Another characterization of provably recursive functions

Evgeny Makarov
emakarov@gmail.com

Abstract. A new characterization of provably recursive functions of first-order arithmetic is described. Its main feature is using only terms consisting of 0, the successor S and variables in the quantifier rules, namely, universal elimination and existential introduction.

1 Introduction

This paper presents a new characterization of provably recursive functions of first-order arithmetic. We consider functions defined by sets of equations. The equations can be completely arbitrary, not necessarily defining primitive recursive, or even total, functions. The main result states that a function is provably recursive iff its totality is provable (using natural deduction) from the defining set of equations, with one restriction: only terms consisting of 0, the successor S and variables can be used in the inference rules dealing with quantifiers, namely universal elimination and existential introduction.

A deduction system with such restrictions can be considered as a way of reasoning about non-denoting terms. A set of equations \( P \) can define non-total functions over natural numbers (for precise definitions see Sect. 2) and a deduction system with regular quantifier rules has quantified variables ranging over all, not necessarily denoting, terms. For example, a formula \( \forall x \exists y \ f(x) = y \) is trivially provable in a regular system regardless of the definition of \( f \): we start by \( f(x) = f(x) \), introduce the existential quantifier to get \( \exists y \ f(x) = y \) and the universal quantifier to get \( \forall x \exists y \ f(x) = y \). In contrast, having the restriction given above makes quantifiers range over terms denoting natural numbers. The result is that the formula \( \forall x \exists y \ f(x) = y \) is no longer trivially provable because existential introduction with the term \( f(x) \) is not allowed in general. Moreover, its provability implies that the totality of \( f \) is provable in first-order arithmetic, i.e., that \( f \) is provably recursive, as the main result shows. Thus, the deductive power of our system is similar to the one of the standard Peano Arithmetic.

Our presentation is heavily influenced by [2], where another framework for reasoning about non-denoting terms, called intrinsic theory, is defined. The intrinsic theory of natural numbers has a unary predicate symbol \( N \) which is supposed to mean that its argument is a natural number. Intrinsic theory has no restriction on the quantifier rules. In effect, quantifiers in our system are equivalent to quantifiers relativized to \( N \) in intrinsic theory. This fact is used to prove one direction of our main result. The other direction is also proved following
the reasoning of a similar statement in [2], but without a reference to intrinsic
typeeory.

2 Definitions

Let \( P \) be a set of first-order equations. Let \( \mathcal{L} \) be the language of \( P \) plus a constant
0 and a unary functional symbol \( S \) (if they are not already used in \( P \)). The theory
\( \mathbf{A}[P] \) is a first-order theory with equality in the language \( \mathcal{L} \). The axioms of \( \mathbf{A}[P] \)
are the universal closures of the equations in \( P \), denoted by \( \forall P \), the separation
axioms \( \forall x S(x) \neq 0 \) and \( \forall x, y S(x) = S(y) \rightarrow x = y \), and induction

\[ A[0] \rightarrow \forall x (A[x] \rightarrow A[S(x)]) \rightarrow \forall x A[x] \]

for all formulas \( A \) in \( \mathcal{L} \). The inference rules are the usual rules of classical natural
deduction (see, e.g., [3]) plus the rules of equality:

\[
\begin{align*}
A[t] & \quad t = s \\
\hline
A[s] & \\
\end{align*}
\]

\[ t = t \]

for all formulas \( A \) and terms \( t, s \) in \( \mathcal{L} \) (\( A[s] \) is obtained from \( A[t] \) by replacing
some occurrences of \( t \) by \( s \)). The natural deduction rules dealing with quantifiers
are shown in Fig. 1. It is easy to see that the rules of equality make it a congruence.
For example, let \( \mathbf{AM} \) be the usual axioms for addition and multiplication
and let \( \mathbf{PR} \) be the set of standard defining equations for all primitive recursive
functions. Then \( \mathbf{A}[\mathbf{AM}] \) is Peano Arithmetic and \( \mathbf{A}[\mathbf{PR}] \) is Peano Arithmetic
with all primitive recursive functional symbols.

\[
\begin{align*}
\frac{A[y]}{\forall x A[x]} \quad (\forall I) & \quad \frac{\forall x A[x]}{A[t]} \quad (\forall E) \\
y \text{ is not free in open assumptions} & \quad t \text{ is free for } x \text{ in } A
\end{align*}
\]

\[
\begin{align*}
\frac{A[t]}{\exists x A[x]} \quad (\exists I) & \quad \frac{\exists x A[x]}{C} \quad (\exists E) \\
t \text{ is free for } x \text{ in } A & \quad y \text{ is not free in } C
\end{align*}
\]

Fig. 1. Quantifier rules of natural deduction

A \textit{program} is a pair \((P, f)\) consisting of a set of equations \( P \) and a functional
symbol \( f \) occurring in \( P \). (When \( f \) is clear from the context or is irrelevant, we
will write \( P \) instead of \((P, f)\).)
We use programs to define functions using an analog of Herbrand-Gödel computability (see [1, 2]). Given a program \( P \), we write \( P \vdash E \) if \( E \) is an equation derivable from \( P \) in equational logic. The rules of equational logic are the following:

1. \( P \vdash E \) for every \( E \in P \);
2. \( P \vdash t = t \) for every term \( t \);
3. if \( P \vdash E[x] \), then \( P \vdash E[t] \) for every term \( t \) and a variable \( x \);
4. if \( P \vdash s[t] = r[t] \) and \( P \vdash t = t' \), then \( P \vdash s[t'] = r[t'] \).

The relation computed by \( (P, f) \) is \( \{ (\bar{n}, m) \mid P \vdash f(\bar{n}) = m \} \) (as usual, \( \bar{n} \) is a numeral for a number \( n \), consisting of \( n \) occurrences of \( S \) applied to \( 0 \)). This relation does not have to be a function. Let us call \( P \) coherent if \( P \nvdash \bar{m} = \bar{n} \) for two distinct numerals \( \bar{m} \) and \( \bar{n} \). It is easy to see that the relation computed by a coherent program is a partial function.

However, even for a coherent program \( P \) the theory \( A[P] \) can be inconsistent because of the separation axioms. This is the case, for example, for \( P = \{ f(g(0)) = S(g(0)), f(x) = g(0) \} \) with fresh functional symbols \( f \) and \( g \). Call a program \( P \) strongly coherent if \( A[P] \) is consistent. It is clear that if a program is strongly coherent, then it is coherent.

Later it will be important that a program containing a functional symbol \( f \) corresponding to a primitive recursive function \( f \) also contains all defining equations for \( f \). Programs that satisfy this property are called full.

A term \( t \) is called function-free if \( t \) consists of \( 0, S \) and variables only. A term \( t \) is called primitive recursive if \( t \) is in the language of PR. If \( T \) is a theory, then a formula \( A \) is called provable with function-free terms (respectively, provable with primitive recursive terms) in \( T \) if there is a classical natural deduction derivation of \( A \) from \( T \) where the eigenterms of the rules of universal elimination and existential introduction (i.e., terms \( t \) in the rules \( (\forall E) \) and \( (\exists I) \) in Fig. 1) are function-free (respectively, primitive recursive). Formulas provable with function-free (primitive recursive) terms in \( T \) are also called ff-provable (pr-provable) in \( T \), and this is denoted \( T \nvdash A \) \( (T \nvdash A) \). More generally, if there is a natural deduction derivation in \( T \) of a formula \( A \) from assumptions \( \Gamma \) with the above restrictions on quantifiers, this is denoted by \( T \nvdash \Gamma \Rightarrow A \) or \( T \nvdash \Gamma \Rightarrow A \).

A function \( f \) is called ff-provable if \( f \) is computed by a strongly coherent full program \( (P, f) \) and \( A[P] \nvdash \forall x \exists y f(x) = y \), and similarly for pr-provable.

### 3 Provable recursive functions are ff-provable

We choose the following definition of provably recursive functions of a theory \( T \): \( f \) is called provably recursive if \( f(x) = h(\mu y g(x, y) = 0) \) where \( \mu \) denotes the minimization operator, \( g \) and \( h \) are primitive recursive and \( T \vdash \forall x \exists y g(x, y) = 0 \).

The proof of the claim that every provably recursive function is ff-provable uses several lemmas.

**Lemma 1.** Suppose \( f \) is a primitive recursive function and \( f \) is the corresponding functional symbol from PR. Then \( A[PR] \nvdash \forall x \exists y f(x) = y \).
Proof. By induction on the construction of \( f \). If \( f \) is one of the base functions, i.e., zero, addition of one or a projection, then the claim is obvious. (Note that here we use the fact that \( S \) can appear in the quantifier rules’ eigenterms.) Suppose \( f \) is defined by composition using the equation
\[
f(x) = h(g_1(x), \ldots, g_k(x))
\]
and suppose that the formulas \( \forall x \exists y, g_i(x) = y_i (i = 1, \ldots, k) \) and \( \forall y \exists u h(y) = u \) are \( \text{ff-provable} \) in \( \mathbf{A}[PR] \). For \( k = 1 \), a derivation for \( f \) is shown in Fig. 2(a).

Suppose \( f \) is defined by primitive recurrence using equations
\[
f(x, 0) = g(x) \\
f(x, S(y)) = h(x, y, f(x, y))
\]
and suppose the formulas \( \forall x \exists u g(x) = u \) and \( \forall x, y, p \exists v h(x, y, p) = v \) are \( \text{ff-provable} \) in \( \mathbf{A}[PR] \). The formula \( \forall y \exists z f(x, y) = z \) is proved using induction on \( y \). The base case is
\[
\begin{align*}
\forall x f(x, 0) &= g(x) \\
\forall x g(x) &= g(x) \\
\end{align*}
\]
and the induction step is shown in Fig. 2(b).

\[\square\]

Lemma 2. For every primitive recursive term \( t[x] \), \( \mathbf{A}[PR] \vdash \forall x \exists y t[x] = y \).

Proof. By induction on \( t \), using Lemma 1 in the induction step. \[\square\]

Lemma 3. For any formula \( A \), if \( \mathbf{A}[PR] \vdash A \), then \( \mathbf{A}[PR] \vdash A \).

Proof. By induction on the derivation. The only non-trivial cases are \((\forall E)\) and \((\exists I)\).

Suppose \( A[t[y]] \) is derived from \( \forall x A[x] \). Since \( t[y] \) is a primitive recursive term, \( \mathbf{A}[PR] \vdash \forall y \exists z t[y] = z \) by Lemma 2. Then the following is the derivation of \( A[t[y]] \).

\[
\begin{align*}
\forall y \exists z t[y] &= z \\
\exists z t[y] &= z \\
A[t[y]] &= A[z] \\
\end{align*}
\]

Suppose \( \exists x A[x] \) is derived from \( A[t[y]] \). As before, \( \mathbf{A}[PR] \vdash \forall y \exists z t[y] = z \). Then the following is the derivation of \( \exists x A[x] \).

\[
\begin{align*}
\forall y \exists z t[y] &= z \\
\exists z t[y] &= z \\
\exists x A[x] &= A[z] \\
\end{align*}
\]
∀x f(x) = h(g(x))  
\frac{f(x) = h(g(x))}{h(y) = y}  
\frac{g_1(x) = y}{f(x) = h(y)}  
\frac{h(y) = y}{\forall \exists y g_1(x) = y}  
\frac{f(x) = h(y)}{\exists f(x) = z}  
\frac{\exists f(x) = z}{\forall \exists f(x) = z}

\text{Fig. 2(a).}

\forall x, y f(x, Sy) = h(x, y, f(x, y))  
\frac{f(x, Sy) = h(x, y, f(x, y))}{h(x, y, p) = v}  
\frac{f(x, y, p) = v}{\exists z f(x, y) = z}  
\frac{\exists z f(x, y) = z}{\exists z f(x, Sy) = z}  
\frac{\exists z f(x, Sy) = z}{\forall y (\exists f(x, y) = z \rightarrow \exists z f(x, Sy) = z)}

\text{Fig. 2(b).}

\forall x, y f(x) = h(k(g(x, y), x, y))  
\frac{f(x) = h(k(g(x, y), x, y))}{g(x, y) = 0}  
\frac{g(x, y) = 0}{\forall x, y k(0, x, y) = y}  
\frac{\forall x, y k(0, x, y) = y}{h(y) = y}  
\frac{h(y) = y}{\forall x, \exists y g'(x, y) = 0}  
\frac{\exists y g'(x, y) = 0}{\exists z f(x) = z}  
\frac{\exists z f(x) = z}{\forall \exists z f(x) = z}

\text{Fig. 3.}
Theorem 1. All provably recursive functions of $A[PR]$ are ff-provable.

Proof. Suppose $f(x) = h(\mu y g(x, y) = 0)$ and $A[PR] \vdash \forall x \exists y g(x, y) = 0$. We would like to change $g$ so that for each $x$ it takes 0 for exactly one $y$, so we define

$$g'(x, y) = g(x, y) + \sum_{z < y} s(g(x, z)),$$

where $s(0) = 1$ and $s(x) = 0$ for $x \neq 0$. It is straightforward to see that $A[PR] \vdash \forall x \exists y g'(x, y) = 0$ and that $f(x) = h(\mu y g'(x, y) = 0)$.

By Lemma 3, $A[PR] \vdash \forall x \exists y g'(x, y) = 0$. Also, by Lemma 1, $A[PR] \not\vdash \forall y \exists y h(y) = u$. Let $P$ be the minimal full program containing equalities from PR for all primitive recursive functional symbols used in these derivations, plus the following equalities.

$$f(x) = h(k(g'(x, y), x, y))$$
$$k(0, x, y) = y$$

A derivation of $\forall x \exists z f(x) = z$ in $A[P]$ is shown in Fig. 3.

It is left to show that $P$ is strongly coherent and computes $f$. If $f$ is interpreted by $f$ and $k$ is interpreted by the total function

$$k(z, x, u) = \begin{cases} u & \text{if } z = 0, \\ \mu y g(x, y) = 0 & \text{otherwise} \end{cases}$$

then $P$ is true in the standard model of natural numbers; therefore, $A[P]$ is consistent. Further, for every $m, n$, if $f(m) = n$ then $P \not\vdash f(\overline{m}) = \overline{n}$. On the other hand, if $f(m) \neq n$, then $P \not\vdash f(\overline{m}) = \overline{n}$ because $f$ is total and $P$ is strongly coherent.

4 Functions that are ff-provable are provably recursive

To remind, under the assumption $A[P] \not\vdash \forall x \exists y f(x) = y$ we have to prove that $f$ is provably recursive according to the definition of Sect. 3, not that $A[P] \vdash \forall x \exists y f(x) = y$, which is trivial. We will prove this statement indirectly, using intrinsic theories introduced by Leivant [2]. An intrinsic theory is a framework for reasoning about inductively generated data.

The intrinsic theory of natural numbers, IT($\mathbb{N}$), is a first-order theory with equality whose vocabulary has functional symbols 0, $S$ and a unary predicate symbol $\mathbb{N}$. The additional inference rules are:

$$\frac{N(0) \quad N(t) \quad A[0]}{\forall x (A[x] \rightarrow A[Sx])}$$

Theorem 1. All provably recursive functions of $A[PR]$ are ff-provable.

Proof. Suppose $f(x) = h(\mu y g(x, y) = 0)$ and $A[PR] \vdash \forall x \exists y g(x, y) = 0$. We would like to change $g$ so that for each $x$ it takes 0 for exactly one $y$, so we define

$$g'(x, y) = g(x, y) + \sum_{z < y} s(g(x, z)),$$

where $s(0) = 1$ and $s(x) = 0$ for $x \neq 0$. It is straightforward to see that $A[PR] \vdash \forall x \exists y g'(x, y) = 0$ and that $f(x) = h(\mu y g'(x, y) = 0)$.

By Lemma 3, $A[PR] \vdash \forall x \exists y g'(x, y) = 0$. Also, by Lemma 1, $A[PR] \not\vdash \forall y \exists y h(y) = u$. Let $P$ be the minimal full program containing equalities from PR for all primitive recursive functional symbols used in these derivations, plus the following equalities.

$$f(x) = h(k(g'(x, y), x, y))$$
$$k(0, x, y) = y$$

A derivation of $\forall x \exists z f(x) = z$ in $A[P]$ is shown in Fig. 3.

It is left to show that $P$ is strongly coherent and computes $f$. If $f$ is interpreted by $f$ and $k$ is interpreted by the total function

$$k(z, x, u) = \begin{cases} u & \text{if } z = 0, \\ \mu y g(x, y) = 0 & \text{otherwise} \end{cases}$$

then $P$ is true in the standard model of natural numbers; therefore, $A[P]$ is consistent. Further, for every $m, n$, if $f(m) = n$ then $P \not\vdash f(\overline{m}) = \overline{n}$. On the other hand, if $f(m) \neq n$, then $P \not\vdash f(\overline{m}) = \overline{n}$ because $f$ is total and $P$ is strongly coherent.

4 Functions that are ff-provable are provably recursive

To remind, under the assumption $A[P] \not\vdash \forall x \exists y f(x) = y$ we have to prove that $f$ is provably recursive according to the definition of Sect. 3, not that $A[P] \vdash \forall x \exists y f(x) = y$, which is trivial. We will prove this statement indirectly, using intrinsic theories introduced by Leivant [2]. An intrinsic theory is a framework for reasoning about inductively generated data.

The intrinsic theory of natural numbers, IT($\mathbb{N}$), is a first-order theory with equality whose vocabulary has functional symbols 0, $S$ and a unary predicate symbol $\mathbb{N}$. The additional inference rules are:

$$\frac{N(0) \quad N(t) \quad A[0]}{\forall x (A[x] \rightarrow A[Sx])}$$
The variant of intrinsic theory that we are using, called discrete intrinsic theory and denoted by $\mathbf{IT}(\mathbb{N})$ in [2], also includes the separation axioms. Note that $\mathbf{IT}(\mathbb{N})$ uses regular first-order quantifier rules.

A function $f$ is called provable in $\mathbf{IT}(\mathbb{N})$ if it is computed by a strongly coherent program $(P, f)$ and $\mathbf{IT}(\mathbb{N}), \forall P \vdash \forall x (\mathbb{N}(x) \to \mathbb{N}(f(x)))$.

The following theorem is proved in [2].

**Theorem 2.** A function is provably recursive in $\mathbf{A}[\mathbf{PR}]$ iff it is provable in $\mathbf{IT}(\mathbb{N})$.

Thus, it is enough to show that ff-provable functions are provable in $\mathbf{IT}(\mathbb{N})$. However, we can show a stronger result, namely, that pr-provable functions are provable in $\mathbf{IT}(\mathbb{N})$.

Let us introduce some notation. If $A$ is a formula, then $A^N$ denotes $A$ with all quantifiers relativized to $\mathbb{N}$, i.e., having all subformulas of the form $\forall x B$ replaced by $\forall x (\mathbb{N}(x) \to B)$ and all subformulas of the form $\exists x B$ replaced by $\exists x (\mathbb{N}(x) \land B)$. If $\Gamma$ is a set of formulas, then $\Gamma^N = \{A^N \mid A \in \Gamma\}$. If $x = x_1, \ldots, x_n$, then $\mathbb{N}(x)$ denotes $\mathbb{N}(x_1) \land \ldots \land \mathbb{N}(x_n)$.

**Lemma 4.** Let $P$ be a full program and let $t[x]$ be a primitive recursive term in the language of $P$. Then $\mathbf{IT}(\mathbb{N}), \forall P \vdash \mathbb{N}(x) \Rightarrow \mathbb{N}(t[x])$.

**Proof.** The proof is similar to Lemma 2. For example, if $t[x]$ is $f(s[x])$ where $f$ is a symbol for a function $f(x, y)$ defined by primitive recurrence on $y$, then one needs to use induction for the formula $\mathbb{N}(y) \land \mathbb{N}(f(x, y))$. The fullness of $P$ is necessary to ensure that the induction hypothesis is true of all subterms of $t$. \hfill \square

**Lemma 5.** Let $P$ be a full program. Suppose that $\Gamma \cup \{A\}$ is a set of formulas in the language of $P$ whose free variables are among $x$. If $\mathbf{A}[P] \not\vdash \Gamma \Rightarrow A$ then $\mathbf{IT}(\mathbb{N}), \forall P \vdash \mathbb{N}(x), \Gamma^N \Rightarrow A^N$.

**Proof.** The proof is by induction on the derivation. If $A$ is an axiom of $\mathbf{A}[P]$ other than induction, then $\mathbf{IT}(\mathbb{N}), \forall P \vdash A$ and $A \vdash A^N$. The only other cases that need attention are those dealing with quantifiers and induction.

If $A[t]$ is derived from $\forall y A[y]$, then by induction hypothesis, $\forall y (\mathbb{N}(y) \to A^N[y])$ is derivable. Since $t$ is a primitive recursive term in the language of $P$, $\mathbb{N}(t)$ is derivable by Lemma 4, so $A^N[t]$ is derivable as well. The case of $(\exists I)$ is similar. The cases of $(\forall I)$ and $(\exists E)$ are also straightforward.

The relativized version of the induction axiom is

$$B^N[0] \to \forall y (\mathbb{N}(y) \to B^N[y] \to B^N[Sy]) \to \forall y (\mathbb{N}(y) \to B^N[y]).$$

It is proved by induction in $\mathbf{IT}(\mathbb{N})$ for the formula $\mathbb{N}(y) \land B^N[y]$. \hfill \square

**Theorem 3.** All pr-provable functions are provably recursive.

**Proof.** Let $f$ be computed by a strongly coherent full program $(P, f)$ and let $\mathbf{A}[P] \not\vdash \forall x \exists y f(x) = y$. Then by Lemma 5, $\mathbf{IT}(\mathbb{N}), \forall P \vdash \forall x (\mathbb{N}(x) \to \exists y \mathbb{N}(y) \land f(x) = y)$. This implies that $\mathbf{IT}(\mathbb{N}), \forall P \vdash \forall x (\mathbb{N}(x) \to \mathbb{N}(f(x)))$, so by Theorem 2, $f$ is provably recursive. \hfill \square
Acknowledgments

I am grateful to Daniel Leivant and Lev Beklemishev for constructive discussion.

References