Transmathematica DOI Prefix 10.36285 ISSN 2632-9212

# Errors, Failures, and Faults in the Context of Elementary Arithmetic

Jan A Bergstra
janaldertb@gmail.com j.a.bergstra@uva.nl
Informatics Institute
University of Amsterdam
Science Park 900, 1098 XH, Amsterdam, The Netherlands

Submitted: 7 August 2024
Revised: 17 December 2024

**Abstract**

It is argued that setting one over zero to the upthorn symbol, elsewhere used to denote bottom, as is done in common meadows, is conceptually meaningful for elementary mathematics. We consider the arguments in its favour, for the restricted context of use in elementary arithmetic, to be stronger than arguments supporting one over zero equals zero, as in Suppes-Ono division and variants of it sometimes used in the model theory of fields as a branch of mathematical logic, or one over zero as an unsigned infinity, as proposed by Riemann and used in the design of wheel arithmetic, or one over zero as positive infinity, as in transrational arithmetic.

## 1 Introduction

In the work on common meadows (e.g. [14, 15, 17, 18]) we make use of an additional value, symbolised with $\perp$, which names a peripheral number, that is, a number outside any conventional collection of numbers. The symbol $\perp$ is used to turn division into a total function, called common division, by adopting $\frac{1}{0} = \perp$, and more generally $\frac{x}{0} = \perp$ for all $x$.

Originally, the same peripheral number was denoted by the letter "**a**" in [12, 13] in order to avoid a premature import of computer science terminology into a subject which belongs rather to logic and mathematics. That convention has also been adopted in [20, 21]. However, in the follow-up work, we have used $\perp$ because this notation is more easily appreciated and recognized in a computer science setting. In addition we refer to $\perp$ as an error, error symbol, or error element.

Alternatively, one might consider $\perp$ to denote a failure, or an indication of failure rather than an error. We will argue why we prefer error over failure as a description of what the semantic idea of $\perp$ is supposed to

deliver. To begin with, we prefer not to refer to $\perp$ as "bottom" in an information-theoretic sense, for the simple reason that a claim that $\frac{1}{x} = \perp$ is not at all uninformative about $x$. For that reason we hesitate to use "bottom" as a verbal reference to $\perp$ is the same way as "zero" is used as a reference to 0 and "one" is used as a word for 1. When we must name the symbol $\perp$, we call it by its typographical name, "upthorn."

In order to argue that $\perp$ represents error rather than failure we need conceptual clarity about the notions error and failure. We will borrow such clarity from software theory.

## 2 Failures, errors, faults, mistakes, and defects in computing

We made a rather detailed investigation of the notions of error, fault, failure and defect in the context of computer programming in various papers ([7, 8, 9, 10, 11]). In this work we have developed in more detail the classical conceptual framework, of [5, 6] making use of older work in [22] and recent work in [23]. We are convinced that said framework which has a focus on the contrast between failure and fault, is still very useful and up to date and that it admits ample expansion into further themes both in computer programming and outside computer programming. For the purposes of the current paper we adopt the following explanation of the key notions:

(i) A failure is an event in which a system behaves otherwise than expected on the basis of its specification;

(ii) A fault is a fragment of system design which may be considered the (or a) cause of the occurrence of a failure (where the notion of cause is linked to the possibility to repair the system by changing the fragment at hand into a similar but better fragment which will make the failure disappear, preferably without introducing new failures);

(iii) A mistake is a designer action which causes a fault;

(iv) An error, or rather error condition, is a state in the processing of a system which indicates (predicts) that a failure is (or may well be) forthcoming. However, an error may also arise if a system is tasked with inputs outside its domain (as specified), in which case, simply by definition, no failure can arise.

(v) A system is defective if it is somehow wrong while the detailed notions of fault, failure, and error, do not capture (that is fail to capture) the problem. Defectiveness is a notion which goes beyond the framework of specification and implementation and which resists a technical definition (we refer to see [11] for more detail). Changing requirements can render an adequate (that is, non-defective) system defective.

## 3 A mathematical task

We suppose that a group $G$ of $A$ persons intends to pay the cost of an activity involving two components $\in C_1$ and $\in C_2$. We will ignore the currency sign, $\in$, below.

Moreover we assume that a subset $S$ of said group $G$, with $B$ persons in it, with $B < A$, will not make any payment and the others, that is the members of $G - S$, will share the amount due on equal terms. So how much will the members of $G - S$ pay? That amount will be called $X$.

We find, as a possible solution, the following text/explanation/story:
S1 = let $C = C_1 + C_2$; let $N = A - B$; now $X = \frac{C}{N}$.

Another solution might be:
S2 = let $C = C_1 + C_2$; let $N = A - A$; now $X = \frac{C}{N}$.

Suppose $C_1 = C_2 = 100, A = 5, B = 3$ then all works well with S1: we find $X = \frac{200}{2} = 100$. However, with S2 we find $X = \frac{200}{0} = \perp$. We notice that the appearance in S2 of $\perp$ constitutes an error, which at the same time constitutes a wrong answer, that is a failure (as the valid response, 100, is not being delivered).

We now notice that the fragment "$N = A - A$" of S2 is faulty, that is, it serves as the location of a fault: indeed an improved text can be made by changing that particular of S2 fragment into "$N = A - B$". That state of affairs suffices to characterize the fragment "$N = A - A$" as a fault, or as containing a fault. Perhaps a rather trivial mistake (typo) was the cause of the fault, but the fault may also have been caused by a misunderstanding (that is a less trivial form of mistake) by the agent in charge of solving the problem.

Now we consider solution S1 and we determine the result with $A = B = 5$. We find $X = \frac{200}{5-5} = \frac{200}{0} = \perp$. Again an error appears, but this time the error is not indicative of the presence of a failure (of S1) as the task (as solved by S1) was performed outside the domain as given in the specification of the problem where $B < A$ is required. We find that the presence of an error is not always an indication of the presence of a fault, as it may equally well be indicative of wrong usage.

Finally we add the information that $C_1 \geq C_2$ and the constraint that no-one should pay more than $C_1$. We consider the case that $C_1 = 100, C_2 = 10, A = 6$, and $B = 5$ and we find that S1 yields $C = 110$ which is a problematic result (a failure) in the sense that $110 > C_1$. However, in this case there is no visible error, and there is also no fault in the text S1, simply because the specification is inconsistent, and no conceivable modification (local or not) of S1 will turn it into a correct solution of the problem at hand.

## 3.1 Appreciation of the example: $\perp$ is conceptually helpful

We hold that the very fact that errors need not come with failures, though failures often (but not always) come with errors, provides ample justification for using a mathematical notation and style of presentation which can be explicit about errors. First of all we notice that quite often adequate solutions may run into an erroneous situation when applied outside the intended domain. By making use of $\perp$, for instance when dividing

by zero, or upon taking a logarithm of 0 or of a negative number, that mathematical notation incorporates the phenomenon of an error in an explicit manner and thereby enables a more precise conceptual grasp of the mechanisms of failure, error and fault than otherwise would be possible without having $\perp$ available.

The example given above also illustrates why we suggest to consider $\perp$ as a representation of error rather than as a representation of failure.

### 3.2 Comparison with alternative approaches to dividing by zero

Other options for dealing with division by zero have ample application in computing. What we call Suppes-Ono division, adopting $\frac{1}{0} = 0$, appears in proof checking systems where it simplifies the type system at hand. Adopting $\frac{1}{0} = +\infty$, as in the transrationals and transreals of [1] models widespread conventions in floating point arithmetic. We will refer to adopting $\frac{1}{0} = +\infty$ together with $\frac{1}{+\infty} = 0$ as transmath division.

Although both approaches (Suppes-Ono division and transmath division) have merits in particular contexts, we feel that using $\frac{1}{0} = \perp$ has as an important advantage, not shared by the other two options, that it comes with an intuitive understanding of fault versus failure and error, and the fundamental role of specifications in these matters.

We also think that the intuition that there is something wrong with division by zero is of critical importance, and introducing $\perp$ upon confrontation with division by zero makes best use of the options to introduce an explicit error symbol in informal mathematics, an option which is not made use of when either adopting $\frac{1}{0} = 0$ or $\frac{1}{0} = +\infty$.

## 4 Surveying options for dealing with division by zero

The various options for dealing with division by zero are listed, admittedly in an arbitrary order.

1. Suppes-Ono division: $\frac{1}{0} = 0$. This option has the major advantage that no peripheral elements are introduced, which explains its use in proof checking systems.

   The metamathematics for Suppes-Ono division is quite attractive, and adopting the convention that $\frac{1}{0} = 0$ works quite well in certain applications. The latter advantages, however, lack a systematic background, and, in our view, do not constitute strong arguments in favour of Suppes-Ono division. From a conceptual point of view we see three disadvantages: (i) the argument made above that adopting $\frac{1}{0} = 0$ deviates too far from the conventional intuition that there is something wrong with dividing by zero, (ii) the argument made above that upon adopting $\frac{1}{0} = 0$ a useful opportunity is missed for introducing explicit error values in informal mathematics, and (iii) the fact that fracterm flattening fails in case of Suppes-Ono division (while it works with common division).

2. Adopting, say $\frac{1}{0} = 87$, just in order to simplify a type system, while making sure that no valid result produces 87, or working with $\frac{1}{0} = 1$ (or in fact $\frac{1}{0} = 0$), works the same. This approach to division by zero has the same weaknesses as were listed above for $\frac{1}{0} = 0$ augmented with the additional disadvantage that 87 is obviously an arbitrary choice. Perhaps the latter objection may be considered to be an advantage rather than a disadvantage as working with Suppes-Ono division suggests that adopting 0 as the value of $\frac{1}{0}$ is a deliberate (that is non-arbitrary) choice, coming with helpful merits, a suggestion for which only limited evidence can be found.

3. Adopting $\frac{1}{0} = \infty$, an unsigned "infinite" peripheral value (see [19]). This idea is clear but its potential for application in computing seems to be marginal at best. Like for Suppes-Ono division, fracterm flattening fails in this case (see [4]).

4. Adopting $\frac{1}{0} = +\infty$, a signed "infinite" peripheral value. This idea, as incorporated in transrationals and transreals, is certainly workable and it provides a reasonable model of various approaches to floating point arithmetic. As with the use of an unsigned infinite value fracterm flattening fails (see [4]). We believe that for elementary mathematics the disadvantages as mentioned for Suppes-Ono division are present for the assumption $\frac{1}{0} = +\infty$ as well.

5. Adopting $\frac{1}{0} = +\infty$, a signed "infinite" peripheral value which has as its inverse a signed infinitesimal value. This approach, which borrows from the symmetric transrationals of [16] is workable, though it does not model any current practical designs (like floating point standards). There are options for application in particular circumstances, but we do not see see any grounds for adopting or promoting symmetric transrationals (or any variation thereof) as a standard solution regarding division by zero for general use outside computing.

6. Adopting common division: $\frac{1}{0} = +\bot$. We expect that this option has technical merits inside computing as well as additional conceptual merits outside computing.

7. Adopting the idea that division is a partial function and that $\frac{1}{0}$ is undefined. This approach comes close to mathematical practice, though with a key difference: division is explicitly included in the signature of the language at hand and for that reason a logic of partial functions must underly the formalisation of reasoning.

   Using the partial division convention one may write, similar to conventions on limit notation:

   - "$\frac{p}{q}$ is undefined", or
   - "$\frac{p}{q}$ has no value", or
   - "$\frac{p}{q}$ is nonexistent".

   However, complications arise when considering equalities such as $\frac{p}{q} = r$.

   We hold that conventional first order logic is implausible when formalising reasoning about partial functions and that using a 3-valued logic with sequential (and non-commutative) logical connectives is

most plausible in a context with partial functions. An advantage of totalising division (as done in the suggestions listed above) lies in the option to make use of classical 2-valued logic.

8. Adopting the conventional explanation of division notation that, (for instance when working in the field rational numbers) once it is known that $q$ is non-zero, one may use the notation $\frac{p}{q}$ for a number, say $r$, with the property that $r \cdot q = p$.

   First of all we notice that this explanation is consistent with each of the 7 conventions that we have listed listed above, as nothing is said about what may or may not be done when it is not known that $q \neq 0$. The general understanding of said explanation seems to be that only when it is known that $q \neq 0$ the notation $\frac{p}{q}$ for a number, say $r$, with the property that $r \cdot q = p$ may be used. Making reference to available knowledge when explaining which syntax may be used constitutes a non-trivial detour which raises issues of textual legality as introduced and discussed in [17]. For instance making an assessment of the legality of using the division symbol in the sentence $x \neq 0 \to \frac{x}{x} = 1$ in advance of processing its semantic content seems to be impossible if asserting $\frac{0}{0} = 1$ is considered to be somehow illegal.

9. The final option is simply not to care about issues concerning division by zero and to claim, either explicitly or implicitly, that by working with sound intuitive guidance, preferably based on arithmetical experience, the assumption that $\frac{a}{b} \cdot b = a$ always works and never leads to problems.

# 5 Fracterms

We adopt the following informal ratio:

$$\text{fracterm} : \text{rational number} = \text{Cauchy sequence} : \text{real number}$$

Classical analysis can do without a notion (term) the meaning of which ambiguously fluctuates between Cauchy sequence and real number. We are interested in developing elementary mathematics as much as possible in a similar manner, that is without making use of a notion (such as fraction) the meaning of which ambiguously fluctuates between fractional expression and rational number. Arguments in favour of the use of "fracterm" have been collected in [3].

In the same way as a Cauchy sequence has a first, second, third, etc., element (which a real number does not) a fracterm has a numerator and a denominator (which a rational number does not). Just as one of the constructions of reals (as equivalence classes of Cauchy sequences) abstracts from differences between various Cauchy sequences as present in a class of such sequences, a construction of rationals may be based on choosing a class of fracterms and abstracting from differences between those.

Comparing fracterms with Cauchy sequences may be criticised for using an insufficiently abstract view of fracterms, and in fact a more precise analogy is given by:

simple fracterm : rational number = Cauchy sequence : real number

## 5.1 Fracterm permissive division convention

Working with division notation may be done on the basis of various disparate conventions. Fracterm permissive conventions allow the use of any fracterm without being worried about the existence of a proper denotation of the fracterm at hand. Fracterm restrictive division conventions, however, will impose or suggest limitations on the usage of fracterms. Below we briefly discuss three fracterm permissive division conventions.

## 5.2 Common division convention

Upon adopting all fracterms, including say $\frac{1}{0}$, as first class citizens, taking a position towards fracterms like $\frac{1}{0}$ can hardly be avoided. The proposition that division by zero produces $\perp$ has advantages and disadvantages which can only be fully and conclusively appreciated on the basis of future work. We propose to use the label *common division convention* for an approach to elementary arithmetic in which fracterms play a central role and in which division is made total and is understood as common division. We hold that the common division convention merits a thorough investigation concerning its advantages and deficiencies. The common division paradigm can only be successful in the long run if it proves helpful for the development of educational practice concerning elementary arithmetic, which in turn is unlikely to happen unless the common division convention proves to be helpful for the practical understanding of elementary arithmetic. All of this remains to be seen.

## 5.3 Suppes-Ono division convention

A first alternative to the common division paradigm is what we will call the *Suppes-Ono convention*, a paradigm which incorporates the conviction that adopting $\frac{1}{0} = 0$ is helpful for an understanding of elementary arithmetic. Neither Suppes in [25] nor Ono in [24] have made a case that said conventionm would be a good idea. Suppes is in [25] rather sceptical about the prospects that $\frac{1}{0} = 0$ will acquire wide-spread acceptance in mathematical circles. For further comments on [25] we refer to [2].

Besides a Suppes-Ono paradigm there is also a Suppes-Ono technique, which consists of adopting Suppes-Ono division for specialized technical work, as for instance is done with the design of proof checking software. The critical point regarding terminology is that one may very well use the Suppes-Ono technique while rejecting the Suppes-Ono convention.

## 5.4 Transarithmetic convention

We propose to label with *transarithmetic convention* the view that adopting transarithmetic (i.e. working with transmath division so that $\frac{1}{0} =$

$+\infty$) will be valuable as an approach to elementary arithmetic. James Anderson endorses the transarithmetic paradigm as based on a sequence of papers including [1]. Besides a transaritmetic paradigm one may discern a transarithmetic technique which merely consists of making use of adopting $\frac{1}{0} = +\infty$ for specialised practical applications, such as for instance the analysis of designs for fixed point systems and floating point systems of computer arithmetic.

# 6 Peripheral numbers for infinity

By adopting $\frac{1}{0} = \bot$ it is necessarily rejected at the same time that $\frac{1}{0} = +\infty$ (positive signed infinity) or $\frac{1}{0} = \infty$ (unsigned infinity). Notwithstanding the potential virtue of the latter assumptions in various applications, by adopting $\frac{1}{0} = \bot$ as a suggested convention for fairly general use, the status of peripherals for infinity such as $\infty$ (as for wheels) and $\pm\infty$ (as for transrationals) is made less prominent.

One may ask which account of peripherals for expressing intuitions about infinity fits common division. We find that at least three notions of signed infinity come to mind upon having adopted common division. We devote some comments to each of these options.

## 6.1 Positive and negative CM-infinity

CM-infinity, a shorthand for common meadow infinity, works as follows. A function $\mathsf{s}$ is adopted which delivers the sign of a number: $\mathsf{s}(x) = 0$ for $x = 0$, $\mathsf{s}(x) = 1$ for positive $x$ and $\mathsf{s}(x) = -1$ for negative $x$, further $\mathsf{s}(\bot) = \bot$. Now $\infty$ stands for positive CM-infinity and $-\infty$ stands for negative CM-infinity. Key properties of positive and negative infinity are listed in Table 1.

In the presence of CM-infinities fracterm flattening works well, a fact which is in notable contrast with all proposals for infinite peripherals we have seen in the literature thus far.

## 6.2 Positive and negative TCM-infinity

Trans-CM-infinity, a shorthand for trans-common meadow infinity, works as follows. TCM-infinities are more like transrational infinities than CM infinities. Again function $\mathsf{s}$ is adopted as before.

Now $\infty_{\mathsf{tr}}$ stands for positive TCM-infinity and $-\infty_{\mathsf{tr}}$ stands for negative TCM-infinity. Key properties of positive and negative infinity are listed in Table 1.

In the presence of TCM-infinities fracterm flattening fails.

## 6.3 Symmetric transrationals

The model of symmetric transrationals of [16] constitutes an enlargement of common signed meadows. We refer for detail on this matter to [16].

$$-\infty = (-1) \cdot \infty \tag{1}$$
$$0 \cdot \infty = \bot \tag{2}$$
$$\infty + \infty = \infty \tag{3}$$
$$\frac{x}{\infty} = \bot \tag{4}$$
$$\frac{x}{-\infty} = \bot \tag{5}$$
$$\mathsf{s}(\infty) = 1 \tag{6}$$
$$\mathsf{s}(-\infty) = -1 \tag{7}$$
$$\infty + \frac{n}{m} = \infty \quad \text{n, m integer numerals with m} \neq 0 \tag{8}$$
$$x \cdot \infty = \mathsf{s}(x) \cdot \infty \tag{9}$$

Table 1: Properties of CM infinities

$$-\infty_{\mathsf{tr}} = (-1) \cdot \infty_{\mathsf{tr}} \tag{10}$$
$$0 \cdot \infty_{\mathsf{tr}} = \bot \tag{11}$$
$$\infty_{\mathsf{tr}} + \infty_{\mathsf{tr}} = \infty_{\mathsf{tr}} \tag{12}$$
$$\frac{x}{\infty_{\mathsf{tr}}} = 0 \tag{13}$$
$$\frac{x}{-\infty_{\mathsf{tr}}} = 0 \tag{14}$$
$$\mathsf{s}(\infty_{\mathsf{tr}}) = 1 \tag{15}$$
$$\mathsf{s}(-\infty_{\mathsf{tr}}) = -1 \tag{16}$$
$$\infty_{\mathsf{tr}} + \frac{n}{m} = \infty_{\mathsf{tr}} \quad \text{n, m integer numerals with m} \neq 0 \tag{17}$$
$$x \cdot \infty_{\mathsf{tr}} = \mathsf{s}(x) \cdot \infty_{\mathsf{tr}} \tag{18}$$

Table 2: Properties of TCM infinities

# 7 Concluding remarks

As a conclusion of this paper we find that there is a case to be made for developing a practical understanding of elementary arithmetic on the basis of common division and the adoption of an explicit notation for an error, cast as a peripheral number. Such work may provide an assessment of the common division paradigm.

We consider it to be a significant additional advantage, shared with the other options for understanding division by zero by totalising division, that expressions like $\frac{1}{0}$ qualify as fracterms without hesitation, so that syntax and semantics can be understood in a more modular manner than is usual in today's approaches to elementary arithmetic.

Moreover we expect that there is significant merit in adopting software science based notions of failure, error and fault in the explanation of elementary arithmetic, because, as we see, arithmetic is just as much about reasoning as it is about measurement, quantification and calculation.

# References


[1] J. A. Anderson, N. Völker, and A. A. Adams. 2007. Perspecx Machine VIII, axioms of transreal arithmetic. In J. Latecki, D. M. Mount and A. Y. Wu (eds), *Proc. SPIE 6499. Vision Geometry XV*, 649902, 2007.

[2] J.A. Anderson and J.A. Bergstra. Review of Suppes 1957 proposals for division by zero. *Transmathematica.* ISSN 2632-9212 (published 06-08-2021), (2021). DOI

[3] J.A. Bergstra. Arithmetical datatypes, fracterms, and the fraction definition problem. *Transmathematica*, ISSN 2632-9212, (published 2020-04-30), (2020). DOI

[4] J.A. Bergstra. Fractions in transrational arithmetic. *Transmathematica.* ISSN 2632-9212 (published 01-27-2020), (2020). DOI

[5] Avižienzis, A. Laprie, J.C., Randell, B. Fundamental concepts of dependability. In Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments, Seoul (2001).

[6] Avižienzis, A. Laprie, J.C., Randell, B., Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on dependable and secure computing, **1** (1), pp.1–23 (2004).

[7] Jan Bergstra. Instruction sequence faults with formal change justification. *Scientific Annals of Computer Science* **30** (2), pp. 105–166 (2020).

[8] Jan Bergstra. Qualifications of instruction sequence failures, faults and defects: dormant, effective, detected, temporary, and permanent. *Scientific Annals of Computer Science* **31** (1), pp. 1–50 (2021).

[9] Jan Bergstra. Four notions of fault for program specifications. *Scientific Annals of Computer Science* **32** (2), pp. 183–209 (2022). DOI

[10] Jan Bergstra. A survey of testing for instruction sequence theory. *Scientific Annals of Computer Science* **32** (1), pp. 5–86 (2022). DOI

[11] Jan Bergstra. Defects and faults in algorithms, programs and instruction sequences. *Transmathematica* (2022). ISSN 2632-9212 (published 09-26-2022), DOI

[12] J.A. Bergstra and A. Ponse. Division by zero in common meadows. *In R. de Nicola and R. Hennicker (editors), Software, Services, and Systems (Wirsing Festschrift),* Lecture Notes in Computer Science 8950, pages 46-61, Springer, 2015. (2015). Also `arXiv:1406.6878v4 [math.RA]` (improved version, 2021).

[13] J.A. Bergstra and A. Ponse. Fracpairs and fractions over a reduced commutative ring. *Indigationes Mathematicae* 27, 727-748, (2016). DOI, URL

[14] J.A. Bergstra and J.V. Tucker. Which arithmetical data types admit fracterm flattening? *Scientific Annals of Computer Science*, 32 (1), 87–107, (2022). DOI

[15] J.A. Bergstra and J.V. Tucker. On the axioms of common meadows: Fracterm calculus, flattening and incompleteness. *The Computer Journal*, 66 (7), 1565-1572, (2023). DOI

[16] J.A. Bergstra and J.V. Tucker. Symmetric transrationals: The data type and the algorithmic degree of its equational theory, in N. Jansen et al. (eds.) *A Journey From Process Algebra via Timed Automata to Model Learning - A Festschrift Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science 13560, 63-80. Springer, 2022. DOI

[17] J.A. Bergstra and J.V. Tucker. Naive fracterm calculus. *J. Universal Computer Science*, 29 (9), 961-987, (2023). DOI

[18] J.A. Bergstra and J.V. Tucker. Synthetic fracterm calculus. *J. Universal Computer Science*, 30 (3), 289-307, (2024). DOI

[19] J. Carlström. 2004. Wheels–on division by zero, *Mathematical Structures in Computer Science*, 14 (1), (2004), 143-184.

[20] J. Dias and B. Dinis. Strolling through common meadows. *Communications in Algebra* (2024), 1–28, DOI, URL.

[21] J. Dias and B. Dinis. Towards an Enumeration of finite common meadows. *International Journal of Algebra and Computation.* (2024), DOI, URL.

[22] Laski, J. Programming faults and errors: towards a theory of software incorrectness. Annals of Software Engineering **4** pp. 79–114 (1997).

[23] Mili, A., Frias, M.F., Jaoua, A. On faults and faulty programs. P. Höfner et al. (Eds.): RAMiCS 2014, LNCS 8428, pp. 191–207, (2014).

[24] H. Ono. Equational theories and universal theories of fields. *Journal of the Mathematical Society of Japan*, 35(2), 289-306, (1983).

[25] P. Suppes. *Introduction to Logic.* Van Nostrand Reinhold Company (1957).