DIVERSITY OF EML-TYPE OPERATORS

A. Odrzywołek

Institute of Theoretical Physics, Jagiellonian University
Łojasiewicza 11, 30-348 Kraków, Poland

Received 5 September 2026, accepted 26 October 2026, published online 16 November 2026

The discovery of the EML operator, sufficient to evaluate the standard explicit purely transcendental elementary functions, has led to considerable interest and discussion across multiple scientific disciplines. However, most authors have focused on the binary EML itself, while numerous similar variants with slightly different properties are now known. This article attempts to close this gap by enumerating and classifying them. We also take this opportunity to clarify common misconceptions related to the EML operator. The principal goal, symbolic regression within an architecture as close as possible to proven neural networks which combine matrix multiplication with a single univariate non-linear activation function, remains beyond reach. Instead, we propose a Möbius layer, with rational functions replacing matrix operations, and showcase the recently discovered activation function $\mathrm{eml}(x, 1/x)$, which allows $\exp(x)$ and $\ln(x)$ to be recovered separately, and hence all elementary functions to be evaluated within a rational generalization of the neural network.

DOI:10.5506/APhysPolB.57.12-A4

1. Introduction and follow-up studies

The EML operator

$$\mathrm{eml}(x, y) = \exp(x) - \ln(y) \tag{1}$$

has been proposed in [1]. It was verified extensively both numerically (including in arbitrary precision) and with a Computer Algebra System (Wolfram Mathematica). The core claim is that using the EML operator, together with input variables and the special distinguished constant 1, one can derive formulas which reconstruct all standard scientific calculator operations. These include the usual arithmetic $(+, -, \times, \div)$, exp and ln, binary exponentiation and logarithm, integers and rationals, constants such as π, e, and i and all trigonometric/hyperbolic functions with their inverses. These formulas work in their respective real domains, except at the endpoints in some

cases. For trigonometric functions and π, the EML operator must use complex intermediate values. This is done entirely within the principal logarithm branch (the fundamental strip of the complex plane). The shortest formulas are obtained using the extended real axis, $\ln 0 = -\infty$ and $e^{-\infty} = 0$. Since then, formalized Lean4 proofs [2] have appeared supporting the reconstruction, with qualifications discussed in Subsection 2.2. In what follows, we assume the validity of the main result of [1]. The operator now has its own entry in MathWorld [3].

The surprisingly simple properties of Eq. (1) sparked internet discussions[1] about its possible applications and its relation to the logical Sheffer stroke [4] (NAND/NOR binary operators) which enables the computation of all Boolean functions. Indeed, the context-free grammar is extremely simple

$$S \to 1 \mid x \mid \mathrm{eml}(S, S) \tag{2}$$

and is isomorphic to binary trees, one of the best-studied structures in computer science.

One possible application of EML operators and their relatives is symbolic regression (SR). SR attempts to fit data with free-form expressions composed of arbitrary functions and operations. Usually, this requires mixing discrete and continuous approaches [5]. First, genetic algorithms generate multi-parameter candidate formulas. Then, the free parameters are optimized using *e.g.* gradient descent. EML, Eq. (1), in principle allows the discrete step to be skipped entirely by using the master formula [1] encoding a full binary expression tree of depth d

$$T_{\mathrm{eml},d}\left(x; \{a_1, \ldots, a_k\}, \{b_1, \ldots, b_k\}, \{c_1, \ldots, c_{2^d-2}\}\right), \; k = 2^{d+1} - 2\,, \tag{3}$$

intentionally written in a form resembling the generalized hypergeometric ${}_pF_q$, Meijer G, and Fox H functions. The master formula (3), similarly to G or H, reduces to elementary functions for a special choice of parameters, *e.g.*,

$$e^x = {}_0F_0(;;x) = G^{1,0}_{0,1}\left(-x \,\middle|\, \begin{matrix} \\ 0 \end{matrix}\right) = T_{\mathrm{eml},1}\left(x; \{0,1\}, \{1,0\}, \{\}\right) .$$

The above property can be used for gradient-based symbolic regression [1, 6, 7] and an experimental algorithm for constant recognition [8]. However, for

[1] See *e.g.* the Hacker News (HN) thread `https://news.ycombinator.com/item?id=47746610` (858 points, 298 comments; accessed August 2026), the 6th highest overall among ∼24,000 HN submissions linking to `arXiv` and the 1st for a computer science/math topic. For comparison, the AI-generated counterexamples of May–August 2026 (Subsection 2.5) scored 632–1429 points on HN. Other discussions included X (Twitter) trending topic on EML, with ∼800 posts and over 10^7 views in April 2026; YouTube and TikTok videos with over 10^6 views in total, ∼ 100 GitHub repositories replicating and formalizing main result, and many more.

EML, depth and symbolic recovery rates were very limited. In the follow-up study [9], three depth-3 architectures were compared on over 12,000 training runs; recovery of the same target ranged from 0% to 100% depending on the architecture and training protocol, and balanced (non-chain) trees were never recovered. In these experiments, the optimization landscape, not expressivity, is therefore the limiting factor. This is precisely where we hope EML variants (Section 3, Table 1) might help.

EML trees were used as building blocks for simultaneous function and antiderivative discovery [10], as response modules in reduced models of biological dynamics [11], and as interpretable edge mechanisms in causal structure learning [12]. A recent theorem also establishes universal approximation by EML trees in $W^{k,\infty}$, together with a practical fitting procedure [13]. Independent implementations now exist in Rust, Python, and JavaScript [14].

The rest of the article is organized as follows. In Section 2, we address some questions which arose after the EML discovery was made public. Section 3 showcases the known variants of the EML operator. In Section 4, we propose a new architecture upgrading neural networks, and the article is concluded with a short summary.

2. Relevant mathematical background and common misconceptions

Months of intense discussion of the EML operator have resulted in a much deeper understanding of its relation to prior mathematical knowledge. While EML and some of its variants were discovered by an exhaustive brute-force numerical sieve, more explainable approaches now exist [15]. Some new formulas are far more complex than a direct search using limited computing resources could reach.

In the following subsections we take the opportunity to explain common misconceptions and *nitty-gritty details*, which resurfaced during discussions of the EML operator.

2.1. All elementary functions?

Some confusion comes from the not universally accepted notion of *elementary functions*. The meaning of this phrase is different not only between scientific disciplines, but also between languages and countries, and even generations. For example, in Poland, math teaching clearly distinguishes between *explicit* and *implicit* ones (Polish: funkcje jawne/uwikłane). While in [1] an unambiguous enumerative definition of what is meant by *elementary functions*, *i.e.* explicit finite expressions composed using standard scientific calculator buttons, was provided, some confusion among unprepared readers arises. Our definition is the same as in complex analysis textbooks, see *e.g.*

[16, Chapter 5, Elementary Functions]; also [17, p. 746]. However, some readers of [1] did not follow further than reading the preprint title, not to mention its abstract or full text, and just a few reached the *Supplementary Information*. The purely mathematical definition is used mainly in differential algebra, and arose in the context of integration in finite terms. This is a historically famous problem, second only to the solution of cubics/quartics/quintics in importance for shaping the modern notion of what we mean by elementary functions. The well-known example providing a rationale for extending the enumerative mathematical definition of *elementary function* to include algebraic functions is the symbolic integration of rational functions. The classic algorithm for that is at the core of the STEM teaching curriculum, and uses decomposition of the rational integrand into partial fractions. Consider $\int 1/(x^4+1)\mathrm{d}x$ *versus* $\int 1/(x^5+2x+1)\mathrm{d}x$. To integrate these, the roots of the denominator, in general, are required to be known. If we insisted that they must be found in the form of radicals[2], then the latter, a generic rational function with fifth or higher degree polynomial in the denominator, would not be symbolically integrable in finite terms. Therefore, roots of all polynomials must be *adjoined* (using differential algebra terminology). The standard definition [18] goes one step further and includes roots t_i of polynomial equations $P(t)=0$ with coefficients being elementary functions of x themselves, recursively. Differential algebra also considers a broader class, formalized by *Liouvillian extensions* [19, Definition 6.9], which allow integration of functions already obtained. This class includes erf, Ei, and li, which are not elementary in the sense of [18, p. 971]. Both definitions are rigorous continuations of the work of Liouville, Ritt, Kolchin and others [20, pp. 12–18]. Our enumerative definition uses only $+,-,\times,/$, exp, and ln. The reader should be warned that some authors apply the word *elementary* to the second class as well, and that a textbook may introduce its own working definition for the purpose of a particular proof, *e.g.* [19, Definition 6.11], which admits logarithms of rational functions only and therefore excludes even $\ln(1+\exp x)$, a function crucial in physics. In this situation, to the great surprise of STEM practitioners, solutions to *e.g.* $t^5-t+z=0$ with unknown t and parameter z (*e.g.* the 2^{nd} real root t_2) are *elementary functions* $f(z)$ of the variable z in the sense of [18], despite being *explicit* only after using a hypergeometric function which does not reduce to exp–log expressions [21]

$$f(z)=t_2=z\,{}_4F_3\left(\left\{\tfrac{1}{5},\tfrac{2}{5},\tfrac{3}{5},\tfrac{4}{5}\right\};\left\{\tfrac{1}{2},\tfrac{3}{4},\tfrac{5}{4}\right\};3125z^4/256\right).$$

Thus, if one interpreted the title of [1] in a broader differential-algebraic strictly mathematical sense, then an EML expression would provide *e.g.* con-

[2] Polynomial x^5+2x+1 is irreducible over $\mathbb{Q}$ with Galois group S_5, which is not solvable, and its roots are not expressible in radicals.

structive exp–log solution $f(a, b, c)$ to Hilbert's 13^{th} problem, which involves roots t_i of the 7^{th} order polynomial $t^7 + a\,t^3 + b\,t^2 + c\,t + 1 = 0$ with 3 parameters a, b, c. This is, of course, not what [1] claims, and the algebraic-function version (*cf.* Subsection 2.7) of the Hilbert's 13^{th} remains open.

The above did not stop some authors from proving that EML is not able to express "ALL" *elementary functions*, despite the difference in definitions of this phrase. An example of a constant which cannot be expressed in terms of the EML operator and 1 is Chaitin's constant [22]. However, it is also not computable. Another blog post discusses the roots of the generic quintic [23], which again is not an EML-class expression. What is expressible by EML are actually Chow's [24] EL numbers. Carney presents the equality of these sets as Proposition 1 in [22].

As a bottom line, I would like to stress that choosing the title for the EML paper [1] was not an easy choice. To include all relevant information in the title, it would have to be ridiculously long and complicated, *e.g.*

A single binary operator for computing exp–log functions on the extended real line, almost everywhere, under IEEE 754 semantics, with algebraic adjunctions replaced by Chow's EL numbers, via complex-valued intermediate values, from the constant 1?

or similar. This would be consistent with the modern struggle to attract the attention of potential readers, who only skim titles, rarely read the abstract, and do not even attempt to read the full text, not to mention the *Supplementary Information*. However, information about the existence of the EML operator might be important to the large community of machine learning and electronic engineers, bioinformaticians, theoretical computer science experts, and educators. A title narrowly aimed at mathematicians would mislead the core audience. On the other hand, a casual title, *e.g.*, *Two-button calculator* would suggest recreational math, without any applications in sight. For many STEM educated people, elementary functions are scientific calculator operations.

2.2. Complex domain and branches

Another common misconception related to the EML operator is the use of complex intermediate values to generate real functions. While this sounds natural for theoretical physicists, *e.g.* quantum mechanics operates exactly that way (complex wave function, real measurements), more mathematically aligned people were confused: what is the domain of (1)? If one presumes it is the natural real domain $\mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$, the entire reconstruction becomes impossible. As proved by Hardy in [25], one cannot obtain $\sin x$ from $\exp, \ln$, and arithmetic within the real domain, because it oscillates at infinity [26,

Chapter III, Section 2, Theorem on p. 18]. If not real, then maybe a complex mapping $\mathbb{C}^2 \to \mathbb{C}$ of EML arguments is enough for the full reconstruction of the elementary function chain? However, while $\exp(x)$ is well-defined on the entire complex plane, $\ln(y)$ is infinitely-valued. It is a true function only when defined on its Riemann surface, except at (complex) zero. In practice, $\mathrm{eml}(x, y)$ is defined on $\mathbb{C}\times(\mathbb{C}\setminus\{0\})$ using the principal branch of the complex Log. Moreover, all functions to be reconstructed by EML are *real*, $\mathbb{R}^n \to \mathbb{R}$. Therefore, complex inputs to $\mathrm{eml}(x, y)$ appear only at internal expression nodes. Both inputs and outputs of EML expressions are real, while inputs are additionally restricted to be within the real domain of the reconstructed expression, with the possible exception of domain endpoints [2].

The main surprise of [1] preprint result is that the above *does work*. In principle, one would expect that for some big EML expression tree internal values escape fundamental strip, and entire reconstruction breaks. Initially, a lot of attempts were made to find such a counterexample. All of them appeared to be bugs in EML compiler. Formal Lean4 results are available in [2], although some trigonometric statements use input-dependent expressions or real/imaginary-part projections.

Note that question whether similar reconstruction for $\mathbb{C} \to \mathbb{C}$ elementary functions, even reduced to principal branch, is possible, is a separate and unanswered question. If it is possible, it likely would require operator much different from (1). Reconstruction of complex-valued multivalued elementary functions in a similar manner looks like an impossible task, as they are defined on distinct Riemann surfaces.

2.3. Use of extended real axis and logarithm at zero

The original construction used in [1] uses a Bourbaki-style extended domain, including infinite points. In particular, the reconstruction relies on the formulas

$$\mathrm{e}^{-\infty} = 0\,, \qquad \ln 0 = -\infty\,.$$

This is confusing for many STEM practitioners, for whom the logarithm at zero is undefined, and $\pm\infty$ are not valid arguments or values. Moreover, since EML operates in the complex domain, one must be careful whether we are using complex infinity, directed infinity, or real infinities. In practice, the above poses no problem, as we use EML expression within Computer Algebra Systems like Mathematica, which handle the above natively. The same applies to IEEE754 numerical floating-point computations, which define and handle infinities automatically. Therefore, depending on the level of error handling, EML expressions are easily evaluated in many programming languages. But not in all of them. For example, Python's and Maple's standard logarithm routine raises an error at zero. This motivated the cre-

ation of *clean* EML compiler, which never touches the troublesome ln(0), at the expense of much longer expressions. So far, this looks doable. In particular, Lean4 proofs must avoid the complex Log(0), because, due to the requirement for Log to be a total function (full $\mathbb{C}$ domain, including zero), "junk" value Log(0) = 0 is imposed. This breaks reconstructions that use the extended value at zero, and [2] must avoid it.

2.4. EML as NAND/NOR equivalent for continuous math

As Rieu pointed out [27], the history is somewhat more intricate than the modern identification of the Sheffer stroke with NAND suggests. Peirce discovered the functional completeness of both NAND and NOR around 1880, although his result remained unpublished, and Stamm published both forms in 1911. Sheffer's 1913 paper selected the NOR interpretation ("neither-nor"), whereas Nicod's 1917 treatment adopted the NAND interpretation now conventionally associated with the stroke [4, 28, 29]. Nicod also showed a single axiom[3] sufficient for propositional calculus, simplified by Łukasiewicz [30]. Their later significance for digital computation rests especially on Shannon's identification of Boolean algebra with relay and switching-circuit synthesis [31].

In some sense, the situation is similar to EML. While (1) was discovered first, at least five equivalents are now known, *cf.* Table 1. The notion of "sole sufficient operator" in both cases (NAND, EML) in fact refers to multiple cases: NAND/NOR *versus* EML/EDL/LDE/PLI/PLM. The analogy between digital NAND and continuous EML is quite close. One difference is the requirement to use the constant 1 for EML to work, while NAND does not require this. But in practice, all circuits use external 0/1 inputs anyway. NAND can compute any Boolean function and approximate any other function. The same is true for EML. It can compute any elementary function, which can, in turn, approximate any functions, using polynomials (series, Chebyshev) rational functions (Padé), Fourier series, integration (Gaussian, double-exponential), ODEs (Runge–Kutta methods) and, last but not least, neural networks. All aforementioned methods have proved useful in science.

The problem of generating EML values (output) is technical, not fundamental. In the late 19$^{\text{th}}$ and early 20$^{\text{th}}$ century, no one had any idea how to implement NAND/NOR in a massive, efficient way. Babbage's Analytical Engine [32] was purely mechanical, Shannon [31] discussed electromechanical switches. Only in the 1960s, first in the Apollo Guidance Computer, did we start using integrated-circuit NOR gates and later NAND TTL gates. We hope the engineering and adoption of EML-type operators will be much faster, in either digital or analog form.

[3] For EML, the analogue of it would be an identity between elementary expressions.

2.5. Use of the brute-force search

Exhaustive computational searches have a long and successful history in the discovery of mathematical counterexamples. For example, Lander and Parkin used a direct search to disprove Euler's sum-of-powers conjecture by finding $27^5+84^5+110^5+133^5 = 144^5$ [33]. Another example is the exhaustive enumeration by Đoković showing that Williamson matrices of order 35 do not exist, thereby disproving the Williamson conjecture [34]. Computational searches have also been used to formulate new conjectures [35]. Here, we used them to select arithmetic operators with given properties. Therefore, enumeration was used recursively. First, we enumerated candidate operators, then all formulas composed of them. A common misconception related to [1] is that this was all that was done. In fact, the exhaustive search was meant to provide *candidate* operators. EML itself, once marked as first possible successful candidate, was subject to very extensive numerical and symbolic hand-crafted verification.

By design, the fast numerical sieve, based on numerical constant recognition, might return false-positives. This can happen due to numerical round-off errors or floating-point tautologies. We recall that floating-point numbers, while intended to approximate reals, in fact are dyadic rationals. Their set is huge, but finite, and calculations are discrete, not continuous. Our procedure works as follows. First, we generate some elementary formula which is representing a two-input operator, similar to (1). Then we want to know, if some combination of these operators is equivalent to a standard mathematical operation, such as square root or multiplication. Instead of slow and error prone symbolic simplification, we attempt to establish floating-point identity at some point. This cannot be value like, say $x = -2/3, y = \pi$, because both of these numbers are already elementary. We need some "truly transcendental" constant. The best situation would be if we know that it is at least irrational, but for most convenient cases, such as Khinchin $K \simeq 2.68545$, Glaisher $A \simeq 1.28243$, Euler gamma $\gamma \simeq 0.577216$, or Catalan $C \simeq 0.915966$ constant, there are no relevant proofs[4]. Later, we proceed as if these constants were truly outside the exp–log class. Once we compute numerically a high-precision value, *e.g.* $\sqrt{A} \simeq 1.13244299154554470529533417519 03$ or $K \times C \simeq 2.45978163779018520957534914060 4$, we can run constant recognition software [8] on the result. If the fast numerical search provides an equivalent candidate formula using *e.g.* $\mathrm{eml}(x, y)$ and 1 only, which agrees to some assumed error of a few ULPs with the expression sought, then additional checks (arbitrary precision, symbolic, multi-point) are executed. If they pass, the formula is marked a valid candidate, and we proceed to the next case [1, 2].

[4] See, however, recent preprint on Catalan [36].

If the entire set of operations which define our "elementary functions" is reconstructed in the above way, the result is subject to detailed, hand-crafted symbolic and numeric verification. For details, see [1].

What might be surprising is that the above procedure was able to discover anything. One might call it luck, however it was the culmination of very long research on brute-force methods in symbolic regression, started as early as 2009[5]. The idea was to use a mix of random/genetic and exhaustive search to sweep unsolved problems at scale. In practice, it was limited to occasional solving algebraic equations, evaluating definite integrals, solving ODEs or identifying numerical constants from various sources of experimental mathematics or physics. To our unpleasant surprise, for a decade, this programme led only to the re-discovery of already known results in different forms. See, for example, [37]. This *unreasonable effectiveness of human science* in solving problems was a surprise. It looked like everything that could be found, was already found. No gaps in knowledge, no missing solutions, no forgotten old problems. In fact, the EML discovery was the first major success of the above philosophy. In some sense, finally it was demonstrated to be right. It is conceivable that throwing substantial computational power and programming effort at some unsolved scientific issues could work in the above way.

Now, in 2026, deep learning neural networks in the form of LLM chatbots (ChatGPT, Grok, Gemini) and programming agents (Codex, Claude Code) are overshadowing any other attempts in machine learning. In the last several weeks, torrents of counterexamples[6] to decades-old conjectures (*mensis mirabilis*) generated by those AI systems have left the world's top-class mathematicians, including Fields medalists, in shock. The broader debate is reflected in the earlier Leiden Declaration [38]. Some of these conjectures are iconic in modern math, and many geniuses attempted solving them. In retrospect, some of these conjectures (*e.g.* the Jacobian conjecture, DGG, Maxwell) could have been disproved many years ago by brute-force search, but no one tried hard enough. Things are happening too fast now to even think about possible consequences. But it looks like *bitter lesson* [39] *i.e.* applying an enormous amount of computing power and data at any optimization problem, is working nearly as well as the most sophisticated dedicated

[5] *Cf. e.g.* https://th.if.uj.edu.pl/~odrzywolek/homepage/presentations/PL/Approx.pdf (in Polish) where the KAN idea [40] was presented as well.

[6] May 20, 2026: internal OpenAI model, Erdős unit-distance conjecture (1946) [41]; July 19: L. Alpöge, Jacobian conjecture (Keller, 1939), false for $n \geq 3$ [42]; July 22: D. Rybin with GPT-5.6 Pro, Dinitz–Garg–Goemans conjecture (1999); July 29: Arathoon, Ball, Kvalheim with GPT-5.6 Sol, Maxwell conjecture (1873) on equilibria of point charges [43]; August 1: OpenAI "Ten advances", including first non-sofic group and disproof of Connes's rigidity conjecture [44].

"intelligent" solutions, including human science. This, in turn, revives the old major questions in the early 21st century: *what is intelligence* and *where does genuinely new knowledge come from*?

Brute-force exhaustive enumerative search[7] seems at first glance to be the opposite of intelligence. The iconic use case of a student who patiently substitute subsequent integers[8] into a homework equation in an attempt to solve it is an example of nonsense work. We expect them to use some algorithm instead. But where did the algorithm come from? We can enumerate candidate procedures as well. Once found, it can be used forever, and possibly enhancing future search for other objects. This looks more like developmental biology than scientific progress. The first question, of intelligence, we leave open, together with other hard problems like consciousness. On the second question, where genuinely new knowledge comes from, our opinion is that brute-force search in various disguises plays a much larger role in discovery than commonly acknowledged. Justification and explanation are usually provided *post factum*, while correct answers are reached through unconscious associations or even plain luck. In this light, there is no surprise that modern AI systems are said to be unable to create new ideas beyond mixing and application of what was in the training data. The fact that "trivial" counterexamples to famous conjectures are now found by AI is then a sign that brute-force methods have been undervalued by the scientific community rather than a symptom of emerging AGI (Artificial General Intelligence) or ASI (Artificial Super Intelligence).

This leaves the intriguing possibility, that redirecting the world's computational infrastructure into exhaustive search instead of AI training might be the most efficient use of it. Notable examples of such searches of the "mathematical universe" [45] include Wolfram's Physics Project [46] and Taelin's discrete program search on the HVM [47].

2.6. Relation to Turing machine and universal computation

A frequent question is how EML relates to established computational models like the Turing machine. EML real-function formalism is closer in spirit to analog computations [48, Chapter 12, Section 4, note (c), "Continuous computation," p. 1128], in which the exponential function is a primi-

[7] Monte Carlo and genetic algorithms can explore a much larger search space of longer expressions in a non-exhaustive and unordered fashion. Running for long enough, however, they should, in principle, be able to return the same "simplest" results as exhaustive enumeration.

[8] For years, we benchmarked the ability of hardware to substitute ALL $2^{32} \simeq 4.3 \times 10^9$ single-precision floats into transcendental equation $e^x = x^2$ in an attempt to solve it. Initially, around 2004, a single-core Athlon XP required 15 minutes to complete this task. Modern multi-core CPUs are at the seconds level. Now, an RTX6000 PRO using CUDA completes the search in ... 10 ms!

tive object. Even integers must be constructed from it. A list of the first 1000 integers expressed in pure EML form is given in [49]. A more confusing example is the representation of π [50]. Its infinite decimal expansion $\pi = 3.14159\ldots$ has been computed to 314 trillion digits, turning it into a ∼130 TB big data object [51]. This is deceptive, as short programs generating its digits exist, and the Kolmogorov complexity of π is small in classic programming languages, and tiny in continued fraction form[9]. EML gives the following, of similar size, *exact finite expression* [52], without loops, conditionals, quantifiers or limits:

```
eml(eml(eml(1,eml(eml(1,eml(1,eml(eml(1,eml(eml(1,eml(eml(1,eml(1,
eml(eml(1,1),1))),1)),eml(eml(eml(eml(eml(1,eml(eml(1,eml(1,eml(eml(1,
eml(1,eml(eml(1,eml(eml(1,eml(eml(1,eml(1,eml(eml(1,1),1))),1)),eml(1,1))),
1))),1))),1)),eml(eml(eml(1,eml(eml(1,eml(1,eml(eml(1,1),1))),1)),
eml(eml(1,eml(eml(1,eml(eml(eml(1,eml(eml(1,eml(1,eml(eml(1,1),1))),1)),
eml(eml(1,eml(eml(1,eml(eml(1,eml(eml(1,1),1)),eml(eml(eml(1,eml(eml(1,
eml(1,eml(eml(1,1),1))),1)),eml(1,1)),1))),1)),1)),1)),1)),1)),1)),1),1),
1))),1))),1)),eml(eml(eml(1,eml(eml(1,eml(1,eml(eml(1,1),1))),1)),
eml(eml(1,eml(eml(1,eml(1,eml(eml(eml(1,eml(eml(eml(1,eml(1,eml(eml(1,1),
1))),1)),eml(1,1))),1))),1)),1)),1)),1)
```

RPN complexity for π in EML form is $53 < K \leq 193$, where the upper limit is from the above formula generated by the EML compiler, and the lower limit by exhaustive search [1]. In contrast, for Euler's γ the decimal/programming perspective is the same, while no finite EML expression is known. In fact, it is not even proved that γ is irrational.

This is confusing for many computer scientists, who reason in the following way. First, we need a computationally intensive binary algorithm for the evaluation of e^x, then we define (1) to compute ... e^x again. This looks circular. It makes no sense until we have a method to evaluate e^x directly. Physics, however, provides exponentials directly in numerous processes: light attenuation, exponential growth of bacteria, RC circuit, radioactivity $T_{1/2} = \tau \ln 2$, statistic $e^{-\frac{E}{k_B T}}$, entropy $S = k_B \log W$, cosmological inflation $e^{H_\infty t}$, the Gaussian curve e^{-x^2}, quantum phase $e^{i\varphi}|\psi\rangle$, and, last but not least, the exponential current-voltage law of the p–n junction. Most of these are models of nature, not usable devices. Therefore, EML might be a better tool for discovering natural or empirical laws than a computational device *cf.* the symbolic regression application in Section 1. Junction law is the exception, and was used to simulate the first EML hardware [55].

[9] For example, $\pi = 4/(2n+1+(n+1)^2/...), n = 0, 1, \ldots$, is ready to use in *continued fraction arithmetic* [53]. Vuillemin later addressed the difficulties with general real inputs using continued fractions with signed terms and more than one representation of the same number [54].

A SkyWater 130 nm layout of a test chip evaluates (1) with 4–134 times less energy than a digital circuit of matched precision [55]. But connecting cells to form EML trees costs so much that the assembled fabric loses to existing digital solutions. The existing digital architecture, in turn, is unsuitable for EML computations. An FPGA implementation may be useful in specific conditions [56], namely limited resources, *e.g.* on micro satellites, but still must use a standard discrete algorithm to compute e^x. For efficiency, we need some unconventional, non-von Neumann architecture. A step in the right direction is provided *e.g.* by [57].

2.7. Prior knowledge

While the discovery of a simple analytical single binary operator for the computation of elementary functions was a true surprise, and even the top AI LLM models trained with a cutoff before April 2026[10] would convincingly state that this was impossible, some hints of its existence were hidden in plain sight long before that.

Mathematicians attempted to answer related questions. In rational function theory, it was probably known [58] that rational functions can be generated over the field of rational numbers $\mathbb{Q}$ using a single operator

$$x\#y \equiv \frac{1}{x-y} \tag{4}$$

plus an unclear number of terminal constants. We found that this is indeed possible using both 0 and 1 as distinguished constants[11]. In this sense, for rational functions with integer coefficients, *e.g.*, $11(x^2+x+3)/(x^4-4)$, the "hash operator" (4) is indeed an analogue of the EML operator. However, it is not clear how to extend (4) to exp–log functions in any other way than by extending the hash operator with $\exp(x)$ and $\ln(x)$ themselves. But if we allow for this, then plain subtraction is already good enough, *cf.* Table 1 or Calc 2 from Table 2 in [1]. Moreover, no distinguished terminal constant is required. The statement that the "hash" operator is equivalent to EML is simply false.

Another piece of knowledge missed by [1] is Hua's identity [59], valid for any *division ring*, hence also for real/complex numbers. For real/complex numbers, Hua's identity allows one to express multiplication using addition/subtraction and inverse

$$aba = a - \left[a^{-1} + \left(b^{-1} - a\right)^{-1}\right]^{-1} . \tag{5}$$

[10] Such knowledge was not in their training data, and was not an obvious consequence of it.

[11] In fact, the constant 1 can be replaced by any non-zero rational.

The Kolmogorov-style RPN complexity of the above formula is too large for exhaustive search, unless we allow for the use of the reduced mass $u||v = (u^{-1} + v^{-1})^{-1}$ (parallel sum) as a primitive arithmetic operation. Then the reciprocal can be computed using $a^{-1} = 1 - [1||(a-1)]$, identity (5) gives multiplication, and the EML reduction of the operator count using the constant 1 sounds less mysterious.

In the abstract theory of binary operators on infinite sets, a construction reducing any finite or countable collection of binary operators to a single one is known [60]. This is an example of *clone theory* [61], which studies operations built by repeatedly applying other operations. Here, we present an example construction. The reader is encouraged to compare it to the elegance of the EML operator. In the following, we reduce the four basic operations $(+, -, \times, /)$ to a single "star" operator using four distinguished integer constants: $-1, -2, -3, -4$. First, we create four "compactified" copies of the real line

$$q_k(x) = 10 + k + \frac{1}{2} + \frac{\arctan x}{\pi}\,.$$

The inverse operation is, of course,

$$Q_k(z) = \tan\left[\pi\,(z - 10 - k - 1/2)\right].$$

Now, we define a single operator as

$$x * y = \begin{cases} q_0(y) & \text{for } x = -1\,, \\ q_1(y) & \text{for } x = -2\,, \\ q_2(y) & \text{for } x = -3\,, \\ q_3(y) & \text{for } x = -4\,, \\ Q_0(x) + y & \text{for } 10 < x < 11\,, \\ Q_1(x) - y & \text{for } 11 < x < 12\,, \\ Q_2(x) \times y & \text{for } 12 < x < 13\,, \\ Q_3(x)/y & \text{for } 13 < x < 14\,. \end{cases} \tag{6}$$

It is a simple exercise to verify that we have

$$\begin{aligned} x + y &= [(-1) * x] * y\,, \\ x - y &= [(-2) * x] * y\,, \\ x \times y &= [(-3) * x] * y\,, \\ \frac{x}{y} &= [(-4) * x] * y\,. \end{aligned}$$

Because $[(-1) * (-1)] * (-1) = -2$ *etc.*, one can reduce the number of terminal constants to one. Moreover, we have verified in Mathematica that another implementation of the [60] idea leads to a single "diamond" operator working without any constants, see Appendix A.

While the procedure (6) works for binary operators only, since arbitrary arity reduces to binary by a classical theorem of Sierpiński [62], one can extend it to unary exp–log using dummy operators, *e.g.*, $\mathrm{pwr}(x,y)=\exp(x)$ and $\mathrm{lg}(x,y)=\ln x$. This affirmatively answers the question raised by [1] in the set-theoretic reading. But not for analytic operators [15].

In this sense, [60] anticipated the existence of a single operator. But a Goldstern-type operator is essentially a set-theoretic multiplexer using a lookup table (see Appendix A). It is more similar to the Kolmogorov–Arnold bypass of Hilbert's 13^{th} problem, allowing for the reduction of any continuous function of n variables to $2n+1$ univariate ones using an f-dependent "lookup function" and addition, in its modern form [63]

$$f(x_1,\ldots,x_n)=\sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n}\lambda_p\phi_q(x_p)\right).$$

Neither is useful in practice due to pathological mathematical properties. Function $g(x)$ is f-dependent, only continuous, and highly irregular even for analytic f, and the inner functions cannot be chosen smooth. This is in contrast to EML-type operators, which are simple analytical formulas.

3. Diversity of operators

3.1. EML variants

Since the original discovery of (1), it was quickly realized that at least two similar close cousins exist. The first is EDL (Exp–Divide–Log)

$$\mathrm{edl}(x,y)=\frac{\mathrm{e}^x}{\ln y}, \tag{7}$$

paired with the Euler number e. Another one is −EML

$$-\mathrm{eml}(y,x)=\ln(x)-\exp(y), \tag{8}$$

paired with $-\infty$. Once a faster sieve implementation became available, it become clear that the above three are not, like initially believed, related by some Möbius transform. Two new variants of the EDL, LDE (Log–Divide–Exp) were found

$$\mathrm{lde}(x,y)=\frac{\ln x}{\mathrm{e}^y}, \tag{9}$$

one working with 0, the other with 1.

Other variants are PLI (Power–Log–Inverse)

$$\mathrm{pli}(x,y) = (\ln x)^{1/y}\,, \tag{10}$$

and PLM (Power–Log–Minus)

$$\mathrm{plm}(x,y) = (\ln x)^{-y}\,, \tag{11}$$

both with the constant 1.

Related constructions of arithmetic operations through bijections occur in non-Newtonian calculus [64]. In [15], Stachowiak proposed a general scheme for generating EML-type operators. Here, we repeat his definitions

$$S(x,y) = M\left(f(x), f^{-1}(y)\right)\,, \tag{12}$$

$$\begin{aligned} \exists_{\hat{e}} M(x,\hat{e}) &= x\,, && \text{(13a)}\\ M(x,x) &= \hat{e}\,, && \text{(13b)}\\ M(x,M(y,z)) &= M(z,M(y,x))\,, && \text{(13c)} \end{aligned}$$

where $S(x,y)$ is an EML-type binary operator, $f(x)$ is a generating function, $M(x,y)$ is a non-commutative operator, $\hat{e}$ is a neutral element, and $c = f(\hat{e})$ is a distinguished constant. Noteworthy, all known operators can be cast in Stachowiak's form, *cf.* Table 1. It is however an open problem what kind of pair M, f, which defines an EML-type operator, is allowed. Stachowiak discussed the example of $f(x) = \cos x$, but was unable to complete the entire elementary function chain. Our brute-force search depth was too shallow to improve significantly on that. However, using *prosthaphaeresis*, one can compute multiplication using *two* constants, $c = 1$ and $a = \arccos(1/4)$ with the same $S = \cos x - \arccos y$. Reciprocal/division is still unreachable, though.

Table 1. Known EML-type operators in Stachowiak's form.

Name	$M(x,y)$	$\hat{e}$	$c = f(\hat{e})$	f	f^{-1}	$S(x,y)$
EML	$x - y$	0	1	$\exp x$	$\ln x$	$\exp(x) - \ln(y)$
−EML	$x - y$	0	$-\infty$	$\ln x$	$\exp x$	$\ln(x) - \exp(y)$
EDL	x/y	1	e	$\exp x$	$\ln x$	$\exp(x)/\ln(y)$
LDE	x/y	1	0	$\ln x$	$\exp x$	$\ln(x)/\exp(y)$
PLI	$x^{(1/\ln y)}$	e	1	$\ln x$	$\exp x$	$(\ln x)^{1/y}$
PLM	$x^{(1/\ln y)}$	e	1	$\frac{1}{\ln x}$	$\mathrm{e}^{1/x}$	$(1/\ln x)^{y}$

The operator $M(x,y)$ is in all cases from Table 1 the inverse of a Bennett's symmetric (commutative) hyperoperation [65]

$$F_n(a,b) = \exp^n(\ln^n a + \ln^n b)$$

of increasing order:

- $n = 0$ addition/subtraction $F_0(x,y) = x + y \to x - y$ (1^{st} order hyperoperation, *i.e.* repeated zeration/successor);
- $n = 1$ multiplication/division $F_1(x,y) = x \times y \to \frac{x}{y}$ (2^{nd} order hyperoperation, *i.e.* repeated addition);
- $n = 2$ symmetrized power $F_2(x,y) = x^{\ln y} \to x^{1/\ln y}$ (commutative analog of the 3^{rd} order hyperoperation, *i.e.* repeated multiplication).

We must conclude that while the framework proposed by [15] gives some order and insight into the structure of EML-type operators, we still lack a general understanding, and exhaustive or fine-tuned search is still the only viable method to find them.

3.2. Ternary variants

The requirement for a distinguished constant at the inputs of EML and its variants is troublesome for the implementation of the master formula (3) and its use for machine learning or symbolic regression. The absence of a constant is also crucial for analog implementations. Physically, a constant is a precision reference, and maintaining it (Weston cells, cryogenic Josephson voltage standards [66]) is among the most demanding tasks of electrical metrology. Zero (ground) is the only inexpensive constant.

However, the use of this form of the "switch" seems unavoidable [58, 60] (see also Subsection 2.7). A workaround is to use a ternary operator instead. So far, two [1] were found using the same enumerative scheme as the one used to find EML

$$T_1(x,y,z) = \frac{\mathrm{e}^x}{\mathrm{e}^y} \times \frac{\ln x}{\ln z}\,, \tag{14a}$$

$$T_2(x,y,z) = \frac{\mathrm{e}^x}{\mathrm{e}^y} \times \frac{\ln z}{\ln x}\,. \tag{14b}$$

$T_i(x,x,x) = 1$, and no distinguished constant is required to generate all expressions. The interpolating ternary of [15] turns any pair of binary operators S_1, S_2 into a ternary one. Using $S_1 = c$ and $S_2 = S$, we can turn all entries of Table 1 into ternary form. Therefore, they are in fact plentiful. Stachowiak's ternary is engineered to "smuggle" an IF statement in the spirit of the set-theoretic constructions of Subsection 2.7, while in (14), it is hidden in multiplicative form.

4. Möbius layer neural networks

Standard feedforward neural networks use a single univariate real activation function. The essential question is whether we can find a similar function, which not only approximates data, but is able to generate elementary expressions in exact form as well. Such a construction could connect trainable networks with symbolic regression. Therefore, the answer to the above question is very important for progress in machine learning.

The EML operator (1) might be viewed as some variant of a 2-input complex-valued activation function, in analogy to sigmoid or ReLU in other variants of computational networks. For EML, the network graph is a parameterized binary tree. As demonstrated in [1], training such a tree is difficult, and so far has been demonstrated in a limited capacity. Its unique property, however, is the ability to express all elementary functions, which have proved useful in STEM over the last centuries. On the opposite side, we have machine learning, in the form of deep neural networks. While networks with sigmoid or ReLU activations cannot express functions like $\sin x$ exactly in the full real domain, they have proven (both in theory and practice) the ability to approximate any of them. But the decisive property favoring the latter in applications is a working optimization procedure (*e.g.* Adam, stochastic gradient method) which can be extended to industrial-scale networks with trillions of parameters as of 2026, with no "wall" in sight preventing further increase.

Therefore, the natural follow-up question, urgent after the discovery of the EML, is whether some of its variants will exhibit the scaling properties of modern deep neural networks, keeping "symbolic" capabilities of the EML-trees. Due to the aforementioned theorem by Hardy (see Subsection 2.2), a neural network with a real exp–log activation is unable to compute *e.g.* $\sin x$. What is the minimal modification then, which would enable exact elementary functions in deep learning, keeping a convenient optimization landscape? Below, we provide some hints as to which direction the search should proceed.

A typical neural network rewritten in EML-style language is a "calculator" with basic arithmetic operators $(+, -, \times)$ ("matrix multiplication") plus a single univariate non-linear activation function, *e.g.* the logistic sigmoid

$$S(x) = \frac{1}{1 + \mathrm{e}^{-x}}\,, \tag{15}$$

known in physics as the Fermi–Dirac distribution. Noteworthy, the set of allowed operations *does not* include division. We now attempt to modify the standard neural network, shown in Fig. 1, top panel, to enable exact elementary functions.

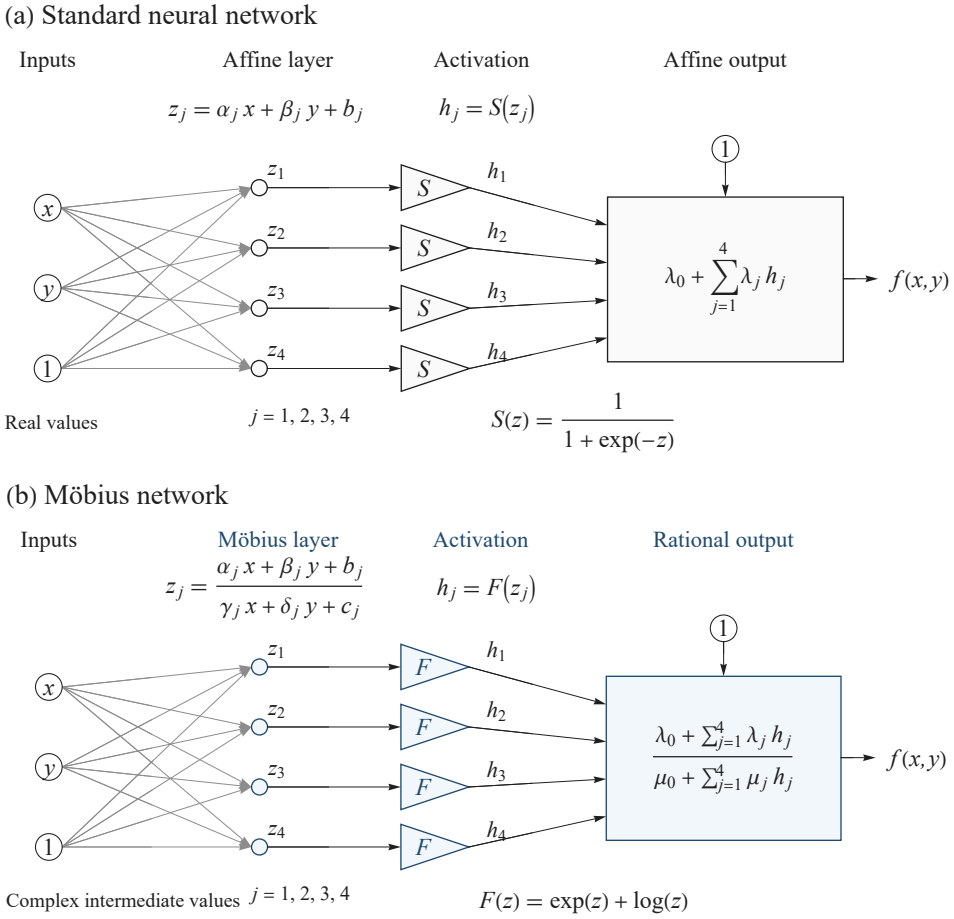


Fig. 1. Comparison of the standard neural architecture (top) and the proposed generalized "Möbius" network (bottom). Both include constant inputs supplying the biases. In the proposed network, the affine input and output maps become ratios of affine forms, and S is replaced by F. Each panel shows one layer with four activation units.

To achieve this goal, we employ three modifications: *(i)* replace the activation function, *(ii)* use complex numbers internally, and *(iii)* add division to the allowed arithmetic operations. The new (complex valued) activation function is

$$F(z) = \mathrm{e}^z + \ln z \overset{\text{a.e.}}{=} \mathrm{eml}\left(z, z^{-1}\right) . \tag{16}$$

The crucial identity

$$\mathrm{e}^z = \frac{F(3z) - F(z) - \ln 3}{F(2z) - F(z) - \ln 2} - 1 \tag{17}$$

allows us to recover the exponential function by standard algebraic simplification. The logarithm is then simply $\ln z = F(z) - \mathrm{e}^z$. Once we have both exp and ln, together with subtraction, we have (1), and from [1] we know that this is enough to evaluate all elementary functions. The constants used in (17), namely $1, 2, 3, \ln 2, \ln 3$, are needed as written but are likely redundant. The complexity of (17) in RPN form is $K = 21$, far beyond direct enumeration reach, and it is not known whether it is the simplest possible construction of this kind. Nevertheless, compositions of rational operations and F can, in principle, generate exact analytical formulas; Fig. 1, bottom panel, shows the basic layer. The matrix multiplication layer, including the output map, must be replaced by a rational function layer to allow the division required by (17), and complex numbers are required to use the EML-style reduction. The activation function is now (16). The question whether such a network can indeed be optimized efficiently is beyond the scope of this note.

5. Conclusions

The article discusses common misconceptions, previous knowledge, follow-up studies and emerging directions which came after the discovery of the EML operator [1]. What was missed by readers is that the EML itself, given by formula (1) is probably only the first known member of a large family of operators and functions, which could open new ways of evaluating elementary functions, enhancing the abilities of machine learning. So far, no systematic theory has been created or exhaustive search done to reveal all its variants. One of the undiscovered variants could possibly enable exact analytical functions in an architecture nearly identical to an existing neural network, keeping standard optimization procedures. In the optimistic variant, the only modification would be complex weights in place of real ones, and a new non-linear activation possibly resembling (16) in an algebraic way. If we are lucky, the new activation will work in the full real domain, keeping the asymptotics of the ramp function (ReLU), *i.e.* zero for $x \to -\infty$ and x for $x \to +\infty$. All EML-type nonlinearity would remain near zero and in imaginary direction. A natural candidate with the above properties is some combination of elementary hyperbolic functions. However, if rational functions are indeed required to achieve the stated goal, it would be no surprise to practitioners, as *e.g.* Padé approximation has been known to be a superior method for a long time.

I would like to thank Henrik Klagges for the invitation to the TNG Big Techday conference in Munich, and the Faculty of Mathematics and Computer Science of the Jagiellonian University for support in the form of

a Maple license. Computational resources were partially provided by Google Cloud Research Credits and the Polish National Science Centre MAESTRO grant No. 2017/26/A/ST2/00530. The author thanks the anonymous referee for valuable comments, and Anthropic for providing access to recent AI models through its Claude Team plan for scientists.

Appendix A

Wolfram Mathematica implementation of a Goldstern-type single operator

Let $\beta\colon \mathbb{Z} \to \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$, be the standard pairing between integers and naturals

$$\beta(m) = \begin{cases} 2m & m \geq 0\,, \\ -2m-1 & m < 0\,, \end{cases} \qquad \beta^{-1}(p) = \begin{cases} p/2 & p \text{ even}\,, \\ -(p+1)/2 & p \text{ odd}\,. \end{cases} \tag{A.1a}$$

Every positive integer factors uniquely as $2^l(2a+1)$. Applying this to $\beta(\lfloor x \rfloor) + 1$ assigns to every real x a "floor number" $l(x) \in \mathbb{N}$ and a "payload" $u(x) \in \mathbb{R}$

$$\beta(\lfloor x \rfloor) + 1 = 2^{l(x)}(2a+1)\,, \qquad u(x) = \beta^{-1}(a) + x - \lfloor x \rfloor\,. \tag{A.1b}$$

The map $x \mapsto \big(u(x), l(x)\big)$ is an explicit bijection $\mathbb{R} \to \mathbb{R} \times \mathbb{N}$, with inverse

$$\rho(v, l) = \beta^{-1}\Big[\, 2^l\big(2\beta(\lfloor v \rfloor) + 1\big) - 1 \,\Big] + v - \lfloor v \rfloor\,, \tag{A.1c}$$

and $S(x) = \rho\big(u(x), l(x) + 1\big)$ moves x one floor up, keeping the payload. The operator is

$$x \diamond y = \begin{cases} S(x) & \text{for } \; y = x\,, \\ \rho(x, 0) & \text{for } \; y = S(x)\,, \\ u(x) + u(y) & \text{for } \; l(x) = 1,\; l(y) = 0\,, \\ u(x) - u(y) & \text{for } \; l(x) = 2,\; l(y) = 0\,, \\ u(x) \times u(y) & \text{for } \; l(x) = 3,\; l(y) = 0\,, \\ u(x)/u(y) & \text{for } \; l(x) = 4,\; l(y) = 0\,,\; u(y) \neq 0\,, \\ x & \text{otherwise}\,. \end{cases} \tag{A.1d}$$

Writing $\sigma(t) = t \diamond t$ (one floor up) and $\varepsilon(x) = x \diamond (x \diamond x)$ (encoding at floor zero), we obtain

$$\begin{aligned} x + y &= \sigma\big(\varepsilon(x)\big) \diamond \varepsilon(y)\,, \\ x - y &= \sigma^2\big(\varepsilon(x)\big) \diamond \varepsilon(y)\,, \\ x \times y &= \sigma^3\big(\varepsilon(x)\big) \diamond \varepsilon(y)\,, \\ \frac{x}{y} &= \sigma^4\big(\varepsilon(x)\big) \diamond \varepsilon(y)\,, \end{aligned}$$

valid for all real x, y (with $y \neq 0$ in the last line). No constant appears anywhere; fully expanded, *e.g.*,

$$x + y = \Big\{ \big[\, x \diamond (x \diamond x) \,\big] \diamond \big[\, x \diamond (x \diamond x) \,\big] \Big\} \diamond \big[\, y \diamond (y \diamond y) \,\big] \,.$$

Constants can also be generated from pure-$\diamond$ terms, *e.g.*

$$0 = \left\{ \Big(\big[\, x \diamond (x \diamond x) \,\big] \diamond \big[\, x \diamond (x \diamond x) \,\big] \Big) \diamond \Big(\big[\, x \diamond (x \diamond x) \,\big] \diamond \big[\, x \diamond (x \diamond x) \,\big] \Big) \right\} \diamond \big[\, x \diamond (x \diamond x) \,\big] \,.$$

Below, you can find a working implementation of the above procedure in Wolfram Mathematica.

```
(* Pairing integers to naturals *)
beta[m_Integer]  := If[m >= 0, 2 m, -2 m - 1];
ibeta[p_Integer] := If[EvenQ[p], p/2, -(p + 1)/2];

(* Floor number and payload: x <-> (val, level) is a bijection
    R <-> R x N *)
level[x_] := IntegerExponent[beta[Floor[x]] + 1, 2];
val[x_]   := Module[{p = beta[Floor[x]] + 1, l},
               l = IntegerExponent[p, 2];
               ibeta[(p/2^l - 1)/2] + (x - Floor[x])];
rho[v_, l_Integer] := ibeta[2^l (2 beta[Floor[v]] + 1) - 1] + (
    v - Floor[v]);
S[x_] := rho[val[x], level[x] + 1];     (* same payload, one
    floor up *)

(* THE single operator *)
op[x_, y_] := Which[
   y == x,    S[x],                     (* climb *)
   y == S[x], rho[x, 0],                (* encode *)
   1 <= level[x] <= 4 && level[y] == 0,
     Switch[level[x],
       1, val[x] + val[y],
       2, val[x] - val[y],
       3, val[x] val[y],
       4, If[val[y] == 0, x, val[x]/val[y]]],
   True, x];

(* Pure-op terms for the four operations *)
s[x_]   := op[x, x];
enc[x_] := op[x, s[x]];
plus[x_, y_]   := op[s[enc[x]], enc[y]];
minus[x_, y_]  := op[s[s[enc[x]]], enc[y]];
times[x_, y_]  := op[s[s[s[enc[x]]]], enc[y]];
divide[x_, y_] := op[s[s[s[s[enc[x]]]]], enc[y]];
```

```
(* Examples *)

xplusy=op[op[op[x, op[x, x]], op[x, op[x, x]]], op[y, op[y, y
    ]]] (* x+y *)
zero=op[op[op[op[x, op[x, x]], op[x, op[x, x]]], op[op[x, op[x,
     x]], op[x, op[x, x]]]],
op[x, op[x, x]]] (* zero *)
(* Some values are required to resolve conditionals *)
xplusy /. {x->EulerGamma, y->Glaisher}
zero /. x->Khinchin
```

REFERENCES

[1] A. Odrzywołek, «All Elementary Functions from a Single Binary Operator», arXiv:2603.21852 [cs.SC], submitted to JACM, 2026.

[2] B. Naskręcki, «EML formalization», https://github.com/nasqret/eml-formalization, 2026.

[3] E.W. Weisstein, «EML Operator», from MathWorld — A Wolfram Web Resource, https://mathworld.wolfram.com/EMLOperator.html; accessed August 2026.

[4] H.M. Sheffer, «A set of five independent postulates for Boolean algebras, with application to logical constants», *Trans. Amer. Math. Soc.* **14**, 481 (1913).

[5] M. Cranmer *et al.*, astroautomata/PySR: v2.7.2, October 18, 2026.

[6] A.M. Alaa, M. van der Schaar, «Demystifying Black-Box Models with Symbolic Metamodels», in: «Advances in Neural Information Processing Systems», volume 32, *Curran Associates, Inc.*, 2019, pp. 11304–11314.

[7] J. Crabbé, Y. Zhang, W.R. Zame, M. van der Schaar, «Learning Outside the Black-Box: The Pursuit of Interpretable Models», in: «Advances in Neural Information Processing Systems», volume 33, *Curran Associates, Inc.*, 2020, pp. 17838–17849.

[8] K. Sroka, A. Odrzywołek, «Constant recognition», https://constantrecognizer.com/, 2026.

[9] C. Gupta, T.J. LaGrow, «Architecture-Induced Recoverability Bias in Differentiable Symbolic Regression», arXiv:2604.23256 [cs.NE].

[10] R. Belaiche, «Additive Atomic Forests for Symbolic Function and Antiderivative Discovery», arXiv:2605.08130 [cs.LG].

[11] A. Erez, «Non-Monotone Response Modules and Cascades from the EML Operator for Reduced Models of Biological Dynamics», arXiv:2605.02972 [math.DS].

[12] S. Asanuma, «EML-CD: Causal Mechanism Recovery via EML Symbolic Trees in Structure Learning», arXiv:2606.05942 [stat.ML].

[13] J. Germany, E. Abdo, J. Bakarji, «EML Trees Are Universal Approximators», arXiv:2606.23179 [cs.LG].

[14] Independent implementations of the EML operator:
OxiEML (Rust): https://github.com/cool-japan/oxieml;
eml (Python): https://github.com/Inknyto/eml;
monogate (JavaScript): https://github.com/agent-maestro/monogate;
emlvm: https://github.com/nullwiz/emlvm; accessed 29 August 2026.

[15] T. Stachowiak, «Algebraic structure behind Odrzywołek's EML operator», arXiv:2604.23893 [math-ph].

[16] R.W. Howell, J.H. Mathews, «Complex Analysis», 2025, online textbook, last updated August 22, 2025; accessed August 28, 2026.

[17] V. Mityushev, S. Rogosin, «On Relations Between Elliptic and Elementary Functions», *Acta Phys. Pol. B Proc. Suppl.* **13**, 745 (2020).

[18] M. Rosenlicht, «Integration in Finite Terms», *Amer. Math. Mon.* **79**, 963 (1972).

[19] A.R. Magid, «Lectures on Differential Galois Theory», in: «University Lecture Series», volume 7, *American Mathematical Society*, Providence, RI 1994.

[20] M.F. Singer, «Comments on Rosenlicht's Integration in Finite Terms», in: C.G. Raab, M.F. Singer (Eds.) «Integration in Finite Terms: Fundamental Sources, Texts & Monographs in Symbolic Computation», *Springer, Cham*, 2022, pp. 11–30. The volume reprints four foundational texts of the theory of integration in finite terms (Ritt 1948, Risch 1968, Rosenlicht 1972, Trager 1984).

[21] F. Beukers, «Hypergeometric functions, how special are they?», *Notices Amer. Math. Soc.* **61**, 48 (2014).

[22] M. Carney, «Inexpressibility in Exp-Minus-Log», arXiv:2605.01636 [math.LO].

[23] R. Smith, «Not all elementary functions can be expressed with exp-minus-log», https://www.stylewarning.com/posts/not-all-elementary/, 2026.

[24] T.Y. Chow, «What Is a Closed-Form Number?», *Amer. Math. Mon.* **106**, 440 (1999).

[25] G.H. Hardy, «Properties of Logarithmico-Exponential Functions», *Proc. London Math. Soc.* **S2-10**, 54 (1912).

[26] G.H. Hardy, «Orders of Infinity: The 'Infinitärcalcül' of Paul du Bois-Reymond», in: «Cambridge Tracts in Mathematics and Mathematical Physics», Number 12, *Cambridge University Press*, Cambridge, UK 1910.

[27] A. Rieu, «Thread on the historical development of NOR and NAND as universal gates», X, June 2026, posted under the username `@superrieu` (in German).

[28] J.G.P. Nicod, «A Reduction in the Number of the Primitive Propositions of Logic», *Proc. Camb. Phil. Soc.* **19**, 32 (1917).

[29] E. Stamm, «Beitrag zur Algebra der Logik», *Monatsh. Math. Phys.* **22**, 137 (1911).

[30] J. Łukasiewicz, «Uwagi o aksyomacie Nicod'a i o „dedukcyi uogólniającej"», in: «Księga pamiątkowa Polskiego Towarzystwa Filozoficznego we Lwowie», *Polskie Towarzystwo Filozoficzne*, Lwów 1931, pp. 366–382 (in Polish).

[31] C.E. Shannon, «A symbolic analysis of relay and switching circuits», *Trans. Am. Inst. Electr. Eng.* **57**, 713 (1938).

[32] A.G. Bromley, «Charles Babbage's Analytical Engine, 1838», *IEEE Ann. Hist. Comput.* **4**, 196 (1982).

[33] L.J. Lander, T.R. Parkin, «Counterexample to Euler's conjecture on sums of like powers», *Bull. Amer. Math. Soc.* **72**, 1079 (1966).

[34] D.Ž. Đoković, «Williamson matrices of order $4n$ for $n = 33, 35, 39$», *Discrete Math.* **115**, 267 (1993).

[35] J.M. Borwein, D.H. Bailey, «Mathematics by Experiment: Plausible Reasoning in the 21st Century», 2nd ed., *A.K. Peters*, Wellesley, MA 2008.

[36] Z.-W. Sun, «Catalan's constant is irrational», arXiv:2609.04176 [math.GM] .

[37] A. Odrzywolek https://math.stackexchange.com/users/307694/andrzejodrzywolek, Integral $\int_{-1}^{1} \frac{1}{x}\sqrt{\frac{1+x}{1-x}} \ln\left(\frac{2\,x^2+2\,x+1}{2\,x^2-2\,x+1}\right) \mathrm{d}x$, Mathematics Stack Exchange question 1625336, https://math.stackexchange.com/q/1625336 (version: 2016-01-24).

[38] J. Alper *et al.*, The Leiden Declaration on Artificial Intelligence and Mathematics, June 2026, endorsed by the International Mathematical Union; featured endorsers include P. Scholze and T. Tao; over 3,700 signatories as of 2 September 2026.

[39] R.S. Sutton, «The bitter lesson», https://www.incompleteideas.net/IncIdeas/BitterLesson.html, essay published March 13, 2019; accessed August 28, 2026.

[40] Z. Liu *et al.*, «KAN: Kolmogorov–Arnold Networks», International Conference on Learning Representations (ICLR) 2025, arXiv:2404.19756 [cs.LG].

[41] OpenAI, «An OpenAI model has disproved a central conjecture in discrete geometry», https://openai.com/index/model-disproves-discrete-geometry-conjecture/, May 20, 2026.

[42] S. Gao, «Counterexamples to the Jacobian conjecture in dimensions greater than two», arXiv:2608.00222 [math.AG].

[43] P. Arathoon, G. Ball, M.D. Kvalheim, «The Maxwell Conjecture Is False», arXiv:2607.27197 [physics.class-ph].

[44] OpenAI, «Ten advances in mathematics and theoretical computer science», https://openai.com/index/ten-advances-in-mathematics/, August 1, 2026.

[45] M. Tegmark, «The Mathematical Universe», *Found. Phys.* **38**, 101 (2008).

[46] S. Wolfram, «A Class of Models with the Potential to Represent Fundamental Physics», *Complex Syst.* **29**, 107 (2020).

[47] V. Taelin, Higher Order Company, «HVM: a massively parallel, optimal functional runtime based on interaction combinators», GitHub repository, 2026; accessed September 2, 2026.

[48] S. Wolfram, «A New Kind of Science», *Wolfram Media*, Champaign, IL 2002, https://www.wolframscience.com/nks/

[49] O. Peters, «Generating integers using EML», Code Golf Stack Exchange, April 2026. Code-golf challenge posted April 13, 2026 under the username `orlp`; the complete list for $0 \le n \le 1000$ is given in the answer by A. Kaseorg; accessed August 28, 2026.

[50] J.-P. Delahaye, «Une calculatrice à deux touches», *Pour Sci.* **589**, 70 (2026).

[51] A. Razeto, N. Rossi, «Can π generate itself? A Monte Carlo analysis of 314 trillion digits», arXiv:2608.06438 [math.GM].

[52] M. Carvalho, «Creating Pi from a Single Binary Operator and the Constant 1», posted July 22, 2026; accessed August 28, 2026.

[53] M. Beeler, R.W. Gosper, R. Schroeppel, HAKMEM, AI Memo 239, MIT Artificial Intelligence Laboratory, February 1972, Item 101B: Continued Fraction Arithmetic.

[54] J.E. Vuillemin, «Exact real computer arithmetic with continued fractions», *IEEE Trans. Comput.* **39**, 1087 (1990).

[55] C. Teuscher, «Composability rather than computation sets the cost of an analog EML hardware fabric», arXiv:2609.17903 [cs.ET].

[56] A. Taylor, «MicroZed Chronicles: EML in FPGA», https://www.adiuvoengineering.com/post/microzed-chronicles-eml-in-fpga, blog post, April 22, 2026; accessed August 28, 2026.

[57] A. Jelinčič *et al.*, «An efficient probabilistic hardware architecture for diffusion-like models», *npj Unconv. Comput.* **3**, 30 (2026).

[58] J.D. Hamkins, «Can We Unify Addition and Multiplication into One Binary Operation? To What Extent Can We Find Universal Binary Operations?», MathOverflow, Question 57465; March 2011 accessed 28 August 2026.

[59] L.-K. Hua, «On the Automorphisms of a Sfield», *Proc. Natl. Acad. Sci. USA* **35**, 386 (1949).

[60] M. Goldstern, «A Single Binary Function Is Enough», in: J. Czermak *et al.* (Eds.) «Contributions to General Algebra 20: Proceedings of the Salzburg Workshop 2011 on General Algebra», *Verlag Johannes Heyn*, Klagenfurt, Austria 2012, pp. 35–37.

[61] M. Goldstern, M. Pinsker, «A survey of clones on infinite sets», *Algebra Univers.* **59**, 365 (2008).

[62] W. Sierpiński, «Sur les fonctions de plusieurs variables», *Fund. Math.* **33**, 169 (1945).

[63] D.A. Sprecher, «On the structure of continuous functions of several variables», *Trans. Amer. Math. Soc.* **115**, 340 (1965).

[64] M. Czachor, «Waves Along Fractal Coastlines: From Fractal Arithmetic to Wave Equations», *Acta Phys. Pol. B* **50**, 813 (2019).

[65] A.A. Bennett, «Note on an Operation of the Third Grade», *Ann. Math.* **17**, 74 (1915).

[66] C.A. Hamilton, «Josephson voltage standards», *Rev. Sci. Instrum.* **71**, 3611 (2000).