Implicit dynamic function introduction and Ackermann-like Function Theory

Marcos Cramer
University of Luxembourg
6, rue Richard Coudenhove-Kalergi, 1359 Luxembourg
marcos.cramer@uni.lu

Abstract

We discuss a feature of the natural language of mathematics – the implicit dynamic introduction of functions – that has, to our knowledge, not been captured in any formal system so far. If this feature is used without limitations, it yields a paradox analogous to Russell's paradox. Hence any formalism capturing it has to impose some limitations on it. We sketch two formalisms, both extensions of Dynamic Predicate Logic, that innovatively do capture this feature, and that differ only in the limitations they impose onto it. One of these systems is based on *Ackermann-like Function Theory*, a novel foundational theory of functions that is inspired by Ackermann Set Theory and that interprets ZFC.

Keywords: Dynamic Predicate Logic, Function Introduction, Ackermann Set Theory, Function Theory.

1 Dynamic Predicate Logic

Dynamic Predicate Logic (*DPL*) [7] is a formalism whose syntax is identical to that of standard first-order predicate logic (PL), but whose semantics is defined in such a way that the dynamic nature of natural language quantification is captured in the formalism:

1. If a farmer owns a donkey, he beats it.
2. PL: $\forall x\, \forall y\, (farmer(x) \wedge donkey(y) \wedge owns(x,y) \rightarrow beats(x,y))$
3. DPL: $\exists x\, (farmer(x) \wedge \exists y\, (donkey(y) \wedge owns(x,y))) \rightarrow beats(x,y)$

In PL, 3 is not a sentence, since the rightmost occurrences of x and y are free. In DPL, a variable may be bound by a quantifier even if it is outside its scope. The semantics is defined in such a way that 3 is equivalent to 2. So in DPL, 3 captures the meaning of 1 while being more faithful to its syntax than 2.

1.1 DPL semantics

We present DPL semantics in a way slightly different but logically equivalent to its definition by Groenendijk and Stokhof in [7]. Structures and assignments are defined as for PL: A structure S specifies a domain $|S|$ and an interpretation a^S for every constant, function or relation symbol a in the language. An S-assignment is a function from variables to $|S|$. Let G_S denote the set of S-assignments. Given two assignments g, h, we define $g[x]h$ to mean that g differs from h at most in what it assigns to the variable x. Given a DPL term t, we recursively define

$$[t]^g_S = \begin{cases} g(t) & \text{if } t \text{ is a variable,} \\ t^S & \text{if } t \text{ is a constant symbol,} \\ f^S([t_1]^g_S, \ldots, [t_n]^g_S) & \text{if } t \text{ is of the form } f(t_1, \ldots, t_n). \end{cases}$$

Groenendijk and Stokhof [7] define an interpretation function $[\![\cdot]\!]_S$ from DPL formulae to subsets of $G_S \times G_S$. We instead recursively define for every $g \in G_S$ an interpretation function $[\![\cdot]\!]^g_S$ from DPL formulae to subsets of G_S:[1]

1. $[\![\top]\!]^g_S := \{g\}$
2. $[\![t_1 = t_2]\!]^g_S := \{h | h = g \text{ and } [t_1]^g_S = [t_2]^g_S\}$[2]
3. $[\![R(t_1, \ldots, t_2)]\!]^g_S := \{h | h = g \text{ and } ([t_1]^g_S, \ldots, [t_2]^g_S) \in R^S\}$
4. $[\![\neg\varphi]\!]^g_S := \{h | h = g \text{ and there is no } k \in [\![\varphi]\!]^h_S\}$
5. $[\![\varphi \wedge \psi]\!]^g_S := \{h | \text{there is a } k \text{ s.t. } k \in [\![\varphi]\!]^g_S \text{ and } h \in [\![\psi]\!]^k_S\}$
6. $[\![\varphi \rightarrow \psi]\!]^g_S := \{h | h = g \text{ and for all } k \text{ s.t. } k \in [\![\varphi]\!]^h_S, \text{ there is a } j \text{ s.t. } j \in [\![\psi]\!]^k_S\}$
7. $[\![\exists x\ \varphi]\!]^g_S := \{h | \text{there is a } k \text{ s.t. } k[x]g \text{ and } h \in [\![\varphi]\!]^k_S\}$

$\varphi \vee \psi$ and $\forall x\ \varphi$ are defined to be a shorthand for $\neg(\neg\varphi \wedge \neg\psi)$ and $\exists x\ \top \rightarrow \varphi$ respectively.

2 Implicit dynamic introduction of function symbols

Functions are often dynamically introduced in an implicit way in mathematical texts. For example, [10] introduces the additive inverse function on the reals as follows:

[1] This can be viewed as a different currying of the uncurried version of Groenendijk and Stokhof's interpretation function.

[2] The condition $h = g$ in cases 2, 3, 4 and 6 implies that the defined set is either $\emptyset$ or $\{g\}$.

(a) For each a there is a real number $-a$ such that $a + (-a) = 0$. [10, p. 1]

Here the natural language quantification "there is a real number $-a$" *locally* (i.e. inside the scope of "For each a") introduces a new real number to the discourse. But since the choice of this real number depends on a and we are universally quantifying over a, it *globally* (i.e. outside the scope of "For each a") introduces a function "$-$" to the discourse.

The most common form of implicitly introduced functions are functions whose argument is written as a subscript, as in the following example:

(b) Since f is continuous at t, there is an open interval I_t containing t such that $|f(x) - f(t)| < 1$ if $x \in I_t \cap [a, b]$. [10, p. 62]

If one wants to later explicitly call the implicitly introduced function a function, the standard notation with a bracketed argument is preferred:

(c) Suppose that, for each vertex v of K, there is a vertex $g(v)$ of L such that $f(\mathrm{st}_K(v)) \subset \mathrm{st}_L(g(v))$. Then g is a simplicial map $V(K) \to V(L)$, and $|g| \backsimeq f$. [8, p. 19]

When no uniqueness claims are made about the object locally introduced to the discourse, implicit function introduction presupposes the existence of a choice function, i.e. presupposes the Axiom of Choice. We hypothesise that the naturalness of such implicit function introduction in mathematical texts contributes to the widespread feeling that the Axiom of Choice must be true.

Implicitly introduced functions generally have a restricted domain and are not defined on the whole universe of the discourse. In the example (c), g is only defined on vertices of K and not on vertices of L. Implicit function introduction can also be used to introduce multi-argument functions, but for the sake of simplicity and brevity, we restrict ourselves to unary functions in this paper.

If the implicit introduction of functions is allowed without limitations, one can derive a contradiction:

(d) For every function f, there is a natural number $g(f)$ such that

$$g(f) = \begin{cases} 0 & \text{if } f \in \mathrm{dom}(f) \text{ and } f(f) \neq 0, \\ 1 & \text{if } f \notin \mathrm{dom}(f) \text{ or } f(f) = 0. \end{cases}$$

Then g is defined on every function, i.e. $g(g)$ is defined. But from the definition of g, $g(g) = 0$ iff $g(g) \neq 0$.

This contradiction is due to the *unrestricted function comprehension* that is implicitly assumed when allowing implicit introductions of functions without limitations. Unrestricted function comprehension could be formalised as an axiom schema as follows:

Axiom Schema 1 (Unrestricted function comprehension)**.** For every formula $\varphi(x, y)$, the following is an axiom: $\forall x\ \exists y\ \varphi(x, y) \rightarrow \exists f\ \forall x\ \varphi(x, f(x))$

The inconsistency of unrestricted function comprehension is analogous to the inconsistency of unrestricted set comprehension, i.e. Russell's paradox.

Russell's paradox led to the abandonment of unrestricted comprehension in set theory. Two radically different approaches have been undertaken for restricting set comprehension: Russell himself restricted it through his Ramified Theory of Types, which was later simplified to Simple Type Theory (STT), mainly known via Church's formalisation in his simply typed lambda calculus [2]. On the other hand, the risk of paradoxes like Russell's paradox also contributed to the development of ZFC (Zermelo-Fraenkel set theory with the Axiom of Choice), which allows for a much richer set theoretic universe than the universe of simply typed sets. Since all the axioms of ZFC apart from the Axiom of Extensionality, the Axiom of Foundation and the Axiom of Choice are special cases of comprehension, one can view ZFC as an alternative way to restrict set comprehension.

Similarly, the above paradox must lead to the abandonment of unrestricted function comprehension. The type-theoretic approach is easily adapted to functions, so we will first sketch the system that formalises this approach, *Typed Higher-Order Dynamic Predicate Logic*. For an untyped approach, there is no clear way to transfer the limitations that ZFC puts onto set comprehension to the case of function comprehension. However, there is an axiomatization of set theory (with classes) called *Ackermann set theory* that is a conservative extension of ZFC. It turns out that the limitations that Ackermann set theory poses on set comprehension can be transferred to the case of function comprehension, and hence to the case of implicit dynamic function introduction.

The need to deal with implicit function introduction arose for us in the context of the *Naproche project*, a project aiming at automatic formalisation of natural language mathematics [3, 5, 6]. It has been implemented in the Naproche system using type restrictions as in Typed Higher-Order Dynamic Predicate Logic, and we plan to implement it using the less strict restrictions of the untyped Higher-Order Dynamic Predicate Logic in a future version of the system.

3 Typed Higher-Order Dynamic Predicate Logic

In this section, we extend DPL to a system called *Typed Higher-Order Dynamic Predicate Logic* (THODPL), which formalises implicit dynamic function introduction, and also allows for explicit quantification over functions. THODPL has variables typed by the types of STT. In the below examples we use x and y as variables of the basic type i, and f as a variable of the function type $i \to i$. A complex term is built by well-typed application of a function-type variable to an already built term, e.g. $f(x)$ or $f(f(x))$.

The distinctive feature of THODPL syntax is that it allows not only variables but any well-formed terms to come after quantifiers. So (1) is a well-formed formula:

$$\forall x\ \exists f(x)\ R(x, f(x)) \tag{1}$$

$$\forall x\ \exists y\ R(x, y) \tag{2}$$

$$\exists f\ (\forall x\ R(x, f(x))) \tag{3}$$

The semantics of THODPL is to be defined in such a way that (1) has the same truth conditions as (2). But unlike (2), (1) dynamically introduces the function symbol f to the context, and hence turns out to be equivalent to (3).

We now sketch how these desired properties of the semantics can be achieved. In THODPL semantics, an assignment assigns elements of $|S|$ to variables of type i, functions from $|S|$ to $|S|$ to variables of type $i \to i$ etc. Additionally, an assignment can also assign an object (or function) to a complex term. For example, any assignment in the interpretation of $\exists f(x)\ R(x, f(x))$ has to assign some object to $f(x)$. The definition of $g[x]h$ can now naturally be extended to a definition of $g[t]h$ for terms t. The definition of $[t]^g_S$ has to be adapted in the natural way to account for function variables.

Just as in the case of DPL semantics, we recursively define an interpretation $[\![\cdot]\!]^g_S$ from DPL formulae to subsets of G_S (the cases 1-5 of the recursive definition are as in Section 1.1):

6. $[\![\varphi \to \psi]\!]^g_S := \{h | h$ differs from g in at most some function variables $f_1, \ldots, f_n$ (where this choice of function variables is maximal), and there is a variable x such that for all $k \in [\![\varphi]\!]^g_S$, there is an assignment $j \in [\![\psi]\!]^k_S$ such that $j(f_i(x)) = h(f_i)(k(x))$ for $1 \leq i \leq n$, and if $n > 0$ then $k[x]g$ $\}$

7. $[\![\exists t\ \varphi]\!]^g_S := \{h |$ there is a k s.t. $k[t]g$ and $h \in [\![\varphi]\!]^k_S\}$

In order to make case 6 of the definition more comprehensible, let us consider its role in determining the semantics of (1), i.e. of $\exists x\ \top \to \exists f(x)\ R(x, f(x))$: First

note that $[\![\exists f(x)\ R(x,f(x))]\!]^k_S$ is the set of assignments j satisfying $R(x,f(x))$ (i.e. for which $[\![R(x,f(x))]\!]^j_S$ is non-empty) such that $j[f(x)]k$. Furthermore note that $[\![\exists x\ \top]\!]^g_S$ is the set of assignments k such that $k[x]g$. So by case 6 with $n=1$,

$$\begin{aligned}[\![\exists x\ \top \rightarrow \exists f(x)\ R(x,f(x))]\!]^g_S &= \{h|h[f]g \text{ and there is a variable } x \text{ such that for all } k \text{ such that } k[x]g, \text{ there is an assignment } j \text{ satisfying } R(x,f(x)) \text{ such that } j[f(x)]k \text{ and } j(f(x)) = h(f)(k(x)), \text{ and } k[x]g\}\\ &= \{h|h[f]g \text{ and for all } k \text{ such that } k[x]g, \text{ there is an assignment } j \text{ satisfying } R(x,f(x)) \text{ such that } j[f(x)]k \text{ and } j(f(x)) = h(f)(k(x))\}\\ &= \{h|h[f]g \text{ and for all } k \text{ such that } k[x]h,\ k \text{ satisfies } R(x,f(x))\}\\ &= [\![\exists f\ (\forall x\ R(x,f(x)))]\!]^g_S\end{aligned}$$

The type restrictions THODPL imposes may be too strict for some applications: Mathematicians sometimes do make use of functions that do not fit into the corset of strict typing, e.g. a function defined on both real numbers and real functions. To overcome this restriction, we will introduce an untyped variant HODPL in Section 6. But for this, we require some foundational preliminaries.

4 Ackermann set theory

Ackermann set theory [1] postulates not only sets, but also proper classes which are not sets.[3] The sets are distinguished from the proper classes by a unary predicate M (from the German word "Menge" for "set").

Ackermann presented a pure version of his theory without urelements, and a separate version with urelements, which we will present here. The language of Ackermann set theory contains three predicates: A binary predicate $\in$, a unary predicate M and a unary predicate U for urelements. We introduce $\mathrm{L}(x)$ ("x is limited") as an abbreviation for $\mathrm{M}(x) \vee \mathrm{U}(x)$. The idea is that sets and urelements are objects of limited size, and are distinguished from the more problematic classes of unlimited size.

The axioms of Ackermann set theory with urelements are as follows:

- *Extensionality axiom:* $\forall x\ \forall y\ (\forall z\ (z \in x \leftrightarrow z \in y) \rightarrow x = y)$

[3]Note, however, that unlike the more well-known class theory NBG, Ackermann set theory also allows for proper classes that contain proper classes.

- *Class comprehension axiom schema:* Given a formula $F(y)$ (possibly with parameters[4]) that does not have x among its free variables, the following is an axiom:
  $\forall y\ (F(y) \to \mathrm{L}(y)) \to \exists x\ \forall y\ (y \in x \leftrightarrow F(y))$
- *Set comprehension axiom schema:* Given a formula $F(y)$ (possibly with parameters that are limited[5]) that does not have x among its free variables and does not contain the symbol M, the following is an axiom:
  $\forall y\ (F(y) \to \mathrm{L}(y)) \to \exists x\ (\mathrm{M}(x) \wedge \forall y\ (y \in x \leftrightarrow F(y)))$
- *Elements and subsets of sets are limited:*
  $\forall x\ \forall y\ (\mathrm{M}(y) \wedge (x \in y \vee \forall z\ (z \in x \to z \in y)) \to \mathrm{L}(y))$

So unlimited set comprehension is replaced by two separate comprehension schemata, one for class comprehension and one for set comprehension. In both cases, the comprehension is restricted by the constraint that only limited objects satisfy the property that we are applying comprehension to. But for set comprehension, we have the additional constraint that the property may not be defined using the setness predicate or using a proper class as parameter. Ackermann justified this approach by appeal to a definition of "set" from Cantor's work [1].

If an Axiom of Foundation for sets is added, Ackermann set theory turns out to be – in what it says about sets – precisely equivalent to ZF [9]. But this equivalence is not a triviality: It is especially hard to establish Replacement for the sets of Ackermann set theory.

5 Ackermann-like function theory

Now we transfer the ideas of a comprehension limited in this way from set comprehension to function comprehension. For this a dichotomy similar to that between sets and classes has to be imposed on functions. We propose the terms *function* and *map* respectively for this dichotomy, and call the theory resulting from these limitations on function comprehension *Ackermann-like Function Theory* (AFT). AFT can be shown to be equiconsistent with Ackermann set theory and hence with ZFC (see Theorem 4 below).

[4] This means that F may actually be of the form $F(\bar{z}, y)$, and that these parameters are universally quantified in the axiom:
$\forall \bar{z}\ (\forall y\ (F(\bar{z}, y) \to \mathrm{M}(y)) \to \exists x\ \forall y\ (y \in x \leftrightarrow F(\bar{z}, y)))$

[5] Formally, with the parameters made explicit, the set comprehension axiom schema reads as follows:
$\forall z_1, \ldots, z_n\ (\mathrm{L}(z_1) \wedge \cdots \wedge \mathrm{L}(z_n) \to (\forall y\ (F(z_1, \ldots, z_n, y) \to \mathrm{L}(y)) \to \exists x\ (\mathrm{M}(x) \wedge \forall y\ (y \in x \leftrightarrow F(z_1, \ldots, z_n, y)))))$

The language of Ackermann-like function theory (L_{AFT}) contains

- a unary predicate F for functions,
- a unary predicate U for urelements,
- a constant symbol u for undefinedness, and
- a binary function symbol a for function application.

Instead of $a(f,t)$ we usually simply write $f(t)$. We write $\mathrm{L}(x)$ instead of $\mathrm{U}(x) \vee \mathrm{F}(x)$. The undefinedness constant u is needed for formalising the idea that a function is only defined for certain values and undefined for others. In this language, the unrestricted function comprehension schema would be as follows:

Axiom Schema 2 (Unrestricted function comprehension in L_{AFT})**.** Given a variable z and formulae $P(z)$ and $R(z,x)$ (possibly with parameters), the following is an axiom: $\forall z\ (P(z) \rightarrow \exists x\ R(z,x)) \rightarrow \exists f\ (\neg \mathrm{U}(f) \wedge \forall z\ ((P(z) \rightarrow R(z,f(z))) \wedge (\neg P(z) \rightarrow f(z) = u)))$

Analogously to the case of Ackermann set theory, AFT has separate comprehension schemata for maps and functions. The restriction that is imposed on both schemata now is $\forall z\ \forall x\ (R(z,x) \rightarrow \mathrm{L}(z) \wedge \mathrm{L}(x))$. In the function comprehension schema, in which $\mathrm{F}(f)$ appears among the conclusions we may draw about f, the additional restriction is that the formula $R(z,x)$ may not contain the symbol F and may not have unlimited objects as parameters.

Additionally to these comprehension schemata, AFT has

- a function extensionality axiom,
- an axiom stating that any value a function takes and any value a function is defined at is limited, and
- an axiom stating that submaps of functions are functions.

In AFT one can interpret Ackermann set theory with Foundation, and hence ZFC (see Theorems 1 and 3 below). Since the map and function comprehension schemata presuppose the existence of choice maps and choice functions, the Axiom of Choice naturally comes out true in these interpretations.

We now state the main theorems about AFT. Their proofs can be found in the author's PhD thesis [5, pp. 58-62].

Theorem 1 (Theorem 4.2.7 in [5, p. 58])**.** *AFT interprets Ackerman set theory with urelements and the Axiom of Choice.*

Theorem 2 (Theorem 4.2.20 in [5, p. 61])**.** *Ackermann set theory with the Axiom of Foundation and the Axiom of Global Choice interprets AFT.*

Theorem 3 (Theorem 4.2.8 in [5, p. 59])**.** *AFT interprets ZFC.*

Theorem 4 (Corollary in [5, p. 62])**.** *AFT is equiconsistent with ZFC.*

6 Higher-order dynamic predicate logic

Now we are ready to sketch the untyped *Higher-Order Dynamic Predicate Logic* (HODPL). The restriction we impose on implicit function introduction are those imposed by AFT. AFT gives us untyped maps, which always have a restricted domain. So instead of using types to syntactically restrict the possible arguments for a given function term, we implement a semantic restriction on function application by integrating a formal account of presuppositions into the HODPL.[6] HODPL syntax thus allows for any term to be applied to any number of arguments to form a new term.

Besides the binary "=", HODPL has two unary logical relation symbols, U for urelements and F for functions. HODPL syntax does not depend on a signature, as we do not allow for constant, function and relation symbols other than "=", U and F. These can be mimicked by variables that respectively denote a non-function, denote a normal function or denote a function that only takes two predesignated urelements ("booleans") as values.

The domain of a structure always has to be a model of AFT. The possibility of presupposition failure is implemented in HODPL semantics by making the interpretation function partial rather than total. For conveniently talking about partial functions, we use the notation $\mathrm{def}(f(x))$ to abbreviate that f is defined on x.

We define the partial interpretation function $[\![\cdot]\!]^g_S \subseteq G_S \times G_S$ by specifying its domain and its values trough a simultaneous recursion (the cases 3-8 of the second part are as in THODPL):

- Domain of $[\![\cdot]\!]^g_S$:
  1. $\mathrm{def}([\![\mathrm{U}(t)]\!]^g_S)$ iff $[t]^g_S \neq u^S$.
  2. $\mathrm{def}([\![\mathrm{F}(t)]\!]^g_S)$ iff $[t]^g_S \neq u^S$.
  3. $\mathrm{def}([\![\top]\!]^g_S)$.
  4. $\mathrm{def}([\![t_1 = t_2]\!]^g_S)$ iff $[t_1]^g_S \neq u^S$ and $[t_2]^g_S \neq u^S$.

[6] See [4] for an introduction to presuppositions in mathematical texts.

5. $\text{def}([\![\neg\varphi]\!]^g_S)$ iff $\text{def}([\![\varphi]\!]^g_S)$.
6. $\text{def}([\![\varphi \wedge \psi]\!]^g_S)$ iff $\text{def}([\![\varphi]\!]^g_S)$ and for all $h \in [\![\varphi]\!]^g_S$, $\text{def}([\![\psi]\!]^h_S)$.
7. $\text{def}([\![\varphi \rightarrow \psi]\!]^g_S)$ iff $\text{def}([\![\varphi]\!]^g_S)$ and for all $h \in [\![\varphi]\!]^g_S$, $\text{def}([\![\psi]\!]^h_S)$.
8. $\text{def}([\![\exists t\ \varphi]\!]^g_S)$ iff for all h s.t. $h[t]g$, $\text{def}([\![\varphi]\!]^h_S)$.

- Values of $[\![\cdot]\!]^g_S$:
  1. $[\![\mathrm{U}(t)]\!]^g_S := \{h | g = h \text{ and } [t]^g_S \in \mathrm{U}^S\}$
  2. $[\![\mathrm{F}(t)]\!]^g_S := \{h | g = h \text{ and } [t]^g_S \in \mathrm{F}^S\}$

One can define a sound proof system for HODPL that can prove everything provable in AFT: In the author's PhD thesis, a proof system for an extension of HODPL is defined [5, pp. 108-113] and proven to be sound [5, pp. 147-148] and complete [5, pp. 156-176]. The details of this proof system are beyond the scope of this paper.

7 Conclusion

We have studied a feature of the natural language of mathematics that has previously not been studied by other logicians or linguists, the *implicit dynamic function introduction*, exemplified by constructs of the form "for every x there is an $f(x)$ such that ...". If this feature is used without limitations, it yields a paradox analogous to Russell's paradox. Hence any formalism capturing it has to impose some limitations on it. We have sketched two higher-order extensions of Dynamic Predicate Logic, *Typed Higher-Order Dynamic Predicate Logic* (THODPL) and *Higher-Order Dynamic Predicate Logic* (HODPL), which capture this feature, and which differ only in the limitations they impose onto it. HODPL is based on *Ackermann-like Function Theory*, a novel foundational theory of functions that is inspired by Ackermann Set Theory and that interprets ZFC.

References

[1] W. Ackermann. Zur Axiomatik der Mengenlehre. *Mathematische Annalen*, 131:336–345, 1956.

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

[3] M. Cramer, B. Fisseni, P. Koepke, D. Kühlwein, B. Schröder, and J. Veldman. The Naproche Project – Controlled Natural Language Proof Checking of Mathematical Texts. In *CNL 2009 Workshop, LNAI 5972*, pages 170–186, 2010.

[4] M. Cramer, D. Kühlwein, and B. Schröder. Presupposition Projection and Accommodation in Mathematical Texts. In *Semantic Approaches in Natural Language Processing: Proceedings of the Conference on Natural Language Processing 2010 (KONVENS)*, pages 29–36. Universaar, 2010.

[5] Marcos Cramer. *Proof-checking mathematical texts in controlled natural language*. PhD thesis, Universität Bonn, 2013.

[6] Marcos Cramer. The Naproche system: Proof-checking mathematical texts in controlled natural language. *Sprache und Datenverarbeitung – International Journal for Language Data Processing*, 38(1-2):9–33, 2014.

[7] J. Groenendijk and M. Stokhof. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14(1):39–100, 1991.

[8] M. Lackenby. Topology and Groups. Retrieved December 1, 2016, from `http://people.maths.ox.ac.uk/lackenby/tg050908.pdf`, 2008.

[9] W. Reinhardt. Ackermann's set theory equals ZF. *Annals of Mathematical Logic*, 2:189–249, 1970.

[10] W. Trench. *Introduction to Real Analysis*. Prentice Hall, 2003.

Received \jreceived