Pure Data Foundations of Mathematics

Saul Youssef
Department of Physics
Boston University
youssef@bu.edu

April 19, 2023

Abstract

This is an abstract.

1 Introduction

2 Foundation

Any system of reasoning must necessarily have at least one foundational concept which is understood before a first definition. In our case, the foundational concept is the **finite sequence**. We assume that finite sequences and obvious operations on finite sequences are understood. On the other hand, logic, logical values and predicates (foundational in ZFC) and types (foundational in type theories) are not assumed to be understood. These will be defined in terms of finite sequences as we proceed. Taking finite sequences as understood, we can define "data" and "coda" by the following.

A **data** is a finite sequence of **codas**.

A **coda** is a pair of **data**.

Notationally speaking, if A and B are data, $A\ B$ denotes the concatenation of A and B as finite sequences and $A : B$ denotes the data consisting of the single coda formed by pairing A and B. For examples, the empty sequence of codas () is data, the pairing of two empty sequences (:) is data and this sequence of three codas (:)(:)((:):(:)(:)) is data consisting of a sequence of three codas. This it "pure data, made of nothing." It is the central concept of coda. We shall see that both data in the ordinary sense (bits, bytes, language expressions) and mathematical concepts such as variables, functions, definitions, logical values, categories, morphisms, types and theorems appear as varieties of pure data with simple definitions in terms of the small algebra of data $A\ B$ and $A : B$. In this algebra, we conventionally hold that colons bind from the right first and less strongly than concatenation, so $A : B : C$ means $(A : (B : C))$ and $A : B\ C$ means $(A : (B\ C))$.

The mathematical content of coda is embodied in a partial function from coda to data called a **context**. Given a context δ data equality is defined by the relations

$$A\ B = \delta(A)\ B = A\ \delta(B)$$

$$A : B = \delta(A) : B = A : \delta(B)$$

for data A and all data B. Here the partial function δ is extended to a function from data to data with identities. Conceptually, relations are assumed until "=" has the same properties as the identity.

The important concept of "atoms" and "atomic data" provides both stability, and a mechanism for adding new definitions to a valid context. Given a context δ, an *atom* is a coda c where $(c, c) \in \delta$. Data with atoms in it's sequence are called *atomic* data. In a context δ with corresponding equality, a *definition on atom* a is a partial function from coda to data where the domain of the partial function is contained within the codas $(a\ A : B)$ for the atom a and for any data A and B. The axiom below defines how definitions may be added to a *valid* context without clashes.

AXIOM OF DEFINITION. *The context* $(:) \to (:)$ *is valid. If* a *is an atom in context* δ*, if* δ_a *is a definition on* a*, and if* δ *and* δ_a *have disjoint domains, then the union of* δ *and* δ_a *is a valid context.*

The point of the initial context $(:) \to (:)$ is to establish $(:)$ as an initial atom. In a typical case, a definition δ_a will be added to a context when a has not been used by some other definition, thus avoiding a clash. The disjoint requirement also insures the stability of atoms. Once a coda is an atom in some context, it remains an atom independent of future added definitions. We say that atomic data is "always atomic" and, similarly empty data is "always empty". This stability is the basis for an internal logic and it also makes it convenient to define a stable 0-bit, 1-bit, stable bit sequences, stable byte sequences in addition to definitions as shown in figure 1.

As a formal axiomatic system, coda has only one axiom: The Axiom of Definition. We shall see in the following sections that coda has an internal logic making logic-related axioms unnecessary. Also, coda has an internal language, making axioms related to language and syntax unnecessary. Unlike type systems with the Curry-Howard correspondence, proof and computation in coda are the same thing. Each is merely a data sequence $A_0 = A_1 = A_2 = \cdots = A_n$ either computing A_n from A_0 or establishing that $A_0 = A_n$ depending on one's point of view.

Unless otherwise indicated, "data" refers to pure data and "sequence" refers to a finite sequence. If data $A = ()$, we say that A is *empty*.

3 Logic

Within coda, items with mathematical meaning are merely pure data of different kinds. This means, roughly speaking, that mathematical questions will appear in the form

> Is data A equal to data B?

Since equality itself will also be definition in context, the answer to the question $A = B$ is also data. But if everything about $A = B$ is encoded in the concrete data $(= A : B)$, then this specific data should contain any "logical" interpretation of $A = B$. This suggests that "logic" in should be identified as a suitable coarsest non-trivial classification of data which is stable under added definitions. This motivates the following definitions.

- data is **true** if it is empty.
- data is **false** if it is atomic.
- data is **undecided** otherwise.

Examples illustrate the flavor of this classification.

True: (), (pass:), (null:a b c), (null: (foo:bar)).

False: (:), a b c, pass : a b c, a b c (foo:bar).

Undefined: (foo:bar), (pass:foo:bar), (foo:1 2 3).

This logic can be thought of as "not quite two valued" where undefined data like (foo:bar) represents data which is currently undefined, but which may receive a value if more definitions are added to the current context. The stability of true or false data under new definitions makes these useful concepts. We use "always" to refer to future definitions as in

- True data is "always true."
- False data is "always false."

Undecided data, on the other hand, may become true or false, depending upon future definitions. Also, as we will see, there is also undecided data which can never become true or false with any choice of future definitions. Such data is called **undecidable**. This will be discussed further in section X.

Although the logic of coda is not the same as classical logic, we will argue that this is the "correct" logic for reasoning in general. It is also quite close to classical logic in the sense that there is a definition that corresponds to each of the familiar 16 binary operations of classical propositional logic. For instance, when data A and B are either true or false, the data (XOR $A : B$) has the value () for *true* and (:) for *false* depending on the classical truth table of values for XOR. If either A or B are undecided, on the other hand, no definition applies to (XOR $A : B$) and it "waits for A and B to be resolved in future definitions."

4 Language

5 Proof and Computation

6 Spaces

7 Is Mathematics consistent?

8 Mathematical Machine Learning

9 Summary

References

[1] R.T.Cox, Am.J.Phys. 14, 1 (1946).

[2] S.Youssef, Mod.Phys.Lett A9, 2571 (1994).

[3] S.Youssef, Phys.Lett. A204, 181(1995).