LegibleMath: Phonetically-Inspired Arithmetic System for Smart Contracts

banteg
Independent Researcher

April 1, 2025

Abstract

The increasing complexity of smart contracts necessitates robust and verifiable arithmetic operations. Current solutions, while functional, often lack intuitive readability, potentially hindering auditability and increasing the surface area for semantic errors. This paper introduces LegibleMath, a novel Solidity library employing a system where numerical values are represented by the multiplicative product of letters comprising their English spelling. We present the formal derivation of a consistent set of real-valued constants for the letters required to represent integers in the range [-11, 11]. We provide the general solution, demonstrate a concrete instantiation, and rigorously prove the inherent limitation of the system, specifically demonstrating the impossibility of extending it to represent the number twelve while maintaining consistency. This work establishes the mathematical foundation for LegibleMath, highlighting its unique approach to enhancing code clarity, albeit within a constrained numerical domain.

1 Introduction

The evolution of blockchain technology and decentralized applications has placed significant emphasis on the security and correctness of smart contracts [1]. Arithmetic operations are fundamental to most contract logic, and errors in these operations can lead to catastrophic financial losses. Libraries like SafeMath [2] were developed to mitigate integer overflow and underflow vulnerabilities, representing a crucial step forward in contract safety.

However, safety alone does not guarantee clarity. Complex arithmetic expressions, even when safe, can be difficult to parse and verify during code reviews and audits. LegibleMath proposes a radical alternative aimed at enhancing the

readability of numerical expressions within Solidity code. The core concept is to represent integer values through the multiplicative interaction of constants associated with the letters forming their standard English spellings (e.g., `t*w*o` evaluates to 2). This approach leverages human language familiarity to potentially make numerical assignments and simple calculations more intuitive.

This paper addresses the fundamental mathematical challenge underpinning LegibleMath: determining a consistent set of values for the required letter constants.

Definition 1.1 (LegibleMath Constraint)**.** Let L be the set of letters appearing in the English spellings of integers from zero to eleven, plus the word *negative*. We seek a mapping $\phi : L \to \mathbb{R} \setminus \{0\}$ (assigning a non-zero real number to each letter) such that for each target word $W = l_1 l_2 \dots l_k$ representing an integer v, the product $\prod_{i=1}^{k} \phi(l_i) = v$. The constant for the letter z is permitted to be zero.

The primary objectives of this work are:

1. To formulate the system of equations defined by the LegibleMath constraints for the integers 0 through 11 and the multiplier -1 (*negative*).
2. To derive the general solution for the letter constants satisfying this system.
3. To provide a concrete numerical example of these constants.
4. To analyze the limitations of this system, specifically proving its inability to consistently represent the number twelve.

The structure of this paper follows these objectives. Section 2 presents the system of equations. Section 3 details the algebraic derivation of the constants. Section 4 summarizes the general solution and provides a specific instance. Section 5 discusses the system's boundaries, including the formal proof regarding the number twelve. Section 6 concludes the findings.

2 Problem Formulation

The LegibleMath concept translates directly into a system of non-linear algebraic equations. We require the product of the letter constants corresponding to the spelling of each number (and *negative*) to equal its numerical value. This yields the following system:

Problem 1 (LegibleMath System)**.** Find real values for the variables $\{a, e, f, g, h, i, l, n, o, r, s, t, u, v, w, x, z\}$ satisfying:

$$z \cdot e \cdot r \cdot o = 0 \tag{1}$$
$$o \cdot n \cdot e = 1 \tag{2}$$
$$t \cdot w \cdot o = 2 \tag{3}$$
$$t \cdot h \cdot r \cdot e \cdot e = 3 \tag{4}$$
$$f \cdot o \cdot u \cdot r = 4 \tag{5}$$
$$f \cdot i \cdot v \cdot e = 5 \tag{6}$$
$$s \cdot i \cdot x = 6 \tag{7}$$
$$s \cdot e \cdot v \cdot e \cdot n = 7 \tag{8}$$
$$e \cdot i \cdot g \cdot h \cdot t = 8 \tag{9}$$
$$n \cdot i \cdot n \cdot e = 9 \tag{10}$$
$$t \cdot e \cdot n = 10 \tag{11}$$
$$e \cdot l \cdot e \cdot v \cdot e \cdot n = 11 \tag{12}$$
$$n \cdot e \cdot g \cdot a \cdot t \cdot i \cdot v \cdot e = -1 \tag{13}$$

We assume, based on the non-zero requirements for most equations, that all variables except potentially z must be non-zero.

3 Solution Derivation

We solve the system presented in Problem 1 step-by-step.

3.1 Determining the Value of z

From equation (1), $z \cdot e \cdot r \cdot o = 0$. From equation (2), $o \cdot n \cdot e = 1$, which implies $o \neq 0$ and $e \neq 0$. If we assume $r \neq 0$ (required for equation (4) or (5) to be non-zero), then the only possibility is:

$$z = 0$$

3.2 Establishing Dependencies and Free Parameters

The system is underdetermined, containing 13 equations and 17 variables (excluding z). We select a set of variables as free parameters, from which the others can be derived. A convenient choice, based on their appearance in early equations, is $\{o, e, r, v\}$. We assume these parameters are non-zero real numbers.

3.3 Deriving Dependent Variables

We proceed sequentially, expressing each remaining variable in terms of the free parameters or previously derived variables.

1. **Deriving n:** From (2), $o \cdot n \cdot e = 1 \implies n = \frac{1}{oe}$.
2. **Deriving t:** From (11), $t \cdot e \cdot n = 10$. Substituting $n = \frac{1}{oe}$, we get $t \cdot e \cdot (\frac{1}{oe}) = 10 \implies \frac{t}{o} = 10 \implies t = 10o$.
3. **Deriving w:** From (3), $t \cdot w \cdot o = 2$. Substituting $t = 10o$, we get $(10o) \cdot w \cdot o = 2 \implies 10o^2 w = 2 \implies w = \frac{2}{10o^2} = \frac{1}{5o^2}$.
4. **Deriving h:** From (4), $t \cdot h \cdot r \cdot e^2 = 3$. Substituting $t = 10o$, we get $(10o) \cdot h \cdot r \cdot e^2 = 3 \implies h = \frac{3}{10ore^2}$.
5. **Deriving i:** From (10), $n \cdot i \cdot n \cdot e = 9$. Substituting $n = \frac{1}{oe}$, we get $(\frac{1}{oe}) \cdot i \cdot (\frac{1}{oe}) \cdot e = 9 \implies \frac{i}{o^2 e} = 9 \implies i = 9o^2 e$.
6. **Deriving f:** From (6), $f \cdot i \cdot v \cdot e = 5$. Substituting $i = 9o^2 e$, we get $f \cdot (9o^2 e) \cdot v \cdot e = 5 \implies 9o^2 e^2 v f = 5 \implies f = \frac{5}{9o^2 e^2 v}$.
7. **Deriving u:** From (5), $f \cdot o \cdot u \cdot r = 4$. Substituting $f = \frac{5}{9o^2 e^2 v}$, we get $(\frac{5}{9o^2 e^2 v}) \cdot o \cdot u \cdot r = 4 \implies \frac{5or}{9o^2 e^2 v} u = 4 \implies u = \frac{4 \cdot 9o^2 e^2 v}{5or} = \frac{36oe^2 v}{5r}$.
8. **Deriving s:** From (8), $s \cdot e \cdot v \cdot e \cdot n = 7$. Substituting $n = \frac{1}{oe}$, we get $s \cdot e^2 \cdot v \cdot (\frac{1}{oe}) = 7 \implies \frac{sev}{o} = 7 \implies s = \frac{7o}{ev}$.
9. **Deriving x:** From (7), $s \cdot i \cdot x = 6$. Substituting $s = \frac{7o}{ev}$ and $i = 9o^2 e$, we get $(\frac{7o}{ev}) \cdot (9o^2 e) \cdot x = 6 \implies \frac{63o^3}{v} x = 6 \implies x = \frac{6v}{63o^3} = \frac{2v}{21o^3}$.
10. **Deriving g:** From (9), $e \cdot i \cdot g \cdot h \cdot t = 8$. Substituting $i = 9o^2 e$, $h = \frac{3}{10ore^2}$, $t = 10o$, we get $e \cdot (9o^2 e) \cdot g \cdot (\frac{3}{10ore^2}) \cdot (10o) = 8 \implies (9o^2 e^2) \cdot g \cdot (\frac{3}{re^2}) = 8 \implies \frac{27o^2}{r} g = 8 \implies g = \frac{8r}{27o^2}$.
11. **Deriving l:** From (12), $e \cdot l \cdot e \cdot v \cdot e \cdot n = 11$. Substituting $n = \frac{1}{oe}$, we get $e^3 \cdot l \cdot v \cdot (\frac{1}{oe}) = 11 \implies \frac{e^2 lv}{o} = 11 \implies l = \frac{11o}{e^2 v}$.
12. **Deriving a:** From (13), $n \cdot e \cdot g \cdot a \cdot t \cdot i \cdot v \cdot e = -1$. Substituting known values: $n = \frac{1}{oe}$, $g = \frac{8r}{27o^2}$, $t = 10o$, $i = 9o^2 e$, we get $(\frac{1}{oe}) \cdot e \cdot (\frac{8r}{27o^2}) \cdot a \cdot (10o) \cdot (9o^2 e) \cdot v \cdot e = -1$. Simplifying: $(\frac{1}{o}) \cdot (\frac{8r}{27o^2}) \cdot a \cdot (10o) \cdot (9o^2 e^2 v) = -1$. Further simplification: $(\frac{8r}{27o^3}) \cdot a \cdot (90o^3 e^2 v) = -1$. Combining terms: $\frac{8r \cdot 90 \cdot o^3 e^2 v}{27o^3} a = -1 \implies \frac{720re^2 v}{27} a = -1 \implies \frac{80re^2 v}{3} a = -1 \implies a = -\frac{3}{80re^2 v}$.

This completes the derivation of all dependent variables in terms of the free parameters $\{o, e, r, v\}$.

4 Results

4.1 General Solution

The general solution for the LegibleMath system (Problem 1), expressing each letter constant in terms of the non-zero free parameters o, e, r, v, is:

$$
\begin{aligned}
z &= 0 & n &= \frac{1}{oe} & t &= 10o \\
w &= \frac{1}{5o^2} & h &= \frac{3}{10ore^2} & i &= 9o^2e \\
f &= \frac{5}{9o^2e^2v} & u &= \frac{36oe^2v}{5r} & s &= \frac{7o}{ev} \\
x &= \frac{2v}{21o^3} & g &= \frac{8r}{27o^2} & l &= \frac{11o}{e^2v} \\
a &= -\frac{3}{80re^2v}
\end{aligned}
$$

The free parameters o, e, r, v can be any non-zero real numbers.

4.2 Concrete Instantiation

For practical implementation and verification, we can instantiate the general solution by choosing specific values for the free parameters. A simple choice is setting all free parameters to unity:

$$o = 1, \quad e = 1, \quad r = 1, \quad v = 1.$$

Substituting these values into the general solution yields the following concrete set of constants:

$$
\begin{aligned}
z &= 0 & o &= 1 & e &= 1 \\
r &= 1 & v &= 1 & n &= 1 \\
t &= 10 & w &= \frac{1}{5} & h &= \frac{3}{10} \\
i &= 9 & f &= \frac{5}{9} & u &= \frac{36}{5} \\
s &= 7 & x &= \frac{2}{21} & g &= \frac{8}{27} \\
l &= 11 & a &= -\frac{3}{80}
\end{aligned}
$$

This specific set of rational constants satisfies all equations (1) through (13).

5 Analysis and System Limitations

The derived solution provides a consistent mathematical basis for LegibleMath within its defined scope (integers -11 to 11). However, the structure of the system imposes inherent limitations, most notably concerning its extensibility.

5.1 Impossibility Proof for Extension to Twelve

A natural question is whether the LegibleMath system can be extended to include the number twelve, i.e., can we satisfy the additional constraint $t \cdot w \cdot e \cdot l \cdot v \cdot e = 12$ simultaneously with equations (1)-(13)?

Theorem 5.1. *Any solution $(\phi(l))_{l \in L}$ satisfying the LegibleMath system equations (1) through (13) necessarily yields $\phi(t)\phi(w)\phi(e)\phi(l)\phi(v)\phi(e) = 22$. Consequently, it is impossible to extend the system to include the constraint $t \cdot w \cdot e \cdot l \cdot v \cdot e = 12$.*

Proof. Let $(\phi(l))_{l \in L}$ be a valid solution satisfying equations (1) through (13). From the derivation in Section 3 (or by direct substitution using the general solution), we have established specific relationships between the constants. Consider the expression for "twelve": $t \cdot w \cdot e \cdot l \cdot v \cdot e$. We can group terms based on known sub-products or derived values:

- From equation (3), we know $t \cdot w \cdot o = 2$. Since $o \neq 0$, this implies $t \cdot w = 2/o$.
- From equation (12), we know $e \cdot l \cdot e \cdot v \cdot e \cdot n = 11$.

Let's use the derived values from the general solution (Section 4):

- $t = 10o$
- $w = \frac{1}{5o^2}$
- $l = \frac{11o}{e^2 v}$

Now, substitute these into the product for "twelve":

$$\begin{aligned} t \cdot w \cdot e \cdot l \cdot v \cdot e &= (t \cdot w) \cdot e^2 \cdot (l \cdot v) \\ &= \left((10o) \cdot \left(\frac{1}{5o^2} \right) \right) \cdot e^2 \cdot \left(\left(\frac{11o}{e^2 v} \right) \cdot v \right) \\ &= \left(\frac{10o}{5o^2} \right) \cdot e^2 \cdot \left(\frac{11ov}{e^2 v} \right) \\ &= \left(\frac{2}{o} \right) \cdot e^2 \cdot \left(\frac{11o}{e^2} \right) \\ &= \frac{2 \cdot e^2 \cdot 11 \cdot o}{o \cdot e^2} \\ &= 2 \cdot 11 \\ &= 22 \end{aligned}$$

The calculation shows that for *any* valid assignment of constants satisfying the original system (regardless of the choice of free parameters o, e, r, v, provided they are non-zero), the product corresponding to the spelling *twelve* inevitably evaluates to 22.

Since $22 \neq 12$, the equation $t \cdot w \cdot e \cdot l \cdot v \cdot e = 12$ is incompatible with the system defined by equations (1) through (13). Therefore, the LegibleMath system cannot be extended to include twelve. □

5.2 Other Considerations

- **Dependence on Language Spelling:** The system is inherently tied to the specific letter sequences of number words in a chosen language. It is not readily adaptable to other languages without a complete reformulation and resolution of the corresponding system of equations.

- **Choice of English and System Scope:** While the methodology could be applied to other languages, and computational methods confirm solvability for alternatives like French (e.g., for spellings *negatif*, *zero* through *dix*) [3], English was selected for the LegibleMath implementation. This choice prioritizes a language prevalent in smart contract development and, crucially, leverages an orthography whose resulting system of equations permits a consistent solution across the integer range [-11, 11], as demonstrated. The scope achievable may differ significantly for other languages.

- **Rational Constants:** The derived constants for the English system are rational numbers, not necessarily integers. While the LegibleMath Solidity

library manages this internally using fractional representations, it's a point of note regarding the mathematical nature of the solution.

- **Magnitude of Constants:** Depending on the choice of free parameters, individual constants can become very large or very small, potentially leading to precision issues in fixed-point implementations if not handled carefully (though the reference library uses rational numbers).

6 Conclusion

This paper presented the formal mathematical underpinning of the LegibleMath system, a novel approach to representing numerical values in Solidity smart contracts through phonetically-inspired letter constant products. We successfully formulated and solved the system of non-linear equations derived from the English spellings of integers in the range [-11, 11]. The general solution, dependent on four non-zero free parameters, was derived, and a concrete instantiation using rational constants was provided.

A key finding is the inherent limitation of the system's scope. We rigorously proved that the structure imposed by the equations for 0-11 and *negative* makes it mathematically impossible to consistently incorporate the number twelve, as the product $t \cdot w \cdot e \cdot l \cdot v \cdot e$ necessarily evaluates to 22 in any valid solution.

While LegibleMath offers a unique perspective on code readability within its constrained domain, its practical applicability is bound by this mathematical limitation and its dependence on English orthography. This work provides the necessary foundational validation for the LegibleMath library and clarifies its operational boundaries. Future investigations might explore similar systems based on different linguistic or symbolic representations.

References

[1] Solidity Team. (2024). *Solidity Documentation*. Accessed April 2025. `https://docs.soliditylang.org/`

[2] OpenZeppelin. (2017). *SafeMath Library*. Accessed April 2025. `https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/SafeMath.sol`

[3] Dr. Raffy.eth [@adraffy]. (2025). *here's a french (up to 10)* `https://x.com/adraffy/status/1913043428151275766`