Lambda Representation of Operations Between Different Term Algebras

Marek Zainc
zainc@cs.buffalo.edu

Abstract. There is a natural isomorphism identifying second order types of the simple typed λ calculus with free homogeneous term algebras. Let τ^1 and τ^B be types representing algebras A and B respectively. Any closed term of the type τ^1 → τ^B represents a computable function between algebras A and B. The problem investigated in the paper is to find and characterize the set of all λ definable functions between structures A and B. The problem is presented in a more general setting. If algebras A_1, ..., A_n, B are represented respectively by second order types τ^A_1, ..., τ^A_n, τ^B then τ^A_1 → (τ^A_n → τ^B) ... is a type of functions from the product A_1 × ... × A_n into algebra B. Any closed term of this type is a representation of algorithm which transforms the tuple of terms of types τ^A_1, ..., τ^A_n respectively into a term of type τ^B, which represents an object in algebra B (see [BoB85]). The problem investigated in the paper is to find an effective computational characteristic of the λ definable functions between arbitrary free algebras and the expressiveness of such transformations. As an example we will consider λ definability between well known free structures such as: numbers, words and trees. The result obtained in the paper is an extension of the results concerning λ definability in various free structures described in [Sch73] [Sta79] [Lei89] [Zai87] [Zai90] and [Zai91]

Introduction

As a contribution to the ongoing research on computing over general algebraic structures, we consider recurrence over free algebras (compare [BoB85], [Lei89], [Lei90], [Zai89]). As a model for computing a simple typed lambda calculus is employed. The lambda calculus introduced by Church [Chu40] is a calculus of expressions, which naturally describes the notion of computable function. Functionals are considered dynamically as rules rather than set theoretic graphs. The lambda calculus mimics the procedure of computation of the program by the process called beta reduction. There is a natural way of expressing objects such as numbers, words, trees and other syntactic entities in the lambda calculus. All those objects are of a considerable value for computer scientists. Dynamic operations on objects of this kind can be described by terms of lambda calculus. Therefore lambda terms may be considered as algorithms or programs working.
on those syntactic objects and producing as a result a new object not necessarily of the same type. It is well known result by Church and Kleene, relating all partial computable numerical functions with lambda terms. Of course, the notion of partial recursive function can be naturally extended to other structures such as words, trees etc. It is natural that the Church-Kleene theorem might be extended and holds for these structures.

The typed version of lambda calculus is obtained by imposing simple types on the terms of the lambda calculus. The problem of representing structures is basically the same in the typed lambda calculus, however, the rigid type structure imposed on the syntax of lambda calculus dramatically reduces expressiveness of functions on these structures. Interestingly enough, the solution for representability problems varies for different structures.

The first result concerning representability in the typed \( \lambda \) calculus have been proved by Schwichtenberg in 1975 and independently by Statman (see [Sch75], [Stat79]). Schwichtenberg studied numerical functions represented in the typed lambda calculus and the following characteristic was proved: lambda definable functions are exactly those generated by composition from the constants 0 and 1 and operations of addition, multiplication and conditional (extended polynomials). The similar result for word operations was obtained by Zaionc in [Zai87]. The word functions represented in typed lambda calculus are exactly those generated by composition from constant \( \lambda \) (empty word) and operations \textit{append}, \textit{substitution} and \textit{cut}. The results of Schwichtenberg and Zaionc were extended to the structure of binary trees [Zai90]. It was shown that \( \lambda \) definable tree operations are those obtained from initial functions by composition and a limited version of primitive recursion. Leivant [Lei89] showed that recursion is essential and can not be removed from this characteristic. A similar result was obtained for \( \lambda \) definable operations on arbitrary homogeneous free algebra.

In this paper we examine the situation when the input and output algebras are generally different. The proof of the main result is obtained by inductive decomposition of closed term which represents a function between two different algebras. While the decomposed terms are generally simpler according to some measure of complexity, they represent operations between definitely different algebras from algebras we started with. Therefore, the problem must be presented in a more general setting in which we consider functions from product of several not necessarily same algebras.

1. Free Algebras

Algebra \( A \) given by a signature \( S_A = [\alpha_1, ..., \alpha_n] \) has \( n \) constructors \( a_1, ..., a_n \) of arities \( \alpha_1, ..., \alpha_n \) respectively. Expressions of the algebra \( A \) are defined by induction as the minimal set such that if \( \alpha_i = 0 \) then \( a_i \) is an expression and if \( \alpha_i > 0 \) and \( t_1, ..., t_{\alpha_i} \) are expressions then \( a_i(t_1, ..., t_{\alpha_i}) \) is an expression. We may assume that at least one \( \alpha_i \) is equal 0, otherwise the set of expressions is empty. By \( A \) we mean the set of all expressions in algebra given by signature \( S_A = [\alpha_1, ..., \alpha_n] \). For simplicity we are going to write \( A = [\alpha_1, ..., \alpha_n] \) to say that \( A \) is an algebra given by the signature \([\alpha_1, ..., \alpha_n]\). If \( A_1, ..., A_n \) are algebras
then by \( A_1 \times \ldots \times A_n \) we mean the product of sets of expressions. A bar over the name (for example \( \overline{A} \)) indicates that \( \overline{A} \) is the product of sets of expressions of some algebras. \( A^n \) is the product \( A \times \ldots \times A \).

**Definition 1.1** If \( A \) is an algebra given by signature \( S_A = [\alpha_1, \ldots, \alpha_n] \) and \( n \) is a nonnegative integer then \( A^{+n} \) is an algebra given by signature \( S_{A^{+n}} = [\alpha_1, \ldots, \alpha_n, 0, \ldots, 0] \) with exactly \( n \) 0’s added.

We are going to investigate the set of functions between arbitrary term algebras.

**Definitions 1.2** Function \( f \) in \( A^p \rightarrow A \) given by \( f(x_1, \ldots, x_p) = x_i \) is called projection. Any constructor \( a_i \) in algebra \( A \) can be seen as a function \( a_i : A^{\alpha_i} \rightarrow A \) including \( 0 \)-ary function \( (a_i = 0) \) considered as element of \( A \).

**Definition 1.3** By the set of nonterminal trees in \( A \) we mean the minimal set of mappings \( A^p \rightarrow A \) for \( p > 0 \) closed for composition and containing all constructors in \( A \) and all projections.

**Definition 1.4** Let \( f : \overline{A} \times \overline{B} \rightarrow C \) and \( \overline{b} \in \overline{B} \). By \( f / \overline{b} \) we mean the function \( (f/\overline{b}) : A \rightarrow C \) defined by \( (f/\overline{b})(x) = f(x, \overline{b}) \).

**Definition 1.5** Let \( A = [\alpha_1, \ldots, \alpha_n] \). Let \( a_1, \ldots, a_n \) are all constructors in algebra \( A \) such that arity of \( a_i \) is \( \alpha_i \). Function \( h : A \times \overline{B} \rightarrow C \) is defined by recursion from functions \( f_1 : C^{\alpha_1} \times B \rightarrow C, \ldots, f_n : C^{\alpha_n} \times B \rightarrow C \) if for all \( i \leq n \) the following equations hold:

\[
h(a_{i}(t_1, \ldots, t_{a_i}), \overline{b}) = f_i(h(t_1, \overline{b}), \ldots, h(t_{a_i}, \overline{b}, \overline{b})
\]

for all expressions \( t_1, \ldots, t_{a_i} \) of the algebra \( A \).

**Definition 1.6** The set of \( \lambda \) functions is defined inductively by:

1. for all algebras \( \overline{A}, \overline{B}, \overline{C} \) any projection in \( \overline{A} \times \overline{B} \times \overline{C} \rightarrow \overline{B} \) given by \( f(x, y, z) = y \) is a \( \lambda \) function.
2. any constant function from \( \overline{A} \) to \( \overline{B} \) given by \( f(\overline{x}) = t \) for some expression \( t \in \overline{B} \) is a \( \lambda \) function.
3. for any algebra \( A \) any constructor \( a_i : A^{\alpha_i} \rightarrow A \) is a \( \lambda \) function.
4. any composition of \( \lambda \) functions is a \( \lambda \) function. i.e. if \( f_1 : A_1 \times \ldots \times A_n \rightarrow B_1, \ldots, f_k : A_1 \times \ldots \times A_n \rightarrow B_k \) are \( \lambda \) functions and \( g : B_1 \times \ldots \times B_k \rightarrow C \) is a \( \lambda \) function then \( h : A_1 \times \ldots \times A_n \rightarrow C \) given by \( h(\overline{x}) = g(f_1(\overline{x}), \ldots, f_k(\overline{x})) \) is a \( \lambda \) function.
5. let algebra \( A \) be given by a signature \([\alpha_1, \ldots, \alpha_n]\) and \( \overline{B} \) be a product of algebras. If \( f_1 : C^{\alpha_1} \times \overline{B} \rightarrow C, \ldots, f_n : C^{\alpha_n} \times \overline{B} \rightarrow C \) are \( \lambda \) functions such that for every expression \( \overline{b} \in \overline{B} \) functions \( f_1 / \overline{b} : C^{\alpha_1} \rightarrow C, \ldots, f_n / \overline{b} : C^{\alpha_n} \rightarrow C \) are nonterminal trees in \( C \) then the function \( h : A \times \overline{B} \rightarrow C \) defined by recursion (see definition 1.5) from \( f_1, \ldots, f_n \) is a \( \lambda \) function.
2. Extended Typed λ Calculus.

Our language is derived from Church’s [Chu40] simple theory of types. Every term possesses a unique type which indicates its position in a functional hierarchy. Let TYPE be a set of types which are defined as follows: O is a type and if τ and μ are types then τ → μ is a type. For any type τ we define numbers rank(τ) and arg(τ) as follows: arg(O)=rank(O)=0 and arg(τ → μ) = 1 + arg(μ) and rank(τ → μ) = max(rank(τ) + 1, rank(μ)). Associated with each type τ is a denumerable set of variables V(τ). Any variable of type τ is a term of type τ. If T is a term of type τ → μ and S is a term of type τ then TS is a term of type μ. If T is a term of type μ and x is a variable of type τ then λx.T is a term of type τ → μ. If T is a term of type τ we write T ∈ τ. We shall use the notation λx1...xn.T for term λx1(...(λxn,T)...) and TS1...Sn for (...(TS1)...Sn). If T is a term and x is a variable of the same type as a term S, then T[x/S] denotes the substitution of the term S for each free occurrence of x in T. The axioms of equality between terms have the form of αβη conversions (see [Bar81] or [Fri73]) and the convertible terms are written as T ≡βη S. By Cl(τ) we mean the set of all closed terms (without free variables) of type τ. Term T is in the long normal form if T = λx1...xn.yT1...Tk where y is an xi for some i ≤ n or y is a free variable, Tj for j ≤ k are in the long normal form and yT1...Tk is a term of type O. Long normal forms exist and are unique for βη conversions.

In order to easily represent and manipulate on finite strings of terms of potentially different types we extend our language by adding new constructors to form a Cartesian product of types and terms. By TYPE* we mean the minimal set containing TYPE and closed for the Cartesian product formation: if τ1, ..., τn ∈ TYPE* then (τ1, ..., τn) ∈ TYPE*. The empty tuple is denoted by ω ∈ TYPE*. We use the abbreviation x^n_{i=1}τi for string (τ1, ..., τn). We assume that string of empty types is empty, so x^n_{i=1}ω = ω. In the case when all τi are the same we use notation τ^n instead of x^n_{i=1}τi. We also extend the type formation → for string types as follows: x^n_{i=1}τi → μ means τi → (τ2 → ...τn → μ)|...| including ω → μ = μ and τ → x^n_{i=1}μj means x^n_{i=1}(τ → μj) including τ → ω = ω. Particularly we have (x^n_{i=1}τi) → (x^n_{i=1}μj) = x^n_{i=1}(x^n_{i=1}τi → μj) and τ^n → μ = μ. Since types are usually employed for describing construction of function spaces the above definitions are in fact typical identifications. The first equation identifies A^{B×C} with A^B C which usually is called ”Currying”. The second equates (A×B)^C with AC×BC. We will call elements of TYPE* types and elements of TYPE simple types. It is easy to observe that every simple type τ admits the unique form x^n_{i=1}τi → O where τi are again simple types called components of τ. We can prove several equations between types, for example:

\[(τ → μ)^n = τ → μ^n\] 2.1
\[(τ^k → μ)^n = τ^k → μ^n\] the special case of 2.1 2.2
for simple type τ of arity k,
\[τ^n = (x^k_{i=1}τi → O)^n = x^k_{i=1}τi → O^n\] 2.3
for simple type $\tau$ of arity $k$, where $\tau_i$ are components of $\tau$

$$\chi^k_{\sum_{i=1}^n \tau_i} = \chi^k_{\sum_{i=1}^n \tau_i} \rightarrow O)^{\alpha_j} = \chi^k_{\sum_{i=1}^n \tau_i} \rightarrow O^{\alpha_j} =$$

$$= \chi^k_{\sum_{i=1}^n \tau_i} \rightarrow \chi^k_{\sum_{i=1}^n \tau_i} O^{\alpha_j} =$$

2.4

Language over tuple types is build in the similar way. If $M$ is a finite string of terms $(M_1, ..., M_n)$ of types $\tau_1, ..., \tau_n$ respectively then $M$ is denoted by $\chi^k_{\sum_{i=1}^n \tau_i}$ of the type $\chi^k_{\sum_{i=1}^n \tau_i}$. We use the same notation $M \in \tau$ to say that $M_i$ is a term of type $\tau_i$ for $i \leq n$. This definition may be iterated so we may consider strings of strings of terms and so on. The empty tuple of terms is denoted by $\emptyset$ of type $\omega$.

In the case when all $M_i$ are the same we use $M^n$ instead of $\chi^k_{\sum_{i=1}^n \tau_i}$ with $\Omega^n = \Omega$. Term formation is following:

If $M \in \chi^m_{\sum_{i=1}^n \tau_i} \rightarrow \mu$ and $N \in \chi^m_{\sum_{i=1}^n \tau_i}$ then by $MN$ we mean $MN_1 ... N_n$ of type $\mu$ with $M \in \Omega^m = \Omega$ when $m = 0$.

If $M \in \mu$ and $x$ is a variable of type $\chi^m_{\sum_{i=1}^n \tau_i}$ then by $\lambda x \mu$ we mean $\lambda x \mu$. If $M \in \chi^m_{\sum_{i=1}^n \tau_i}$ then by parallel application $M \circ N$ we mean the term $\chi^m_{\sum_{i=1}^n \tau_i}$. If $x = (x_1 ... x_n)$ is a variable of type $\chi^m_{\sum_{i=1}^n \tau_i}$ and $M \in \chi^m_{\sum_{i=1}^n \tau_i}$ then by parallel substitution $\lambda x \mu$ we mean the term $(\lambda x_1 M_1, ..., \lambda x_n M_n)$ of type $\chi^m_{\sum_{i=1}^n \tau_i}$.

We may summarize all those definitions by:

$$M(\chi^m_{\sum_{i=1}^n \tau_i}N_i) = MN_1 ... N_n \text{ with } M \in \Omega = \Omega = 2.5$$

$$(\chi^m_{\sum_{i=1}^n \tau_i}M_i)N = \chi^m_{\sum_{i=1}^n (M_iN)} \text{ with } \Omega^m = \Omega = 2.6$$

$$\lambda x \chi^m_{\sum_{i=1}^n \tau_i} = \chi^m_{\sum_{i=1}^n (\lambda x \mu)} \text{ with } \lambda x \mu = \Omega = 2.7$$

$$\chi^m_{\sum_{i=1}^n \tau_i} \circ (\chi^m_{\sum_{i=1}^n \tau_i}M_i) = \chi^m_{\sum_{i=1}^n (\lambda x \mu_i)} \text{ with } \lambda \mu = \Omega = 2.8$$

$$\lambda x_1 \chi^m_{\sum_{i=1}^n \tau_i} \circ (\chi^m_{\sum_{i=1}^n \tau_i}M_i) = \chi^m_{\sum_{i=1}^n (\lambda x_1 \mu_i)} \text{ with } \lambda \mu = \Omega = 2.9$$

$$\lambda x_1 \chi^m_{\sum_{i=1}^n \tau_i} \circ (\chi^m_{\sum_{i=1}^n \tau_i}M_i) = \chi^m_{\sum_{i=1}^n (\lambda x_1 \mu_i)} \text{ with } \lambda \mu = \Omega = 2.10$$

By simple term we mean a term of a simple type. If $M$ is a simple term and $x$ is a variable of type $\chi^m_{\sum_{i=1}^n \tau_i}$ then $M(x/\mu)$ denotes the term obtained by simultaneous substitution of the terms $N_i$ for each free occurrences of $x_i$ respectively in $M$. If $M \in \chi^m_{\sum_{i=1}^n \tau_i}$ then $M(x/\mu)$ means $(M_1(x/\mu_1), ..., M_n(x/\mu_n))$ and if $n = k$ then $M(x/\mu)$ means $(M_1(x_1/\mu_1), ..., M_k(x_n/\mu_n))$. We define $\beta \eta$ conversions and the notion of long normal form by recursion with respect to the complexity of tuple types. Two terms $M$ and $N$ of the same type $\chi^m_{\sum_{i=1}^n \tau_i}$ are equal modulo conversions $M = \beta \eta N$ if and only if $M_i = \beta \eta N_i$ for all $i \leq n$. We say that term $M$ of type $\chi^m_{\sum_{i=1}^n \tau_i}$ is in long-normal form if every $M_i$ is in long-normal form and $M$ is called closed if every $M_i$ is closed. We can prove the following extensions of conversions

5
\[ \lambda x. M \equiv_{\beta \eta} \lambda y. M(x/y) \] 2.11 sequential \( \alpha \) conversion
\[ \lambda x \cdot M \equiv_{\beta \eta} \lambda y \cdot M(x/y) \] 2.12 parallel \( \alpha \) conversion
\[ (\lambda x. M) N \equiv_{\beta \eta} M(x/N) \] 2.13 (sequential \( \beta \) conversion)
\[ (\lambda x \cdot M) \circ N \equiv_{\beta \eta} M(x/N) \] 2.14 parallel \( \beta \) conversion
\[ \lambda x. (M x) \equiv_{\beta \eta} M \] 2.15 sequential \( \eta \) conversion
\[ \lambda x \cdot (M \circ x) \equiv_{\beta \eta} M \] 2.16 parallel \( \eta \) conversion

Having proved the existence of the long-normal form in the ordinary typed \( \lambda \) calculus we can show the same for tuple terms by induction. If \( M \) is a term of type \( \times^n_{i=1} \mu_i \) then by \( M_i \) we mean \( i \)-th coordinate of \( M \), therefore we have

\[ (MN)_i = M_i N \] 2.17
\[ (\lambda x. M)_i = \lambda x_i. M_i \] 2.18
\[ (M \circ N)_i = M_i N_i \] 2.19
\[ (\lambda x \cdot M)_i = \lambda x_i. M_i \] 2.20
\[ (\times^n_{i=1} M)_i = M_i \] 2.21
\[ \Omega_i = \Omega \] 2.22

3. Representability

If \( A \) is an algebra given by a signature \( S_A = [\alpha_1, ..., \alpha_a] \) then by \( \tau^A \) we mean a type \( (O^{\alpha_1} \to O) \to ... \to (O^{\alpha_a} \to O) \to O \). By \( \tau^A_i \) for \( i \leq a \) we mean \( i \)-th component of type \( \tau^A \) i.e. \( \tau^A_i = O^{\alpha_i} \to O \). We will see that closed terms of this type reflect constructions in algebra \( A \). Assuming that at least one \( \alpha_i = 0 \) we have that \( \tau^A \) is not empty. \( \tau^A \) is the simple type for any algebra \( A \). There is a natural 1-1 isomorphism between expressions of algebra \( A \) and closed terms of type \( \tau^A \). Let \( c_1, ..., c_n \) are all constructors in term algebra, of arity \( \alpha_1, ..., \alpha_a \) respectively. If \( a_0 \) is an 0-ary constructor in \( A \) then the closed term \( \lambda x_1 ... x_n. x_i \) represents \( a_i \). If \( \alpha_i > 0 \) and \( t_1, ..., t_{\alpha_i} \) are expressions in \( A \) represented by closed terms \( T_1, ..., T_{\alpha_i} \), then an expression \( a_i(t_1, ..., t_{\alpha_i}) \) is represented by the term \( \lambda x_1 ... x_n. x_i(T_1 x_1 ... x_n) ... (T_{\alpha_i} x_1 ... x_n) \). Thus, we have a 1-1 correspondence between closed terms of type \( \tau^A \) and expressions of algebra \( A \). The unique (up to \( \beta \eta \) conversion) term of type \( \tau^A \) which represents an expression \( h \) in algebra \( A \) is denoted by \( \underline{h} \). Let \( A_1, ..., A_n \) and \( B \) be algebras. A function \( h : A_1 \times ... \times A_n \to B \) is represented by a closed term \( H \) of type \( \tau^A : \tau^{A_1} \to ... \to \tau^{A_n} \to \tau^B \) if for all expressions \( t_1 \in A_1, ..., t_n \in A_n \), the following terms are \( \beta \eta \) convertible

\[ H(t_1, ..., t_n) = \beta \eta h(t_1, ..., t_n). \]

Let \( \overline{B} \) be a product of algebras \( B_1 \times ... \times B_k \). We define a \( \overline{\tau^B} \) to be the type \( \times^k_{\tau=1} \tau^{B_k} \). By analogy, there is a natural isomorphism between terms of type \( \overline{\tau^B} \) and the product of expressions \( B_1 \times ... \times B_k \).

**Example 3.1** The algebra \( N \) of positive integers based on the signature \( S_N = [1, 0] \) is represented by the type \( \tau^N = (0 \to 0) \to (0 \to 0) \). Every number
n is represented by a term (Church's numerals) of the form $\lambda x.x(\ldots x)$.$\ldots$ The algebra $\Sigma$ of binary words based on the signature $S_\Sigma = \{1, 0\}$ is represented by the type $\tau^\Sigma = (0 \to 0) \to ((0 \to 0) \to (0 \to 0))$. For example, the word $aba$ over the alphabet $\Sigma = \{a, b\}$ is represented by the term $\lambda u x. u(v(u)x))$. The term $\lambda w x. z w(y,y)x$ of type $\tau^\Sigma \to \tau^N$ represents the function counting the number of letters $a$ in the given word.

**Example 3.2** $U = [2, 0]$ is the algebra of binary trees and $N = [1, 0]$ is the algebra of Church's numerals. Let $e$ be a 0-ary constructor (empty tree) and $\wedge$ be a binary tree constructor in the algebra $U$. In this example infix notation is used for the binary constructor $\wedge$. By $t_1 \wedge t_2$ we denote the tree such that $t_1$ and $t_2$ are, respectively, left and right subtrees. In this example $U$ is the set of all binary trees and $N$ is the set of all Church's numerals. Type $\tau^U$ is $(O \to (O \to O)) \to (O \to O) \to (O \to O)$ and $\tau^N = (O \to O) \to (O \to O)$. Let $H$ be a closed term $\lambda T u x. T(\lambda y z. uy)x$ of type $\tau^U \to \tau^N$. It is easy to see that $H$ represents the function $\text{leftmost} : U \to N$ which computes the length of the leftmost path of a tree.

The function $\text{leftmost}$ is obtained by the recursion schema from the functions $f_i(y, z) = y + 1$ and $f_0 = 0$. Since $f_1$ and $f_2$ are $\lambda$ functions which are nonterminal in $N$, the function $\text{leftmost}$ is also a $\lambda$ function (see definition 1.6).

**Definition 3.3** Let $A$ be an algebra based on signature $[a_1, \ldots, a_n]$. Let $Z_i$ be a variable of type $(\tau^A)^{a_i}$ for $i \leq a$ and $x$ be a variable of type $\times_{i=1}^a (O^{a_i} \to O)$. Let $\text{Cons}_i$ be a closed term of type $\times_{i=1}^a ((\tau^A)^{a_i} \to \tau^A)$, defined by $\text{Cons}_i = (\text{Cons}_1, \ldots, \text{Cons}_a)$, where $\text{Cons}_i$, of the type $(\tau^A)^{a_i} \to \tau^A$, is defined by $\text{Cons}_i = \lambda Z x. x_i(Zx)$. Note that $\text{Cons}_i$ represents constructor $a_i$, which can be seen as a function $a_i : A^{a_i} \to A$. Note also that when $a_i = 0$, $\text{Cons}_i$ is a projection $\lambda x. x_i$ which represents constant $a_i$.

**Definition 3.4** Let $A = [a_1, \ldots, a_n]$ be an algebra. Let $\kappa^A$ be a collection of closed terms of types $(\tau^A)^{a_i} \to \tau^A$ for all $p \geq 0$ defined by recursion as the minimal set containing projections $\lambda s. s_i$ and constant functions $\lambda s. \text{Cons}_i$ if $a_i = 0$, and satisfying following property: If $D$ is a closed term of type $(\tau^A)^{p} \to (\tau^A)^{a_i}$ for $a_i > 0$, such that $D_j \in \kappa^A$ for all $j \leq a_i$, then a closed term $\lambda s. \text{Cons}_i(Ds)$ of type $(\tau^A)^{p} \to \tau^A$ belongs to $\kappa^A$.

Next four lemmas 3.5, 3.6, 3.7, 3.8 are concern with type checking of particular terms and will be used afterward in lemmas 3.10, 3.13 and 3.14 as well as in theorem 4.6.

**Lemma 3.5** Let $A = [a_1, \ldots, a_n]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, \ldots, \gamma_c]$ be an algebra. For every closed term
\( T \) of the type \( \tau_{\mathbf{T}} \text{·} (x_{j=1}^{\alpha - 1} t_{j}^{C} \rightarrow x_{j=1}^{\alpha - 1} \tau_{j}^{C}) \) a term \( \lambda Y \cdot (\lambda T.x.((\mathbf{T}T.x) \circ (Y.x))) \) is well-formed in type \( x_{j=1}^{\alpha - 1} ((\tau_{C})^{\alpha} \rightarrow \tau_{C}) \).

**Lemma 3.6** Let \( B \) be a product of algebras represented by type \( \tau_{\mathbf{B}} \). Let \( C = [\gamma_{1}, \ldots, \gamma_{c}] \) be an algebra. For every closed term \( \mathbf{T} \) of the type \( \tau_{\mathbf{B}} \rightarrow (\tau_{C})^{\gamma_{i}} \) for \( i \leq c \) the closed term \( \lambda T.x.((\mathbf{T}T.x) \circ (Y.x))) \) is well-formed in type \( \tau_{\mathbf{B}} \rightarrow \tau_{C} \).

**Lemma 3.7** Let \( A = [\alpha_{1}, \ldots, \alpha_{a}] \) be an algebra. Let \( B \) be a product of algebras represented by type \( \tau_{\mathbf{B}} \). Let \( C = [\gamma_{1}, \ldots, \gamma_{c}] \) be an algebra. For every closed term \( \mathbf{T} \) of type \( \tau_{\mathbf{B}} \), for every closed \( \mathbf{Y} \) of type \( x_{j=1}^{\alpha - 1} ((\tau_{C})^{\alpha} \rightarrow \tau_{C}) \) and every closed \( \mathbf{H} \) of type \( \tau_{A} \rightarrow \tau_{\mathbf{B}} \rightarrow \tau_{C} \) the closed term \( \mathbf{T} \circ (\mathbf{H} \circ (\mathbf{F} \circ (\mathbf{S} \circ (\mathbf{T} \circ (\mathbf{Y} \circ (\mathbf{B} \circ (\mathbf{A} \circ (\mathbf{C} \circ (\mathbf{D} \circ (\mathbf{E} \circ (\mathbf{F} \circ (\mathbf{G} \circ (\mathbf{H} \circ (\mathbf{I} \circ (\mathbf{J} \circ (\mathbf{K} \circ (\mathbf{L} \circ (\mathbf{M} \circ (\mathbf{N} \circ (\mathbf{O} \circ (\mathbf{P} \circ (\mathbf{Q} \circ (\mathbf{R} \circ (\mathbf{S} \circ (\mathbf{T} \circ (\mathbf{U} \circ (\mathbf{V} \circ (\mathbf{W} \circ (\mathbf{X} \circ (\mathbf{Y} \circ (\mathbf{Z} \circ (\mathbf{\alpha} \circ (\mathbf{\beta} \circ (\mathbf{\gamma} \circ (\mathbf{\delta} \circ (\mathbf{\epsilon} \circ (\mathbf{\zeta} \circ (\mathbf{\eta} \circ (\mathbf{\theta} \circ (\mathbf{\iota} \circ (\mathbf{\kappa} \circ (\mathbf{\lambda} \circ (\mathbf{\mu} \circ (\mathbf{\nu} \circ (\mathbf{\xi} \circ (\mathbf{\omicron} \circ (\mathbf{\pi} \circ (\mathbf{\rho} \circ (\mathbf{\sigma} \circ (\mathbf{\tau} \circ (\mathbf{\upsilon} \circ (\mathbf{\phi} \circ (\mathbf{\chi} \circ (\mathbf{\psi} \circ (\mathbf{\omega} \circ (\mathbf{\alpha} \circ (\mathbf{\beta} \circ (\mathbf{\gamma} \circ (\mathbf{\delta} \circ (\mathbf{\epsilon} \circ (\mathbf{\zeta} \circ (\mathbf{\eta} \circ (\mathbf{\theta} \circ (\mathbf{\iota} \circ (\mathbf{\kappa} \circ (\mathbf{\lambda} \circ (\mathbf{\mu} \circ (\mathbf{\nu} \circ (\mathbf{\xi} \circ (\mathbf{\omicron} \circ (\mathbf{\pi} \circ (\mathbf{\rho} \circ (\mathbf{\sigma} \circ (\mathbf{\tau} \circ (\mathbf{\upsilon} \circ (\mathbf{\phi} \circ (\mathbf{\chi} \circ (\mathbf{\psi} \circ (\mathbf{\omega} \circ (\mathbf{\alpha} \circ (\mathbf{\beta} \circ (\mathbf{\gamma} \circ (\mathbf{\delta} \circ (\mathbf{\epsilon} \circ (\mathbf{\zeta} \circ (\mathbf{\eta} \circ (\mathbf{\theta} \circ (\mathbf{\iota} \circ (\mathbf{\kappa} \circ (\mathbf{\lambda} \circ (\mathbf{\mu} \circ (\mathbf{\nu} \circ (\mathbf{\xi} \circ (\mathbf{\omicron} \circ (\mathbf{\pi} \circ (\mathbf{\rho} \circ (\mathbf{\sigma} \circ (\mathbf{\tau} \circ (\mathbf{\upsilon} \circ (\mathbf{\phi} \circ (\mathbf{\chi} \circ (\mathbf{\psi} \circ (\mathbf{\omega} \circ (\mathbf{\alpha} \circ (\mathbf{\beta} \circ (\mathbf{\gamma} \circ (\mathbf{\delta} \circ (\mathbf{\epsilon} \circ (\mathbf{\zeta} \circ (\mathbf{\eta} \circ (\mathbf{\theta} \circ (\mathbf{\iota} \circ (\mathbf{\kappa} \circ (\mathbf{\lambda} \circ (\mathbf{\mu} \circ (\mathbf{\nu} \circ (\mathbf{\xi} \circ (\mathbf{\omicron} \circ (\mathbf{\pi} \circ (\mathbf{\rho} \circ (\mathbf{\sigma} \circ (\mathbf{\tau} \circ (\mathbf{\upsilon} \circ (\mathbf{\phi} \circ (\mathbf{\chi} \circ (\mathbf{\psi} \circ (\mathbf{\omega})})}))}))}))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))}}}
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T|T)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T))) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)) = \beta \eta \]
\[ \times^{\alpha}_{\beta}(\lambda x.((\lambda y.\beta(Y_i;x)|T)) = \beta \eta \]

**Lemma 3.11** The collection \( \kappa^A \) (see definition 3.4) of terms is just the set of representatives of nontrivial trees in \( A \).

**Proof.** By induction of the construction of elements from \( \kappa^A \). The constructions are represented (see definition 3.3). Projections are represented. Let \( f : A^n \to A \) be a function represented by \( F \in \kappa^A \). Let functions \( g_1 : A^k \to A, ..., g_n : A^k \to A \) be represented by terms \( G_1, ..., G_n \) from \( \kappa^A \). A function \( h \) is defined by \( h(e_1, ..., e_k) = f(g_1(e_1, ..., e_k), ..., g_n(e_1, ..., e_k)) \) is represented by term \( \lambda x.(G_1[T_1]...[G_n[T_n]]) \). By simple induction on the construction on \( F \) we can check that \( H \in \kappa^A \).

**Lemma 3.12** Let \( A = [a_1, ..., a_n] \) be an algebra. If \( \overline{G} \) is a closed term of type \( \tau^A \to \tau^A \) representing a nontermal tree in the algebra \( A \) then for every closed term \( \overline{F} \) terms \( \lambda x.(\overline{G} \circ \lambda x.(\overline{F})) x \) and \( \overline{G} \circ \overline{F} \) are \( \beta \eta \) convertible.

**Proof.** It is easy to check that both terms have the same type \( \tau^A \). The proof is by induction on the construction of terms from the set \( \kappa^A \). (see definition 3.4)

If \( \overline{G} \) is \( \lambda x.(\overline{F}) x \)

\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{F}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{F}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{F}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{F}) x) = \beta \eta \]

If \( \overline{G} \) is \( \lambda x.(\overline{G}) x \) when \( \alpha = 0 \) then

\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]
\[ \lambda x.((\lambda x.(\overline{G}) x) = \beta \eta \]

Let \( \overline{F} \) be a closed term of type \( \tau^A \to \tau^A \) such that every \( \overline{F}_j \in \kappa^A \) for \( j \leq \alpha \). For induction we assume that every \( \overline{F}_j \) satisfy lemma which means that
\( \lambda x. \overline{D}(\lambda z. \overline{F}x) x \) and \( \overline{DF} \) are \( \beta \eta \) convertible. We want to prove that the lemma holds for term \( \overline{F} = \lambda x. x_i(Dsx) \).

\[
\begin{align*}
\lambda x. \overline{E}(\lambda z. \overline{F}x) x &= _{\beta \eta} \\
\lambda x. ((\lambda y. y_i(Dsy)) (\lambda z. \overline{F}x) x) &= _{\beta \eta} \\
\lambda x. x_i(\overline{D}(\lambda z. \overline{F}x) x) &= _{\beta \eta} \\
\lambda x. x_i((\lambda y. \overline{D}(\lambda z. \overline{F}y) y) x) &= _{\beta \eta} \\
\lambda x. x_i(D\overline{F}x) &= _{\beta \eta} \\
\end{align*}
\]

**Lemma 3.13** Let \( A = [\alpha_1, \ldots, \alpha_n] \) be an algebra. Let \( \overline{B} \) be a product of algebras represented by type \( \tau^B \). Let \( C = [\gamma_1, \ldots, \gamma_m] \) be an algebra. Let \( \overline{F} \) be a closed term of type \( x^a_{i-1}(\tau^C)^{\alpha_i} \to \tau^B \to \tau^C \) such that for every closed term \( \overline{G} \) of type \( \tau_1^B \) terms \( G_i = \lambda \overline{z} \overline{x}, \tau_1^C \tau T \) of types \( (\tau_1^C)^{\alpha_i} \to \tau_1^C \) for all \( i \leq a \) represents nonterminal trees in algebra \( C \). Let \( \overline{T} \) be a closed term of type \( \tau^A \to \tau^B \to \tau^C \) defined by \( \lambda STx.S(\lambda R \bullet (\overline{F} \circ (\lambda \overline{z} \overline{R})) \overline{T}x) \) (see lemma 3.9). For every closed term \( \overline{T} \) of type \( \tau^A \) and every closed term \( \overline{Y} \) of type \( x^a_{i-1}(\tau^A)^{\alpha_i} \), the terms \( \overline{H}^i \circ (\overline{Cons}^\tau \circ \overline{Y}) \overline{T} \) and \( \overline{F} \circ ((\lambda x_{i-1}(\overline{H}^a \circ \overline{Y}_i) \overline{T} \overline{T}) \overline{T} \overline{T} \) have the same type \( (\tau^A)^{\alpha_i} \). Let \( \overline{T} \) and \( \overline{Y} \) be closed terms of appropriate types. By lemma 3.8 the term \( \lambda STx.S(\lambda R \bullet (\overline{F} \circ (\lambda \overline{z} \overline{R})) \overline{T}x) \) is well typed in \( \tau^A \to \tau^B \to \tau^A \).

\[
\begin{align*}
(\overline{H}^i \circ (\overline{Cons}^\tau \circ \overline{Y}) \overline{T}) &= \_ {\beta \eta} \\
[((x^a_{i-1}(\overline{T}^i) \circ (\overline{x}^a_{i-1}(\overline{Cons}^\tau)) \circ (x^a_{i-1}(\overline{Y}_i))) \overline{T}) &= \_ {\beta \eta} \\
(((x^a_{i-1}(\overline{H}^i) \circ (\overline{x}^a_{i-1}(\overline{Cons}^\tau) \overline{Y}_i)) \overline{T}) &= \_ {\beta \eta} \\
((x^a_{i-1}(\overline{H}^i \circ (\overline{Cons}^\tau) \overline{Y}_i)) \overline{T}) &= \_ {\beta \eta} \\
x^a_{i-1}(\lambda x. [(\overline{Cons}^\tau \overline{Y}_i) \overline{R} \bullet ((\overline{F} \circ (\lambda \overline{z} \overline{R})) \overline{T}x)]) &= \_ {\beta \eta} \\
\end{align*}
\]

In lemmas 3.7 and 3.8 we checked that terms \( (\overline{H}^i \circ (\overline{Cons}^\tau \circ \overline{Y}) \overline{T}) \) and \( \overline{F} \circ ((\lambda x_{i-1}(\overline{H}^a \circ \overline{Y}_i) \overline{T} \overline{T} \overline{T} \overline{T} \overline{T}) \overline{T} \overline{T} \overline{T} \overline{T} \) have the same type \( (\tau^A)^{\alpha_i} \). Let \( \overline{H}^i \) and \( \overline{Y} \) be closed terms of appropriate types. By lemma 3.8 the term \( \lambda STx.S(\lambda R \bullet (\overline{F} \circ (\lambda \overline{z} \overline{R})) \overline{T}x) \) is well typed in \( \tau^A \to \tau^B \to \tau^A \).
\[ x^\rho_{x-1}(\lambda x.[(\lambda R \cdot G_i(\lambda z.R)\cdot x)\cdot \overline{y_i}(\lambda R \cdot (\lambda z.R) \cdot \overline{T}x))]) = \beta_\eta \]
\[ x^\rho_{x-1}(\lambda x.[G_i(\lambda z.\overline{y_i}(\lambda R \cdot (\lambda z.R) \cdot \overline{T}x))]) = \beta_\eta \]
\[ x^\rho_{x-1}(\lambda x.[(G_i(\lambda z.((\overline{F} \cdot \overline{T}) \cdot \overline{T}x))\cdot x]) = \beta_\eta \]

\[ F \circ \langle x^\rho_{x-1}(\overline{F} \cdot \overline{T}) \cdot \overline{T} \rangle \]
\[ x^\rho_{x-1}(\overline{F} \cdot \overline{T}) \cdot \overline{T} = \beta_\eta \]
\[ F \circ \langle F \circ \langle \overline{F} \cdot \overline{T} \rangle \cdot \overline{T} \rangle = \beta_\eta \]

Lemma 3.14 Let \( A = [\alpha_1, \ldots, \alpha_n] \) be an algebra. Let \( \overline{F} \) be a product of algebras represented by type \( T_1 \). Let \( C = [\gamma_1, \ldots, \gamma_n] \) be an algebra. Let \( \overline{T} \) be a closed term of type \( \overline{T} \rightarrow (x^\alpha_{x-1}T_1^\alpha \rightarrow x^\alpha_{x-1}T_1^\alpha) \). Let \( \overline{F} \) be a closed term of type \( \times_{x-1}^\alpha((\tau^C)^{\alpha_l} \rightarrow \tau^C) \) defined by \( \overline{F} = \lambda X \cdot (\lambda T x.((\overline{T} \cdot \overline{T}x) \cdot \overline{y}_i)) \). For every closed term \( \overline{T} \) a closed term \( \overline{C} \) defined as \( \lambda X \cdot (\overline{F} \cdot \overline{X}) \cdot \overline{T} \) of type \( \times_{x-1}^\alpha((\tau^C)^{\alpha_l} \rightarrow \tau^C) \) represents a tuple of nonterminal trees in the algebra \( C \).

Proof. Type checking for term \( \overline{F} \) has been done in Lemma 3.5. We prove that \( \overline{G} \) represents a nonterminal tree the algebra \( C \) for every \( r \leq \alpha \). Let \( \overline{T} \) be a closed term. The term \( \overline{G} \) is \( \lambda X \cdot \overline{F} \cdot \overline{T} \cdot \overline{T}x \cdot \overline{x}_r \cdot \overline{y}_r \). \( \overline{T} \cdot \overline{F} \) is a simple closed term of type \( \times_{x-1}^\alpha((\tau^C)^{\alpha_l} \rightarrow \tau^C) \rightarrow (\tau^C)^{\alpha_l} \rightarrow \tau^C \). Since \( \text{rank}(\times_{x-1}^\alpha((\tau^C)^{\alpha_l} \rightarrow (\tau^C)^{\alpha_l} \rightarrow \tau^C)) \leq 2 \) then there is a finite term grammar which produces all closed terms of this type. Consult for details [Zai87] page 4 Lemma 2.3. The grammar is following: Let \( q \in (\tau^C)^{\alpha_l} \cdot x \in \times_{x-1}^\alpha(T_1^\alpha) \) and let \( K \) be a variable.

\[
K \Rightarrow \lambda x q, q_1 \\
\vdots \\
K \Rightarrow \lambda x q_{n_r} \\
K \Rightarrow \lambda x q_j \\
K \Rightarrow \lambda x (q_j (Kx) \ldots (Kx)) \quad \text{when } \gamma_j = 0 \\
K \Rightarrow \lambda x (q_j (Kx) \ldots (Kx)) \quad \text{when } \gamma_j > 0 \\
\gamma_j \text{ times}
\]

This grammar produces all closed terms of the type \( \times_{x-1}^\alpha(T_1^\alpha) \). The proof is by induction on the grammar construction of the term \( K = \overline{T} \cdot \overline{F} \). Case 1. If \( \overline{T} \cdot \overline{F} \) is \( \lambda x q_p \) for \( p \leq \alpha_r \) then \( \overline{G} = \lambda x r.((X_r)p) = \beta_\eta \lambda X r.((X_r)p) \). Therefore \( \overline{G} \) represents a projection. Case 2. If \( \overline{T} \cdot \overline{F} \) is \( \lambda x q_j \) for \( \gamma_j = 0 \) then \( \overline{G} = \lambda x r.((X_r)p \cdot (X_r) x_j) = \beta_\eta \lambda X r.((X_r)p \cdot (X_r) x_j) \). Therefore \( \overline{G} \) represents the constant function. Case 3 inductive step. Suppose the theorem is true for closed terms \( \overline{K}_1, \ldots, \overline{K}_n \) which means that terms \( \overline{G}_1 = \lambda X x.((X_r)p) \ldots \overline{G}_n = \lambda X x.((X_r)p) \ldots \overline{G}_n x_j \) represent
nonterminal trees. Let $G' = (G_1, \ldots, G_n)$ and $K = (K_1, \ldots, K_n)$. So we have $G = \lambda x_r x. K x (X_r x)$. Let us check that the theorem also holds for $K' = \lambda x q x (K x q)$. Let $G'$ be $\lambda x_r x. K' x (X_r x)$
\begin{align*}
G' &= \lambda x_r x. K' x (X_r x) & \text{definition of } K' \\
\lambda x_r x. x (K x (X_r x)) &= \beta q & \text{definitions of } G \text{ and } 3.3 \text{ of } \\
\lambda x_r x. Cons_i^j (G X_r x) &= \eta conversion 2.15 \\
\lambda x_r x. Cons_i^j (G X_r x)
\end{align*}

Lemma 3.15 Let $F$ be a product of algebras $B_1 \times \ldots \times B_k$ such that an algebra $B_i$ is based on the signature $S_{B_i} = [\beta_1, \ldots, \beta_n]$. Let the product $F$ is represented by the type $\tau^F$. Let $C = [\gamma_1, \ldots, \gamma_k]$ be an algebra. Let $T$ be a variable of the type $\tau^F$ and $x$ be a variable of the type $\times_{j=1}^n \gamma_j$. Every long-normal closed term $\overline{T}$ of the type $\tau^F \to \tau^C$ is in one of three possible forms
1. $P = \lambda T' x. x$ if $\gamma_i = 0$
2. $P = \lambda T' x. x (T' T)$ if $\gamma_i > 0$ for some closed term $T$ of the type $\tau^F \to (\tau^C)^{\gamma_i}$
3. $P = \lambda T' x. T (T' T)$ for some closed term $T$ of type $\tau^F \to \times_{j=1}^n \gamma_j \to \times_{j=1}^n \gamma_j$

Proof. From the definition of the long-normal form.

4. Main Result

Lemma 4.1 Let $A$ be an algebra based on signature $[\alpha_1, \ldots, \alpha_a]$. Let $\overline{B}$ be a product of algebras. Let $C$ be an algebra. Let $\overline{F}$ be a closed term of type $\times_{i=1}^a ((\tau^C)^{\alpha_i} \to \tau^C)$ representing the system of functions $f_1, \ldots, f_a$. Let $h : A \times \overline{F} \to C$ be a function defined by recursion from functions $f_1, \ldots, f_a$ (see definition 1.5.) Let $\overline{F}$ be a closed term of type $\tau^A \to \tau^C \to \tau^C$. The following two statements are equivalent:

1. Term $\overline{F}$ represents $h$

2. $(\overline{F}^a \circ (Cons^A \circ \overline{G})) \overline{F} = \beta q (\overline{F} \circ ((\times_{i=1}^a (\overline{F}^{\alpha_i} \circ \overline{Y}^i)) \overline{T} \overline{T})$

for every closed term $\overline{T}$ of type $\tau^F$ and $\overline{Y} \times_{i=1}^a (\tau^A)^{\alpha_i}$.

Proof. The second equation is a simple encoding in $\lambda$ calculus the definition of primitive recursion.

Theorem 4.2 (soundness) If $f$ is a $\lambda$ function then $f$ is $\lambda$ definable.

Proof. By induction on the construction of $\lambda$ functions. Trivially all projections and constant functions are represented and representability is preserved by composition. Let $A$ be an algebra based on signature $[a_1, \ldots, a_a]$. Always $i - th$ constructor in $A$ is represented by $Cons_i^A$ (see definition 3.3). We want to show that representability is also preserved by primitive recursion. Let $\overline{F}$ be a product of algebras. Let $C$ be an algebra. Let $f_1, \ldots, f_a$ are $\lambda$ functions
such that \( f_i : C^\alpha_i \times \overline{\mathbb{T}} \to C \) for all \( i \leq a \). Let us assume that for every \( b \in \mathbb{T} \) functions \((f_1/b) : C^{\alpha_1} \to C, ..., (f_a/b) : C^{\alpha_a} \to C\) are nonterminal trees. Let \( h : A \times \overline{\mathbb{T}} \to C \) be a function defined by primitive recursion from \( f_1, ..., f_a \). Let the system \( f_1, ..., f_a \) be represented by a closed term \( \overline{T} \) of type 
\[
\times_{i=1}^a (\tau^C)^{\alpha_i} \to \tau^\overline{T} \to \tau^C.
\]
For every \( b \in \overline{\mathbb{T}} \) we define functions \( g_1^b, ..., g_a^b \) by \( g_i^b(x) = f_i(x, b) \). Function \( g_i^b \) is represented by the term \( G_i^b = \lambda XFX^b_i \) for \( i \leq a \). Therefore the tuple term \( G^b = (G_1^b, ..., G_a^b) \) is given by \( \lambda X \bullet FX^b \). Since \( g_i^b = f_i/b \) for \( i \leq a \) are nonterminal trees in \( C \) then according to lemma 3.11 the term \( G_i^b \) belongs to \( \kappa^C \). Let \( \overline{\mathbb{T}} \) be a closed term of type \( \tau^A \to \tau^\overline{T} \to \tau^C \) defined by \( \lambda STx. S(\lambda R \bullet (\overline{\mathbb{T}} \circ (\lambda x.R || Tx)) \) (see lemma 3.9). By lemma 3.13 it holds that for every \( T \) and \( Y \), \( (\overline{\mathbb{T}} \circ (\overline{\mathbb{C}} \circ \overline{\mathbb{Y}})) = \beta^\eta (\overline{\mathbb{T}} \circ (\times_{i=1}^a (\overline{\mathbb{G}}^i \circ \overline{\mathbb{Y}}^i))) \). According to lemma 4.1 \( \overline{\mathbb{H}} \) represents \( h \). It means that the function \( h \) is \( \lambda \) definable.

**Definition 4.3 (Measures of complexity)** Let us introduce a complexity measure \( \rho \) for closed terms. If \( T \) is a closed term written in the long normal form and \( T \) is a projection \( \lambda x_1...x_n.x_i \) then \( \pi(T) = 0 \). If \( T = \lambda x_1...x_n.x_iI_1...I_k \) then \( \pi(T) = \max_{j=1...k}(\pi(\lambda x_1...x_n.T_j)) + 1 \). In fact \( \pi \) corresponds with the height of Böhm trees for a term \( T \). Let us introduce also a special measure of complexity \( \rho \) which apply only to closed terms of type \( \tau^\overline{T} \to \tau^C \) for any product of algebras \( \overline{\mathbb{T}} \) and for any algebra \( C = [\gamma_1, ..., \gamma_n] \). Let \( A \) be the \( i-th \) algebra in this product \( \overline{\mathbb{T}} \), and let \( A \) be based on signature [\( \alpha_1, ..., \alpha_n \)]. Let \( \lambda T.x.X \) be a closed term in the long normal form of \( \tau^\overline{T} \to \tau^C \) type where \( T = [T_1, ..., T_k] \) is a variable of type \( \tau^\overline{T} \), \( x = [x_1, ..., x_c] \) is a variable of type \( \times_{i=1}^c \tau_i^C \) and \( X \) is a term of type \( \delta \).

By \( \rho(\lambda T.x.X) \) we mean a number of such occurrences of \( T_1, ..., T_k \) in the long normal form of the term \( X \) that any \( T_j \) for \( j \leq k \) does not occur in a context \( T_jx \). A formal definition is the following:

\[
\begin{align*}
\rho(\lambda T.x_1.x_i) & = 0 & \text{for all } i \leq c \text{ such that } \gamma_i = 0 \\
\rho(\lambda T.x_1.x_i(\overline{T}x)) & = \sum_{j=1}^n \rho(\overline{T}_j) & \text{for } \gamma_i > 0 \text{ where } \overline{T}_j \text{ for } j \leq \gamma_i \\
\rho(\lambda T.x_1.T_1(\overline{T}x)) & = 0 & \text{if } \overline{T}x = \beta^\eta x \\
\rho(\lambda T.x_1.T_1(\overline{TT}x)) & = 1 + \sum_{j=1}^n \rho(\overline{T}_j) & \text{if } \overline{TT}x \neq \beta^\eta x \text{ where } I_1, ..., I_n \text{ are closed terms of types } \tau^\overline{T} \to \tau^C \to \tau^C \to \tau^C \to \tau^C ... \text{ respectively (see definition 1.1)}
\end{align*}
\]

In the next theorem we are going to design a procedure which reduces the problem of representability of a closed term to possibly few “simpler” problems. For reason of termination of this procedure we are going to investigate some quasi-order of terms. Let us consider the set of pairs of natural numbers well-ordered in the ordinal \( \omega \times \omega \). For every closed term \( \overline{\mathbb{T}} \) of type \( \tau^\overline{T} \to \tau^C \) where \( \overline{\mathbb{T}} \) is
a product of algebras and \( C \) is an algebra we define the pair of two numbers \((\rho(\overline{P}), \pi(\overline{P}))\). In the completeness theorem we will see that the procedure works in this way that the pair is decreasing in the ordinal \( \omega \times \omega \).

**Definition 4.4** Let \( \overline{B} \) and \( \overline{B'} \) be two products of algebras. Let \( C \) and \( C' \) be an algebra. Let \( \overline{P} \) be a term of the type \( \tau \overline{B} \rightarrow \tau C \) and \( \overline{P'} \) be a term of the type \( \tau \overline{B'} \rightarrow \tau C' \). We call the term \( \overline{P} \) “simpler” than \( \overline{P'} \) if \((\rho(\overline{P}), \pi(\overline{P})) < (\rho(\overline{P'}), \pi(\overline{P'}))\) in the ordinal \( \omega \times \omega \). It means that \( \rho(\overline{P}) < \rho(\overline{P'}) \) or if \( \rho(\overline{P}) = \rho(\overline{P'}) \) then \( \pi(\overline{P}) < \pi(\overline{P'}) \).

**Lemma 4.5** Let \( \overline{B} \) be a product of algebras \( B_1, \ldots, B_n \). Let \( C \) be an algebra. Let \( A \) be the \( i \)-th algebra from the product \( \overline{B} \) for \( i \leq n \). Let the algebra \( A \) have the signature \( \{\alpha_1, \ldots, \alpha_s\} \). Let \( \overline{T} \) be a closed term of type \( \tau \overline{B} \rightarrow (\times_{j=1}^{s} \tau_{j}^{C}) \). Let \( \overline{P} \) be a closed term of the type \( \times_{j=1}^{s} (\tau^{C} \overline{A}) \). Let \( \overline{P} \) be a closed term of type \( \times_{j=1}^{s} (\tau^{C} \overline{A}) \) defined by \( \overline{P} = \lambda x. (\tau \overline{B} \overline{T}(\tau x) \circ (\tau C x)) \). Let \( \overline{P} \) be a closed term of type \( \tau \overline{B} \rightarrow \tau C \) defined by \( \overline{P} = \lambda T \overline{T}(\tau x). T(\tau x) \). If \( \overline{T} \neq \beta x \) then \( \rho(\overline{P}) < \rho(\overline{P'}) \) for all \( i \leq n \).

**Proof.** First of all we can notice that terms \( \overline{T} \) and \( x \) may be \( \beta \eta \) convertible only if algebras \( A \) and \( C \) are the same. The term \( \overline{P} \) has the form \( \lambda T \overline{T}(\tau x). T(\tau x) \). It is obvious that the number of occurrences of \( T' \)’s in \( \overline{P} \) is less than the number of \( T' \)’s in \( \lambda T \overline{T}(\tau x). T(\tau x) \). New variables \( Y_i \) for \( i \leq \gamma_i \) of type \( \tau C \) in the term \( \overline{P} \) are in the contexts of \( Y_i \)’s and therefore their occurrences do not increase the measure \( \rho \).

**Theorem 4.6** (completeness) If a function \( p \) is a \( \lambda \)-definable function between the product of algebras \( \overline{B} \) and the algebra \( C \) then \( p \) is a \( \lambda \)-function.

**Proof.** Let \( \overline{P} \) be a closed term of type \( \tau \overline{B} \rightarrow \tau C \) representing \( p \). We are going to prove the theorem inductively on the complexity \((\rho(\overline{P}), \pi(\overline{P}))\) of the term \( \overline{P} \). If \((\rho(\overline{P}), \pi(\overline{P})) = (0, m)\) then \( \overline{P} \) has to be in one of two possible forms, \( \lambda T \overline{T}(\tau x) \) where \( \gamma_i = 0 \) or \( \lambda T \overline{T}(\tau x) \) if \( \overline{T} \neq \beta y \). (see lemma 3.15 and definition of \( \rho \)). If \( \overline{P} = \lambda x. x \) then \( \overline{P} \) represents function \( x : \overline{B} \rightarrow C \) which maps onto \( i \)-th constructor of the algebra \( C \) constantly. Therefore \( p \) is a \( \lambda \)-function. If \( \overline{P} = \lambda T \overline{T}(\tau x) \) where \( \overline{T} \neq \beta y \) then it means that the algebra \( C \) and \( i \)-th algebra in the product \( \overline{B} \) are the same and moreover \( \overline{P} \neq \beta y \). \( \lambda T \overline{T}(\tau x) \) is \( \beta y \) \( \lambda T \overline{T}(\tau x) \). In this case \( \overline{P} \) represents projection \( p(b_1, \ldots, b_k) = b_i \) so it is a \( \lambda \)-function.

**Induction step.** Suppose the theorem is true for all closed terms \( \overline{P} \) of any of the types \( \tau \overline{B} \rightarrow \tau C \) such that \((\rho(\overline{P}), \pi(\overline{P})) < (\rho(\overline{P}), \pi(\overline{P}))\). If \( \rho(\overline{P}) > 0 \) then according to the definition of \( \rho \) and lemma 3.15 \( \overline{P} \) must be in one of two possible forms: \( \overline{P} = \lambda T \overline{T}(\tau x) \) for some closed term \( \overline{T} \) or \( \overline{P} = \lambda T \overline{T}(\tau x) \) for some closed term \( \overline{T} \) such that \( \overline{T} \neq \beta y \).

(Case 1) If \( \overline{P} = \lambda T \overline{T}(\tau x) \) for some closed term \( \overline{T} = (\overline{T}_1, \ldots, \overline{T}_r) \) then \( \pi(\overline{P}) < \pi(\overline{P}) \) for all \( r \leq \gamma_i \) and \( \rho(\overline{P}) < \rho(\overline{P}) \) for all \( r \leq \gamma_i \). All closed terms \( \overline{T}_1, \ldots, \overline{T}_r \) of the type \( \tau \overline{B} \rightarrow \tau C \) (see lemma 3.6). Therefore according to induction assumption \( \overline{T}_r \) represents function \( j_r : \overline{B} \rightarrow C \) which are \( \lambda \)-functions. We have that \( p(b_1, \ldots, b_k) = c_i(j_1(b_1, \ldots, b_k), \ldots, j_n(b_1, \ldots, b_k)) \) where \( c_i \) is the \( i \)-th
constructor of the algebra $C$ of arity $\gamma_i$. The function $p$ as the composition of $\lambda$ functions is a $\lambda$ function.

(Case 2) Let $\overline{F} = \lambda T x. T_i (\overline{T} T x)$ for some closed term $\overline{T}$ of the type $\tau^{\overline{F}} \to \times_{i=1}^n \tau_j^A$ where $A$ is the $i$ - th algebra in the product $\overline{F}$ given by signature $\{a_1, \ldots, a_n\}$. Let $\overline{T} T x \not\equiv_{\eta} x$. Let $\overline{F}$ be a closed term of type $\times_{i=1}^n ((\tau_i^C)^{a_i} \to \tau^{\overline{F}} \to \tau^C)$ defined by $\overline{F} = \lambda X \cdot (\lambda T x. ((\overline{T} T x) \circ (Y x)))$. By lemma 4.5 $\rho(\overline{F}) < \rho(\overline{P})$ for all $j \leq a$. Since any term $F_j$ is "smaller" than the term $P_j$ then all $F_j$ are representing $\lambda$ functions. Let $f_1, \ldots, f_a$ be the system of $\lambda$ functions represented by $\overline{F}$ where $f_j : C^{a_j} \times \overline{T} \to C$. By lemma 3.14 it holds that for every $\overline{T}$, closed term $\overline{C}$ defined by $\lambda X \cdot (\overline{F} \circ X)\overline{T}$ belongs to $\kappa^C$. Since $\lambda X \cdot (\overline{F} \circ X)\overline{T}$ is obtained from the system $f_1, \ldots, f_a$ by $\lambda$ functions. $\overline{T}$ represents $h$. Therefore $h$ is a $\lambda$ function.

5. Examples of Applications

The characteristic obtained in theorem 4.6 may be used in order to find the classes of $\lambda$ definable functions for some specific free algebras. Let us consider $\lambda$ definability between algebras of Church's numerals $N$ and algebra of binary words $\Sigma^*$ for some binary alphabet $\Sigma$. (Compare [Mad91]). $N$ is based on signature $\{0, 1\}$ and $\Sigma$ on $\{[1, 1, 0]\}$. The $\lambda$ definability for operations in $(\Sigma^*)^p \to \Sigma^*$ is given in [Zai87]. $\lambda$ definable word operations are those obtained by composition from $append : (\Sigma^*)^{2} \to \Sigma^*$, $substitution : (\Sigma^*)^{3} \to \Sigma^*$ and $if \ then \ else : (\Sigma^*)^{3} \to \Sigma^*$ (see [Zai87] for details). As a simple consequence of theorems 4.6 and 4.2 for $\lambda$ definable mappings in $N \to \Sigma^*$ we get following theorem:

Theorem 5.1. Let $\Sigma = \{a, b\}$ be a binary alphabet and $N$ be a set of natural numbers. Constant functions $f(n) = e$ the empty word $f(n) = a$ and $f(n) = b$ are $\lambda$ definable. Function $f(n) = a^n$ is $\lambda$ definable. If functions $f, g, h : N \to \Sigma^*$ are $\lambda$ definable then $n \to append(f(n), g(n))$, $n \to substitution(f(n), g(n), h(n))$ and $n \to if f(n) = e \ then \ g(n) \ else \ h(n)$ are also $\lambda$ definable. The functions defined above are the only $\lambda$ definable functions in $N \to \Sigma^*$.

Example 5.2 This is a continuation of example 3.1. The algebra $N$ of positive integers based on the signature $S_N = \{[1, 0]\}$ is represented by the type $\tau^N = (0 \to 0) \to (0 \to 0)$. Any number $n$ is represented by term $\lambda x. s(... s x)$. The algebra $\Sigma$ of binary words based on the signature $\{[1, 1, 0]\}$ is represented by the type $\tau^\Sigma = (0 \to 0) \to ((0 \to 0) \to (0 \to 0))$. Let $\Sigma$ be a binary alphabet $\{a, b\}$. $\lambda$ denotes the empty word in $\Sigma^*$. We are going to prove
that the term \( F = \lambda w s x . w s (\lambda y . y) x \) of the type \( \tau^2 \rightarrow \tau^N \) represents the function computing the number of letters \( a \) in the given word. Term \( F \) is of the form 
\( \lambda w s x . w s (\lambda y . y) x \) where \( \tau \) is the term of type \( \tau^2 \rightarrow \times_{k=1}^n \tau^N \rightarrow \times_{k=1}^n \tau^\sigma \) such that 
\( \tau = (\tau_1, \tau_2, \tau_3) \) where the term \( \tau_1 = \lambda w s x . y \) the term \( \tau_2 = \lambda w s x . y \) and the term \( \tau_3 = \lambda w s x . x \). Let us define the term \( F = (F_1, F_2, F_3) \) by formula \( F = \lambda y \cdot (\lambda x . (T x) \circ (Y x)) \). Therefore \( F_1 = \lambda y w s x . s (\lambda y . y) x \), \( F_2 = \lambda y w s x . s (\lambda y . y) x \) and \( F_3 = \lambda w s x . x \). Term \( F = (F_1, F_2, F_3) \) represents functions \( f_1 : N \times \Sigma \rightarrow N \), \( f_2 : N \times \Sigma \rightarrow N \) and \( f_3 : \Sigma \rightarrow N \) given by \( f_1(y, w) = 1 + y \), \( f_2(y, w) = y \) and \( f_3(w) = 0 \). So the term \( \overline{F} = \lambda S T . S . S . (\lambda y . y) (\lambda y . y) x \) represents the function \( h : (\Sigma)^2 \rightarrow N \) defined from \( f_1, f_2, f_3 \) by recursion:

\[
\begin{align*}
h(a, w) &= 1 + h(v, w) \\
h(b, v) &= h(v, w) \\
h(\Lambda, w) &= 0
\end{align*}
\]

Let \( p : \Sigma \rightarrow N \) be a function represented by the term \( F \). We have \( p(w) = h(w, w) \)

References.

[Mad91] Maudy M. On the \( \lambda \) definable functions between numbers, words and trees Fundamenta Informaticae, 1991
[Sta79] Statman R. Intuitionistic propositional logic is polynomial-space complete, Theoretical Computer Science 9, 67-72 (1979)
[Zai87] Zaionc M. Word operations definable in the typed \( \lambda \) calculus, Theoretical Computer Science 52 (1987) pp. 1-14