Self Reference

PARADOX AND INFINITY
Benjamin Brast-McKie
February 28, 2024

An Untyped Language

Vicious Circle Priciple: "Whatever contains an apparent variable must not be a possible value of that variable."

Language: Names $c_1, c_2, \ldots \in C$, variables $x_1, x_2, \ldots \in V$, predicates $R_1, R_2, \ldots \in P$, operators $\neg, \vee, \wedge, \rightarrow, \leftrightarrow, \forall\alpha, \exists\alpha$ where α is any variable.

Formulas: The set of *formulas* F is defined recursively:

- $R(\alpha_1, \ldots, \alpha_n)$ is a formula in F if $R \in P$ and $\alpha_1, \ldots, \alpha_n \in C \cup V$.
- $z(\varphi_1, \ldots, \varphi_n)$ is a formula in F if $z \in V$ and $\varphi_1, \ldots, \varphi_n \in C \cup V \cup F$.
- $\neg\varphi, \varphi \vee \psi, \varphi \wedge \psi, \ldots, \forall\alpha\varphi, \exists\alpha\varphi$ are formulas in F if $\varphi, \psi \in F$ and $\alpha \in V$.
- Nothing else is a formula in F.

Example: In the following examples z is of higher type:

- Mathematical induction shows that for any property z, if $z(0)$ and $z(x')$ whenever $z(x)$, then $z(x)$ for all numbers x.
- $\forall z(z \vee \neg z)$.

Self Reference: Observe that $\neg z(z)$ is a formula, but it will not have a type.

Ramified Theory of Types

Simple Types: The simple types will be defined recursively.

- 0 is the simple type of *individuals*.
- If $t_1, \ldots, t_n$ are simple types, then $(t_1, \ldots, t_n)$ is a simple type.
- Nothing else is a simple type.

Example: 'Kim' is type 0, 'is running' is type (0), and 'Kim is running' is type $()$.

Ramified: Also defined recursively:

- 0^0 is a ramified type.
- If $t_1^{o_n}, \ldots, t_n^{o_n}$ are ramified types, $o \in \mathbb{N}$, and $o > \max\{o_1, \ldots, o_n\}$, then $(t_1^{o_1}, \ldots, t_n^{o_n})^o$ is a ramified type where $o \geqslant 0$ if $n = 0$.
- Nothing else is a ramified type.

Example: 'Kim' is type 0^0, 'is running' is type $(0^0)^1$, and 'Kim is running' is type $()^1$.

Predicative Types: t^o is *predicative* if o is as low as it can be, c.f., $(0^0)^2$.

A Typed Language

Atomic Formulas: $R(c_1, \ldots, c_n)$ is *atomic* if R is a predicate and $c_1, \ldots, c_n$ are constants.

Typed Expresions: The expressions of the language will be typed recursively.

- $c : 0^0$ if c is a constant.
- $\varphi : ()^0$ if φ is atomic.
- If $\varphi : (t_1^{o_1}, \ldots, t_n^{o_n})^a$ and $\psi : (d_1^{r_1}, \ldots, d_n^{r_n})^b$, then:
  - $\neg\varphi : (t_1^{o_1}, \ldots, t_n^{o_n})^a$.
  - $\varphi \vee \psi : (t_1^{o_1}, \ldots, t_n^{o_n}, d_1^{r_1}, \ldots, d_n^{r_n})^{\mathtt{max}\{a,b\}}$.

  $\vdots$
- If $(t_1^{o_1}, \ldots, t_n^{o_n})^a$ is predicative and $\varphi_1 : t_1^{o_1}$, ..., $\varphi_n : t_n^{o_n}$ for expressions $\varphi_1, \ldots, \varphi_n$, then $z(\varphi_1, \ldots, \varphi_n) : ((t_1^{o_1}, \ldots, t_n^{o_n})^a, t_1^{o_1}, \ldots, t_n^{o_n})^{a+1}$.
- If $\varphi : (t_1^{o_1}, \ldots, t_n^{o_n})^a$, then $\forall x : t_i^{o_i} \varphi : (t_1^{o_1}, \ldots, t_n^{o_n})^a$.
- There is more that we won't get into...

Typed Formuals: A *typed formula* is a typed expression that is a formula.

- $\neg z(z)$ is a formula, but cannot be typed.
- $\forall z(z \vee \neg z)$ can be typed.

Type Restrictions: "Whatever contains an apparent variable must be of a different type from the possible values of that variable..."

Axiom of Reducibility

Identity of Indiscernibles: $x = y$ *iff* $\forall z[z(x) \leftrightarrow z(y)]$.

Stratification: For each order $n \in \mathbb{N}$, we may articulate a version of the principle above: $x = y$ *iff* $\forall z : (0^0)^n[z(x) \leftrightarrow z(y)]$.

Planets Example: Do Hesperus and Phosphorus have the same first-order properties?

- But what if they differ on second-order properties?
- Maybe someone loves Hesperus but no one loves Phosphorus.

Induction Example: "A finite number is one which possesses *all* properties possessed by 0 and by the successors of all numbers possessing them."

- If something holds of all first-order properties, why think it holds for all higher-order properties?

Axiom of Reducibility: "[E]very propositional function is equivalent, for all its values, to some predicative function." (pp. 242-3)

Translation: Every typed formula is logically equivalent to some formula with a predicative type, and so: $x = y$ *iff* $\forall z : (0^0)^1[z(x) \leftrightarrow z(y)]$.