Available online at www.sciencedirect.com
SCIENCE DIRECT®
Annals of Pure and Applied Logic 142 (2006) 1–18
ANNALS OF PURE AND APPLIED LOGIC
www.elsevier.com/locate/apal

Ackermann's substitution method (remixed)☆

Georg Moser

University of Innsbruck, Institute of Computer Science, Technikerstraße 21a, A-6020 Innsbruck, Austria

Received 5 November 2001; received in revised form 24 June 2004; accepted 19 September 2005
Available online 29 March 2006
Communicated by S.N. Artemov

Abstract

We aim at a conceptually clear and technically smooth investigation of Ackermann's substitution method [W. Ackermann, Zur Widerspruchsfreiheit der Zahlentheorie, Math. Ann. 117 (1940) 162–194]. Our analysis provides a direct classification of the provably recursive functions of $\mathsf{PA}(\varepsilon)$, i.e. Peano Arithmetic framed in the ε-calculus.

Keywords: Epsilon substitution method; Provably recursive functions

1. Introduction

A classification of the provably recursive functions of Peano Arithmetic (PA) in terms of Kreisel's class of ordinal recursive functions was suggested in [1]. This class can in turn be characterised by hierarchies of number-theoretic functions defined by transfinite recursion up to the ordinal ε_0, cf. [2]. Kreisel's solution of the classification problem for the provably recursive function of PA is based on *Ackermann's consistency proof* of arithmetic [3], framed in Hilbert's ε-calculus.

Hilbert's ε-calculus [4–6] is based on an extension of the language of predicate logic by a term-forming operator ε_x. This operator is governed by the *critical axiom*

$$A(t) \supset A(\varepsilon_x A(x)),$$

where t is an arbitrary term. Within the ε-calculus quantifiers become definable by $\exists x A(x) \Leftrightarrow A(\varepsilon_x A(x))$ and $\forall x\ A(x) \Leftrightarrow A(\varepsilon_x \neg A(x))$. The expression $\varepsilon_x A(x)$ is called *ε-term*.

When considering arithmetical systems the ε-substitution method [3,4] provides an analogue to Gentzen's famous extension [7,8] of his cut-elimination method. Tait [9] describes the substitution method as the general problem of associating with a formal system S, admitting quantifiers, a free-variable system $S^\star$ without quantifiers and to give an effective procedure of transforming statements A in (the language of) S into statements $A^\star$ in (the language of) $S^\star$. Assume S proves A, then the transform of A is to be an *ε-substitution instance* $A^\star$ of A. It is obtained by replacing ε-terms by terms in the language of $S^\star$. For Peano Arithmetic coached in the ε-calculus, this procedure of eliminating

☆ Work partially supported by Marie Curie grant no. HPMF-CT-2002-015777.
E-mail address: georg.moser@uibk.ac.at.

0168-0072/$ - see front matter
doi:10.1016/j.apal.2005.09.016

bounded variables from arbitrary proofs is sufficient to establish consistency (and even 1-consistency). The difficult part is to show that the substitution method terminates.

Let $\mathsf{PA}(\varepsilon)$ denote Peano arithmetic framed in the ε-calculus. Based on Gentzen's work, revealing the role played by transfinite induction up to ε_0, Ackermann [3] presented a constructive termination proof of the substitution method for $\mathsf{PA}(\varepsilon)$. As an important achievement he defined functions, ordinal recursive in ε_0, that bound the *complexity* of the transformation procedure.[1] It is a direct consequence of Ackermann's proof, firstly observed by Kreisel [1], that the provably recursive functions of PA are primitive recursive in some $\prec \varepsilon_0$-recursive functions. Thus [3] renders a Π^0_2-analysis of PA and establishes 1-consistency of PA; see also [10].

We analyse Ackermann's solution and in particular the given complexity analysis of the substitution method. In our presentation we follow the original treatment closely. The novelty being that we are able to measure the complexity of the substitution method directly in terms of the fast-growing *Hardy hierarchy* (see [11]), i.e., functions from the Hardy hierarchy replace the specific ordinal recursive functions – seemingly ad-hoc defined – employed in [3]. Thus we show that any provably recursive function of $\mathsf{PA}(\varepsilon)$ can be elementarily defined in some H_α, $\alpha \prec \varepsilon_0$ and therefore the class of provably recursive functions of $\mathsf{PA}(\varepsilon)$ equals the Hardy class $\mathcal{H}$. The same machinery is applied to characterise the provably recursive functions of a weak arithmetic theory without induction axiom (or rule); here the Hardy hierarchy can be replaced by the *slow-growing hierarchy*. We have replaced the set-theoretical ordinals employed in Ackermann's proof by (structured) tree-ordinals.

The reader may wonder why we have based our investigation on the original – quite old – treatment of the substitution method; the work by Arai [12,13], Avigad [10], Buchholz, Mints, and Tupailo [14], Mints [15,16], and Tait [17,9] spring to mind as more adequate starting points. However, to our surprise, it turned out that once we understood how to replace Ackermann's original representation and codings of (set-theoretical) ordinals by structured tree-ordinals, the desired results followed quite easily. Thus by changing the employed ordinal notation we can establish the direct characterisation result, but still follow the original presentation closely enough to render a modern presentation of Ackermann's ideas.

In contrast to Gentzen-style proof theory by cut-elimination the substitution method is less dependent on the structure of a given derivation in S. We employ this fact to separate the actual substitution method and the ε-calculus. This allows us to make an abstract assessment of the transformation procedure incorporated in the substitution method apart from the ε-calculus trade. In the next section we define a class of tautologies $\mathcal{S}$ and we re-formulate the problem of the substitution method accordingly. Only after we have studied the behaviour of the transformation procedure with respect to the class $\mathcal{S}$ in some detail, we relate our findings to a suitable axiomatisation of Peano arithmetic in the ε-calculus and thus obtain the main result of this work.

2. The formal system $\mathcal{S}$

We assume an arbitrary but fixed *language* $\mathcal{L}$ of arithmetic, such that $\mathcal{L}$ does not contain quantifiers. Instead of including $\neg$ as a logical connective, negation is defined by asserting that atomic formulas $R(t_1, \ldots, t_n)$ occur in complementary pairs $\overline{R}(t_1, \ldots, t_n)$. Note that $\overline{\overline{R}}(\ldots) := R(\ldots)$. In this sense the classical double negation law becomes a syntactic equality. Using de Morgan's laws this definition is lifted to the general level.

It is notationally convenient to distinguish between *bound* $(x, y, z, \ldots)$ and *free* variables $(a, b, c, \ldots)$, respectively. Bound variables are collected in the set BV, while free variables are collected in the set FV; we set $\mathcal{V} := \mathsf{FV} \cup \mathsf{BV}$. *Terms* in $\mathcal{L}$ are constructed from *constants*, *free variables*, and *function symbols* as usual. *Semi-terms* are like terms but may also contain *bound* variables. *Formulas* are defined with the proviso that only bound variables are allowed to be quantified and only free variables may occur free. *Semi-formulas* are similar to formulas with the exception that both free and bound variables may occur free in a semi-formula. An *expression* is either a (semi-)term or a (semi-)formula.

We use the metasymbols $f, g, h, \ldots$ to denote function symbols, while the metasymbols $P, Q, R, \ldots$ vary through predicate symbols. We write $\mathrm{ar}(f)$ $(\mathrm{ar}(P))$ to denote the arity of a function (predicate) symbol f (P). Within this text we are eager to use only the symbols k, l, m, n, p, q as denotations of natural numbers. Deviations from this convention will be clearly marked. We write $[1, n]$ to denote the interval of natural numbers from 1 to n. Occasionally we abbreviate tuples of terms $(t_1, \ldots, t_n)$ as $\overline{t}$. The length of the tuple will always be clear from the context.

[1] By *complexity* of the substitution method we understand the maximal number of approximation steps necessary till the final substitution is rendered.