Transmathematica DOI Prefix 10.36285 ISSN 2632-9212

Arithmetical Datatypes, Fracterms, and The Fraction Definition Problem

Jan A. Bergstra
janaldertb@gmail.com
Informatics Institute
Science Park 904, 1098 XH Amsterdam, The Netherlands

Submitted: 3 November 2019
Revised: 30 April 2020

Abstract

Datatypes and abstract datatypes are positioned as results of importing aspects of universal algebra into computer science and software engineering. It is suggested that 50 years later, by way of a transfer in the opposite direction, outcomes of research on datatypes can be made available via elementary arithmetic. This idea leads to the notions of an arithmetical signature, an arithmetical datatype and an arithmetical abstract datatype and to algebraic specifications for such entities. The area of fractions in elementary arithmetic is chosen as an application area and while taking a common meadow of rational numbers as the basis, an arithmetical datatype equipped with an absorptive element. The use of datatypes and signatures makes syntax available for giving precise definitions in cases where lack of precision is common place. Fracterm is coined as the name for a fraction when primarily understood as a syntactic entity. The main contribution of the paper is to provide a detailed terminology of fracterms. Subsequently the fraction definition problem is stated, a distinction between explicit definitions of fractions and implicit definitions of fractions is made, and an outline of a survey of both forms of definitions of the notion of a fraction is given.

1 Introduction

The signature Σ_{urd}, for unital rings with division, is the signature of unital rings $\Sigma_{\mathsf{ur}} = (0, 1, +, \cdot, -)$ augmented with a two place division operator written x/y or alternatively $\frac{x}{y}$. An arithmetical signature is defined as either Σ_{urd} or any signature which extends Σ_{urd}. Σ_{ur} is termed an arithmetical signature without division.

A signature is a collection of names for sorts, constants for these sorts and functions over these sorts. For a function name in a signature an arity is given, i.e. information on how many arguments it takes, and from what sort in the signature the respective arguments come, as well as in what sort a result is obtained. Different notations may be used to denote the members of a signature, for instance $f\colon S_1 \times S_2 \times S_3 \to S_4$ may indicate a function named f with arity 3 which takes arguments in sorts S_1, S_2, and S_3 respectively and which yields results in sort S_4. The same notation also asserts the presence of the four (or less if some coincide) sorts S_1,..,S_4

I will assume that each algebra $\mathbb{A}$ comes with a unique signature $\Sigma = \Sigma(\mathbb{A})$. If $\Sigma(\mathbb{A}) = \Sigma$ then $\mathbb{A}$ is called a Σ-algebra. An arithmetical algebra is an algebra with an arithmetical signature. A Σ-algebra constitutes an interpretation of its signature providing for each sort a set as its interpretation and for each constant name an element of the corresponding set and for each function name the graph of a total function taking its arguments in the respective interpretations of the argument sort names and producing a result in the interpretation of its output sort. Below I will often ignore the distinction between a name and its interpretation and speak of sorts as sets, constants as elements of sorts and functions (names) as functions.

I will assume that the sorts of an algebra are non-empty sets. Homomorphisms and isomorphisms between algebras must leave the signature invariant. An algebra is minimal if for each of its sorts, each element is the interpretation of a closed expression over its signature. A minimal algebra has no proper subalgebras. The converse fails, however: let Σ only contain the sort V, no constants, and no functions. The structure $\mathbb{A}$ with $V_{\mathbb{A}} = \{0\}$ is a minimal algebra without proper subalgebras. But 0 is not the interpretation of a closed expression in $\Sigma = \Sigma(\mathbb{A})$. Datatypes are minimal algebras. An arithmetical datatype is a minimal arithmetical algebra.

Let $\Sigma \subseteq \Gamma$ be signatures, and let $\Sigma(\mathbb{A}) = \Gamma$. Now $\mathbb{A}|_{\Sigma}$, the Σ-reduct of $\mathbb{A}$ results from $\mathbb{A}$ by forgetting the sorts constants and functions outside Σ. When reducing an algebra the various sorts do not change. Complementary to the idea of reduction is that of expansion: $\mathbb{A}$ is an expansion of $\mathbb{A}|_{\Sigma}$. Besides reduct and expansion there are the notions of subalgebra and extension. $\mathbb{A}$ is a subalgebra of $\mathbb{B}$ if (i) both have the same signature, (ii) each sort of $\mathbb{A}$ is a subset of the corresponding sort of $\mathbb{B}$, (iii) constants have the same interpretations in both structures, and (iv) the functions of $\mathbb{A}$ are restrictions of the equally named functions of $\mathbb{B}$ to the respective sorts of $\mathbb{A}$. In case $\mathbb{A}$ is a subalgebra of $\mathbb{B}$ it is also said that $\mathbb{B}$ is an extension of $\mathbb{A}$. In case $\mathbb{B}$ is an expansion of an extension of $\mathbb{A}$, it is also said that $\mathbb{B}$ is an enlargement of $\mathbb{A}$.

This paper has two objectives: (i) to provide a manifesto on arithmetical datatypes: providing a rationale for the use of the theory of datatypes and abstract dataypes in elementary arithmetic, and (ii) to exemplify the relevance of this approach by introducing the notion of a fracterm and then providing a detailed terminology for fracterms, and for related notions of fracsign and fracpair. The underlying objective for introducing such terminology is that it is expected to be helpful for investigating the notion of a fraction. That fractions are worth systematic attention cannot be convincingly explained, however, in advance of actually investigating fractions in detail, and the observation that fraction is a complicated concept, if not a problematic concept, emerges only after careful preparations.

In computer science the notion of a datatype serves as an abstraction of a data structure, the latter understood as featuring in program notations and in operating systems and sometimes in hardware designs. The notion of an abstract datatype has emerged as yet a further step of mathematical abstraction of a datatype. An abstract datatype is the isomorphism class of a datatype. The theory of datatypes and abstract datatypes can be viewed as an application and elaboration of concepts and results from mathematics and logic, mainly from universal algebra, to the theory and practice of software construction.

In this paper I intend to move in the opposite direction and to make results from (abstract) datatype theory available for application in elementary mathematics. The point of departure in computer science, with respect to this "move" requires some preliminary clarification. Throughout the literature on datatypes and abstract datatypes various terms and phrases have not always been used in a consistent manner. As a consequence of such inconsistencies in the literature I will need to make some choices which are only partially supported by the available literature. This holds in particular for the choice to view an abstract datatype as an isomorphism class of a datatype.

1.1 Symbolic numbers

In [34] it is explained how the symbolic approach to arithmetical algebra came to accept negative numbers and imaginary numbers, while a vocal opposition against these innovations was present. Arithmetical datatypes with or without division provide ample freedom for systematically extending arithmetical structures with new elements thereby extending the symbolic approach beyond its classical successes: negative numbers, imaginary numbers, quaternions, p-adic numbers, dual numbers, and transcendental numbers.

Elements in a domain of numbers which embody an idea rather than a measurable quantity may be called symbolic values. A prime example of a symbolic value is $\bot$. Below some functions, which conventionally are considered partial, are made total by taking the value $\bot$ to represent an (otherwise) absent outcome. $\bot$ may be understood as the unique solution of the system of equations $\{0 \cdot x = x, 1 + x = x\}$. This amounts to working with the signature $\Sigma_{\mathsf{ur},\bot}$ which extends Σ_{ur} with $\bot$. Examples of the use of $\bot$ in integer arithmetic are: $1-2 = \bot$ ($\bot$-totalised subtraction on natural numbers), $3 : 4 = \bot$ ($\bot$-totalized division on integers), $3\backslash 4 = 1$ and $3\backslash 0 = \bot$ with $-\backslash-$ naming integer division with remainder, and $-\backslash\backslash-$ naming the remainder so that $7\backslash 2 = 3$, $7\backslash\backslash 2 = 1$, and $7\backslash\backslash 0 = \bot$.

The value $\bot$ is an absorptive element of the domain of numbers. Any operation taking $\bot$ as one of its arguments must return $\bot$. Including an absorptive element in arithmetic is not new. Wheels (see [37, 16, 17]) contain $\bot$, though without introduction of a new constant name, as the result of dividing 0 by 0. The transrational numbers of [1] contain an absorptive element (written Φ instead of $\bot$) named nullity. Wheels and transrationals feature the absorptive element in the presence of symbolic values for infinity (a non-zero solution of the equations $\{1/0 = x, 1/x = 0\}$). Common meadows [10, 11] contain $\bot$(written a for additional/absorptive in these papers) as a symbolic number. Currently $\bot(\Phi), \infty$, and $-\infty$ constitute the most common symbolic numbers. The constant ϵ in the context of dual numbers may be considered a symbolic number which has gained significant acceptance. I refer to [4] for a discussion of dual numbers in a setting of (total) arithmetical datatypes.

Including symbolic numbers like $\bot$ in an arithmetical algebra has not yet gained significant acceptance. The rationale for adopting $\bot$ is based on expected advantages of working with total functions, which is both formally simpler than working with partial functions, and is more closely linked to computer operation. Whether or not such advantages will materialise to such an extent that symbolic numbers in general, and $\bot$ in particular, will gain wider acceptance, remains to be seen.

1.2 Which functions to include in a signature

In software engineering the signature of a datatype names at least those constants and operations which a user, that is a programmer, is allowed or supposed to apply when making use of an implementation of the abstract datatype. This explanation provides a criterion as to whether or not to include certain operations in a signature.

When a datatype is used outside the context of software engineering, for instance for an explanation of some aspects of elementary arithmetic, another criterion for the inclusion of sorts constants and functions in a signature is required than mere actual or expected occurrence in computer programs. Like in software engineering, it is fruitful to presuppose some distinction between developer (author, designer) and user. For instance in elementary arithmetic one may think of students as users, and of the authors of course material as developers. Teachers feature a spectrum: some may be considered developers and others may best be considered users.

When working outside software engineering a datatype is presented together with its signature and perhaps together with a specification of it. Specifications often take the form of sets of equations, which are understood as rewrite rules from left (LHS) to right (RHS). A specification is called a DDRS (datatype defining rewrite system) if it is weakly terminating and ground confluent.

Importantly it will more often than not be the case that the designer/developer has access to operations which are not listed in the signature. The prime example of such an operation is the semantic function $[\![-]\!]$ which assigns an entity $[\![P]\!]$ in a sort $S_{\mathbb{A}}$ of the datatype $\mathbb{A}$ to a closed expression P of sort S over signature $\Sigma(\mathbb{A})$.

Moreover a datatype may come with notations for elements of its sorts which are then taken for constants in the signature at hand. In the case of arithmetical datatypes without division it is plausible to have decimal number notations $(-12, 0, 275$ etc.$)$ as constants. Upon taking natural numbers in decimal notation for constants, say $57+101 = [\![57+101]\!]$ becomes a valid assertion expressing the same assertion as $57 + 101 = 158$.

1.3 Which functions not to include in a signature

The choice of a signature in some context of interest is a matter of design, and different signatures may serve different but related objectives. For each application area, and for each approach to it which involves the introduction of datatypes, it is unavoidable that below some level of granularity constants and operations which might be important are nevertheless left outside a signature. The semantic function $[\![-]\!]$ has already been mentioned as an example of a function left outside a signature, but more appealing examples are easily found. For instance in the case of arithmetic one may consider the details of working with decimal notation. Decimal number notation uses lists of decimals and operations on these, which can be easily specified with appropriate auxiliary functions, which, however one may prefer not to expose to a user.

1.4 Datatypes and abstract datatypes

Conventionally a datatype is a particular kind of feature of a program notation, or of an instance of the use of that feature in a program, or a mechanism offered by an operating system which enables a program to make use of that kind of feature. Secondly it is a single sorted or many sorted minimal algebra, which is a notion from mathematical logic. In the practical literature on computer programming the first meaning of datatype is most prominent. However, I will use datatype in the second sense only.

Definition 1. *An algebra of signature Σ is minimal if each of its elements is the interpretation of a closed Σ-expression.*

Definition 2. *A datatype (of signature Σ) is a minimal Σ algebra with non-empty sorts only. A datatype is a datatype of some signature Σ.*

As stated above the literature on computer programming is not unanimous on this matter, and both definitions incorporate a definite choice, for which equally valid alternatives exist. Having made this choice, however, an abstract datatype is the isomorphism class of a datatype.

Definition 3. *An abstract datatype (of signature Σ) is the isomorphism class of a datatype of signature Σ. An abstract datatype is an abstract datatype of some signature.*

In addition to the mentioned ambiguity regarding the notion of a datatype there is additional ambiguity in the literature regarding the notion of an abstract datatype. Authors on the theory and practice of datatypes, who understand datatypes as a systems feature prefer to view an algebra as an abstract datatype. If, however, one perceives a datatype as an algebra then abstraction leads to the isomorphism class. These definitions comply with [15]. I will denote the isomorphism relation with $\cong$. For a datatype $\mathbb{A}$, the abstract datatype to which it belongs, i.e. the isomorphism class of $\mathbb{A}$ is denoted with $[\mathbb{A}]/\cong$.

1.5 Abstract datatype specification

Central to the theory and application of abstract datatypes is the notion of a formal specification which has originally been advocated by the ADJ group some 50 years ago (see e.g. [24]). Specification of abstract datatypes constitutes one of the roots of so-called formal methods in software engineering.

Definition 4. *A specification of an abstract datatype (of signature Σ) is an instance of use of any method of description which unambiguously singles out a unique abstract datatype of the given signature.*

Specifications are often classified in terms of the specification method which is used. Algebraic specifications allow the use of collections of equations and conditional equations to determine an appropriate class of algebras from which either by requiring initiality or by requiring finality, a subclass of mutually isomorphic algebras is singled out. (See e.g. [13]). First order specifications often allow to determine an isomorphism class of models by merely requiring minimality. If a specification determines a class of abstract datatypes rather than a single one, it is customary to speak of a loose specification.

Definition 5. *A loose specification of an abstract datatype (of signature Σ) is an instance of use of any method of description which unambiguously singles out a class of abstract datatypes of the given signature.*

1.6 Datatype specification

Algebraic specifications may sometimes be used to specify a datatype rather than merely an abstract datatype. If the equations constituting an algebraic specification of an abstract datatype, when considered as a term rewrite system, give rise to a weakly terminating and ground confluent term rewrite system (so that normal forms of closed expressions exist and are unique), then by choosing normal forms as elements a unique datatype is singled out.

In [14] it is shown that if auxiliary functions are admitted a computable datatype admits an equational specification which, by being confluent and terminating, specifies a unique datatype at the same time.

Definition 6. *A specification of a datatype (of signature Σ) is an instance of use of any method of description which unambiguously singles out a unique datatype of the given signature.*

Definition 7. *A loose datatype specification (of signature Σ) is an instance of use of any method which unambiguously singles out a class of datatypes of the given signature.*

For the objectives of this paper it is of vital importance to be precise about the distinction between datatypes and abstract datatypes.

2 Arithmetical datatypes: rationale and survey

The development of abstract datatypes in software engineering came with several claims about the potential benefits of the use of formal specification methods for abstract data types in software engineering. Together these claims provided a convincing rationale for research on algebraic aspects of datatypes and abstract datatypes. I will briefly summarise these expected benefits.

2.1 A rationale for abstract datatypes in software

The rationale for using datatypes and abstract datatypes in software technology decomposes into different aspects. (i) Viewing datatypes as algebras provides a path for the introduction of known methods from universal algebra for obtaining a semantic basis for concepts in computer programming. (ii) Parametrized datatypes, parameter instantiation, refinement, subsorts and partiality, error handling, the relation between specification and implementation, and formal verification of datatype implementations, constitute concepts of high relevance for software engineering and program notation design. The latter disciplines may both profit from the application of the conceptual framework of abstract datatypes. (iii) By using a formal specification method precise, detailed, and unambiguous descriptions of software modules can be provided. (iv) On the basis of one or more specification techniques quite general software specification languages may be designed, together with corresponding software development methods. (v) Term rewriting and logic programming often allow the generation of a prototype implementation of a datatype from an abstract data type specification. This aspect comes with a wealth of options for specialising and generalising term rewriting. Conditions, priorities, applications of pattern matching and selection of rewriting strategies come into play. (vi) Given an application area one may design a portfolio of specifically relevant abstract datatypes for that area and one may construct a special purpose program notation, a so-called domain specific language (DSL), around this portfolio which is optimised for the use of precisely those datatypes. DSL's allow so-called low coding.

2.2 The fate of algebraic methods in software engineering

By now these objectives have persisted for half a century and hundreds of small and large method engineering projects based on such principles have been carried out, each coming with a wealth of case studies and, often with realistic applications of various sizes. The mentioned objectives have each been met, while at the same time it seems fair to say that it has proven exceptionally hard to convince individual professional programmers of the idea that the significant investments in time and effort required when making use of these particular formal techniques is justified by its practical value.

2.3 A rationale for arithmetical datatypes in arithmetic

The introduction of arithmetical datatypes as a concept of use for elementary mathematics constitutes an attempt to channel back results from computer science to its original source. The following advantages may be obtained from doing so.

2.3.1 Introducing syntax into elementary arithmetic

Whereas for computing in general and for software engineering in particular the explicit use of syntactic considerations and concepts is common practice, the explicit use of syntax is still completely absent in school mathematics, except for the backdoor of educational software which unavoidably comes with some form of syntax. However, I claim that in the setting of elementary arithmetic taking syntax into account is useful at the conceptual level already. This claim is spelled out in more detail in the following paragraphs.

2.3.2 Conceptual clarification

Some aspects of elementary mathematics merit further attention and may profit from the clarity which taking syntax into account can provide. For instance there is a remarkable diversity in definitions of fractions, and of notions and distinctions related to fractions, which transpires from the vast literature on fraction teaching. Providing a systematic survey of these definitional options is supported by having a framework of arithmetical datatypes in mind.

2.3.3 Development of educational languages and systems

Thinking in terms of arithmetical datatypes may be helpful for the design, development, and implementation of educational software systems for teaching arithmetic as well as for comparing such systems. Such systems may either be specific for a preferred arithmetical datatype or may be generic for a spectrum of different underlying arithmetical datatypes.

2.3.4 Supporting the human understanding of automated arithmetic

Under the assumption that a computer implementation of elementary arithmetic is unavoidably based on a selection of particular arithmetical datatypes and implementations thereof, awareness of the diversity of such datatypes may prove helpful for the human understanding of the the software tools at hand.

2.3.5 Novelty and scope

Admittedly there is less novelty in moving focus from computer science back to basic arithmetic than there was 50 years ago in exporting results from universal algebra to computer programming. There is no teaching crisis in arithmetic that might be considered comparable to the software crisis for which formal methods were initially meant as a cure.

Teaching elementary arithmetic, however, may well constitute the largest teaching process world-wide with focus on a specific theme. Much effort goes into the design of improvements to that particular educational process. Therefore I consider the effort of investigating the foundations of school arithmetic from the perspective of arithmetical datatypes even a without specific application to the relevant teaching processes in mind, to be amply justified.

2.4 Application of (abstract) datatypes as an objective

The work in this paper may be understood from several different perspectives and may be judged accordingly in different ways. The following different objectives may lead to comparable considerations: focus on conceptual problems with fractions, while restricting the tools employed to a minimum, (ii) introducing syntactic aspects into elementary arithmetic so as to make the explicit use of syntax "school proof", (iii) integrating formal logic with elementary mathematics, (with as a consequence that working with arithmetical datatypes is merely an option, and not a must), and (v) redesigning the definitions of various arithmetical datatypes, finding optimal signatures, etc.

Contemplating the question "how to make use of datatypes and abstract datatypes in school arithmetic", the subsequent question arises which aspects of the theory of (abstract) datatypes may be of relevance in the setting of elementary arithmetic. A partial answer to this question is given in the following Paragraphs. In the technical part of this paper, involving the definition of fracterms and the survey/design of fracterm terminology only the first item (signature design) is exploited. A signature is used to provide a precise language for speaking of fracterms.

2.4.1 Signature design and informal description of datatypes

Making signatures explicit (or rather designing a signature) constitutes a major designs phase which (abstract) datatype theory prescribes and requires. Precisely this phase is less familiar for mathematicians. Having a signature at hand both formal and informal methods for describing a datatype with that signature are available.

2.4.2 Abstract datatype specification

For all arithmetical (abstract) datatypes under consideration a survey of algebraic/formal specifications thereof is relevant. In particular it matters whether or not finite specifications exist and to what extent good term rewriting properties can be obtained when reading the equations from left to right as rewrite rules. Confluence and (weak) termination, perhaps modulo permuting rewrite rules (such as commutativity and associativity), are the main virtues for rewrite systems which a designer hopes to achieve. These virtues first of all support the theoretical understanding of the (abstract) datatypes involved.

2.4.3 Infinite signatures and schematic (conditional) equations

This feature seems not to have gained any prominence in the computer science literature. It is unproblematic and potentially useful to introduce infinite signatures in the context of algebraic specifications. Below I will introduce a constant σ for each non-zero decimal natural (a decimal digit sequence starting with a non-zero digit; such digit sequences are collected in the set C_D), and I will make use of schemes involving metavariables $u, v, w, \ldots$ ranging over C_D for equations and conditional equations which translate into infinite collections of equations or conditional equations by substituting all (combinations of) constants $\sigma \in \mathsf{C}_\mathsf{D}$ for these metavariables. The existence of initial algebras and of final algebras under certain restrictions, as well as the theory of term rewriting are unaffected by the use of infinitary specifications in the way just outlined.

2.4.4 Datatype specification and prototyping

Adequate term rewriting properties for a datatype specification (i.e. having developed a datatype defining abstract datatype specification) leads to a so-called executable specification. Once an executable datatype specification has been obtained automatic prototyping allows practical experimentation with it, in advance of further optimization of an implementation.

2.4.5 (Abstract) datatype design by stepwise refinement

A classic idea is that complex (abstract) datatypes are designed on the basis of simpler one's by way of a sequence of enrichments. In an enrichment the signature is enriched and additional equations for specification are introduced thereby obtaining a refinement of a specification. Enrichment may be understood as an instance of parameter passing for a parametrised datatype (see below).

2.4.6 Final algebra semantics

An awareness of the potential gap between initial algebra semantics (involving negation by failure) and final algebra semantics (involving double negation by failure) may be useful. Often final algebra semantics is closer to the intuition, in spite of the fact that it is technically more involved. Final algebra semantics matches perfectly with co-algebras.

2.4.7 Hidden sorts, constants and functions

Below a datatype for rational numbers will be described with elements (a, b) where a and b are integers, $b > 0$ and $\gcd(a, b) = 1$. It is an option to view entities (a, b) as hidden constants. That implies that its use is not advised for an end user of the system (i.e. a student) but that awareness of the presence of the constant may be helpful for someone in charge of delivering the system, for instance a teacher, or a person designing educational material. Technically more involved is the option to provide constants and functions that specify addition and multiplication on decimal notations. Such parts of a signature may be declared hidden after having been introduced in order to obtain a higher level of abstraction.

2.4.8 Modularity and parametrization

Below it will be assumed that multiplication and addition for integers "is known" so that a focus on division and subtraction as newly introduced can be made independent of differences in perspective on addition and multiplication of integers. This matter can be understood as a modular combination of various specifications (now called specification modules or simply modules), a specification of addition and multiplication, and a specification for division which merely makes use of names for addition and multiplication. Module algebra can be used to deal with the combination of specifications involving hidden sorts and functions, while the pushout construction of category theory provides a tool to describe parameter passing.

2.4.9 Error handling

The most straightforward option, as has emerged in (abstract) datatype theory, for dealing with problematic values is to introduce an absorptive element in each sort, often denoted with $\perp$, or $\perp_S$ if the notation must be made specific for a sort S. Below this strategy is applied to elementary arithmetic in order to deal with the problematic value of the expression $1/0$. However, a large literature concerning different options for dealing with problematic values has been developed in the context of (abstract) datatypes. And other options for dealing with the evaluation of problematic expressions may be just as well useful for elementary arithmetic.

3 Arithmetical datatypes with a single absorptive symbolic value

For the sequel of the paper I will consider the following particular instance of a field of rational numbers, denoted $\widehat{\mathbb{Q}}$. $\widehat{\mathbb{Q}}$ has the signature Σ_{ru} of unital rings. The domain $|\widehat{\mathbb{Q}}|$ consists of pairs $(0, 1), (n, m)$ and $(-n, m)$ with n and m natural numbers in decimal notation without leading zeros, and such that

$\gcd(n,m) = 1$. Below I will provide some comments on decimal notation which as such exceeds the grammar provided by Σ_{ur}.

The interpretation of the constant 0 is $(0,1)$, the interpretation of the constant 1 is $(1,1)$, and addition, multiplication and opposite are defined in the usual manner while an enrichment with subtraction is defined via $x - y = x + (-y)$. By writing $\widehat{\mathbb{Q}}$ rather than $\mathbb{Q}$ it is made explicit that a specific datatype is meant, including a definite choice for its domain as a set, rather than an abstract datatype, i.e. a mere isomorphism class, or a fixed but arbitrary choice from an isomorphism class.

Using semantic bracket notation, as has become customary in computer science, in $\widehat{\mathbb{Q}}$ it is the case that $[\![4/6]\!] = (2,3)$ and so on. In logical notation one may write $(\widehat{\mathbb{Q}} \models 4/6) = (2,3)$, where in the expression $4/6$ 4 abbreviates $1+(1+(1+1))$ etc. Neither of these notations will not be used below, however.

$\widehat{\mathbb{Q}}$ is an arithmetical algebra whereas it is not an arithmetical datatype because it is not a minimal algebra. For instance $(1,2)$ is not the interpretation of a closed term over the signature of unital rings. In order to obtain an arithmetical datatype which includes $\widehat{\mathbb{Q}}$, introducing an enrichment with one or more functions is unavoidable, while extension with one or more elements is optional. I will consider various forms of division as candidates for enrichment for the purpose of finding a minimal algebra which is closely related to $\widehat{\mathbb{Q}}$.

3.1 Datatypes for rationals: expansions and extensions of $\widehat{\mathbb{Q}}$

The introduction of division in the context of $\widehat{\mathbb{Q}}$ comes with a range of different options.

3.1.1 Meadows of rational numbers

$\widehat{\mathbb{Q}}_0$, a meadow of rational numbers with inversive notation is obtained by expanding $\widehat{\mathbb{Q}}$ with an inverse function $^{-1}$ given by: $(0,1)^{-1} = (0,1)$, $(n,m)^{-1} = (m,n)$, and $(-n,m)^{-1} = (-m,n)$. An initial algebra specification of $\mathbb{Q}_0 = \widehat{\mathbb{Q}}_0/\cong$ is given in [15]. The abstract datatype $\mathbb{Q}_0$ occurs in [26] and in [32], where such structures are named pseudo-fields. Implicitly $\mathbb{Q}_0$ also occurs in [29].

Several definitions of the class of meadows can be given, for instance meadows constitute the smallest variety (i.e. class of algebras that satisfy an equational first order theory) which includes the commutative rings expanded with an inverse operator that satisfies: $0^{-1} = 0$ and $\forall x(x = 0 \vee x \cdot x^{-1} = 1)$.

Yet another way to focus on meadows is to work in the signature of unital rings expanded with an inverse operation such that $0^{-1} = 0$ and to adopt the following additional proof rule IR_m (inverse rule for involutive meadows) for (conditional) equational logic (E is a collection of equations and or conditional equations) :

$$\frac{E \vdash x = 0 \to t = r, E \vdash x \cdot x^{-1} = 1 \to t = r}{E \vdash t = r} \quad \mathsf{IR}_m$$

3.1.2 Meadows of rational numbers with divisive notation

$\widehat{\mathbb{Q}}_0^d$ is the meadow of rational numbers with divisive notation which results from $\widehat{\mathbb{Q}}_0$ by first applying an enrichment with $x/y = x \cdot y^{-1}$ and subsequently applying a reduction by forgetting the inverse function. One may consult [8] for more information on the connections between inversive notation and divisive notation, and for an initial algebra specification of $\mathbb{Q}_0^d$ which avoids the detour with the inverse function. The additional proof rule (now referred to as IR_m^d) reads:

$$\frac{E \vdash x = 0 \to t = r, x/x = 1 \to t = r}{E \vdash t = r} \quad \mathsf{IR}_m^d$$

3.1.3 Common meadows of rational numbers

$\widehat{\mathbb{Q}}_\perp^d$ is a common meadow of rational numbers with divisive notation. The domain of this structure extends that of $\widehat{\mathbb{Q}}$ with $(0,0)$ which serves as the intepretetaion of the constant $\perp$ and which represents the result of dividing by zero (i.e. $x/0 = \perp$). $(0,0)$ is an absorptive element in the algebra (so that $x + \perp = x \cdot \perp = \perp$ holds in $\widehat{\mathbb{Q}}_\perp^d$). The signature of $\widehat{\mathbb{Q}}_\perp^d$ is $\Sigma_{\mathsf{urd},\perp} = \Sigma_{\mathsf{urd}} \cup \{\perp\}$. The (conditional) equational logic for common meadows results from adopting the following additional rule IR_{cm}^d

$$\frac{E \vdash x = 0 \to t = r, x = \perp \to t = r, x/x = 1 \to t = r}{E \vdash t = r} \quad \mathsf{IR}_{cm}^d$$

$\widehat{\mathbb{Q}}_\perp$ is a common meadow with inversive notation. Common meadows as an abstract datatype, are introduced in [10] where equations for common meadows are given, named $\mathsf{Md}_{\mathbf{a}}$ with $\mathbf{a}$, instead of $\perp$ as the name for the absorptive element. An initial algebra specification of $\mathbb{Q}_\perp$ is given in [11], where also an independent construction is given of the initial algebra of Md_a.

In [10] the symbol $\mathbf{a}$ is used for the absorptive element with the idea that the latter notation is (would be) preferable if calculation with an absorptive element is (hypothetically) put into practice, say in school arithmetic, while using $\perp$ is preferable in theoretical work which is not supposed to prepare for "end user" application in a literal manner.

3.1.4 Wheels of rational numbers

$\widehat{\mathbb{Q}}_{\infty,\perp}^d$ is a wheel of rational numbers with divisive notation. This structure introduces in addition to $(0,0)$ an element $(1,0)$ which serves as unsigned infinity (with the corresponding constant written ∞), the value of $1/0$. Now both $(0,1) \cdot (1,0)$ and $(1,0) + (1,0)$ produce the absorptive element $(0,0)$, which also results as the value of $0/0$. For more information on wheels I refer to [37] and [16].

An equational logic for wheels can be presented by adopting the following additional rule IR_w^d

$$\frac{E \vdash x = 0 \to t = r, x = \perp \to t = r, x = \infty \to t = r, x/x = 1 \to t = r}{E \vdash t = r} \quad \mathsf{IR}_w^d$$

3.1.5 Transrational numbers

$\widehat{\mathbb{Q}}^d_{\pm\infty,\Phi}$ is an arithmetical datatype for transrationals with divisive notation. This arithmetical datatype extends $\widehat{\mathbb{Q}}$ with two elements which will play the role of signed infinite values $(1, 0)$ and $(-1, 0)$, and enriches the resulting structure with a division operator. For more information transrational arithmetic I refer to [1] and [19] and the papers cited therein.

In the setting of transrational arithmetic and transreals ([19]) instead of $\perp$ the symbol Φ is used for the unique absorptive element. Following [2], Φ is used below in case positive and negative infinite values are both present and are distinguished, which is the case in transrationals and transreals, while $\perp$ is used in the case of a single infinite element (as in wheels), and also in the presence of an absorptive element in the absence of infinite values. A dedicated proof rule is as follows, writing e for $t = r$: IR^d_{tr}

$$\frac{E \vdash x = 0 \to e, x = \Phi \to e, x = \infty \to e, x = -\infty \to e, x/x = 1 \to e}{E \vdash e} \quad \mathsf{IR}^d_{tr}$$

3.1.6 True fractions

In [12] the notion of a true fraction is proposed, which refers to elements arithmetical datatypes in which different though equivalent common fracterms have different interpretations. Datatypes with true fractions are complicated structures which, for that reason are unlikely candidates for being used for educational purposes.

3.2 Focus on $\mathbb{Q}^d_\perp$: common meadows of rational numbers

From the structures listed above I consider the abstract arithmetical datatype $\mathbb{Q}^d_\perp$, to be the most appropriate one as a basis for the description and analysis of elementary arithmetic in an educational setting.

$\mathbb{Q}^d_\perp$ models an understanding of division as a partial function, while technically working with total functions, especially when reading $t = \perp$ as "t is undefined", or as "t has no proper value".

The choice for $\widehat{\mathbb{Q}}^d_\perp$ as a particular arithmetical datatype within $\mathbb{Q}^d_\perp$ is made for the purposes of this paper only. This choice embodies an important degree of freedom for the analysis of elementary arithmetic, and different choices may serve different purposes in that area. Providing a systematic survey of common meadows of rational numbers is left for further work.

Technically the arithmetical datatype $\widehat{\mathbb{Q}}^d_0$ may equally well be used in the context of elementary arithmetic, but the somewhat controversial assumption that $1/0 = 0$ lacks sufficient justification at an elementary level. Advantages of working with involutive meadows (and thereby assuming $1/0 = 0$ when using divisive notation) may appear primarily in two ways: (i) when developing theoretical work on formal systems involutive meadows are simpler to deal with than its alternatives, and (ii) when working on applications expressions and equations may be kept simpler. In rare cases the choice of 0 as a value of $1/0$

matches with the intuitions in the setting at hand, but such cases are somehow accidental, and therefore do not, in my view, provide significant arguments for setting $1/0$ equal to 0. In this matter I disagree with the views that underly [29], where it is claimed that taking 0 as the value of $1/0$ is preferable to other choices on mathematical grounds, that is because it provides the best fit with relevant parts of mathematics.

If a conceptually more sophisticated alternative is sought for $\widehat{\mathbb{Q}}^d_{\perp}$ serving as a basis for work on elementary arithtmetic then wheels of rational numbers, i.e. the abstract arithmetical datatype $\mathbb{Q}^d_{\infty,\perp}$ provides a plausible candidate. In [16] a more elegant and more generally applicable construction for specific algebras in this abstract datatype than the datatype $\widehat{\mathbb{Q}}^d_{\infty,\perp}$ as defined above is presented.

$\widehat{\mathbb{Q}}^d_{\pm\infty,\Phi}$ becomes relevant in diverse circumstances: (a) if close proximity to computer arithmetic matters, (b) if an application requires and ordering on symbolic values, (c) in applications involving reals, topology, and analysis.

3.2.1 Decimal notation

It is plausible to introduce all decimal digit sequences without redundant leading zeros as constants and to provide algebraic specifications for naturals and integers consisting of an infinite number of (conditional) equations which are generated by substitution of constants in a finite number of schemes. Such specifications can be combined without any problem with conventional algebraic specifications of rings, meadows, and common meadows.

I write $\mathsf{C_D}$ for the collection of all non-empty digit sequences different from 0 without leading zeros. For instance $70553 \in \mathsf{C_D}$. With $u = 7055 \in \mathsf{C_D}$ one may write $u3$ for 70553. The decomposition of an element w of $\mathsf{C_D}$ with two or more characters into $w \equiv ud$ with $u \in \mathsf{C_D}$ and $d \in \{0, \ldots, 9\}$ will be used extensively. Now for instance 752 is a constant which can be made up from 7 by first postfixing 5 and the 2.

$\mathsf{C_D}$ is considered the collection of positive natural numbers, or rather of positive decimal natural numbers. The algebra $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C_D}}$ results from $\widehat{\mathbb{Q}}^d_{\perp}$ by including each $\sigma \in \mathsf{C_D}$ in the signature (which already contains 0 and 1) and by choosing as the interpretation for $\sigma \in \mathsf{C_D}$ the pair $(\sigma, 1)$. An algebraic specification of the abstract datatype $\mathbb{Q}^d_{\perp,\mathsf{C_D}}$ is given in table 1.

The specification in table 1 is found by combining (i) the specification of common meadows of [10] (though writing $\perp$ instead of $\mathbf{a}$) with (ii) an explicit definition of division in terms of the inverse function ($x/y = x \cdot y^{-1}$), (iii) an axiom scheme ($u/u = 1$ for $u \in \mathsf{C_D}$) which captures the requirement that the characteristic is zero, in the light of the fact that elements of $\mathsf{C_D}$ are supposed to represent nonzero decimal natural numbers, and (iv) a specification of decimal numbers by means of an infinite set of equations which consists of ten equations for the successor of digits and an equation scheme which specifies the value of ud in terms of the values of u and of d. In this specification inverse merely serves as an auxiliary function. Following [11] the axiom scheme $u/u = 1$ may be replaced by the single axiom $(x^2 + y^2 + 1)/(x^2 + y^2 + 1) = 1 + 0 \cdot (x + y)$. Following [8], the use of inversive notation can be avoided. I have not done so in the presentation of the axioms in order to keep the relation with equational axiom systems in [10] transparent.

By having constants for all decimal natural numbers, for natural numbers at least, no distinction between syntax and semantics is made, and say 173 considered a natural number rather than merely a notation for a natural number.

So one works with decimal natural numbers rather than with natural numbers, with the understanding that binary numbers and hexadecimal numbers are other entities, which however may have the same value as decimal numbers. Importantly stating that two entities having the same value is admissible even if no collection of values has been defined. "Having the same value" is merely a way of indicating a certain equivalence relation. It should be noticed that the decimal notation cannot simply be mixed with, say, binary notation. Indeed table 1 asserts $10 = 9 + 1$ which will not hold for binary natural numbers where $10 = 1 + 1$ instead.

Whole decimal numbers extend decimal natural numbers with signed entities, except for 0 which is unsigned. Below I will work with decimal wholes with the understanding these are signed digit sequences without redundant leading zeroes, and I will in most cases speak of wholes, (natural numbers, integers etc.) without labelling these explicitly as decimal.

This identification of non-negative whole numbers with decimal sequences may, but need not, be made. It is an instance of association as suggested by Nicaud et. al. in [31]. Below I will assume that on decimal whole numbers an ordering $<$ is present.

3.2.2 Schematic equations

An equation $t(u, v) = r(u, v)$ involving, say, two meta-variables u and v represents a schematic equation which abbreviates the following set of equations $\{t(\sigma, \sigma') = r(\sigma, \sigma') | \sigma, \sigma' \in \mathsf{C_D}\}$. Similarly a schematic conditional equation $t_1(u, v) = r_1(u, v) \wedge \cdots \wedge t_n(u, v) = r_n(u, v) \to t(u, v) = r(u, v)$ represents the collection of its substitutions instances for all $\sigma, \sigma' \in \mathsf{C_D}$. An equation of the form $t(d, u) = r(d, u)$ for $d \in \{0, \ldots, 8\}$ stands for a collection of 9 schematic equations: $t(0, u) = r(0, u), \ldots, t(8, u) = r(8, u)$. Let $(\Sigma_{\mathsf{C_D}}, E)$ be a (conditional) equational specification consisting of finitely many schemes, then $\mathsf{Mod}(\Sigma_{\mathsf{C_D}}, E)$ has a semi-computable initial algebra.

Disadvantages of replacing schematic (conditional) equations by the additional syntax as described are these: (a) unless conventions (vi) and (vii) are ignored a non-trivial exposition about type inference must precede the use of this form of specification, (b) a conditional equation $(i(x) = i(y) \to x = y)$ enters the picture which is not usable as a rewrite rule, (c) working with schemes and metavariables over $\mathsf{C_D}$ mimics a subsort rather than an additional sort. The idea that $\mathsf{C_D}$ constitutes a subsort of V is closer to the intuition than that it constitutes an additional sort with elements that must be copied into V.

The device of schematic equations may be understood as one of several methods to avoid the use of a subsort, thereby avoiding the well-known and non-trivial complications of equational logic in the presence of subsorts. The

$$(x+y)+z = x+(y+z) \tag{1}$$
$$x+y = y+x \tag{2}$$
$$x+0 = x \tag{3}$$
$$x+(-x) = 0\cdot x \tag{4}$$
$$(x\cdot y)\cdot z = x\cdot(y\cdot z) \tag{5}$$
$$x\cdot y = y\cdot x \tag{6}$$
$$1\cdot x = x \tag{7}$$
$$x\cdot(y+z) = x\cdot y + x\cdot z \tag{8}$$
$$-(-x) = x \tag{9}$$
$$0\cdot(x\cdot x) = x \tag{10}$$
$$(x^{-1})^{-1} = x + 0\cdot x^{-1} \tag{11}$$
$$x\cdot x^{-1} = 1 + 0\cdot x^{-1} \tag{12}$$
$$(x\cdot y)^{-1} = x^{-1}\cdot y^{-1} \tag{13}$$
$$1^{-1} = 1 \tag{14}$$
$$0^{-1} = \bot \tag{15}$$
$$x+\bot = \bot \tag{16}$$
$$x/y = x\cdot y^{-1} \tag{17}$$
$$u/u = 1 \quad (\text{for } u \in \mathsf{C_D}) \tag{18}$$
$$1+1 = 2 \tag{19}$$
$$2+1 = 3 \tag{20}$$
$$3+1 = 4 \tag{21}$$
$$4+1 = 5 \tag{22}$$
$$5+1 = 6 \tag{23}$$
$$6+1 = 7 \tag{24}$$
$$7+1 = 8 \tag{25}$$
$$8+1 = 9 \tag{26}$$
$$9+1 = 10 \tag{27}$$
$$ud = 10\cdot u + d \quad (\text{for } d \in \{0,\ldots,9\}, u \in \mathsf{C_D}) \tag{28}$$

Table 1: $\mathsf{CM}^d_{\mathsf{C_D}}$: an initial algebra specification of the abstract datatype $\mathbb{Q}^d_{\bot,\mathsf{C_D}}$ using inverse as an auxiliary function

motivation for this preference for infinite signatures and schemes is that, in contrast with the use of subsorts, working with infinite signatures and infinite sets of (conditional) equations requires no significant adaptations of the theory of algebraic specifications.

3.2.3 The arithmetical abstract datatype $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$

Having available decimal number constants the datatype $\widehat{\mathbb{Q}}^d_{\perp}$ can be expanded to include constants in C_D by merely providing the interpretation $(u, 1)$ for constant u. This expansion creates the arithmetical datatype $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$, and the corresponding arithmetical abstract datatype $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$ For applications in elementary arithmetic the abstract arithmetical datatype $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$ constitutes a a plausible basis for further investigation. An algebraic specification for it is $\{\_^{-1}\}\,\Delta\,\mathsf{CM}^d_{\mathsf{C}_\mathsf{D}}$ with $\_\Delta\_$ the module algebra hiding operator (see [7]), and $\mathsf{CM}^d_{\mathsf{C}_\mathsf{D}}$ the specification of table 1. The specification $\{\_^{-1}\}\,\Delta\,\mathsf{CM}^d_{\mathsf{C}_\mathsf{D}}$ is logically equivalent ($\equiv_{fol}$ in terms of module algebra) with the specification $\mathsf{CM} + \chi = 0$ from [10]. For a field K the common meadow $K_\perp$ is defined by adding a new element $\perp$ to its domain, introducing division so that $1/0 = \perp$ and assuming that addition and multiplication are strict on $\perp$. Each algebra of the form $K_\perp$ can be equipped with interpretations for the constants C_D so that it satisfies $\{\_^{-1}\}\,\Delta\,\mathsf{CM}^d_{\mathsf{C}_\mathsf{D}}$ and moreover (the basis theorem from [10]) each equation that is valid in all $K_\perp$ is derivable from $\{\_^{-1}\}\,\Delta\,\mathsf{CM}^d_{\mathsf{C}_\mathsf{D}}$.

3.2.4 Datatypes in the abstract datatype $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$

Returning to the datatype $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$ one may notice that $[\![2]\!] = (2, 1)$ introduces an unnecessary overhead. Choosing a different datatype in $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$ may overcome that disadvantage.

The datatype $\widetilde{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}} \cong \widehat{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$ is obtained from $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$ by means of the following transformation, in fact a homomorphism, of its elements:

- $(0, 0) \to \perp$,
- $(0, 1) \to 0$,
- $(u, 1) \to u$,
- $(-u, 1) \to -u$,
- $(u, v) \to u/v$ for $v \neq 1$,
- $(-u, v) \to -u/v$ for $v \neq 1$.

$\widetilde{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$ i,s just as $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C}_\mathsf{D}}$, merely one of many arithmetical datatypes which implement (i.e. are a member of) the abstract datatype $\mathbb{Q}^d_{\perp,\mathsf{C}_\mathsf{D}}$.

Another construction for a datatype in $\mathbb{Q}^d_{\perp,\mathsf{C_D}}$ is results form the unique congruence $\equiv_\perp$ on $T_{\Sigma_{\mathsf{urd},\mathsf{C_D},\perp}}$ (the closed terms for signature $\Sigma_{\mathsf{urd},\mathsf{C_D},\perp}$) for which $T_{\Sigma_{\mathsf{urd},\mathsf{C_D},\perp}} / \equiv_\perp \cong \widehat{\mathbb{Q}}^d_{\perp,\mathsf{C_D}}$.

Yet another option is to have the domain of the datatype consisting of equivalence classes $[(n,m)]_\equiv$ $(n, m \in \mathbb{Z}, m > 0)$ of the equivalence relation $[n,m] \equiv [z,b] \iff n \cdot b = m \cdot a$, augmented with $\{(1,0)\}$ as the interpretation of $\perp$.

4 Fracterms and fracterm related terminology

Fracterms were introduced as a class of expressions in [2]. The notion of a fracterm admits a reasonably clear definition as stated in definition 8 below. Loose ends of this definition relate to variation in the notion of an arithmetical signature as well as in the flexibility in the precise syntax of terms one prefers to adopt.

Definition 8. *A fracterm is a finite expression over an arithmetical signature (with division) with the division operator as its leading function symbol.*

A fracterm may contain variables $x, y, z, ..$ ranging over the domain V (so-called ordinary variables), as well as metavariables $u, v, w, ..$ for decimal number constants (elements of $\mathsf{C_D}$ if $\mathsf{C_D}$ is part of the signature) and postfix extended metavariables (e.g $u7, v08$ and $w112$).

Definition 9. *A fracterm is closed if it contains no variables and no metavariables, otherwise it is non-closed. A fracterm is half-closed if it contains no ordinary variables.*

Open fracterms are just fracterms, though with the attribute open one indicates that the fracterm might contain one or more variables. Half-open fracterms may contain zero or more metavariables and postfix extended metavariables. This aspect of the terminology only matters if meta-variables for fracterms (or for components of fracterms) are used.

In the presence of different notations for division the same fracterm may be written in different ways: $\frac{2+1}{3+5}$ and $(2+1)/(3+5)$ will be considered the same fracterms. Stated differently, different fracsigns may represent the same fracterms.

Remarkably the notion of a fracterm is trivial and problematic at the same time. The existence of fracterms is self-evident once signatures including a division operator are taken for granted. However, in the absence of syntactic considerations the notion of a fracterm becomes less obvious. Nevertheless I suggest that the notion of a fracterm, as well as the word fracterm for denoting such expressions be included in the language of elementary mathematics, where both the notion, which evidently exists without being named, and its name (fracterm) may serve a useful purpose in clarifying the concept of a fraction. In this paper I will only discuss fracterms and I will leave an investigation of fractions for future work.

4.1 Fracterms versus fractions

One might object to the introduction of fracterms as additional terminology by claiming fracterms are just fractions, but that is not the case. The essential observation is that in mathematical use, and for a significant "fraction" of the educational literature, a fraction is a number, a value, i.e. an unstructured entity. I see no virtue in providing a definition of fractions which excludes the latter use of the term "fraction", as if that were wrong after all, and instead I propose to speak of fracterms which are definitely not numbers, if structured entities with a numerator and a denominator are meant.

4.2 Terminology for fracterms

I will now provide an adaptation to fracterms of the traditional terminology regarding fractions. The fraction terminology which I take as the point of departure includes: common fraction, (equivalently simple fraction, vulgar fraction), unit fraction, proper fraction, improper fraction, simplified fraction, fraction equivalence, fraction simplification, pseudo fraction (from the German *Scheinbruch*), composite fraction, like fractions, and mixed fraction. These notions, understood as predicates on fractions, are not easy to define in a precise manner because of dependency on the meaning of fraction, because fractions are not easy to define. Replacing fraction by fracterm, and assuming evaluation of expressions in a common meadow, a new list of phrases results each of which can be given a precise definition.

In addition to the notions inherited from fraction terminology the following fracterm related phrases are defined: quotient, problem fracterm, regular fracterm, fraction addition, resulterm, and simplified resulterm. Assuming characteristic zero the following elements of fracterm terminology are reasonable:

1. A wholeterm is a term over the signature of rings.

   (A wholeterm may be open or closed.)

2. For a closed fracterm P/Q the value of P/Q in $\widehat{\mathbb{Q}}^{d}_{\perp,\mathsf{C_D}}$ is called the quotient of P and Q (also referred to as the quotient of the fracterm P/Q).

3. The numerator of a fracterm P/Q is P.

   (Each fracterm has a unique numerator.)

4. The denominator of a fracterm P/Q is Q.

   (Each fracterm has a unique denominator.)

5. A singular fracterm is a fracterm P/Q with the property that $\widehat{\mathbb{Q}}^{d}_{\perp,\mathsf{C_D}} \models Q = 0$.

   The archetypical singular fracterm is $1/0$. $(2+3)/((4-5)+1)$ is a singular fracterm. $1/(x-x)$ is an open singular fracterm.

6. A fracterm is regular if it is not a singular fracterm and it has no singular fracterm as a subterm.

7. A fracterm is irregular if it is not regular.

8. A common fracterm (alternatively called simple fracterm or vulgar fracterm) is a closed fracterm of the form P/Q where P and Q are elements of C_D and Q is nonzero.

   (A common fracterm is regular by definition.)

9. A fracterm which is not common is uncommon.

10. A half-open common fracterm is a closed fracterm of the form P/Q where P and Q are 0 or are elements of C_D, or metavariables for C_D or postfix extended metavariables for C_D, with Q nonzero.

    Examples: $17/u2, u16/v554, u/v61, u/2, u/v$.

    (A half-open common fracterm is regular by definition.)

11. A negated common fracterm is an expression of the form $-(P/Q)$ with P/Q a common fracterm. (A negated fracterm is not a fracterm.)

12. An integer fracpair is a regular closed fracterm P/Q such that P is in C_D or in $-\mathsf{C}_\mathsf{D}$ and Q is in C_D.

    (So $(-2)/3$ is an integer fracpair while it is not a common fracterm and neither is it a negated common fracterm; $-2/3$ is a negated common fracterm, not a fracterm, with the same value as $(-2)/3$. Integer fracpairs are studied in detail in [11], and in a different manner in the theory of wheels ([37], [16]).)

13. A half-open integer fracpair is like an integer fracpair but now the numerator may also be a signed metavariable or a postfixed metavariable and the denominator may be a metavariable or a postfixed metavariable.

14. A rational fracpair is a closed fracterm P/Q. Two rational fracpairs P/Q and R/S are the same $P/Q \equiv_{\mathsf{rfp}} R/S$ if $P = R$ and $Q = S$. For instance $2/3 \not\equiv_{\mathsf{rfp}} 4/6$ and $2/3 \equiv_{\mathsf{rfp}} (4/2)/(6/2)$.

15. Two common fracterms P/Q and R/S are equivalent if $P \cdot S = Q \cdot R$.

    Equivalent fracterms are equal (i.e. have the same value) and there is no other notation for fracterm equivalence than the equality sign, $P/Q = R/S$, which expresses about two fracterms just "having the same value" ("having the same quotient"), which is the intended meaning of equivalence.

    The relation between fracterm equivalence and fracterm equality is tricky. The idea is as follows: (i) that two fracterms which are not the same may be equivalent, (ii) fracterm equivalence can be defined without making use of the notion of the value of a fracterm, (iii) after some preparations it may be concluded that calling equivalent fracterms equal works in the same way as asserting that $2 + 2 = 1 + 3$. Both sides have the same value, but are not the same expressions (sumterms), (iv) for quotients the distinction between sameness and equality is not made, quotients being values.

16. Fracterm equivalence extends to half-open common fracterms, in which case the condition $P \cdot S = Q \cdot R$ amounts to a universal quantification over all instances of both sides when substituting decimal numbers for the respective metavariables. The validity of such identities is decidable.

17. Two common fracterms P/Q and R/S are the same (written $P \equiv Q$) if $P \equiv R$ and $Q \equiv S$.

18. A proper fracterm is a common fracterm for which the numerator is smaller than the denominator.

19. A unit fracterm is a proper fracterm for which the numerator equals 1.

20. A fracterm is improper if it is common and not proper.

21. A pseudo fracterm (*Scheinbruch* in German) is a common fracterm of which the value of the numerator is a multiple of the value of denominator.

22. A fracterm P/Q is composite if either P or Q (or both) contains an occurrence of division.

23. A fracterm P/Q is simplified if it is common and moveover $\gcd(P, Q) = 1$.

24. A common fracterm P/Q allows simplification if it is not simplified.

25. A fracterm P/Q is flat if both P and Q are expressions over the signature of rings.

    (A flat fracterm is a non-composite fracterm. In [9] simple fraction is used instead of flat fraction under the assumption that no confusion with making reference to fraction simplification arises, but were I will comply to the convention that a simple fraction has no free variables.)

26. A flat (non-common) fracterm P/Q allows simplification if for some whole number $k \neq 0, 1$ P and Q are product expressions both having k as a factor.

    (This definition is at the border of informality as no definition of product expressions will be provided. It is plausible to require that in product expressions redundant brackets are removed while assuming association of multiplication to the right.)

27. A common fracterm P/Q is a simplification of fracterm R/S if R/S is a common fracterm, both fracterms are equivalent, and $Q < S$.

28. A negated common fracterm $-(P/Q)$ is a simplification of negated common fracterm $-(R/S)$ if the fracterms P/Q and R/S are equivalent, and $Q < S$.

29. A whole n is the simplification of the common fracterm P/Q if $P = n \cdot Q$.

30. Two fracterms P/Q and R/S have a common denominator (are like) if Q and S are the same. (Thus $2/(1+1)$ and $(1+1)/(1+1)$ are like but $(1+1)/2$ and $(1+1)/(1+1)$ are unlike (i.e. not like).)

31. For fracterms P/Q and R/S the task of addition consists of finding a fracterm U/V such that $P/Q + R/S = U/V$.

    (It is customary that addition of flat fracterms results in a flat fracterm and that addition of closed fracterms results in a closed fracterm. Addition of fracterms is not a function on pairs of fracterms, it is a function on quotients instead).

32. Viewed as a transformation on pairs of fracterms addition is a nondeterministic function (that is a relation, sometimes also called a multivalued function) and it is not a function.

    Addition of fracterms satisfies $P/Q + R/S = (P \cdot S + Q \cdot R)/(Q \cdot S)$ without any restriction on R or on S. However, it is not the case that $P/Q + R/S \equiv T/U$ for any fracterm T/U.

33. The most well-known rule of calculation for fracterms is the so-called quasi-cardinality rule (QCR) which indicates how to add two like common fracterms thereby obtaining a flat (but uncommon) fracterm with the same denominator. Quasi cardinality rule is a phrase ascribed to Griesel [23] in Padberg [33]. QCR may be formulated as an arithmetical fact with variables P, Q, R ranging over integers, with R non-zero.

    Proposition 4.1. *The sum $\alpha+\beta$ of two like common fracterms $\alpha \equiv P/R$ and $\beta \equiv Q/R$ has the same value as the flat fracterm $(P+Q)/R$. In equational form: $P/R + Q/R = (P+Q)/R$.*

34. Simplification of an uncommon fracterm amounts to the simplification one or of both of its components: thus the expression $(7+(3+(4+0)))/2$ can be simplified by calculating its numerator thus yielding $14/2$.

    Simplification of an expression does not necessarily produce a "simplified expression" according to some definition of being simplified. If a notion of calculation is given and P can be calculated (evaluated by way of calculation) to P' then the fracterm P/Q can be calculated to P'/Q.

35. A fracterm can be simplified if either it is a common fracterm and it can be simplified on the basis of item 24 or it is an uncommon fracterm and it can be simplified on the basis of item 34.

36. A mixed fracterm is a closed expression of the form $P\ Q/R$ where P is a nonzero natural number (expression) in decimal notation without redundant leading zeros and where P/Q is a common fracterm.

    The value of mixed fracterms is determined as follows: if P is a decimal natural number then $P\ Q/R = P + Q/R$, and if $P \equiv -P'$ with P' a decimal natural number then $P\ Q/R = P - Q/R$.

    A mixed common fracterm is not a fracterm. A mixed fracterm is an instance of a molecular term, that is a combination of components, each of which are required to be certain terms, that is to have a specific form.

    Another instance of molecular forms are decimal numbers which take the form P, QR with P a decimal whole, Q a list of zero or more zeroes, and R a decimal natural number. For instance $-17, 00350$ with $P \equiv -17$, $Q \equiv 00$, and $R \equiv 350$.

4.3 Fracterms over a larger signature

One may enrich the signature, and contemplate fracterms in the larger signature. I will consider the case that the signature is enriched with a successor function $S(\_){:}V \to V$. Adaptations to the enriched signature are a matter of design. Here is a proposal on how that can be done:

1. $S(0)/(1 + S(34))$ is a fracterm with numerator $S(0)$ and denominator $1 + S(34)$.

2. Wholeterms may also involve $S(\_)$. For instance $S(17 + (3 \cdot S(5))$ is a wholeterm. This is a matter of choice motivated by the idea that the successor is supposed to turn integers into integers.

3. $S(u7)/2$ is a half-open fracterm.

4. Common fracterms do not contain occurrences of S (The idea is that the numerator and the denominator of a common fracterm have been calculated, and for that reason cannot be calculated any further.)

5. Unless stated otherwise the notion of a desirable outcome is such that there are no occurrences of $S(\_)$ in desirable outcomes.

6. A flat fracterm my contain occurrences of S is the nominator as well as in the denominator. $S(1/2)/3$ is a composite fracterm and is not a not flat fracterm.

4.4 Arithmetical quantities, a container class for fracterms

It is plausible to ask what kind of thing is a fracterm. Are fracterms instances of some other more inclusive type? The obvious answer is that fracterms are terms, or expressions if one so prefers. This reply, however, has an implicit syntactic bias which I prefer to avoid. I prefer to use *arithmetical quantity* (AQ) as the name of a container class for fracterms. The idea is that $2 + 7$, $31 \cdot 56 - 2 \cdot 3$ etc. are AQs, and if P and Q are AQs then so is P/Q. AQs of the form P/Q are fracterms.

Unlike syntactic notions in theoretical computer science, however, AQ is still an informal notion. For instance one may hold that $7+3+5$ is an AQ consisting of three components and at the same time be uncommitted on whether or not $7+(3+5)$ and $7+3+(5)$ are AQs. In other words AQs don't necessarily come with a fully worked out sytnax.

5 Fracterms applied to the fraction definition problem

One may think that a fraction is just a fracterm, and that therefore the word "fracterm" is superfluous. However, taking fractions for fracterms constitutes just one of many possible different definitions of a fraction. As a consequence of the existence of a plurality of definitions of fractions it is counterproductive to equate fracterms with fractions, thereby obscuring the existence of different definitions of fractions. In order to see the existence of a plurality of notions of fraction it is useful to notice that in many mathematical works fraction is rather used as a synonym for quotient. That diverges sharply from taking a fraction for a fracterm.

For expositions on fractions from an educational viewpoint I mention [27], and [33]. In [2] many other references to educational research papers on fractions can be found. The presence of different views on the ontology of fractions clearly emerges from these documents.

I will distinguish two kinds of fraction definitions: implicit definitions, and explicit definitions. Explicit definitions are given in the presence of the notion of a fracterm, and may make use of fracterms. Below I will only consider explicit definitions of fractions which identify fractions with a type (i.e. subset) of fracterms. I will merely mention the existence of other explicit definitions of fractions. Implicit definitions are semantic indications about the meaning of fraction which emerge from patterns of use of the word "fraction" in various texts.

5.1 The fraction definition problem

I prefer to think in terms of the existence of a "fraction definition problem", that is the problem of how to define a fraction. Each educational work on fractions must to some extent confront this problem, and each author of such works is likely to have an opinion about it, whether or not such opinions are mentioned in the resulting works. Solutions may range from (i) claiming nonexistence of the problem, via (ii) claiming that defining fractions is like defining proofs, a matter for advanced (higher) mathematics (or logic) which need not be reflected upon at an elementary level, and (iii) various suggestions for fraction definitions that hardly qualify as such, to (v) detailed proposals for fraction definitions.

Below I will, as an application of fracterms discuss several possible definitions of fractions, that is several potential solutions of the fraction definition problem.

5.2 The relevance of "fracterm" as a novel concept

One may dispute that fracterms are novel, and one may dispute as well that the introduction of fracterms is useful. Fracterms are novel only in as much as these are meant to serve as first class citizens in actual expositions of elementary mathematics.

Then one may think that the introduction of fracterms merely constitutes a marginal addition to the well-known framework of elementary arithmetic. But that this is not the case, because syntactic notions are remote from conventional mathematical presentation. At present it is not even obvious that it is feasible to incorporate in a natural and productive manner, a syntactic notion like fracterm within a framework for ordinary elementary mathematics.

The possibility to provide a well-defined terminology for fracterms serves as a justification for very introduction of the notion. Providing similar definitions for fractions without having a definition of fractions at hand is not possible. The simplest (though unconventional) definition of a fraction is that a fraction is a fracterm.

One may think that it is preferable to commit to a precise notion of a fraction before introducing additional notions such as fracterm and fracsign. However, the idea is that not so much the selection of a specific definition of fractions is relevant but rather giving an explanation of the existence of a plurality of notions of fraction. Moreover it is conceivable that different notions of fraction each have advantages in different circumstances so that making a choice for a definition of fractions for once and for all is impractical, and may even be counterproductive.

Arithmetical datatypes come with a theory of fracterms. The difference between the various arithmetical datatypes when judged from the perspective of the respective fracterm theories is significant. For instance: in $\widehat{\mathbb{Q}}^d_{\perp}$ each expression (open or closed) is equivalent to a flat fraction (see [10]). In $\widehat{\mathbb{Q}}^d_0$ each expression equals a sum of flat fractions, and no bound on the number of summands may be imposed (see [9]). For $\widehat{\mathbb{Q}}^d_{\infty,\perp}$ no work on fracterm theory is available, whereas as for $\widehat{\mathbb{Q}}^d_{\pm\infty,\Phi}$ the corresponding fracterm theory has been invesigated in [5].

5.3 Implicit definitions of fractions

An implicit definition of fractions provides a collection of patterns of use for the word fraction from which a definition, be it an informal definition, may be extracted, where the extraction of a conceptual definition may even be performed in different ways.

In principle there is no objection against working with an implicitly defined notion of fraction and at the same time making use of the word fracterm, though in practice that is not te be expected.

Three examples of collections of assertions giving rise to an implicit definition of fractions aregiven in the following Paragraphs:

5.3.1 A fraction is a kind of number

Besides examples of closed identities involving fracterms "fractions as a kind of number" may come with these assertions:

(i) a fraction is a kind of number,

(ii) some but not all fractions are equal to a whole number,

(iii) fractions have a numerator and a denominator which are separated by a horizontal bar,

(iv) addition of fractions is easy, just follow the rules,

(v) fraction equivalence is often considered a difficult notion, but that is unnecessary,

(vi) mixed fractions are not really fractions, but composite fractions are.

5.3.2 A fraction is a numeral

Besides examples of closed identities involving fracterms "fractions as numerals" (of which there are several different instances) may come with these assertions:

(i) a fraction is a numeral, a numeral is a value,

(ii) a fraction is a pair of numerals,

(iv) fractions can only be added if the denominators of both are the same,

(v) in order to understand fraction equality one must first understand fraction equivalence,

(vi) the fraction $(1+2)/7$ is not simplified, but the fraction $x/7$ is simplified.

5.3.3 A fraction is an equivalence class

Besides examples of closed identities involving fracterms "fractions as equivalence classes" (of which there are several different instances) may come with these assertions:

(i) a fraction has three parts, a numerator, a denominator, and a fraction bar,

(ii) later (but not now!) it wil be explained that a fraction is a set, technically called an equivalence class, of notations for numbers,

(iii) one may think of a fraction as a number,

(iv) all fractions can be simplified, simplification is a method for computing a fraction,

(v) the fraction x/y with different variables x and y cannot be written in a simpler form.

5.4 Explicit definitions of fractions: fracterm type based fraction definitions

One may contemplate the fracterm terminology as a scheme for generating a fraction terminology for an arbitrary definition of fraction by simply replacing fracterm by fraction. However, if a fraction is considered a quotient, i.e. number, then a substantial part (fraction in another sense) of the terminology for fracterms fails to apply because numbers have no numerator and no denominator. In fact the fracterm terminology is specific for the notion of fracterm and also for the arithmetical datatype at hand (that is $\widetilde{\mathbb{Q}}^d_{\perp,\mathsf{C_D}}$) and deriving from that terminology a terminology for a specific notion of fraction requires care as well as the awareness that not all items of the terminology can be inherited. Rather than surveying many conceptions of fractions two definitions of fractions may be contrasted. A survey of notions of fraction can be found in [2].

5.4.1 A fraction is a fracterm

When fraction is understood (defined) as fracterm, the fracterm terminology becomes the corresponding fraction terminology, simply by replacing fracterm by fraction in each item in the description of the terminology. In addition the keyword fracterm is made redundant as fraction can be used instead.

Working along these lines an adequate perspective of fractions results. A disadvantage of taking fractions for fracterms is that this very idea is unfamiliar to most teaching staff and is incompatible with many explanations of fractions in educational methods and books.

5.4.2 A fraction is a quotient

If one takes a fraction to be a quotient, say in the arithmetical datatype $\widehat{\mathbb{Q}}^d_{\perp,\mathsf{C_D}}$ or in $\widetilde{\mathbb{Q}}^d_{\perp,\mathsf{C_D}}$ then there is no notion of numerator or denominator for a fraction and much of the fracterm terminology does not inherit to the specific case (of fractions as quotients).

The idea of taking fracterms seriously as first class citizens in this case suggest to maintain a notion of fracterm besides fraction (that is quotient), inclusive the given fracterm terminology and to simply use fraction as a synonym of quotient. If fractions are considered quotients then fraction terminology does away with the following notions (which come about when simply replacing fracterm by fraction in the terminology of Section 4): the quotient of a fraction, numerator, denominator, problem fraction, regular fraction, common fraction (simple fraction, vulgar fraction), negated common fraction, equivalence of fractions, mixed fraction, composite fraction, simplified fraction, flat fraction, like fractions, QCR as a rule for the addition of fractions, open versus closed fraction.

Only the following notions "survive", that is can be inherited from fracterm terminology to fraction terminology upon adopting the fractions a quotients perspective: unit fraction, proper fraction, improper fraction, pseudo fraction (i.e. an integer).

5.4.3 A fraction is a pair of an arbitrary integer and a positive integer

Conceiving a fraction as a pair of an arbitrary integer and a positive integer corresponds to viewing a fraction as a common fracterm. Upon adopting that a fraction is a pair of whole numbers, the second one being positive, many more aspects of the fracterm terminology can be turned in to terminology for fractions. In particular common (simple, vulgar) fractions exist, as well as the notions of numerator, denominator, simplification, simplified fraction. The items of fracterm terminology which have no counterpart if fractions are considered pairs as mentioned are these: mixed fracterm, composite fracterm, and flat fracterm.

6 Fraction as an ambiguous noun

So, after these preparations one may ask once more: what is a fraction? If so many different meanings can be given to the noun fraction, should the word be given up and be replaced for something else? No, another path can be taken.

Claim 6.1. *Fraction is an ambiguous noun, even when restricted to the context of arithmetic. The meaning of the noun fraction includes as options: fracterm (i.e. open fracterm), closed fracterm, integer fracpair, rational fracpair, and rational number (fracvalue). This listing of options is not exhaustive, other options may included (such as "fracterm or mixed fracterm") and new options may be be invented.*

A major advantage of adopting Claim 6.1 is that unclear or self-contradictory fragments of texts about fractions may be qualified as being sloppy with the ambiguity of fraction. Most assertions about fractions can be easily made unambiguous, precise, and valid, by writing in terms of the listed options for "fraction". By using open fracterm, closed fracterm, integer fracpair, rational fracpair, and rational number (fracvalue) in places where fraction might have been written a text may be disambiguated.

However, it is not always the case that working with disambiguated terminology clarifies a text fragment: for instance the claim that "fractions can only be added if the denominators are equal" appears every now and then in sketchy explanations of fractions and I did not find any plausible analysis of the meaning of this assertion in a disambiguated terminology which renders it true.

6.1 What kind of ambiguity covers fraction? Multi-bias ambiguity

Sennet [36] provides a survey of occurrences of ambiguity. It is not obvious that (arithmetic) fraction as having a range of different meanings is comprised by the options for ambiguity as listed in [36]. I found no description of the form of ambiguity which comes with the noun fraction and for that reason an ad hoc name for such ambiguities will be suggested. I propose to consider the noun fraction to be multi-bias ambigious. Multi-bias ambiguity requires a definition.

A noun α is multi-bias ambiguous of the following conditions are met.

1. α can be disambiguated into $\alpha_1, \ldots, \alpha_n$ for some less ambiguous nouns $\alpha_1 \ldots, \alpha_n$. It may be the case that in certain cases yet different refinements of α are needed, but then the idea is that disambiguation with the listed α_i covers the majority of cases.

2. Most users of the noun α have a partial ordering of preferences on the options α_i for disambiguation. Many users may disambiguate α into a subset of the listed options only. These preferences constitute a bias towards a certain interpretation of α.

3. It is not uncommon for a user (say A) to be hostile to the very idea that α is ambiguous and that another user, say B who maintains a different bias about α consider A's preferences for the meaning of α as biased.

4. Many users of the noun α are unaware of its ambiguity, and may use a strong preference for one of the disambiguations, calling for another one only in exceptional cases. (For instance many users of the noun fraction may always consider a fraction to be a rational number, unless they need to consider it an integer fracpair, whereas many other users of the same noun may always consider it to be an integer fracpair only to consider it a rational number when forced to do so in a certain context. Yet other users may always think of an open fracterm.)

5. There is no generally adopted perspective on why it is practical or helpful to group the disparate named notions $\alpha_1 \ldots, \alpha_n$ together as the different interpretations for a single noun (in this case α).

7 Concluding remarks

Writing about fractions in a paradigm sensitive manner is less straightforward than dealing with the different definitions of real numbers via say Dedekind cuts, Cauchy sequences, and infinite decimal expansions. Unlike with the case of real numbers, the different viewpoints on fractions cannot be so easily be grasped in terms of a spectrum of different and well-known definitions. Currently there is no catalogue available of competing definitions of fractions. Definitions of fractions must be retrieved from thousands of texts about fractions in an indirect manner.

Having a definition of rational numbers at hand does not trivialise the writing on fractions. For instance the question whether or not "$(1+2)/5$" is a common fraction is independent of an underlying definition of rational numbers. In this connection one may consider the 3-th principle mentioned by Frege in [21] (p xxii)

> ... never to ask for the meaning of a word in isolation, but only in the context of a proposition; ...

As a proposition one may consider: "the fractions 2/3 and 4/6 satisfy $2/3 = 4/6$", and one may assume that it is known what a rational number is, and in that context the question arises: what is a fraction?

Frege discusses the definition of natural numbers and raises fundamental questions about these. He writes (in 1848, see e.g. [21], introduction p. xv) "The first prerequisite for learning anything is thus utterly lacking – I mean the knowledge that we do not know." Moreover he mentions the diversity and disparity of (then) existing descriptions of natural numbers. I hold that both motives apply nowadays to fractions and justify further inquiries into these.

The position that fractions are expressions can be recognized in [30]. In [38] a fractions as fracpairs position is adopted, just as in [28]. In [35] the position that fractions are numbers has been adopted. In [22] it is stated that fraction is not a mathematical concept, thereby explaining why it has no well-known and rigorous definition. A preliminary discussion of logics of fractions is reported in [6] where paraconsistent logic comes into play.

Issues beyond mere equations between closed arithmetical quantities relate to logics of fractions beyond equational. For instance consider the following question:

Problem 7.1. *What is the cardinality of the set* $\{1/2, 2/4\}$*?*

It seems plausible to claim that for any view of fractions the answer to this question is 1. Indeed, (i) the symbols 1/2 and 2/4, as occurring in the question, are fracsigns, (ii) fracsigns allow polymorphic typing: either as a quotient or as a fracterm or as something in between of these (which may in turn serve as a reference to a quotient), (iii) the set theory notation context coerces the typing of a fracsign (or at least of these particular fracsigns) into a quotient. If one insists that a fracsign is read otherwise this must be somehow made clear in the notation, e.g. by writing $\{\ldots\}_{ft}$ for a set of fracterms so that $\#(\{1/2, 2/4\}_{ft}) = 2$ (with $\#(v)$ denoting the cardinality of V)

A working hypothesis for further work on fractions I wish to put forward the idea that the answer to the question "why is learning fraction arithmetic so difficult", as stated in [28] lies in part in the conceptual difficulties innate in the concept of fraction. These difficulties are not even mentioned in [28].

Upon asking a mathematician what a fraction is, one may be pointed out that a fraction is an equivalence class of entities (for some plausible definition of equivalence), for instance pairs, each of which arise as the interpretation of a fracsign as a structured entity. Then one may ask in response: "how to define numerator and denominator", and be pointed out that "in fact" fractions don't come with such components, that it is fraction expressions rather than fractions which can be decomposed into a numerator, a denominator, and an operator symbol. I hold that the latter viewpoint is quite distant from the conventions in educational practice where it is more often than not assumed that fractions allow a decomposition into the three familiar components. This gap must not be taken for granted. Leaving this gap unbridged is not helpful for the development of methods for teaching arithmetic.

I propose that fracterm rather than (rational) number is considered the principal notion of (rational) arithmetic, that is the notion which can be most unambiguously defined. Rational numbers come about only when a semantic model for fracterms is sought, and just as in computer programming, semantic models for a given syntax are by no means unique. The suggestion that numbers constitute a rigid and known world of entities which serve as the subject of arithmetic is an illusion. On the contrary, fracterms are syntactic entities just as programs are, and semantic models and objects are sought (and found) in order to improve one's understanding of the syntactic entities at hand. Semantic models may be tuned in different ways towards specific applications in which fracterms and other arithmetical quantities play a role.

Logic, and especially formal logic are at a distance from elementary mathematics. In [18] it is argued that one of the advantages of working with formalized logic is the side-effect of de-semantification. Using axiom based logic reasoning is made less dependent on semantic intuitions. My objective, however, is to use formal aspects, and in particular the formal methods as embodied in the theory of (abstract) datatypes, as an additional tool for elementary mathematics in such a manner that making semantic intuitions more remote is avoided.

Acknowledgement. The reviewer has made many useful suggestions including the use of the adjective singular for fracterms with a denominator equal to zero.

References

[1] J.A. Anderson, N. Völker, and A. A. Adams. Perspecx Machine VIII, axioms of transreal arithmetic. Vision Geometry XV, eds. J. Latecki, D. M. Mount and A. Y. Wu, 649902; https://doi.org/10.1117/12.698153, (2007).

[2] J.A. Bergstra. Meadow based fracterm theory. arXiv:1508.01334v3, https://arxiv.org/pdf/1508.01334.pdf, (2019).

[3] J.A. Bergstra. Division by zero, a survey of options. *Transmathematica*, ISSN 2632-9212, https://doi.org/10.36285/tm.v0i0.17, published 2019-06-25 2019, (2019).

[4] J.A. Bergstra. Dual number meadows. *Transmathematica.* ISSN 2632-9212, https://doi.org/10.36285/tm.v0i0.11, published 06-25-2019, (2019).

[5] J.A. Bergstra. Fractions in Transrational Arithmetic. *Transmathematica.* ISSN 2632-9212, https://doi.org/10.36285/tm.19, published 02-27-2020, (2020).

[6] J.A. Bergstra, I. Bethke. Note on paraconsistency and reasoning about fractions. *J. of Applied Non-Classical Logics*, 25 (2) 120–124, (2015).

[7] J. A. Bergstra, J. Heering and P. Klint. Module Algebra. *Journal of the ACM*, 37(2): 335–372, (1990).

[8] J.A. Bergstra and C.A. Middelburg. Inversive meadows and divisive meadows. *Journal of Applied Logic*, 9(3): 203–220 (2011).

[9] J.A. Bergstra and C.A. Middelburg. Transformation of fractions into simple fractions in divisive meadows. *Journal of Applied Logic*, 16: 92–110 (2015). Also https://arxiv.org/abs/1510.06233.

[10] 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),* LNCS 8950, pages 46-61, Springer, 2015. Also available at https://arxiv.org/pdf/1406.6878.pdf[math.RA], (2015).

[11] J.A. Bergstra and A. Ponse. Fracpairs and fractions over a reduced commutative ring. *Indigationes Mathematicae* 27, 727-748, (2016). http://dx.doi.org/10.1016/j.indag.2016.01.007. Also https://arxiv.org/abs/1411.4410.

[12] J.A. Bergstra and A. Ponse. Arithmetical datatypes with true fractions. *Acta Informatica* (2020). https://doi.org/10.1007/s00236-019-00352-8.

[13] J.A. Bergstra and J.V. Tucker. Initial and final algebra semantics for data type specification: two characterization theorems. *SIAM J. Comput.*, Vol. 12 (2), 366-387 (1983).

[14] J.A. Bergstra and J.V. Tucker. Equational specifications, complete term rewriting systems, and computable and semicomputable algebras. *Journal of the ACM*, Vol. 42 (6), 1194-1230 (1995).

[15] J.A. Bergstra and J.V. Tucker. The rational numbers as an abstract data type. *Journal of the ACM*, 54 (2), Article 7 (2007).

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

[17] J. Carlström. Partiality and choice, foundational contributions. *PhD. Thesis, Stockholm University*, http://www.diva-portal.org/smash/get/diva2:194366/FULLTEXT01.pdf, (2005).

[18] Catarina Dutilh Novaes. Towards a practice-based philosophy of logic: formal languages as a case study. *Philosphia Scientiae*, 16 (1) pp. 71–102, (2012).

[19] T.S. dos Reis, W. Gomide, and J.A.D.W. Anderson. Construction of the transreal numbers and algebraic transfields. *IAENG International Journal of Applied Mathematics*, 46 (1), 11–23, (2016). http://www.iaeng.org/IJAM/issues_v46/issue_1/IJAM_46_1_03.pdf

[20] H. D. Ehrich, M. Wolf, and J. Loeckx. Specification of Abstract Data Types. *Vieweg + Teubner*, ISBN-10:3519021153, (1997)

[21] Gottlob Frege. *The foundations of arithmetic, a logico-mathematical enquiry into the concept of number.* Translated by J.L. Austin. Harper & Brothers, New York, (1950, second revised edition 1953, reprinted 1960). http://www.naturalthinker.net/trl/texts/Frege,Gottlob/Frege,%20Gottlob%20-%20The%20Foundations%20of%20Arithmetic%20(1953)%202Ed_%207.0-2.5%20LotB.pdf

[22] Martha I.F. Fandino Pinilla. Fractions: conceptual and didactic aspects. *Acta Didactica Universitatis Comenianae*, 7: 82–115 (2007).

[23] Heinz Griesel. Der quasikardinale Aspekt in der Bruchrechnung, (in German). *Der Math.-Unt.*, 27 (4), 87–95, (1981).

[24] Joseph A. Goguen. Memories of ADJ. *Bulletin of the EATCS* no. 36, October 1989. Available at https://cseweb.ucsd.edu/~goguen/pps/beatcs-adj.ps, (1989).

[25] Arthur C. Howard. Addition of fractions–the unrecognized problem. *The Mathematics Teacher*, 84 (9), 710–713, (1991).

[26] Y. Komori. Free algebras over all fields and pseudo-fields. Report 10, pp. 9-15, Faculty of Science, Shizuoka University (1975).

[27] Susan J. Lamon *Teaching fractions and ratios for understanding–Essential content knowledge and instructional strategies for teachers.* Routledge, 1999, (3rd edition 2012).

[28] H. Lortie-Forgues, J. Tian, and R. S. Siegler. Why Is Learning Fraction and Decimal Arithmetic So Difficult? *Developmental Review*, 38, 201-221, https://files.eric.ed.gov/fulltext/ED565462.pdf (2015).

[29] H. Michiwaki, S. Saitoh, and N. Yamada. Reality of the division by zero $z/0 = 0$. *International Journal of Applied Physics and mathematics*, http://www.ijapm.org/vol6/345-P00120.pdf (2016).

[30] Francis J. Mueller. On the fraction as a numeral. *The Arithmetic Teacher*, 8 (5), 234–238, (1961).

[31] J-. F. Nicaud, D. Bouhineau, and J-. M. Gelis. Syntax and semantics in algebra. *Proc. 12th ICMI Study Conference, The University of Melbourne, 2001.* HAL archives-ouvertes https://hal.archives-ouvertes.fr/hal-00962023/document, (2001).

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

[33] Friedhelm Padberg. Didaktik der Bruchrechnung, (In German). fourth ed., in: Series: Mathematik, Primar und Secundarstufe, Springer-Spectrum, (2012).

[34] Helena M. Pycior. Early criticism of the symbolical approach to algebra. *Historia Mathematica*, 9: 392–412 (1992).

[35] S. Rollnik. Das pragmatische Konzept für den Bruchrechenunterricht. (In German.) PhD thesis, University of Flensburg, Germany, (2009).

[36] Adam Sennet. Ambiguity. *In The Stanford Encyclopedia of Philosophy, editor Edward N. Zalta*, https://plato.stanford.edu/archives/spr2016/entries/ambiguity/, edition Spring 2016,(2016),

[37] A. Setzer. Wheels (draft). http://www.cs.swan.ac.uk/~csetzer/articles/wheel.pdf, (1997).

[38] Henry van Engen. Rate pairs, fractions, and rational numbers. *The Arithmetic Teacher*, 7 (8), 389–399, (1960).