Notre Dame Journal of Formal Logic
Volume 54, Number 1, 2013

The First-Order Syntax of Variadic Functions

Samuel Alexander

Abstract We extend first-order logic to include variadic function symbols and prove a substitution lemma. Two applications are given: one to bounded quantifier elimination and one to the definability of certain Borel sets.

1 Introduction

A variadic function is a function which takes a variable number of arguments: for example, a function from $\mathbb{N}^{<\mathbb{N}}$ to $\mathbb{N}$ is variadic, where $\mathbb{N}^{<\mathbb{N}}$ denotes the set of finite sequences of naturals. In classical first-order logic, a language has function symbols of fixed arities. In this paper I will explore how variadic function symbols can be added to first-order logic. In so doing, we will also formalize the syntax of the ellipsis, ..., which of course is closely related to variadic function symbols.

To get an idea of the subtleties of the ellipsis, consider the following "proof" that $5050 = 385$.

1. We know $1 + \cdots + 100 = 5050$.
2. We know $1 = 1^2$ and $100 = 10^2$.
3. By replacement, $1^2 + \cdots + 10^2 = 5050$.
4. Also, $1^2 + \cdots + 10^2 = (10)(10 + 1)(2 \cdot 10 + 1)/6 = 385$. So $5050 = 385$.

Evidently, mathematicians implicitly impose some special syntax on the ellipsis. This will be made explicit in the paper.

Of course, we can already talk about unary functions $\mathbb{N} \to \mathbb{N}$ which interpret their input as the code for a finite sequence. My hope is that some coding can be avoided by allowing variadic function symbols.

I was led to investigate the syntax of variadic function symbols when I was investigating a certain class of subsets of Baire space and realized that I could characterize that class with the help of first-order logic extended by variadic function symbols. The results are written up in Alexander [1]. Some of the basic results of this paper were first published there.

Received May 18, 2011; accepted January 23, 2012
2010 Mathematics Subject Classification: Primary 03B10
Keywords: variadic, syntax, ellipsis, substitution lemma
 10.1215/00294527-1731380

Variadic functions are used in many programming languages. What little literature presently exists mostly seems to be in this context (e.g., Byrd and Friedman [4]) and in the related context of λ-calculus (e.g., Goldberg [7]).

2 Basic Definitions

Definition 2.1

- A *first-order variadic language* (or simply a *variadic language*) is a first-order language, including a constant symbol $\mathbf{n}$ for every $n \in \mathbb{N}$, together with a set of *variadic function symbols*, and a special symbol $\cdots_x$ for every variable x.
- A *structure* (or a *model*) for a variadic language $\mathcal{L}$ is a structure $\mathcal{M}$ for the first-order part of $\mathcal{L}$, with universe $\mathbb{N}$, and which interprets each $\mathbf{n}$ as n, together with a set of variadic functions $\mathbb{N}^{<\mathbb{N}} \to \mathbb{N}$, one for every variadic function symbol of $\mathcal{L}$. If G is a variadic function symbol of $\mathcal{L}$, $G^{\mathcal{M}}$ will denote the corresponding function $G^{\mathcal{M}} : \mathbb{N}^{<\mathbb{N}} \to \mathbb{N}$.

Definition 2.2

- (Terms) If $\mathcal{L}$ is a variadic language, then the *terms* of $\mathcal{L}$, and their free variables, are defined as follows.
  1. If c is a constant symbol, then c is a term with $\mathrm{FV}(c) = \emptyset$.
  2. If x is a variable, then x is a term with $\mathrm{FV}(x) = \{x\}$.
  3. If f is an n-ary or variadic function symbol and $u_1, \ldots, u_n$ are terms, then $f(u_1, \ldots, u_n)$ is a term with free variables $\mathrm{FV}(u_1) \cup \cdots \cup \mathrm{FV}(u_n)$.
  4. If G is a variadic function symbol, u, v are terms, and x is a variable, then
  $$G\big(u(\mathbf{0}) \cdots_x u(v)\big)$$
  is a term with free variables $(\mathrm{FV}(u)\backslash\{x\}) \cup \mathrm{FV}(v)$.
- (Term substitution) If r, t are terms and x is a variable, then the term $r(x \mid t)$ obtained by *substituting t for x in r* is defined by induction in the usual way, with two new cases.
  1. If r is $G(u(\mathbf{0}) \cdots_x u(v))$, then $r(x \mid t)$ is
  $$G\big(u(\mathbf{0}) \cdots_x u(v(x \mid t))\big).$$
  2. If r is $G(u(\mathbf{0}) \cdots_y u(v))$ where $y \neq x$, then $r(x \mid t)$ is
  $$G\big(u(x \mid t)(\mathbf{0}) \cdots_y u(x \mid t)(v(x \mid t))\big).$$

Lemma 2.3 (Unique readability) *Assume that $\mathcal{L}$ has the following properties.*

1. *Every symbol of $\mathcal{L}$ is* exactly one *of the following: a left parenthesis, a right parenthesis, a logical connective, =, a constant symbol, a variable, an n-ary predicate symbol for some n, an n-ary function symbol for some n, a variadic function symbol, or an ellipsis $\cdots_x$ for some variable x.*
2. *If some symbol is an n-ary function (resp., predicate) symbol and also an m-ary function (resp., predicate) symbol, then $n = m$.*
3. *If some symbol is $\cdots_x$ and $\cdots_y$, then x and y are the same variable.*

Then the terms of $\mathcal{L}$ have the unique readability property.

Proof This is proved by the usual inductive argument. □

Henceforth, we will always assume that every language satisfies the hypotheses of Lemma 2.3. Thus the free variables of a term are well defined, as is term substitution.

Example 2.4 (Finite sigma notation) If $\mathcal{L}$ is a first-order language, we can extend it to a variadic language $\mathcal{L}_\Sigma$ by adding a variadic function symbol Σ (along with ellipses and numerals). In practice, the term $\Sigma(u(\mathbf{0}) \cdots_x u(v))$ is often written

$$\sum_{x=0}^{v} u.$$

The obvious interpretation $\Sigma^{\mathcal{M}}$ of Σ in a structure is the variadic addition function $\Sigma^{\mathcal{M}}(a_0, \ldots, a_n) = a_0 + \cdots + a_n$.

Throughout the paper, if $\mathcal{M}$ is a structure, then an *assignment* shall mean a function which maps variables to elements of the universe of $\mathcal{M}$. If s is an assignment and $n \in \mathbb{N}$, I will write $s(x \mid n)$ for the assignment which agrees with s everywhere except that it maps x to n.

We would like to define the interpretation of a term in a model by an assignment. This is straightforward in classic logic, but when variadic terms are introduced, interpretation becomes more subtle. There are actually two possible definitions; they are equivalent, but to show it, we will first need to establish a substitution lemma for one of the two.

When naively defining a numerical value for $\sum_{i=0}^{v} u(i)$, where $u(i)$ and v are mathematical expressions, we implicitly use a definition by recursion on expression complexity, as each summand $u(i)$ may itself involve nested summations. The process terminates because each summand $u(i)$ is strictly simpler than $\sum_{i=0}^{10} u(i)$, which is true because i itself is not a compound expression but a natural number. Now, we would like to say $\sum_{i=0}^{v} u(i) = u(0) + \cdots + u(v)$, where the summands on the right-hand side are recursively computed using the definition currently being made. But there are two ways to get here formally.

1. (Syntactic) Write a list of $v + 1$ terms, the nth of which is obtained by syntactically replacing the variable i in u by the constant n. Recursively compute and add each of these new terms, using the same values for variables as we are currently using.
2. (Semantic) Write a list of $v + 1$ terms, each of which is exactly u. Recursively compute and add them, but when computing the nth one, do it assuming the value of variable i is n.

This motivates the following definition of two interpretations. (In Corollary 3.6 we will see that both interpretations are equivalent.)

Definition 2.5 (Term interpretation) Let $\mathcal{M}$ be a structure for a variadic language $\mathcal{L}$, and let s be an assignment. Assume we have defined natural number interpretations $u^{s'}$, $u_{s'}$ (resp., *syntactic* and *semantic* interpretations of u) for every assignment s' and every term u strictly simpler than t. We define t^s and t_s inductively according to the following cases:

1. If t is a constant symbol c, then $t^s = t_s = c^{\mathcal{M}}$.
2. If t is a variable x, then $t^s = t_s = s(x)$.
3. If t is $f(u_1, \ldots, u_k)$, then $t^s = f^{\mathcal{M}}(u_1^s, \ldots, u_k^s)$ and $t_s = f^{\mathcal{M}}(u_{1s}, \ldots, u_{ks})$.

4. If t is $G(u(\mathbf{0}) \cdots_x u(v))$, then
$$G\big(u(\mathbf{0}) \cdots_x u(v)\big)^s = G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, \ldots, u(x \mid \overline{v^s})^s\big),$$
$$G\big(u(\mathbf{0}) \cdots_x u(v)\big)_s = G^{\mathcal{M}}\big(u_{s(x|0)}, \ldots, u_{s(x|v_s)}\big).$$
Here $\overline{v^s}$ denotes the constant symbol corresponding to the natural v^s.

Example 2.6 To illustrate the definition, assume $v^s = v_s = 5$. Then by definition,
$$\begin{aligned}
&G\big(u(\mathbf{0}) \cdots_x u(v)\big)^s\\
&\quad = G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, \ldots, u(x \mid \mathbf{5})^s\big)\\
&\quad = G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, u(x \mid \mathbf{1})^s, u(x \mid \mathbf{2})^s, u(x \mid \mathbf{3})^s, u(x \mid \mathbf{4})^s, u(x \mid \mathbf{5})^s\big),
\end{aligned}$$
$$\begin{aligned}
G\big(u(\mathbf{0}) \cdots_x u(v)\big)_s &= G^{\mathcal{M}}(u_{s(x|0)}, \ldots, u_{s(x|5)})\\
&= G^{\mathcal{M}}(u_{s(x|0)}, u_{s(x|1)}, u_{s(x|2)}, u_{s(x|3)}, u_{s(x|4)}, u_{s(x|5)}).
\end{aligned}$$

Remark 2.7 Definition 2.5 may seem somewhat suspect because of how it uses meta-ellipses to define the semantics of ellipses. If we were forbidden from using meta-ellipses to define the semantics of ellipses, there are two approaches we could take. One approach would be to use simultaneous induction to simultaneously define interpretations t^s and t_s and also define sequences $\langle t(x \mid \mathbf{0})^s, \ldots, t(x \mid \mathbf{n})^s\rangle$ and $\langle t_{s(x|0)}, \ldots, t_{s(x|n)}\rangle$. The latter would be defined by induction on n by means of *concatenation*:
$$\langle t(x \mid \mathbf{0})^s, \ldots, t(x \mid \overline{n+1})^s\rangle = \langle t(x \mid \mathbf{0})^s, \ldots, t(x \mid \overline{n})^s\rangle \frown \langle t(x \mid \overline{n+1})^s\rangle,$$
$$\langle t_{s(x|0)}, \ldots, t_{s(x|n+1)}\rangle = \langle t_{s(x|0)}, \ldots, t_{s(x|n)}\rangle \frown \langle t_{s(x|n+1)}\rangle,$$
assuming that $(t')^{s'}$ and $(t')_{s'}$ are already defined for every assignment s' and every term t' which is at most as complex as t; meanwhile, t^s and t_s would be defined as in Definition 2.5, except that we would let
$$G\big(u(\mathbf{0}) \cdots_x u(v)\big)^s = G^{\mathcal{M}}\big(\langle u(x \mid \mathbf{0})^s, \ldots, u(x \mid \overline{v^s})^s\rangle\big),$$
$$G\big(u(\mathbf{0}) \cdots_x u(v)\big)_s = G^{\mathcal{M}}(\langle u_{s(x|0)}, \ldots, u_{s(x|v_s)}\rangle).$$
This approach does not truly use meta-ellipses except as a name; the name could be changed without changing the definition. Another alternative approach would be to use *generalized structures*, which we will discuss in Section 6.

Remark 2.8 The syntactic part of Definition 2.5 relies on the fact that $u(x \mid \mathbf{c})$ is strictly simpler than $G(u(\mathbf{0}) \cdots_x u(v))$ for any constant symbol $\mathbf{c}$. The minimalist might wonder whether we can treat first-order variadic function symbols without so many constant symbols, using only the constant symbol $\mathbf{0}$ and the successor function symbol S. Maybe the natural way to translate the definition of $G(u(\mathbf{0}) \cdots_x u(v))^s$ would be to take Definition 2.5 using numerals $\mathbf{n} = SS \ldots S(\mathbf{0})$. But then $u(x \mid \mathbf{c})$ would no longer necessarily be simpler than $G(u(\mathbf{0}) \cdots_x u(v))$, casting doubt on the productiveness of the definition. One way around this dilemma would be to define term complexity not as a natural number but as an ordinal in ϵ_0, defining the complexity of $G(u(\mathbf{0}) \cdots_x u(v))$ to be, say, $\omega^{c(u)+c(v)}$, where $c(u)$ and $c(v)$ are the complexities of u and v. Of course, such a radical approach is not *necessary*, but it is more elegant than other solutions to the dilemma, and this author considers it a nice and unexpected application of ordinals to syntax.

3 The Substitution Lemma

We will deal mainly with syntactic interpretations t^s. We will obtain a substitution lemma for these and use it to show that the two interpretations are identical. The choice is arbitrary: one could also obtain a substitution lemma about semantic interpretations and use that to show equality. Once either version of the substitution lemma is obtained, and the two interpretations are shown to be equal, the other substitution lemma becomes trivial. In any case, technical lemmas are required.

Lemma 3.1 *Suppose u, t are terms and x, y are variables.*

1. *If x is not a free variable of u, then $u(x \mid t) = u$.*
2. *If x is not a free variable of u, and s is an assignment, then u^s does not depend on $s(x)$.*
3. *If x is not a free variable of u or t, then x is not a free variable of $u(y \mid t)$.*

Proof This is proved by a straightforward induction. □

In first-order logic, substitutability is a property of formulas, but it is not needed for terms: if r, t are any terms and x is any variable, then t is substitutable for x in r (in first-order logic). This breaks down in variadic logic, requiring a notion of substitutability into terms.

Definition 3.2 Fix a term t and a variable x. The substitutability of t for x in a term r is defined inductively:

- If r is a variable or constant symbol, then t is substitutable for x in r.
- If r is $f(u_1, \dots, u_n)$ where $u_1, \dots, u_n$ are terms and f is an n-ary or variadic function symbol, then t is substitutable for x in r if and only if t is substitutable for x in all the u_i.
- If r is $G(u(\mathbf{0}) \cdots_y u(v))$ where G is a variadic function symbol, u, v are terms, and y is a variable (which may or may not be x), then t is substitutable for x in r if at least one of the following holds:
  - x is not a free variable of r, or
  - $y = x$ and t is substitutable for x in v, or
  - y is not a free variable of t and t is substitutable for x in both u and v.

For a nonsubstitutability example, consider the term $\sum_{y=0}^{\mathbf{10}} x \cdot y$, and try substituting $t = y$ for x. The result is $\sum_{y=0}^{\mathbf{10}} y \cdot y$, which is no good since the new occurrence of y becomes bound by the summation.

We define substitutability for formulas in the usual way, with just one change: if p is a predicate symbol (or $=$), and $u_1, \dots, u_n$ are terms, we say t is *substitutable* for x in $p(u_1, \dots, u_n)$ if and only if t is substitutable for x in each u_i.

Lemma 3.3 *Suppose t is a term which is substitutable for the variable x in $G(u(\mathbf{0}) \cdots_y u(v))$. Then t is substitutable for x in v. And if $x \neq y$, then t is substitutable for x in u.*

Proof This is proved by induction. □

Lemma 3.4 *Suppose r, t are terms, x, y are distinct variables, and c is a constant symbol. If t is substitutable for x in r, and y does not occur free in t, then $r(x \mid t)(y \mid c) = r(y \mid c)(x \mid t)$.*

Proof This is proved by induction on complexity of r. If r is a constant symbol or $f(r_1, \ldots, r_n)$ for some function symbol f and terms $r_1, \ldots, r_n$ (wherein t is substitutable for x), the lemma is clear by induction. If r is a variable, the claim follows since y does not occur free in t. But suppose r is $G(u(\mathbf{0}) \cdots_z u(v))$ for some variadic function symbol G, terms u, v, and variable z (which may be x, y, or neither). Since t is substitutable for x in r, at least one of the following holds: x is not free in r; or $z = x$ and t is substitutable for x in v; or t is substitutable for x in u and v. If x is not free in r, then the lemma follows from Lemma 3.1. But suppose x is free in r. Unraveling definitions, we have the following table:

If $z\ldots$	Then $r(x \mid t)$ equals...
$= x$	$G(u(\mathbf{0}) \cdots_z u(v(x \mid t)))$
$\neq x$	$G(u(x \mid t)(\mathbf{0}) \cdots_z u(x \mid t)(v(x \mid t)))$
If $z\ldots$	Then $r(x \mid t)(y \mid c)$ equals...
$= x$	$G(u(y \mid c)(\mathbf{0}) \cdots_z u(y \mid c)(v(x \mid t)(y \mid c)))$
$= y$	$G(u(x \mid t)(\mathbf{0}) \cdots_z u(x \mid t)(v(x \mid t)(y \mid c)))$
$\notin \{x, y\}$	$G(u(x \mid t)(y \mid c)(\mathbf{0}) \cdots_z u(x \mid t)(y \mid c)(v(x \mid t)(y \mid c)))$

By Lemma 3.3, t is substitutable for x in v, so $v(x \mid t)(y \mid c) = v(y \mid c)(x \mid t)$ by induction. And if $z \neq x$, then Lemma 3.3 tells us that t is substitutable for x in u as well, and so by induction $u(x \mid t)(y \mid c) = u(y \mid c)(x \mid t)$. The lemma follows by using these facts to rewrite the last row of the table and compare with a similar table for $r(y \mid c)(x \mid t)$. □

Theorem 3.5 (The variadic substitution lemma for terms) *Let $\mathcal{M}$ be a structure for a variadic language $\mathcal{L}$, and let s be an assignment. If r and t are terms such that t is substitutable for x in r, then $r(x \mid t)^s = r^{s(x|t^s)}$.*

Proof We induct on the complexity of r, and most cases are straightforward. If x is not free in r, the claim is trivial; assume x is free in r. The two important cases follow.

We must show $G(u(\mathbf{0}) \cdots_y u(v))(x \mid t)^s = G(u(\mathbf{0}) \cdots_y u(v))^{s(x|t^s)}$ when y is a different variable than x and t is substitutable for x in $G(u(\mathbf{0}) \cdots_y u(v))(x \mid t)$. Using induction,

$$\begin{aligned}
&G\big(u(\mathbf{0}) \cdots_y u(v)\big)(x \mid t)^s \\
&\quad = G\big(u(x \mid t)(\mathbf{0}) \cdots_y u(x \mid t)(v(x \mid t))\big)^s \\
&\quad = G^{\mathcal{M}}\big(u(x \mid t)(y \mid \mathbf{0})^s, \ldots, u(x \mid t)(y \mid \overline{v(x \mid t)^s})^s\big) \\
&\quad = G^{\mathcal{M}}\big(u(y \mid \mathbf{0})(x \mid t)^s, \ldots, u(y \mid \overline{v(x \mid t)^s})(x \mid t)^s\big) \qquad (*) \\
&\quad = G^{\mathcal{M}}\big(u(y \mid \mathbf{0})^{s(x|t^s)}, \ldots, u(y \mid \overline{v^{s(x|t^s)}})^{s(x|t^s)}\big) \\
&\quad = G\big(u(\mathbf{0}) \cdots_y u(v)\big)^{s(x|t^s)}.
\end{aligned}$$

To reach line $(*)$, we need the fact that $u(x \mid t)(y \mid c) = u(y \mid c)(x \mid t)$ for any constant symbol c. If x is not free in r, then it is not free in u (since $y \neq x$), and so this follows from Lemma 3.1. Otherwise, since t is substitutable for x in r by Lemma 3.3, we must have that y does not occur free in t, and so we can invoke Lemma 3.4.

We must also show $G(u(\mathbf{0})\cdots_x u(v))(x \mid t)^s = G(u(\mathbf{0})\cdots_x u(v))^{s(x|t^s)}$ when t is substitutable for x in $G(u(\mathbf{0})\cdots_x u(v))(x \mid t)$. Using induction,

$$\begin{aligned} G\big(u(\mathbf{0})\cdots_x u(v)\big)(x \mid t)^s &= G\big(u(\mathbf{0})\cdots_x u(v(x \mid t))\big)^s \\ &= G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, \ldots, u(x \mid \overline{v(x \mid t)^s})^s\big) \\ &= G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, \ldots, u(x \mid \overline{v^{s(x|t^s)}})^s\big) \\ &= G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^{s(x|t^s)}, \ldots, u(x \mid \overline{v^{s(x|t^s)}})^{s(x|t^s)}\big) \quad (**) \\ &= G\big(u(\mathbf{0})\cdots_x u(v)\big)^{s(x|t^s)}. \end{aligned}$$

In line (**), I am able to change "exponents" from s to $s(x \mid t^s)$ because the terms in question do not depend on x. □

Corollary 3.6 *For any term t and assignment s, $t^s = t_s$.*

Proof This is proved by induction on t. All cases are immediate except the case when t is $G(u(\mathbf{0})\cdots_x u(v))$. Note that constant symbols are always substitutable, and write

$$\begin{aligned} G\big(u(\mathbf{0})\cdots_x u(v)\big)^s &= G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, \ldots, u(x \mid \overline{v^s})^s\big) \quad \text{(by definition)} \\ &= G^{\mathcal{M}}(u^{s(x|0)}, \ldots, u^{s(x|v^s)}) \quad \text{(by Theorem 3.5)} \\ &= G^{\mathcal{M}}(u_{s(x|0)}, \ldots, u_{s(x|v_s)}) \quad \text{(by induction)} \\ &= G\big(u(\mathbf{0})\cdots_x u(v)\big)_s \quad \text{(by definition).} \end{aligned}$$

□

First-order formulas over a variadic language are now defined in the obvious way. By Corollary 3.6, we can define $\mathcal{M} \models t = r[s]$ if and only if $t^s = r^s$ or, equivalently, $t_s = r_s$; that is, we are saved from having to make an arbitrary choice. The remaining semantics are defined inductively in exactly the same way they are for first-order logic. If $\mathcal{M}$ is a structure, s an assignment, and φ a formula, then $\mathcal{M} \models \varphi[s]$ is defined in the usual way from the above atomic case, and $\mathcal{M} \models \varphi$ means that $\mathcal{M} \models \varphi[s]$ for every assignment s. Term substitution in a formula is defined as usual. Substitutability of a term for a variable in a formula is defined as usual, except that in the atomic case, we say t is *substitutable* for x in $r = q$ if and only if t is substitutable for x in r and q (in the sense of Definition 3.2).

Corollary 3.7 (The variadic substitution lemma) *If t is a term which is substitutable for the variable x in the formula φ, and s is an assignment and $\mathcal{M}$ a structure, then $\mathcal{M} \models \varphi(x \mid t)[s]$ if and only if $\mathcal{M} \models \varphi[s(x \mid t^s)]$.*

Proof The proof is identical to the proof of the first-order substitution lemma, except that Theorem 3.5 is invoked for the atomic case. □

Example 3.8 Working in an appropriate language and structure, it can be shown that

$$\sum_{x=0}^{x} x = x(x + \mathbf{1})/\mathbf{2},$$

showing that it is safe to use the same variable in different roles, so long as we use Definition 2.5 to be completely clear what the truth of the formula means.

4 Bounded Quantifier Elimination

In this section, we shall assume our languages have no predicate symbols. If a language has a binary function symbol $\leq$ and a constant symbol $\mathbf{1}$, I will write $u \leq v$ to abbreviate $\leq (u, v) = \mathbf{1}$.

Definition 4.1

- If $\mathcal{L}$ is a variadic language, the *quantifier-free* formulas of $\mathcal{L}$ are defined inductively: φ is quantifier-free whenever φ is atomic; and if φ and ψ are quantifier-free, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$, and $\neg\varphi$.
- If $\mathcal{L}$ contains a binary function symbol $\leq$ and constant symbol $\mathbf{1}$, the *unbounded quantifier-free* (or uqf) formulas of $\mathcal{L}$ are defined inductively: φ is uqf whenever φ is atomic; and if φ and ψ are uqf and x, y are distinct variables, then $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$, $\neg\varphi$, $\exists x\ (x \leq y \wedge \varphi)$, and $\forall x\ (x \leq y \rightarrow \varphi)$ are also uqf.

Proposition 4.2 (Bounded quantifier elimination) *Suppose $\mathcal{L}$ is a variadic language containing (possibly among other things) binary function symbols $\leq$, $+$, and δ, and a variadic function symbol G. Suppose $\mathcal{M}$ is an $\mathcal{L}$-model which interprets $+$ as addition and interprets $\leq$, δ, and G by*

$$\leq^{\mathcal{M}} (m, n) = \begin{cases} 1 & \text{if } m \leq n, \\ 0 & \text{otherwise,} \end{cases} \qquad \delta^{\mathcal{M}}(m, n) = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{otherwise,} \end{cases}$$

$$G^{\mathcal{M}}(m_0, \ldots, m_n) = \begin{cases} 1 & \text{if } m_i \neq 0 \text{ for some } 0 \leq i \leq n, \\ 0 & \text{otherwise.} \end{cases}$$

For any uqf $\mathcal{L}$-formula φ, there is a quantifier-free $\mathcal{L}$-formula ψ, with the same free variables as φ, such that $\mathcal{M} \models \varphi \leftrightarrow \psi$.

Proof I will show more strongly that, for any uqf formula φ, there is a term t_φ, with exactly the free variables of φ, such that $\mathcal{M} \models \varphi \leftrightarrow (t_\varphi = \mathbf{1})$ and $\mathcal{M} \models \neg\varphi \leftrightarrow (t_\varphi = \mathbf{0})$. This is proved by induction on φ.

- If φ is $u = v$, take $t_\varphi = \delta(u, v)$.
- If φ is $\psi \wedge \rho$, take $t_\varphi = \delta(t_\psi + t_\rho, \mathbf{2})$.
- If φ is $\neg\psi$, take $t_\varphi = \delta(t_\psi, \mathbf{0})$.
- If φ is $\exists x\ (x \leq y \wedge \psi)$ where $x \neq y$, take $t_\varphi = G(t_\psi(\mathbf{0}) \cdots_x t_\psi(y))$.
- All other cases for φ are reduced to the above by basic logic. (There is no predicate case by assumption.)

In all but the $\exists$-case, it is routine to check $\mathcal{M} \models \varphi \leftrightarrow (t_\varphi = \mathbf{1})$, $\mathcal{M} \models \neg\varphi \leftrightarrow (t_\varphi = \mathbf{0})$. The $\exists$-case goes as follows. Assume φ is $\exists x\ (x \leq y \wedge \psi)$, $y \neq x$. By induction, $\mathcal{M} \models \psi \leftrightarrow (t_\psi = \mathbf{1})$ and $\mathcal{M} \models \neg\psi \leftrightarrow (t_\psi = \mathbf{0})$. So $\mathcal{M} \models \psi \leftrightarrow (t_\psi = \mathbf{1})[s]$ for every assignment s. Let s be an assignment. Then

$$\begin{aligned} \mathcal{M} &\models \varphi[s] \text{ iff} \\ \mathcal{M} &\models \exists x\ (x \leq y \wedge \psi)[s] \text{ iff} \\ \exists n \in \mathbb{N} \text{ s.t. } \mathcal{M} &\models x \leq y \wedge \psi[s(x \mid n)] \text{ iff} \\ \exists n \leq s(y) \text{ s.t. } \mathcal{M} &\models t_\psi = \mathbf{1}[s(x \mid n)] \text{ iff} \qquad (*) \\ \exists n \leq s(y) \text{ s.t. } & t_\psi^{s(x|n)} = 1 \text{ iff} \end{aligned}$$

$$G^{\mathcal{M}}(t_\psi^{s(x|0)},\ldots,t_\psi^{s(x|s(y))}) = 1 \text{ iff}$$
$$G^{\mathcal{M}}\big(t_\psi(x \mid \mathbf{0})^s,\ldots,t_\psi(x \mid \overline{y^s})^s\big) = 1 \text{ iff} \qquad (**)$$
$$\mathcal{M} \models G\big(t_\psi(\mathbf{0}) \cdots_x t_\psi(y)\big) = \mathbf{1}[s].$$

In line $(*)$ we use the fact that $s(x \mid n)(y) = s(y)$ since $y \neq x$. In line $(**)$ we invoke the variadic substitution lemma (noting that constant symbols are always substitutable for x). □

Corollary 4.3 *Let $\mathcal{L}$ be the language with constant symbols $\mathbf{n}$ for all $n \in \mathbb{N}$, binary function symbols $+$, $\cdot$, δ, and $\leq$, and a variadic function symbol G. Let $\mathcal{M}$ be the model which interprets everything in the obvious way (interpreting δ and G as above). A set $X \subseteq \mathbb{N}$ is computably enumerable if and only if there is a quantifier-free $\mathcal{L}$-formula φ, with free variables a subset of $\{x, y\}$, such that for all $n \in \mathbb{N}$, $n \in X \leftrightarrow \mathcal{M} \models \exists y\varphi(x \mid \mathbf{n})$.*

Proof Let $\mathcal{L}_0 = \mathcal{L}\backslash\{G\}$ be the first-order part of $\mathcal{L}$. Assume $X \subseteq \mathbb{N}$ is c.e. By computability theory, there is a uqf formula φ_0 of $\mathcal{L}_0$ with the desired properties. The corollary follows by bounded quantifier elimination. The converse is clear by Church's thesis. □

5 Defining Borel Sets

I will further extend the concept of variadic function symbols and apply the idea to show that a certain language can define any $\mathbf{\Sigma}^0_n$- or $\mathbf{\Pi}^0_n$-subset of $\mathbb{N}^\mathbb{N}$ with a formula of complexity Σ_n or Π_n, respectively, in a rather nice way. My interest in using powerful language to define Borel sets is partially influenced by Vanden Boom [13, pp. 276–77].

By an *extended variadic language* I mean a first-order language together with various *n-ary-by-variadic* function symbols (for various $n \geq 0$), as well as constant symbols $\mathbf{n}$ for every $n \in \mathbb{N}$ and ellipses $\cdots_x$. A structure for an extended variadic language is a structure $\mathcal{M}$ for the first-order part, with universe $\mathbb{N}$ and which interprets each $\mathbf{n}$ as n, together with a function $G^{\mathcal{M}} : \mathbb{N}^n \times \mathbb{N}^{<\mathbb{N}} \to \mathbb{N}$ for every n-ary-by-variadic function symbol G. Terms, term substitution, term interpretation, and term substitutability are defined in ways very similar to our work in Section 2, and the variadic substitution lemma is proved in almost an identical way.

Definition 5.1 Let $\langle\rangle$ be the empty sequence.

- By $\mathcal{L}_{\text{Bor}}$ I mean the extended variadic language with a special unary function symbol $\mathbf{f}$ along with, for every $n > 0$ and every $\iota : \mathbb{N}^n \to \mathbb{N}^{<\mathbb{N}}\backslash\{\langle\rangle\}$, an n-ary function symbol ℓ_ι and an n-ary-by-variadic function symbol τ_ι.
- For any $f : \mathbb{N} \to \mathbb{N}$, $\mathcal{M}_f$ is the $\mathcal{L}_{\text{Bor}}$ structure which interprets $\mathbf{f}$ as f and which, for any $n > 0$ and $\iota : \mathbb{N}^n \to \mathbb{N}^{<\mathbb{N}}\backslash\{\langle\rangle\}$, interprets

$$\ell_\iota^{\mathcal{M}_f}(a_1,\ldots,a_n) = \text{the length of } \iota(a_1,\ldots,a_n), \text{ minus } 1,$$
$$\tau_\iota^{\mathcal{M}_f}(a_1,\ldots,a_n,b_1,\ldots,b_m) = \begin{cases} 1 & \text{if } (b_1,\ldots,b_m) = \iota(a_1,\ldots,a_n), \\ 0 & \text{otherwise.} \end{cases}$$

- If φ is an $\mathcal{L}_{\text{Bor}}$-sentence and $S \subseteq \mathbb{N}^\mathbb{N}$, say that φ *defines* S if, for every $f : \mathbb{N} \to \mathbb{N}$, $f \in S$ if and only if $\mathcal{M}_f \models \varphi$.

Theorem 5.2 *Let $n > 0$, and let $S \subseteq \mathbb{N}^{\mathbb{N}}$. Then S is $\mathbf{\Sigma}^0_n$ (resp., $\mathbf{\Pi}^0_n$) if and only if S is defined by a Σ_n (resp., Π_n) sentence of $\mathcal{L}_{\text{Bor}}$.*

Proof This is obvious if $S = \emptyset$ or $S = \mathbb{N}^{\mathbb{N}}$; assume not. If f_0 is a finite sequence of naturals, I will write $[f_0]$ for the set of infinite extensions of f_0.

($\Rightarrow$) Assume S is $\mathbf{\Sigma}^0_n$. If n is odd, we can write $S = \bigcup_{i_1\in\mathbb{N}}\cdots\bigcup_{i_n\in\mathbb{N}}[f_{i_1\cdots i_n}]$ where the $\{f_{i_1\cdots i_n}\}$ are finite, nonempty sequences. If n is even, we can write $S = \bigcup_{i_1\in\mathbb{N}}\cdots\bigcap_{i_n\in\mathbb{N}}[f_{i_1\cdots i_n}]^c$. Let $\iota : \mathbb{N}^n \to \mathbb{N}^{<\mathbb{N}}\backslash\{\langle\rangle\}$ be the map which sends $(i_1,\ldots,i_n)$ to $f_{i_1\cdots i_n}$. Let $f : \mathbb{N} \to \mathbb{N}$. For any $(i_1,\ldots,i_n)$, f extends $f_{i_1\cdots i_n}$ if and only if $\tau_\iota^{\mathcal{M}_f}\big(i_1,\ldots,i_n, f(0),\ldots,f(\ell_\iota^{\mathcal{M}_f}(i_1,\ldots,i_n))\big) = 1$. So if n is odd, then $f \in S$ iff

$$\mathcal{M}_f \models \exists x_1\cdots\exists x_n \tau_\iota\big(x_1,\ldots,x_n,\mathbf{f}(z)(\mathbf{0})\cdots_z \mathbf{f}(z)(\ell_\iota(x_1,\ldots,x_n))\big) = \mathbf{1}.$$

And if n is even, then $f \in S$ iff

$$\mathcal{M}_f \models \exists x_1\cdots\forall x_n \tau_\iota\big(x_1,\ldots,x_n,\mathbf{f}(z)(\mathbf{0})\cdots_z \mathbf{f}(z)(\ell_\iota(x_1,\ldots,x_n))\big) = \mathbf{0}.$$

The $\mathbf{\Pi}^0_n$-case is similar.

($\Leftarrow$) By induction on n. For the base case, first use an induction argument on formula complexity to show that if φ is a quantifier-free sentence of $\mathcal{L}_{\text{Bor}}$ and $\mathcal{M}_f \models \varphi$, then there is some k so big that whenever $g : \mathbb{N} \to \mathbb{N}$ extends $(f(0),\ldots,f(k))$, then $\mathcal{M}_g \models \varphi$. Thus a set defined by a quantifier-free formula is open, and hence clopen since its complement is also defined by that formula's negation. The base case follows: for example, if S is defined by a sentence $\exists x\varphi$, then (by variadic substitution) $S = \bigcup_{i\in\mathbb{N}}\{g : \mathbb{N} \to \mathbb{N} : \mathcal{M}_g \models \varphi(x \mid \mathbf{i})\}$, a union of clopen sets, showing that S is $\mathbf{\Sigma}^0_1$. The induction case is straightforward. □

6 A Partial Mechanization

We partially automate Sections 2 and 3 using the Coq proof assistant (see Alexander [2]). In Coq, it is easier to work with the *semantic*, rather than the *syntactic*, term interpretations of Definition 2.5. This is because semantic term interpretation is recursive in a direct structural way: to interpret a term, one need only interpret direct subterms. This is in contrast with syntactic term interpretation, which is recursive in term depth. To syntactically interpret a term, one must interpret terms which are not direct subterms. This makes it much more tedious to automate proofs about syntactic interpretations, so our automation primarily deals with semantic interpretations. We do, however, automate Corollary 3.6, in light of which the distinction disappears.

Very often when automating mathematics, it is actually easier to prove a stronger result. This is certainly the case here. By a *generalized structure* $\mathcal{M}$ for a variadic language $\mathcal{L}$ we mean a structure for the first-order part of $\mathcal{L}$, together with a set of interpretations $G^{\mathcal{M}} : \mathbb{N}^{\mathbb{N}} \times \mathbb{N} \to \mathbb{N}$ for each variadic function symbol G of $\mathcal{L}$. This is a generalization in an obvious way: given a structure $\mathcal{M}'$ as in Section 2, there corresponds a generalized structure $\mathcal{M}$ which agrees with $\mathcal{M}'$ on the first-order part of $\mathcal{L}$ and is otherwise defined by

$$G^{\mathcal{M}}(f,v) = G^{\mathcal{M}'}\big(f(0),\ldots,f(v)\big)$$

whenever G is a variadic function symbol, $v \in \mathbb{N}$, and $f \in \mathbb{N}^{\mathbb{N}}$. The syntactic and semantic interpretations in $\mathcal{M}$ of a term $G(u(\mathbf{0})\cdots_x u(v))$ by an assignment s are, respectively,

$$G\big(u(\mathbf{0})\cdots_x u(v)\big)^s = G^{\mathcal{M}}\big(k \mapsto u(x \mid \overline{k})^s, v^s\big),$$
$$G\big(u(\mathbf{0})\cdots_x u(v)\big)_s = G^{\mathcal{M}}(k \mapsto u_{s(x|k)}, v_s).$$

All the results of Sections 2 and 3 generalize accordingly. It is easier to automate these stronger results because Coq has better built-in support for working with functions $\mathbb{N} \to \mathbb{N}$ than for working with finite sequences.

For even further simplicity, we also assume that all functions are either variadic or binary, we assume the special constant symbols $\overline{c}$ are the only constant symbols in the language, and we assume there are no predicate symbols.

Syntactic term interpretation seems to lie on the border of what Coq can handle. Coq cannot automatically detect that the definition is total. We are able to convince Coq of its totality using an experimental feature of Coq called Program Fixpoint (see Sozeau [12]). Chung-Kil Hur [8], [9] helped us tremendously with the details of getting Program Fixpoint to work.

In performing this partial mechanization, we were influenced by R. O'Connor's (see [11]) mechanization of ordinary first-order logic.

7 Future Work

There are several directions to take this study from here. For one thing, Sections 2 and 3 could easily be extended to other types of logic. In order to inject variadic terms into a logic, there are two basic requirements: first, that function terms make any sense at all in that logic; second, that the logic has a semantics which plays well with variadic function symbols, especially the ellipsis. Some potential logics where we could add variadic function symbols include second-order logic, more general multisorted logic, and nominal logic, just to name three. The question is not so much whether the machinery can be added to the logic, but rather, what interesting applications result.

In the direction of multisorted logic, we could deal with semantics where one sort ranges over (say) $\mathbb{R}$ and another ranges over $\mathbb{N}$, and thereby rigorously study variadic functions living in the real numbers. (Single-sorted first-order logic falls short here: how are we to interpret a term like $\sum_{i=0}^{\pi} i$?)

One of the shortcomings of this first-order treatment is that we were not able to give what should be a basic example: the general Apply function from computer science. If $G : \mathbb{N}^{<\mathbb{N}} \to \mathbb{N}$ is a variadic function and $n_1, \ldots, n_k \in \mathbb{N}$, then $\mathrm{Apply}(G, n_1, \ldots, n_k)$ is defined to be $G(n_1, \ldots, n_k)$. This could be formalized using our variadic machinery in various typed logics where it makes sense to have a function symbol whose "arity" is some Cartesian product of types.

Another direction we can go from here is to consider function symbols of *infinite arity*. The basic idea is that if G is an infinitary function symbol in a language and u is a term and x a variable, then $G(u(\mathbf{0})\cdots_x)$ is another term, whose intended interpretation by a model $\mathcal{M}$ and assignment s is

$$G\big(u(\mathbf{0})\cdots_x\big)^s = G^{\mathcal{M}}\big(u(x \mid \mathbf{0})^s, u(x \mid \mathbf{1})^s, \ldots\big),$$

where G itself is interpreted as some infinitary $G^{\mathcal{M}} : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$. In fact, much of the work needed for this is already done in the Coq mechanization of Section 6. The reason that this direction would be exciting is that the bounded-quantifier elimination of Section 4 could be strengthened to full quantifier elimination.

Finally, we are interested in embedding the hydra game of Kirby and Paris [10] (a short and very readable introduction is given by Bauer [3]) into term interpretation. A binary operator $+$ is (left and right) *self-distributive* if it satisfies $a + (b + c) = (a + b) + (a + c)$ and $(a + b) + c = (a + c) + (b + c)$. (Self-distributive operators were studied by Frink [6] and more recently (left-sided only) by set theorists and knot theorists, as surveyed by Dehornoy [5].) For such an operator (assuming also associativity),

$$\begin{aligned}
&\sum_{i_1=0}^{v_1} t_1 + \cdots + \sum_{i_k=0}^{v_k} t_k \\
&\quad = \sum_{i_1=0}^{v_1} t_1 + \cdots + \sum_{i_{k-1}=0}^{v_{k-1}} t_{k-1} + t_k(i_k \mid 0) + \cdots + t_k(i_k \mid v_k) \\
&\quad = \Big(\sum_{i_1=0}^{v_1} t_1 + \cdots + \sum_{i_{k-1}=0}^{v_{k-1}} t_{k-1} + t_k(i_k \mid 0)\Big) \\
&\qquad + \cdots + \Big(\sum_{i_1=0}^{v_1} t_1 + \cdots + \sum_{i_{k-1}=0}^{v_{k-1}} t_{k-1} + t_k(i_k \mid v_k)\Big),
\end{aligned}$$

which bears a certain resemblance to the act of cutting a hydra's head and having many isomorphic copies of its subtree regrow.

References

[1] Alexander, S., "On guessing whether a sequence has a certain property," *Journal of Integer Sequences*, vol. 14 (2011), no. 11.4.4. MR 2792160. 47

[2] Alexander, S., "Mechanization of the first-order syntax of variadic functions in Coq," letter to the referees of this paper, December 2011. 56

[3] Bauer, A., "The hydra game," 2008, http://math.andrej.com/2008/02/02/the-hydra-game/ 58

[4] Byrd, W., and D. Friedman, "From variadic functions to variadic relations: A miniKanren perspective," pp. 105–117 in *Scheme and Functional Programming (Portland, Ore., 2006)*, University of Chicago Technical Report TR-2006-06, University of Chicago Department of Computer Science, Chicago, 2006. 48

[5] Dehornoy, P., "Braids and Self-Distributivity," vol. 192 of *Progress in Mathematics*, Birkhäuser, Basel, 2000. MR 1778150. 58

[6] Frink, O., "Symmetric and self-distributive systems," *American Mathematical Monthly*, vol. 62 (1955), pp. 697–707. MR 0073606. 58

[7] Goldberg, M., "A variadic extension of Curry's fixed-point combinator," *Higher Order Symbolic Computation*, vol. 18 (2005), pp. 371–88. 48

[8] Hur, C., "Re: Program Fixpoints not unfolding correctly," email to the Coq-Club mailing list, 2011. 57

[9] Hur, C., personal correspondence, November 2011. 57

[10] Kirby, L., and J. Paris, "Accessible independence results for Peano arithmetic," *Bulletin of the London Mathematical Society*, vol. 14 (1982), pp. 285–93. MR 0663480. 58

[11] O'Connor, R., "Essential incompleteness of arithmetic verified by Coq," pp. 245–60 in *Theorem Proving in Higher Order Logics*, vol. 3603 of *Lecture Notes in Computer Science*, Springer, Berlin, 2005. MR 2197012. 57

[12] Sozeau, M., "Program," Chapter 22 in *The Coq Proof Assistant Reference Manual*, version 8.3, available at http://coq.inria.fr/doc/index.html (accessed October 16, 2012). 57

[13] Vanden Boom, M., "The effective Borel hierarchy," *Fundamenta Mathematicae*, vol. 195 (2007), pp. 269–89. Zbl 1125.03035. MR 2338544. 55

Department of Mathematics
The Ohio State University
231 West 18th Avenue
Columbus, Ohio 43210
USA
alexander@math.ohio-state.edu
http://www.math.osu.edu/~alexander/