WeTEX and Hegelian contradictions in classical mathematics

S. K. Venkatesan

Abstract

We consider the contradiction between WYSIWYG and LATEX markup, in order to demonstrate a Hegelian synthesis (WeTEX, a KaTEX-based JavaScript implementation) of this contradiction. We then briefly consider other contradictions such as form vs. content in mathematics typesetting. After that we move on to the Hegelian contradictions in classical mathematics, starting from Zeno's paradox, leading to the hierarchy of infinities, continuum hypothesis, and finally the problem of algorithmic complexity classes.

"Reason has always existed, but not always in a reasonable form" — Karl Marx.

1 Introduction

Contradictions are mills through which reality ebbs and flows. The right and left hand sides exist as opposing manifestations. The fact that the perfect symmetry of the left and right is broken is what establishes their distinction. If Earth were a perfect sphere, then there would not be rivers and valleys that bring life to it. However, since the radius of the earth is almost a constant (to more than 99% accuracy), we are qualified to call it a sphere.

Contradictions are also a great source of changes to society and transitions from one form to another. Goliath was powerful when combat was direct physical conflict with hand-held weapons. When the giant Goliath calls David to come near him to fight, it is clear that only at that range can he overcome David. David, at the same time, keeps his distance to deny him that opportunity and defeats him with his long-distance weapon. Inside the atomic nucleus such a conflict between the Goliath of nuclear forces that operate only at short distances releases an enormous amount of energy when they come in conflict with electrical repulsion that operates at long distances, when the size of the nucleus becomes sufficiently large, as in the case of the uranium or plutonium nucleus. At the other end of the spectrum, when gravitational forces crush electrical repulsion in hydrogen ions, the Goliathian nuclear forces take over, releasing the enormous energy that powers suns.

In this article we first consider WeTEX, a KaTEX-based JavaScript implementation of an equation editor that resolves the conflict between the WYSIWYG (What You See Is What You Get) paradigm and the LATEX macros, a WYSIWYM (What You See Is What You Mean) system. We see how both of these contradictory approaches work at different user parametric ranges and use cases. The synthesis of these opposing paradigms produces a new application, WeTEX.

In the third section we consider Zeno's paradox, countable infinity and uncountable infinities of higher order. We also show how these Hegelian contradictions lead to a hierarchy of infinities.

In the fourth section we consider the Cantor set in the context of the continuum hypothesis and in the fifth and final section we consider a class of recursive algorithms for constructing the Cantor set and its complexity.

2 WeTEX — a synthesis between WYSIWYG and WYSIWYM

LATEX markup solved the problem of typesetting documents using a system of macros involving braces and backslashes. WYSIWYG systems, on the other hand, use a system of graphical menu objects with place holders for inserting text and symbols. This allows the user to directly visualize the output instantly with instant gratification. On the other hand, it is not easy to search for symbols in a character palette in a WYSIWYG system, so LATEX-like backslashed named entities are more convenient. Autocompletion prompts can further improve productivity in authoring such macro entities in LATEX, as some LATEX editors provide.

The WeTEX system was created as a proof of concept for a hybrid system. At present we deal only with authoring equations. We have used KaTEX's JavaScript rendering engine to implement this math editor. The KaTEX source code is available in a Github repository, `github.com/Khan/KaTeX`. WeTEX's open source (GPL-licensed) code is available in the Github repository `github.com/Sukii/WeTeX`.

In addition to the standard LATEX (KaTEX flavor) macros, WeTEX defines some additional macros and keyboard shortcuts, described in Table 1.

In addition to typesetting equations we will also be making an attempt to make the mathematics computable, wherever possible. A preliminary attempt at integration with Maxima [1], an open source GPL licensed maths computing library, has been made at `mathml.in`. Here the contradiction is between form and content, as we will be attempting to walk on two legs, quite similar to the literate programming approach for LATEX macro packages where one has to make a fine balance between the code and documentation. One the one hand the attempt of presentation aspect is beautiful typography, as an endeavor of a metal-based art form developed for many centuries in Europe. However, all this beauty has to be rooted in objectivity and realism, especially

Table 1: WeTEX macros and shortcut keys

WeTEX input	*Shortcut key*	*Description*	*Sample output*
`_`	shift `_`	Subscript	A_d
`^`	shift `^`	Superscript	A^d
`\t[]`	ctrl `-`	Tensor	A^{ijk}_{abc}
`\f[]`	shift `%`	Fraction	$\frac{a}{d}$
`\r[]`	shift `!`	Square root	$\sqrt{456}$
`\x[]`	shift `@`	Unicode text	Unicode
`\q[]`	shift `#`	Cube root	$\sqrt[3]{082}$
`\d[]`	ctrl `d`	Differential derivative	$\frac{\mathrm{d}X}{\mathrm{d}Z}$
`\p[]`	ctrl `p`	Partial derivative	$\frac{\partial X}{\partial Z}$
`\i[]`	alt `i`	Integration	$\int_X^Z Y\,\mathrm{d}X$
`\s[]`	alt `s`	Summation	$\sum_X^Y Z$
`(())`	ctrl `(`	Left & right parentheses	$\left(\frac{a}{z}\right)$
`[[]]`	ctrl `[`	Left & right brackets	$\left[\frac{a}{z}\right]$
`{{}}`	ctrl `{`	Left & right braces	$\left\{\frac{a}{z}\right\}$
`\8`	ctrl `8`	Infinity symbol	∞
`\0`	ctrl `.`	Centered single dot	$\cdot$
`shift ~`		Similar	$\sim$
`ctrl =`		Equivalent	$\equiv$

the functional requirement of mathematics and its development rooted in industrial society, publishing books and journals both in electronic and print forms. Modern electronic devices also now allow us to develop newer forms that can interact more closely with users both in form and content, allowing much broader dissemination of content, forcing us to think about accessibility requirements of content transcending the current form. However the spirit of accessibility is not only a technical requirement, but also requires that we change the form (pun unintended) of the content also, making it accessible to a broader audience.

3 Classical Hegelian hierarchy of infinities

We will begin with the classical Zeno's paradox that is usually explained in terms of Achilles and the tortoise [2].

3.1 Zeno's paradox

However, we will illustrate this in a much simpler way using the concept of recursive decimals:

$$0.99999\cdots = 1\,.$$

We know from this that there is an infinite sequence of numbers, 0.9, 0.99, 0.999, …, which are bounded above by the finite value 1.0. The finite value 1.0 can be expressed by the infinite sequence of decimals represented by the above equation. Of course, there are many such infinite sequences that approach 1.0, each with its own rhyme and rhythm. The fact that the infinite is contained in the finite value of 1.0 is indeed a paradox in classical logic, but not in paraconsistency logic [3], which allows such an inconsistency to exist without great consternation. In classical logic A and −A cannot exist as truth together. It would create a blow-up. Consider the classical Boolean logic statement:

$$(\mathrm{A\ or\ B})\ \mathrm{and}\ (-\mathrm{A\ or\ B})\ \mathrm{is\ true}\,.$$

In classical logic we can conclude from this statement that B is true. This is considered a blow-up, as in classical logic an unrelated statement B would be true if both A and −A were to be true. Paraconsistency logic [3] mitigates this blow-up. The name "paraconsistency" was coined by the Peruvian philosopher Francisco Miró Quesada [4].

3.2 Hegelian hierarchy of infinities in classical mathematics

Let us now consider the cardinality of a set as the number of elements in a set, for example, if

$$A = \{\text{dog, cat, horse, car, bus, train, aeroplane}\}$$

then $\#(A) = 7$. Now consider the set of all subsets of A, symbolically written as the set 2^{A}. This notation for the power-set is justified by the relation

$$\#(2^{\mathrm{A}}) = \sum_{i=0}^{\#(\mathrm{A})} \binom{n}{i} = 2^{\#(\mathrm{A})}\,.$$

We would like to point out that this process can be continued indefinitely, i.e., we can construct the power-set of a power-set or, in plain language, that there can be a set of all subsets of the set of all subsets of a set, etc., i.e.,

$$\mathrm{A}, 2^{\mathrm{A}}, 2^{2^{\mathrm{A}}}, \ldots\,.$$

Now let us consider the set of natural numbers,

$$\mathrm{N} = \{1, 2, 3, \ldots\}$$

The cardinality of natural numbers is denoted by aleph,

$$\aleph = \#(N)\,.$$

Now consider the cardinality of the set of all real numbers between 0 and 1, i.e.,

$$c = \#([0,1])\,.$$

Just as the number of drops of water in a glass of water cannot be counted like a bunch of bananas, we intuitively know that the set of real numbers in [0,1] cannot be counted like natural numbers. We will prove this now.

Let us represent all real numbers between 0 and 1 in terms of their binary representation in some counting order,

$$a_1 = 0.01010\cdots$$
$$a_2 = 0.110010\cdots$$
$$a_3 = 0.101011\cdots$$
$$\cdots\,.$$

By a diagonalization process discovered by Cantor we can construct a real number,

$$b = 0.100\cdots ,$$

where b is obtained by reversing the 0s and 1s of the i-th digit of a_i, such that

$$b \notin \{a_1, a_2, a_3, \ldots\} ,$$

leading to a contradiction, thus proving that it is not countable. However, this is only a qualitative result. We will now prove a quantitative result, that

$$c = 2^{\aleph} .$$

In order to prove this, consider the binary representation of a real number in (0,1),

$$x = \{0.010011001\cdots\} .$$

We now obtain the corresponding subset of natural numbers by considering all the index positions of "1" in the above binary representation, i.e.,

$$\mathrm{S} = \{2, 5, 6, 9, \ldots\} .$$

Similarly, for every subset of natural numbers we can construct a real number in (0,1). This implies a bijective mapping between the power-set of natural numbers and the set (0,1), proving the result. QED

So finally we also have the result that we can construct an infinite hierarchy of infinities, i.e.,

$$\aleph, 2^{\aleph}, 2^{2^{\aleph}}, \ldots .$$

Although here we are only discussing about numbers and the mathematics of set theory, these results have much greater implication in computer science as there is a corresponding categorical mapping at higher levels that maps these domains to similar problems there. At an abstract level, decision problems can be considered as function mappings,

$$f :\ A \to \{0,1\} ,$$

where $A \subset \mathrm{N}$.

So in essence a decision problem can be mapped to a real number. However, at the same time it can be shown that the set of algorithms or procedural programs is countable, as a Turing machine can be reduced to a natural number, a binary state of the computer. Putting the two facts together, we get the result that not all decision problems can be solved accurately by a computer as real numbers are uncountable. However, rational numbers are countable (as they are like two-dimensional natural numbers, they can be counted in a zig-zag way starting from the top right corner) and they are dense in real numbers (which means that a sequence of rational numbers can sufficiently approximate any given real number). So every decision problem can be solved approximately by a computer, although the degree of approximation varies depending on the decision problem and the computer algorithm.

4 The Cantor set and the continuum hypothesis

The continuum hypothesis states that there are no cardinal numbers between $\aleph$ and $2^{\aleph}$. We will now argue against this, but the argument has to be considered from the point of view that there are better measures that distinguish between different shades of infinities than just counting bananas, namely, by weighing them.

Let us now consider the Cantor set, $\mathscr{C}$, which can be obtained by recursively removing the middle one-third (but keeping the end points during the removal) of the set of real numbers in [0,1]. Consider the sequence of sets, $C_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, $C_2 = [0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{1}{3}] \cup [\frac{2}{3}, \frac{7}{9}] \cup [\frac{8}{9}, 1], \ldots$, defining the Cantor set to be

$$\mathscr{C} = \bigcap_{k=1}^{\infty} C_k .$$

It can also be symbolically written as a geometric set following the relation

$$3 \times \mathscr{C}(0) = \mathscr{C}(0) \cup \mathscr{C}(2/3) .$$

Or to put it more simply, when we scale the Cantor set by 3, we get two Cantor sets, i.e.,

$$3^D = 2 ,$$

where $0 < D < 1$ is the scaling dimension of the Cantor set and from this relation we obtain $D = \log 2/\log 3 \approx 0.63$.

For example, we can see that natural numbers are 0-dimensional points in Euclidean space that don't scale, while the interval [0,1] scales linearly, a two-dimensional square scales quadratically, a three-dimensional cube grows to the cubic power, etc.

We can also prove easily that the Cantor set is uncountable. Consider the ternary representation of a real number in (0,1), i.e.,

$$x = 0.0102201\cdots .$$

The points in the Cantor sets will not have digit "1" in them, i.e.,

$$x = 0.002022002\cdots$$

Similar to the proof involving the binary representation of real numbers we can show that the Cantor set, $\mathscr{C}$, is uncountable. Using a similar procedure we can also show that

$$\#(\mathscr{C}) = 2^{\aleph} .$$

One then wonders how this is a counter example to the continuum hypothesis? The answer lies not in counting bananas but weighing bananas, as there is more geometric information in the Cantor set that is lost in counting rather than weighing them. It is this extra geometric information that is captured by the scaling dimension, which distinguishes it as a fractal object existing between the 0-dimension and 1-dimension. This quantitative aspect becomes clear as we deal with algorithms that generate the Cantor set and their complexity in the next section.

5 Algorithmic complexity of the Cantor set and the power of iterative functional formulations

Let us now consider the iterative algorithm for constructing a Cantor set by considering the n-digit ternary representation of real number between 0 and 1, such as

$$x = 0.\underbrace{012102002\cdots 1020}_{n}\,.$$

The Cantor set follows the recursive relation,

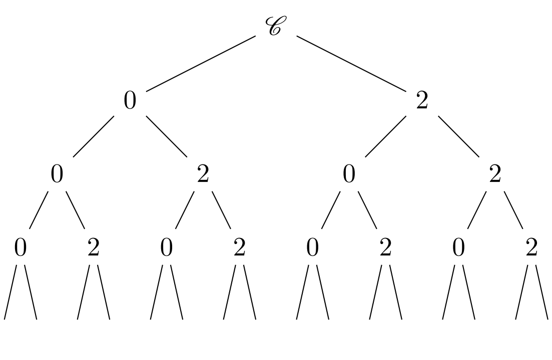
$$\mathscr{C} = \frac{1}{3}\mathscr{C} \cup \left(\frac{1}{3}\mathscr{C} + \frac{2}{3}\right)\,.$$

This can be used to define a recursive push-down from the left,

$$x \to \begin{pmatrix} \frac{1}{3}x \\ \frac{1}{3}x + \frac{2}{3} \end{pmatrix},$$

which can be represented using a constant push of 0s and 2s from the left as in the tree-like representation shown in Figure 1.

Figure 1: Binary tree formulation of the Cantor set



To construct the real numbers between 0 and 1 in an n-digit ternary representation, we need a decision tree with 3^n steps, while to construct the Cantor set, we need a tree of 2^n steps. As there are no further information or constraints that can be retrieved either from geometry or from its ternary representation, so this is the minimal complexity that can be achieved in order to compute the points in the Cantor set, $\mathscr{C}$. This then proves that the Cantor set computation problem cannot be solved in polynomial time.

However, we have discovered how a single algorithmic step in recursive functional formulation requires 2^n operations in the state machine. Checking the results (2^n states of n-digits) of this output to see if these n-digits are distinct points in $\mathscr{C}$ requires equal or more effort. However, if we are using a stateless function to generate the output then it is enough to test only a few points in the Cantor set to check for the veracity of 2^n values (just as in a cooked pot it is enough to test a few particles of rice to see if it is cooked). So this effectively reduces the problem to a NP-class problem. This shows the power of recursive functions and functional formulations of the lambda calculus. The power of neural networks in modelling data comes precisely from this iterative functional formulation.

The Cantor set is a simple example of a computable fractal. However, there are more complex (pun unintended) fractals that are not even computable. Penrose [5] conjectured that some Mandelbrot sets are not computable, and this has been confirmed [6]. However, here again, these Mandelbrot fractals are formulated in terms of complex functions, another example of iterative functional formulation, which in this case is not even computable in terms of state machines.

Finally, we would like to mention that the power of recursion was considered in the early 1960s by Noam Chomsky who quoted the famous phrase of Wilhelm von Humboldt, “infinite by finite means” and later on by Douglas Hofstadter in his popular work [7], where he also makes interesting references to Metafont.

References

[1] Maxima (1982). A Computer Algebra System. `maxima.sourceforge.net`

[2] Aristotle (350 BCE), Zeno’s paradox, translated by R.P. Hardie and R.K. Gaye. `classics.mit.edu/Aristotle/physics.6.vi.html`

[3] Béziau, J.-Y. (1970). Future of Paraconsistency Logic. `wwwa.unine.ch/unilog/jyb/future-pl.pdf`

[4] New Directions in Paraconsistent Logic, 5th WCP (2014) Kolkata, India, February 2014, Béziau, J.-Y., Chakraborty, M., and Dutta, S., eds.. Springer-Verlag.

[5] Penrose, R. (1989). The Emperor’s New Mind. Concerning Computers, Minds and The Laws of Physics. Oxford University Press, New York.

[6] Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. Amer. Math. Soc., 21(1):1–46.

[7] Hofstadter, D. (1979). Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books.

◇ S. K. Venkatesan
TNQ Technologies, Chennai, India
`skvenkat (at) tnqsoftware dot co dot in`