# A Set Theory with Support for Partial Functions *

William M. Farmer (wmfarmer@mcmaster.ca)
*Department of Computing and Software*
*McMaster University*
*1280 Main Street West*
*Hamilton, Ontario L8S 4L7, Canada*

Joshua D. Guttman (guttman@mitre.org)
*The MITRE Corporation*
*202 Burlington Road*
*Bedford, MA 01730-1420, USA*

**Abstract.** Partial functions can be easily represented in set theory as certain sets of ordered pairs. However, classical set theory provides no special machinery for reasoning about partial functions. For instance, there is no direct way of handling the application of a function to an argument outside its domain as in partial logic. There is also no utilization of lambda-notation and sorts or types as in type theory. This paper introduces a version of von-Neumann-Bernays-Gödel set theory for reasoning about sets, proper classes, and partial functions represented as classes of ordered pairs. The underlying logic of the system is a partial first-order logic, so class-valued terms may be nondenoting. Functions can be specified using lambda-notation, and reasoning about the application of functions to arguments is facilitated using sorts similar to those employed in the logic of the IMPS Interactive Mathematical Proof System. The set theory is intended to serve as a foundation for mechanized mathematics systems.

**Keywords**: Set theory, NBG, mechanized mathematics, theorem proving systems, partial functions, undefinedness, sorts.

## 1. Introduction

Set theory is, at least among mathematicians, the most popular foundation for mathematics. The reasons for its popularity are readily apparent. It is based on two of the simplest and most familiar notions in mathematics: set and membership. Since nearly all mathematical concepts can be expressed in terms of set and membership, it is extremely expressive. And there is a first-order formalization of set theory, Zermelo-Fraenkel (ZF) set theory, which is widely used as a *lingua franca* for mathematics.

* Supported by the MITRE-Sponsored Research program. Published in *Partiality and Modality*, eds., E. Thijsse, F. Lepage, and H. Wansing, special issue of *Logica Studia*, 66:59–78, 2000.

On the other hand, set theory has not been a popular foundation for *mechanized* mathematics. A *mechanized mathematics system (MMS)* is a computer environment that is intended to support and improve rigorous mathematical activity. MMSs include mechanical theorem provers and systems for specifying and verifying computer software and hardware. Contemporary MMSs are based on a wide range of both special- and general-purpose logical formalisms. Weak logics such as first-order logic and its various sublogics are often chosen for ease of implementation, while type theories are often chosen for ease of use. Few MMSs are based on standard formalizations of set theory. The exceptions include:

1. The EVES program verification system [5] based on ZF.
2. M. Gordon's augmentation of HOL with ZF axioms [15].
3. N. Megill's Metamath proof verifier [18] based on ZF.
4. The Mizar proof development system [27] based on Tarski-Grothendieck set theory.
5. L. Paulson's implementation of ZF [24] in the Isabelle generic theorem prover [22].
6. A. Quaife's formalization of von-Neumann-Bernays-Gödel set theory [25] in the Otter resolution theorem prover [17].

The notions of function and application are at least as basic and important in mathematics as the notions of set and membership. A major issue in the design of an MMS is the kind of support the system provides for reasoning about functions. The main deficiency of first-order set theory as a logical basis for an MMS is certainly its lack of support for functions. The logical machinery—operator symbols and operator application—does not support higher-order or partial functions, quantification over functions, or function abstraction. Although (higher-order and partial) functions can be easily represented as certain sets of ordered pairs, the set-theoretic machinery does not support the direct manipulation of terms that denote functions. For example, there is no built-in mechanism for directly applying a term denoting a function to a term denoting an argument to the function to form a new term; applications can only be represented verbosely with the use of quantifiers or by introducing a special operator symbol that denotes function application. There is also no facility for keeping track of when an application of a function is defined and what kind of value an application of a function may have when it is defined.

In contrast to set theory, type theory provides special machinery for reasoning about higher-order functions which includes term constructors for function application and abstraction, and syntactic types for managing the application of functions. This machinery is effective for reasoning about *total* functions, but it usually can only be used to reason about *partial* functions in indirect and artificial ways. LUTINS[1] [6, 7, 8], a version of simple type theory with partial functions, undefined terms, and subtypes called *sorts*, is exceptional in this respect. As the logic of the IMPS Interactive Mathematical Proof System [11, 12], it has proved to be highly effective for specifying and reasoning about partial functions.

Our objective is to devise a formalization of set theory that would be a suitable basis for mechanized mathematics. In line with this primary goal, we would like the formalization to satisfy the following secondary goals:

- The underlying logic should be based on the principles of classical predicate logic.
- The set theory should be in the tradition of ZF.
- There should be strong support for reasoning about partial functions.
- The formalization should be amenable to implementation in an MMS.

The paper presents a version of von-Neumann-Bernays-Gödel (NBG) set theory called NBG*. NBG is a well-known (first-order) set theory in which variables range over both sets and proper classes. This means that the universe of sets $V$ can be defined as an individual constant in NBG even though it is a proper class. Also functions from $V$ to $V$, such as the cardinality function, are first-class objects in NBG even when they are proper classes. (A good introduction to NBG is found in [19].)

NBG is closely related to ZF. NBG and ZF share the same intuitive model of the iterated hierarchy of sets. The nonlogical axioms of NBG are very similar to those of ZF; most of them are simply ZF axioms with some of the quantifiers restricted to sets. And there is a faithful interpretation of ZF in NBG [23, 26, 29], which implies that ZF is consistent iff NBG is consistent. However, NBG is finitely axiomatizable ([14] and [19] present finite axiomatizations of NBG), while ZF is not (see [16] or [21] for a proof).

NBG* is derived as follows from NBG. First, the underlying logic of NBG, ordinary first-order logic, is replaced with Partial First-Order

[1] Pronounced as the word in French.

Logic (PFOL), a variant of first-order logic in which operator symbols may denote partial functions and terms may be undefined. The nonlogical axioms of NBG* are essentially the same as the nonlogical axioms of the axiomatization of NBG given by K. Gödel in [14]. Next, term constructors for function application and abstraction—like those used in most type theories—are defined using the definite description operator of PFOL. Finally, a system of sorts—similar to the system of sorts in LUTINS—is added for classifying terms.

The paper [10] presents a directly defined version of NBG* called STMM and discusses the benefits STMM offers as a foundation for mechanized mathematics.

The rest of the paper is organized as follows. Section 2 contains the syntax and semantics of PFOL and a theorem that shows that PFOL extends but does not alter the conceptual framework of classical first-order logic. A Hilbert-style axiomatic system for PFOL which is complete with respect to the semantics of PFOL is given at the end of this section. Section 3 presents an axiomatization of NBG in the form of a PFOL theory called Partial NBG (PNBG) and defines term constructors for function application and abstraction. Section 4 introduces the notion of a sort and defines NBG* as a combination of PNBG and a system of sorts. The paper ends with a short conclusion in section 5.

## 2. Partial First-Order Logic

This section presents a variant of first-order logic called *Partial First-Order Logic* (PFOL) in which operator symbols may denote partial functions and terms may be undefined. It is a formalization of what we call the *traditional approach to partial functions* [9]. PFOL is similar to the partial logics proposed by R. Schock [28], T. Burge [3, 4], M. Beeson [1, 2], and L. Monk [20].

### 2.1. Syntax

A *variable* of PFOL is a member of a fixed infinite set $\mathcal{V}$ of symbols. A *language* of PFOL is a tuple $(\mathcal{C}, \mathcal{O}, \mathcal{P})$ such that:

1. $\mathcal{C}$ is a set of *individual constants*.
2. $\mathcal{O}$ is a set of *operator symbols*, each with an assigned arity $\geq 1$.
3. $\mathcal{P}$ is a set of *predicate symbols*, each with an assigned arity $\geq 1$. $\mathcal{P}$ contains the binary predicate symbol $=$.
4. $\mathcal{V}$, $\mathcal{C}$, $\mathcal{O}$, and $\mathcal{P}$ are pairwise disjoint.

In the remainder of this section, let $\mathcal{L} = (\mathcal{C}, \mathcal{O}, \mathcal{P})$ be a language of PFOL.

A *term* and a *formula* of $\mathcal{L}$ are defined inductively by:

1. Each $x \in \mathcal{V}$ and $a \in \mathcal{C}$ is a term.
2. If $x \in \mathcal{V}$ and $\varphi$ is a formula, then $(\mathrm{I}\, x\, .\, \varphi)$ is a term.
3. If $o \in \mathcal{O}$ is $n$-ary and $t_1, \ldots, t_n$ are terms, then $o(t_1, \ldots, t_n)$ is a term.
4. If $p \in \mathcal{P}$ is $n$-ary and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a formula.
5. If $\varphi$ and $\psi$ are formulas and $x \in \mathcal{V}$, then $\neg\varphi$ and $(\varphi \supset \psi)$, and $(\forall\, x\, .\, \varphi)$ are formulas.

The symbols $\neg$, $\supset$, $\forall$, and I are the *logical constants* of PFOL. The first three are operators for negation, implication, and universal quantification. The last one, I, is a definite description operator. A term or formula is *regular* if it does not contain any occurrences of I. Regular terms and formulas are the same as the terms and formulas of ordinary first-order logic.

Parentheses in terms and formulas may be suppressed when meaning is not lost, and sometimes we use the symbols [ and ] in place of parentheses. For convenience, we also employ the following abbreviations:

| | | |
|---|---|---|
| $(s = t)$ | for | $= (s, t)$. |
| $(s \neq t)$ | for | $\neg(s = t)$. |
| $(\varphi \wedge \psi)$ | for | $\neg(\varphi \supset \neg\psi)$. |
| $(\varphi \vee \psi)$ | for | $\neg\varphi \supset \psi$. |
| $(\varphi \equiv \psi)$ | for | $(\varphi \supset \psi) \wedge (\psi \supset \varphi)$. |
| $(\exists\, x\, .\, \varphi)$ | for | $\neg(\forall\, x\, .\, \neg\varphi)$. |
| $(t\!\downarrow)$ | for | $\exists\, x\, .\, x = t$ where $x \in \mathcal{V}$ and $x$ does not occur in $t$. |
| $(t\!\uparrow)$ | for | $\neg(t\!\downarrow)$. |
| $(s \simeq t)$ | for | $(s\!\downarrow \vee t\!\downarrow) \supset s = t$. |
| $\bot$ | for | $\mathrm{I}\, x\, .\, x \neq x$ where $x \in \mathcal{V}$. |
| $\mathsf{if}(\varphi, s, t)$ | for | $\mathrm{I}\, x\, .\, (\varphi \supset x = s) \wedge (\neg\varphi \supset x = t)$ where $x \in \mathcal{V}$ and $x$ does not occur in $\varphi$, $s$, or $t$. |

$t\!\downarrow$, $t\!\uparrow$, and $s \simeq t$ are read as "$t$ is defined", "$t$ is undefined", and "$s$ and $t$ are quasi-equal". $\bot$ is a canonical undefined term, and $\mathsf{if}$ is an if-then-else term constructor.

Let an *expression* of $\mathcal{L}$ be either a term or formula of $\mathcal{L}$. "Free variable", "closed", and similar notions are defined in the obvious way.

A *sentence* is a closed formula. Given a formula $\varphi$, $\varphi[x \mapsto t]$ is the result of simultaneously replacing each free occurrence of the variable $x$ in $\varphi$ with the term $t$.

A *theory* of PFOL is a pair $\mathcal{T} = (\mathcal{L}, \Gamma)$ where $\mathcal{L}$ is a language of PFOL and $\Gamma$ is a set of sentences of $\mathcal{L}$. $\mathcal{T}$ is said to be *over* $\mathcal{L}$ and the members of $\Gamma$ are called the *nonlogical axioms* of $\mathcal{T}$. ($\Gamma$ need not be closed under logical consequence.) $\mathcal{T}$ is *regular* if each sentence in $\Gamma$ is regular. Regular theories are the same as the theories of ordinary first-order logic.

## 2.2. Semantics

A *model* for $\mathcal{L}$ is a pair $\mathcal{M} = (\mathcal{D}, I)$ where $\mathcal{D}$ is a nonempty domain (set) and $I$ is a total function on $\mathcal{C} \cup \mathcal{O} \cup \mathcal{P}$ such that:

1. If $a \in \mathcal{C}$, $I(a) \in \mathcal{D}$.
2. If $o \in \mathcal{O}$ is $n$-ary, $I(o)$ is a partial function from $\mathcal{D} \times \cdots \times \mathcal{D}$ ($n$ times) to $\mathcal{D}$.
3. If $p \in \mathcal{P}$ is $n$-ary, $I(p)$ is a total function from $\mathcal{D} \times \cdots \times \mathcal{D}$ ($n$ times) to $\{\mathrm{T}, \mathrm{F}\}$ (the domain of truth values). $I(=)$ is the identity relation on $\mathcal{D}$.

$\mathcal{M}$ is *regular* if $I(o)$ is a *total* function for each $o \in \mathcal{O}$. Regular models are the same as the models of ordinary first-order logic.

Let $\mathcal{M} = (\mathcal{D}, I)$ be a model for $\mathcal{L}$. A *variable assignment* into $\mathcal{M}$ is a function which maps each $x \in \mathcal{V}$ to an element of $\mathcal{D}$. Given a variable assignment $A$ into $\mathcal{M}$, $x \in \mathcal{V}$, and $d \in \mathcal{D}$, let $A[x \mapsto d]$ be the variable assignment $A'$ into $\mathcal{M}$ such $A'(x) = d$ and $A'(y) = A(y)$ for all $y \neq x$.

Define $V = V^{\mathcal{M}}$ to be the binary function such that the following conditions are satisfied for all variable assignments $A$ into $\mathcal{M}$ and all expressions of $\mathcal{L}$:

1. If $t \in \mathcal{V}$, then $V_A(t) = A(t)$.
2. If $t \in \mathcal{C}$, then $V_A(t) = I(t)$.
3. Let $t = \mathrm{I}\, x\, .\, \varphi$. If there is a unique $d \in \mathcal{D}$ such that $V_{A[x \mapsto d]}(\varphi) = \mathrm{T}$, then $V_A(t) = d$; otherwise $V_A(t)$ is undefined.
4. Let $t = o(t_1, \ldots, t_n)$. If $V_A(t_1), \ldots, V_A(t_n)$ are defined and $I(o)$ is defined at $\langle V_A(t_1), \ldots, V_A(t_n)\rangle$, then $V_A(t) = I(o)(V_A(t_1), \ldots, V_A(t_n))$; otherwise $V_A(t)$ is undefined.
5. Let $\varphi = p(t_1, \ldots, t_n)$. If $V_A(t_1), \ldots, V_A(t_n)$ are defined, then $V_A(\varphi) = I(p)(V_A(t_1), \ldots, V_A(t_n))$; otherwise $V_A(\varphi) = \mathrm{F}$.

6. Let $\varphi = \neg\varphi'$. If $V_A(\varphi') = \text{F}$, then $V_A(\varphi) = \text{T}$; otherwise $V_A(\varphi) = \text{F}$.
7. Let $\varphi = \varphi' \supset \varphi''$. If $V_A(\varphi') = \text{T}$ and $V_A(\varphi'') = \text{F}$, then $V_A(\varphi) = \text{F}$; otherwise $V_A(\varphi) = \text{T}$.
8. Let $\varphi = \forall x\,.\,\varphi'$. If $V_{A[x\mapsto d]}(\varphi') = \text{T}$ for all $d \in \mathcal{D}$, then $V_A(\varphi) = \text{T}$; otherwise $V_A(\varphi) = \text{F}$.

For an expression $E$, $V_A^{\mathcal{M}}(E)$ is called the *value* of $E$ in $\mathcal{M}$ with respect to $A$ (when it is defined). Notice that $V_A^{\mathcal{M}}$ is a partial valuation function on terms but a total valuation function on formulas. A term $t$ of $\mathcal{L}$ is *defined* in $\mathcal{M}$ with respect to $A$ if its value $V_A^{\mathcal{M}}(t)$ is defined. If $\mathcal{M}$ is regular, $V_A^{\mathcal{M}}$ is a total valuation function on regular terms. Hence regular terms are always defined in regular models.

A formula $\varphi$ of $\mathcal{L}$ is *valid* in $\mathcal{M}$ if $V_A^{\mathcal{M}}(\varphi) = \text{T}$ for every variable assignment $A$ into $\mathcal{M}$. Let $\mathcal{T} = (\mathcal{L}, \Gamma)$ be a theory. A *model* for $\mathcal{T}$ is a model for $\mathcal{L}$ in which each $\varphi \in \Gamma$ is valid. A formula $\varphi$ is *valid* in $\mathcal{T}$ *in the partial sense*, written $\mathcal{T} \models_{\text{par}} \varphi$, if it is valid in each model for $\mathcal{T}$. A formula $\varphi$ is *valid* in $\mathcal{T}$ *in the regular sense*, written $\mathcal{T} \models_{\text{reg}} \varphi$, if it is valid in each regular model for $\mathcal{T}$.

## 2.3. The Elimination Theorem

The machinery in PFOL for partial functions and undefined terms—the operator symbols and the I operator—is purely a convenience; it extends but does not alter the conceptual framework of classical first-order logic. In fact, as we will soon see, any theory of PFOL can be translated into a logically equivalent theory of ordinary first-order logic by eliminating the use of operator symbols and I.

THEOREM 2.1. (Elimination Theorem). *For every theory $\mathcal{T} = (\mathcal{L}, \Gamma)$, there is a regular theory $\mathcal{T}^* = (\mathcal{L}^*, \Gamma^*)$ and a translation from each formula $\varphi$ of $\mathcal{L}$ to a regular formula $\varphi^*$ of $\mathcal{L}^*$ such that*

$$\mathcal{T} \models_{\text{par}} \varphi \quad \textit{iff} \quad \mathcal{T}^* \models_{\text{reg}} \varphi^*.$$

*Moreover, $\mathcal{T}^* = \mathcal{T}$ if $\mathcal{L}$ contains no operator symbols and $\mathcal{T}$ is regular, and $\varphi^* = \varphi$ if $\varphi$ contains no operator symbols and is regular.*

*Proof.* The first step of the proof is to eliminate operator symbols by replacing them with predicate symbols that denote their graphs.

Assume $\mathcal{L} = (\mathcal{C}, \mathcal{O}, \mathcal{P})$ and define $\mathcal{L}^* = (\mathcal{C}, \emptyset, \mathcal{P} \cup \{p_o : o \in \mathcal{O}\})$ where $p_o \notin \mathcal{P}$ and $p_o$ is $(n+1)$-ary if $o$ is $n$-ary for all $o \in \mathcal{O}$ and $p_{o_1} \neq p_{o_2}$ for all $o_1, o_2 \in \mathcal{O}$. For each $n$-ary $o \in \mathcal{O}$, define $U_o$ to be the formula

$$\begin{aligned}&\forall x_1 \ldots \forall x_n \forall y_1 \forall y_2\,.\\ &\quad p_o(x_1, \ldots, x_n, y_1) \wedge p_o(x_1, \ldots, x_n, y_2) \supset y_1 = y_2.\end{aligned}$$

$U_o$ says that $p_o$ is the graph of a (partial) $n$-ary function.

Consider the following rewrite rule:

**R1** $p(s_1, \ldots, o(t_1, \ldots, t_n), \ldots, s_m) \Rightarrow$
$\exists\, y \,.\, p_o(t_1, \ldots, t_n, y) \wedge p(s_1, \ldots, y, \ldots, s_m)$

where $p \in \mathcal{P} \cup \{p_o : o \in \mathcal{O}\}$, $o \in \mathcal{O}$, and $y$ does not occur in

$$p(s_1, \ldots, o(t_1, \ldots, t_n), \ldots, s_m).$$

Let $\varphi$ be a formula of $\mathcal{L}$. Define the transformation T1 as follows. If $\varphi$ contains no operator symbols, then $\mathrm{T1}(\varphi) = \varphi$. Otherwise, $\mathrm{T1}(\varphi)$ is the result of applying R1 to $\varphi$ so that the first occurrence of an operator symbol in $\varphi$ is eliminated. Let $\varphi'$ be the result of repeatedly applying T1 to $\varphi$ until all operator symbols have been eliminated. Notice the $\varphi'$ is well-defined since each application of R1 reduces by one the number of occurrences of operator symbols in $\varphi$. $\varphi'$ is clearly a formula of $\mathcal{L}^*$.

Define $\mathcal{T}' = (\mathcal{L}^*, \Gamma')$ where

$$\Gamma' = \{\varphi' : \varphi \in \Gamma\} \cup \{U_o : o \in \mathcal{O}\}.$$

By a straightforward argument,

$$\mathcal{T} \models_{\mathrm{par}} \varphi \;\text{ iff }\; \mathcal{T}' \models_{\mathrm{par}} \varphi'$$

for all formulas $\varphi$ of $\mathcal{L}$.

The second step is to eliminate I.

Consider the following rewrite rule:

**R2** $p(s_1, \ldots, (\mathrm{I}\, x \,.\, \psi), \ldots, s_m) \Rightarrow$
$\exists y \,.\, \psi[x \mapsto y] \wedge (\forall z \,.\, \psi[x \mapsto z] \supset z = y) \wedge p(s_1, \ldots, y, \ldots, s_m)$

where $p \in \mathcal{P} \cup \{p_o : o \in \mathcal{O}\}$, $\psi$ is regular, $y$ does not occur in

$$p(s_1, \ldots, (\mathrm{I}\, x \,.\, \psi), \ldots, s_m),$$

$y \neq z$, and $z$ does not occur in $\psi$.

Given a formula $\varphi$ of $\mathcal{L}^*$. Define the transformation T2 as follows. If $\varphi$ is regular, then $\mathrm{T2}(\varphi) = \varphi$. Otherwise, $\mathrm{T2}(\varphi)$ is the result of applying R2 to $\varphi$ so that the first occurrence of I in $\varphi$, that is in a term of the form $\mathrm{I}\, x \,.\, \psi$ where $\psi$ is regular, is eliminated. Let $\varphi^\bullet$ be the result of repeatedly applying T2 to $\varphi$ until all occurrences of I have been eliminated. Notice that $\varphi^\bullet$ is well-defined since each application of R2 reduces by one the number of occurrences of I in $\varphi$. $\varphi^\bullet$ is clearly a regular formula of $\mathcal{L}^*$.

Define $\mathcal{T}^* = (\mathcal{L}^*, \Gamma^*)$ to be the regular theory where

$$\Gamma^* = \{\varphi^\bullet : \varphi \in \Gamma'\}.$$

It follows from the semantics of I that

$$\mathcal{T}' \models_{\text{par}} \varphi \text{ iff } \mathcal{T}^* \models_{\text{par}} \varphi^\bullet$$

for all formulas $\varphi$ of $\mathcal{L}^*$.

Since $\mathcal{L}^*$ contains no operator symbols,

$$\mathcal{T}^* \models_{\text{par}} \varphi \text{ iff } \mathcal{T}^* \models_{\text{reg}} \varphi$$

for all formulas $\varphi$ of $\mathcal{L}^*$. For $\varphi$ of $\mathcal{L}$, define $\varphi^* = (\varphi')^\bullet$. This completes the proof. □

### 2.4. An Axiomatic System

The following axiom schemas together with the rules of modus ponens and generalization constitute a Hilbert-style axiomatic system for $\mathcal{L}$ which is complete with respect to the semantics of PFOL given in the previous subsection.

**PFOL1** $\varphi \supset (\psi \supset \varphi)$.

**PFOL2** $[\varphi \supset (\psi \supset \theta)] \supset [(\varphi \supset \psi) \supset (\varphi \supset \theta)]$.

**PFOL3** $(\neg\varphi \supset \neg\psi) \supset (\psi \supset \varphi)$.

**PFOL4** $(\forall x \,.\, \varphi \supset \psi) \supset (\varphi \supset \forall x \,.\, \psi)$ where $x$ is not free in $\varphi$.

**PFOL5** $[(\forall x \,.\, \varphi) \wedge t\!\downarrow] \supset \varphi[x \mapsto t]$ where $t$ is free for $x$ in $\varphi$.

**PFOL6** $\forall x \,.\, x = x$.

**PFOL7** $s \simeq t \supset (\varphi \supset \varphi^*)$ where $\varphi^*$ is the result of replacing one occurrence of $s$ in $\varphi$ by an occurrence of $t$, provided that the occurrence of $s$ is not a variable immediately after $\forall$ or I.

**PFOL8** $x\!\downarrow$ where $x \in \mathcal{V}$.

**PFOL9** $a\!\downarrow$ where $a \in \mathcal{C}$.

**PFOL10** $(\mathrm{I}\, x \,.\, \varphi)\!\downarrow \equiv [\exists x \,.\, \varphi \wedge (\forall y \,.\, \varphi[x \mapsto y] \supset y = x)]$ where $y$ does not occur in $\varphi$.

**PFOL11** $(\mathrm{I}\, x \,.\, \varphi)\!\downarrow \supset \varphi[x \mapsto (\mathrm{I}\, x \,.\, \varphi)]$ where $(\mathrm{I}\, x \,.\, \varphi)$ is free for $x$ in $\varphi$.

**PFOL12** $[t_1\!\uparrow \vee \cdots \vee t_n\!\uparrow] \supset o(t_1, \ldots, t_n)\!\uparrow$ where $o \in \mathcal{O}$ is $n$-ary.

**PFOL13** $[t_1{\uparrow} \vee \cdots \vee t_n{\uparrow}] \supset \neg p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$ is $n$-ary.

**Modus Ponens** From $\varphi \supset \psi$ and $\varphi$ infer $\psi$.

**Generalization** From $\varphi$ infer $\forall\, x\,.\, \varphi$.

A very important property of PFOL, which is a consequence of PFOL7, is that undefined terms are indiscernible. This means that an undefined term in a formula can be replaced by any other undefined term without changing the meaning of the formula. PFOL has this property by virtue of being a "logic of definedness" in contrast to a "logic of existence" (see [13] for a discussion of this distinction).

## 3. Partial NBG

We present in this section an axiomatization of NBG (with choice) in the form of a PFOL theory called *Partial* NBG (PNBG). Our axiomatization will be very similar to the finite axiomatization of NBG given by K. Gödel in [14]. It will be easy to define in PNBG term constructors for function application and abstraction using the definite description operator of PFOL.

### 3.1. An Axiomatization of NBG

Let $\mathcal{L}_0 = (\emptyset, \emptyset, \{=, \in\})$ where $\in$ is a binary predicate symbol which is intended to denote the membership relation. It is possible to axiomatize NBG as a regular theory of PFOL over $\mathcal{L}$. However, PNBG will take full advantage of the partial functions machinery of PFOL; it will be a theory over an expansion of $\mathcal{L}_0$.

We begin by defining the following abbreviations:

| | | |
|---|---|---|
| $(s \in t)$ | for | $\in (s, t)$. |
| $(s \notin t)$ | for | $\neg(s \in t)$. |
| $\forall\, x : p\,.\, \varphi$ | for | $\forall\, x\,.\, p(x) \supset \varphi$. |
| $\exists\, x : p\,.\, \varphi$ | for | $\exists\, x\,.\, p(x) \wedge \varphi$. |
| $\mathrm{I}\, x : p\,.\, \varphi$ | for | $\mathrm{I}\, x\,.\, p(x) \wedge \varphi$. |
| $\Box\, x_1, \ldots, x_n : p\,.\, \varphi$ | for | $\Box\, x_1 : p \ldots \Box\, x_n : p\,.\, \varphi$. |

Here $p$ is a unary predicate symbol and $\Box$ is $\forall$ or $\exists$.

We now list the 32 axioms of PNBG. The first 16 of them are *definitions*, sentences which define new symbols in terms of old symbols. The remaining 16 are the "proper" axioms.

We define two new predicate symbols $C$ and $V$. $C(x)$ says "$x$ is a class", and $V(x)$ says "$x$ is a set", i.e., a class which is the member of some other class. They are defined by the following axioms:

**PNBG1 (Definition of $C$)** $\forall x\,.\,C(x) \equiv (x = x)$.

**PNBG2 (Definition of $V$)** $\forall x\,.\,V(x) \equiv \exists y\,.\,x \in y$.

In the rest of this subsection we will use the variables $a, b, c$ to denote classes and $w, x, y, z$ to denote sets.

The next fourteen axioms define additional operator and predicate symbols.

**PNBG3 (Definition of Pair)** $\forall a, b\colon C\,.\,\{a, b\} \simeq$
$[\mathrm{I}\,x\colon V\,.\,V(a) \wedge V(b) \wedge \forall y\colon V\,.\,y \in x \equiv (y = a \vee y = b)]$.

**PNBG4 (Definition of Singleton)** $\forall a\colon C\,.\,\{a\} \simeq \{a, a\}$.

**PNBG5 (Definition of Ordered Pair)** $\forall a, b\colon C\,.\,\langle a, b\rangle \simeq$
$\{\{a\}, \{a, b\}\}$.

**PNBG6 (Definition of Ordered Triple)** $\forall a, b, c\colon C\,.\,\langle a, b, c\rangle \simeq$
$\langle a, \langle b, c\rangle\rangle$.

**PNBG7 (Definition of Subset)** $\forall a, b\colon C\,.\,a \subseteq b \equiv$
$(\forall x\colon V\,.\,x \in a \supset x \in b)$.

**PNBG8 (Definition of Proper Subset)** $\forall a, b\colon C\,.\,a \subset b \equiv$
$(a \subseteq b \wedge a \neq b)$.

**PNBG9 (Definition of Emptiness)** $\forall a\colon C\,.\,\mathsf{empty}(a) \equiv$
$\forall x\colon V\,.\,x \notin a$.

**PNBG10 (Definition of Univocal)** $\forall a\colon C\,.\,\mathsf{univocal}(a) \equiv$
$[\forall x, y, z\colon V\,.\,(\langle x, y\rangle \in a \wedge \langle x, z\rangle \in a) \supset y = z]$.

**PNBG11 (Definition of Function)** $\forall a\colon C\,.\,\mathsf{function}(a) \equiv$
$[\mathsf{univocal}(a) \wedge \forall x\colon V\,.\,x \in a \equiv (\exists y, z\colon V\,.\,x = \langle y, z\rangle)]$.

**PNBG12 (Definition of Intersection)** $\forall a, b\colon C\,.\,a \cap b \simeq$
$[\mathrm{I}\,c\colon C\,.\,\forall x\colon V\,.\,x \in c \equiv (x \in a \wedge x \in b)]$.

**PNBG13 (Definition of Complement)** $\forall a\colon C\,.\,\overline{a} \simeq$
$[\mathrm{I}\,b\colon C\,.\,\forall x\colon V\,.\,x \in b \equiv x \notin a]$.

**PNBG14 (Definition of Domain)** $\forall\, a\colon C\,.\, \mathsf{domain}(a) \simeq [\mathrm{I}\, b\colon C\,.\, \forall\, x\colon V\,.\, x \in b \equiv (\exists\, y\colon V\,.\, \langle x, y\rangle \in a)]$.

**PNBG15 (Definition of Sum Class)** $\forall\, a\colon C\,.\, \mathsf{sum}(a) \simeq [\mathrm{I}\, b\colon C\,.\, \forall\, x\colon V\,.\, x \in b \equiv (\exists\, y\colon V\,.\, x \in y \wedge y \in a)]$.

**PNBG16 (Definition of Power Class)** $\forall\, a\colon C\,.\, \mathsf{power}(a) \simeq [\mathrm{I}\, b\colon C\,.\, \forall\, x\colon V\,.\, x \in b \equiv x \subseteq a]$.

The first two proper axioms of PNBG are:

**PNBG17 (Extensionality)** $\forall\, a, b\colon C\,.\, (\forall\, x\colon V\,.\, x \in a \equiv x \in b) \supset a = b$.

**PNBG18 (Pairing)** $\forall\, x, y\colon V\,.\, \{x, y\}\!\downarrow$.

The next group of axioms specify the existence of certain classes:

**PNBG19 (Membership Class)** $\exists\, a\colon C\,.\, \forall\, x, y\colon V\,.\, \langle x, y\rangle \in a \equiv x \in y$.

**PNBG20 (Intersection)** $\forall\, a, b\colon C\,.\, (a \cap b)\!\downarrow$.

**PNBG21 (Complement)** $\forall\, a\colon C\,.\, \overline{a}\!\downarrow$.

**PNBG22 (Domain)** $\forall\, a\colon C\,.\, \mathsf{domain}(a)\!\downarrow$.

**PNBG23 (Direct Product)** $\forall\, a\colon C\,.\, \exists\, b\colon C\,.\, \forall\, x, y\colon V\,.\, \langle x, y\rangle \in b \equiv x \in a$.

**PNBG24 (Permutation 1)** $\forall\, a\colon C\,.\, \exists\, b\colon C\,.\, \forall\, x, y\colon V\,.\, \langle x, y\rangle \in b \equiv \langle y, x\rangle \in a$.

**PNBG25 (Permutation 2)** $\forall\, a\colon C\,.\, \exists\, b\colon C\,.\, \forall\, x, y, z\colon V\,.\, \langle x, y, z\rangle \in b \equiv \langle y, z, x\rangle \in a$.

**PNBG26 (Permutation 3)** $\forall\, a\colon C\,.\, \exists\, b\colon C\,.\, \forall\, x, y, z\colon V\,.\, \langle x, y, z\rangle \in b \equiv \langle x, z, y\rangle \in a$.

The next four axioms specify the existence of certain sets:

**PNBG27 (Infinity)** $\exists\, x\colon V\,.\, \neg\mathsf{empty}(x) \wedge [\forall\, y\colon V\,.\, y \in x \supset (\exists z\colon V\,.\, z \in x \wedge y \subset z)]$.

**PNBG28 (Sum Set)** $\forall\, x\colon V\,.\, V(\mathsf{sum}(x))$.

**PNBG29 (Power Set)** $\forall\, x : V \,.\, V(\mathsf{power}(x))$.

**PNBG30 (Replacement)** $\forall\, a : C \,.\, \mathsf{univocal}(a) \supset$
$[\forall\, x : V \,.\, \exists\, y : V \,.\, \forall\, z : V \,.\, z \in y \equiv (\exists\, w : V \,.\, w \in x \wedge \langle w, z\rangle \in a)]$.

The last two axioms are the axiom of foundation and the axiom of global choice:

**PNBG31 (Foundation)** $\forall\, a : C \,.\, \neg\mathsf{empty}(a) \supset$
$[\exists\, x : V \,.\, x \in a \wedge \forall\, y : V \,.\, \neg(y \in x \wedge y \in a)]$.

**PNBG32 (Global Choice)** $\exists\, a : C \,.\, \mathsf{function}(a) \wedge$
$[\forall\, x : V \,.\, \neg\mathsf{empty}(x) \supset (\exists\, y : V \,.\, y \in x \wedge \langle x, y\rangle \in a)]$.

Notes:

1. $\{a, b\}$ and $\langle a, b\rangle$ are undefined whenever $a$ is not a set or $b$ is not a set.
2. In Gödel's axiomatization of NBG in [14], $\langle x, y\rangle$ represents a mapping of $y$ to $x$; in our axiomatization it represents a mapping of $x$ to $y$.
3. Axiom PNBG23 says that, for all classes $a$, the direct product $a \times V$ is a class, where $V$ is the class of all sets.
4. It is an exercise in [19, p. 164] that the first permutation axiom, PNBG24, follows from axioms PNBG22, PNBG23, PNBG25, and PNBG26.
5. The axiom of foundation, PNBG31, is dispensable and could be replaced with an "antifoundation" axiom.
6. The axiom of global choice, PNBG32, implies the axiom of local choice (as given, for example, in [19]).

Let $\mathcal{L}_{\text{PNBG}} = (\emptyset, \mathcal{O}_{\text{PNBG}}, \mathcal{P}_{\text{PNBG}})$ be $\mathcal{L}_0$ plus the operator and predicate symbols defined in axioms PNBG1, ..., PNBG16, and let $\Gamma_{\text{PNBG}} = \{\text{PNBG1}, \ldots, \text{PNBG32}\}$. Then $\text{PNBG} = (\mathcal{L}_{\text{PNBG}}, \Gamma_{\text{PNBG}})$.

### 3.2. Function Application and Abstraction

The following abbreviations introduce term constructors for (unary) function application and abstraction:

| | | |
|---|---|---|
| $f[a]$ | for | $\mathrm{I}\, b \,.\, \mathsf{function}(f) \wedge \langle a, b\rangle \in f$. |
| $(\lambda\, x \,.\, t)$ | for | $\mathrm{I}\, g \,.\, \mathsf{function}(g) \wedge$ |
| | | $[\forall\, x \,.\, \mathsf{if}(V(x) \wedge V(t), g[x] = t, g[x]\!\uparrow)]$. |

Here $f, a, t$ are terms, $x$ is a variable, $b$ does not occur in $f$ or $a$, $g \neq x$, and $g$ does not occur in $t$. Terms of the form $f[a]$ and $(\lambda\, x\,.\; t)$ are called *applications* and *abstractions*, respectively.

The following abbreviation provides notation for a function over a restricted domain:

$$(\lambda\, x : p\,.\; t) \qquad \text{for} \qquad \lambda\, x\,.\; \mathsf{if}(p(x), t, \bot).$$

Here $x$ is a variable, $p$ is a unary predicate symbol, and $t$ is a term.

## 4. NBG*: Partial NBG plus Sorts

A *sort* is a syntactic object intended to denote a nonempty domain of values. In this section we will add a system of sorts to PNBG for classifying terms, particularly terms that denote partial functions. More precisely, we will define a new logic called NBG* in which a theory is an extension of PNBG plus a sort system. A sort system in NBG* will be similar to a sort system in LUTINS in both structure and use.

### 4.1. Sort Systems

The set of sort symbols built from a set $S$ of atomic sort symbols will be the set $\Omega(S)$ defined below.

Let $S$ be a set of symbols. $\Omega(S)$ is the set defined by:

1. $S \subseteq \Omega(S)$.
2. If $\alpha, \beta \in \Omega(S)$, then $\alpha \rightharpoonup \beta \in \Omega(S)$.

A sort of the form $\alpha \rightharpoonup \beta$ is intended to denote the domain of partial functions from the domain denoted by $\alpha$ to the domain denoted by $\beta$.

Given a total function $f : S \to \Omega(S)$, $\preceq_f$ is the smallest binary relation on $\Omega(S)$ such that:

1. If $\alpha \in S$, then $\alpha \preceq_f f(\alpha)$.
2. $\preceq_f$ is reflexive, i.e., for all $\alpha \in \Omega(S)$, $\alpha \preceq_f \alpha$.
3. $\preceq_f$ is transitive, i.e., for all $\alpha, \beta, \gamma \in \Omega(S)$, if $\alpha \preceq_f \beta$ and $\beta \preceq_f \gamma$, then $\alpha \preceq_f \gamma$.
4. If $\alpha_1 \preceq_f \alpha_2$ and $\beta_1 \preceq_f \beta_2$, then $\alpha_1 \rightharpoonup \beta_1 \preceq_f \alpha_2 \rightharpoonup \beta_2$.

$\preceq_f$ is *noetherian* if every ascending sequence of members of $\Omega(S)$,

$$\alpha_1 \preceq_f \alpha_2 \preceq_f \alpha_3 \preceq_f \cdots,$$

is eventually stationary, i.e., there is some $m$ such that $\alpha_i = \alpha_m$ for all $i \geq m$. If $\preceq_f$ is noetherian, then $\preceq_f$ is obviously antisymmetric (i.e., for all $\alpha, \beta \in \Omega(S)$, if $\alpha \preceq_f \beta$ and $\beta \preceq_f \alpha$, then $\alpha = \beta$). Hence, $\preceq_f$ is a partial order if it is noetherian.

A *sort system* of NBG* is a pair $(\mathcal{A}, \xi)$ where $\mathcal{A}$ is a set of symbols with $\mathbf{C}, \mathbf{V} \in \mathcal{A}$ and $\xi$ is a total function from $\mathcal{A}$ to $\Omega(\mathcal{A})$ such that:

1. For all $\alpha \in \mathcal{A}$, $\xi(\alpha) = \alpha$ iff $\alpha = \mathbf{C}$.
2. $\preceq_\xi$ is noetherian.

We will see later why $\preceq_\xi$ must be noetherian instead of just a partial order.

A *sort* of $(\mathcal{A}, \xi)$ is any member of $\Omega(\mathcal{A})$. The sorts in $\mathcal{A}$ and $\Omega(\mathcal{A}) \setminus \mathcal{A}$ are called *atomic sorts* and *compound sorts*, respectively. The *enclosing sort* of $\alpha \in \mathcal{A}$ is the sort $\xi(\alpha)$. The *least upper bound* of $\alpha$ and $\beta$, written $\alpha \sqcup_\xi \beta$, is the least upper bound of $\alpha$ and $\beta$ in the partial order $\preceq_\xi$. (The least upper bound of two sorts is not always defined.) The maximal sorts in $\preceq_\xi$ are $\mathbf{C}$ plus the compound sorts formed from $\mathbf{C}$ alone.

A *function sort* is either a compound sort or an atomic sort $\alpha$ such that $\alpha \preceq_\xi \beta$ for some compound sort $\beta \in \Omega(\mathcal{A})$. The *range sort* of a function sort $\alpha$, written $\mathrm{ran}(\alpha)$, is defined as follows: if $\alpha = \beta \rightharpoonup \gamma$, then $\mathrm{ran}(\alpha) = \gamma$; otherwise, $\mathrm{ran}(\alpha) = \mathrm{ran}(\xi(\alpha))$. (We leave it as an exercise to the reader to verify that the notion of a range sort is well-defined.)

### 4.2. Syntax and Semantics

A *language* of NBG* is a tuple $(\mathcal{C}, \mathcal{O}, \mathcal{P}, \mathcal{A}, \xi, \sigma)$ such that:

1. $(\mathcal{C}, \mathcal{O}, \mathcal{P})$ is a language of PFOL with $\mathcal{O}_{\mathrm{PNBG}} \subseteq \mathcal{O}$ and $\mathcal{P}_{\mathrm{PNBG}} \subseteq \mathcal{P}$.
2. $(\mathcal{A}, \xi)$ is a sort system.
3. $\sigma : \mathcal{V} \cup \mathcal{C} \to \Omega(\mathcal{A})$ is total function such that

$$\mathcal{V}_\alpha = \{x \in \mathcal{V} : \sigma(x) = \alpha\}$$

is infinite for all $\alpha \in \Omega(\mathcal{A})$.

In the remainder of this section, let $\mathcal{L} = (\mathcal{C}, \mathcal{O}, \mathcal{P}, \mathcal{A}, \xi, \sigma)$ be a language of NBG*.

The *terms* and *formulas* of $\mathcal{L}$ are exactly the same as the terms and formulas of the PFOL language $(\mathcal{C}, \mathcal{O}, \mathcal{P})$. Define $\widetilde{\mathcal{L}}$ to be the following language of PFOL:

$$(\mathcal{C}, \mathcal{O}, \mathcal{P} \cup \{p_\alpha : \alpha \in \Omega(\mathcal{A})\})$$

where $p_{\mathbf{C}} = C$, $p_{\mathbf{V}} = V$, $p_\alpha$ is a unary predicate symbol not in $\mathcal{P}$ for all $\alpha \in \Omega(\mathcal{A}) \setminus \{\mathbf{C}, \mathbf{V}\}$, and $p_\alpha \neq p_\beta$ for all $\alpha, \beta \in \Omega(\mathcal{A})$. For an expression $E$ of $\mathcal{L}$, let $\widetilde{E}$ be the expression of $\widetilde{\mathcal{L}}$ obtained by replacing each "$\mathrm{I}\, x$" and "$\forall\, x$" in $E$ with "$\mathrm{I}\, x : p_{\sigma(x)}$" and "$\forall\, x : p_{\sigma(x)}$", respectively.

Given a model $\mathcal{M} = (\mathcal{D}, I)$ for $\widetilde{\mathcal{L}}$ and $\alpha \in \Omega(\mathcal{A})$, let $\mathcal{D}_\alpha$ be the domain of all $d \in \mathcal{D}$ such that $I(p_\alpha)(d) = \mathrm{T}$.

A *model* for $\mathcal{L}$ is a model $\mathcal{M} = (\mathcal{D}, I)$ for $\widetilde{\mathcal{L}}$ such that:

1. For all $\alpha \in \Omega(\mathcal{A})$, $\mathcal{D}_\alpha$ is nonempty.
2. For all $\alpha, \beta \in \Omega(\mathcal{A})$, if $\alpha \preceq_\xi \beta$, then $\mathcal{D}_\alpha \subseteq \mathcal{D}_\beta$.
3. $\mathcal{D}_{\mathbf{C}} = \mathcal{D}$.
4. For all $\alpha, \beta \in \Omega(\mathcal{A})$,

$$\forall\, f \,.\, p_{\alpha \rightharpoonup \beta}(f) \equiv \\ [\mathsf{function}(f) \wedge [\forall\, a, b \,.\, f[a] = b \supset (p_\alpha(a) \wedge p_\beta(b))]]$$

   is valid in $\mathcal{M}$.

5. $I(a) \in \mathcal{D}_{\sigma(a)}$ for all $a \in \mathcal{C}$.

One can easily construct a model for $\mathcal{L}$ by appealing to the fact that $\preceq_\xi$ is noetherian.

Let $\mathcal{M} = (\mathcal{D}, I)$ be a model for $\mathcal{L}$. A variable assignment $A$ into $\mathcal{M}$ *respects* $\mathcal{L}$ if $A(x) \in \mathcal{D}_{\sigma(x)}$ for all $x \in \mathcal{V}$. Define $U^{\mathcal{M}}$ to be the binary function such that, for all variable assignments $A$ into $\mathcal{M}$ that respect $\mathcal{L}$ and all expressions $E$ of $\mathcal{L}$,

$$U^{\mathcal{M}}_A(E) = V^{\mathcal{M}}_A(\widetilde{E}).$$

A formula $\varphi$ of $\mathcal{L}$ is *valid* in $\mathcal{M}$ if $U^{\mathcal{M}}_A(\varphi) = \mathrm{T}$ for every variable assignment $A$ into $\mathcal{M}$ that respects $\mathcal{L}$. We thus see that a sort denotes a nonempty domain of classes, and that the assignment of sorts to variables and individual constants defined by $\sigma$ restricts the values that the variables and individual constants may have in a model.

A *theory* of NBG$^*$ is a pair $\mathcal{T} = (\mathcal{L}, \Gamma)$ where $\mathcal{L}$ is a language of NBG$^*$ and $\Gamma$ is a set of sentences of $\mathcal{L}$ with the property that, for each $\varphi \in \Gamma_{\mathrm{PNBG}}$, there is some $\psi \in \Gamma$ such that $\widetilde{\psi}$ is alpha-equivalent to $\varphi$ (i.e., $\psi$ converts to $\varphi$ by renaming variables). Let $\mathcal{T} = (\mathcal{L}, \Gamma)$ be a theory of NBG$^*$ . A *model* for $\mathcal{T}$ is a model for $\mathcal{L}$ in which each $\varphi \in \Gamma$ is valid. A formula $\varphi$ of $\mathcal{L}$ is *valid* in $\mathcal{T}$, written $\mathcal{T} \models \varphi$, if it is valid in each model for $\mathcal{T}$, and is *universally valid*, written $\models \varphi$, if it is valid in each theory $\mathcal{T}' = (\mathcal{L}, \Gamma')$ of NBG$^*$.

The following theorem is an immediate consequence of the previous definitions:

THEOREM 4.1. *For every theory* $\mathcal{T} = (\mathcal{L}, \Gamma)$ *of* NBG$^*$*, there is a theory* $\widetilde{\mathcal{T}}$ *of* PFOL *over* $\widetilde{\mathcal{L}}$ *such that*

$$\mathcal{T} \models \varphi \quad \textit{iff} \quad \widetilde{\mathcal{T}} \models_{\text{par}} \widetilde{\varphi}$$

*for all sentences* $\varphi$ *of* $\mathcal{L}$*. Moreover,* $\widetilde{\mathcal{T}}$ *is an extension of* PNBG *(renaming variables if necessary).*

## 4.3. Sorts Assigned to Terms

Sorts serve two purposes. The first, which we have seen above, is to restrict the value that a variable or individual constant may have in a model. The second is to indicate what range of value a term has, provided it is defined. This is achieved by extending $\sigma$ to all the terms of $\mathcal{L}$. Then, for example, if the sort $\alpha$ is assigned to a term $t$, the value of $t$ would be in the domain denoted by $\alpha$, provided $t$ is defined.

The *sort* of a term $t$, written $\bar{\sigma}(t)$, is defined by:

1. $\bar{\sigma}(a) = \sigma(a)$ if $a \in \mathcal{V} \cup \mathcal{C}$.
2. $\bar{\sigma}(\mathrm{I}\, x\,.\, \varphi) = \sigma(x)$.
3. $\bar{\sigma}(o(t_1, \ldots, t_n)) = \mathbf{C}$ if $o \in \mathcal{O}$.

In order to take advantage of this assignment of sorts to terms, we must control what sorts are assigned to the terms formed by the term constructors we have introduced. Therefore, we introduce the following modified abbreviations:

$\bot_\alpha$ for $\mathrm{I}\, x\,.\, x \neq x$ where $\sigma(x) = \alpha$.

$\mathsf{if}(\varphi, s, t)$ for $\mathrm{I}\, x\,.\, (\varphi \supset x = s) \wedge (\neg\varphi \supset x = t)$
where $x$ does not occur in $\varphi$, $s$, or $t$ and
$$\sigma(x) = \begin{cases} \bar{\sigma}(s) \sqcup_\xi \bar{\sigma}(t) & \text{if this is defined} \\ \mathbf{C} & \text{otherwise} \end{cases}$$

$f[a]$ for $\mathrm{I}\, b\,.\, \mathsf{function}(f) \wedge \langle a, b\rangle \in f$
where $b$ does not occur in $f$ or $a$ and
$$\sigma(b) = \begin{cases} \mathrm{ran}(\bar{\sigma}(f)) & \text{if } \bar{\sigma}(f) \text{ is a function sort} \\ \mathbf{V} & \text{otherwise} \end{cases}$$

$(\lambda\, x\,.\, t)$ for $\mathrm{I}\, g\,.\, \mathsf{function}(g) \wedge$
$[\forall\, x\,.\, \mathsf{if}(V(x) \wedge V(t), g[x] = t, g[x]\uparrow)]$
where $g \neq x$, $g$ does not occur in $t$, and
$\sigma(g) = \sigma(x) \rightharpoonup \bar{\sigma}(t)$.

Let us also introduce the following new abbreviations:

| | | |
|---|---|---|
| $(t \downarrow \alpha)$ | for | $\exists\, x \,.\, x = t$<br>where $\sigma(x) = \alpha$ and $x$ does not occur in $t$. |
| $(t{\downarrow})$ | for | $(t \downarrow \bar{\sigma}(t))$. |
| $(\alpha \ll \beta)$ | for | $\forall\, x \,.\, (x \downarrow \beta)$ where $\sigma(x) = \alpha$. |
| $\langle \alpha \rangle$ | for | $\mathrm{I}\, x \,.\, \forall\, y \,.\, [y \in x \equiv (y \downarrow \alpha)]$<br>where $\sigma(x) = \sigma(y) = \mathbf{C}$. |

Here $t$ is a term and $\alpha$ and $\beta$ are sorts. $(t \downarrow \alpha)$, which is read as "$t$ is defined in $\alpha$", asserts that $t$ has a value in the domain denoted by $\alpha$. $(t{\downarrow})$, which as before is read as "$t$ is defined", asserts that $t$ has a value in the domain denoted by its assigned sort. $\alpha \ll \beta$, which is read as "$\alpha$ is a subsort of $\beta$", asserts that each member of the domain denoted by $\alpha$ is a member of the domain denoted by $\beta$. $\langle \alpha \rangle$ is a term that, if it is defined, denotes the same class denoted by $\alpha$.

The following statements are true for all terms $t$ and sorts $\alpha$, $\alpha_1$, $\alpha_2$, $\beta$, $\beta_1$, $\beta_2$, $\gamma$:

1. $\models (t \downarrow \mathbf{C}) \equiv C(t)$.
2. $\models (t \downarrow \mathbf{V}) \equiv V(t)$.
3. $\models (t \downarrow \alpha) \supset t{\downarrow}$.
4. $\models t{\downarrow} \equiv (t \downarrow \mathbf{C})$.
5. $\models \alpha \ll \mathbf{C}$.
6. $\models \alpha \ll \beta$ provided $\alpha \preceq_\xi \beta$.
7. $\models (\alpha \ll \beta \wedge \beta \ll \gamma) \supset \alpha \ll \gamma$.
8. $\models (\alpha_1 \ll \beta_1 \wedge \alpha_2 \ll \beta_2) \supset \alpha_1 \rightharpoonup \alpha_2 \ll \beta_1 \rightharpoonup \beta_2$.
9. $\models \langle \alpha \rangle{\downarrow} \supset \alpha \ll \mathbf{V}$.
10. $\models \alpha \rightharpoonup \beta \ll \mathbf{V} \rightharpoonup \mathbf{V}$.
11. $\models [(\langle \alpha \rangle \downarrow \mathbf{V}) \wedge (\langle \beta \rangle \downarrow \mathbf{V})] \supset (\langle \alpha \rightharpoonup \beta \rangle \downarrow \mathbf{V})$.

### 4.4. Alternate Syntaxes

The syntax for expressions of $\mathcal{L}$ given above has two small shortcomings. First, all occurrences of a variable must be assigned the same sort, even when the occurrences are in completely separate expressions. Second, one must have knowledge of $\sigma$ to understand the meaning of

an expression. These shortcomings can be avoided by using a different syntax in which the assignment of sorts to variables is specified directly within an expression.

One way of doing this is to tag each occurrence of a variable or individual constant with a sort. (A variable $x$ tagged with a sort $\alpha$, for example, might be written as $x_\alpha$ or $x^\alpha$.) Then there would be no need for $\sigma$, and different occurrences of the same variable could be tagged with different sorts. For example, the formula that says the variable $x$ of sort $\alpha$ denotes an element in the domain denoted by $\beta$ could be written in this syntax as

$$\exists\, y_\beta \,.\, y_\beta = x_\alpha.$$

This kind of syntax lacks the shortcomings described above, but it is more verbose (since every occurrence of a variable or individual constant would be tagged with a sort).

Another possibility is to assign sorts to bound variables where they are bound and to free variables at the beginning of an expression using a pseudo variable binder $\diamond$. ($\sigma$ would be used only to assign sorts to individual constants.) For example, the previous formula would be written as

$$\diamond\, x : \alpha \,.\, \exists\, y : \beta \,.\, y = x.$$

This syntax doesn't have any of the shortcomings mentioned above, but it does admit ambiguous expressions such as

$$\diamond\, x : \alpha \,.\, \forall\, x : \beta \,.\, x = x.$$

But there are various ways to deal with such expressions.

### 4.5. Individual Constants vs. Operator Symbols

A function can be represented in NBG* by either an individual constant or an operator symbol. Individual constants can represent only functions which are classes, that is, functions which map sets to sets. Operator symbols can represent functions which map classes to classes, which includes functions which are classes.

These two modes of representation, however, do not have equal status in NBG*. Individual constants are terms, but operator symbols are not terms and thus predicate symbols and other operator symbols can not be applied to them. Instead of being a defect of NBG*, this dichotomy between individual constants and operator symbols reflects the natural division between the different roles functions play in mathematics. A function that is to be *reasoned about*—such as a function

that maps real numbers to real numbers—is usually a class; it can be represented as an individual constant that has the full status of a term. A function that is only to assist in *forming assertions*—such as the function that maps a class of ordered pairs to its domain—may not be a class; it can be represented as a operator symbol that can be used to form terms but is not a term itself.

It would be easy to assign sorts to operator and predicate symbols in addition to variables and individual constants (see [7]). We have not done this for the sake of simplicity. One would certainly want the sort mechanism to support the full apparatus of terms in an implemented version of NBG*.

## 5. Conclusion

We have presented a formalization of set theory called NBG* which has the following characteristics:

- It is based on ideas and principles that are very familiar from mathematics practice.
- It has the same expressive power as ZF.
- It has the same convenient machinery for reasoning about functions as LUTINS.
- It can be implemented and used in much the same way that LUTINS is implemented and used in IMPS.

We believe that, by virtue of these characteristics, NBG* is well-suited to serve as a foundation for mechanized mathematics systems.

## References

1. M. Beeson. Formalizing constructive mathematics: Why and how? In F. Richman, editor, *Constructive Mathematics: Proceedings, New Mexico, 1980*, volume 873 of *Lecture Notes in Mathematics*, pages 146–190. Springer-Verlag, 1981.
2. M. J. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag, Berlin, 1985.
3. T. Burge. *Truth and Some Referential Devices*. PhD thesis, Princeton University, 1971.
4. T. Burge. Truth and singular terms. In K. Lambert, editor, *Philosophical Applications of Free Logic*, pages 189–204. Oxford University Press, 1991.
5. D. Craigen, S. Kromodimoeljo, I. Meisels, B. Pase, and M. Saaltink. EVES: An overview. Technical Report CP-91-5402-43, ORA Corporation, 1991.

6. W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–91, 1990.
7. W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.
8. W. M. Farmer. Theory interpretation in simple type theory. In J. Heering et al., editor, *Higher-Order Algebra, Logic, and Term Rewriting*, volume 816 of *Lecture Notes in Computer Science*, pages 96–123. Springer-Verlag, 1994.
9. W. M. Farmer. Reasoning about partial functions with the aid of a computer. *Erkenntnis*, 43:279–294, 1995.
10. W. M. Farmer. STMM: A Set Theory for Mechanized Mathematics. 26:269–289, 2001.
11. W. M. Farmer, J. D. Guttman, and F. J. Thayer Fábrega. IMPS: An updated system description. In M. McRobbie and J. Slaney, editors, *Automated Deduction—CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 298–302. Springer-Verlag, 1996.
12. W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.
13. S. Feferman. Definedness. *Erkenntnis*, 43:295–320, 1995.
14. K. Gödel. *The Consistency of the Axiom of Choice and the Generalized Continuum Hypothesis with the Axioms of Set Theory*, volume 3 of *Annals of Mathematical Studies*. Princeton University Press, 1940.
15. M. Gordon. Set theory, higher order logic or both? In J. Grundy and J. Harrison, editors, *Theorem Proving in Higher Order Logic: 9th International Conference, TPHOLs'96*, volume 1125 of *Lecture Notes in Computer Science*, pages 191–202. Springer-Verlag, 1996.
16. K. Kunen. *Set Theory: An Introduction to Independence Proofs*. North-Holland, 1980.
17. W. McCune. OTTER 2.0. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 663–664. Springer-Verlag, 1990.
18. N. D. Megill. *Metamath: A Computer Language for Pure Mathematics*. 1997. Available at `http://www.shore.net/~ndm/java/mm.html`.
19. E. Mendelson. *Introduction to Mathematical Logic*. Van Nostrand, 1964.
20. L. G. Monk. PDLM: A Proof Development Language for Mathematics. Technical Report M86-37, The MITRE Corporation, Bedford, Massachusetts, 1986.
21. R. Montague. Semantic closure and non-finite axiomatizability. In *Infinitistic Methods*, pages 45–69. Pergamon, 1961.
22. T. Nipkow and L. C. Paulson. Isabelle-91. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 673–676. Springer-Verlag, 1992.
23. I. L. Novak. A construction for models of consistent systems. *Fundamenta Mathematicae*, 37:87–110, 1950.
24. L. C. Paulson. Set theory for verification: I. from foundations to functions. *Journal of Automated Reasoning*, 11:353–389, 1993.
25. A. Quaife. Automated deduction in von Neumann-Bernays-Gödel set theory. *Journal of Automated Reasoning*, 8:91–147, 1993.
26. J. B. Rosser and H. Wang. Non-standard models for formal logics. *Journal of Symbolic Logic*, 15:113–129, 1950.
27. P. Rudnicki. An overview of the MIZAR project. Technical report, Department of Computing Science, University of Alberta, 1992.

28. R. Schock. *Logics without Existence Assumptions*. Almqvist & Wiksells, Stockholm, Sweden, 1968.
29. J. Shoenfield. A relative consistency proof. *Journal of Symbolic Logic*, 19:21–28, 1954.