+1)$. Accordingly, the natural number set associated with $\mathbf{H}_{N+1}$ can be expressed as:

$$\mathbf{S}_{N+1} = \delta(\mathbf{A}_{N+1}) \cup \delta(\mathbf{B}_{N+1}) = \mathbf{S}_N \cup (2^N \oplus \mathbf{S}_N). \tag{9.2}$$

[2] A logical Kronecker delta is a symbol $\delta(E)$ where E is an expression, which if $E = .True.$ returns $\delta(T) = 1$ and if $E = .False.$, then $\delta(F) = 0$.

A result meaning that natural numbers can be constructed within a recursive framework, associated in turn with the building of Boolean hypercubes via an iterative concatenation as defined in the equation 8.2.
Moreover, in the equation, 9.2 the structure of the recursion involving the decimal representation of Boolean hypercubes suggests one can define a recursion operator R, such that:

$$\mathbf{S}_{N+1} = R[\mathbf{S}_N] = \mathbf{S}_N \cup (2^N \oplus \mathbf{S}_N).$$

Consequently, fuzzy logical arbitrarily complex structures can be iteratively constructed in the same way.

9.1. Mersenne Numbers and Their Twins

According to the equations 9.1 and 9.2, the lower and upper limits of the new natural numbers attached to a Boolean hypercube digital transcription, are respectively the Mersenne number and a power of 2, both associated with the appropriate dimension of the hypercube.
Furthermore, it is interesting to note the role of the Mersenne number: $\mu_2(N) = 2^N - 1$, contained in a natural number sequence similar to the equation 5.2, and another number contained as a second element of the sequence of the equation 9.1 $\upsilon_2(N) = 2^N + 1$, which is a member of a general sequence, which can be called Mersenne twins [10]. While any Mersenne number is translated into binary form as a 1-bit string of dimension N, a unity vector: $\langle \mathbf{1}_N| = (1,1,1,...,1,1)$, the Mersenne twins are well-defined bit strings too, being nearly a zero vector but having two 1-bits, one at the most significative left bit and the other at the rightmost position:$\langle \mathbf{v}_{N+1}| = (1,0,0,...,0,1)$. Between any Mersenne number and his twin there is the vertex corresponding to the power 2^N , which obviously enough corresponds to an elements of the canonical basis $\langle \mathbf{e}_N|$.

9.2. Complex Logical Object Sets

Coming back to a logical object set Ω, one can choose some function, which applied on an object yields a Boolean hypercube vertex, that is:

$$\forall \omega \in \Omega : f(\omega) = \langle \mathbf{h}_I| \in \mathbf{H}_N$$

or similarly as done in the equation 3.1, one can associate to each object belonging to the set Ω a Boolean hypercube vertex, forming a Boolean tagged set. That is, first forming a tagged Set using the vertices of a Boolean hypercube as tags:

$$\mathbf{T_H} = \Omega \times \mathbf{H}_N \rightarrow \forall t \in \mathbf{T_H} : t = (\omega; \langle \mathbf{h}_I|) \Leftarrow \omega \in \Omega \wedge \langle \mathbf{h}_I| \in \mathbf{H}_N. \tag{9.3}$$

Alternatively, one can use as tags the elements of the decimal transcription $\mathbf{S}_N$ of a given Boolean hypercube $\mathbf{H}_N$:

$$\mathbf{T_S} = \Omega \times \mathbf{S}_N \rightarrow \forall t \in \mathbf{T_S} : t = (\omega; s_I) \Leftarrow \omega \in \Omega \wedge s_I \in \mathbf{S}_N = \delta(\mathbf{H}_N). \tag{9.4}$$

In equations 9.3 and 9.4, it is supposed that $\mathbf{H}_N$ and what is the same, the natural transcription $\mathbf{S}_N$, contain sufficient information on the elements of Ω, which in turn can be considered in particular as logical objects, to leave almost all sensible knowledge covered up to some extent and experience, whose measure can be easily associated to the dimension N. Of course both Boolean hypercubes and the decimal transcript forming natural number sequences are equivalent to the logical hypercubes $\mathbf{L}_N$, whose vertices are made by logical elements. One can write the implication of interchange among the three descriptions of the same concept:

$$\mathbf{L}_N \Leftrightarrow \mathbf{H}_N \Leftrightarrow \mathbf{S}_N.$$

10. The Distance of a Boolean Hypercube Vertex from the $\langle \mathbf{0}|$ and $\langle \mathbf{1}|$ Vectors

While the extreme vectors $\langle \mathbf{0}|$ and $\langle \mathbf{1}|$ represent the most far away vertices of any Boolean hypercube $\mathbf{H}_N$, representing the natural numbers 0 and $(2^N - 1)$ respectively, the rest of Boolean-logical vertices can be also located concerning these extremal vertices belonging to any hypercube.
Boolean vertices can be classified in a complementary manner by the number of 0-bits or 1-bits they have, but this just describes some trivial classification of Boolean vertices. There are in a Boolean hypercube $\mathbf{H}_N$, $(N+1)$ vertex classes, which also can be complementary to another set of classes with the same number of elements. Depending on that, each class corresponds to having: $\{0,1,2,...,(N-1),N\}$ 1-bits, or in a complementary way, the vertex classes might be contemplated as possessing: $\{N,(N-1),...,2,1,0\}$ 0-bits.
Although there are several ways to measure the similarity between two-bit strings, see for instance references [17]- [22] as an assorted example, the distance of a precisely given Boolean vertex say $\langle \mathbf{v}_I|$ to both extreme vectors can be also used. Such a measure might be obtained first via its decimal transcription: $\delta[\langle \mathbf{v}_I|] = d_I$.
Once obtained the decimal transcription of a Boolean vertex, then two Minkowski-like distances: $D_0\big|\langle \mathbf{v}_I| - \langle \mathbf{0}|\big|$ and $D_1\big|\langle \mathbf{v}_I| - \langle \mathbf{1}|\big|$ can be constructed, taking directly the decimal transcription d_I for the first distance to the zero vertex, and using the absolute difference: $|d_I - (2^N - 1)| \equiv |d_I + 1 - 2^N| \equiv |2^N - (d_I + 1)|$ for the second one to the unity vertex.
Thus, every vertex could be associated with a pair of natural numbers: $(D_0; D_1)$, which will locate it nearest one or another of the extremal vertices or make it equidistant to both.
For example, one could be interested to locate from the zero and unity vertices respectively, in the way explained above, the 5-dimensional Boolean hypercube vertex: $\langle \mathbf{h}| = (1,0,1,0,1) \rightarrow \delta[\langle \mathbf{h}|] = 21$. Thus, the distance to the zero vector is just 21, while the distance to the Mersenne vertex, which in this case has a decimal value: $\mu_2(5) = 2^5 - 1 = 31$ is 10. One can conclude that this specific vertex example is nearby to the Mersenne vertex than to the zero vertex. Better, one can represent this situation with a two-dimensional vector, in this specific case: $(21,10)$. From the logical point of view, if the zero vertex represents absolute falsehood and the unity vertex absolute trueness, the vertex of the above example can be classified as nearer truth than from untruth.

10.1. Double distances and the falsehood-trueness content of logical objects

Such double distance classification, providing for every vertex a two-dimensional non-negative definite coordinate element, as explained above, might be interesting when using Boolean hypercube vertices as sequences of false and true associations, because even if every vertex will correspond to a non-trivial sequence of false and true elements, the double distances to the extremal hypercube vertices can locate an associated complicated logical object as being lying next to complete falsehood, represented by the zero vertex $\langle \mathbf{0}|$, or sitting nearby to the absolute truth, represented by the unity vector $\langle \mathbf{1}|$.
Another short example will provide the already mentioned possible use of the double distance to associate Boolean hypercube vertices to the extremal vertices for logical purposes. In the equation 8.1, the vertices of a two-dimensional Boolean hypercube are given. The natural number sequence associated with every one of these vertices is: $\mathbf{S}_2 = \{0,1,2,3\}$. Then the two middle vertices double distances are simply obtained as: $(1,2)$ and $(2,1)$ respectively. The meaning of this resultant pattern corresponds to associate the vertex $(0,1)$ nearby to the zero vector, while the vertex $(1,0)$, which is complementary to the former one, is located closer to the Mersenne vertex.
The above simple case, connected with the two-dimensional Boolean hypercube of the equation 8.1, can be also seen from a multidimensional logical point of view and to the three-valued logic system [23]. After substituting the 0-bit by a logical $F-$value and the 1-bit by a logical $T-$value, then immediately one can consider that the two vertices (F,T) and (T,F) can be classified as undecidable logical objects, from the point of view of absolute falsity-trueness values. However, using the defined double distance technique, the first vertex can be thought located nearby to the absolute falsehood vector (F,F), while the second one, could be seen lying nearest to the absolute truth vertex (T,T). This interpretation means that even undecidable logical objects, corresponding to a fuzzy or tagged logical frame, might be studied possessing extra logical nuances, allowing them to consider holding truth and false contents in different amounts when compared with respect Aristotelian complete falsehood and trueness vertices.

11. Gödel-Like Properties of Boolean Tagged Sets

The recursive structure of the natural numbers as shown in the previous pages, essentially in section 9, permit us to observe how every time a Boolean hypercube concatenates with a mono-dimensional one, then, as a result, the number of vertices or elements in the decimal transcription, is duplicated. What is the same, the tag set construct duplicates itself its elements every time a simple concatenation of this kind is performed, as it is also shown in the equation 9.2.
That is: one-bit concatenation is enough to duplicate the amount of information, which one can suppose is ready to be attached as Boolean or logical vertices or natural numbers tags to the elements of an object set. In case this extra information is not needed; even so, the new tags, which can be generated in this one-bit way, can be used to enrich the available information with minimal effort.
However, if the new extended tags are used, then the possibility appears to create this augmented information once again and thus new extended tags can be added to the initial tag set.
No end could be envisaged to stop this iterative process, which will reappear along every time one might need more information to be added as Boolean or logical vertex tags to an object set. Such characteristics shared by both Boolean and logical basis sets and the attached natural numbers can be taken as a Gödel-like theorem [34] property, a characteristic which was already discussed in reference [9]. With this, it is meant the intrinsic incompleteness and unprovability of any logical structure which can be based on a Boolean or logical basic description. One can advance that in general almost all, if not all, human knowledge might be translated into a Boolean-logical framework. Therefore, any human knowledge construct is incomplete and unprovable.
Such a recursive procedure illustrates the possibility to assemble a simple path leading to an increase in the information knowledge about a known object set, and the impossibility that any (in the present case: logical) theory becomes complete. A simple 1-bit reconstructs a completely new set of logical information structures, equal in number to the initial ones, which are also preserved with the addition of a complementary single 0-bit.
Describing this issue again in short: a unique 1-bit concatenated to any initial set of Boolean-logical tags can completely transform the tagged set information contents, while the initial original tags are preserved via a 0-bit concatenation.
Therefore, no Boolean-logical information about some object set can be considered final and complete, but always lying one step back, until the use of a 1-bit concatenation changes everything, doubling the potentially available information.
Of course, the same incompleteness occurs in the set of decimal transcription of a given Boolean-logical hypercube. That is, in the information duplication: $\mathbf{H}_N \to \mathbf{H}_{N+1}$, there is also a duplication of the sequence of the natural number transcription, where: $\mathbf{S}_N \to \mathbf{S}_{N+1}$ implies that: $\mathbf{S}_{N+1} = \mathbf{S}_N \cup 2^N \oplus \mathbf{S}_N$. Meaning, as before, that to every element of $\mathbf{S}_N$ the power: 2^N is added. Evidently, because of this information doubling, a similar expansion appears associated with the tags or fuzzy logical values in logical object sets.

12. The Limits of Artificial Intelligence

The issue of fuzzy Boolean tagged sets might not only be associated with a Gödel-like impossibility to get rid of incompleteness in any kind of theoretical or experimental structure, whenever such structure can be transformed into bit strings. It has immediate consequences in handling Artificial Intelligence (AI) devices (wherein this denomination also machine learning procedures are included to simplify the reasoning of the present discussion) constructed for any purpose and at any complexity level, the development of logical systems included. Something similar has been commented on machine learning not long ago [30].
What it seems not commented at all, as far as the present author knows, is the nature of any AI procedure, by extension of any computational procedure, which is or can be running in some computer, from small desktop to supercomputers or clusters of them. They undoubtedly run as programs written in any available programming language.
Any application program constructed for any purpose is, sorry for the redundancy, programmed by one or several programmers. Because of this, a program cannot perform anything else *outside* of the programmed code provided by the programmer(s). That is: not included by the programmer(s) within the code. This is not the *only* limit of any given computing procedure. In old and modern computers results are in need to be reproducible, and the digital machines are thus lacking the freedom to provide different results than these expected to be yielded by a given program.

That is, every time that the same input is fed up *with* a program, no matter how complex can be, the *same* output will be obtained. In this sense, programs running on digital computers are predictable. Certainly, one can simulate randomness through adequate programming techniques, but these techniques need a random seed, which repeats the pseudo-random series every time the same seed is introduced as input. In brief, computers' lack of freedom of will, is *deterministic*.

No one can predict if the announced quantum computers will continue to be fully deterministic or for some purposes can be programmed to act with some output freedom of choice. Certainly, for engineering purposes, say, even quantum computers must behave deterministically, otherwise, it will become impossible to compute any object with reasonable confidence.

However, with some easy thought, one can describe fuzzy Boolean hypercubes, which can possess even the ability to change the information content, depending on some parameter(s) which can act as a time-like dependence or evolution, see reference [35]. That is, with the aid of this kind of hypercubes not only the fuzziness of the involved (pseudo-)Boolean tags, which can be chosen as the hypercube vertices, can be present and attached to any kind of logical objects, but one can even envisage their time-like evolution depending on ad hoc parameter(s).

Such previous statements appear to be obvious if one analyzes the essence of an AI procedure. One can consider that they are based on a complex mathematical scheme, built on computational programming grounds, and previously trained by a finite object subset to become operative. An AI procedure is hopefully trained, so it achieves the stage of being able to recognize and handle new objects, which have been not previously used in the training or learning period stage.

It seems that the forest of methods where AI is grounded; among them logic, statistical learning, and classifiers, image recognition, ... could also be essentially based on a large variety of the so-called Neural Networks, see for an excellent resume reference [24].

A brief description example of the use and characteristics of AI within the neural network context will be done within the interesting background of medicinal chemistry. The use of neural networks in drug design is quite old, see for instance reference [25]. However, modern AI users in this field can be quite preposterous sometimes, offering publications that intend to astonish the naive readers. A good example of the idea of obtaining exhaustivity within the field of drug design can be well represented by the article of reference [26]. There, the authors describe an AI way to obtain molecular information.

Pretending to have produced a sample of all molecular structures bearing potential biological activity [3]. In the not very far years there is a large amount of paper, which in one way or another use AI to the aim of molecular design, it is too short space to describe and quote all of them. However, some examples are worth comment. Another at least not so pretending set of studies on the same AI application field was recently published [27], this time various authors were offering a more reasonable point of view to the public, though. But a preposterous and overwhelming very recent contribution, precludes a not particularly good perspective in the field of AI application to molecular engineering [28] via theoretically dubious docking techniques. The same can be told about another pair AI recently published papers, which claim to have solved old chemical problems. The first describes a solution of the protein folding problem (see the web site [30] for more information and references), while the basic background relies on a classical parametric structure of the computations. The second claims to provide a black box Schrödinger equation solution [31] of any quantum chemical problem but based on a Monte Carlo procedure to solve the wave function equation, but using simple ideas to construct the wave functions, as atomic centered basis sets. One can wonder, among several technical questions if a pseudo random procedure will appear reliable enough for large molecular systems.

This kind and other multiple applications of AI algorithms, in case one agrees that these processes (usually acting as black boxes) can be called in this way, try to simulate as much as possible a brain activity knowledge process. Similarly, as brain most of the development does, these procedures need a training initialization, a specific information gathering as a starting point for every purpose of AI deployment.

The resultant computational trained structure, according to the AI users, must be hopefully able to build new information concerning objects not used in the training period.

Now, one must be aware that in the present discussion the AI structure sits into a computer in the form of a program. Thus, a discussion about the limits of AI must consider such a program background, and the boundaries of such a situation are to be linked to the proper AI limits.

These points above are coupled to the fact that new information is generated from the initial one, which has been provided during the training step. Then, additionally, some important questions arise.

For instance, how pretended AI efficiency can happen? Whenever the AI system, must consist of a program written by some operator, and thus cannot generate new information within the program itself if left intact. Unless such an issue is provided from an external extra training step or by a program modifying source: included in the program or feed on it in some moment of the analysis?

Is any new source of information pre-programmed and based on the knowledge provided by every new object feed to the procedure, analyzed by the trained algorithm, and finally used to modify the initial information? If so, the AI will act on every new analysis as being in another stage of the training step. Therefore, entering a never-ending loop, taking for granted the intrinsic Gödel-like incompleteness of any information system [30].

Alternatively, the AI answer on a new object analysis is just given by an as sophisticated as one can imagine kind of interpolation (or extrapolation, which constitutes a shady workout, prone to failure) among the information of the training objects used?

After these preliminary discussions on the nature of computing and AI, one can also consider now the problem of causality, which is dubious that any AI-based procedure can include, at least with the actual level of programmed AI systems. Nobody can say how these issues will evolve with a new breed of quantum computers, which even might be able (difficult to think how this be reached, though) to escape in some circumstances the deterministic behavior of digital computers. To peruse an excellent discussion on the nature of the important causality topic by Bunge, reference [33] is recommended.

The acausal character of AI appears to be another issue and can be stored on the same shelf where statistical correlation remains. It is well-known that such a statistical relationship does not imply a causal connection between the variables involved. No doubt about this, because there is a shared set of similarities between statistical correlation and AI. Both might use a training set, and both provide acausal relationships on new objects and some parameters, both present Gödel-like incompleteness.

Perhaps the AI acausal background is a consequence of the fact that the information content of the related procedures also bears the paradox of apparently creating new information, starting from an a priori established set of computational structures. This formal appearance is certainly shared by AI with the well-known set of statistical correlation procedures.

Such a problem is not the only one that can be found in AI procedures. As it has been previously commented, unprovability is another one,

[3] The title of the quoted paper might be taken as representative of the authors' attitude: "Stochastic Voyages into Unchartered Chemical Space Produce a Representative Library of All Possible Drug-Like Compounds". One can say that, not only the apparently complete advertised results seem a bit preposterous, considering of the Gödel-like properties of information manipulation, but even the wording at the beginning of the title statement has been duly criticized some years ago [28].

see reference [30] to have some information about this problem too, which can be linked to one of Gödel's theorems [34], and thus also to the discussion contained in the previous section.
Even more, another similar problem that also can be found within AI procedures will be discussed now. When setting up initial information, any AI might be considered that produces the equivalent of some bit string, provided that any information or logical string is constructed, bearing a dimension, say, as large as the nature of the problem requires.
Such an initial setup cannot provide new appearances of the same kind of bit strings showing out of nowhere, except information already contained into the original bit string appears. What could be conceivable to consider is the possibility to obtain Boolean vertex transformations belonging to the Boolean-logical hypercube containing the initial bit string. This might be allowed as a posterior step of information manipulation because, given a bit string of dimension N, the remaining $2^N - 1$ vertex components of the hypercube holding them are well-known and can be easily chosen as vertices of some Boolean hypercube $\mathbf{H}_N$ for any purposes.
That is, such information changes shall be found within the set of the initial well-defined dimension of the Boolean-logical bit structures, where belongs to the gathered training information.
To better visualize this situation, suppose that the AI training set has been able to construct information contained into some $N-$dimensional Boolean hypercube: $\mathbf{H}_N$. In principle, no limit must hinder the dimension N. Thus, theoretically one can suppose that the information generated after the training period might be contained in some Boolean vertex:$\langle \mathbf{h}_I| \in \mathbf{H}_N$.
What one can easily grasp is that without adding at all even a new 1-bit of information to the trained system represented by $\langle \mathbf{h}_I|$, any answers the AI procedure can provide are compelled to be located among the Mersenne number $\mu_2(N)$ of the remaining vertices of the original Boolean-logical hypercube, where the Boolean vertex representing the trained AI belongs.
It has already been previously discussed how by addition of one bit, the amount of information of any Boolean hypercube duplicates. If one bit is added to the initial Boolean-logical AI structure, then the training period might be invalidated or transformed into another completely different result.
Therefore, no new information content must be created by an AI post-training manipulation out of the vertices belonging to $\mathbf{H}_N$. The AI resulting outcome must be just the consequence of a shift from a Boolean-logical vertex to another, that is: $\langle \mathbf{h}_I| \to \langle \mathbf{h}_J|$ or any other manipulation among the known vertices of $\mathbf{H}_N$.
Such a program could be modified if and only if, with the analysis of a new object, the AI answer and the information of the newly tested object could be absorbed into the old pool of training objects. Then, the resulting algorithm issued from the training set period results in being necessarily modified. Subsequently, one could face the transformation of the original Boolean-logical hypercube $\mathbf{H}_N$ into another one of larger dimensions, say: $\mathbf{H}_N \to \mathbf{H}_{N+M}$.
In case this possibility appears to be feasible (an insecure question considering all possible problems to be treated), without the intervention of an external operator cooperating with the whole AI process, then it might appear the same as a digital machine can program itself. Nowadays, such self-programming ability still is a bit difficult to admit, without another program taking care of the initial one and doing the reprogramming job, say automatically. But such an exploit within a program's ability must be made with another program made by an external operator beforehand and somehow included in the whole procedure. Further reprogramming will initiate a never-ending process.
Hence, unless an external programmer re-programs the code or simpler: the AI algorithm acting with a re-programming issue, might re-educate itself accordingly.
An AI algorithm once trained means that one has fed the training stage with objects, described in some manner, and their attached tags. Once the AI is trained, the desideratum is such that the learned AI system in front of a new object, described in the same fashion as the ones previously used in the training period, could be able to generate part of some new unknown object tag.
If this previously unknown tag is not contained in the AI training phase, then that will be the same as new information has appeared from nowhere. In the case it is so, this freshly new information must be feed again as part of the training source. And the AI process will enter an unending loop or become corrupt at the end. The only possibility to avoid the appearance of such a loop is that the information generated because of the new object input corresponds to an interpolation of the information already known by the trained AI.
Therefore, it seems a bit out of question pretending that, after a correct training phase, a computational AI structure can generate by itself, even partially, new Boolean-logical structures, not contained in the Boolean-logical pool, the $N-$dimensional hypercube $\mathbf{H}_N$, organized by the initial training object set.
Nevertheless, even if some kind of procedure, generating new information not contained in the old experience, could be successfully implemented with the aid of new computing machines, and possessing additional self-programming abilities, one must be again aware that a unique bit, added to any old information structure, can duplicate the information content of the whole affair. This might revert an AI process into a never-ending learning procedure and drive the whole computational structure into a combinatorial information explosion.
For the moment, the best one can say about this problem is that any AI result appears forcibly deterministic, incomplete, and acausal. In agreement with the author of reference [30], one might learn and accept to become a bit humbler about the abilities of AI procedures.

13. Conclusion

A succinct revision of the fuzzy set definition, extending this mathematical structure into the tagged set construction, permits with the aid of Boolean hypercubes to create a general formalism involving Boolean vertices, fuzzy logical vertices, and natural number tags. This point of view results in an equivalent formalism, which permits to study of falsity-verity contents of logical object sets from three equivalent kinds of points of view, represented by the three kinds of different tags available. Because of the option to construct Boolean hypercubes recursively, then logical objects tags can be generated also recursively. This possibility causes the equivalent of Gödel's unprovability and incompleteness in fuzzy multivalued logical theories. Finally, such a result also permits the description of some limitations appearing in the AI procedures, which can be involved in the development of fuzzy logical systems based on computational grounds.

Conflict of Interest

The author declares no conflict of interest.

References

[1] R. Carbó-Dorca, *Fuzzy sets and boolean tagged sets*, J. Math. Chem., **22**(1997), 143-147.
[2] R. Carbó-Dorca,*Fuzzy sets and Boolean tagged sets, vector semispaces and convex sets, QSM and ASA density functions, diagonal vector spaces and quantum chemistry*, Adv. Molec. Simil., **2**(1998), 43-72.
[3] R. Carbó-Dorca, *N-dimensional Boolean hypercubes and the Goldbach conjecture*, J. Math. Chem., **54**(2016), 1213-1220.
[4] R. Carbó-Dorca, *Natural vector spaces, (inward power and Minkowski norm of a natural vector, natural Boolean hypercubes) and Fermat's last theorem*, J. Math. Chem., **55**(2017), 914-940.
[5] R. Carbó-Dorca, *Boolean hypercubes as time representation holders*, J. Math. Chem., **56**(2018), 1349-1352.
[6] R. Carbó-Dorca, Boolean hypercubes and the structure of vector spaces, J. Math. Sci. Model., **1**(2018), 1-14.
[7] R. Carbó-Dorca, *Transformation of Boolean hypercube vertices into unit interval elements: QSPR workout consequences*, J., Math. Chem., **57**(2019), 694-696.
[8] R. Carbó-Dorca, *Role of the structure of Boolean hypercubes when used as vectors in natural (Boolean) vector semispaces*, J. Math. Chem., **57**(2019), 697-700.
[9] R. Carbó-Dorca, *Cantor-like infinity sequences and Gödel-like incompleteness revealed by means of Mersenne infinite dimensional Boolean hypercube concatenation*, J. Math. Chem., **58**(2020), 1-5.
[10] R. Carbó-Dorca, *Boolean Hypercubes, Mersenne numbers and the Collatz conjecture* Research Gate Preprint DOI:10.13140/RG.2.2.28323.40482.
[11] R. Carbó-Dorca, T. Chakraborty, J. Comp. Chem., **40**(2019), 2653-2653.
[12] R. Carbó-Dorca, J. Math. Chem., **56**(2018), 1353-1356.
[13] M. Bunge, *Matter and Mind. A Philosophical Inquiry*, Boston Studies in the Philosophy of Science, Vol 287, Springer (Dordrecht) (2010).
[14] L.A. Zadeh, Information and Control, **8**(1965), 338-353.
[15] https://en.wikipedia.org/wiki/Fuzzy_set
[16] E. A. J. Sloane(Editor), *Online Encyclopedia of Integer Sequences*, https://oeis.org/wiki/Welcome, Sequence A083318.
[17] P.Jaccard, Bulletin de la Société Vaudoise des Sciences Naturelles, **37**(1901), 547-579.
[18] P.Jaccard, New Phytologist, **11**(1912), 37-50.
[19] T.T. Tanimoto, *An Elementary Mathematical theory of Classification and Prediction* ,Internal IBM Technical Report, (1958).
[20] D. J. Rogers, T. T. Tanimoto, Science, **132**(1960), 1115-1118.
[21] J. T. Tou, R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Pub. Co. Reading Mass., USA (1974).
[22] A. Rácz A, D. Bajusz, Héberger, J. Cheminform., **10**(2018), 48, https://doi.org/10.1186/s13321-018-0302-y.
[23] S. Gottwald S, *Many-Valued Logic*, http://plato.stanford.edu/entries/logic-manyvalued/, Stanford Encyclopedia of Philosophy, Metaphysics Research Lab, Stanford University (2009).
[24] https://en.wikipedia.org/wiki/Artificial_intelligence#Tools
[25] J. Devillers (Editor), *Neural Networks in QSAR and Drug Design*, Vol 2 in the Series: Principles of QSAR and Drug Design, Academic Press, San Diego, (1996).
[26] A. M. Virshup, J. Contreras-García, P. Wipf P, W. Yang W, D. N. Beratan, J. Am. Chem. Soc., **135**(2013), 7296-7303.
[27] L. K. Boerner (Editor), *Artificial Intelligence in Drug Discovery*, Chemical and Engineering News Discovery Report, American Chemical Society (2019).
[28] A. Acharya, R. Agarwal, M. B. Baker, J. Baudry, D. Bhowmik, S. Boehm, K. G. Byler, S. Y. Chen, L. Coates, C. J. Cooper, O. Demerdash, I. Daidone, J. D. Eblen, S. Ellingson, S. Forli, J. Glaser, J. C. Gumbart, J. Gunnels, O. Hernandez, S. Irle, D. W. Kneller, A. Kovalevsky, J. Larkin, T. J. Lawrence, S. LeGrand, S.-H. Liu, J.C. Mitchell, G. Park, J.M. Parks, A. Pavlova, L. Petridis, D. Poole, L. Pouchard, A. Ramanathan, D. M. Rogers, D. Santos-Martins, A. Scheinberg, A. Sedova, Y. Shen, J. C. Smith, M. D. Smith, C. Soto, A. Tsaris, M. Thavappiragasam, A. F. Tillack, J. V. Vermaas, V. Q. Vuong, J. Yin, S. Yoo, M. Zahran, L. Zanetti-Polzi, *Supercomputer-Based Ensemble Docking Drug Discovery Pipeline with Application to Covid-19*, J. Chem. Inf. Model., **60**(2020), 5832-5852.
[29] R. Carbó-Dorca, J. Math. Chem., **51**(2013), 413-419.
[30] https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology.
[31] J. Hermann, J. Schätzle, F. Noé, *Deep-Neural-Network Solution of the Electronic Schrödinger Equation*, Nature Chemistry, **12**(2020), 891-897.
[32] L. Reyzin, *Unprovability comes to machine learning*, Nature **65**(2019), 166-167.
[33] M. Bunge, *Causality and Modern Science*, Dover Publications (New York) 3rd Revised Edition, 2011.
[34] K. Gödel K, Monatsh. Math., **38**(1931),173-198.
[35] J. Chang, R. Carbó-Dorca, *Fuzzy Hypercubes and their Time-Like Evolution*, Research Gate Preprint: DOI: 10.13140/RG.2.2.26064.25605 (2020).