On Machine Integers and Arithmetic

Pavel Loskot
ZJU-UIUC Institute
Haining, China
e-mail: *pavelloskot@intl.zju.edu.cn*

***Abstract*—All signal and data processing is performed on computing machines. However, the computing efficiency requires that numbers are represented in a finite memory space. It is claimed that all such numbers can be considered to be integers, and that decimal point has purely syntactical meaning to align numbers in arithmetic operations. This subtle, but fundamental observation seems to have been ignored so far. As an introductory exploration of integer arithmetic, this paper introduces a dual modulo operator to select digits in string representations of machine numbers. Moreover, it is proposed that natural integers offset by a real-valued constant satisfy Peano axioms. The Fermat last theorem is then considered as an example of Diophantine equation. It is shown how it can be modified to allow the solutions to exist. A Fermat metric is newly introduced to define distances between integers to allow their partitioning into subsets. These results point at the importance of investigating integer arithmetic, integer algebra, and integer analysis in designing and modeling computing systems.**

***Keywords*—dual modulo arithmetic; Fermat last theorem; Fermat metric; natural numbers.**

I. INTRODUCTION

Numbers are abstract mathematical objects that can also carry semantic meaning of quantity. The former leads to rich axiomatic algebraic systems, and the latter enables performing arithmetic operations in computing problems. Since computing machines have limited resources, and must execute numerical algorithms in a time and memory efficient manner, they have to represent numbers as constant size objects. This limits the largest and the smallest number values as well as precision, which can be considered. Therefore, any algorithm described or implemented in a programming language can only compute numbers from a finite set, $\mathcal{N} = \{N_1 < N_2 < \ldots\}$, such that,

$$\forall i: \; -\infty < \inf(\mathcal{N}) \leq N_i \leq \sup(\mathcal{N}) < \infty. \tag{1}$$

Thus, two machine numbers, N_i, and, N_j, can be compared, i.e., ordered, and the difference, $\min_{i\neq j}|N_i - N_j| = \varepsilon_0$, represents the precision. The set, $\mathcal{N}$, is necessarily computable [1].

Most machine number systems use floating point and fixed point number representations. These representations including basic arithmetic operations are precisely defined by the IEEE 754 standard [2]. They enable efficient utilization of hardware and software resources to achieve the time and space efficiency in implementing computing algorithms. Some languages (e.g., Python) support infinite-precision integer arithmetic, or perform computations at the user-defined precision (e.g., Mathematica). The GNU library [3] is a popular and efficient implementation in C programming language of the multi-precision arithmetic for integer and floating-point numbers; this library is also used in several commercial software products (e.g., Mathematica and Maple). The smallest and the largest integers and single and double precision floating point numbers are defined in Matlab toolbox, Elementary Matrices, and in the C standard libraries, limits.h and float.h.

The algorithms described in various programming languages represent numbers as strings of digits in a given basis. In particular, the number, $N \in \mathcal{N}$, in basis, B, is represented as,

$$\begin{aligned} N &= \sum_{i=i_{\min}}^{i_{\max}} D_i \times B^i \\ &\leftrightarrow \quad D_{i_{\max}} D_{i_{\max}-1} \cdots D_1 D_0 \,.\, D_{-1} \cdots D_{i_{\min}} \end{aligned} \tag{2}$$

where the digits, $D_i \in \{0,1,\ldots,9,\mathrm{A},\mathrm{B},\mathrm{C},\ldots B-1\}$, and the orders, $i_{\min} \leq 0 \leq i_{\max}$. It is customary to place decimal point between the digits D_0 and D_{-1}, which divides the digits into the integral and the fractional part, respectively. More importantly, the decimal point allows aligning digits of numbers when performing arithmetic operations and comparisons.

The most common bases are decimal ($B = 10$), hexadecimal ($B = 16$), and binary ($B = 2$). However, internally, the numbers are stored much more efficiently in a byte-size basis, i.e., $B = 2^{8\times\#\text{bytes}-1}$, with one bit reserved for the sign. The total number of bytes used for each number is usually fixed for different number classes (types) such as short and long integers, and single and double precision floating point numbers. The conversions between the string notation and the internal representation are performed automatically by compilers.

The textbook [2] provides a comprehensive overview of the number systems used on computers. The computability of functions of natural numbers is established in [4]. The mismatch between exact mathematical description and practical implementation of algorithms with approximate number representations has been studied in [5] including the methods how to mitigate such a discrepancy. The construction of large-scale real numbers, which are suitable for software implementations is considered in [6]. Binary approximations of real-numbers are investigated in [7]. Other representations of real numbers such as binary expansion, Dedekind cut and Cauchy sequence are compared in [1]. The p-adic number systems allow defining real-numbers as an arithmetic of rational numbers [8]. Logical statements involving comparisons of real-numbers are evaluated in satisfiability modulo theories [9]. The article [10] conclusively argues that a finite precision is usually sufficient in practical engineering applications. Many number-theoretic theorems and conjectures can be found in [11].

 ISBN: 978-1-68558-057-5

In this paper, it is argued that the number systems commonly used on computers can be assumed to be integer-valued, which also includes single and double precision floating point numbers and the corresponding arithmetic operations. Consequently, computing machines are inherently governed by integer algebras and arithmetic. The paper contributions are formulated as four claims and one proposition. In particular, in Section II, dual modulo operator is introduced to select digits in string representations of numbers. It can be exploited to define equivalences between numbers in integer arithmetic. In addition, it is proposed that natural numbers can be offset by a real-valued constant, and still be considered as being integers. In Section III, several modifications of Fermat last theorem (FLT) are devised to allow the solutions to exist. A Fermat metric is newly defined, which is then used to compute distances between natural numbers, and to divide the numbers into subsets. The paper is concluded in Section IV.

II. Machine Integers and Arithmetic

Constraining machine computations to numbers, $\mathcal{N}$, has several fundamental consequences. First, the results of arithmetic operations can overflow the limits, $\inf(\mathcal{N})$, or, $\sup(\mathcal{N})$. Second, the results of arithmetic operations can underflow the precision, ε_0, so the results may have to be truncated, rounded, or otherwise approximated. Third, the decimal point to align numbers can be arbitrarily placed in-between any digits as long as the placement is consistent in the number system and arithmetic used. This is formalized as the following claim.

Claim 1. *The machine numbers allocated a predefined memory space are integers, $\mathcal{N}$, isomorphic to a finite set of finite integers, $\mathcal{Z} = \{\ldots,-1,0,+1,\ldots\}$.*

The important consequence is that (without a formal proof) any machine arithmetic is isomorphic to integer arithmetic. However, implementing such integer arithmetic at large scale and precision to be efficient and also error-free is non-trivial.

The memory allocated by compilers of programming languages allows adding only a finite number of digits before and after a decimal point. If the numbers are padded by zero-digits from both ends, all numbers are represented by strings of the same length, and the decimal point becomes a hypothetical construct. The non-zero digits at the right end of the number string represent the precision (resolution), whereas the first non-zero digits from the left represent the scale.

The algorithms usually contain many logical statements (predicates). These statements involve comparisons of numerical values. The two integers are said to be exactly equal, provided that all digits in their string representations are the same. The exact comparison can be rather restrictive in some applications, where the differences in scale and precision could be or must be tolerated. Specifically, if the differences are tolerated in precision (the right-end sub-strings), it is equivalent to comparing quantized numbers. If the differences are tolerated at scale (the left-end sub-strings), it is equivalent to comparing periodically repeated values.

Mathematically, removing the right-end or the left-end sub-strings from the string representations of numbers can be expressed by a canonical modulo operator. In particular, for any integer a, and any positive integer b, let, $(a \bmod b) = (|a| \bmod b) \in \{0,1,\ldots,b-1\}$, to be a reminder after the integer division of a by b. Note that this can be readily extended to real numbers as, $0 \leq (a \bmod b) = (|a| \bmod b) < b$, assuming a real division of, $a \in \mathcal{R}$, by integer, b. Then, the numbers, a_1, and, a_2, are said to be equivalent in a sense of congruence, provided that, $a_1 \equiv a_2 \pmod b$. Both equality (indicated by symbol, $=$) and equivalence (indicated by symbol, $\equiv$) satisfy axiomatic properties of reflexivity, symmetry, and transitivity, and the equality implies equivalence.

If the machine numbers, $N_i = \sum_{i=0}^{L-1} D_i B^i$, are represented by strings of L digits in some basis B, then the first L_1 digits and the last L_2 digits, $(L_1 + L_2) < L$, can be zeroed by applying a dual modulo operator introduced next.

Definition 1. *The dual modulo operator has two parameters, m_1, and, m_2, and it is defined as the difference,*

$$\begin{aligned} N_i \ \mathrm{Mod}(m_1,m_2) &= (N_i \bmod m_1) - (N_i \bmod m_2) \\ &= \underbrace{0\cdots0}_{L_1} D_{L-L_1-1}\cdots D_{L_2+1} D_{L_2} \underbrace{0\cdots0}_{L_2}. \end{aligned} \tag{3}$$

where $m_1 = B^{L-L_1}$, and $m_2 = B^{L_2}$.

Modular arithmetic with dual modulo operator has similar properties as the arithmetic involving canonical modulo operator. In particular, given integers a, b, m_1, and m_2, then,

$$\begin{aligned} a \ \mathrm{Mod}(0,m_2) &= a - (a \bmod m_2) \\ a \ \mathrm{Mod}(m_1,1) &= a \bmod m_1 \\ a \ \mathrm{Mod}(m_1,m_1) &= 0. \end{aligned} \tag{4}$$

Furthermore, it is straightforward to prove that,

$$\begin{aligned} a+b &\equiv a \ \mathrm{Mod}(m_1,m_2) + b \ \mathrm{Mod}(m_1,m_2) \ (\mathrm{Mod}(m_1,m_2)) \\ a-b &\equiv a \ \mathrm{Mod}(m_1,m_2) - b \ \mathrm{Mod}(m_1,m_2) \ (\mathrm{Mod}(m_1,m_2)) \\ a\cdot b &\equiv a \ \mathrm{Mod}(m_1,m_2) \cdot b \ \mathrm{Mod}(m_1,m_2) \ (\mathrm{Mod}(m_1,m_2)). \end{aligned} \tag{5}$$

However, in general, for integer division with a reminder,

$$a/b \not\equiv a \ \mathrm{Mod}(m_1,m_2)/b \ \mathrm{Mod}(m_1,m_2) \ (\mathrm{Mod}(m_1,m_2)). \tag{6}$$

The Chinese reminder theorem [11] can be restated for dual modulo operator as follows. If m_{11} and m_{12} are co-prime, and,

$$\begin{aligned} N_i &\equiv a_1 \ (\mathrm{Mod}(m_{11},m_2)) \\ N_i &\equiv a_2 \ (\mathrm{Mod}(m_{12},m_2)) \end{aligned} \tag{7}$$

for some integers, N_i, and, m_2, then there is a unique integer, a, such that,

$$N_i \equiv a \ (\mathrm{Mod}(m_{11}m_{12},m_2)). \tag{8}$$

The proof is based on the property that, if $N_i \equiv a \pmod{m_1}$, then also, $N_i \equiv a \ (\mathrm{Mod}(m_1,m_2))$.

Furthermore, it is useful to consider how the machine integers used in algorithms are approximations of infinite precision real-numbers obtained from mathematical analysis.

 ISBN: 978-1-68558-057-5

The dual modulo operator defined in (3) produces a finite-size integer, x Mod(m_1,m_2), from a real number, $x \in \mathcal{R}$. This introduces a periodicity due to truncation from the left (specified by the parameter, m_1), and the quantization due to truncation from the right (specified by the parameter, m_2).

It is also useful to define countably infinite integer sets,

$$\tilde{\mathbb{N}}_x = \{x, x+1, x+2, \ldots\} \tag{9}$$

parameterized by finite constants, $x \in \mathcal{R}$, so that, $\tilde{\mathbb{N}}_0$, is the set of natural numbers. Such integer sets can provide exact solutions to some integer (Diophantine) problems, which otherwise do not have any solution. More importantly, for all finite x, the integers, $\tilde{\mathbb{N}}_x$, satisfy Peano axioms [11].

III. Case Study: FLT Problems

The FLT states that there are no positive integers a, b, c, and $n > 2$, such that, $a^n + b^n = c^n$. This has been first verified numerically until the proof was obtained only recently [11]. Note also that, $|a^n + b^n - c^n| \leq 1$, has a trivial solution, $a = 1$, and, $b = c$, for $\forall n > 1$. The Fermat Number Transform (FNT) resembles Discrete Fourier Transform, however, the former assumes the sums modulo a prime [12].

More importantly, the original formulation of FLT can be modified to allow the solutions to exist.

Claim 2. *For every n, there exist infinitely many natural integers a, b, c, m_1 and m_2 satisfying the congruence,*

$$a^n + b^n \equiv c^n \ (\text{Mod}(m_1, m_2)). \tag{10}$$

For example, assuming the first 100 natural numbers as strings of $l = 9$ digits in the number basis, $B = 8$, and, $B = 10$, the total number of solutions, n_l, and, n_r, respectively, of (10) for the first l_1 digits and the last $l_2 = l - l_1$ digits is given in Table I. It can be observed that, always, $n_\text{l} > n_\text{r}$, since the number strings often contain zeros at the left to make up the given width, l.

TABLE I. The number of solutions of (10)

	$B=8$				$B=10$			
	$n=3$		$n=4$		$n=3$		$n=4$	
(l_1,l_2)	(3,6)	(4,5)	(3,6)	(4,5)	(3,6)	(4,5)	(3,6)	(4,5)
n_l	69627	22278	5505	2318	1284	44532	10666	3622
n_r	212	644	730	2076	198	207	230	596

Claim 3. *For any integer, $n \geq 1$, the equation,*

$$a^n + b^n = c^n \tag{11}$$

has infinitely many solutions among the integers, $\cup_x \tilde{\mathbb{N}}_x$, for specific real-values (from some set), $x > 0$.

Proof. Let $c = y \in \mathcal{R}$, $a = y - d_1$, and, $b = y - d_2$, where d_1 and d_2 are arbitrarily chosen positive natural integers. Then, for any n, the polynomial (11) has at least one real-valued solution, $y > \max(d_1, d_2)$. Let $d_0 = \lfloor y \rfloor$ (floor function), so that $x = (y - d_0) < 1$. This defines the positive integers, $c = d_0 + x$, $a = d_0 - d_1 + x$, and $b = d_0 - d_2 + x$, from the set, $\tilde{\mathbb{N}}_x$. □

Proposition 1. *For any natural integer, n, there exists an integer, $m \geq n$, such that the expression,*

$$\sum_{i=1}^{m} a_i^n = b^n \tag{12}$$

is satisfied for a set of natural integers, $\{a_1, a_2, \ldots, a_m\} \cup \{b\}$.

The proof of Proposition 1 appears to be rather non-trivial, except when $n = 1$ and $n = 2$ (Pythagorean theorem). However, it is easy to find examples satisfying the expression (12), e.g.,

$$\begin{aligned} 3^2 + 4^2 &= 5^2 \ \ (m = n = 2) \\ 3^3 + 4^3 + 5^3 &= 6^3 \ \ (m = n = 3) \\ 2^4 + 2^4 + 3^4 + 4^4 + 4^4 &= 5^4 \ \ (m = n + 1 = 5) \\ 19^5 + 43^5 + 46^5 + 47^5 + 67^5 &= 72^5 \ \ (m = n = 5). \end{aligned} \tag{13}$$

In general, the sequence, $a^n + b^n$, obtained by enumerating all natural integers, a, and, b, becomes rapidly very sparse as the exponent, n, is increased. Given n, it is easy to show that the best approximation of $(a^n + b^n)$ by c^n is obtained when, $c = \lfloor (a^n + b^n)^{1/n} \rceil$ (rounding function). This motivates the following Fermat metric.

Definition 2. *The Fermat metric for positive numbers, a, and, b, is computed as,*

$$\mathcal{F}_n(a,b) = a^n + b^n - \lfloor (a^n + b^n)^{1/n} \rceil^n \tag{14}$$

where $n = 2, 3, \ldots$ is a natural number, and always, $\mathcal{F}_1(a,b) = 0$. The Fermat distance between the numbers, a, and, b, is the absolute value of the Fermat metric, i.e.,

$$D_n(a,b) = |\mathcal{F}_n(a,b)|. \tag{15}$$

The distribution of Fermat metric values by enumerating all pairs of natural integers up to 10^5 are shown in Figure 1 for $n = 2$ and $n = 3$, respectively. It can be observed that the Fermat metric values are spread much more evenly when $n = 2$, and the distributions are asymmetric about 0.

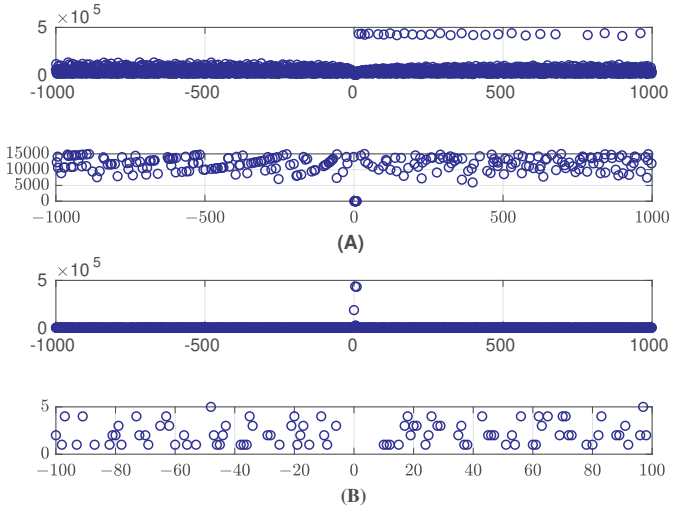

Figure 1. The counts of Fermat metric values for all pairs of natural integers up to 10^5, for the exponents $n = 2$ (A), and $n = 3$ (B).

The Fermat distance can be used to cluster natural numbers into subsets. Figure 2 shows the dendrogram assuming the distance, $\mathcal{F}_2(a,b)$. The corresponding assignment of the first 50 natural numbers into four subsets based on the distances, $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{F}_4$, and $\mathcal{F}_5$ are shown in Figure 3.

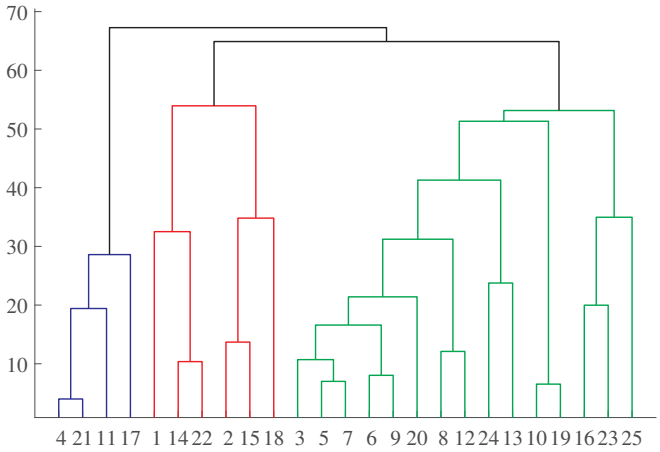

Figure 2. The dendrogram of natural numbers constructed assuming the Fermat distance, $\mathcal{F}_2(a,b)$, defined in (15).

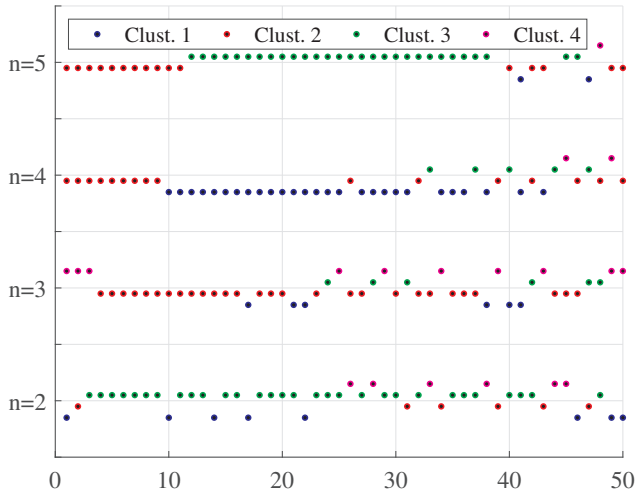

Figure 3. The first 50 natural numbers partitioned into four clusters (subsets) using the Fermat distances, $\mathcal{F}_n(a,b)$, for $n = 2,3,4$, and 5.

IV. Discussion and Conclusion

This paper investigated modular arithmetic and introduced dual modulo operator under the premise that all machine numbers can be assumed to be integers, when they are pre-allocated a fixed space in a computer memory. This is the case with fixed point as well as floating point number representations. The meaning of a decimal point is mainly syntactical to allow aligning the operands in arithmetic operations. This leads to another fundamental claim.

Claim 4. *Any computing algorithm utilizing finite number representations can be represented by a system of Diophantine equations.*

Hence, there is a large gap between mathematical description based on real analysis, and the actual implementation of corresponding algorithms on computers [5].

Improving accuracy of machine numbers by p-adic representations [8] and by Diophantine approximations [13] is impractical, since the underlying arithmetic operations require more time and more memory. More efficient multi-precision arithmetic is available as a *C*-library [3]. In many practical applications, finite accuracy is often sufficient [10]. On the other hand, the full accuracy is required in cryptography [14].

The FLT can be modified to allow the solutions to exist. The key ideas introduced in this paper are to define equivalences between numbers assuming only a subset of digits in their number string representations, and to consider sets of natural numbers offset by real-valued constants. In addition, the Fermat metric can be used to define distances between natural and other numbers.

Future work can define and prove further properties of machine numbers and arithmetic, which are explicitly considered to be integers. This can lead to more efficient design of integer-based models and architectures for large-scale computing machines, and improved approximations of real-valued systems.

Acknowledgment

This work was funded by a research grant from Zhejiang University.

References

[1] X. Zheng and R. Rettinger, "Weak computability and representation of reals," *Mathematical Logic Quarterly*, vol. 50, no. 4–5, pp. 431–442, Sep. 2004.

[2] R. T. Kneusel, *Numbers and Computers*, 2nd ed. Springer International Publishing, Cham, Switzerland, 2017.

[3] T. Granlund, "The GNU multiple precision arithmetic library, GNU MP 6.2.1," https://gmplib.org, Jan. 2020, accessed: 2023-01-30.

[4] M. Wrocławski, "Representations of natural numbers and computability of various functions," in *Conference on Computability in Europe*, 2019, pp. 298–309.

[5] R. Krebbers and B. Spitters, "Type classes for efficient exact real arithmetic in Coq," *Logical Methods in Computer Science*, vol. 9, no. 1:01, pp. 1–27, Feb. 2013.

[6] R. O'Connor, "A monadic, functional implementation of real numbers," *Mathematical Structures in Computer Science*, vol. 17, no. 1, pp. 129–159, 2007.

[7] J. van der Hoeven, "Computations with effective real numbers," *Theoretical Computer Science*, vol. 351, pp. 52–60, 2006.

[8] F. Q. Gouvêa, *p-adic Numbers: An Introduction*, 3rd ed. Springer, Cham, Switzerland, 2020.

[9] G. Kremer, F. Corzilius, and E. Ábrahám, "A generalised branch-and-bound approach and its application in SAT modulo nonlinear integer arithmetic," in *Computer Algebra in Scientific Computing*, vol. 9890, 2016, pp. 315–335.

[10] NASA/JPL Edu, "How many decimals of Pi do we really need?" https://www.jpl.nasa.gov/edu/news/2016/3/16/how-many-decimals-of-pi-do-we-really-need, Oct. 2022, accesssed: 2023-01-30.

[11] T. Gowers, J. Barrow-Green, and I. Leader, *The Princeton Companion to Mathematics*. Princeton University Press, Princeton, NJ, USA, 2008.

[12] M. Křížek, , F. Luca, and L. Somer, *17 Lectures on Fermat Numbers: From Number Theory to Geometry*. Springer New York, USA, 2001, ch. Fermat Number Transform and Other Applications, pp. 165–186.

[13] B. Church, "Diophantine approximation and transcendence theory," Apr. 2019, lecture Notes.

[14] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, 2nd ed. Springer, New York, NY, USA, 2014.

 ISBN: 978-1-68558-057-5