AN APPROACH TO THE FORMAL REPRESENTATION OF MATHEMATICAL PROPOSITIONS

by
Harvey M. Friedman*
Distinguished University Professor of Mathematics, Philosophy, and
Computer Science Emeritus
Ohio State University

Presentation at the
Semantic Representation of Mathematical Knowledge Workshop
Fields Institute
Toronto, Canada
February 3, 2016
Revised February 7, 2016

*This material is based upon work supported by the National Science Foundation under Grant No. CCF-1162331. Any opinions, findings, conclusions, or recommendations expressed here are those of the author and do not necessarily reflect the views of the National Science Foundation.

1. POLITICALLY INCORRECT REMARKS

We want to develop a language for the formal representation of mathematical propositions (including axioms and definitions), with a precise syntax and semantics, which supports truly friendly reading and writing. The output must be syntactically and semantically perfect in fully rigorous senses.

I have come to the conclusion that this is impossible to accomplish in a satisfactory way without IMPROVING on existing mathematical notation. Not just modifying existing mathematical notation.

I mean overhauling it in major ways with powerful unifying ideas that preserve, and often improve, readability - at the same time supporting perfectly constructed statements. What I normally see - that is perfectly constructed - doesn't look too much like normal mathematics, and is

generally much less readable.

The overhauling of existing mathematical notation is going to require a lot of creative logic oriented new ideas, if it is really going to be an improvement.

A difficulty is that you start with a few good ideas, but not enough of them. So your prototypes not only don't look like existing mathematics, but are also comparatively unreadable. So nobody is going to encourage you to continue your efforts.

So this project may well be only for retired people. I retired in 2012.

We describe a relatively simple language for expressing mathematical propositions, sketching its absolutely rigorous syntax and semantics. It should work well on some limited kinds of mathematics. In particular, mathematics with a medium level of abstraction, not mired in much conventional notation.

Most obviously, it should be well suited to elementary set theory. It seems like it should support elementary number theory and elementary analysis, but some of this notation needs to be overhauled in order to work with this approach. I'm thinking particularly of summation and integration. In this approach, there is the simplifying and unifying idea of having only one variable binding operator, and that is set abstraction. So summation and integration are to be handled as an ordinary operator from functions to numbers, or functions to functions.

But how are the functions to be presented and handled? By a combination of operators and set abstraction - and some new clever ideas needed to keep things readable.

Also, in multivariable analysis, you see this awkward stuff with partial derivatives and dx dy, and so forth. Under the overhaul, you want to emphasize, e.g., operators that take a function of two variables, and a first argument, and yield the function of one variable obtained by fixing the first argument.

What we propose here is a combination of well known ideas, some going back forever, with some new twists combined in perhaps new ways. Your instinct may be to reject this kind

of overhaul as venturing too far from what is normally written. But I have the opposite reaction. I want to persist with the overhaul, and to make the overhaul work, there will have to be yet more overhaul, and tutorials with candid feedback in the wider mathematical community.

I have some confidence that in the end, it will result in systematic improvements over existing notation - and make a digital library of the kind people are envisioning, more feasible.

I now want to delve into the approach.

2. ABOUT LIBRARIES

Readability of the library can be improved by providing user controlled tools for modifying the displays of selected portions of the library. We envision these tools to be only for personal use, and the system makes no guarantee as to the soundness of such modified displays.

Ultimately, we expect third parties to provide packages, and perhaps soundness proofs for these packages.

The most ambitious kind of digital library project will support logical manipulation of statements in the library.

This more ambitious aim would include recognizing closely related but different forms of the same theorems.

For such more ambitious purposes, there will be pressure to use a much more complicated universal language than is being proposed here - likely borrowing from existing facilities in general purpose formal verification.

However, I fear that this more ambitious enterprise will lead to a language not suited for general non computer science mathematicians, and delay the rolling out of a useful product.

We think that this limited digital library project is well worth the effort - and can also serve as a tangible prototype for more ambitious digital library projects.

This language UL has ZFC at its core, and is relatively simple, though necessarily significantly more involved than any standard formaliza-tion of ZFC used by mathematical

logicians.

ZFC itself is based on a very clean division between logic and rudimentary set theory. Here we go a long way to keep a clean division between "extended free logic" and rudimentary set theory. But we compromise on this as we see that pushing this clean division further leads to undesirable complications. This naturally leads to the present setup that is biased toward the set theoretic way of doing things.

All entries in the envisioned library will consist of statements, in a hypertext environment. Statements are further divided into definitions, axioms, theorems, and open questions. Informal information concerning each entry is available by clicking on it or its various components.

Theorems entered must pass a stringent process ensuring that it follows logically from the axioms (ZFC) and definitions that precede it. This may not include formal verification.
The crucial relevant notion of logical implication here is subject to fully rigorous analysis.

In this preliminary discussion of UL, I won't attempt to incorporate TEX, as has already been done with some systems.

We expect that UL can be made reasonably TEX friendly.

3. FREE LOGIC - basics

We strictly adhere to a well known standard version of free logic, where terms may be undefined. However, propositions have definite truth values, true or false.

$s = t$ means that s,t are both defined and equal.
$s \approx t$ means that $s = t$ or s,t are both undefined.
$t\downarrow$ means that t is defined.
$t\uparrow$ means that t is undefined.

In order for t to be defined, it must be the case that all of its subterms are defined.

Variables and constants are always defined.

Any atomic formula that connects terms by relation symbols, including $=, \uparrow$, but excluding $\approx, \downarrow$, is automatically false if any of the terms being connected are undefined.

E.g., 1 div 0 is undefined. 1 div 0 = 1 div 0 is false. 1 div 0 $\approx$ 1 div 2 is true. Here div is 2-ary division of real numbers.

There is a well known appropriate relative completeness theorem for free logic.

4. EXTENDED FREE LOGIC

The part of our setup that is prior to ZFC, or any set existence axioms whatsoever, is "extended free logic". Almost all of it is not set theoretic. Removing all set theoretic bias in extended free logic seems to cause some difficulties that are best avoided.

In extended free logic, we have a fixed set of variables, constant symbols, relation symbols, and function symbols. There are the special variables x_i, $i \geq 1$, that range over everything. The remaining variables are subject to being restricted to some nonempty range of objects.

We support prefix and infix notation. The semantics of infix notation is derived from the semantics of prefix notation.

The only bracketing operators are { } and < >, and they are of arbitrary arity. All relation and function symbols are of fixed arity.

The only variable binding operator is set abstraction, {_ : _}.

There is a standard relative completeness theorem for extended free logic. The library conforms to this relative validity for extended free logic, relative to the axioms of ZFC.

The semantics is based on (extended free logic) structures. The variables, constant symbols, relation symbols, and function symbols are fixed annotated strings (see below).

The domain is a nonempty set D. Variables are assigned subsets of D as their range of values. The special variables xi, i ≥ 1, are assigned D.

Constant symbols are assigned at most one element of D. k-ary relation symbols: subsets of D^k. k-ary function symbols: partial $f:D^k \rightarrow D$. { } and k ≥ 1: partial $f:D^k \rightarrow D$. < > and k ≥ 1: partial $g:D^k \rightarrow D$. {_:_} - a partial $h:\wp(D) \rightarrow D$.

A crucial feature of extended free logic is: every object takes on the role of object and k-ary relation and par-tial k-ary function, simultaneously, for all k ≥ 1. Accordingly, as part of the structure, to each x ∈ D and k ≥ 1, we assign an $A \subseteq D^k$ and a partial $f:D^k \rightarrow D$.

The syntax is sketched below. Among the notable features are the atomic formulas $s[t_1,...,t_k]$, for "s, as a k-ary relation, holds of $t_1,...,t_k$", and the terms $s(t_1,...,t_k)$, for "the value of s, as a partial k-ary function, at $t_1,...,t_k$".

The Tarskian semantics is carefully defined as expected. There is a relative completeness theorem, with nice axioms and rules of inference.

5. THE INTENDED SET THEORETIC STRUCTURES

We use the usual cumulative hierarchy of sets, given semi formally as

$V(0) = \varnothing$
$V(\alpha+1) = \wp(V(\alpha))$.
$V = \cup_\alpha V(\alpha)$.

The 2-ary relation symbol ∈ is interpreted as membership.

We interpret $\{x_1,...,x_k\}$ as expected. We interpret $<x,y>$ as $\{\{x\},\{x,y\}\}$. For k ≥ 2, we interpret $<x_1,...,x_{k+1}>$ as $<x_1,<x_2,..., x_{k+2}>>$. We interpret <x> as x.

We interpret {_:_} as $f:\wp(V) \rightarrow V$, where f(X) is X if X is a set; undefined otherwise

We interpret $x[y_1,...,y_k]$ as $<y_1,...,y_k> \in x$, k ≥ 1.

We interpret $x(y_1,...,y_k)$ as the unique z such that $<y_1,...,y_k,z> \in x$ if it exists; undefined otherwise.

The ranges of variables other than the xi, i ≥ 1, and the interpretations of constant, relation, and function symbols, are fluid as indicated earlier.

All library entries must hold in all set theoretic structures.

We also use the more general ZFC structures, where (V,∈) is replaced by any model of ZFC.

All library entries must hold in all ZFC structures.

Since the library contains definitions, including variable restrictions, we need to be careful about what "hold" here means.

There is a relative completeness theorem for libraries, involving library continuations.

6. LETTERS, SIGNS

We use a large standard finite set V of letters. A few of these letters have special significance for extended free logic:

= ↑ ↓ ≈ ( ) [ ] { } < > , : ¬ ∧ ∨ → ↔ ∀ ∃ ! x 0 1 2 3 4 5 6 7 8 9

Every letter comes with a normal version, a subscripted version, and a superscripted version.

Every library starts with the core, which is a standard axiomatization of ZFC in extended free logic using only the above symbols plus the special 2-ary relation symbol ∈. These ZFC axioms include the standard treatments of {...}, <...>, {_:_}, x[...], x(...).

The library must continue in a manner that is semantically valid relative to this core.

We envision a standard initial package that goes well beyond this core. It will treat the obvious nonlogical statements involving the following wider letter list:

∈ ∉ = ≠ ∪ ∩ ⊆ ⊇ ⊂ ⊃ ⊄ \ ℘ ∅
↑ ↓ ≈
( ) [ ] { } < > , :
¬ ∧ ∨ → ↔ ∀ ∃ !

We now begin to provide some details concerning the syntax.

The signs are of four disjoint kinds: variables, constant symbols, relation symbols, function symbols. Each of these four consist of the nonempty finite strings from V, with tiny characters in the front and back that tell us which of the four categories we intend, and the intended arity. However, normally, the category and arity can be inferred, and so it does not have to appear. In any case, the user has the ability to show or hide these tiny characters.

The variables are strings from V, with a tiny v in front and a tiny v in back.

The constant symbols are strings from V, with a tiny c in front and a tiny c in back.

The relation symbols are strings from V, with a tiny r in front, and its arity (≥1) in back, again tiny.

The function symbols are strings from V, with a tiny f in front, and its arity (≥1) in back, again tiny.

The user has the option of showing or hiding the tiny characters.

In a revised form of UL, we should be able to omit the use of these tiny characters.

7. USING SIGNS

Only certain signs are actually used in the library. With exceptions, all signs used in the library are introduced by a definition prior to its use. A sign is introduced at most once. Thus the meaning of a sign cannot change. This is workable, as the user can choose to modify the display.

The exceptional signs that do not have to be introduced, are the 2-ary relation symbols ∈,=, the unrestricted variables xi, i ≥ 1, and those variables that the initial

system package, and later the user, chooses to range over everything. I.e., restricted variables are introduced with restrict-ion spelled out, before use. Variable introductions are considered definitions.

8. INTRODUCING VARIABLES

DEFINITION. Introducing α. $\alpha\downarrow \leftrightarrow \varphi$.

Here α is a variable (not an xi) and φ is a formula with exactly the free variable α.

UPDATE: An alternative is to only use variables ranging over everything, and restrict variables by local hypotheses or where clauses. This and some other related design issues will be carefully addressed as we begin to create actual mathematical text (axioms, definitions, theorems only).

9. INTRODUCING CONSTANT SYMBOLS

DEFINITION. $c \approx t$.

Here c is a constant symbol and t is a term without c.

10. INTRODUCING RELATION SYMBOLS

DEFINITION. $R(x_1,...,x_k) \leftrightarrow \varphi$.

Here R is a k-ary relation symbol, and φ is a formula not mentioning R whose free variables are among x1,...,xk.

11. INTRODUCING FUNCTION SYMBOLS

DEFINITION. $F(x_1,...,x_k) \approx t$.

Here F is a k-ary function symbol, and t is a term not mentioning F, whose free variables are among $x_1,....,x_k$. NOTE: All relation and function symbols are introduced in prefix form. But we will support infix notation, as indicated below.

NOTE: In the appropriate sense, all of these definition forms are unambiguous and benign.

12. TERMS

Every constant symbol and variable is a term.

$s(t_1,...,t_k)$, $\alpha(t_1,...,t_k)$ are terms if α is a k-ary function symbol and $s,t_1,...,t_k$ are terms, $k \geq 1$.

$(!t)(\varphi)$, $(!t\ \gamma\ s)(\varphi\}$ are terms if s,t are terms, φ is a formula, and γ is a 2-ary relation symbol.

$\{t_1,...,t_k\}$, $<t_1,...,t_k>$ are terms if $t_1,...,t_k$ are terms.

$\{t: \varphi\}$, $\{t\ \gamma\ s: \varphi\}$ are terms if s,t are terms, φ is a formula, and γ is a 2-ary relation symbol.
NOTE: $\{t: \varphi\}$, etc., is the set of all values of t such that the free variables in t obey φ. E.g., $\{F(x,y): \varphi\}$. If we have y fixed, this is a one dimensional sum, and we write $\{F(x,z): \varphi[y/z] \wedge z = y\}$.

$t_1\ t_2\ ...\ t_k$ is a term if

a) k is odd.
b) each t_i, i odd, is a term.
c) each t_i, i even, is a term or a 2-ary function symbol.

Here terms t_i, i even, represent sets in their role as binary functions. For $k \geq 5$, $t_1\ t_2\ ...\ t_k$ is evaluated left associatively.

13. ATOMIC FORMULAS

$t\uparrow$, $t\downarrow$, where t is a term.
$s[t_1,...,t_k]$, $\alpha[t_1,...,t_k]$, where α is a relation symbol and $s,t_1,...,t_k$ are terms, $k \geq 1$.

$t_1\ t_2\ ...\ t_k$ is an atomic formula if

a) k is odd.
b) each t_i, i odd, is a term.
c) each t_i, i even, is a term or a 2-ary relation symbol.

Here terms t_i, i even, represent sets in their role as binary relations. $t_1\ t_2\ ...\ t_k$ is read conjunctively as $t_1\ t_2\ t_3 \wedge t_3\ t_4\ t_5 \wedge ... \wedge t_{k-2}\ t_{k-1}\ t_k$.

14. FORMULAS

Closure under $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, with precedence rules.

$(\forall \alpha_1, \ldots, \alpha_k)(\varphi)$, $(\exists \alpha_1, \ldots, \alpha_k)(\varphi)$, $(\exists! \alpha_1, \ldots, \alpha_k)(\varphi)$.
$(\forall \alpha_1, \ldots, \alpha_k\ \gamma\ t)(\varphi)$, $(\exists \alpha_1, \ldots, \alpha_k\ \gamma\ t)(\varphi)$, $(\exists! \alpha_1, \ldots, \alpha_k\ \gamma\ t)(\varphi)$.

where $\alpha_1, \ldots, \alpha_k$ are distinct variables, t is a term, φ is a formula, and: γ is a 2-ary relation symbol.

15. ASSUME, THEN, TFAE, EIN

Assume $\varphi_1, \ldots, \varphi_k$. THEN $\psi_1, \ldots, \psi_r$.

The following are equivalent:
φ_1.
...
φ_k.

Each implies the next:
φ_1.
...
φ_k.

16. DEVELOPMENT

There are enough details here to actually produce a small prototype library, and see how it looks. Then the language will be fine tuned.
I propose to start with the easiest: elementary set theory. Then try elementary number theory, elementary algebra, elementary analysis. And so forth.

I will use careful mathematical thinking to "guarantee" that all entries are in the appropriate sense theorems of ZFC.

I suspect that additional features need to be carefully added to the language that are easily managed, but support some worthwhile additional readability. To judge the tradeoffs between language simplicity and readability, we have to get our hands dirty.

17. CORE LIBRARY

AXIOM 1. $x_1 \in \{x_2,...,x_{k+1}\} \leftrightarrow x_1 = x_2 \vee ... \vee x_1 = x_{k+1}$.

AXIOM 2. $\langle x_1,x_2\rangle = \{\{x_1\},\{x_1,x_2\}\} \wedge \langle x_1\rangle = x_1$.

AXIOM 3. $\langle x_1,...,x_k\rangle = \langle\langle x_1,x_2\rangle,x_3,...,x_k\rangle$.

AXIOM 4. $x_1(x_2,...,x_{k+1}) = (!x_{k+2})(\langle x_2,...,x_{k+2}\rangle \in x_1)$.

AXIOM 5. $x_1[x_2,...,x_{k+1}] \leftrightarrow \langle x_2,...,x_{k+1}\rangle \in x_1$.

AXIOM 6. $x_1 \in \{t: \varphi\} \leftrightarrow (\exists x_2,...,x_{k+1})(\varphi \wedge x_1 = t)$. Here t is a term, φ is a formula, x_1 is not in t,φ, and the free variables in t are $x_2,...,x_{k+1}$.

AXIOMS. ZFC, taking advantage of axioms 1-6.

18. INITIAL LIBRARY

The initial library package includes the core library plus a number of definitions involving the most commonly used symbols.

There is the basic initial library, which is small and simple. This would fall well short of, say, the definition of the standard number system.

The full initial library package would be much larger, treating the elementary mathematics that virtually every user of mathematics worldwide needs.

DEFINITION 1. $x \notin y \leftrightarrow \neg x \in y$.

DEFINITION 2. $x \neq y \leftrightarrow \neg x = y$.

DEFINITION 3. $\varnothing \approx \{x: x \neq x\}$.

THEOREM 1. $\varnothing\downarrow$.

DEFINITION 4. $x \subseteq y \leftrightarrow (\forall z \in x)(z \in y)$.

DEFINITION 5. $x \supseteq y \leftrightarrow y \subseteq x$

DEFINITION 6. $x \subset y \leftrightarrow x \subseteq y \wedge x \neq y$.

DEFINITION 7. $\supset(x,y) \leftrightarrow x \supseteq y \wedge x \neq y$.

DEFINITION 8. $x \not\subset y \leftrightarrow$
$\neg x \subset y$.

DEFINITION 9. $x \cup y \approx \{z: z \in x \vee z \in y\}$.

DEFINITION 10. $x \cap y = \{z: z \in x \wedge z \in y\}$.

DEFINITION 11. $x \backslash y \approx \{z \in x: z \notin y\}$.

DEFINITION 12. $\wp(x) = \{y: y \subseteq x\}$.

DEFINITION 13. $\cup(x) = \{y: (\exists z \in x)(y \in z)\}$.

DEFINITION 14. $\cap(x) \approx \{y: (\forall z \in x)(y \in z)\}$.

THEOREM 2. $x \cup y\downarrow$. $x \cap y\downarrow$. $x \backslash y\downarrow$. $\wp(x)\downarrow$. $\cup(x)\downarrow$.

THEOREM 3. $x \neq \varnothing \rightarrow \cap x\downarrow$.

DEFINITION 15. $0 \approx \varnothing$. $1 \approx 0 \cup \{0\}$. $2 \approx 1 \cup \{1\}$. $3 \approx 2 \cup \{2\}$. $4 \approx 3 \cup \{3\}$. $5 \approx 4 \cup \{4\}$. $6 \approx 5 \cup \{5\}$. $7 \approx 6 \cup \{6\}$. $8 \approx 7 \cup \{7\}$. $9 \approx 8 \cup \{8\}$.

THEOREM 4. $0 = \varnothing$. $1 = 0 \cup \{0\}$. $2 = 1 \cup \{1\}$. $3 = 2 \cup \{2\}$. $4 = 3 \cup \{3\}$. $5 = 4 \cup \{4\}$. $6 = 5 \cup \{5\}$. $7 = 6 \cup \{6\}$. $8 = 7 \cup \{7\}$. $9 = 8 \cup \{8\}$.
52085 is 5 α 2 α 0 α 8 α 5, for the suitable 2-ary function symbol α from the more advanced part of the initial library package, which is to be evaluated left associatively. c's, and spaces, to see 52085.

Obviously, we want n α m = 10n+m.

19. CONTINUING...

In an elaboration of ideas, we plan to

i. Treat extended free logic, its semantics, and the relative completeness theorem.

ii. Define the notion of extended free logic library.

iii. Define extended free logic libraries over ZFC.

iv. Cover a variety of rich mathematical statements. Refine language according to experience.

v. Years ago, I proposed to create a database of what I called CONCEPT TREES. This is basically the definitional structure only in the library. A mathematical notion is traced back to the set theoretic core, obtaining a labeled tree. How do the structure of these trees compare from area to area of mathematics?