

Compositionality

Pascal Amsili

December 2020

Overview

Frege's principle

Typed λ -calculus

Type theory

Montague's language

First fragment

Frege's principle

The meaning of an expression is uniquely determined by the meanings of its parts and their mode of combination.

Motivation : Humbolt's view on finite means for infinite sentences

Consequences

- Locality principle : lexical items have a meaning that is independant of the expression they occur in.
- Substitution principle : synonymous expressions may be substituted for each other without changing the meaning of the complex expression in which they occur.
- Parts of well formed sentences have « meaning »
- Meanings can be « composed » : Frege's saturation idea

λ -terms can represent individual meanings and functional application can represent semantic composition.

Overview

Frege's principle

Typed λ -calculus

Type theory

Montague's language

First fragment

Type theory

1. e is a type
2. t is a type
3. if a and b are types, then $\langle a, b \rangle$ is a type

Type theory

1. e is a type
2. t is a type
3. if a and b are types, then $\langle a, b \rangle$ is a type
 - $D_e = A$
 - $D_t = \{0, 1\}$
 - $D_{\langle a, b \rangle} =$ the set of mappings from D_a to D_b .

Meaningful expressions

For a, b types :

- variables and individual constants of type a belong to ME_a .
- if $\alpha \in ME_{\langle a,b \rangle}$ and $\beta \in ME_a$ then $(\alpha)\beta \in ME_b$.
- if u is a variable of type a and $\alpha \in ME_b$, then $\lambda u.\alpha \in ME_{\langle a,b \rangle}$.
- if φ and ψ are in ME_t , then the following expressions are also in ME_t : $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$.
- if φ is in ME_t and u is a type a variable, then $\forall u\varphi$ and $\exists u\varphi$ are in ME_t .

Overview

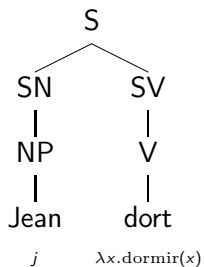
Frege's principle

Typed λ -calculus

Type theory

Montague's language

First fragment



<i>S</i>	→	<i>SN</i>	<i>SV</i>
0	←	(2)	1
<i>SN</i>	→	<i>NP</i>	
0	←	1	
<i>SV</i>	→	<i>V</i>	
0	←	1	
<i>NP</i>	→	Jean	
0	←	<i>j</i>	
<i>V</i>	→	dort	
0	←	$\lambda x.dormir(x)$	