Robbert van Dalen
robbert.van.dalen@gmail.com

Abstract

Division by zero is undefined, so we have decided to eradicate zeros completely. We do this by introducing a new kind of Integer, called an Account, which is an ordered pair of two Naturals - Debit and Credit.
Naturally, we also introduce a new kind of Rational, called the Super-Rational, which is an ordered pair of two Accounts - Numerator and Denominator.

Definitions

An **Account**(`Acc`) has type `Nat*Nat` and a **Super-Rational**(`Sra`) has type `Acc*Acc`. We define operations on `Acc` and `Sra` as **accounting** operations to differentiate them from ordinary arithmetical operations.

Accounting Naturals

Everything starts with **one** tally symbol `1` and multiples of tallies. Note that the actual choice of the tally symbol is insignificant: we could even choose to use the zero symbol, but we will not.

Addition is defined to be the **concatenation** of two tally sequences. **Multiplication** is the **repeated** concatenation of one tally sequence, times another tally sequence.

```
one = 1

+   :  a+b  => ab
*   :  a*b  => aaa.. (repeat b times)
```

Accounting Accounts

We extend addition(++) and multiplication(**) together with **additive inverse**(-) on Accounts. All accounting on Accounts are based on (operations on) Naturals.
We syntactically pair the Debit and Credit of an Account with the infix backslash symbol, i.e. `Debit\Credit`.
We follow Coltharp's[1] semantics and Wildberger's syntax[2] resulting in the following definitions:

```
- :  -a\b          => b\a
++:   a\b ++ c\d   => (a+c)\(b+d)
**:   a\b ** c\d   => ((a*c)+(b*d)\((a*d)+(b*c))
```

Interpreting Accounts

An Account is **balanced** when Debit and Credit are equal. Such a balanced Account can be interpreted as (being in the equivalence class of) a zero but we won't.

When Debit is greater than Credit, we can subtract Debit by Credit, up to leaving `1` Credit, and vice versa when Credit is greater than Debit. When Debit equals Credit we could replace both of them with `1`.

We call such dubious substitutions **simplifications** of Accounts. Note that the simplified version of an Account is in the same equivalence class as its unsimplified version.

Accounting Super-Rationals

We extend addition(+++), multiplication(***), additive inverse(_) with **multiplicative inverse**(`) on Super-Rationals:

```
e  =   a\b
f  =   c\d

`  :   e/f          =>  f/e
_  :  _e/f          => -e/f
+++:   e/f +++ g/h  =>  ((e**h)++(g**f))/(f**h)
***:   e/f *** g/h  =>  (e**g)/(f**h)
```

Interpreting Super-Rationals

So accounting Super-Rationals is similar to accounting Rationals, with the exception that the multiplicative inverse of **any** Super-Rational is allowed. So accounting Super-Rationals is **total**: there are no **exceptions**.

We give you the following glimmer of hope:

```
`1\1/2\1   ==   2\1/1\1
```

Note that we also can decide to simplify a Super-Rational as we would a Rational by calculating the Greatest Common Divisor (GCD) between Numerator and Denominator (and then divide them by their GCD). There is a catch, but we leave that for further research.

Discussion

So what is the real problem with zeros?

Zeros **destroy** information

That is why they don't have a multiplicative inverse: because it is impossible to rebuilt something you have just destroyed.

Hopefully this short paper will make the reader consider the author's firm **believe** that:

One should **never** destroy anything, if one can help it

We practically abolished zeros. Should we also abolish simplifications? Not if we want to stay practical.

References

[1] F L Coltharp, Introducing the Integers as Ordered Pairs. *School Science and Mathematics, Volume 66, Issue 3, pages 277–282, March 1966.*

[2] N J Wilberger, MathFoundations12: Introducing the integers. *http://www.youtube.com/user/njwildberger*

