Modeling Formal Languages in Grammatical Framework

On the Grammar of Proof

Warrick Macmillan

7^{th} August 2021

Table of Contents

1. Explore abstract relationships between mathematics, CS (TT in particular), and linguistics
2. Practical and brief intro to MLTT and Agda
3. Grammars elaborating the abstractions above
4. Thoughts about the future and conclusions

Abbreviations

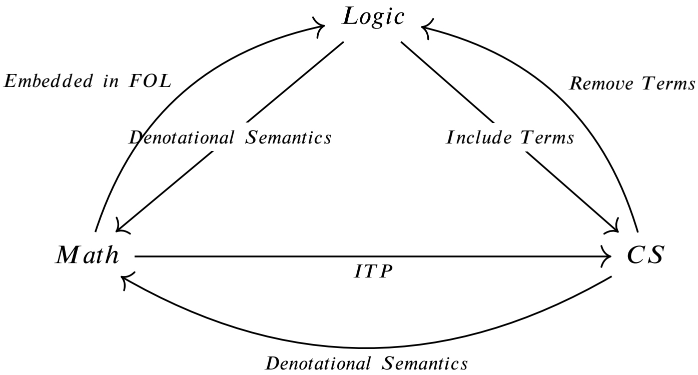
- **TT** : Type Theory
- **MLTT** : Martin-Löf Type Theory
- **MLTT** : Homotopy Type Theory
- **CTT** : Cubical Type Theory
- **NL** : Natural Language
- **PL** : Programming Language
- **GF** : Grammatical Framework
- **PGF** : Portable Grammar Format
- **ITP** : Interactive Theorem Prover
- **FOL** : First Order Logic
- **BNF** : Backus-Naur form
- **CADE** : Conference on Automated Deduction
- **HOL** : Higher Order Logic

Abstraction Ladders

$$
\begin{tikzcd}
Strings \arrow[d, "Lexical\ Analysis"] \arrow[dd, bend right=60, "Front\ End"'] \\
Lexemes \arrow[d, "Parsing"] \\
ASTs \arrow[d, "Type\ Checker"] \\
Typed\ ASTs \arrow[d, bend right, "Interpreter"'] \arrow[d, bend left, "Code\ Generator"] \arrow[d, phantom, "\cdots"] \\
Target\ Language
\end{tikzcd}
$$

$$
\begin{tikzcd}
Phonemes \arrow[d, dashed, "Morhphophonological"] \\
Morphemes \arrow[d, "Parse"] \\
\{\ Syntactic\ Representation\ \} \arrow[d, bend right, "Montague"'] \arrow[d, dashed, "\cdots"] \arrow[d, bend left, "Ranta"] \\
\{\ STLC,\ \ldots,\ DTT\ \} \arrow[d, dashed, "?"] \\
\{\ NearalEncoding\ ,\ \ldots\ ,\ I\ \ Language\ \} \arrow[d, dashed, "?"] \\
Phonemes
\end{tikzcd}
$$

Computational Trinitarianism



Interpretation of Language

Observation 1.1

Only semantic interpretations in these domains. One may decide on syntactic, pragmatic, or other paradigms to view this through.

$$
\begin{tikzcd}
 & Logic & \\
 & Linguistics \arrow[u, "Montague\ Semantics"] \arrow[dl, "Distributional\ Semantics"'] \arrow[dr, "TT\ Semantics"] & \\
Math & & CS\ (MLTT)
\end{tikzcd}
$$

Trinitarian Linguistics

$$
\begin{tikzcd}
 & Logic \arrow[d, "Embedding"] & \\
 & Linguistics & \\
Math \arrow[ur, "Language\ Of\ Mathematics"] & & CS\ (MLTT) \arrow[ul, "Meaning\ Explanations"']
\end{tikzcd}
$$

Trinitarian Grammars

$$
\begin{tikzcd}
& Logic \arrow[d, "Ranta\ Logic\ (CADE\ 11)"] & \\
& GF & \\
Math \arrow[ur, "Ranta\ (HoTT\ 14)"] & & CS\ (MLTT) \arrow[ul, "cubicalTT"']
\end{tikzcd}
$$

Models of GF

$$
\begin{tikzcd}
 & Logic & \\
 & GF \arrow[u, "GF\ Parser\ Specification"'] \arrow[dl, "Theory\ of\ Operads"'] \arrow[dr, bend left, "Implementation\ of"] \arrow[dr, bend right, "Agda\ Embedding"'] & \\
Math & & CS\ (MLTT)
\end{tikzcd}
$$

Remarks

Concrete syntax is, in some sense, where programming meets psychology.

(Robert Harper, OPLSS)

- Trinitarian doctrine is in the “formal” space
- Trinitarianism + linguistics is partially informal and very underexplored
- Introduces many philosophical concerns
- Perhaps a rereading of Wittgenstein should take place in this context

Predecessors

- Frege : Formal proof, predicate logic
- Russel : Invents TT to resolve his paradox
- Brouwer : Constructivism
- Heyting, Kolmogorov, Church, Gödel, Kleene, Bishop, ...

Mathematical logic and the relation between logic and mathematics have been interpreted in at least three different ways:

i. mathematical logic as symbolic logic, or logic using mathematical symbolism;
ii. mathematical logic as foundations (or philosophy) of mathematics;
iii. mathematical logic as logic studied by mathematical methods, as a branch of mathematics.

We shall here mainly be interested in mathematical logic in the second sense. What we shall do is also mathematical logic in the first sense, but certainly not in the third.

(Per Martin-Löf, Padua Italy, June 1980)

Props vs Types

First Order Logic

- $\forall$
- $\exists$
- $\supset$
- $\wedge$
- $\vee$
- $\neg$
- $\top$
- $\bot$
- $=$

Dependent Type Theory

- Π
- Σ
- $\rightarrow$
- $\times$
- $+$
- $\neg$
- $\top$
- $\bot$
- $\equiv$

Sets vs Types

Sets

- $\mathbb{N}$
- $\mathbb{N} \times \mathbb{N}$
- $\mathbb{N} \to \mathbb{N}$
- $\{x | P(x)\}$
- $\varnothing$
- $?$
- $\cup$
- $?$

Types

- Nat
- $Nat \times Nat$
- $Nat \to Nat$
- $\Sigma x : \_.P(x)$
- $\bot$
- $\top$
- $?$
- U_1

Sets

- 1
- $(1, 0)$

Programs

- $suc\ zero$
- $(suc\ zero, zero)$

Judgments

Type Theoretic Judgments

- T is a type
- T and T' are equal types
- t is a term of type T
- t and t' are equal terms of type T

Mathematical Judgments

- P is a proposition
- P is true

- Notice that judgmental equality is uniquely type theoretic
- Judgments in type theory are decidable
- Truth (inhabitation) is not decidable
- More exotic judgments are available in TT, i.e. P is possible.

TT vs classical FOL

- The rules of the types make explicit that they are not equivalent to those of classical FOL
- An existential assertion in type theory requires data
- Excluded middle and double negation are not admitted in MLTT
- To be *not unhappy* is clearly of a different meaning than to be *happy*.
- This makes our approach to general translation of non-constructive mathematics *impossible*... at least such that it type-checks
- perhaps this is possible for other TTs, like those based of HOL

Other issues

- One doesn't define logics and type systems in mathematics (e.g. metamathematics)
- Encoding concepts like rational and real numbers in TT are difficult
- Sets are just one way of encoding mathematics
- Already category theorists and set theorists are at odds. Think small and large categories, higher categories, simplicial, cubical, globular, ... enrichment, etc.
- Intensional TT comes with two distinct notions of equality : judgmental (definitional, computational) and propositional

Donkey Anaphora

- Interpret the sentence “every man who owns a donkey beats it” in MLTT via the following judgment :

$$\Pi z : (\Sigma x : man.\ \Sigma y : donkey.\ owns(x, y)).\ beats(\pi_1 z, \pi_1(\pi_2 z))$$

- We judge $\vdash man$:type and $\vdash donkey$:type.
- type really denotes a universe