Semantic Theory
Week 2 – Type Theory

Noortje Venhuizen
Harm Brouwer

Universität des Saarlandes

Summer 2020

First-order logic

First-order logic talks about:

- Individual objects
- Properties of and relations between individual objects
- Quantification over individual objects

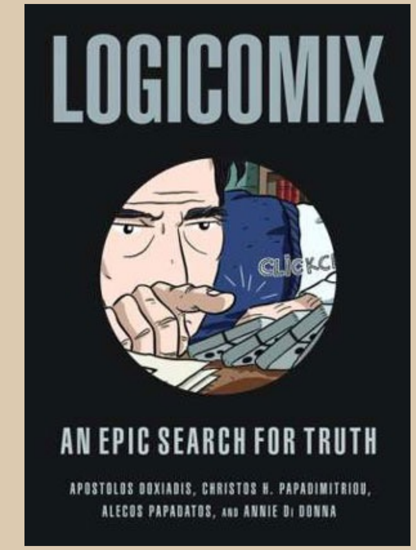
Limitations of first-order logic

FOL is not expressive enough to capture all meanings that can be expressed by basic natural language expressions:

Jumbo is a <u>small</u> elephant.	(Predicate modifiers)
Happy is a <u>state of mind.</u>	(Second-order predicates)
<u>Yesterday</u>, it rained.	(Non-logical sentence operators)
Bill and John have <u>the same</u> hair color.	(Higher-order quantification)

➔ What *logically sound* system can capture this diversity?

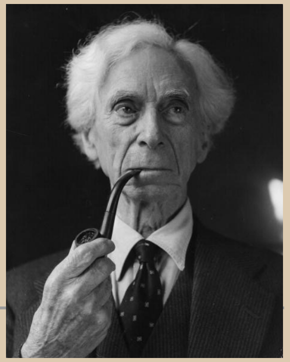
Introducing Russell's paradox



From: **Logicomix — An epic search for truth**; A. Doxiadis, C.H. Papadimitriou, A. Papadatos and A. Di Donna

Bertrand Russell

Russell's paradox for Higher-Order Logic



What if we extend the FOL interpretation of predicates, and simply interpret higher-order predicates as sets of sets of properties?

Then, for every predicate P, we can define a set {x | P(x)} containing all and only those entities for which P holds.

Now what if we define a set S = {X | X ∉ X} representing the set of all sets that are not members of itself..

Paradox: does S belong to itself?
If it does, then S must satisfy its constraints, namely that it doesn't belong to itself, which is not possible if we assume it belongs to S.
If not, then S is a set that doesn't belong to itself, hence it belongs to S.

➔ **Conclusion:** We need a more restricted way of talking about *properties* and *relations between properties*!

Type Theory

Winter-EFS Ch3
Page 50

In Type Theory, all logical expressions are assigned a *type* (that may be basic or complex), which restricts how they can be combined.

Basic types:

- **e** – the type of individual terms ("entities")
- **t** – the type of formulas ("truth-values")

Complex types:

- If $\boldsymbol{\sigma}$, $\boldsymbol{\tau}$ are types, then $\langle \boldsymbol{\sigma}, \boldsymbol{\tau} \rangle$ is a type

Note: this is a "tau" (the Greek letter), not a "t"!

➔ This represents a **functor** expression that takes an expression of type $\boldsymbol{\sigma}$ as its **argument** and returns an expression of type $\boldsymbol{\tau}$; this functor is sometimes written as $(\boldsymbol{\sigma} \rightarrow \boldsymbol{\tau})$ or simply $(\boldsymbol{\sigma\tau})$ (as in Winter-EFS)

Types & Function Application

Winter-EFS Ch3
Page 53

Types of first-order expressions:

- Individual constants (Luke, Death Star) : **e** ➔ **entity**
- One-place predicates (walk, jedi): ⟨**e, t**⟩ ➔ **function from entities to truth values (i.e., a property)**
- Two-place predicates (admire, fight with): ⟨**e**, ⟨**e, t**⟩⟩ ➔ **function from entities to properties**
- Three-place predicates (give, introduce): ⟨**e**, ⟨**e**, ⟨**e, t**⟩⟩⟩ ➔ **function from entities to entities to properties**

Function application: Combining a **functor** of complex type ⟨**α, β**⟩ with an appropriate **argument** of type **α**, results in an expression of type **β**: ⟨**α, β**⟩(**α**) ↦ **β**

- jedi'(luke') :: ⟨**e, t**⟩(**e**) ⟹ **t** ➔ **"luke is a jedi" has a truth value (true or false)**
- admire'(luke') :: ⟨**e**,⟨**e, t**⟩⟩(**e**) ⟹ ⟨**e, t**⟩ ➔ **"(to) admire luke" is a property**

More examples of types

Types of higher-order expressions:

- Predicate modifiers (expensive, small): ⟨⟨**e, t**⟩, ⟨**e, t**⟩⟩ ➔ **function from properties to properties**
- Second-order predicates (state of mind): ⟨⟨**e, t**⟩, **t**⟩ ➔ **property of properties**
- Sentence operators (yesterday, unfortunately): ⟨**t, t**⟩ ➔ **function from truth values to truth values**
- Degree particles (very, too): ⟨⟨⟨**e, t**⟩, ⟨**e, t**⟩⟩, ⟨⟨**e, t**⟩, ⟨**e, t**⟩⟩⟩ ➔ **complex function..** 😉

Tip: If **σ**, **τ** are basic types, ⟨**σ, τ**⟩ can be abbreviated as **στ**. Thus, the type of predicate modifiers and second-order predicates can be more conveniently written as ⟨**et, et**⟩ and ⟨**et, t**⟩, respectively.

Type Theory — Vocabulary

Non-logical constants:

- For every type **τ** a (possibly empty) set of non-logical constants CON_τ (pairwise disjoint)

Variables:

- For every type **τ** an infinite set of variables VAR_τ (pairwise disjoint)

Logical symbols: $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $=$

Brackets: (,)

Type Theory — Syntax

For every type τ, the set of well-formed expressions WE_τ is defined as follows:

(i) $CON_\tau \subseteq WE_\tau$ and $VAR_\tau \subseteq WE_\tau$;

(ii) If $\alpha \in WE_{\langle\sigma, \tau\rangle}$, and $\beta \in WE_\sigma$, then $\alpha(\beta) \in WE_\tau$; (function application)

(iii) If A, B are in WE_t, then $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$ are in WE_t;

(iv) If A is in WE_t and x is a variable of arbitrary type, then $\forall xA$ and $\exists xA$ are in WE_t;

(v) If α, β are well-formed expressions of the same type, then $\alpha = \beta \in WE_t$;

(vi) Nothing else is a well-formed expression.

NB. This prevents us from running into Russell's paradox!

Type inferencing

Winter-EFS Ch3
Page 59-60

Types can be derived for all expressions that constitute the logical form of a sentence, as defined by its syntactic structure.

"Luke is a talented jedi"

$$\dfrac{\text{luke}' :: e \qquad \dfrac{\text{talented}' :: \langle\langle e, t\rangle, \langle e, t\rangle\rangle \qquad \text{jedi}' :: \langle e, t\rangle}{\text{talented}'(\text{jedi}') :: \langle e, t\rangle}}{\text{talented}'(\text{jedi}')(\text{luke}') :: t}$$

Note: we here ignore the semantic contribution of "is" and "a" (see Winter, pg 61)

Type inferencing: examples

Recommended strategy: Start by describing the logical form of the sentences (how are expressions combined logically, based on the given syntactic bracketing), then derive types from there (see previous slide).

1. Yoda$_e$ [<u>is faster than</u> Palpatine$_e$].

2. Yoda$_e$ [is <u>much</u> [faster than]] Palpatine$_e$.

3. [[Han Solo]$_e$ fights] [<u>because</u> [[the Dark Side]$_e$ is rising]].

4. Obi-Wan$_e$ [[<u>told</u> [Qui-Gon Jinn]$_e$] he will take [the Jedi-exam]$_e$].

Higher-order predicates

Higher-order quantification:

- *Leia has the same <u>hair colour</u> as Padmé*

  $$\exists C\ (\text{hair\_colour}(C) \wedge C(l') \wedge C(p'))$$

  $\langle\langle e, t\rangle, t\rangle$ $\quad\quad$ $\langle e, t\rangle$ $\quad$ e

Higher-order equality:

- For $p, q \in CON_t$, “$p=q$” expresses material equivalence: “$p \leftrightarrow q$”.

- For $F, G \in CON_{\langle e, t\rangle}$, “$F=G$” expresses co-extensionality: “$\forall x(Fx \leftrightarrow Gx)$”

- For any formula ϕ of type *t*, $\phi=(x=x)$ is a representation of “ϕ is true”.

Type Theory — Semantics [1]

Winter-EFS Ch3
Page 51

Let **U** be a non-empty set of entities.

The domain of possible denotations $\mathbf{D}_\tau$ for every type $\boldsymbol{\tau}$ is given by:

- $D_e = U$
- $D_t = \{0,1\}$
- $D_{\langle\sigma,\tau\rangle}$ is the set of all functions from D_σ to D_τ

For any type $\boldsymbol{\tau}$*, expressions of type* $\boldsymbol{\tau}$ *denote elements of the domain* $\mathbf{D}_\tau$

Characteristic functions

Winter-EFS Ch3
Page 46-47

Many natural language expressions have a type ⟨**σ, t**⟩

Expressions with type ⟨**σ, t**⟩ are functions mapping elements of type **σ** to truth values: {**0,1**}

Such functions with a range of {**0,1**} are called *characteristic functions*, because they uniquely specify a subset of their domain $\mathbf{D_\sigma}$

The characteristic function of set M in a domain U is the function F_M: U→{0,1} such that for all a ∈ U, F_M(a) = 1 iff a ∈ M.

NB: For first-order predicates, the FOL representation (using sets) and the type-theoretic representation (using characteristic functions) are equivalent.

Interpretation with characteristic functions: example

For $M = \langle U, V \rangle$, let U consist of five entities. For selected types, we have the following sets of possible denotations:

- $D_t = \{0,1\}$
- $D_e = U = \{e_1, e_2, e_3, e_4, e_5\}$
- $D_{<e,t>} = \left\{ \begin{bmatrix} e_1 \rightarrow 1 \\ e_2 \rightarrow 0 \\ e_3 \rightarrow 1 \\ e_4 \rightarrow 0 \\ e_5 \rightarrow 1 \end{bmatrix}, \begin{bmatrix} e_1 \rightarrow 1 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 0 \\ e_4 \rightarrow 1 \\ e_5 \rightarrow 1 \end{bmatrix}, \begin{bmatrix} e_1 \rightarrow 0 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 1 \\ e_4 \rightarrow 0 \\ e_5 \rightarrow 0 \end{bmatrix}, \ldots \right\}$

Alternative set notation: $D_{<e,t>} = \{\{e_1,e_3,e_5\},\{e_1,e_2,e_4,e_5\},\{e_2,e_3\},\ldots\}$

Type Theory — Semantics [2]

A model structure for a type theoretic language is a tuple $\mathbf{M} = \langle \mathbf{U}, \mathbf{V} \rangle$ such that:

- **U** is a non-empty domain of individuals
- **V** is an interpretation function, which assigns to every $\boldsymbol{\alpha} \in \mathbf{CON}_{\boldsymbol{\tau}}$ an element of $\mathbf{D}_{\boldsymbol{\tau}}$ (where $\boldsymbol{\tau}$ is an arbitrary type)

The variable assignment function g assigns to every typed variable $\mathbf{v} \in \mathbf{VAR}_{\boldsymbol{\tau}}$ an element of $\mathbf{D}_{\boldsymbol{\tau}}$

Type Theory — Interpretation

Given a model structure $M = \langle U, V \rangle$ and a variable assignment g:

- $[\![\alpha]\!]^{M,g}$ $= V(\alpha)$ if α is a constant
  $= g(\alpha)$ if α is a variable
- $[\![\alpha(\beta)]\!]^{M,g}$ $= [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$
- $[\![\alpha = \beta]\!]^{M,g}$ $= 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$
- $[\![\neg\phi]\!]^{M,g}$ $= 1$ iff $[\![\phi]\!]^{M,g} = 0$
- $[\![\phi \wedge \psi]\!]^{M,g}$ $= 1$ iff $[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$
- $[\![\phi \vee \psi]\!]^{M,g}$ $= 1$ iff $[\![\phi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$

  …

For any variable v of type σ:

- $[\![\exists v\phi]\!]^{M,g}$ $= 1$ iff there is a $d \in D_\sigma$ such that $[\![\phi]\!]^{M,g[v/d]} = 1$
- $[\![\forall v\phi]\!]^{M,g}$ $= 1$ iff for all $d \in D_\sigma$: $[\![\phi]\!]^{M,g[v/d]} = 1$

Interpretation: Example

Luke is a talented jedi

$$\dfrac{\dfrac{}{\text{luke'} :: e} \qquad \dfrac{\text{jedi'} :: \langle e, t\rangle \qquad \text{talented'} :: \langle\langle e, t\rangle, \langle e, t\rangle\rangle}{\text{talented'(jedi')} :: \langle e, t\rangle}}{\text{talented'(jedi')(luke')} :: t}$$

$[\![\text{talented'(jedi')(luke')}]\!]^{M,g}$

$= [\![\text{talented'(jedi')}]\!]^{M,g} ([\![\text{luke'}]\!]^{M,g})$

$= [\![\text{talented'}]\!]^{M,g}([\![\text{jedi'}]\!]^{M,g}) ([\![\text{luke'}]\!]^{M,g})$

$= V_M(\text{talented'})(V_M(\text{jedi'}))(V_M(\text{luke'}))$

Interpretation: Example (cont.)

$[\![\textit{Luke is a talented jedi}]\!]^{M,g} = V_M(\text{talented'})(V_M(\text{jedi'}))(V_M(\text{luke'}))$

$V_M(\text{luke'}) = e_1 \ (\in \mathbf{D_e})$

$$V_M(\text{jedi'}) = \begin{bmatrix} e_1 \to 1 \\ e_2 \to 0 \\ e_3 \to 1 \\ e_4 \to 0 \\ e_5 \to 1 \end{bmatrix} (\in \mathbf{D}_{\langle e,t \rangle}) \quad \Leftrightarrow \quad \{e_1, e_3, e_5\}$$

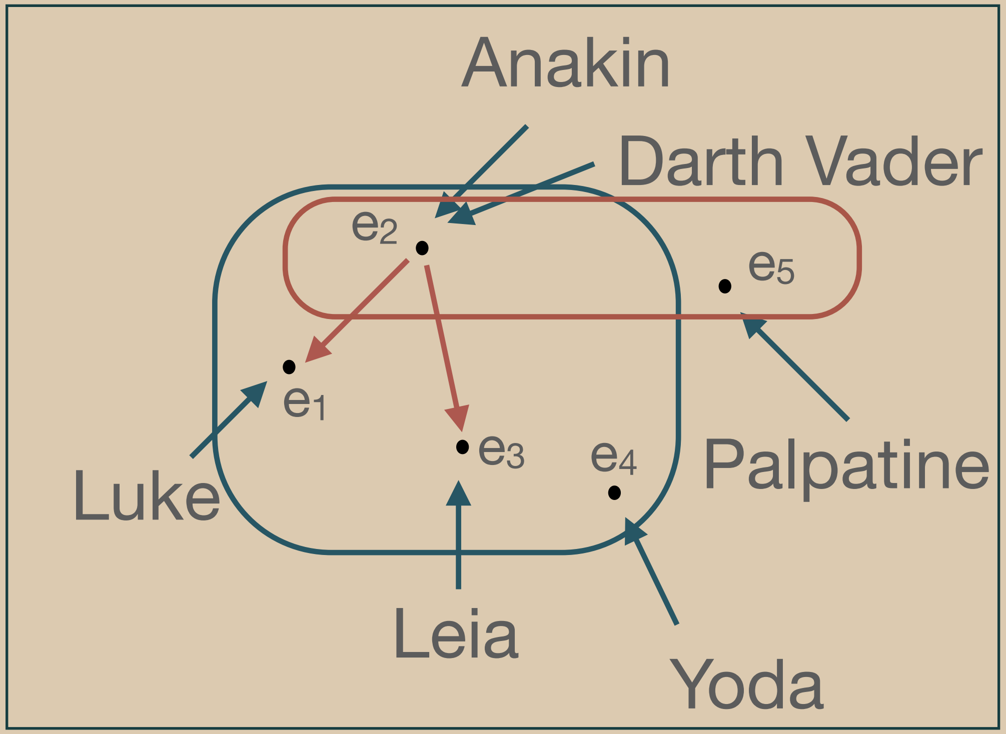
$$V_M(\text{talented'}) = \begin{bmatrix} \begin{bmatrix} e_1 \to 1 \\ e_2 \to 0 \\ e_3 \to 1 \\ e_4 \to 0 \\ e_5 \to 1 \end{bmatrix} \to \begin{bmatrix} e_1 \to 1 \\ e_2 \to 0 \\ e_3 \to 0 \\ e_4 \to 0 \\ e_5 \to 1 \end{bmatrix} \\ \begin{bmatrix} e_1 \to 0 \\ e_2 \to 0 \\ e_3 \to 1 \\ e_4 \to 1 \\ e_5 \to 1 \end{bmatrix} \to \begin{bmatrix} e_1 \to 0 \\ e_2 \to 0 \\ e_3 \to 0 \\ e_4 \to 0 \\ e_5 \to 1 \end{bmatrix} \\ \ldots \end{bmatrix} (\in \mathbf{D}_{\langle \langle e,t \rangle \langle e,t \rangle \rangle}) \quad \Leftrightarrow \quad \begin{bmatrix} \{e_1,e_3,e_5\} \to \{e_1,e_5\} \\ \{e_3,e_4,e_5\} \to \{e_5\} \\ \ldots \end{bmatrix}$$

$V_M(\text{talented'})(V_M(\text{jedi'}))$

$V_M(\text{talented'})(V_M(\text{jedi'}))(V_M(\text{luke'}))$

Defining the right model

Consider the following Model M:

M: 

$D_e = U_M = \{e_1, e_2, e_3, e_4, e_5\}$

$V_M(\text{anakin'}_e) = V_M(\text{darth\_vader'}_e) = e_2$

$V_M(\text{jedi'}_{\langle e,t\rangle}) = \begin{bmatrix} e_1 \rightarrow 1 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 1 \\ e_4 \rightarrow 1 \\ e_5 \rightarrow 0 \end{bmatrix}$ $V_M(\text{dark\_sider'}_{\langle e,t\rangle}) = \begin{bmatrix} e_1 \rightarrow 0 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 0 \\ e_4 \rightarrow 0 \\ e_5 \rightarrow 1 \end{bmatrix}$

$$V_M(\text{powerful'}_{\langle\langle e,t\rangle\langle e,t\rangle\rangle}) = \begin{bmatrix} \begin{bmatrix} e_1 \rightarrow 1 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 1 \\ e_4 \rightarrow 1 \\ e_5 \rightarrow 0 \end{bmatrix} \rightarrow \begin{bmatrix} e_1 \rightarrow 0 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 0 \\ e_4 \rightarrow 1 \\ e_5 \rightarrow 0 \end{bmatrix} \\ \begin{bmatrix} e_1 \rightarrow 0 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 0 \\ e_4 \rightarrow 0 \\ e_5 \rightarrow 1 \end{bmatrix} \rightarrow \begin{bmatrix} e_1 \rightarrow 0 \\ e_2 \rightarrow 1 \\ e_3 \rightarrow 0 \\ e_4 \rightarrow 0 \\ e_5 \rightarrow 1 \end{bmatrix} \\ \dots \end{bmatrix}$$

Note that here “powerful” is truth-preserving:
Powerful $X_{\langle e,t\rangle} \vDash X_{\langle e,t\rangle}$

Adjective classes & Meaning postulates

Some valid inferences in natural language:

- Bill is a poor piano player ⊨ Bill is a piano player
- Bill is a blond piano player ⊨ Bill is blond
- Bill is a former professor ⊨ Bill isn't a professor

➔ These entailments do not hold in type theory by definition. Why?

Meaning postulates: restrictions on models which constrain the possible meaning of certain words

Adjective classes & Meaning postulates (cont.)

Restrictive or Subsective adjectives (“poor”)

- $[\![\text{poor N}]\!] \subseteq [\![\text{N}]\!]$
- Meaning postulate: $\forall G \forall x(\text{poor}(G)(x) \rightarrow G(x))$

Intersective adjectives (“blond”)

- $[\![\text{blond N}]\!] = [\![\text{blond}]\!] \cap [\![\text{N}]\!]$
- Meaning postlate: $\forall G \forall x(\text{blond}(G)(x) \rightarrow (\text{blond*}(x) \wedge G(x))$
- NB: $\text{blond} \in WE\langle\langle e, t\rangle, \langle e, t\rangle\rangle \neq \text{blond*} \in WE\langle e, t\rangle$

Privative adjectives (“former”)

- $[\![\text{former N}]\!] \cap [\![\text{N}]\!] = \varnothing$
- Meaning postlate: $\forall G \forall x(\text{former}(G)(x) \rightarrow \neg G(x))$

Reading material

- Winter: Elements of Formal Semantics (Chapter 3, Part I & II)
  http://www.phil.uu.nl/~yoad/efs/main.html