PURE DATA SPACES....DRAFT...DRAFT...DRAFT

SAUL YOUSSEF
Department of Physics
Boston University

April 9, 2025

Abstract

All about spaces.

1 Overview

In Reference 1, we introduced a tiny axiomatic framework, based on the **finite sequence** as a foundational concept. With finite sequences understood, we had

1 *"Data" is a finite sequence of "codas," where each coda is a pair of data.*

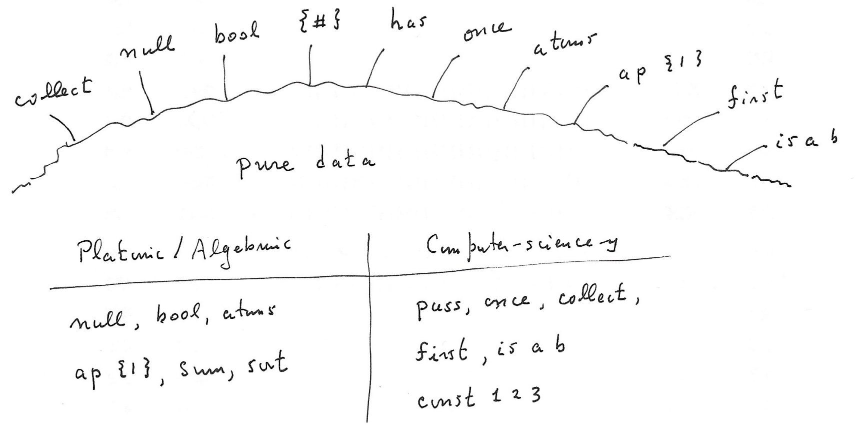
This kind of data, such as (:) (:(:)) ((:):(:) (:)), is "pure data" in the sense that it is "made of nothing." Such data has two natural operations: *concatenation* of data A with data B, written (A B), and *pairing* data A and data B as a *coda*, written (A:B).

2 *Definitions are embodied by a chosen partial function δ from codas to data called a context.*

As modest as it is, 1 and 2 appear to be able to capture Mathematics in general. Consequences include:

- Fixed points of δ (*atoms*), represent fixed data such as bits, bytes and text;
- Data outside the domain of δ are *variables*, which might become defined over time as *definitions* are added to δ;
- δ contains it's own *language*, allowing user specification of data and adding definitions to δ;
- Equality of data A=B is determined by $c \sim \delta(c)$ and by compatibility with finite sequences. Proof and computation are then defined by the equality. A sequence $A_1 = A_2 = \cdots = A_n$ is both a proof of A_n given A_1 and a computation $A_1 \mapsto A_n$. Proof and computation are essentially the same thing;
- Data is *true* if it is empty, *false* if it is atomic, and *undecided* otherwise. Undecided data may also be *undecidable*, as in the Godel phenomena or other seeming paradoxes. Data which is *never false* as definitions are added is a *theorem*.

In this work, we investigate the concept of a *space*, introduced in Reference 1.
...SEARCHING....ORGANICS....



Platonic / Algebraic	Computer-science-y
null, bool, atoms	pass, once, collect,
ap {1}, sum, sort	first, is a b
	const 1 2 3

Figure 1: *Growing spaces "organically" from pure data.*

2 Foundation

In [1], we argue that Mathematics and Computing as a whole can be captured within a tiny framework, based only on **finite sequence** as the foundational undefined concept. Assuming that finite sequences are understood, we define *data* and *coda* as follows.

Definition 1. *Data is a finite sequence of* **codas***, where each* **coda** *is a pair of* **data**.

Concatenation of data A and data B as finite sequences is denoted by "A B" and the pairing of data A and data B as a coda is written as "A:B" with a colon indicating the pairing from the definition. By the definition, the empty sequence "written ()" qualifies as data and, therefore, () paired with itself is a coda (written "(:)"). Any finite sequence of codas is data, so, for example (:) (:(:)) ((:):(:(:))) is data consisting of a sequence of three codas. We call this *pure data* because it is *data made of nothing.* By convention, the colon binds from the right first and binds less strongly than concatenation, so that A:B:C is defined to mean (A:(B:C)) and A:B C is defined to mean (A:(B C)). Data is typically written with upper case, and codas are typically written in lower case. To indicate the left and right data of a coda, we sometimes use L/R superscripts so that $c = (c^L : c^R)$.

All meaning within the system is determined by a single chosen partial function from coda to data called a **context**. Contexts are partially ordered by inclusion, so if δ and δ' are contexts, $\delta \leq \delta'$ if δ and δ' are equal on the domain of δ. Given a context δ, equality of data is determined by $c \sim \delta(c)$ for any coda c, and by compatibility with concatenation and colon: if A$\sim$B, then (A X)$\sim$(B X), (X A)$\sim$(X B), (A:X)$\sim$(B:X) and (X:A)$\sim$(X:B) for any data X. Thus, if A$\overset{\delta}{=}$B and $\delta \leq \delta'$, then A$\overset{\delta'}{=}$B. We may say that if A and B are equal then they are *always equal*, thinking of $\delta \to \delta'$ as the passage of time. Note that any fixed point c of δ is stable in the sense that any "later" δ' must be identical to δ on c. The fixed points of a given context δ are called *atoms.*

Definition 2. *Within a given context δ, coda c is an* **atom** *if $\delta : c \mapsto c$. If data A contains an atom, A is* **atomic** *data. Data A is* **invariant** *if every coda a in the sequence of A is an atom and if a^L and a^R are both* **invariant**.

Note that empty data is invariant. If a and b are atoms in context δ and if A and B are data, we have: 1) a and b remain atoms in any context greater than or equal to δ; 2) $a = b$ if and only if

$a^L = b^L$ and $a^R = b^R$; 3) If (a A)=(b B), then $a = b$ and A=B; 4) If (A a)=(B b), then A=B and $a = b$; 5) If A and B are invariant and equal, then A and B are identical as pure data.

Given a context δ, we may write the corresponding equality as $\overset{\delta}{=}$ or, when not ambiguous, as =. If data A and B satisfy A:X$\overset{\delta}{=}$B:X for all X, this equivalence is also often written simply as "=" if is clear in context, as in the following definitions.

Definition 3. *Data A is*

- *a* **constant** *if (A:X) = (A:) for all X;*
- **idempotent** *if (A:A:X) = (A:X) for all X;*
- *an* **involution** *if (A:A:X) = X for all X;*
- **commutative** *if (A:X Y) = (A:Y X) for all X, Y;*
- **distributive** *if (A:X Y) = (A:X) (A:Y) for all X, Y;*
- **associative** *if (A:X Y) = (A:(A:X) Y) = (A:X (A:Y)) for all X, Y.*

If one thinks of A:X Y as an A-specified product X$\overset{A}{*}$Y, then associativity of A guarantees (X$\overset{A}{*}$Y)$\overset{A}{*}Z$ = X$\overset{A}{*}$(Y$\overset{A}{*}Z$) for all X,Y,Z. It is also convenient to define a product and sum of data in general, corresponding to the two foundational finite sequence operations: concatenation and colon. For data A and data B, let (A·B):X = A:B:X and let (A$\oplus$B):X = (A:X) (B:X). As binary operators on data, $\cdot$ and $\oplus$ are both associative, but neither is generally commutative.

3 Genesis

The mathematical objects that we are used to will all be represented as pure data, and all meaning about mathematical objects, including what constitutes a valid proof or a valid computation, is determined by an assumed context embodying a chosen collection of definitions. For instance, a valid proof in δ is just a sequence A$_1\overset{\delta}{=}$ A$_2\overset{\delta}{=}\ldots\overset{\delta}{=}A_n$, which can be viewed either as proving A$_1\overset{\delta}{=}$A$_n$ or as computing A$_1\mapsto$A$_n$. This is discussed in [1] and will gradually become clearer as we go. The immediate issue is to understand what constitutes a "valid" definition and how to choose an initial context.

In the beginning, there are no definitions, and the corresponding context is uniquely the empty partial function from coda to data, δ_0. Within δ_0, data are equal only if they are identical as pure data, the only valid proofs are $X\overset{\delta}{=}X$ for any X and the only valid computations do nothing $X \mapsto X$ for any X. To define a non-empty context $\delta_0 \leq \delta$, we need a way to specify the domain of δ which is unchanged by any later later definition. Since the empty sequence is the only invariant, the way to specify if a coda (A:B) is in the domain of δ is to require either A or B or both to be the empty sequence. In each of these cases, the coda (:) is within the domain of δ, and so we must decide on what (:) maps to. There are three possibilities;

1. $\delta : (:) \mapsto ()$,
2. $\delta : (:) \mapsto (:)$, or
3. $\delta : (:) \mapsto$ *anything other than* () *or* (:).

Since pure data is *made of (:)*, choice 1 trivially causes all data to be equal to the empty sequence. On the other hand, choice 3 means that for any data A the number of (:) atoms in A grows without limit. Choice 3 is degenerate in the sense that no computation could produce a final answer. Thus, we are constrained to choice 2, meaning that (:) must be an atom in δ and, therefore, is an atom for any $\delta' \geq \delta$. It is convenient to generalize choice 2 and let δ :(:X)$\mapsto$(:X) for all data X, so that every coda (:X) is an atom.

A context of the form

$$\delta_a : (a\ A : B) \mapsto \delta_a(A, B) \tag{1}$$

for some invariant atom a is called a **definition**. We conventionally restrict ourselves to contexts $\delta \cup \delta_a \cup \delta_b \cup \ldots$ where a is an invariant atom in δ, b is an invariant atom in $\delta \cup \delta_a$, c is an invariant atom in $\delta \cup \delta_a \cup \delta_b$ and so forth. By requiring a, b,... to be disjoint, we maintain the required context partial ordering. If one thinks of our framework as an axiomatic system, the one axiom asserting that the empty context δ_0 is *valid*, and any context greater than or equal to a valid context is also *valid*.

4 Spaces

Mathematics requires a way to make abstract collections of other mathematical objects. Since data as defined is the only available material, the collection must "contain" data and the collection itself must be specified by some data, say S. Given data S, there are a couple of plausible ways for S to define a collection.

- The data in the collection is S:X for any data X;
- The data in the collection are the fixed points of S.

These options can be made to coincide if we require S to be idempotent, so that S:X is always a fixed point. For a collection S to be compatible with concatenation, we might expect that if S:X and S:Y are in the collection, then so is (S:X) (S:Y). This is guaranteed if

$$\mathrm{S : X\ Y = S : (S : X)\ (S : Y)} \tag{2}$$

which follows if S is both idempotent and associative. Thus we come to the concept of a *pure data space* or just *a space*.

Definition 4. *Data S is a* **space** *if S is idempotent and associative.*

Notice that idempotence guarantees compatibility with the colon and associativity guarantees compatibility with concatenation.

The data (S:) is called the **neutral data** of S. It is easy to check that idempotent distributive data is also associative and is thus also a space. Distributive spaces have the empty sequence as their neutral data [since (S:)=(S:() ())=(S:) (S:)=()]. A distributive space can be thought of as acting independently on each atom of "input" since if data X is a sequence $\mathrm{x}_1\ \mathrm{x}_2 \ldots \mathrm{x}_n$ of atoms, S:X = (S:x_1) (S:x_2)... (S:x_n).

4.1 Plan

4.2 Algebraic Data

4.3 Growing Spaces Organically

5 The theory of Spaces

5.1 Commuting spaces

If spaces S and T commute, then S·T is a space. To see this, note that S commuting with T means that S·T is idempotent, and for any data X and Y, (S·T):X Y = S:T:X Y = S:T:(T:X) (T:Y) = T:S:(T:X) (T:Y) = S:T:(S:T:X) (S:T:Y) = (S·T):((S·T):X) ((S·T):Y).

5.2 Compatible Spaces

We say that data A **absorbs** data B if A·B=A. If A absorbs B and B absorbs A, we say that A and B are **compatible**. *Absorbs* is transitive in general, reflexive for idempotent data such as spaces, but is not symmetric. Compatiblity, however, is an equivalence relation. For examples:

- Idempotent data is compatible with itself.
- **null** absorbs any data.
- Any data absorbs **pass**.
- The constant spaces {a} and {b} are compatible, but are not equal.

Any data that is compatible with a space is a space itself. To prove this, let S be a space and A be any data where S·A=S and A·S=A. Then A·A=A·S·A·S=A·S=A, so A is idempotent, and A:X Y = (A·S):X Y = A:S:(S:X) (S:Y) = A:S:(S:A:X) (S:A:Y) = A:S:(A:X) (A:Y) = A:(A:X) (A:Y), so A is a space.

5.3 Kernels, Positive Spaces, and Complementary Spaces

Data X is **in the kernel** of space S if (S:X)=(S:), and we denote this by X∈Ker(S). For any space S, if X∈Ker(S) and Y∈Ker(S), then X Y∈Ker(S). If the converse is true, we say that *S* is a **positive** space. Simple facts and examples follow.

- Every distributive space is positive, since if S:X Y=(S:)=(), so (S:X) (S:Y)=(), and, thus (S:X) and (S:Y) are both empty.
- However, not all positive spaces are distributive, for example, (**front** | |) is a positive space, but not a distributive space.
- If S is positive and a sequence of atoms $x_1\ x_2 \ldots x_n \in$ Ker(S), then $x_i \in$ Ker(S) for each atom.
- If S is positive, the kernel of S is the distributive space **apif** {(S:B)=(S:)}.

Spaces S and T are **complementary** if they are contained in each other's kernels, that is, if S:T:X=S: and T:S:X=T: for all data X. For instance, **pass** and **null** are complementary spaces. In fact **null** is the only space complementary to **pass**. On the other hand, every space is complementary to **null**.

5.4 Morphisms

Given spaces S and T, data F is a morphism from S to T if F is equal to T·f·S for some data f, and this may be denoted by $S\xrightarrow{F}T$. Note that F·S=F=T·F, and, in particular, any endomorphism of a space S commutes with S. As an example, F = T·**pass**·S = T·S, so T·S is a morphism from S to T and every space S is an endomorphism of itself, $S\xrightarrow{S}S$. An endomorphism (like S), which happens to be a space as well as an endomorphism is called a *space endomorphism.* As defined, a morphism can be thought of as an ordinary function mapping data S:X in S to data F:S:X in T. In order to restrict ourselves to morphisms 'repecting a structure', we say that a *space with structure* is a space with one distinguished space endomorphism.

> *A* **space with structure** *is a space with a distinguished space endomorphism. A* **morphism** *from space S with structure s to space T with structure t is a morphism F from S to T where t·F = F·s.*

As an example, consider the space **N** of natural numbers stored in n-atoms, with typical data (n:2) (n:6) (n:0) (n:4). As with all spaces, **N** is a space endomorphism of itself. There is also an endomorphism that sums such data: **sum**:(n:2) (n:6) (n:0) (n:4) = (n:12); and an endomorphism that sorts such data **sort**:(n:2) (n:6) (n:0) (n:4) = (n:0) (n:2) (n:4) (n:6). Both **sum** and **sort** are spaces as well as morphisms and therefore are potential structure endomorphisms. Endomorphisms of the space **N** with structure **N** are just ordinary functions from **N** to **N**. Endomorphism of **N** with structure **sum** are the linear maps from **N** to **N**, and endomorphisms of **N** with structure **sort** are the order-preserving maps from **N** to **N**. Since **sum** and **sort** commute, **N** with structure **sum·sort** is also a space, and the resulting structure morphisms preserve both sums and order. Since space S with structure S is just our original definition, we don't have to distinguish *spaces* from *spaces with structure*, and we will refer to both as just *spaces*, where the structure of a space S is understood to be S unless another choice is specified.

$$\begin{tikzcd}
S \arrow[r, "F"] \arrow[d, "s"'] & T \arrow[r, "G"] \arrow[d, "t"'] & \mathcal{U} \arrow[d, "u"'] \\
S \arrow[r, "F"] & T \arrow[r, "F"] & \mathcal{U}
\end{tikzcd}
\qquad
\begin{tikzcd}
S \arrow[loop left] \arrow[r, bend left, "\alpha"] & T \arrow[l, bend left, "\alpha'"] \arrow[loop right]
\end{tikzcd}$$

Figure 2: *The diagram on the left demonstrates that $S\xrightarrow{F}T$ and $T\xrightarrow{G}U$ implies $S\xrightarrow{G\cdot F}U$, where space S has structure s, space T has structure t, and space U has structure u. Spaces S and T are said to be isomorphic, if the diagram on the right commutes for some morphisms α and α'.*

Figure 2 illustrates these definitions and demonstrates that composition of morphisms is associative. If $S\xrightarrow{F}T$ and $T\xrightarrow{G}U$, then $S\xrightarrow{G\cdot F}U$, where S, T and U are spaces (with structure). Figure 2 also illustrates how definitions can be adopted from Category Theory. For example, we say that spaces S and T are **isomorphic** if the diagram of Figure 2(c) commutes for some morphisms α and α', denoted by $S\overset{\alpha}{\cong}T$.

Given $S\xrightarrow{F}T$, data F:X for any X is said to be in the *image* of F and data (S:X) is said to be in the *kernel* of F if (F:X)=(F:). F may happen to be a space as well as a morphism. In this case, the image of F is the same as the contents of F as a space. If F and G are morphisms from space S with structure s to space T with structure t, we can define the *sum* of F and G by

$$\text{F+G} = t\cdot(\text{F}\oplus\text{G})$$

Note the fact that F+G is a morphism relies on t being a space, so that (F+G)$\cdot s$=$t\cdot$(F+G). This addition is associative and is commutative if T is an algebraic space. If S and T are the same space, the composition F·G is also a morphism since F·G·s=F·s·G=s·F·G, so the endomorphisms of a space form an algebraic structure.

5.5 The Algebra of Endomorphisms

If f, g, and h are endomorphisms of a space S with structure s, then $f \cdot (g \cdot h) = (f \cdot g) \cdot h$ is an associative multiplication and $f + (g + h) = (f + g) + h$ is an associative addition if we define $f + g = s \cdot (f \oplus g)$. Letting 1=S and 0=S·**null**·S, we have $1 \cdot f = f \cdot 1 = f$ and $0 + f = f + 0 = f$. Multiplication distributes over addition from the right $(f + g) \cdot h = (f \cdot h) + (g \cdot h)$.

Definition 5. *A* **semiring** *is a collection of data with an associative multiplication with unit 1, with associative addition with unit 0, and where multiplication distributes over addition, from the right, as in* $(f + g) \cdot h = (f \cdot h) + (g \cdot h)$.

In summary, the endomorphisms of a space are a semiring with operations as described. If f, g, and h are endomorphisms of S with structures s, we have the following.

- If f is a space as well as an endomorphism, $f \cdot (g + h) = f \cdot ((f \cdot g) + (f \cdot h))$. If f is a distributive space then $f \cdot (g + h) = (f \cdot g) + (f \cdot h)$, so f distributes from the left and the right.

- $0 \cdot f = 0$. If f is a space, $f \cdot 0 = 0$ also. Since 1=S is always a space, $1 \cdot 0 = 0$. If 1=0, S=S·**null**, so (S:X)=(S:) for all data X, in other words, S is a *constant* space.

- If f:X Y = f:Y X for all X,Y, f is an **algebraic** endomorphism. If f is algebraic, and g is any endomorphism, $g \cdot f$ is algebraic. If f and g are algebraic, so is $f + g$. Thus, if Alg(S) are the algebraic endomorphisms, Alg(S) is a sub-semiring and is an ideal in the sense that f·Alg(S)=Alg(S).

- Data (S:X) in S is **in the kernel** of f if (f:S:X) = (S:). If f is a space, then (S:X) and (S:Y) in the kernel of f implies (S:X) (S:Y) is also in the kernel. If the converse holds, we say that f is a **positive** space. If f and g are positive, so is $f \cdot g$ and $f + g$ and 1 and 0, so positive endomorphism are a sub-semiring of mor(S).

- f is an *involution* if $f \cdot f = 1$. Involutions f and g commute if and only if $f \cdot g$ is an involution. If $f \cdot f = f$, f is *idempotent.*

- If $f \cdot f' = f' \cdot f = 1$ for some endomorphism f', then f and f' are **units** in the group of units.

- If s=S, then every endomorphism is a product $u \cdot i$ of a unit u and an idempotent endomorphism i.

6 Examples

6.1 pass and null

By definition, **pass**:X=X and **null**:X=() for any data X, so **pass** and **null** are both distributive, positive spaces. As containers, **pass** contains all data and **null** contains only the empty sequence. Of the two, only **null** is an *algebraic* space, and is "more mathematical" than **pass**, which, after all, contains everything. From a preorder perspective, **null**≤X≤**pass** for all X, **pass** is only *compatible* with itself, and **null** is *compatible* with every **constant** space. The kernel of **pass** is the empty sequence only, while every data is in the kernel of **null**. **pass** and **null** are also *complementary* since they contain each other's kernels.

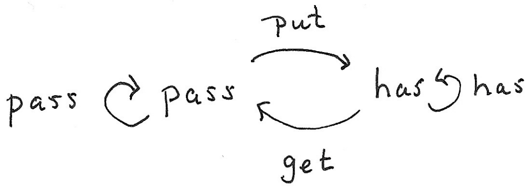


Figure 3: *In a definition adopted from Category Theory, if S with structure s and T with structure t are isomorphic, the top diagram commutes for some morphism α and some morphism α′ where S and T function as their own 'identity morphisms.'*

Since **null**·f**null**=**null**, **null** is the only endomorphism of **null**, so the semiring of **null** contains only one data. On the other hand, the semiring of **pass** includes all data with product $f \cdot g$ and with $f + g =$ **pass** $\cdot\, (f \oplus g)$. Since **pass**·f·**pass**=f, any data f is an endomorphism of **pass**. Thus, every idempotent is an idempotent endomorphism of **pass**, any space is a space endomorphism of **pass**. In a sense, **pass** and **null** are the simplest examples of spaces. On the other hand a complete analysis of the semiring of **pass** is far out of reach.

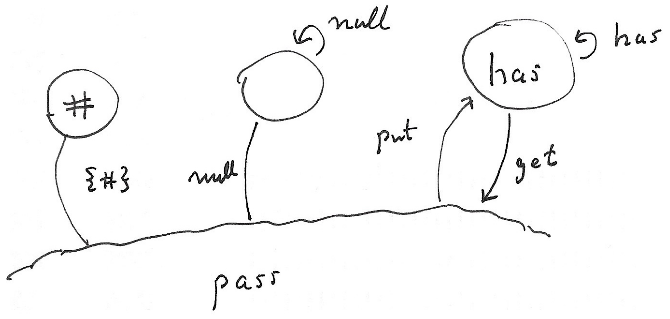


Figure 4: *In a definition adopted from Category Theory, if S with structure s and T with structure t are isomorphic, the top diagram commutes for some morphism α and some morphism α′ where S and T function as their own 'identity morphisms.'*

As might be expected, **pass** is isomorphic to subspaces of itself. For example (Figure X), **pass** is isomorphic to the space **has** of (:X) atoms.

6.2 Natural Numbers

Since concatenation is one of the two foundational operations, that suggests that the most "organic" version of natural numbers is to represent a natural number n by a sequence of n copies of some standard atom, for instance, if N is **ap** {|}, N:X is a sequence of "sticks", one for each atom in X, and (N:X Y) is the natural number sum of the number of atoms in X plus the number of atoms in Y. Since {|} is idempotent, N is a distributive space, and, therefore, a positive space with empty neutral data. The kernel of N is also the empty sequence since only Without additional structure, the endomorphisms of N are data N$\cdot f \cdot$N for some data f, and this includes any function from naturals to naturals in the ordinary sense.

To add structure to N or to any space, one must find a space endomorphism of N, such as

(a) N is a space endomorphism of itself.

(b) **const** | | |, is a space endomorphism, since all constants are spaces.

(c) **mod** | |, is a space endomorphism (mod | |:X is the number of atoms in X modulo 2)

(d) **less** | | |, (min | | |:X is the minimum of 3 and the number of atoms in X)

where **const** and **less** happen to be pre-defined (see Glossary), and **mod** has been defined via

- def mod : {while frontstrip A:B}

so, for instance, **mod** | |:X repeatedly removes two vertical bars, if possible, stopping when the result no longer changes.

6.3 An isomorphism theorem

Fact. *If space $S \overset{\alpha}{\cong} T$ is an isomorphism of spaces, then the monoids of S and T are isomorphic. If α and it's inverse are distributive, the semirings of S and T are isomorphic as well.*

As a consequence, S and T have the same involutions, idempotents, units, structure endomorphisms. If α and it's inverse are distributive, then S and T also have the same spaces. [Q?] If S is algebraic, is T also? If S is positive, is T also?

7 Glossary

For example, starting with context δ, we can add more atoms of the form $\delta_a : c \mapsto c$:

- $\delta_{(:)}$: ((:) A : B) $\mapsto$ ((:) A : B), *single bit atoms where ((:):), ((:):(:)) are zero,one respectively.*

- $\delta_{((:):)}$: (((:):) A : B) $\mapsto$ (((:):) A : B), *eight bit bytes.*

- $\delta_{((:):(:))}$: (((:):(:)) A : B) $\mapsto$ (((:):(:)) A : B), *byte sequences.*

so that text strings or other data are atomic data within $\delta \cup \delta_{(:)} \cup \delta_{((:):)} \cup \delta_{((:):(:))}$. We can then proceed to add named definitions that 'get coda components':

- $\delta_{\textbf{pass}}$: (**pass** A:B) $\mapsto$ B

- $\delta_{\textbf{null}}$: (**null** A:B) $\mapsto$ ()

- $\delta_{\textbf{right}}$: (**right** A:B) $\mapsto$ B

$$\begin{tikzcd}
S \arrow[r, phantom, "\circlearrowleft"] & S \arrow[r, bend left, "\alpha"] & T \arrow[l, bend left, "\alpha'"] \arrow[r, phantom, "\circlearrowright"] & T
\end{tikzcd}$$

$$\begin{tikzcd}
S \arrow[r, "\alpha"] \arrow[d, "s"'] & T \arrow[d, "t"] & & S & T \arrow[l, "\alpha'"'] \arrow[d, "t"] \\
S \arrow[r, "\alpha"] & T & \text{and} & S \arrow[u, phantom] & T \arrow[l, "\alpha'"']
\end{tikzcd}$$

$$\begin{tikzcd}
S \arrow[d, "s"'] & T \arrow[l, "\alpha'"'] \arrow[d, "t"] \\
S & T \arrow[l, "\alpha'"']
\end{tikzcd}$$

Figure 5: *In a definition adopted from Category Theory, if S with structure s and T with structure t are isomorphic, the top diagram commutes for some morphism α and some morphism α' where S and T function as their own 'identity morphisms.'*

- $\delta_{\mathbf{arg}}$: (**arg** A:B) $\mapsto$ A

and can add definitions for simple combinatorics

- $\delta_{\mathbf{ap}}$: (**ap** A : b B) $\mapsto$ (A:b) (ap A:B), *apply A to each atom in B.*
- $\delta_{\mathbf{rev}}$: (**rev** A : B b) $\mapsto$ b (rev : B), *reverse the order of atoms in B.*

definitions computing the boolean value of data

- $\delta_{\mathbf{bool}}$: (**bool** A : B b) $\mapsto$ () if B is empty, (:) if B is atomic, *logical value of B.*

definitions making new definitions

- $\delta_{\mathbf{def}}$: (**def** a : B) $\mapsto$ Add δ_a to context if a is an unused invariant atom.

As explained in reference [1], a language is introduced merely as one more definition where the domain of the language context $\delta_{\{\}}$ is codas starting with a curly brace as in ({*language expression*} A : B).

- $\delta_{\mathbf{bool}}$: (**bool** A : B b) $\mapsto$ () if B is empty, (:) if B is atomic, *logical value of B.*

definitions making new definitions

- $\delta_{\mathbf{def}}$: (**def** a : B) $\mapsto$ Add δ_a to context if a is an unused invariant atom.

As explained in reference [1], a language is introduced merely as one more definition where the domain of the language context $\delta_{\{\}}$ is codas starting with a curly brace as in ({*language expression*} A : B).

References

[1] *Pure Data Foundation of Mathematics and Computing*, Saul Youssef, 2023.

[2] Nicholas Griffin (2003-06-23). *The Cambridge Companion to Bertrand Russell*. Cambridge University Press. p. 63. ISBN 978-0-521-63634-6.

[3] Barry Mazur, *When is one thing equal to some other thing?*, Harvard University, 2007, https://people.math.harvard.edu/∼mazur/preprints/when_is_one.pdf.