Modelling implicit dynamic introduction of function symbols in mathematical texts

Marcos Cramer

University of Luxembourg
marcos.cramer@uni.lu

Abstract

The specialized language of mathematics has a number of linguistically and logically interesting features. One of them, which to our knowledge has not been systematically studied before, is the *implicit dynamic introduction of function symbols*, exemplified by constructs of the form "for every x there is an $f(x)$ such that ...". We present an extension of Groenendijk and Stokhof's Dynamic Predicate Logic – *Typed Higher-Order Dynamic Predicate Logic* – which formally models this feature of the language of mathematics. Furthermore, we illustrate how the implicit dynamic introduction of function symbols is treated in the proof checking algorithm of the Naproche system.

dynamic quantification, Dynamic Predicate Logic, function introduction, language of mathematics, formal mathematics

1 Introduction

Like other sciences, mathematics has developed its own specialized language, which we call the *language of mathematics*. This specialized language has a number of linguistically and logically interesting features: For example, on the syntactic level, it can incorporate complex symbolic material into natural language sentences. On the semantic level, it refers to rigorously defined abstract objects, and is in general less open to ambiguity than most other text types. On the pragmatic level, it reverses the expectation on assertions, which have to be implied by the context rather than adding new information to it.

The work presented in this paper has been conducted in the context of the *Naproche project*, an interdisciplinary project at the universities of Bonn and Duisburg-Essen which analyses the language of mathematics with methods from mathematical logic and computational and formal linguistics (see section 1.4 of [5]). The main aim of the Naproche project has been to develop a *controlled natural language* (*CNL*) – i.e. a subset of a natural language defined through a formal grammar – for writing mathematical texts, and a computer program, the *Naproche system*, that can check the correctness of mathematical proofs written in this CNL. For the development of this CNL and this system, we had to develop new linguistic and logical machinery, which is thoroughly discussed in the author's PhD thesis [5]. In this paper we focus on one particular phenomenon of the language of mathematics, the *implicit dynamic introduction of function* *symbols*, and show how it can be modelled in a formal system. To our knowledge, this phenomenon has not been previously described in the literature or formally modelled in a formalism.[1]

Since this phenomenon is a special case of the phenomenon of dynamic quantification, we first briefly discuss this well-known phenomenon and one of the standard solutions to it, Groenendijk and Stokhof's *Dynamic Predicate Logic*.

2 Dynamic quantification and *Dynamic Predicate Logic*

When translating natural language sentences to standard first-order formulae, there is a problem in the treatment of natural language quantifiers. For example, the indefinite article *a* is normally translated by an existential quantifier, but the natural translation (

Because of this difference between the functioning of quantifiers in natural language and in standard first-order logic, formal linguists say that natural language has *dynamic quantifiers*, whereas standard first-order logic has *static quantifiers*. Jeroen Groenendijk and Martin Stokhof have developed a variant of first-order logic called *Dynamic Predicate Logic* (*DPL*) [7] which has dynamic instead of static quantifiers. In DPL,

The natural language quantification used in mathematical texts also exhibits these dynamic features, as can be seen in the following quotation from [8, p. 36]:

> If a space X retracts onto a subspace A, then the homomorphism $i_* : \pi_1(A, x_0) \to \pi_1(X, x_0)$ induced by the inclusion $i : A \hookrightarrow X$ is injective.

2.1 DPL semantics

We present DPL semantics in a way slightly different but logically equivalent to its definition in [7]. Structures and assignments are defined as for standard first-order logic: 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 $|S|$. G_S is 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

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

[1]The only exception being the author's paper [4], which sketched part of the material presented in this paper.

In [7], DPL semantics is defined via an interpretation function $[\![\bullet]\!]_S$ from DPL formulae to subsets of $G_S \times G_S$. We instead recursively define for every $g \in G_S$ an interpretation function $[\![\bullet]\!]^g_S$ from DPL formulae to subsets of G_S:[2]

1. $[\![\top]\!]^g_S := \{g\}$
2. $[\![t_1 = t_2]\!]^g_S := \{h|h = g \text{ and } [t_1]^g_S = [t_2]^g_S\}$[3]
3. $[\![R(t_1, \dots, t_2)]\!]^g_S := \{h|h = g \text{ and } ([t_1]^g_S, \dots, [t_2]^g_S) \in R^S\}$
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.

3 Implicit dynamic function introduction

Functions are often dynamically introduced in an implicit way in mathematical texts. For example, [10, p. 1] introduces the additive inverse function on the reals as follows:

(1) For each a there is a real number $-a$ such that $a + (-a) = 0$.

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:

(2) 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 (or a *map*), the standard notation with a bracketed argument is preferred:

(3) Hence for each $u \in \mathbf{R}^n$ there is a number $f(u) \in \mathbf{C}$ with $f(u) \neq 0$ such that $(\sigma(\alpha(u))^3, \sigma(\alpha(u))\ \Sigma(\alpha(u)), T(\alpha(u))) = f(u)(x_1(u), x_2(u), x_3(u))$.
The function f is locally a quotient of continuous functions, so it is itself continuous. [2, p. 489]

(4) Suppose that, for each vertex v of K, there is a vertex $g(v)$ of L such that $f(st_K(v)) \subset st_L(g(v))$. Then g is a simplicial map $V(K) \rightarrow V(L)$, and $|g| \simeq f$. [9, p. 19]

[2]This can be viewed as a different currying of the uncurried version of the interpretation function in [7].

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

(5) Since the multi-map Φ^{-1} is surjective, for every $x \in X$ there is a point $f(x) \in Y$ with $x \in \Phi^{-1}(f(x))$, which is equivalent to $f(x) \in \Phi(x)$. It follows from the bornologity of Φ that the map $f : X \to Y$ is bornologous. [1, p. 5]

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 hypothesize that the naturalness of such implicit function introduction in mathematical texts contributes to the wide-spread feeling that the Axiom of Choice must be true.

Implicitly introduced functions are generally *partial functions*, i.e. they have a restricted domain and are not defined on the whole universe of the discourse. For example in (

Implicit function introduction can also be used to introduce multi-argument functions. For example, subtraction on the reals could be introduced by a sentence of the following form:

(6) For all reals a, b, there is a real $a - b$ such that $(a - b) + b = a$.

For the sake of simplicity and brevity, we restrict ourselves to unary functions for the rest of this paper. See section 5.1 of [5] for an account of how to extend the presented formalization of implicit dynamic function introduction to multi-argument and curried functions.

4 Typed Higher-Order Dynamic Predicate Logic

Typed Higher-Order Dynamic Predicate Logic (*THODPL*) extends *DPL* to a higher-order system that formalizes the implicit dynamic introduction of function symbols.

The type system used in *THODPL* is Church's Simple Type Theory [3] with two base types i (for objects) and o (for propositions), and a function type $T_1 \to T_2$ for any two types T_1, T_2. We assume a countably infinite supply of variables and constants of each type. In the examples below we use x and y as variables of the basic type i, f as a variable of the function type $i \to i$ and R as a constant of type $i \to (i \to o)$.

The distinctive feature of *THODPL* syntax is that it allows not only variables but any well-formed terms to come after quantifiers. Writing t_T for a well-formed term of type T, we can define *THODPL* formulae as follows:

$$\varphi ::= t_o | t_T = t_T | \mathrm{def}(t_T) | \top | \exists t_T\ \varphi | \neg\varphi | \varphi\varphi | \varphi \to \varphi$$

The intended meaning of $\mathrm{def}(t_T)$ is that t_T is a defined term. This is needed because *THODPL* allows for partial functions and hence for undefined terms.

Similarly as in DPL, $\varphi \vee \psi$ and $\forall t\ \varphi$ are defined to be a shorthand for $\neg(\neg\varphi \wedge \neg\psi)$ and $\exists t\ \top \to \varphi$ respectively. Instead of $R(t_1)(t_2)$, we also write $R(t_1, t_2)$ in uncurried notation.

Since terms can come after quantifiers, (

4.1 *THODPL* semantics

Since implicitly introduced functions can be partial, *THODPL* does not assume function variables to refer to total functions. Partial functions can give rise to undefined terms, so we need to handle these in *THODPL* semantics. This is done by extending the domain of discourse by an object u used as the value of undefined terms. This motivates the following two definitions:

Let D be a non-empty set. Fix some $u \notin D$. Then we define D_T for every type T inductively as follows:
$D_i := D$
$D_o := \{\top, \bot\}$
$D_{T_1 \to T_2} := (D_{T_2} \cup \{u\})^{D_{T_1}}$

A *THODPL structure* is a pair $S = (D, F)$, where D is a non-empty set called the *domain* of S and F is a map that assigns to every constant of type T an element of D_T.

In order to handle quantifiers followed by complex terms, assignments in *THODPL* can not only assign values to variables, but also to complex terms:

Given a *THODPL* structure $S = (D, F)$, an S-assignment is a partial function g from *THODPL* terms to $\bigcup_T D_T$ satisfying the following two properties:

- g is defined on all variables.
- For every type T and every term t of type T, if $g(t)$ is defined, then $g(t) \in D_T$.

We denote the set of S-assignments by G_S.

Given two assignments g and h, we define $g[t_1, \ldots, t_n]h$ to mean that $dom(g) = dom(h) \cup \{t_1, \ldots, t_n\}$ and for all $s \in dom(h) \setminus \{t_1, \ldots, t_n\}$, $g(s) = h(s)$.

Now we are ready to present the semantics of *THODPL* in two definitions, one for *THODPL* terms and one for *THODPL* formulae:

Given a *THODPL* structure $S = (D, F)$ and an S-assignment g, we recursively define $[t]^g_S$ for *THODPL* terms t as follows:

$$[t]^g_S := \begin{cases} g(t) & \text{if } t \text{ is a variable,} \\ F(t) & \text{if } t \text{ is a constant symbol,} \\ [t_0]^g_S([t_1]^g_S) & \text{if } t \text{ is of the form } t_0(t_1) \text{ and } [t_i]^g_S \neq u \text{ for } i \in \{1,2\}, \\ u & \text{if } t \text{ is of the form } t_0(t_1) \text{ and } [t_i]^g_S = u \text{ for some } i \in \{1,2\}. \end{cases}$$

Given a *THODPL* structure $S = (D, F)$ and an S-assignment g, we recursively define for every $g \in G_S$ an interpretation function $[\![\bullet]\!]^g_S$ from *THODPL* formulae to subsets of G_S:

1. $[\![t]\!]^g_S := \begin{cases} \{g\} & \text{if } [t]^g_S = \top^S \\ \emptyset & \text{otherwise} \end{cases}$

2. $[\![t_1 = t_2]\!]^g_S := \begin{cases} \{g\} & \text{if } [t_1]^g_S = [t_2]^g_S \\ \emptyset & \text{otherwise} \end{cases}$
3. $[\![\text{def}(t)]\!]^g_S := \begin{cases} \{g\} & \text{if } [t]^g_S \neq u \\ \emptyset & \text{otherwise} \end{cases}$
4. $[\![\top]\!]^g_S := \{g\}$
5. $[\![\exists t\ \varphi]\!]^g_S := \{h \mid \text{there is a } k \text{ such that } k[t]g \text{ and } h \in [\![\varphi]\!]^k_S\}$
6. $[\![\neg\varphi]\!]^g_S := \begin{cases} \{g\} & \text{if there is no } h \text{ such that } h \in [\![\varphi]\!]^g_S \\ \emptyset & \text{otherwise} \end{cases}$
7. $[\![\varphi\psi]\!]^g_S := \{h \mid \text{there is a } k \text{ such that } k \in [\![\varphi]\!]^g_S \text{ and } h \in [\![\psi]\!]^k_S\}$
8. $[\![\varphi \to \psi]\!]^g_S := \{h|h$ 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 $k[x]g$ $\}$

In order to make case

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

The truth condition of a formula φ under (S, g) is determined by $[\![\varphi]\!]^g_S$ being empty or non-empty (with emptiness corresponding to falsehood). So the claim made above that (

$[\![\exists x\ \top \to \exists f(x)\ R(x, f(x))]\!]^g_S \neq \emptyset$

iff $\{h \mid h[f]g$ and for all k such that $k \in [\![\exists x\ \top]\!]^g_S$, there is an assignment $j \in [\![\exists f(x)\ R(x, f(x))]\!]^g_S$ such that $j(f(x)) = h(f)(k(x))\} \neq \emptyset$

iff $\{h \mid h[f]g$ and for all k such that $k[x]g$, there is an assignment j such that $j[f(x)]k$, $j \in [\![R(x, f(x))]\!]^g_S$ and $j(f(x)) = h(f)(k(x))\} \neq \emptyset$

iff $\{h \mid h[f]g$ and for all k such that $k[x]g$, $F(R)(k(x), h(f)(k(x))) = \top\} \neq \emptyset$ (where we write $S = (D, F)$ again)

iff there is a function $\bar{f}$ such that for all k such that $k[x]g$, $F(R)(k(x), \bar{f}(k(x))) = \top$

iff for all k such that $k[x]g$, there is a $\bar{y}$ such that $F(R)(k(x), \bar{y}) = \top$ (the right-to-left implication follows from the Axiom of Choice)

iff $[\![\exists x \;\; \top \rightarrow \exists y \; R(x, y)]\!]^g_S \neq \emptyset$.

5 Implicit dynamic function introduction and proof-checking

The Naproche system mentioned in the introduction can check mathematical texts written in a controlled natural language. It is interesting to see how implicit dynamic function introduction can be used to introduce functions to the discourse without having to explicitly prove their existence. For this, we first briefly sketch the general working of the proof-checking implemented in the Naproche system.

For checking single proof steps, the Naproche system makes use of state-of-the-art *automated theorem provers* (ATPs) for standard first-order logic. Given a set of *premises*[4] Γ and a *conjecture* φ, an ATP tries to find either a proof that the Γ logically implies φ, or build a model for $\Gamma \cup \{\neg\varphi\}$, which shows that they do not imply it. A conjecture together with a set of premises handed to an ATP is called a *proof obligation.* We denote the proof obligation consisting of premises Γ and conjecture φ by $\Gamma \vdash^? \varphi$.

The proof checking algorithm keeps track of a list of first-order premises considered to be true. This list gets updated continuously during the checking process. Each assertion is checked by an ATP based on the currently active premises.

We illustrate the functioning of the proof checking algorithm on an example. Suppose that the Naproche system has to check the text in (

The Naproche system does not check the Naproche CNL input directly, but first translates it into a semantic representation format which is an extension of *THODPL.* The proof checking algorithm is then defined on input in this extension of *THODPL* (see chapter 6 of [5]).

In order to show how the proof checking works in the context of implicitly introduced function symbols, we reconsider the example sentence (

So when checking (

[4]In the ATP community, the term "axiom" is usually used for what we call "premise" here; the reason for our deviation from the standard terminology is that in the context of our work the word "axiom" means a completely different thing, namely an axiom stated inside a mathematical text that is to be checked by the Naproche system. The premises that we are considering here can originate either from axioms, from definitions, from assumptions or from previously proved results.

6 Conclusion

The phenomenon of implicit dynamic introduction of function symbols discussed in this paper is interesting both from a theoretical and from a practical perspective:

From a theoretical perspective, it is an interesting case of dynamic quantification which needs to be taken into consideration if one wants to give a full account of the nature of quantification in natural language (at least if one accepts specialized languages like the language of mathematics as instances of natural language).

From a practical point of view, developers of a formal mathematics computer system can make their system more easily usable by allowing for implicit dynamic introduction of function symbols in the input language and treating it similarly as described in section

To our knowledge, the phenomenon of implicit dynamic introduction of function symbols has not been previously described in the linguistic or logical literature. In this paper, we have presented a formalization of this phenomenon in a higher-order extension of Dynamic Predicate Logic, and have illustrated its functioning in the proof checking algorithm of the Naproche system.

References

[1] Banakh, T., Zarichnyy, I.: The coarse classification of homogeneous ultra-metric spaces. Preprint (2008), arXiv:0801.2132

[2] Bonk, M.: On the second part of Hilbert's fifth problem. Mathematische Zeitschrift 210(1) (1992)

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

[4] Cramer, M.: Implicit dynamic function introduction and its connections to the foundations of mathematics. In: Prosorov, O. (ed.) Proceedings of the International interdisciplinary conference on Philosophy, Mathematics, Linguistics: Aspects of Interaction (PhML 2012). pp. 35–42 (2012)

[5] Cramer, M.: Proof-checking mathematical texts in controlled natural language. Ph.D. thesis, University of Bonn (2013)

[6] Geach, P.T.: Reference and Generality. An Examination of Some Medieval and Modern Theories. Cornell University Press, Ithaca, NY (1962)

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

[8] Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)

[9] Lackenby, M.: Topology and Groups (2008), `http://people.maths.ox.ac.uk/lackenby/tg050908.pdf`

[10] Trench, W.: Introduction to Real Analysis. Prentice Hall, Upper Saddle River, NJ (2003)