A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths

FAIROUZ KAMAREDDINE, *Department of Computing Science, University of Glasgow, 17 Lilybank Gardens, Glasgow G12 8QQ, Scotland. E-mail: fairouz@dcs.gla.ac.uk*

TWAN LAAN, *Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. E-mail: laan@win.tue.nl*

Abstract

Both in Kripke's Theory of Truth KTT [8] and Russell's Ramified Type Theory RTT [16, 9] we are confronted with some hierarchy. In RTT, we have a double hierarchy of orders and types. That is, the class of propositions is divided into different orders where a propositional function can only depend on objects of lower orders and types. Kripke on the other hand, has a ladder of languages where the truth of a proposition in language L_n can only be made in L_m where $m\rangle n$. Kripke finds a fixed point for his hierarchy (something Russell does not attempt to do). We investigate in this paper the similarities of both hierarchies: At level n of KTT the truth or falsehood of all order-n-propositions of RTT can be established. Moreover, there are order-n-propositions that get a truth value at an earlier stage in KTT. Furthermore, we show that RTT is more restrictive than KTT, as some type restrictions are not needed in KTT and more formulas can be expressed in the latter.

Looking back at the double hierarchy of Russell, Ramsey [11], and Hilbert and Ackermann [7] considered the orders to cause the restrictiveness, and therefore removed them. This removal resulted in Church's Simple Type Theory STT [1]. We show however that orders in RTT correspond to levels of truth in KTT. Hence, KTT can be regarded as the dual of STT where types have been removed and orders are maintained. As RTT is more restrictive than KTT, we can conclude that it is the combination of types and orders that was the restrictive factor in RTT.

Keywords: The Hierarchies of Types, orders and truth levels, Principia's Substitution, the restrictiveness of combining types and orders

1 Introduction

The role of Type Theory in Logic and Mathematics has always been a restrictive one. The need for restrictions was realised at the beginning of this century, when Bertrand Russell showed that Frege's *Begriffsschrift* [5], a formalisation of logic, was inconsistent[1]. Russell considered self-application to be the cause of the contradictions, and hence excluded all possibilities of self-application in his Theory of Types [13, 16].

As paradoxical sentences in Natural Language play a role similar to that of the paradoxes in Logic and Mathematics, Type Theory eliminated the paradoxical sentences (see for instance [10]). Paradoxes moreover have been classified in two categories (see

[1]An English translation of Russell's letter to Frege in which this inconsistency is described can be found in [6]

[11]): the logical and the semantical. The famous Russell's paradox is logical whereas the famous liar's paradox is semantical. The semantical paradoxes usually involve the truth predicate `T` which gives the truth value of a proposition. Tarski [14] shows that truth is undefinable and that having the truth predicate inside the language leads to contradictions. For this reason, he distinguishes between (object-) language and meta-language and allows the truth predicate only at the meta-level. Now, to talk about the truth of sentences in the meta-language, one needs a meta-meta-language and so on. Kripke [8], however, considers Russell's Theory of Types and the Theory of Truth by Tarski to be too restrictive for a proper formalisation of Natural Language and presents a type-free theory where the truth predicate belongs to the language, in which nevertheless the known paradoxes do not occur. Kripke's idea is to follow a certain hierarchy as with Russell but to take the fixed point of his hierarchy of languages to reach a language which has its own truth predicate.

We start this paper by presenting an overview of both Russell's system (in Section 2, using a formalisation presented in [9]) and Kripke's (in Section 3). In Section 4 we carefully compare both theories. As Russell's system is said to be more restrictive than Kripke's, this comparison is carried out by coding Russell's expressions in Kripke's theory. The stronger restrictions in the Ramified Type Theory can be seen clearly: at several parts in the definition of the embedding the reader will notice that some type-theoretic properties of Russell's expressions are mentioned, but not used in this definition. We show that the embedding is conservative, i.e. that truth in Russell's theory and in Kripke's theory are the same, as far as formulas expressible in Russell's (more restrictive) system are concerned.

2 The Ramified Theory of Types RTT

In this section we give a short, formal description of Russell's Ramified Theory of Types (RTT). Our formalisation of Russell's theory is the first of its kind and is worth attention. This formalisation is both faithful to Russell's original informal presentation and compatible with the present formulations of type theories. The basic aim of RTT is to exclude the logical paradoxes from logic by eliminating all self-references. An extended philosophical motivation for this theory can be found in *Principia Mathematica* [16], pages 38–55. We will not go into the full details of the formalisation of Russell's theory (these details can be found in [9], the presentation by Russell himself in *Principia* is informal).

In Subsection 2.1 we introduce propositional functions, the logical formulas of the 'naive' system of logic. In Subsection 2.2 we present a rule to assign a type to some of these propositional functions. The propositional functions that lead to the logical paradoxes are, of course, not typable. In Subsection 2.3 substitution for RTT is discussed. This part is rather technical, but we need it in the proof of lemma 4.8, which is essential in the proof of one of our fundamental results (theorem 4.10). That is, lemma 4.8 helps us in showing that KTT can be regarded as a system based on RTT of which the types and not the orders have been removed.

2.1 Propositional functions

In this section we shall describe the set of propositions and propositional functions which Whitehead and Russell use in *Principia*. We give a modernised, formal definition which corresponds to the description in *Principia*.

At the basis of the system of our formalization there is

- an infinite set $\mathcal{A}$ of *individual-symbols*;
- an infinite set $\mathcal{V}$ of *variables*;
- an infinite set $\mathcal{R}$ of *relation-symbols* together with a map $\mathbf{a} : \mathcal{R} \to \mathrm{I\!N}^+$ (indicating the *arity* of each relation-symbol).

0-ary relations are not explicitly used in *Principia* but could be added without problems. Since functions are relations in *Principia*, we will not introduce a special set of function symbols.

We assume that $\{\mathtt{a}_1, \mathtt{a}_2, \ldots\} \subseteq \mathcal{A}$; $\{\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{y}, \mathtt{y}_1, \ldots, \mathtt{z}, \mathtt{z}_1, \ldots\} \subseteq \mathcal{V}$; $\{\mathtt{R}, \mathtt{R}_1, \ldots, \mathtt{S}, \mathtt{S}_1, \ldots\} \subseteq \mathcal{R}$. We will use the letters $x, y, z, x_1, \ldots$ as meta-variables over $\mathcal{V}$, and $R, R_1, \ldots$ as meta-variables over $\mathcal{R}$. Note that variables are written in `typewriter` style and that meta-variables are written in *italics*: $\mathtt{x}$ denotes one, *fixed* object in $\mathcal{V}$ whilst x denotes an *arbitrary* object of $\mathcal{V}$.

We assume that there is an *order* (e.g. alphabetical) on the collection $\mathcal{V}$, and write $x \langle y$ if the variable x is ordered before the variable y. In particular, we assume that

$$\mathtt{x} \langle \mathtt{x}_1 \langle \ldots \langle \mathtt{y} \langle \mathtt{y}_1 \langle \ldots \langle \mathtt{z} \langle \mathtt{z}_1 \langle \ldots$$

We also have the logical symbols $\vee$, $\neg$ and $\forall$ in our alphabet, and the non-logical symbols: parentheses and the comma.

Definition 2.1 (Propositional functions)
We define a collection $\mathcal{F}$ of *propositional functions*, and for each element f of $\mathcal{F}$ we simultaneously define the collection $\mathrm{FV}(f)$ of *free variables* of f:

1. If $R \in \mathcal{R}$ and $i_1, \ldots, i_{\mathbf{a}(R)} \in \mathcal{A} \cup \mathcal{V}$ then $R(i_1, \ldots, i_{\mathbf{a}(R)}) \in \mathcal{F}$.
   $\mathrm{FV}(R(i_1, \ldots, i_{\mathbf{a}(R)})) \stackrel{\text{def}}{=} \{i_1, \ldots, i_{\mathbf{a}(R)}\} \cap \mathcal{V}$;
2. If $z \in \mathcal{V}$, $n \in \mathrm{I\!N}$ and $k_1, \ldots, k_n \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{F}$, then $z(k_1, \ldots, k_n) \in \mathcal{F}$.
   $\mathrm{FV}(z(k_1, \ldots, k_n)) \stackrel{\text{def}}{=} \{z, k_1, \ldots, k_n\} \cap \mathcal{V}$.
   If $n = 0$, we write $z()$ so as to distinguish the propositional function $z()$ from the variable z;[2]
3. If $f, g \in \mathcal{F}$ then $f \vee g \in \mathcal{F}$ and $\neg f \in \mathcal{F}$. $\mathrm{FV}(f \vee g) \stackrel{\text{def}}{=} \mathrm{FV}(f) \cup \mathrm{FV}(g)$; $\mathrm{FV}(\neg f) \stackrel{\text{def}}{=} \mathrm{FV}(f)$;
4. If $f \in \mathcal{F}$ and $x \in \mathrm{FV}(f)$ then $\forall x[f] \in \mathcal{F}$. $\mathrm{FV}(\forall x[f]) = \mathrm{FV}(f) \setminus \{x\}$.
5. All propositional functions can be constructed by using the rules 1, 2, 3 and 4 above.

We use the letters f, g, h as meta-variables over $\mathcal{F}$.

[2]It is important to note that a variable is not a propositional function. See for instance [12], Chapter VIII: 'The variable', p. 94 of the 7th impression.

Convention 2.2
[Variable convention] We make the usual convention that a variable x in a propositional function f that is bound by the quantifier $\forall$ does not occur as a free variable in f. Moreover, different bound variables in f have different names.

A propositional function f must be seen as a proposition in which some parts (the free variables) have been left undetermined. It will turn into a proposition as soon as we assign values to all the free variables occurring in it. In this light, a *proposition* can be seen as a degenerated propositional function (with 0 free variables).

It will be clear now what the intuition behind propositional function of the form $R(i_1, \ldots, i_{\mathbf{a}(R)})$, $f \vee g$, $\neg f$ and $\forall x[f]$ is. The intuition behind propositional functions of the second kind is not so obvious. $z(k_1, \ldots, k_n)$ is a propositional function of *higher order: z is a variable for a propositional function with n free variables; the argument list $k_1, \ldots, k_n$ indicates what should be substituted*[3] *for these free variables as soon as one assigns such a propositional function to z.*

Notice that there are propositional functions of the form $z(k_1, \ldots, k_n)$ (where $z \in \mathcal{V}$) but that expressions of the form $f(k_1, \ldots, k_n)$, where $f \in \mathcal{F}$, are not propositional functions. Even substituting f for z in $z(k_1, \ldots, k_n)$ does not lead to $f(k_1, \ldots, k_n)$, as the notion of substitution in RTT will appear to be quite different from the usual notion of substitution in first order logic (see Subsection 2.3 for more details).

Example 2.3
Here are some higher-order propositional functions from ordinary mathematics.

- The propositional functions $\mathtt{z(x)}$ and $\mathtt{z(y)}$ in the definition of Leibniz-equality:

$$\forall \mathtt{z}[\mathtt{z(x)} \leftrightarrow \mathtt{z(y)}]$$

- The propositional functions $\mathtt{z(0)}$, $\mathtt{z(x)}$ and $\mathtt{z(y)}$ in the formulation of complete induction:

$$[\mathtt{z(0)} \rightarrow (\forall \mathtt{x} \forall \mathtt{y}[\mathtt{z(x)} \rightarrow (\mathtt{S(x,y)} \rightarrow \mathtt{z(y)})])] \rightarrow \forall \mathtt{x}[\mathtt{z(x)}]$$

- $\mathtt{z()}$ in the formulation of the law of the excluded middle:

$$\forall \mathtt{z}[\mathtt{z()} \vee \neg \mathtt{z()}]$$

2.2 *Ramified types*

Not all propositional functions should be allowed in our language. For instance, the expression $\neg x(x)$ is a perfectly legal element of $\mathcal{F}$, nevertheless, it is the propositional function that makes it possible to derive the Russell Paradox. Therefore, types are introduced.

Definition 2.4 (Ramified types)
1. ι^0 is a ramified type ($\mathtt{0}$ is called the *order* of this type);

[3] In the Principia, it is not made clear *how* we should carry out such substitutions. We must depend on our intuition and on the way in which substitution is *used* in the Principia. Nevertheless, it is hard and elaborate to give a proper definition of substitution. We present a short overview of this definition in Subsection 2.3; for a motivation of this definition and its relation to β-reduction in the λ-calculus the reader should consult [9].

2. If $t_1, \ldots, t_n$ are ramified types of orders $a_1, \ldots, a_n$ respectively, and $a \rangle \max(a_1, \ldots, a_n)$, then $(t_1, \ldots, t_n)^a$ is a ramified type of order a;
3. All ramified types can be constructed using the rules 1 and 2.

ι^0 represents the type of the individuals, and one can think of $(t_1, \ldots, t_n)^a$ as being the type of the propositional functions with n free variables, say $x_1, \ldots, x_n$, such that if we assign values k_1 of type t_1 to x_1, $\ldots$, k_n of type t_n to x_n, then we obtain a proposition. The type $()^a$ stands for the type of propositions of order a.

Russell strictly divides his propositional functions in orders. For instance, both $\forall \mathtt{p}[\mathtt{p}() \vee \neg \mathtt{p}()]$ and $\mathtt{R}(\mathtt{a})$ are propositions, but they are of different level: The earlier one presumes a full collection of propositions, hence (according to Russell) it cannot belong to the same collection of propositions as the propositions $\mathtt{p}$ over which it quantifies (among which $\mathtt{R}(\mathtt{a})$). This made Russell decide to let $\forall \mathtt{p}[\mathtt{p}() \vee \neg \mathtt{p}()]$ belong to a type of a *higher* order (level) than the order of $\mathtt{R}(\mathtt{a})$.

This can already be seen in the definition of ramified types: $(t_1, \ldots, t_n)^a$ can only be a type if a is strictly greater than each of the orders of the t_is.

Definition 2.5
Let $x_1, \ldots, x_n$ be a list of distinct variables, and $t_1, \ldots, t_n$ be a list of ramified types. We call $x_1{:}t_1, \ldots, x_n{:}t_n$ a *context* and call $\{x_1, \ldots, x_n\}$ its *domain.*

We write $\Gamma \vdash f : t$ to express that $f \in \mathcal{F}$ has type t in context Γ, and extend the variable convention to contexts: If x is bound in f, then x does not occur in the domain of Γ.

We use Γ, Δ to range over contexts and $t_1, t_2, \ldots$ to range over types.

We now present a set of typing rules for RTT. These rules are derived from and equivalent to the rules in [9], which are as close as possible to Russell's original ideas. We change our notation for propositional functions slightly: Instead of $\forall x[f]$ we write $\forall x{:}t[f]$, where t is some ramified type.

Definition 2.6 (Typing rules for RTT)

- If $c \in \mathcal{A}$, then $\Gamma \vdash c : \iota^0$ for any context Γ;
- If $f \in \mathcal{F}$, and $x_1 \langle \ldots \langle x_n$ are the free variables of f, and $t_1, \ldots, t_n$ are types such that $x_i{:}t_i \in \Gamma$, then $\Gamma \vdash f : (t_1, \ldots, t_n)^a$ if and only if
  - If $f \equiv R(i_1, \ldots, i_{\mathsf{a}(R)})$ then $t_i = \iota^0$ for all i, and $a = 1$ (if $n \rangle 0$) or $a = 0$ (if $n = 0$);
  - If $f \equiv z(k_1, \ldots, k_m)$ then there are $u_1, \ldots, u_m$ such that $z{:}(u_1, \ldots, u_m)^{a-1} \in \Gamma$, and $\Gamma \vdash k_i{:}u_i$ for all $k_i \in \mathcal{A} \cup \mathcal{F}$, and $k_i{:}u_i \in \Gamma$ for all $k_i \in \mathcal{V}$;
  - If $f \equiv f_1 \vee f_2$ then there are $u_1^{a_1}, u_2^{a_2}$ such that $\Gamma \vdash f_i : u_i^{a_i}$ and $a = \max(a_1, a_2)$; if $f \equiv \neg f'$ then $\Gamma \vdash f' : (t_1, \ldots, t_n)^a$.
  - If $f \equiv \forall x{:}t_0[f']$ then there is j such that $\Gamma, x{:}t_0 \vdash f' : (t_1, \ldots, t_{j-1}, t_0, t_j, \ldots, t_n)^a$.

Example 2.7
$\neg \mathtt{x}(\mathtt{x})$ is not typable in any context Γ.

Assume, we would have $\Gamma \vdash \neg \mathtt{x}(\mathtt{x}) : t$.

Then t must be of the form $(u)^a$, with $\mathtt{x}{:}u \in \Gamma$, as $\neg \mathtt{x}(\mathtt{x})$ has one free variable.

This implies $\Gamma \vdash \mathtt{x}(\mathtt{x}) : (u)^a$, hence by Unicity of Types below, $u \equiv (u')^{a-1}$, with $\mathtt{x} : u' \in \Gamma$.

As Γ is a context, we have $u \equiv u'$, hence $u \equiv (u)^{a-1}$, which is impossible.

An important result is the following (a proof can be found in [9]):

Theorem 2.8 (Unicity of types)
If $\Gamma \vdash f : t$ and $\Gamma \vdash f : u$ then $t \equiv u$.

2.3 Substitution in RTT

Substitution in RTT is not simply a syntactic operation of replacing a variable by an object, as is usual in first-order logic. This can be understood if we read the interpretation of the propositional function $z(k_1, \ldots, k_m)$. Substituting a propositional function f for the variable z should have as a result f, in which $k_1, \ldots, k_m$ are substituted for the free variables in f. So a substitution may result in a new substitution (and we may wonder whether this process will ever terminate). Below, we give a formal definition of substitution in RTT (needed in the proof of the Substitution lemma 4.8). For examples and an extended motivation of the definition the reader may consult [9].

Definition 2.9
Let $f \in \mathcal{F}$, $\Gamma \vdash f : t$, $k_1, \ldots, k_m \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{F}$ and $x_1, \ldots, x_n \in \mathcal{V}$ such that

- If $k_i \in \mathcal{A}$ then $x_i{:}\iota^0 \in \Gamma$;
- If $k_i \in \mathcal{V}$ then there is t such that both $k_i{:}t \in \Gamma$ and $x_i{:}t \in \Gamma$;
- If $k_i \in \mathcal{F}$ then there is t such that $\Gamma \vdash k_i{:}t$ and $x_i{:}t \in \Gamma$.

We define $f[x_1, \ldots, x_m{:=}k_1, \ldots, k_m]$, the (simultaneous) substitution of $k_1, \ldots, k_m$ for $x_1, \ldots, x_m$ in f (shorthand $f[x_i{:=}k_i]$ if no confusion arises) by a double induction on the order and structure of f:

- $f \equiv R(i_1, \ldots, i_{\mathbf{a}(R)})$. Define $i'_j \stackrel{\text{def}}{=} \begin{cases} k_\ell & \text{if } i_j \equiv x_\ell \\ i_j & \text{if } i_j \notin \{x_1, \ldots, x_m\} \end{cases}$

  $f[x_i{:=}k_i] \stackrel{\text{def}}{=} R(i'_1, \ldots, i'_{\mathbf{a}(R)})$.
- $f \equiv z(h_1, \ldots, h_n)$. We distinguish two cases:
  1. $z \notin \{x_1, \ldots, x_m\}$. Define $h'_j \stackrel{\text{def}}{=} \begin{cases} k_\ell & \text{if } h_j \equiv x_\ell \\ h_j & \text{if } h_j \notin \{x_1, \ldots, x_m\} \end{cases}$

     $f[x_i{:=}k_i] \stackrel{\text{def}}{=} z(h'_1, \ldots, h'_n)$.
  2. $z \in \{x_1, \ldots, x_m\}$, assume $z \equiv x_p$. Define $h'_j \stackrel{\text{def}}{=} \begin{cases} k_\ell & \text{if } h_j \equiv x_\ell \\ h_j & \text{if } h_j \notin \{x_1, \ldots, x_m\} \end{cases}$

     Notice that, as z, x_p and k_p have the same type, k_p is a propositional function with n free variables, say $y_1 \langle \ldots \langle y_n$. Now: $f[x_i{:=}k_i] \stackrel{\text{def}}{=} k_p[y_1, \ldots, y_n{:=}h'_1, \ldots, h'_n]$. Note that the object on the right is a correct substitution (with respect to the types of the y_j and the h'_j) and has already been defined, as k_p has the same order as z, which is exactly one less than the order of $z(h_1, \ldots, h_n)$.
- $f \equiv f_1 \vee f_2$. Then $f[x_i{:=}k_i] \stackrel{\text{def}}{=} f_1[x_i{:=}k_i] \vee f_2[x_i{:=}k_i]$.
- $f \equiv \neg f'$. Then $f[x_i{:=}k_i] \stackrel{\text{def}}{=} \neg f'[x_i{:=}k_i]$.
- $f \equiv \forall x{:}t[f']$. Then $f[x_i{:=}k_i] \stackrel{\text{def}}{=} \forall x{:}t[f'[x_i{:=}k_i]]$ (we assume that $x \notin \{x_1, \ldots, x_m\}$).

Substitution in RTT is quite different from usual notions of substitution in, for example, first order logic or λ-calculus. For a good understanding of the rest of this article it is essential to see these differences.

There is no definition of substitution in *Principia*. The above definition is based on what happens in *Principia* when a substitution seems to take place. The hardest part of the definition is a substitution of the form $z(h_1,\ldots,h_n)[x_1,\ldots,x_m{:=}k_1,\ldots,k_m]$ where z is among the x_i: say, $z \equiv x_p$. We can assume that k_p is a propositional function with n free variables, say, $y_1 \langle \ldots \langle y_n$.

According to the definition, we first carry out the substitutions that have nothing to do with z (the definition of the h'_js). This part is similar to a usual first-order substitution.

Now we must substitute k_p for z in $z(h'_1,\ldots,h'_n)$. The intuition on the propositional function $z(h'_1,\ldots,h'_n)$, that was explained at the end of Subsection 2.1, prescribes that the arguments $h'_1,\ldots,h'_n$ must be substituted for the free variables $y_1,\ldots,y_n$ of k_p, as soon as k_p is substituted for z. This leads to a *new* substitution $k_p[y_1,\ldots,y_n{:=}h'_1,\ldots,h'_n]$. As the order of k_p is lower than the order of $z(h_1,\ldots,h_p)$, we may assume that the final result of this new substitution has already been defined.

To understand the notion better it may be helpful to treat the substitution $z(h_1,\ldots,h_n)[x_i{:=}k_i]$ first as if it was a usual, first order substitution, and write down $k_p(h'_1,\ldots,h'_n)$ as an informal, intermediate result. Then the substitution of the h'_j for the y_j in k_p can be seen as the contraction of the n β-redexes in the λ-term $(\lambda y_1 \cdots y_n.k_p)h'_1 \cdots h'_n$. Notice, however, that $k_p(h'_1,\ldots,h'_n)$ is *not* a propositional function (see the explanation in Subsection 2.1). More on the relation between substitution in RTT and β-reduction in λ-calculus can be found in [9].

We give some examples of RTT-style substitutions in order to make the reader more familiar with this notion.

Example 2.10

- $\mathtt{R}(\mathtt{x}_1,\mathtt{x}_2)[\mathtt{x}_1{:=}\mathtt{a}_1] = \mathtt{R}(\mathtt{a}_1,\mathtt{x}_2)$. On first order level, RTT-substitution is the same as in first order logic.
- $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})[\mathtt{x}{:=}\mathtt{a}] = \mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})$. Note that $\mathtt{x}$ is not a free variable of $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})$! The substitution does not 'continue' in the arguments of $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})$: $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})[\mathtt{x}{:=}\mathtt{a}] \neq \mathtt{z}(\mathtt{R}(\mathtt{x})[\mathtt{x}{:=}\mathtt{a}],\mathtt{y})$.
- $\mathtt{z}(\mathtt{a})[\mathtt{z}{:=}\mathtt{R}(\mathtt{x})] = \mathtt{R}(\mathtt{x})[\mathtt{x}{:=}\mathtt{a}] = \mathtt{R}(\mathtt{a})$.
- $\mathtt{z}_1(\mathtt{R}(\mathtt{x}))[\mathtt{z}_1{:=}\mathtt{z}_2(\mathtt{a})] = \mathtt{z}_2(\mathtt{a})[\mathtt{z}_2{:=}\mathtt{R}(\mathtt{x})] = \mathtt{R}(\mathtt{x})[\mathtt{x}{:=}\mathtt{a}] = \mathtt{R}(\mathtt{a})$.
- $\mathtt{z}_1(\mathtt{x}_2,\mathtt{R}(\mathtt{x}_1))[\mathtt{x}_2,\mathtt{z}_1{:=}\mathtt{a},\mathtt{z}_2(\mathtt{y})] = \mathtt{z}_2(\mathtt{y})[\mathtt{y},\mathtt{z}_2{:=}\mathtt{a},\mathtt{R}(\mathtt{x}_1)] = \mathtt{R}(\mathtt{x}_1)[\mathtt{x}_1{:=}\mathtt{a}] = \mathtt{R}(\mathtt{a})$. The reader might want to make some informal, intermediate steps in this substitution (as explained above): $\mathtt{z}_1(\mathtt{x}_2,\mathtt{R}(\mathtt{x}_1))[\mathtt{x}_2,\mathtt{z}_1{:=}\mathtt{a},\mathtt{z}_2(\mathtt{y})]$ first leads to $(\mathtt{z}_2(\mathtt{y}))(\mathtt{a},\mathtt{R}(\mathtt{x}_1))$ as an intermediate result, and then to $\mathtt{z}_2(\mathtt{y})[\mathtt{y},\mathtt{z}_2{:=}\mathtt{a},\mathtt{R}(\mathtt{x}_1)]$.

  Similarly, this new substitution first leads to $(\mathtt{R}(\mathtt{x}_1))(\mathtt{a})[\mathtt{z}_2{:=}\mathtt{R}(\mathtt{x}_1)]$ and then to $\mathtt{R}(\mathtt{x}_1)[\mathtt{x}_1{:=}\mathtt{a}]$.

We will need the following results about substitutions. They are proved in [9].

Lemma 2.11
The order of f is greater than or equal to the order of the substitution $f[x_i{:=}k_i]$.

Lemma 2.12
$\textsc{fv}(f[x_i{:=}g_i]) = (\textsc{fv}(f) \setminus \{x_1,\ldots,x_n\}) \cup \{g_i | g_i \in \mathcal{V} \text{ and } x_i \in \textsc{fv}(f)\}$.

2.4 Logical truth for RTT *in Tarski's style*

With substitution properly defined, we can give a definition of logical truth in Tarski-style for RTT:

DEFINITION 2.13 (Logical truth for RTT)
Let $f \in \mathcal{F}$ and assume $\text{FV}(f) = \varnothing$. We define $\text{RTT} \models f$:

- If $(a_1, \ldots, a_m) \in R$ then $\text{RTT} \models R(a_1, \ldots, a_m)$, for all individuals $a_1, \ldots, a_m$.
- If $\text{RTT} \models f_1$ or $\text{RTT} \models f_2$ then $\text{RTT} \models f_1 \vee f_2$.
- If not $\text{RTT} \models f$, then $\text{RTT} \models \neg f$.
- If $f \equiv \forall x{:}t[h]$ and for all g of type t, $\text{RTT} \models h[x{:=}g]$[4], then $\text{RTT} \models \forall x{:}t[h]$.

REMARK 2.14
At first sight, the reader might expect a clause for the case $f \equiv z(k_1, \ldots, k_m)$ in the above definition. However, $\text{FV}(z(k_1, \ldots, k_m)) \supseteq \{z\}$, so $\text{FV}(z(k_1, \ldots, k_m)) \neq \varnothing$. Propositional functions of the form $z(k_1, \ldots, k_m)$ only occur in the above definition in a form in which the variable z has been bound by a quantifier. As was noted earlier (in Subsection 2.1) expressions of the form $f(k_1, \ldots, k_n)$, where f is a propositional function, do not exist in RTT.

REMARK 2.15
This definition of logical truth is quite informal. For example, the first clause 'If $(a_1, \ldots, a_m) \in R$ then $\text{RTT} \models R(a_1, \ldots, a_m)$' requires the symbol R to be already fully interpreted and to denote a relation independently of any Tarskian assignment function. This is faithful to Russell, for whom the Tarskian notion of an uninterpreted formal language was quite alien.

3 Kripke's Theory of Truth KTT

In this section, we shortly describe Kripke's Theory of Truth KTT (see [8]). Kripke expresses higher-order formulas within a first-order language, using the fact that many interesting languages are rich enough to express their own syntax (for instance, via a Gödel Numbering).

Let us assume a first-order language L, with variables ranging over a domain D, and primitive predicates interpreted by (totally defined) relations on D. Let us also assume two subsets S_1 and S_2 of D such that $S_1 \cap S_2 = \varnothing$. Kripke extends L to $L(S_1, S_2)$ by adding a monadic predicate $\mathtt{T}$. The main idea is to interpret $\mathtt{T}$ as a 'truth predicate'. S_1 contains the elements d of D for which $\mathtt{T}(d)$ holds (so it contains the (codes of) formulas which we consider to be 'true'); S_2 contains those $d \in D$ for which $\neg\mathtt{T}(d)$ holds (hence it contains the (codes of) formulas which we consider to be 'false'). We do not demand that $S_1 \cup S_2 = D$, hence $\mathtt{T}$ is a partial predicate over D.

DEFINITION 3.1 (Logical truth for KTT)
Let L be a first-order language over a domain D with $\mathcal{R}$ as set of primitive predicates.

[4]$\text{FV}(h[x{:=}g]) = \varnothing$ by lemma 2.12

Let f be a sentence in L. We define $L \models f$ as follows[5]:

f	$L \models f$	$L \models \neg f$
$R(d_1, \ldots, d_m)$	$(d_1, \ldots, d_m) \in R$	$(d_1, \ldots, d_m) \notin R$
$g_1 \wedge g_2$	$L \models g_1$ and $L \models g_2$	$L \models (\neg g_1) \vee (\neg g_2)$
$g_1 \vee g_2$	$L \models g_1$ or $L \models g_2$	$L \models (\neg g_1) \wedge (\neg g_2)$
$\forall x[g]$	$L \models g[x{:=}d]$ for all $d \in D$	$L \models \exists x[\neg g]$
$\exists x[g]$	$L \models g[x{:=}d]$ for some $d \in D$	$L \models \forall x[\neg g]$
$\neg\neg g$	$L \models g$	$L \models \neg g$

Here, $R \in \mathcal{R}$; $d, d_1, \ldots, d_m \in D$, and g, g_1, g_2 are formulas of L. Now let $S_1, S_2 \subseteq D$ such that $S_1 \cap S_2 = \varnothing$. KTT $\equiv L(S_1, S_2)$ is the first order language over D with $\mathcal{R} \cup \{\mathtt{T}\}$ as the set of primitive predicates ($\mathtt{T} \notin \mathcal{R}$). We extend the definition of $L \models f$ to KTT $\models f$ by putting KTT $\models \mathtt{T}(d)$ iff $d \in S_1$ and KTT $\models \neg\mathtt{T}(d)$ iff $d \in S_2$.

It is important (and easy) to notice that the extension of L to $L(S_1, S_2)$ is conservative:

Lemma 3.2
Let L be a first order language over a domain D, let $S_1, S_2 \subseteq D$ such that $S_1 \cap S_2 = \varnothing$, and assume that f is a sentence in L. Then $L \models f$ if and only if $L(S_1, S_2) \models f$.

Now Kripke uses $\mathtt{T}$ as a predicate expressing truth by defining a hierarchy of languages. This hierarchy has much in common with Russell's hierarchy of orders. L was assumed to be able to express its own syntax, hence so is $L(S_1, S_2)$, for any S_1, S_2. Notice that the sentences of $L(S_1, S_2)$ do not depend on the sets S_1 and S_2, so we can take one Gödel Numbering $\langle\rangle$, being a map from the formulas of $L(S_1, S_2)$ to D. The Kripke-hierarchy of languages is defined by presenting a hierarchy of pairs of sets (S_1, S_2):

Definition 3.3
For any ordinal α we define a pair of sets $(S_{\alpha,1}, S_{\alpha,2})$ and a language KTT_α.

- $S_{0,1} \overset{\text{def}}{=} \varnothing$; $S_{0,2} \overset{\text{def}}{=} \varnothing$; $\text{KTT}_0 \overset{\text{def}}{=} L(S_{0,1}, S_{0,2})$.
- If $S_{\alpha,1}$, $S_{\alpha,2}$ and KTT_α have been defined, then we define:

$$\begin{aligned} S_{\alpha+1,1} &\overset{\text{def}}{=} \{\langle f\rangle | f \text{ is a sentence and } \text{KTT}_\alpha \models f\} \\ S_{\alpha+1,2} &\overset{\text{def}}{=} \{\langle f\rangle | f \text{ is a sentence and } \text{KTT}_\alpha \models \neg f\} \cup \\ &\cup \quad \{d \in D | d \not\equiv \langle f\rangle \text{ for all sentences } f \text{ of } \text{KTT}_\alpha\} \\ \text{KTT}_{\alpha+1} &\overset{\text{def}}{=} L(S_{\alpha+1,1}, S_{\alpha+1,2}) \end{aligned}$$

- If α is a limit ordinal and $S_{\beta,1}$, $S_{\beta,2}$ and KTT_β have been defined for all $\beta\langle\alpha$, then

$$\begin{aligned} S_{\alpha,i} &\overset{\text{def}}{=} \bigcup_{\beta\langle\alpha} S_{\beta,i} \\ \text{KTT}_\alpha &\overset{\text{def}}{=} L(S_{\alpha,1}, S_{\alpha,2}) \end{aligned}$$

[5]Notice that even though this definition is different from Tarski's definition, especially with respect to the definition of $L \models \neg f$, it is easy to prove the equivalence of both definitions. This is because all primitive predicates of L are totally defined. We took this definition however as we need to extend it for the partial predicate $\mathtt{T}$.

Lemma 3.4 (Conservation of knowledge)
If $\alpha\langle\beta$ then $S_{\alpha,1} \subseteq S_{\beta,1}$ and $S_{\alpha,2} \subseteq S_{\beta,2}$.

We can see the construction of the languages KTT_α as a process of obtaining knowledge. At the initial stage, KTT_0, $\mathtt{T}(d)$ is not defined for any $d \in D$. There is no knowledge at all.
Applying the definition of truth given for KTT_0, we obtain knowledge: Some sentences f can be judged true ($\text{KTT}_0 \models f$; we store the code of f in $S_{1,1}$), some other sentences g can be judged false ($\text{KTT}_0 \models \neg g$; the code of g is stored in $S_{1,2}$). It is not possible to judge all sentences. For instance, neither $\text{KTT}_0 \models \forall \mathtt{x}[\mathtt{T}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$ nor $\text{KTT}_0 \models \neg\forall \mathtt{x}[\mathtt{T}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$ hold, so $\langle\forall \mathtt{x}[\mathtt{T}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]\rangle$ neither belongs to $S_{1,1}$, nor to $S_{1,2}$.

The knowledge we obtained is expressed by the predicate $\mathtt{T}$ in KTT_1. In KTT_1 we know more about $\mathtt{T}$ than in KTT_0. Hence more sentences can be judged true or false; we store their codes in $S_{2,1}$ and $S_{2,2}$ respectively. The lemma on Conservation of Knowledge 3.4 guarantees that this process only extends our knowledge, i.e.:

- Sentences that were judged to be true at level KTT_1 remain true at level KTT_2;
- Sentences that were judged to be false at level KTT_1 remain false at level KTT_2.

By iterating this process we arrive at the levels $\text{KTT}_3, \text{KTT}_4, \ldots, \text{KTT}_\omega, \text{KTT}_{\omega+1}, \ldots$. This limit does terminate however in that it has a fixed point.

4 RTT in KTT

Both in RTT and in KTT we are confronted with a hierarchy. Russell constructs a hierarchy by dividing propositions and propositional functions into different orders, taking care that a propositional function f can only depend on objects of a lower order than the order of f.

Kripke does not make this distinction beforehand. He has only one truth-predicate ($\mathtt{T}$), but decisions about truth of propositions are split into different levels: At the first level only decisions about propositions that do not involve $\mathtt{T}$ are made, at the second level decisions about propositions involving $\mathtt{T}$ for codes of first-level propositions are made, and so on.

In Subsection 4.1 we investigate the similarity between both hierarchies, by describing RTT within KTT. In Subsection 4.2 we investigate in which way RTT is more restrictive with respect to self-reference than KTT.

4.1 RTT *embedded in* KTT

To embed RTT in a first order language L, we have to cope with two technical problems:

- We need to encode the notion of and the manipulation with (higher-order) propositional functions into a first-order language. The manipulation is especially important with respect to substitution, which in the higher-order situation is much more complicated than in the first order case (cf. the definition of substitution 2.9).
- In Russell's theory, it is possible (and, due to the hierarchy of orders, in fact *only* possible) to quantify over only a part of all propositions. This makes it

impossible to translate, for instance, the proposition $\forall \mathtt{p}{:}()^1[\mathtt{p}() \vee \neg\mathtt{p}()]$ directly by $\forall \mathtt{x}[\mathtt{T}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$, as the quantifier in the latter also quantifies over (codes of) higher-order propositions.

As we do not want contexts to be involved in this coding, we assume that each variable in $\mathcal{V}$ has (implicitly) a superscript t, indicating its type. This makes it possible to do without contexts, as the types of the variables are now clear from the function in which they occur. For reasons of clarity, we will not write this superscript explicitly, as long as no confusion arises.

We propose the following solutions to the problems sketched above (we first give the definition and afterwards explain our thoughts behind it):

Definition 4.1
Let KTT be the language L with domain $D = \mathcal{A}$, extended with for each ramified type t a monadic predicate $\mathtt{Typ}_t$, for each $n \in \mathbb{N}$ a $(n{+}1)$-ary function $\mathtt{App}_n$, and the monadic predicate $\mathtt{T}$ ($\mathtt{T}$ will play the same role as in Section 3). We code the typable propositional functions f of $\mathcal{F}$ to formulas $\overline{f}$ in the language KTT. We do this by induction on the structure of f.

- If $f \equiv R(i_1, \ldots, i_{\mathbf{a}(R)})$, then f is present in the original language L and we take $\overline{f} \stackrel{\text{def}}{=} f$.
- If $f \equiv z(k_1, \ldots, k_m)$, write $K_i \equiv \langle\overline{k_i}\rangle$ for $k_i \in \mathcal{F}$, and $K_i \equiv k_i$ for $k_i \in \mathcal{A} \cup \mathcal{V}$. Define $\overline{f} \stackrel{\text{def}}{=} \mathtt{T}(\mathtt{App}_m(z, K_1, \ldots, K_m))$.
- If $f \equiv f_1 \vee f_2$, then $\overline{f} \stackrel{\text{def}}{=} \overline{f_1} \vee \overline{f_2}$.
- If $f \equiv \neg f'$, then $\overline{f} \stackrel{\text{def}}{=} \neg\overline{f'}$.
- If $f \equiv \forall x : u[f']$, then $\overline{f} \stackrel{\text{def}}{=} \forall x[\neg\mathtt{Typ}_u(x) \vee \overline{f'}]$.

We now give a formal interpretation to the newly introduced predicates $\mathtt{Typ}_t$ and $\mathtt{App}_n$.

Definition 4.2
For all ramified types $t \neq \iota^0$, let $\mathrm{Typ}_t \stackrel{\text{def}}{=} \{\langle\overline{f}\rangle | f \in \mathcal{F} \text{ and } f : t\}$ and $\mathrm{Typ}_{\iota^0} \stackrel{\text{def}}{=} D$.

Assume: $n \in \mathbb{N}$, $f \in \mathcal{F}$ is of type $(t_1, \ldots, t_n)$ and has free variables $x_1\langle\ldots\langle x_n$. Assume also: for $i = 1, \ldots, n$, $k_i : t_i$ and either $d_i = k_i$ (if $t_i = \iota^0$) or $d_i = \langle\overline{k_i}\rangle$ (if $t_i \neq \iota^0$). We define:

$$\mathrm{App}_n(\langle\overline{f}\rangle, d_1, \ldots, d_n) \stackrel{\text{def}}{=} \langle\overline{f[x_1, \ldots, x_n{:=}k_1, \ldots, k_n]}\rangle.$$

From now on, we will interpret the function symbol $\mathtt{App}_n$ as the function App_n, and the relation symbol $\mathtt{Typ}_t$ as the relation Typ_t.

We make some remarks with respect to these definitions.

Remark 4.3
It is clear that the newly introduced functions App_n are used for carrying out substitutions, thus solving the first of the technical problems stated at the beginning of this subsection. The predicates Typ_t solve the second problem, as can be seen in the definition of $\overline{\forall x[f]}$.

Remark 4.4
Notice that we did not define the functions App_n on the full domain D^{n+1}. We could have done that, but will not need App_n on other elements of D^{n+1} than defined above.

Remark 4.5
At this point, our work is related to (but independent of) Paul Gilmore's work on NaDSet 1. NaDSet 1 is a theory of generalized abstraction which makes n-ary predication a primitive of the system, with the unary truth predicate being trivially definable upon this basis. For a useful connection between KTT and NaDSet 1, see [4].

Remark 4.6
The extensions suggested above are of a mere technical character. Therefore, we think that we can still speak of an *embedding* of RTT within KTT.

Notation 4.7
To keep notations uniform, we sometimes want to speak about $\langle\overline{x}\rangle$ when we only intend to mention x, for $x \in \mathcal{V}$, and about $\langle\overline{a}\rangle$ when only meaning a, for $a \in \mathcal{A}$. Hence, we formally define: $\langle\overline{x}\rangle \overset{\text{def}}{=} x$ and $\langle\overline{a}\rangle \overset{\text{def}}{=} a$ for all $x \in \mathcal{V}$ and all $x \in \mathcal{A}$.

Below, we work in two systems: RTT and KTT. These systems have a different notion of substitution, though they use the same notation for expressing substitution. From the context, however, it will always be clear which kind of substitution is meant.

The language KTT above is similar to that presented in Section 3, and we construct KTT_α for each ordinal α as described in that section. We need the following lemma:

Lemma 4.8 (Substitution lemma)
Assume g is a propositional function of order m and $g[x{:=}k]$ is a proposition of order n. If $\mathrm{KTT}_n \models \overline{g[x{:=}k]}$ then $\mathrm{KTT}_m \models \overline{g}[x{:=}\langle\overline{k}\rangle]$.

Proof. We make the proof a little easier by proving that if: *If g is a propositional function of order m and $g[x_1,\ldots,x_p{:=}k_1,\ldots,k_p]$ is a proposition of order n, then 1 and 2 hold where*

1. $\mathrm{KTT}_n \models \overline{g[x_1,\ldots,x_p{:=}k_1,\ldots,k_p]}$ implies $\mathrm{KTT}_m \models \overline{g}[x_1,\ldots,x_p{:=}\langle\overline{k_1}\rangle,\ldots,\langle\overline{k_p}\rangle]$
2. $\mathrm{KTT}_n \models \overline{\neg g[x_1,\ldots,x_p{:=}k_1,\ldots,k_p]}$ implies $\mathrm{KTT}_m \models \overline{\neg g}[x_1,\ldots,x_p{:=}\langle\overline{k_1}\rangle,\ldots,\langle\overline{k_p}\rangle]$

We write $g[x_i{:=}k_i]$ as a shorthand for $g[x_1,\ldots,x_p{:=}k_1,\ldots,k_p]$ as long as no confusion arises, and use similar abbreviations for other substitutions. The proof is by induction on the structure of g.

- $g \equiv R(i_1,\ldots,i_{\mathbf{a}(R)})$. Then, by definition of $g[x_i{:=}k_i]$, $\overline{g[x_i{:=}k_i]} \equiv \overline{g}[x_i{:=}\langle\overline{k_i}\rangle]$. As $n \leq m$, the lemma follows by the lemma on Conservation of Knowledge 3.4.
- $g \equiv z(h_1,\ldots,h_q)$. If $z \notin \{x_1,\ldots,x_p\}$ then again $\overline{g[x_i{:=}k_i]} \equiv \overline{g}[x_i{:=}\langle\overline{k_i}\rangle]$[6] and again the lemma follows from $n \leq m$ and the lemma on Conservation of Knowledge 3.4.
  The interesting case is when $g \equiv z(h_1,\ldots,h_q)$ and $z \in \{x_1,\ldots,x_p\}$. To keep notations clear, we assume $p = 1$ and $z = x_1$. The reader may verify that the case $p\rangle 1$ only complicates notation, not the proof. We only show 1 as 2 is similar. Assume $\mathrm{KTT}_n \models \overline{g[x_i{:=}k_i]}$.

[6]This is because in this case, no higher order substitutions occur, and the notion of RTT-substitution coincides with ordinary, first order substitution.

As k_1 and z have the same type, k_1 has q free variables, say $y_1\langle\ldots\langle y_q$, and by definition of substitution in RTT, $z(h_1,\ldots,h_q)[x_1{:=}k_1] \equiv k_1[y_i{:=}h_i]$. Notice that z and k_1 have the same order $(m{-}1)$, and that n, the order of $k_1[y_i{:=}h_i]$, is at most the order of k_1 (lemma 2.11). This means: $n \leq m-1$. Using lemma 3.4: $\text{KTT}_{m-1} \models \overline{k_1[y_i{:=}h_i]}$.

By the definition of $\mathtt{T}$ we have: $\text{KTT}_m \models \mathtt{T}\left(\left\langle\overline{k_1[y_i{:=}h_i]}\right\rangle\right)$. We are now done because:

$$\begin{array}{rcl}
\overline{g}[x_1{:=}\langle\overline{k_1}\rangle] & \equiv & \overline{z(h_1,\ldots,h_q)}[z{:=}\langle\overline{k_1}\rangle] \\
 & \equiv & \mathtt{T}(\mathtt{App}_q(z,\langle\overline{h_1}\rangle,\ldots,\langle\overline{h_q}\rangle))[z{:=}\langle\overline{k_1}\rangle] \\
 & \equiv & \mathtt{T}(\mathtt{App}_q(\langle\overline{k_1}\rangle,\langle\overline{h_1}\rangle,\ldots,\langle\overline{h_q}\rangle)) \\
 & \equiv & \mathtt{T}\left(\left\langle\overline{k_1[y_i{:=}h_i]}\right\rangle\right)
\end{array}$$

- $g \equiv g_1 \vee g_2$.
  First, assume $\text{KTT}_n \models \overline{g[x_i{:=}k_i]}$. As $\overline{g[x_i{:=}k_i]} \equiv \overline{g_1[x_i{:=}k_i]} \vee \overline{g_2[x_i{:=}k_i]}$, there is j such that $\text{KTT}_n \models \overline{g_j[x_i{:=}k_i]}$. By the induction hypothesis, there is j such that $\text{KTT}_m \models \overline{g_j}[x_i{:=}\langle\overline{k_i}\rangle]$, as the order of g_j is $\leq m$. Hence $\text{KTT}_m \models \overline{g_1}[x_i{:=}\langle\overline{k_i}\rangle] \vee \overline{g_2}[x_i{:=}\langle\overline{k_i}\rangle]$, so we are done.
  Now assume $\text{KTT}_n \models \neg\overline{g[x_i{:=}k_i]}$. This means: $\text{KTT}_n \models \neg(\overline{g_1[x_i{:=}k_i]} \vee \overline{g_2[x_i{:=}k_i]})$. Hence $\text{KTT}_n \models \neg\overline{g_j[x_i{:=}k_i]}$ for $j=1,2$, and by the induction hypothesis, this means (again the order of the g_js are $\leq m$) $\text{KTT}_m \models \neg\overline{g_j}[x_i{:=}\langle\overline{k_i}\rangle]$ for $j=1,2$, hence
  $\text{KTT}_m \models \neg\overline{g_1}[x_i{:=}\langle\overline{k_i}\rangle] \wedge \neg\overline{g_2}[x_i{:=}\langle\overline{k_i}\rangle]$.
  So $\text{KTT}_m \models \neg(\overline{g_1}[x_i{:=}\langle\overline{k_i}\rangle] \vee \overline{g_2}[x_i{:=}\langle\overline{k_i}\rangle])$, and $\text{KTT}_m \models (\neg(g_1 \vee g_2))[x_i{:=}\langle\overline{k_i}\rangle]$.
- $g \equiv \neg g'$.
  If $\text{KTT}_n \models \overline{g[x_i{:=}k_i]}$ then use the induction hypothesis for g'.
  If $\text{KTT}_n \models \neg\overline{g[x_i{:=}k_i]}$ then $\text{KTT}_n \models \overline{g'[x_i{:=}k_i]}$, so by induction $\text{KTT}_m \models \overline{g'}[x_i{:=}\langle\overline{k_i}\rangle]$, so $\text{KTT}_m \models \overline{\neg\neg g'}[x_i{:=}\langle\overline{k_i}\rangle]$.
- $g \equiv \forall x{:}t[g']$.
  If $\text{KTT}_n \models \overline{g[x_i{:=}k_i]}$, then for all d such that $\text{Typ}_t(d)$, $\text{KTT}_n \models \overline{g'[x_i{:=}k_i]}[x{:=}d]$, hence for all these d, $\text{KTT}_m \models \overline{g'}[x_i{:=}\langle\overline{k_i}\rangle][x{:=}d]$, so $\text{KTT}_m \models \forall x[\overline{g'}[x_i{:=}\langle\overline{k_i}\rangle]]$, and $\text{KTT}_m \models \overline{g}[x_i{:=}\langle\overline{k_i}\rangle]$.
  If $\text{KTT}_n \models \neg\overline{g[x_i{:=}k_i]}$ then there is $d \in D$ such that $\text{Typ}_t(d)$ and $\text{KTT}_n \models \neg\overline{g'[x_i{:=}k_i]}[x{:=}d]$, hence $\text{KTT}_m \models \neg\overline{g'}[x_i{:=}\langle\overline{k_i}\rangle][x{:=}d]$, and $\text{KTT}_m \models \exists x[\neg\overline{g'}[x_i{:=}\langle\overline{k_i}\rangle]]$.
  Hence $\text{KTT}_m \models \exists x[\neg\overline{g'}[x_i{:=}\langle\overline{k_i}\rangle]]$ and $\text{KTT}_m \models \neg\forall x[\overline{g'}[x_i{:=}\langle\overline{k_i}\rangle]]$.

∎

REMARK 4.9
We have actually proven a stronger fact: *Assume g is a propositional function of order m and $g[x{:=}k]$ is a proposition of order n. If* $\text{KTT}_n \models \overline{g[x{:=}k]}$ *then* $\text{KTT}_p \models \overline{g}[x{:=}\langle\overline{k}\rangle]$, *where* $p = min(m, n{+}1)$. This tells us more about the role of the predicate $\mathtt{T}$: Although a substitution may lower the order of a propositional function by more than one, only one application of the $\mathtt{T}$-predicate is involved (hence only one level in the hierarchy of truths). However, in the theorem below we only need the (weaker) form in which we presented the substitution lemma originally.

Theorem 4.10
Let $f : ()^n \in \mathcal{F}$. Then: RTT $\models f$ if and only if $\text{KTT}_n \models \overline{f}$.

Proof. [$\Leftarrow$] Due to the use of $\neg$ in the definition of $\text{KTT}_n \models \overline{f}$, we prove a little bit more:

- If RTT $\models f$ then $\text{KTT}_n \models \overline{f}$;
- If RTT $\models \neg f$ then $\text{KTT}_n \models \overline{\neg f}$.

These claims are proved simultaneously by induction on the definition of RTT $\models f$.

- $f \equiv R(d_1, \dots, d_{\mathbf{a}(R)})$ for a $R \in \mathcal{R}$ and some $d_1, \dots, d_{\mathbf{a}(R)} \in D$. Then $\overline{f} \equiv f$. As RTT $\models f$, we know that $(d_1, \dots, d_{\mathbf{a}(R)}) \in R$, hence $\text{KTT}_n \models \overline{f}$. The proof is similar for $\neg f$.
- $f \equiv g_1 \vee g_2$. Then the orders of the g_is are either equal to, or smaller than n.
  First assume RTT $\models f$. Then we know that RTT $\models g_i$ for $i = 1$ or $i = 2$. By the induction hypothesis (and Conservation of Knowledge, if the order of g_i is $\langle n \rangle$), $\text{KTT}_n \models \overline{g_i}$, As $\overline{f} \equiv \overline{g_1} \vee \overline{g_2}$, $\text{KTT}_n \models \overline{f}$.
  Now assume RTT $\models \neg f$. Then it is not true that RTT $\models f$, so it is not true that RTT $\models g_i$ for $i = 1$ or $i = 2$. So RTT $\models \neg g_i$ for $i = 1, 2$. By the induction hypothesis (and, again, possibly Conservation of Knowledge), we have $\text{KTT}_n \models \overline{\neg g_i}$, hence, $\text{KTT}_n \models \neg \overline{g_i}$ for $i = 1, 2$. So $\text{KTT}_n \models \neg \overline{g_1} \wedge \neg \overline{g_2}$,and hence so $\text{KTT}_n \models \overline{\neg f}$.
- $f \equiv \neg g$. If RTT $\models f$ then use IH on g to get $\text{KTT}_n \models \overline{\neg g}$, hence $\text{KTT}_n \models \overline{f}$.
  If RTT $\models \neg f$, then RTT $\models g$, so by induction $\text{KTT}_n \models \overline{g}$, so $\text{KTT}_n \models \neg\neg\overline{g}$, so $\text{KTT}_n \models \overline{\neg f}$.
- $f \equiv \forall x{:}t[g]$. Notice that g has order n.
  If RTT $\models f$ then for all $k{:}t$, RTT $\models g[x{:=}k]$. By the induction hypothesis, we know that for all $k : t$, $\text{KTT}_{m_k} \models \overline{g[x{:=}k]}$, where m_k is the order of $g[x{:=}k]$.
  By the substitution lemma 4.8 we have: For all $k : t$, $\text{KTT}_n \models \overline{g}[x{:=}\langle k \rangle]$. Hence, for all $d \in D$, $\text{KTT}_n \models \neg \mathtt{Typ}_t(d) \vee \overline{g}[x{:=}d]$. Hence $\text{KTT}_n \models \forall x : t[g]$.
  The argument for RTT $\models \neg f$ is similar.

[$\Rightarrow$] This is easy now. Assume, for the sake of the argument, not RTT $\models f$. Then RTT $\models \neg f$, hence $\text{KTT}_n \models \neg \overline{f}$ and $\text{KTT}_n \models \overline{f}$, which is a contradiction.

∎

This theorem clearly shows the relation between the orders in RTT and the levels of truth in KTT. The heart of the proof of theorem 4.10 is in the proof of case $z(h_1, \dots, h_q)$ of the substitution lemma 4.8. This is the only place in the proof where the properties of the predicate $\mathtt{T}$ are used. It is understandable that these properties must be used at exactly this place when we look at the definition of propositional functions and the typing rules for propositional functions. Exactly the possibility of constructing a propositional function of the form $z(h_1, \dots, h_q)$ makes it possible to arrive at higher-order propositional functions and higher-order propositions. So exactly at this spot, Kripke's predicate $\mathtt{T}$ must appear, in order to raise one level in KTT as well.

Corollary 4.11
Rtt $\models f$ if and only if $\text{KTT}_\omega \models \overline{f}$.

We cannot improve the result of theorem 4.10 in general: There are propositions f of order n in RTT whose code is provable at level KTT_n in KTT, but not at any lower level.

THEOREM 4.12
Let $n\rangle 0$, and let f_n be the nth-order-proposition $\forall \mathtt{p}{:}()^{n-1}[\mathtt{p}() \vee \neg\mathtt{p}()]$. Then:

$$\text{KTT}_m \models \overline{f_n} \text{ if and only if } m \geq n.$$

PROOF. [$\Leftarrow$] follows from theorem 4.10 and lemma 3.4. [$\Rightarrow$] is by induction on n. Observe that

$$\overline{f_n} \equiv \forall \mathtt{p}[\neg\mathtt{Typ}_{()^{n-1}}(\mathtt{p}) \vee (\mathtt{T}(\mathtt{App}_0(\mathtt{p})) \vee \neg\mathtt{T}(\mathtt{App}_0(\mathtt{p})))].$$

- $n = 1$. Let g be any proposition of order 0 in RTT. Then $\text{KTT}_0 \models \mathtt{Typ}_{()^0}(g)$ but as $\mathtt{T}$ is completely undefined at level 0, $\text{KTT}_0 \not\models \mathtt{T}(\mathtt{App}_0(g)) \vee \neg\mathtt{T}(\mathtt{App}_0(g))$. Hence, $\text{KTT}_0 \not\models \overline{f_1}$.
- Assume the theorem has been proved for all $n'\langle n$. Assume $m\langle n$ and $\text{KTT}_m \models \overline{f_n}$. By definition of $\models$, we have: $\text{KTT}_m \models \mathtt{T}(\mathtt{App}_0(\langle\overline{f_{n-1}}\rangle)) \vee \neg\mathtt{T}(\mathtt{App}_0(\langle\overline{f_{n-1}}\rangle))$, and for reasons of consistency: $\text{KTT}_m \models \mathtt{T}(\mathtt{App}_0(\langle\overline{f_{n-1}}\rangle))$, hence $\text{KTT}_m \models \mathtt{T}(\overline{f_{n-1}})$, so, by the definition of $\mathtt{T}$: $\text{KTT}_{m-1} \models \overline{f_{n-1}}$, which contradicts the induction hypothesis, as $m - 1\langle n - 1$.

∎

There are, however, propositions f of order n in RTT for which $\text{KTT}_m \models \overline{f}$ or $\text{KTT}_m \models \overline{\neg f}$ can already be established for $m\langle n$.

EXAMPLE 4.13
Consider a proposition $g \equiv g_1 \vee g_2$ where g_1 is a true proposition of order m and g_2 is any proposition of order n. As g_1 is true in RTT, we have $\text{KTT}_m \models \overline{g_1}$, and therefore $\text{KTT}_m \models \overline{g}$.

4.2 The restrictiveness of Russell's theory

We illustrate the different approaches of Russell and Kripke by an example given by Kripke himself.

EXAMPLE 4.14
Let D, $\mathcal{R}$, L, $S_{\alpha,i}$ and KTT_α be as in Section 3 where $\mathcal{R}$ contains two monadic predicates $\mathtt{V}$ and $\mathtt{W}$ which are collections of (codes of) utterances of persons **V** and **W**. Now define

$$P \equiv \forall \mathtt{x}[\neg\mathtt{W}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$$
$$Q \equiv \forall \mathtt{x}[\neg\mathtt{V}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$$

(informally, P denotes: All utterances of **W** are false, and Q denotes: All utterances of **V** are false). Now distinguish two situations. In both situations, we want to know whether P and Q become true or false when passing through the hierarchy of languages KTT_0, $\text{KTT}_1, \ldots$. Or, more formally, whether there is α such that $\langle\mathtt{V}\rangle$ and $\langle\mathtt{W}\rangle$ belong to $S_{\alpha,1} \cup S_{\alpha,2}$.

1. $\mathtt{V} = \{\langle P\rangle\}$ and $\mathtt{W} = \{\langle Q\rangle\}$ (notice that $\mathtt{V}$ and $\mathtt{W}$ are just subsets of D).
   In this case, P is logically equivalent to $\neg\mathtt{T}(\langle Q\rangle)$ and Q is logically equivalent to $\neg\mathtt{T}(\langle P\rangle)$. As a consequence we have: if $\langle Q\rangle \in S_{\alpha,i}$ then $\langle P\rangle \in S_{\beta,3-i}$ for some $\beta\langle\alpha$, and if $\langle P\rangle \in S_{\alpha,i}$ then $\langle Q\rangle \in S_{\beta,3-i}$ for some $\beta\langle\alpha$. Hence $\langle P\rangle, \langle Q\rangle \notin S_{\alpha_i}$, for all α, i, so neither the truth of P nor the truth of Q will ever be established.

2. In the situation above, the only utterance of **V** was that anything said by **W** is false, and vice versa. In that case, it is also intuitively clear that it is impossible to say anything about the truth of P or Q. Now we change the situation. We assume that $\mathcal{R}$ also contains a third monadic predicate $\mathtt{R}$, and that $\mathtt{d}$ is an element of $\mathtt{R}$. We redefine $\mathtt{W}$:
$$\mathtt{W} \stackrel{\mathrm{def}}{=} \{\langle Q\rangle, \langle \mathtt{R}(\mathtt{d})\rangle\}.$$
This has drastical consequences. As $\textsc{ktt}_0 \models \mathtt{R}(\mathtt{d})$, $\langle\mathtt{R}(\mathtt{d})\rangle \in S_{1,1}$, so $\textsc{ktt}_1 \models \mathtt{T}(\langle\mathtt{R}(\mathtt{d})\rangle)$, hence $\textsc{ktt}_1 \models \neg P$. Therefore, $\langle P\rangle \in S_{2,2}$, so $\textsc{ktt}_2 \models \neg\mathtt{T}(\langle \mathtt{P}\rangle)$, hence:
$$\begin{aligned} \textsc{ktt}_2 &\models \neg P \\ \textsc{ktt}_2 &\models Q \end{aligned}$$
The fact that **W** utters a true sentence makes it possible to conclude at level 1 that P is false, irrespective of the fact that **W** has also uttered another sentence Q, of which we can't establish the truth at level 1. The falsehood of P makes it possible to decide about Q at the next level, so the falsehood of P and the truth of Q could have been established at level 2.

In Russell's terminology it wouldn't be possible to write expressions like P and Q at all: They are excluded beforehand, as P involves Q, therefore has to be of higher order than Q, and Q involves P, therefore has to be of higher order than P.

This indicates an important difference between RTT and KTT: Kripke allows much more expressions to be written down. In some situations these expressions will never obtain any truth-value (like P and Q in the first example), but in other situations (so: with other definitions of the primitive predicates) the same expressions will get a truth-value. Kripke concludes: *'it would be fruitless to look for an* intrinsic *criterion that will enable us to sieve out – as meaningless, or ill-formed – those sentences which lead to paradox'*.

EXAMPLE 4.15
Another, more formal, example of a proposition f in KTT for which there is no $g \in \mathcal{F}$ with $\overline{g} \equiv f$ is the proposition $f \stackrel{\mathrm{def}}{=} \forall\mathtt{x}[\mathtt{T}(\mathtt{x}) \vee \neg\mathtt{T}(\mathtt{x})]$:

Assume, for the sake of the argument, that $\overline{g} \equiv f$. Let m be the order of g. Then $\textsc{ktt}_m \models f$ or $\textsc{ktt}_m \models \neg f$. This implies $\textsc{ktt}_m \models \mathtt{T}(f_m) \vee \neg\mathtt{T}(f_m)$, where f_m is as in theorem 4.12. By definition of $\mathtt{T}$ this means $\textsc{ktt}_{m-1} \models f_m$ or $\textsc{ktt}_{m-1} \models \neg f_m$, both contradicting theorem 4.12.

5 Orders and types

RTT is based on a double hierarchy: One of types and one of orders. This double hierarchy is too restrictive. It is possible to develop Logic and Mathematics within RTT, but for instance the proof of the Supremum Theorem, which is fundamental in real analysis, cannot be given. The origin of the problem is the use of the so-called predicative and impredicative propositional functions.

DEFINITION 5.1
Let $f \in \mathcal{F}$ be typable in RTT. Assume f has free variables $x_1, \ldots, x_n$ of orders $m_1, \ldots, m_n$ respectively. f is called *predicative* if its order is equal to $\max(m_1, \ldots, m_n)+1$; if its order is greater then f is called *impredicative*.

As the impredicative propositional functions cause problems, the 'Axiom of Reducibility' is proposed in 'Principia Mathematica' (1910–1912). This axiom is as follows:

For each $f \in \mathcal{F}$ there is a predicative *$g \in \mathcal{F}$ that is logically equivalent to f*

This axiom has been controversial from the moment it was introduced. Russell himself admits that *'it has a purely pragmatic justification: it leads to the desired results, and to no others. But clearly it is not the sort of axiom with which we can rest content.'* Though serious efforts have been made to develop Mathematics within RTT (for instance by Weyl [15]), this has not become the usual practice. In 1925, Ramsey [11] shows that, by making distinction between language and meta-language, the orders can be removed from the system without re-introducing any known paradox. Hilbert and Ackermann [7] present a similar idea. With this remark the type-theoretic foundations for the Simple Theory of Types STT, introduced by Church [1] in 1940, were laid, and orders have remained out of the important modern type systems up till now.

It is therefore interesting to notice the relation between orders in RTT and levels of truth in KTT, as formulated in theorem 4.10. It shows that Kripke's system can be regarded as a system based on RTT, of which not the *orders*, but the *types* have been removed. In this way, KTT can be seen as a system that is dual to STT.

KTT, however, has a more subtle approach than many type theories as it does not exclude any, possibly 'paradoxical', expression from the syntax, which is the usual type-theoretic approach. If an expression is paradoxical, it will not get a truth value at any level α of the hierarchy of Truths. Whether an expression is paradoxical or not does not only depend on its syntactic structure, but also on the domain D (see example 4.14). So paradoxes are only excluded at the level of semantics.

The discussion above shows that the orders of RTT are not to be blamed for the restrictiveness of RTT. KTT is a system which contains orders but has only few restrictions towards self-application.

It is the *combination* of orders and types that makes RTT restrictive.

6 Conclusion

6.1 Results

We presented a formalisation of Russell's Ramified Theory of Types RTT which is faithful to both Russell's original informal presentation and the present formulations of type theories.

We used this formalisation to compare RTT with Kripke's Theory of Truth KTT. We established the relation between Russell's hierarchy of orders and Kripke's hierarchy of truth-levels. In particular we showed that

1. A proposition of RTT of order n is true if and only if it is true at level n in Kripke's Truth Hierarchy (theorem 4.10).
2. The truth of some propositions of order n of RTT cannot be established in KTT at a level of truth hierarchy smaller than n (theorem 4.12). Yet for some other propositions, it can be established at an earlier level (example 4.13).

We also saw that Russell's theory has many restrictions. On the one hand, all propositional functions of RTT can be coded in Kripke's Truth Theory; on the other hand there are formulas of Kripke's theory that cannot be expressed in RTT, respecting both hierarchies.

We conclude, as so often has been concluded in Logic, in Mathematics and in Natural Language, that Russell's Theory of Types is too restrictive. However, the usual objections against RTT in Logic and Mathematics is the use of orders. After Ramsey [11] and Hilbert and Ackerman [7] had given motivations for leaving out these orders, they have hardly been used anywhere in logic or mathematics (though Weyl [15] has tried to give a build-up of mathematics within RTT).

Here the situation is completely different. Orders in RTT and truth-levels in KTT go hand in hand; on the other hand the types do not appear any more in KTT. This establishes KTT as the dual to STT (Church's Simple Type Theory) which removes the orders from RTT.

As far as we know, our contribution is the first statement of a formal correspondence between finite levels of truth in Kripkean Theory of Truth (KTT) and orders of quantification in Russell's Ramified Type Theory (RTT). Our conclusion is that, contra Ramsey, it is the restriction of the *mixture of orders and types* on predication rather than *order* restriction on quantification *alone* that accounts for the very restrictive nature of RTT. This is important and takes an added significance when seen in the context of the logicisation of second order arithmetic in a type free first order logic utilizing Kripke–Gilmore models which realises the hope of Russell's earlier type free substitutional theory.

6.2 Future work

Kripke's theory has a transfinite hierarchy of orders whereas Russell did not investigate such transfinity. It would be interesting hence to see how far one can build types in Russell's theory and what properties would hold at such level.

We concluded that some order-n-properties of RTT get their truth-value only at level n of KTT whilst others get it already at an earlier level. This divides propositions into two classes and an accurate description of these classes may be interesting.

As to the question of Kripke being more liberal in that any well-formed sentence can be expressed but its truth value may not be calculated (think of the paradoxical sentences), one may compare this approach to the implicit typing of Curry's Type Theory CTT [2, 3] where self-referential sentences may be expressed but are not typable. Hence, even though we said that KTT is the dual of STT, it may be the twin-brother of CTT where only truth or falsehood of *typable* terms can be determined. We are currently investigating this issue.

Acknowledgements

We would like to thank Rob Nederpelt and the anonymous referees for their constructive comments on an earlier version of this paper.

This work has been partially supported by EPSRC grant GR/K 25014.

Laan is supported by the Co-operation Centre Tilburg and Eindhoven Universities. He is grateful to the Department of Computing Science, University of Glasgow, for their hospitality, and to the Dutch Foundation for Scientific Research (NWO), for their financial support, during the preparation of this article.

References

[1] A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

[2] H.B. Curry and R. Feys. *Combinatory Logic*, volume I of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1958.

[3] H.B. Curry, J.R. Hindley, and J.P. Seldin. *Combinatory Logic*, volume II of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1972.

[4] S. Feferman. Toward useful type-free theories I. *Journal of Symbolic Logic*, 49:75–111, 1984.

[5] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879. Also in [6].

[6] J. van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, Cambridge, Massachusetts, 1967.

[7] D. Hilbert and W. Ackermann. *Grundzüge der Theoretischen Logik*. Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII. Springer Verlag, Berlin, first edition, 1928.

[8] S. Kripke. Outline of a theory of truth. *Journal of Philosophy*, 72:690–716, 1975.

[9] T.D.L. Laan. A formalization of the Ramified Type Theory. Technical Report 33, TUE Computing Science Reports, Eindhoven University of Technology, 1994.

[10] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*. Dordrecht, 1973.

[11] F.P. Ramsey. The foundations of mathematics. *Proceedings of the London Mathematical Society*, pages 338–384, 1925.

[12] B. Russell. *The Principles of Mathematics*. Allen & Unwin, London, 1903.

[13] B. Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30, 1908. Also in [6].

[14] A. Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica*, 1:261–405, 1936. German translation by L. Blauwstein from the Polish original (1933) with a postscript added.

[15] H. Weyl. *Das Kontinuum*. Veit, Leipzig, 1918. German; also in: Das Kontinuum und andere Monographien, Chelsea Pub.Comp., New York, 1960.

[16] A.N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1910^1, 1927^2. (All references in this paper are to the first volume).

Received 6 May 1995. Revised 7 September 1995