Modeling Formal Languages in Grammatical Framework

On the Grammar of Proof

Warrick Macmillan

7^{th} August 2021

Recapitulation

1. types such as "$(n\backslash s)$", "n";
2. Has ways of constructing new types and a grammar for "well-formedness";
3. A (finite set) of symbols (such as "poor" and "John"), to which are matched said types.

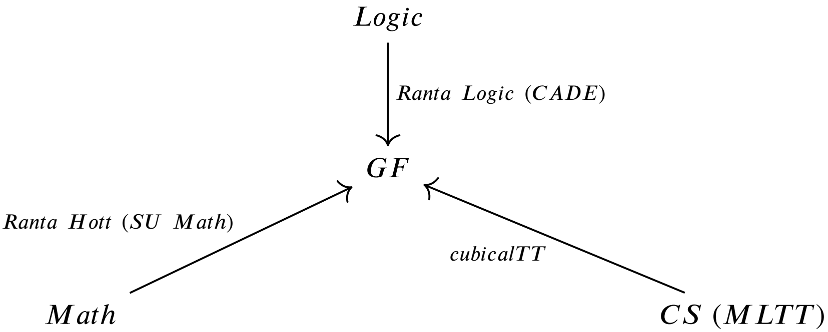
$$
\begin{tikzcd}
Strings \arrow[d, "Lexical\ \ Analysis"] \arrow[dd, bend right=60, "Front\ \ End"'] \\
Lexemes \arrow[d, "Parsing"] \\
ASTs \arrow[d, "Type\ \ Checker"] \\
Typed\ \ ASTs \arrow[d, bend right, "Interpreter"'] \arrow[d, phantom, "\cdots"] \arrow[d, bend left, "Code\ \ Generator"] \\
Target\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

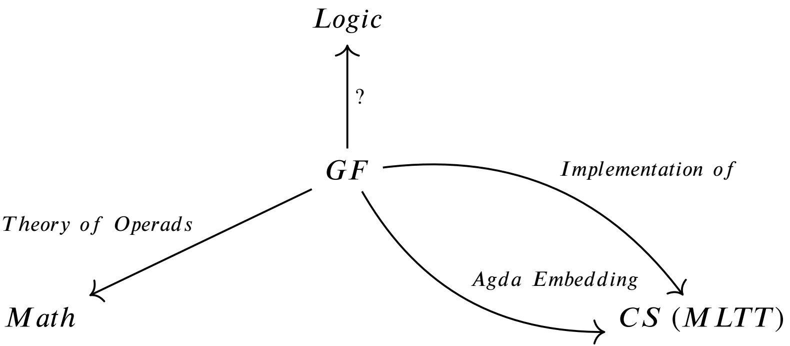
$$
\begin{tikzcd}
Phonemes \arrow[d, dashed, "Morhphophonological" description] \\
Morphemes \arrow[d, "Parse"] \\
\{\ Syntactic\ \ Representation\ \} \arrow[d, bend right, "Montague"'] \arrow[d, dashed, "\cdots" description] \arrow[d, bend left, "Ranta"] \\
\{\ STLC,\ \ldots\ ,\ DTT\ \} \arrow[d, dashed,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

$$
\begin{tikzcd}
 & Logic \arrow[dl, "Denotational\ Semantics"'] \arrow[dr, "Include\ Terms"] & \\
Math \arrow[ur, bend left, "Embedded\ in\ FOL"] \arrow[rr, "ITP"'] & & CS \arrow[ul, bend right, "Remove\ Terms"'] \arrow[ll, bend left, "Denotational\ Semantics"]
\end{tikzcd}
$$

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

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


Logic
Ranta Logic (CADE)
GF
Ranta Hott (SU Math)
cubicalTT
Math
CS (MLTT)


Logic
?
GF
Implementation of
Theory of Operads
Agda Embedding
Math
CS (MLTT)

MLTT and its relation to FOL Set theory Implement mathematics constructively More recent incarnations HoTT, and now cubical

Examples Donkey anaphora | V Dependent Types in GF

What is Agda - Logical Framework - Interactive proof development environment - Inductive Types, Modules, - Implementation of MLTT

Boolean example
(type theoretic syntax)

What is a Proof?

Defintions
Syntctic Completeness Semantic Adequacy

$$
\begin{tikzcd}
Lexicon\ Size & & Syntactic\ Completeness \\
 & & {} \\
Specturm\ of\ GF \arrow[uu, "Statistical\ Methods?"] \arrow[uurr, "cubicalTT"] \arrow[urr, "Ranta\ Logic\ '14"'] \arrow[rr, "Ranta\ HoTT\ '14"'] & & Semantic\ Adequacy
\end{tikzcd}
$$

Overview

1. *Negation* is a contentions notion, in a sense (but we'll not get into this here)
2. Buszkowski added axioms for a kind of negation in categorial grammars
3. Wansing presents a different kind, which leads to motivation of connexive logic

Negative Information

To allow for the expression that "'sleeps John' is an invalid sentence" (it's not just "not valid", it's *invalid*, one could assign it the type "$\neg(s\backslash n)$"
There are many nice connections to algebra, and even category theory (Lambek calculus was inspired on that), but we won't be touching upon (no time).

The Negation Normal Form

Observation 3.4

For every type symbol x, x' is in NNF and $\vdash_S x \Leftrightarrow x'$, for $S \in \{\mathbf{NL}^\neg, \mathbf{L}^\neg\}$.

Definition 3.1: type symbol

1. atomic type symbols $x, y, w, \ldots$ are type symbols;
2. if X and Y are type symbols, also $(X * Y)$ is a type symbol, for $* \in \{\backslash, /, x\}$
3. if X is a type symbol, also $\neg X$ is a type symbol $(X \neq Y \times Z)$;

Negation Normal Form

Define a function $'$ such that:

$$\begin{aligned} x' &= x \quad \text{(x atomic)} \\ (\neg x)' &= \neg x \quad \text{(x atomic)} \\ (\neg\neg X)' &= X' \\ (X * Y)' &= (X' * Y') \\ (\neg(Y/X))' &= ((\neg Y)'/X') \\ (\neg(X\backslash Y))' &= (X'\backslash(\neg Y)') \end{aligned}$$

Proof:

A. x is a type symbol $\rightarrow x'$ is its negation normal form;

B. x is a type symbol $\rightarrow \vdash_S x \Leftrightarrow x'$

Proof:

A. x is a type symbol $\rightarrow x'$ is its negation normal form;

B. x is a type symbol $\rightarrow \vdash_S x \Leftrightarrow x'$

A.

Proof is straightforward by induction on the complexity of type symbols.

(note: $\neg(X * Y)$ is not a valid type symbol here).

B. $(\Rightarrow case)$

By induction on the complexity of x:

1. x is atomic: $x' = x$ and by $(id) \vdash x \Rightarrow x'$;

B. $(\Rightarrow \mathit{case})$

By induction on the complexity of x:

1. x is atomic: $x' = x$ and by $(id) \vdash x \Rightarrow x'$;
2. $x = (y \times w)$: by IH $\vdash y \Leftrightarrow y'$ and $\vdash w \Leftrightarrow w'$

$$\dfrac{\dfrac{y \Rightarrow y' \qquad w \Rightarrow w'}{y, w \Rightarrow (y' \times x')}\,(\rightarrow \times)}{(y \times w) \Rightarrow (y' \times x')}\,(\times \rightarrow)$$

3. $x = (y/w)$: same IH

$$\dfrac{\dfrac{w' \Rightarrow w \qquad y \Rightarrow y'}{(y/w), w' \Rightarrow y'}\,(/ \rightarrow)}{y/w \Rightarrow y'/w'}\,(\rightarrow /)$$

4. $x = (y\backslash w)$: dual.

5 $x = \neg y$: by IH $\vdash y \Leftrightarrow y'$
We can't deduce $\neg y \Leftrightarrow (\neg y)'$ directly. We need to look at y:

5 $x = \neg y$: by IH $\vdash y \Leftrightarrow y'$
We can't deduce $\neg y \Leftrightarrow (\neg y)'$ directly. We need to look at y:

- y atomic then $(\neg y)' = \neg y$ and $\vdash \neg y \Rightarrow \neg y$ by (id);
- $y = \neg w$. We want a proof of $\neg\neg w \Leftrightarrow (\neg\neg w)'$. By IH we know that $\vdash w \Leftrightarrow w'$

$$\dfrac{\dfrac{w \Rightarrow w'}{\neg\neg w \Rightarrow w'}\,(\neg\neg \to)}{\neg\neg w \Rightarrow \neg\neg w'}\,(\to \neg\neg)$$

5 $x = \neg y$: by IH $\vdash y \Leftrightarrow y'$
We can't deduce $\neg y \Leftrightarrow (\neg y)'$ directly. We need to look at y:

- y atomic then $(\neg y)' = \neg y$ and $\vdash \neg y \Rightarrow \neg y$ by (id);
- $y = \neg w$. We want a proof of $\neg\neg w \Leftrightarrow (\neg\neg w)'$. By IH we know that $\vdash w \Leftrightarrow w'$

$$\dfrac{\dfrac{w \Rightarrow w'}{\neg\neg w \Rightarrow w'}\,(\neg\neg \rightarrow)}{\neg\neg w \Rightarrow \neg\neg w'}\,(\rightarrow \neg\neg)$$

- $y = w/z$. We want a proof of $\neg(w/z) \Rightarrow ((\neg w')/z')$ since $(\neg(w/z))' = ((\neg w')/z')$ By IH we can assume

$$z \Leftrightarrow z' \qquad \neg w \Leftrightarrow (\neg w)'$$

$$\dfrac{\dfrac{z' \Rightarrow z \qquad \neg w \Rightarrow (\neg w)'}{\neg(w/z), z' \Rightarrow (\neg w)'}\,(\neg/ \rightarrow)}{\neg(w/z) \Rightarrow ((\neg w)'/z')}\,(\rightarrow /)$$

Definition 3.1: type symbol

1. Atomic type symbols $x, y, w, \ldots$ are type symbols;
2. if X and Y are type symbols, also $(X * Y)$ is a type symbol, for $* \in \{\backslash, /, x\}$
3. if X is a type symbol, also $\neg X$ is a type symbol $(X \neq Y \times Z)$;
4. nothing else is a type symbol.

Definition 3.2: categorial entailment

$$\frac{}{x \Rightarrow x}\ (id)$$

$$\frac{x, X \Rightarrow y}{X \Rightarrow (x\backslash y)}\ (\rightarrow \backslash)$$

$$\frac{X \Rightarrow x \qquad Y, y, Y' \Rightarrow z}{Y, X, (x\backslash y), Y' \Rightarrow z}\ (\backslash \rightarrow)$$