Application of hyperoperations for engineering practice

Application of hyperoperations for engineering practice

Konstantin A. Rubtsov, Igor S. Konstantinov, Sergey A. Lazarev, Konstantin A. Polshchykov, Vladimir E. Kiselev

Belgorod State University 85 Pobedy str., Belgorod, Russia, 308015,
rubcov@mail.ru

Abstract. This article presents some of the main points of the study for the practical application of various hyperoperations in engineering practice. The authors examined the use of hyperoperations in mathematical formalization of analytical solutions of branching algorithms, the use of new number formats for encoding information based on large numbers. The methods of applying hyperoperations considered by the authors allow them to be effectively used on modern microprocessor-based computers with a built-in mathematical coprocessor, which is an integral part of the processor core, which allows implementing algorithms based on operations of the order higher than "addition" and "subtraction".
Keywords: hyperoperations, homomorphism, number formats, superlogarithm, tetration, zeration, Ackermann function.

1. Introduction

In connection with the mass distribution of microprocessor computing in recent decades, increasing its productivity and lowering prices, mathematical modeling and digital processing of information are widespread in engineering practice. The use of mathematical modeling as a special case of sign modeling, in which the description of the modeling object is carried out in the mathematical language, and model research is carried out on the basis of mathematical methods, is currently one of the most productive and frequently used methods of scientific knowledge. The low cost of high-performance microcontrollers, signal processors and FPGAs (programmable logic device, PLD) allowed for mass production of various industrial and household equipment based on digital algorithms for processing information and signals. The emergence of new powerful hardware information processing allows the use of algorithms, previously difficult to implement. One of the ways to create new algorithms is homomorphism [1]. Such algorithms are widely used in mathematical logic and cybernetics [2]. This article considers another aspect of the application of mathematical formalism in engineering practice - the application of new functions based on the expansion of the classical mathematical basis: addition-subtraction, multiplication-division, power-exponential function, root function and logarithmic function.

2. Methods

The works [3, 4] present a mathematical description of homomorphism. The work [5] describes a homomorphic method for constructing individual mathematical objects: numbers, operations, functions, differentials, derivatives, integrals, series, numerical methods, and differential equations. Each object can have an infinite number of homomorphisms. Based on the homomorphism, it is possible to create new methods of studying functions on the basis of existing methods, new numerical methods, and expand the set of classes of solved differential equations [6]. The basis for the construction of hyperoperations is the recursive Ackermann function [7]. Ackermann function finds

practical application for testing the compiler's ability to optimize recursion. The first who used the Ackermann function for this purpose was Yngve Sundblad [8].

3. Problem statement

This article considers the main directions of research on the use of hyperoperations for engineering practice: the construction of number formats that can encode large numbers, the use of the hyperoperation zeration for writing in the form of a single formula of functions previously defined as a system with conditions.

4. Main part

4.1 Ackermann function and hyperoperator

In 1928, William Ackermann, a student of David Hilbert, published his function $\varphi(m,n,p)$, which was defined for $p=0,1,2$ and corresponded to the well-known operations of addition, multiplication, raising to a power: $\varphi(m,n,0)=m+n$, $\varphi(m,n,1)=m\cdot n$, $\varphi(m,n,2)=m^n$ [7].
In 1948, Rafael Robinson presents a function with two arguments constructed on the basis of the Ackerman function [9]. Currently, this variant of the Ackerman function is the most popular. According to Robinson, Ackerman's function is defined recursively for $m,n\in\mathbf{N}$:

$$A(m,n)=\begin{cases} n+1, m=0; \\ A(m-1,1), m>0, n=0; \\ A(m-1,A(m,n-1)), m>0, n>0. \end{cases} \quad (1)$$

In 1976, Donalt Knuth expanded the Ackermann function $\varphi(m,n,p)$ for $p>2$, who proposed to write down a sequence of operations higher in rank than degree in the form of arrows [10],[11]. This notation was called Knuth's up-arrow notation. Its meaning was that multiplication can be represented as a repetition several times of addition, and raising to a power is a multiplication performed several times. This notation made it possible to compactly record the repeated exponentiation, called super-degree or tetration. The idea of the existence of tetration, an operation with a rank higher than the degree was described as far back as 1844, and in 1901 a record of the tetration was proposed as a "degree to the left" [12, 20]. In fact, the Ackeramann function $\varphi(m,n,p)$ at $p>2$ allows describing an infinite number of operations with a rank higher than a degree - tetration, pentation, hexation, etc. The whole set of operations formed using the Ackermann function and denoted by a single operator is called a hyperoperator. Operations obtained with the help of a hyperoperator are called hyperoperations. In an extended version, the hyperoperator also describes the inverse operations. In 1989, K.A. Rubtsov proposed writing such a hyperoperator ${}_p^i R_m^n$ as a function of four arguments ($i=1,2,3$), which in the case of non-commutativity has two inverse functions ($i=2,3$) [13]. For degree, this is the root and logarithm, for tetration (superdegree) it is the superroot and superlogarithm [13]. In the same work, the notation of the function of superlogarithms as "slog" was proposed. Tetration and its inverse functions are currently still at the stage of research and development of algorithms for their exact calculation [14, 15].

4.2 Zeration and its application

The Ackerman function $\varphi(m,n,p)$ for $p=0$ corresponds to addition. In Robinson's notation according to formula (1), $m=1$ corresponds to addition, $m=2$ - to multiplication, $m=3$ - to degree, $m=4$ - to tetration, etc. It should be noted that formula (1) is defined for $m=0$. It is logical

to assume that formula (1) for $m = 0$ gives the result of computing an unknown operation with a rank less than addition. Such an operation was called "zeration" and was first described in 1989 as a "zero-action" paper [13]. Ackerman's function in Robinson's record gives only the result of this operation, and to determine it, it is necessary to consider addition as a result of repeating zeration:

$$\underbrace{m \circ m \circ \ldots \circ m}_{n} = m + n \tag{2}$$

Formula (2) for natural numbers produces the definition of zeration [13, 15]:

$$m \circ n = \begin{cases} m+1, m > n; \\ n+1, m < n; \\ m+2 = n+2, m = n. \end{cases} \tag{3}$$

Formula (2) does not contradict the Ackermann function (1) for $n > 2$. In table 1, zero rank should be written as $2 \circ (n+3) - 3$. Given that the level of operation of zeration is lower than addition, then its priority is lower than addition and subtraction. Then $2 \circ (n+3) - 3 = 2 \circ n + 3 - 3 = 2 \circ n$. From (3) it follows that for $n > 2$: $2 \circ n = n + 1$. It was proposed to supplement definition (3) so that it corresponds to the value of the Ackermann function $n+1$ for $n = 0,1,2$. However, such additions contradict the definition of the operation of zeration according to the formula (2).

Table 1. Ackermann function values and operations used.

Level	Ackermann function	Value	Operation
0	$A(0, n)$	$n+1$	Zeration
1	$A(1, n)$	$2 + (n+3) - 3$	Addition
2	$A(2, n)$	$2 \cdot (n+3) - 3$	Multiplication
3	$A(3, n)$	$2^{n+3} - 3$	Degree
4	$A(4, n)$	${}^{n+3}2 - 3$	Superdegree (tetration)
…	…	…	…
k	$A(k, n)$	${}_1^k R_2^{n+3} - 3$	k-rank operation [1]

It is now accepted that elementary arithmetic operations are addition and subtraction. The remaining arithmetic operations are their repetition. Addition and subtraction refer to operations of the first level; multiplication and division is the second level; exponentiation, root extraction and logarithms - the third level; tetration, superroot, and superlogarithm - the fourth level. Formulas (2, 3) determine the arithmetic operation of the zero level. In fact, addition and subtraction cease to be elementary arithmetic operations.

Addition and subtraction are the most commonly used arithmetic operations. Even the simplest microprocessor contains integer addition instructions. After determining zeration, the questions arise: "Where can zeration be used?", "Why is it needed if there is mathematics and computer technology?".

In practice, most processes cannot be described by a single mathematical formula without the use of systems with conditions. For example, the relationship of the transition of a car from a state of rest to movement and subsequent changes in its speed to a stop. Using a finite number of arithmetic operations, it is impossible to write a single formula for the process of motion. To solve this problem, systems with the condition were introduced into mathematical formalism. When creating algorithms for solving practical problems in computer technology, it was necessary to use Boolean algebra operations (priority below addition and subtraction), since the arithmetic operations available are not enough to describe real processes. In mathematical formalism a lot of functions have also appeared that cannot be written by a finite number of arithmetic operations without using conditional systems.

For example, the module function $|x|$, the function of the sign of the number "$\operatorname{sgn} x$", the maximum and minimum of two numbers, and others. Zeration solves this problem in the framework of arithmetic operations. We shall consider examples of writing some functions [15].
Real value module:

$$|x| = (1-(-x \circ 0)) \cdot ((x \circ 0)-2) + ((x \circ 0)-1) \cdot (2-(-x \circ 0)). \quad (4)$$

Number sign function:

$$\operatorname{sgn}(x) = (2-(-x \circ 0)) \cdot (2-(1-(x \circ 0) \circ 0)) - ((x \circ 0)-2) \cdot ((1-(-x \circ 0) \circ 0)-2). \quad (5)$$

Within the framework of arithmetic operations, zeration describes the elements of Boolean algebra - the basis of modern computing systems. We shall consider the description of the basic set of operations: disjunction $(a \vee b = c)$, conjunction $(a \wedge b = c)$, negation $(\neg b = c)$ [15].

$$c = \neg b = (b \circ 0)-(b+1), \quad c = a \vee b = a+b-(a+b \circ 2)+3, \quad a \wedge b = \neg((\neg a) \vee (\neg b)). \quad (6)$$

In some cases, zeration is not enough. For example, to describe the Dirac function $\delta -$. In this case, it is possible to use the inverse operation - “deltation” [15].

4.3 Notation of numbers based on homomorphism

The use of hyperoperations of the fourth level of tetration, superroot and superlogarithm is difficult due to problems with limiting the format of numbers like $D = d \cdot a^n$, where $d -$ is the mantissa, $a -$ is the base of the number system, and $n -$ is the exponent. For compact coding of large numbers in the case of applying level 4 hyperoperations, new number formats are needed [16, 17].
In [18],[19], the use of number formats based on the hyperoperations RRH1, RRH2, RRH3, and RRH4, called hyperformats, was proposed.

RRH1 format $D_{\mathrm{RRH1}} = {}^{(n+\operatorname{slog}_a d)}a = d {}^{[a]}*({}^{n}a)$. For calculations, the base $a = 10$ or $a = 2$ is used. The main feature of this format is its similarity with the well-known floating point format: $D = d \cdot a^n$. RRH1 hyperformat is constructed by increasing the rank of operations of the classical floating-point format: multiplication is replaced by the addition homomorphism based on the superlogarithm, and the degree by tetration. The algorithm for converting a standard real number $D = d \cdot a^n$ to RRH1 is very simple. To do this, it is necessary to perform a logarithm or antilogarithm with a base a until we get a number $D \in [1, a[$, then $d := D$. Initially $n := 0$, with each logarithm, numbers D should be incremented $n := n+1$, and with antilogarithm $n := n-1$. Thus, we can write numbers in RRH1 as follows: $-100 = 1{,}00000000\ 000000 * {}^{-2}10$, $-10 = 1{,}00000000\ 023026 * {}^{-2}10$, $-1 = 1{,}25892541\ 179417 * {}^{-2}10$, $-0{,}5 = 2{,}07122732\ 048046 * {}^{-2}10$, $0 = 1 * {}^{-1}10$, $0{,}5 = 3{,}16227766\ 016838 * {}^{-1}10$, $1 = 1 * {}^{0}10$, $5 = 5 * {}^{0}10$, $10 = 1 * {}^{1}10$, $100 = 2 * {}^{1}10$, $10^{10} = 1 * {}^{2}10$, $10^{100} = 2 * {}^{2}10$, $5 \cdot 10^{100} = 2{,}00302502\ 84114 * {}^{2}10$, $1 \cdot 10^{1000} = 3 * {}^{2}10$, $5 \cdot 10^{1000} = 3{,}00030345\ 277606 * {}^{2}10$.

RRH1 can encode negative numbers without the additional sign bit. However, this case poses problems with accuracy. For negative numbers, the RRH2 format is used, which has the sign of a number.

RRH2 format $D_{\mathrm{RRH2}} = \{+,-,\cdot,:\}^{(n+\operatorname{slog}_a d)}a = \{+,-,\cdot,:\}d {}^{[a]}*({}^{n}a)$. In fact, RRH2 is the RRH1 format with the addition of one or more characters in front of the number D_{RRH1}. A number in RRH2 is encoded according to the rules of RRH1 if $D \geq 0$. If $D < 0$, then RRH2 is presented as $D_{\mathrm{RRH2}} = -d {}^{[a]}*({}^{n}a)$, and the calculation d and n is performed similarly to RRH1, after calculating the module of the number $|D|$. The use of the sign "–" eliminates the problem of accuracy of the RRH1 format for

negative numbers. However, if one or more signs of the number are used (advanced RRH2), the accuracy of RRH1 increases for given ranges of numbers. For RRH2, the range $0 < D < 1$ is written in the reciprocal of the number and the sign $":"$ is put. For example, a number $0{,}5$ can be written as $0{,}5 = 1/2 = 1:2 = :2$, by analogy with $0 - 2 = -2$. For positive numbers, by default, the sign $"+"$, for numbers $D \geq 1$, by default, the sign "multiplication": $5 = 0 + 5 = +5$ and $5 = 1 \cdot 5 = \cdot 5$. Such a mathematical formalism made it possible to increase the accuracy of RRH2 in the range of numbers $-1 < D < 1$.

RRH3 hyperformat of the number D is a complex number. This format consists of two numbers: the real part is a floating point number or in the RRH1 format, the imaginary part is a pseudo-character number, i.e. calculation history of the function $\ln|x|$. The algorithm for converting a real number to RRH3 format will be performed according to the scheme given for RRH1. With a negative argument in the function $\ln x$, we perform the calculation $\ln|x|$, while incrementing the imaginary part of the number and multiply by 2. If the argument is positive, then we calculate $\ln x$ and multiply the imaginary part of the number by 2 without increment. Thus, the imaginary part of the number accumulates information about the history of negative arguments in the function $\ln|x|$.

For example, we need to calculate ${}^{-60,34}10$ using an approximation based on the approximation $\mathrm{sln}(x) \approx \ln(x)$ for $x \in [1; e[$. In this case, we can write: $\mathrm{sln}(x) \approx \mathrm{sln}(e^x) - 1$ at $x \leq 0$, $\mathrm{sln}(x) \approx x - 1$ at $0 < x \leq 1$, $\mathrm{sln}(x) \approx \mathrm{sln}(\ln(x)) + 1$ at $1 < x$ [6, 14].

In Mathematica we get ${}^{-60,34}10 \approx \lg(\lg \ldots(\lg(1 - 0{,}34))) \approx -0{,}119193 + 0{,}750583 \cdot i$. In the inverse problem, we obtain $\mathrm{slg}(-0{,}119193 + 0{,}750583 \cdot i) \approx -60{,}334776 - 0{,}0161977 \cdot i$ instead of the exact value $-60{,}34$. Relative module error does not exceed $8{,}7 \cdot 10^{-3}\ \%$. The increase in bit depth does not lead to a significant increase in accuracy. For the RRH3 format, we get ${}^{-60,34}10 \approx (-0{,}348672 + 571674577919191807 \cdot i)_{\mathrm{RRH3}}$. RRH3 is resistant to imaginary rounding. For example, we can write $(-0{,}348672 + 571674577919191807 \cdot i)_{\mathrm{RRH3}} \approx (-0{,}34867 + 5{,}7167 \cdot 10^{17} \cdot i)_{\mathrm{RRH3}}$. The inverse solution to $\mathrm{slg}(-0{,}34867 + 5{,}7167 \cdot 10^{17} \cdot i)_{\mathrm{RRH3}} \approx -60{,}3398918065$, i.e. the relative error does not exceed $1{,}8 \cdot 10^{-4}\ \%$ and the result of the inverse transformation without the imaginary part.

In the RRH3 format, complex numbers are used only for storage and compatibility with existing program libraries. The rules for working with RRH3 numbers for logarithms and antilogarithms differ from Euler's formulas. For example, $\lg(-1) = (\pi i + 2\pi n)/\ln(10)$ is known, where $n \in \mathbf{Z}$, for RRH3: $\lg(-1) = (0 + 1 \cdot i)_{\mathrm{RRH3}} = i_{\mathrm{RRH3}}$. Standard calculation: $\lg(\lg(-1)) \approx 0{,}134934 + 0{,}682188\pi i$, and calculation in the RRH3 format: $\lg(\lg(-1)) \approx (-4{,}34294 + 2 \cdot i)_{\mathrm{RRH3}}$.

The RRH3 format reduces the error of multiple logarithms using the properties of tetration and superlogarithm.

RRH4 hyperformat of a number D is a positive number constructed according to the rules of RRH1 and the positional number system. In fact, in RRH4, the pseudo-sign (imaginary) part of RRH3 was converted to a fractional part of the number and an integer part was formed on the basis of the real part of RRH3. The algorithm for converting a regular number to RRH4 contains a bitwise calculation of the division homomorphism based on the superlogarithm with the accumulation of the result. The resulting number is positive real and compatible with all mathematical libraries [18],[19].

As an example, we shall write some numbers in RRH4. Translation algorithms are implemented in Mathematica: $10000\,! \approx 2{,}846 \cdot 10^{35629} \approx (455{,}21747)_{\mathrm{RRH4}} \approx (100011{,}00010101111001)_{2/\mathrm{RRH4}}$, where $2/\mathrm{RRH4}-$ is the designation RRH4 for the binary number system.

$-15 \approx (0{,}01000100010101001010 1)_{2/\mathrm{RRH}4}$. The above calculation example ${}^{-60{,}34}10 \approx \left(-0{,}3478362 + 5{,}71674577912 \cdot 10^{17} \cdot i\right)_{\mathrm{RRH}3}$ in RRH4: ${}^{-60{,}34}10 \approx \left(4{,}57095 \cdot 10^{-61}\right)_{\mathrm{RRH}4}$. This number can be rounded without serious loss of accuracy:

$\left(4{,}57095 \cdot 10^{-61}\right)_{\mathrm{RRH}4} \approx \left(4{,}57 \cdot 10^{-61}\right)_{\mathrm{RRH}4}$, $\mathrm{slg}\left(4{,}57 \cdot 10^{-61}\right)_{\mathrm{RRH}4} \approx \lg\left(4{,}57 \cdot 10^{-61}\right) \approx -60{,}3400838 \approx -60.34$.

The RRH4 format allows compact encoding of large numbers and the creation of unique numerical identifiers in sorting algorithms for large data arrays.

5. Conclusion

The hyperoperator and the hyperoperations obtained on its basis are a relatively new direction in mathematics. The use of the new hyperoperation - zeration - makes it possible to expand the set of basic arithmetic operations and the mathematical apparatus as a whole when modeling complex physical processes. Using the fundamental principles of mathematics (the concept of numbers and operations), zeration unifies the recording of a whole class of functions and algorithms. On its basis, one can get a single method for studying special functions, the formation of computing systems and algorithms directly related to basic mathematical elements. In the future, such unification will create the ability to effectively carry out research on mathematical models of complex objects and processes. Tetration is useful for building new number formats. The considered examples of new number formats provide a compact record of large numbers with the possibility of using standard formats for real and complex floating-point numbers.

6. Summary

The authors examined several special cases of the use of hyperoperations of the zero and fourth levels. The operation of the fourth level (tetration) is currently being actively studied, but has limited application due to the complexity of its calculation. The hyperformats of the considered numbers make it easier to apply tetration in practice. Zero-level hyperoperation (zeration) is implicitly used everywhere in practice in computing systems in the form of Boolean algebra, condition functions, conditional operators in programming languages, and mathematics to describe systems with conditions. The authors show the possibility of expanding arithmetic operations to solve the problems of engineering practice. In the long term, zeration can be included in the basic instruction set of microprocessor technology, since it has the simplest calculation algorithm. This will allow solving various classes of problems in engineering practice, using only arithmetic operations.

References

[1] Mathematical encyclopedia 1977 Soviet encyclopedia Vol 1: 1152 p
[2] Parmar PV 2014 Survey of various homomorphic encryption algorithms and schemes Intern J of Computer Applications Vol 91 no 8
[3] Maltsev AI 1970 Algebraic systems Nauka: 392 p
[4] Chang CC Keisler HJ 2013 Model Theory Courier Corporation: 672 p
[5] Rubtsov KA 1996 New mathematical objects BelGTASM NPP INFORMAVTOSIM: 251 p
[6] Rubtsov KA Konstantinov IS Lazarev SA Romerio GF 2018 On Application of Homomorphism in Engineering Practice and Scientific Research International Journal of Pharmaceutical Research Vol 10 Issue 1 : 283-288
[7] Ackermann W 1928 Zum Hilbertschen Aufbau der reellen Zahlen 99 : 118-133
[8] Sundblad Y 1971 The Ackermann function A theoretical computational and formula manipulative study BIT journal Vol 11 : 107-119
[9] Robinson RM 1948 Recursion and double recursion Bulletin of the American mathematical society Vol 54 № 10 : 987-993
[10] Knuth DE 1976 Mathematics and Computer Science: Coping with Finiteness Science journal Vol 194 : 1235-1242

[11] Maheswaran G Balamurali R Sundarambal B Soundarya Priyadharshini S Vinodhini P 2020 A Smart Plug for Controlling and Automation Anywhere Using Iot International Journal of Advanced Science and Technology 293 pp8570 - 8577
[12] Maurer H 1901 Uber die Funktion fur ganzzahliges Argument Abundanzen Mittheilungen der Mathematische Gesellschaft in Hamburg 4: 33-50
[13] Rubtsov KA 1989 Algorithmization of ingredients in a variety of algebraic operations Cybernetics No 3: 111-112
[14] Rubtsov KA Romerio GF 2006 Hyper-operations as a tool for science and engineering International Congress of Mathematicians ICM-2006 Abstracts Posters Short Communications Mathematical Software Other Activities: 22-23
[15] Rubtsov KA Romerio GF 2011 Hyperoperations for Science and Technology: New Algorithmic Tools for Computer Science Lambert Academic Publishing: 185 p
[16] Rubtsov K Giovanni FR 2010 Applications of a number notation hyperformat for science and engineering International Congress of Mathematicians ICM-2010
[17] Rubtsov KA Romerio GF 2010 The hyperformat of numbers in the homomorphism of numerical methods for solving differential equations Proceedings of the international scientific conference MMTT-23 Vol 1: 76-78
[18] Rubtsov KA Romerio GF 2012 Algorithmization of the hyperformat of numbers in scientific research Proceedings of the international scientific conference MMTT-25 Vol 10: 118-122
[19] Janarthanan R Doss S and Balamurali R 2020 Robotic-based nonlinear device fault detection with sensor fault and limited capacity for communication JOURNAL OF AMBIENT INTELLIGENCE AND HUMANIZED COMPUTING
[20] Eisenstein G 1844 Entwicklung von $a^{a^{a^{\cdots}}}$ J reine angew Math 28: 49-52