Lambda Representation of Operations Between Different Term Algebras[1]

Marek Zaionc[2]
zaionc@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^A and τ^B be types representing algebras A and B respectively. Any closed term of the type $\tau^A \to \tau^B$ 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 $\tau^{A_1}, ..., \tau^{A_n}, \tau^B$ then $\tau^{A_1} \to (...(\tau^{A_n} \to \tau^B)...)$ is a type of functions from the product $A_1 \times ... \times A_n$ into algebra B . Any closed term of this type is a representation of algorithm which transforms the tuple of terms of types $\tau^{A_1}, ..., \tau^{A_n}$ respectively into a term of type τ^B, which represents an object in algebra B (see [BöB85]). 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 [Sch75] [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

[1]This research is supported by KBN Grant 0384/P4/93. The version of this paper will be published in Lecture Notes in Computer Science

[2]Permanent address of the author is: Instytut Informatyki, Uniwersytet Jagielloński, Nawojki 11, 30-072 Krakow, Poland. This paper was partially prepared while author was visiting Computer Science Department at State University of New York at Buffalo, USA

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 λ calculus have been proved by Schwichtenberg in 1975 and independently by Statman (see [Sch75], [Sta79]). 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 Λ (empty word) and operations $append, substitution$ and cut. The results of Schwichtenberg and Zaionc were extended to the structure of binary trees [Zai90]. It was shown that λ 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 λ 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 α_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 $A = [\alpha_1, ..., \alpha_a]$ to say that A is an algebra given by the signature $[\alpha_1, ..., \alpha_a]$. If $A_1, ..., A_n$ are algebras then by $A_1 \times ... \times A_n$ 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 $A \times ... \times A$.

Definition 1.1 If A is an algebra given by signature $S_A = [\alpha_1, ..., \alpha_n]$ and n is a nonnegative integer then A^{+n} is an algebra given by signature $S_{A^{+n}} = [\alpha_1, ..., \alpha_n, 0, ...0]$ 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 $f(x_1, ..., x_p) = x_i$ 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 ($\alpha_i = 0$) 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 $(f/\overline{b}) : \overline{A} \rightarrow C$ defined by $(f/\overline{b})(\overline{x}) = f(\overline{x}, \overline{b})$.

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

$$h(a_i(t_1, ..., t_{\alpha_i}), \overline{b}) = f_i(h(t_1, \overline{b}), ..., h(t_{\alpha_i}, \overline{b}), \overline{b})$$

for all expressions $t_1, ..., t_{\alpha_i}$ of the algebra A

Definition 1.6 The set of λ functions is defined inductively by:

1. for all algebras $\overline{A}, B, \overline{C}$ any projection in $\overline{A} \times B \times \overline{C} \rightarrow B$ given by $f(x, y, z) = y$ is a λ function
2. any constant function from $\overline{A}$ to B given by $f(\overline{x}) = t$ for some expression $t \in B$ is a λ function.
3. for any algebra A any constructor $a_i : A^{\alpha_i} \rightarrow A$ is a λ function
4. any composition of λ functions is a λ function. i.e. if $f_1 : A_1 \times ... \times A_n \rightarrow B_1, ..., f_k : A_1 \times ... \times A_n \rightarrow B_k$ are λ functions and $g : B_1 \times ... \times B_k \rightarrow C$ is a λ function then $h : A_1 \times ... \times A_n \rightarrow C$ given by $h(\overline{x}) = g(f_1(\overline{x}), ..., f_k(\overline{x}))$ is a λ function.
5. let algebra A be given by a signature $[\alpha_1, ..., \alpha_n]$ and $\overline{B}$ be a product of algebras. If $f_1 : C^{\alpha_1} \times \overline{B} \rightarrow C, ..., f_n : C^{\alpha_n} \times \overline{B} \rightarrow C$ are λ functions such that for every expression $\overline{b} \in \overline{B}$ functions $(f_1/\overline{b}) : C^{\alpha_1} \rightarrow C, ..., (f_n/\overline{b}) : C^{\alpha_n} \rightarrow C$ are nonterminal trees in C then the function $h : A \times \overline{B} \rightarrow C$ defined by recursion (see definition 1.5) from $f_1, ..., f_n$ is a λ 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 $\tau \rightarrow \mu$ is a type. For any type τ we define numbers $rank(\tau)$ and $arg(\tau)$ as follows: arg(O)=rank(O)=0 and $arg(\tau \rightarrow \mu) = 1 + arg(\mu)$ and $rank(\tau \rightarrow \mu) = max(rank(\tau) + 1, rank(\mu))$. Associated with each type τ is a denumerable set of variables $V(\tau)$. Any variable of type τ is a term of type τ. If T is a term of type $\tau \rightarrow \mu$ 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 $\lambda x.T$ is a term of type $\tau \rightarrow \mu$. If T is a term of type τ we write $T \in \tau$. We shall use the notation $\lambda x_1...x_n.T$ for term $\lambda x_1.(\lambda x_2.(...(\lambda x_n.T)...))$ and $TS_1...S_n$ for $(...(TS_1)...S_n)$. 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 $\alpha\beta\eta$ conversions (see [Bar81] or [Fri75]) and the convertible terms are written as $T =_{\beta\eta} S$. By the $Cl(\tau)$ we mean the set of all closed terms (without free variables) of type τ. Term T is in the long normal form if $T = \lambda x_1...x_n.yT_1...T_k$ where y is an x_i for some $i \leq n$ or y is a free variable, T_j for $j \leq k$ are in the long normal form and $yT_1...T_k$ is a term of type O. Long normal forms exist and are unique for $\beta\eta$ conversions.

In order to easily represent and manipulate on finite strings of terms of potentially different types we extend our language by adding new constructor 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 $\tau_1, ..., \tau_n \in TYPE^*$ then $(\tau_1, ..., \tau_n) \in TYPE^*$. The empty tuple is denoted by $\omega \in TYPE^*$. We use the abbreviation $\times_{i=1}^n \tau_i$ for string $(\tau_1, ..., \tau_n)$. We assume that string of empty types is empty, so $\times_{i=1}^n \omega = \omega$. In the case when all τ_i are the same we use notation τ^n instead of $\times_{i=1}^n \tau$. We also extend the type formation $\rightarrow$ for string types as follows: $\times_{i=1}^n \tau_i \rightarrow \mu$ means $\tau_1 \rightarrow (\tau_2 \rightarrow ...(\tau_n \rightarrow \mu)...))$ including $\omega \rightarrow \mu = \mu$ and $\tau \rightarrow \times_{j=1}^m \mu_j$ means $\times_{j=1}^m (\tau \rightarrow \mu_j)$ including $\tau \rightarrow \omega = \omega$. Particularly we have $(\times_{i=1}^n \tau_i) \rightarrow (\times_{j=1}^m \mu_j) = \times_{j=1}^m (\times_{i=1}^n \tau_i \rightarrow \mu_j)$ and $\tau^0 \rightarrow \mu = \mu$. Since types are usually employed for describing construction of function spaces the above definitions are in fact typical identifications. The first equation identifies $A^{B \times C}$ with A^{B^C} which usually is called "Currying". The second equates $(A \times B)^C$ with $A^C \times B^C$. 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 $\times_{i=1}^n \tau_i \rightarrow O$ where τ_i are again simple types called components of τ. We can prove several equation between types, for example

$(\tau \rightarrow \mu)^n = \tau \rightarrow \mu^n$	2.1
$(\tau^k \rightarrow \mu)^n = \tau^k \rightarrow \mu^n$ the special case of 2.1	2.2
for simple type τ of arity k,	
$\tau^n = (\times_{i=1}^k \tau_i \rightarrow O)^n = \times_{i=1}^k \tau_i \rightarrow O^n$	2.3

for simple type τ of arity k, where τ_i are components of τ

$$\times_{j=1}^{n}\tau^{\alpha_j} = \times_{j=1}^{n}(\times_{i=1}^{k}\tau_i \to O)^{\alpha_j} = \times_{j=1}^{n}(\times_{i=1}^{k}\tau_i \to O^{\alpha_j}) =$$
$$= \times_{i=1}^{k}\tau_i \to \times_{j=1}^{n}O^{\alpha_j} \qquad 2.4$$

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 $\times_{i=1}^{n}M_i$ of the type $\times_{i=1}^{n}\tau_i$. We use the same notation $M \in \tau$ to say that M_i is a term of type τ_i for $i \leq n$. This definition may by iterated so we may consider strings of strings of terms and so on. The empty tuple of terms is denoted by Ω of type ω. In the case when all M_i are the same we use M^n instead of $\times_{i=1}^{n}M$ with $\Omega^n = \Omega$. Term formation is following: If $M \in \times_{i=1}^{n}\tau_i \to \mu$ and $N \in \times_{i=1}^{n}\tau_i$ then by MN we mean $MN_1...N_n$ of type μ with $M\Omega = M$ when $\times_{i=1}^{n}\tau_i = \omega$ (n=0). If $M \in \tau \to \times_{j=1}^{m}\mu_j$ and $N \in \tau$ then by MN we mean $(M_1N, ..., M_mN)$ or equivalently $\times_{i=1}^{n}M_iN$ of type $\times_{j=1}^{m}\mu_j$ with ΩN=Ω in case when m=0. If $M \in \mu$ and x is a variable of type $\times_{i=1}^{n}\tau_i$ (which means that x has a form of tuple term $(x_1, ..., x_n)$) then by $\lambda x.M$ we mean $\lambda x_1...x_n.M$ of type $\times_{i=1}^{n}\tau_i \to \mu$ with $\lambda x_\omega.M = M$ for variable x of type ω. If $M \in \times_{j=1}^{m}\mu_j$ and x is a variable of type τ then by $\lambda x.M$ we mean $(\lambda x.M_1, ..., \lambda x.M_m)$ which is identical with $\times_{i=1}^{n}\lambda x.M_i$ of type $\tau \to \times_{j=1}^{m}\mu_j$ with $\lambda x.\Omega = \Omega$ when m=0. If M is a term of type $\times_{i=1}^{n}(\tau_i \to \mu_i)$ and N is a term of type $\times_{i=1}^{n}\tau_i$ then by parallel application $M \diamond N$ we mean the term $\times_{i=1}^{n}M_iN_i$ of type $\times_{i=1}^{n}\mu_i$. If $x = (x_1...x_n)$ is a variable of type $\times_{i=1}^{n}\tau_i$ and $M \in \times_{i=1}^{n}\mu_i$ then by parallel abstraction $\lambda x \bullet M$ we mean the term $(\lambda x_1.M_1, ..., \lambda x_n.M_n)$ of type $\times_{i=1}^{n}(\tau_i \to \mu_i)$. We may summarize all those definitions by:

$$M(\times_{i=1}^{n}N_i) = MN_1...N_n \text{ with } M\Omega = M \qquad 2.5$$
$$(\times_{i=1}^{n}M_i)N = \times_{i=1}^{n}(M_iN) \text{ with } \Omega N = \Omega \qquad 2.6$$
$$\lambda(\times_{i=1}^{n}x_i).M = \lambda x_1...x_n.M \text{ with } \lambda x_\omega.M = M \qquad 2.7$$
$$\lambda x.(\times_{i=1}^{n}M_i) = \times_{i=1}^{n}(\lambda x.M_i) \text{ with } \lambda x.\Omega = \Omega \qquad 2.8$$
$$(\times_{i=1}^{n}M_i) \diamond (\times_{i=1}^{n}N_i) = \times_{i=1}^{n}(M_iN_i) \qquad 2.9$$
$$\lambda(\times_{i=1}^{n}x_i) \bullet (\times_{i=1}^{n}M_i) = \times_{i=1}^{n}(\lambda x_i.M_i) \qquad 2.10$$

By simple term we mean a term of a simple type. If M is a simple term and x is a variable of type $\times_{i=1}^{n}\tau_i$ and $N \in \times_{i=1}^{n}\tau_i$ then $M(x/N)$ denotes the term obtain by simultaneous substitution of the terms N_i for each free occurrences of x_i respectively in M. If $M \in \times_{i=1}^{k}\mu_i$ then $M(x/N)$ means $(M_1(x/N), ..., M_k(x/N))$ and if n=k then $M(x//N)$ means $(M_1(x_1/N_1), ..., M_k(x_n/N_n))$. 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 $\times_{i=1}^{n}\mu_i$ 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 $\times_{i=1}^{n}\mu_i$ 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

$\lambda x.M =_{\beta\eta} \lambda y.M(x/y)$	2.11 sequential α conversion
$\lambda x \bullet M =_{\beta\eta} \lambda y \bullet M(x//y)$	2.12 parallel α conversion
$(\lambda x.M)N =_{\beta\eta} M(x/N)$	2.13 (sequential β conversion)
$(\lambda x \bullet M) \diamond N =_{\beta\eta} M(x//N)$	2.14 parallel β conversion
$\lambda x.(Mx) =_{\beta\eta} M$	2.15 sequential η conversion
$\lambda x \bullet (M \diamond x) =_{\beta\eta} M$	2.16 parallel η conversion

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

$(MN)_i = M_i N$	2.17
$(\lambda x.M)_i = \lambda x.M_i$	2.18
$(M \diamond N)_i = M_i N_i$	2.19
$(\lambda x \bullet M)_i = \lambda x_i.M_i$	2.20
$(\times_{j=1}^{n} M_j)_i = M_i$	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 τ^A we mean a type $(O^{\alpha_1} \to O) \to ... \to (O^{\alpha_a} \to O) \to O$. By τ_i^A for $i \leq a$ we mean i-th component of type τ^A i.e. $\tau_i^A = O^{\alpha_i} \to O$. We will see that closed terms of this type reflect constructions in algebra A. Assuming that at least one α_i is 0 we have that τ^A is not empty. τ^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 τ^A. Let $c_1, ..., c_n$ are all constructors in term algebra, of arity $\alpha_1, ..., \alpha_a$ respectively. If a_i 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 $T_1, ..., T_{\alpha_i}$ of type τ^A, 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 τ^A and expressions of algebra A. The unique (up to $\beta\eta$ conversion) term of type τ^A which represents an expression t in algebra A is denoted by $\underline{t}$. 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 $\tau^{A_1} \to ... \to \tau^{A_n} \to \tau^B$ if for all expressions $t_1 \in A_1, ..., t_n \in A_n$, the following terms are $\beta\eta$ convertible

$$H\underline{t_1}...\underline{t_n} =_{\beta\eta} \underline{h(t_1, ..., t_n)}.$$

Let $\overline{B}$ be a product of algebras $B_1 \times ... \times B_k$. We define a $\tau^{\overline{B}}$ to be the type $\times_{i=1}^{k}\tau^{B_i}$. By analogy, there is a natural isomorphism between terms of type $\tau^{\overline{B}}$ 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 $\lambda sx.s(...sx)$. The algebra Σ of binary words based on the signature $S_\Sigma = [1,1,0]$ 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 $\lambda uvx.u(v(ux))$. The term $\lambda wsx.ws(\lambda y.y)x$ 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 ϵ 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 τ^U is $(O \to (O \to O)) \to (O \to O)$ and $\tau^N = (O \to O) \to (O \to O)$. Let H be a closed term $\lambda Tux.T(\lambda yz.uy)x$ of type $\tau^U \to \tau^N$. It is easy to see that H represents the function $leftmost : U \to N$ which computes the length of the leftmost path of a tree.

$$\text{leftmost}(\epsilon)=0$$
$$\text{leftmost}(t_1{}^\wedge t_2)=\text{leftmost}(t_1)+1$$

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

Definition 3.3 Let A be an algebra based on signature $[\alpha_1, ..., \alpha_a]$. Let Z_i be a variable of type $(\tau^A)^{\alpha_i}$ for $i \le a$ and x be a variable of type $\times_{i=1}^a (O^{\alpha_i} \to O)$. Let $\overline{Cons^A}$ be a closed term of type $\times_{i=1}^a ((\tau^A)^{\alpha_i} \to \tau^A)$, defined by $\overline{Cons^A} = (\overline{Cons_1^A}, ..., \overline{Cons_a^A})$, where $\overline{Cons_i^A}$, of the type $(\tau^A)^{\alpha_i} \to \tau^A$, is defined by $\overline{Cons_i^A} = \lambda Z_i x.x_i(Z_i x)$. Note that $\overline{Cons_i^A}$ represents constructor a_i which can be seen as a function $a_i : A^{\alpha_i} \to A$. Note also that when $\alpha_i = 0$, $\overline{Cons_i^A}$ is a projection $\lambda x.x_i$ which represents constant a_i.

Definition 3.4 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let κ^A be a collection of closed terms of types $(\tau^A)^p \to \tau^A$ for all $p \ge 0$ defined by recursion as the minimal set containing projections $\lambda s.s_i$ and constant functions $\lambda s.\overline{Cons_i^A}$ if $\alpha_i = 0$, and satisfying following property: If $\overline{D}$ is a closed term of type $(\tau^A)^p \to (\tau^A)^{\alpha_i}$ for $\alpha_i > 0$, such that $\overline{D}_j \in \kappa^A$ for all $j \le \alpha_i$, then a closed term $\lambda s.\overline{Cons_i^A}(\overline{D}s)$ of type $(\tau^A)^p \to \tau^A$ belongs to κ^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 $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. For every closed term

$\overline{I}$ of the type $\tau^{\overline{B}} \to (\times_{j=1}^{c}\tau_j^C \to \times_{i=1}^{a}\tau_i^A)$ a term $\lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$ is well-formed in type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$.

Lemma 3.6 Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. For every closed term $\overline{J}$ of the type $\tau^{\overline{B}} \to (\tau^C)^{\gamma_i}$ for $i \le c$ the closed term $\lambda Tx.x_i(\overline{J}Tx)$ is well-formed in type $\tau^{\overline{B}} \to \tau^C$.

Lemma 3.7 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. For every closed term $\overline{T}$ of type $\tau^{\overline{B}}$, for every closed $\overline{Y}$ of type $\times_{i=1}^{a}(\tau^A)^{\alpha_i}$ and every closed $\overline{H}$ of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ and $\overline{F}$ of type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$ the closed term $(\overline{F} \diamond \times_{i=1}^{a}((\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T}))\overline{T}$ is well-formed in type $(\tau^C)^a$.

Lemma 3.8 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. For every closed terms $\overline{T}$ of type $\tau^{\overline{B}}$, for every closed $\overline{Y}$ of type $\times_{i=1}^{a}(\tau^A)^{\alpha_i}$ and every closed $\overline{H}$ of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ the closed term $(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T}$ is well-formed in type $(\tau^C)^a$.
Proof. See the definition 3.3.

Lemma 3.9 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. For every closed term $\overline{F}$ of the type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$ a closed term $\lambda STx.S(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))Tx)$ is well-formed in type $\tau^A \to \tau^{\overline{B}} \to \tau^C$.

Lemma 3.10 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. Let $\overline{I}$ be a closed term of type $\tau^{\overline{B}} \to (\times_{j=1}^{c}\tau_j^C \to \times_{i=1}^{a}\tau_i^A)$. Let $\overline{F}$ be a closed term of type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$ defined by $\overline{F} = \lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$. Let $\overline{H}$ be a closed term of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ defined by $\overline{H} = \lambda STx.S(\overline{I}Tx)$. For every closed term $\overline{T}$ of type $\tau^{\overline{B}}$ and every closed term $\overline{Y}$ of type $\times_{i=1}^{a}(\tau^A)^{\alpha_i}$, the terms $(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T}$ and $(\overline{F} \diamond ((\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y}_i))\overline{T})\overline{T}$ are $\beta\eta$ convertible.
Proof. The term $\overline{Cons^C}$ is defined in 3.3. In lemmas 3.7 and 3.8 we checked that both terms have the same type $(\tau^C)^a$. We must remember that since $\overline{Y}$ is a closed term of type $\times_{i=1}^{a}(\tau^A)^{\alpha_i}$, then $\overline{Y}$ has a form $\times_{i=1}^{a}\overline{Y}_i =_{\beta\eta} \times_{i=1}^{a}\times_{j=1}^{\alpha_i}(\overline{Y}_i)_j$

$(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T} =_{\beta\eta}$	definition 2.9
$((\times_{i=1}^{a}\overline{H}) \diamond (\times_{i=1}^{a}\overline{Cons_i^A \overline{Y}_i}))\overline{T} =_{\beta\eta}$	definition 2.9
$(\times_{i=1}^{a}(\overline{H}(\overline{Cons_i^A \overline{Y}_i}))\overline{T} =_{\beta\eta}$	definition 2.6
$\times_{i=1}^{a}\overline{H}(\overline{Cons_i^A \overline{Y}_i})\overline{T} =_{\beta\eta}$	definition of $\overline{H}$
$\times_{i=1}^{a}(\lambda x.(\overline{Cons_i^A \overline{Y}_i})(\overline{IT}x)) =_{\beta\eta}$	definition 3.3 of $\overline{Cons_i^A}$

$$
\begin{array}{ll}
\times_{i=1}^{a}(\lambda x.((\lambda y.y_i(\overline{Y}_i y))(\overline{IT}x))) =_{\beta\eta} & \beta \text{ conversion } 2.13 \\
\times_{i=1}^{a}(\lambda x.((\overline{IT}x)_i(\overline{Y}_i(\overline{IT}x))) =_{\beta\eta} & \text{definition } 2.17 \\
\times_{i=1}^{a}(\lambda x.((\overline{I}_i\overline{T}x)(\overline{Y}_i(\overline{IT}x))) =_{\beta\eta} & \text{definition of } \overline{Y}_i \\
\times_{i=1}^{a}(\lambda x.((\overline{I}_i\overline{T}x)((\times_{j=1}^{\alpha_i}(\overline{Y}_i)_j)(\overline{IT}x))) =_{\beta\eta} & \text{definition } 2.6 \\
\times_{i=1}^{a}(\lambda x.((\overline{I}_i\overline{T}x)(\times_{j=1}^{\alpha_i}((\overline{Y}_i)_j(\overline{IT}x)))) =_{\beta\eta} & \text{definition } \overline{H} \\
\times_{i=1}^{a}(\lambda x.(\overline{I}_i\overline{T}x)(\times_{j=1}^{\alpha_i}(\overline{H}(\overline{Y}_i)_j\overline{T}x))) =_{\beta\eta} & \text{definition } 2.6 \\
\times_{i=1}^{a}(\lambda x.(\overline{I}_i\overline{T}x)((\times_{j=1}^{\alpha_i}\overline{H}(\overline{Y}_i)_j)\overline{T}x)) =_{\beta\eta} & \text{definition } 2.6 \\
\times_{i=1}^{a}(\lambda x.(\overline{I}_i\overline{T}x)(((\times_{j=1}^{\alpha_i}\overline{H})\diamond(\times_{j=1}^{\alpha_i}(\overline{Y}_i)_j))\overline{T}x)) =_{\beta\eta} & \text{form of } \overline{Y}_i \\
\times_{i=1}^{a}(\lambda x.(\overline{I}_i\overline{T}x)((\overline{H}^{\alpha_i}\diamond\overline{Y}_i)\overline{T}x)) =_{\beta\eta} & \text{definition of } \overline{F}_i \\
\times_{i=1}^{a}(\lambda x.\overline{F}_i((\overline{H}^{\alpha_i}\diamond\overline{Y}_i)\overline{T})\overline{T}x) =_{\beta\eta} & \eta \text{ conversion } 2.15 \\
\times_{i=1}^{a}(\overline{F}_i((\overline{H}^{\alpha_i}\diamond\overline{Y}_i)\overline{T})\overline{T}) =_{\beta\eta} & \text{definition } 2.9 \\
((\times_{i=1}^{a}\overline{F}_i)\diamond\times_{i=1}^{a}((\overline{H}^{\alpha_i}\diamond\overline{Y}_i)\overline{T}))\overline{T} =_{\beta\eta} & \text{definition of } \overline{F} \\
(\overline{F}\diamond\times_{i=1}^{a}((\overline{H}^{\alpha_i}\diamond\overline{Y}_i)\overline{T}))\overline{T} &
\end{array}
$$

Lemma 3.11 The collection κ^A (see definition 3.4) of terms is just the set of representatives of nonterminal trees in A.

Proof. By induction of the construction of elements from κ^A. The constructors 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 κ^A. A function h defined by $h(e_1, ..., e_k) = f(g_1(e_1, ..., e_k), .., g(e_1, ..., e_k))$ is represented by term $\lambda T.F(G_1T)...(G_nT)$. By simple induction on the construction on F we can check that $H \in \kappa^A$.

Lemma 3.12 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. If $\overline{G}$ is a closed term of type $(\tau^A)^p \to \tau^A$ representing a nonterminal tree in the algebra A then for every closed term $\overline{P}$ terms $\lambda x.\overline{G}(\lambda z.\overline{P}x)x$ and $\overline{G}\ \overline{P}$ are $\beta\eta$ convertible.

Proof. It is easy to check that both terms have the same type τ^A. The proof is by induction on the construction of term from the set κ^A. (see definition 3.4) If $\overline{G}$ is $\lambda s.s_i$ then

$$
\begin{array}{ll}
\lambda x.\overline{G}(\lambda z.\overline{P}x)x =_{\beta\eta} & \text{definition of } \overline{G} \\
\lambda x.((\lambda s.s_i)(\lambda z.\overline{P}x)x) =_{\beta\eta} & \beta \text{ conversion } 2.13 \\
\lambda x.((\lambda z.\overline{P}x)_i)x) =_{\beta\eta} & \text{formulas } 2.18 \text{ and } 2.17 \\
\lambda x.((\lambda z.\overline{P}_i x)x) =_{\beta\eta} & \beta \text{ conversion } 2.13 \\
\lambda x.(\overline{P}_i x) =_{\beta\eta} & \eta \text{ conversion } 2.15 \\
\overline{P}_i =_{\beta\eta} & \text{definition of } \overline{G} \\
\overline{GP} &
\end{array}
$$

If $\overline{G}$ is $\lambda s.\overline{Cons_i^A}$ when $\alpha_i = 0$ then

$$\lambda x.\overline{G}(\lambda z.\overline{P}x)x =_{\beta\eta} \lambda x.((\lambda s.\overline{Cons_i^A})(\lambda z.\overline{P}x)x) =_{\beta\eta} \lambda x.\overline{Cons_i^A}x =_{\beta\eta} \overline{GP}$$

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

$\lambda x.\overline{E}(\lambda z.\overline{P}x)x =_{\beta\eta}$	definition of $\overline{E}$
$\lambda x.((\lambda sy.y_i(Dsy))(\lambda z.\overline{P}x)x) =_{\beta\eta}$	β conversion 2.13
$\lambda x.x_i(\overline{D}(\lambda z.\overline{P}x)x) =_{\beta\eta}$	β conversion 2.13
$\lambda x.x_i((\lambda y.\overline{D}(\lambda z.\overline{P}y)y)x) =_{\beta\eta}$	inductive assumption for $\lambda y.\overline{D}(\lambda z.\overline{P}y)y$
$\lambda x.x_i(\overline{DP}x) =_{\beta\eta}$	definition of $\overline{E}$
$\overline{EP}$	

Lemma 3.13 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. Let $\overline{F}$ be a closed term of type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$ such that for every closed term $\overline{T}$ of type $\tau^{\overline{B}}$ terms $G_i = \lambda Z.\overline{F}_i ZT$ of types $(\tau^C)^{\alpha_i} \to \tau^C$ for all $i \leq a$ represents nonterminal trees in algebra C. Let $\overline{H}$ be a closed term of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ defined by $\lambda STx.S(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))Tx)$ (see lemma 3.9). For every closed term $\overline{T}$ of type $\tau^{\overline{B}}$ and every closed term $\overline{Y}$ of type $\times_{i=1}^{a}(\tau^A)^{\alpha_i}$, the terms $(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T}$ and $(\overline{F} \diamond ((\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i}))\overline{T})\overline{T}$ are $\beta\eta$ convertible.

Proof. In lemmas 3.7 and 3.8 we checked that terms $(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T}$ and $(\overline{F} \diamond ((\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i}))\overline{T})\overline{T}$ have the same type $(\tau^A)^a$. Let $\overline{T}$ and $\overline{Y}$ be closed terms of appropriate types. By lemma 3.8 the term $\lambda STx.S(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))Tx)$ is well typed in $\tau^A \to \tau^{\overline{B}} \to \tau^A$.

$(\overline{H}^{a} \diamond (\overline{Cons^{A}} \diamond \overline{Y}))\overline{T} =_{\beta\eta}$	definition 3.3 of $\overline{Cons^{A}}$
$((\times_{i=1}^{a}\overline{H}) \diamond ((\times_{i=1}^{a}\overline{Cons_i^{A}}) \diamond (\times_{i=1}^{a}\overline{Y_i})))\overline{T} =_{\beta\eta}$	formula 2.9
$((\times_{i=1}^{a}\overline{H}) \diamond ((\times_{i=1}^{a}\overline{Cons_i^{A}}\overline{Y_i}))\overline{T} =_{\beta\eta}$	formula 2.9
$((\times_{i=1}^{a}\overline{H}(\overline{Cons_i^{A}}\overline{Y_i}))\overline{T} =_{\beta\eta}$	formula 2.6
$\times_{i=1}^{a}(\overline{H}(\overline{Cons_i^{A}}\overline{Y_i})\overline{T}) =_{\beta\eta}$	definition of $\overline{H}$ and β conversion 2.13
$\times_{i=1}^{a}(\lambda x.[(\overline{Cons_i^{A}}\overline{Y_i})(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x)]) =_{\beta\eta}$	definition of $\overline{Cons_i^{A}}$, and β conversion 2.13
$\times_{i=1}^{a}(\lambda x.[(\lambda y.y_i(\overline{Y_i}y))(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x)]) =_{\beta\eta}$	β conversion 2.13
$\times_{i=1}^{a}(\lambda x.[((\lambda R \bullet (\overline{F} \diamond (\lambda z.\overline{R}))\overline{T}x)_i(\overline{Y_i}(\lambda R \bullet (\overline{F} \diamond (\lambda z.\overline{R}))\overline{T}x)))]) =_{\beta\eta}$	formulas 2.20 and 2.17
$\times_{i=1}^{a}(\lambda x.[((\lambda R_i.(\overline{F} \diamond (\lambda z.R))_i\overline{T}x)(\overline{Y_i}(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x)))]) =_{\beta\eta}$	formulas 2.19 and 2.18
$\times_{i=1}^{a}(\lambda x.[((\lambda R_i.(\overline{F}_i(\lambda z.R_i))\overline{T}x)(\overline{Y_i}(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x)))]) =_{\beta\eta}$	definition of $\overline{G}_i$

$\times_{i=1}^{a}(\lambda x.[((\lambda R_i.\overline{G}_i(\lambda z.R_i)x)(\overline{Y_i}(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x)))]) =_{\beta\eta}$	β conversion 2.13
$\times_{i=1}^{a}(\lambda x.[(\overline{G}_i(\lambda z.(\overline{Y_i}(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))\overline{T}x))x]) =_{\beta\eta}$	definition of $\overline{H}$
$\times_{i=1}^{a}(\lambda x.[(\overline{G}_i(\lambda z.((\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T}x)x)]) =_{\beta\eta}$	lemma 3.12 with a term P equal $(\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T}$ when G_i is nonterminal tree
$\times_{i=1}^{a}(\overline{G}_i((\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T})) =_{\beta\eta}$	definition of $\overline{G}_i$
$\times_{i=1}^{a}(\overline{F}_i((\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T})\overline{T})$	formula 2.9
$\times_{i=1}^{a}(\overline{F}_i) \diamond (\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T})\overline{T} =_{\beta\eta}$	definition of $\overline{F}$
$\overline{F} \diamond (\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i})\overline{T})\overline{T}$	

Lemma 3.14 Let $A = [\alpha_1, ..., \alpha_a]$ be an algebra. Let $\overline{B}$ be a product of algebras represented by type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. Let $\overline{I}$ be a closed term of type $\tau^{\overline{B}} \to (\times_{j=1}^{c}\tau_j^{C} \to \times_{i=1}^{a}\tau_i^{A})$. Let $\overline{F}$ be a closed term of type $\times_{i=1}^{a}((\tau^{C})^{\alpha_i} \to \tau^{\overline{B}} \to \tau^{C})$ defined by $\overline{F} = \lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$. For every closed term $\overline{T}$ a closed term $\overline{G}$ defined as $\lambda X \bullet (\overline{F} \diamond X)\overline{T}$ of type $\times_{i=1}^{a}((\tau^{C})^{\alpha_i} \to \tau^{C})$ 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

$\overline{G_r}$ represents a nonterminal tree the algebra C for every $r \leq a$. Let $\overline{T}$ be a closed term. The term $\overline{G_r}$ is $\lambda X_r.\overline{F}_r X_r \overline{T} =_{\beta\eta} \lambda X_r x.\overline{I}_r \overline{T} x(X_r x)$. $\overline{I}_r\overline{T}$ is a simple closed term of type $\times_{j=1}^{c}\tau_j^C \rightarrow \tau_r^A = \times_{j=1}^{c}\tau_j^C \rightarrow (O^{\alpha_r} \rightarrow O)$. Since $rank(\times_{j=1}^{c}\tau_j^C \rightarrow (O^{\alpha_r} \rightarrow O)) \leq 2$ 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 $q \in O^{\alpha_r}$, $x \in \times_{j=1}^{c}\tau_j^C$ and let K be a variable.

$$\begin{array}{ll}
K \Longrightarrow \lambda xq.q_1 & \\
\vdots & \\
K \Longrightarrow \lambda xq.q_{\alpha_r} & \\
K \Longrightarrow \lambda xq.x_j & \text{when } \gamma_j = 0 \\
K \Longrightarrow \lambda xq.x_j \underbrace{(Kxq)...(Kxq)}_{\gamma_j\ times} & \text{when } \gamma_j > 0
\end{array}$$

This grammar produces all closed terms of the type $\times_{j=1}^{c}\tau_j^C \rightarrow \tau_r^A$. The proof is by induction on the grammar construction of the term $K = \overline{I}_r\overline{T}$. Case 1. If $\overline{I}_r\overline{T}$ is $\lambda xq.q_p$ for $p \leq \alpha_r$ then $\overline{G_r} = \lambda X_r x.(X_r)_p x =_{\beta\eta} \lambda X_r.(X_r)_p$. Therefore $\overline{G_r}$ represents a projection. Case 2. If $\overline{I}_r\overline{T}$ is $\lambda xq.x_j$ for $\gamma_j = 0$ then $\overline{G_r} = \lambda X_r x.x_j =_{\beta\eta} \lambda X_r.\overline{Cons_j^C}$ therefore $\overline{G_r}$ represents the constant function. Case 3 inductive step. Suppose the theorem is true for closed terms $\overline{K_1}, ..., \overline{K_{\gamma_j}}$ which means that terms $\overline{G_1} = \lambda X_r x.\overline{K}_1 x(X_r x) ... \overline{G_{\gamma_j}} = \lambda X_r x.\overline{K_{\gamma_j}} x(X_r x)$ represent nonterminal trees. Let $\overline{G} = (\overline{G_1}, ..., \overline{G_{\gamma_i}})$ and $\overline{K} = (\overline{K_1}, ..., \overline{K_{\gamma_i}})$. So we have $G = \lambda X_r x.Kx(X_r x)$. Let us check that the theorem also holds for $K' = \lambda xq.x_i(Kxq)$. Let G' be $\lambda X_r x.K'x(X_r x)$

$G' = \lambda X_r x.K'x(X_r x) =_{\beta\eta}$	definition of K'
$\lambda X_r x.x_i(Kx(X_r x)) =_{\beta\eta}$	definitions of G and 3.3 of $\overline{Cons_i^C}$,
$\lambda X_r x.\overline{Cons_i^C}(GX_r)x$	η conversion 2.15
$\lambda X_r.\overline{Cons_i^C}(GX_r)$	

Lemma 3.15 Let $\overline{B}$ be a product of algebras $B_1 \times ... \times B_k$ such that a algebra B_i is based on the signature $S_{B_i} = [\beta_1^i, ...\beta_{b_i}^i]$. Let the product $\overline{B}$ is represented by the type $\tau^{\overline{B}}$. Let $C = [\gamma_1, ..., \gamma_c]$ be an algebra. Let T be a variable of the type $\tau^{\overline{B}}$ and x be a variable of the type $\times_{j=1}^{c}\tau_j^C$ Every long-normal closed term $\overline{P}$ of the type $\tau^{\overline{B}} \rightarrow \tau^C$ is in one of three possible forms

1. $P = \lambda Tx.x_i$ if $\gamma_i = 0$
2. $P = \lambda Tx.x_i(\overline{J}Tx)$ if $\gamma_i > 0$ for some closed term $\overline{J}$ of the type $\tau^{\overline{B}} \rightarrow (\tau^C)^{\gamma_i}$
3. $P = \lambda Tx.T_i(\overline{I}Tx)$ for some closed term $\overline{I}$ of type $\tau^{\overline{B}} \rightarrow \times_{j=1}^{c}\tau_j^C \rightarrow \times_{j=1}^{b_i}\tau_j^{B_i}$

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

4. Main Result

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

1. Term $\overline{H}$ represents h
2. $(\overline{H}^a \diamond (\overline{Cons^A} \diamond \overline{Y}))\overline{T} =_{\beta\eta} (\overline{F} \diamond ((\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i}))\overline{T})\overline{T}$

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

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

Theorem 4.2 (soundness) If f is a λ function then f is λ definable.

Proof. By induction on the construction of λ functions. Trivially all projections and constant functions are represented and representability is preserved by composition. Let A be an algebra based on signature $[\alpha_1, ..., \alpha_a]$. Always $i-th$ constructor in A is represented by $\overline{Cons_i^A}$ (see definition 3.3). We want to show that representability is also preserved by primitive recursion. Let $\overline{B}$ be a product of algebras. Let C be an algebra. Let $f_1, ..., f_a$ are λ functions such that $f_i : C^{\alpha_i} \times \overline{B} \to C$ for all $i \leq a$. Let us assume that for every $b \in \overline{B}$ functions $(f_1/\overline{b}) : C^{\alpha_1} \to C$,..., $(f_a/\overline{b}) : C^{\alpha_a} \to C$ are nonterminal trees. Let $h : A \times \overline{B} \to C$ 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 $\overline{F}$ of type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$. For every $b \in \overline{B}$ 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 $G_i^b = \lambda X_i.FX_i\underline{b}$ for $i \leq a$. Therefore the tuple term $G^b = (G_1^b, ..., G_a^b)$ is given by $\lambda X \bullet FX\underline{b}$. 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 κ^C. Let $\overline{H}$ be a closed term of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ defined by $\lambda STx.S(\lambda R \bullet (\overline{F} \diamond (\lambda z.R))Tx)$ (see lemma 3.9). By lemma 3.13 it holds that for every $\overline{T}$ and $\overline{Y}$, $(\overline{H}^a \diamond (\overline{C} \diamond \overline{Y}))\overline{T} =_{\beta\eta} (\overline{F} \diamond (\times_{i=1}^{a}(\overline{H}^{\alpha_i} \diamond \overline{Y_i}))\overline{T})\overline{T}$. According to lemma 4.1 $\overline{H}$ represents h. It means that the function h is λ definable.

Definition 4.3 (Measures of complexity) Let us introduce a complexity measure π 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 $T = \lambda x_1...x_n.x_iT_1...T_k$ then $\pi(T) = \max_{j=1...k}(\pi(\lambda x_1...x_n.T_j)) + 1$. In fact π corresponds with the height of Böhm tress for a term T. Let us introduce also a special measure of complexity ρ which apply only to closed terms of type $\tau^{\overline{B}} \to \tau^C$ for any product of algebras $\overline{B}$ and for any algebra $C = [\gamma_1, ..., \gamma_c]$. Let A be the $i-th$ algebra in this product $\overline{B}$, and let A be based on signature $[\alpha_1, ..., \alpha_a]$. Let $\lambda Tx.X$ be a closed term in the long normal form of $\tau^{\overline{B}} \to \tau^C$ type where $T = [T_1, ..., T_k]$ is a variable of type $\tau^{\overline{B}}$, $x = [x_1, ..., x_c]$ is a variable of type $\times_{i=1}^{c} \tau_i^C$ and X is a term of type O. By $\rho(\lambda Tx.X)$ 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 $T_j x$. A formal definition is the following:

$\rho(\lambda Tx.x_i) = 0$	for all $i \leq c$ such that $\gamma_i = 0$
$\rho(\lambda Tx.x_i(\overline{J}Tx)) = \sum_{j=1}^{\gamma_i} \rho(\overline{J_j})$	for $\gamma_i > 0$ where $\overline{J_i}$ for $j \leq \gamma_i$ are closed terms of the type $\tau^{\overline{B}} \rightarrow \tau^C$
$\rho(\lambda Tx.T_i(\overline{I}Tx)) = 0$	if $\overline{I}Tx =_{\beta\eta} x$
$\rho(\lambda Tx.T_i(\overline{I}Tx)) = 1 + \sum_{j=1}^{n} \rho(\overline{I_j})$	if $\overline{I}Tx \neq_{\beta\eta} x$ where $I_1, ..., I_a$ are closed terms of types $\tau^{\overline{B}} \rightarrow \tau^{C^{+\alpha_1}}$, ..., $\tau^{\overline{B}} \rightarrow \tau^{C^{+\alpha_n}}$ respectively (see definition 1.1)

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 $\overline{P}$ of type $\tau^{\overline{B}} \rightarrow \tau^C$ where $\overline{B}$ 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 P' "simpler" than 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 $B_1, ..., B_n$. 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 $(\alpha_1, ..., \alpha_a)$. Let $\overline{I}$ be a closed term of type $\tau^{\overline{B}} \rightarrow (\times_{j=1}^{c} \tau_j^C \rightarrow \times_{i=1}^{a} \tau_i^A)$. Let $\overline{F}$ be a closed term of the type $\times_{i=1}^{a}((\tau^C)^{\alpha_i} \rightarrow \tau^B \rightarrow \tau^C)$ defined by $\overline{F} = \lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$. Let $\overline{P}$ be a closed term of type $\tau^{\overline{B}} \rightarrow \tau^C$ defined by $\overline{P} = \lambda Tx.T_i(\overline{I}Tx)$. If $\overline{I}Tx \neq_{\beta\eta} x$ then $\rho(\overline{F_i}) < \rho(\overline{P})$ for all $i \leq$n.

Proof. First of all we can notice that terms $\overline{I}Tx$ and x may be $\beta\eta$ convertible only if algebras A and C are the same. The term $\overline{F_i}$ has the form $\lambda Y_i Tx.\overline{I_i}Tx(Y_i x)$. It is obvious that the number of occurrences of $T's$ in the $\overline{F_i}$ is less by one than the number of $T's$ in $\lambda Tx.T_i(\overline{I}Tx)$. New variables $(Y_i)_j$ for

$j \leq \gamma_i$ of type τ^C in the term $\overline{F_i}$ are in the contexts of $Y_i x$ and therefore their occurrences do not increase the measure ρ.

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

Proof. Let $\overline{P}$ be a closed term of type $\tau^{\overline{B}} \to \tau^C$ 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, $\lambda Tx.x_i$ when $\gamma_i = 0$ or $\lambda Tx.T_i(\overline{I}Tx)$ if $\overline{I}Tx =_{\beta\eta} x$. (see lemma 3.15 and definition of ρ). If $\overline{P} = \lambda Tx.x_i$ then $\overline{P}$ represents function p:$\overline{B} \to C$ which maps onto $i-th$ constructor of the algebra C constantly. Therefore p is a λ function. If $\overline{P} = \lambda Tx.T_i(\overline{I}Tx)$ where $\overline{I}Tx =_{\beta\eta} x$ then it means that the algebra C and $i-th$ algebra in the product $\overline{B}$ are the same and moreover $\overline{P} =_{\beta\eta} \lambda Tx.T_i x =_{\beta\eta} \lambda T.T_i$. In this case $\overline{P}$ represents projection $p(b_1, ...b_k) = b_i$ so it is a λ function.
Induction step. Suppose the theorem is true for all closed terms $\overline{P'}$ of any of the types $\tau^{\overline{B'}} \to \tau^{C'}$ 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 ρ and lemma 3.15 $\overline{P}$ must be in one of two possible forms: $\overline{P} = \lambda Tx.x_i(\overline{J}Tx)$ for some closed term $\overline{J}$ or $\overline{P} = \lambda Tx.T_i(\overline{I}Tx)$ for some closed term $\overline{I}$ such that $\overline{I}Tx \neq_{\beta\eta} x$.
(Case 1) If $\overline{P} = \lambda Tx.x_i(\overline{J}Tx)$ for some closed term $\overline{J} = (\overline{J_1}, ..., \overline{J_{\gamma_i}})$ then $\pi(\overline{J_r}) < \pi(\overline{P})$ for all $r \leq \gamma_i$ and $\rho(\overline{J_r}) \leq \rho(\overline{P})$ for all $r \leq \gamma_i$. All closed terms $\overline{J_1}, ..., \overline{J_{\gamma_i}}$ are of the type $\tau^{\overline{B}} \to \tau^C$ (see lemma 3.6). Therefore according to induction assumption $\overline{J}_r$ represents function $j_r : \overline{B} \to C$ which are λ functions. We have that $p(b_1, ..., b_k) = c_i(j_1(b_1, ..., b_k), ..., j_{\gamma_i}(b_1, ..., b_k))$ where c_i is the $i-th$ constructor of the algebra C of arity γ_i. The function p as the composition of λ functions is a λ function.
(Case 2) Let $\overline{P} = \lambda Tx.T_i(\overline{I}Tx)$ for some closed term $\overline{I}$ of the type $\tau^{\overline{B}} \to \times_{j=1}^{c}\tau_j^C \to \times_{j=1}^{a}\tau_j^A$ where A is the $i-th$ algebra in the product $\overline{B}$ given by signature $[\alpha_1, ..., \alpha_a]$. Let $\overline{I}Tx \neq_{\beta\eta} x$. Let $\overline{F}$ be a closed term of type $\times_{j=1}^{a}((\tau^C)^{\alpha_i} \to \tau^{\overline{B}} \to \tau^C)$ defined by $\overline{F} = \lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$. By lemma 4.5 $\rho(\overline{F_j}) < \rho(\overline{P})$ for all $j \leq a$. Since any term F_j is "simpler" than the term P then all F_j are representing λ functions. Let $f_1, ..., f_a$ be the system of λ functions represented by $\overline{F}$ where $f_j : C^{\alpha_j} \times \overline{B} \to C$. By lemma 3.14 it holds that for every $\overline{T}$, closed term $\overline{G}$ defined by $\lambda X \bullet (\overline{F} \diamond X)\overline{T}$ belongs to κ^C. Since $\lambda X \bullet (\overline{F} \diamond X)\overline{T}$ belongs to κ^C then according to lemma 3.11 all functions f_j/b for $j \leq a$ are nonterminal trees in C. Let $h : A \to \overline{B} \to C$ (where A is an B_i) be a function defined by limited primitive recursion from the system $f_1, ..., f_a$. Let $\overline{H}$ be a closed term of type $\tau^A \to \tau^{\overline{B}} \to \tau^C$ defined by $\lambda STx.S(\overline{I}Tx)$. By lemma 3.10 we know that for every $\overline{T}$ and $\overline{Y}$, $(\overline{H}^a \diamond (\overline{C} \diamond \overline{Y}))\overline{T} =_{\beta\eta} (\overline{F} \diamond ((\times_{j=1}^{a}(\overline{H}^{\alpha_j} \diamond \overline{Y_j}))\overline{T})\overline{T}$. Thanks to lemma 4.1 it means that $\overline{H}$ represents h. Therefore h is a λ function. Function h is represented by $\lambda STx.S(\overline{I}Tx)$ but function p is represented by $\lambda Tx.T_i(\overline{I}Tx)$. Therefore the following relation between functions h and p holds: $p(b_1, ..., b_k) = h(b_i, b_1, ..., b_k)$ for all expressions $b_1, ..., b_k$. Since the class of λ functions is closed for compositions it holds that p is a λ function.

5. Examples of Applications

The characteristic obtained in theorem 4.6 may be used in order to find the classes of λ definable functions for some specific free algebras. Let us consider λ definability between algebras of Church's numerals N and algebra of binary words Σ^* for some binary alphabet Σ (compare [Mad91]). N is based on signature $[1,0]$ and Σ on $[1,1,0]$. The λ definability for operations in $(\Sigma^*)^p \rightarrow \Sigma^*$ is given in [Zai87]. λ definable word operations are those obtained by composition from $append : (\Sigma^*)^2 \rightarrow \Sigma^*$, $substitution : (\Sigma^*)^3 \rightarrow \Sigma^*$ and if_then_else : $(\Sigma^*)^3 \rightarrow \Sigma^*$ (see [Zai87] for details). As a simple consequence of theorems 4. 6 and 4.2 for λ definable mappings in $N \rightarrow \Sigma^*$ 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 $f(n) = \epsilon$ (the empty word) $f(n) = a$ and $f(n) = b$ are λ definable. Function $f(n) = a^n$ is λ definable. If functions $f, g, h : N \rightarrow \Sigma^*$ are λ definable then $n\rightarrow$ *append(f(n),g(n))*, $n\rightarrow$ *substitution(f(n),g(n),h(n))* and $n\rightarrow$ *if f(n)=ϵ then g(n) else h(n)* are also λ definable. The functions defined above are the only λ definable functions in $N \rightarrow \Sigma^*$.

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 $\tau^N = (0 \rightarrow 0) \rightarrow (0 \rightarrow 0)$. Any number n is represented by term (Church's numerals) $\lambda sx.s(...sx)$. The algebra Σ of binary words based on the signature $[1,1,0]$ is represented by the type $\tau^\Sigma = (0 \rightarrow 0) \rightarrow ((0 \rightarrow 0) \rightarrow (0 \rightarrow 0))$. Let Σ be a binary alphabet $\{a, b\}$. Λ denotes the empty word in Σ^* We are going to prove that the term $\overline{P} = \lambda wsx.ws(\lambda y.y)x$ of the type $\tau^\Sigma \rightarrow \tau^N$ represents the function computing the number of letters a in the given word. Term $\overline{P}$ is of the form $\lambda wsx.w(\overline{I}wsx)$ where $\overline{I}$ is the term of type $\tau^\Sigma \rightarrow \times_{i=1}^2 \tau_i^N \rightarrow \times_{i=1}^3 \tau_i^\Sigma$ such that $\overline{I} = (\overline{I_1}, \overline{I_2}, \overline{I_3})$ where the term $\overline{I_1} = \lambda wsxy.sy$ the term $\overline{I_2} = \lambda wsxy.y$ and the term $\overline{I_3} = \lambda wsx.x$. Let us define the term $\overline{F} = (\overline{F_1}, \overline{F_2}, \overline{F_3})$ by formula $\overline{F} = \lambda Y \bullet (\lambda Tx.((\overline{I}Tx) \diamond (Yx)))$. Therefore $\overline{F_1} = \lambda Ywsx.s(Ysx)$, $\overline{F_2} = \lambda Ywsx.Ysx$ and $\overline{F_3} = \lambda wsx.x$. Term $\overline{F} = (\overline{F_1}, \overline{F_2}, \overline{F_3})$ 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 $\overline{H} = \lambda STsx.S(\lambda y.sy)(\lambda y.y)x$ represents the function $h : (\Sigma)^2 \rightarrow N$ defined from f_1, f_2, f_3 by recursion:

$$\begin{aligned} h(av, w) &= 1 + h(v, w) \\ h(bv, w) &= h(v, w) \\ h(\Lambda, w) &= 0 \end{aligned}$$

Let $p : \Sigma \rightarrow N$ be a function represented by the term $\overline{P}$. We have $p(w) = h(w, w)$

References.

[Bar81] H. O. Barendregt, *The Lambda Calculus*, Studies in Logic and the Foundations of Mathematics (North- Holland, Amsterdam, 1981).

[BöB85] Corrado Böhm and Allessandro Berarducci, *Automatic synthesis of typed λ programs on term algebras*, Theoretical Computer Science 39 (1985) 135-154

[Chu40] A. Church *A formulation of the simple theory of types*, Journal of Symbolic Logic 5(1) 56-68 (1940)

[Fri75] H. Friedman, *Equality between functionals*, in: Lecture Notes in Mathematics 453 (Springer, Berlin, 1975) pp 22-37.

[Lei89] Daniel Leivant *Subrecursion and lambda representation over free algebras*, in S. Buss and P Scott (eds.), Feasible Mathematics (Proceedings of June 1988 Workshop at Cornell)

[Mad91] Madry M, *On the λ definable functions between numbers, words and trees* Fundamenta Informaticae, 1991

[Sch75] Schwichtenberg H., *Definierbare Funktionen im λ -Kalkül mit Typen*, Arch Math. Logik Grundlagenforsch 17 (1975-76) pp 113-114.

[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 λ calculus,* Theoretical Computer Science 52 (1987) pp. 1-14

[Zai90] Zaionc M. *A Characteristic of λ definable Tree Operations*, Information and Computation 89 No.1, (1990) 35-46

[Zai91] Zaionc M. *λ definability on free algebras,* Annals of Pure and Applied Logic 51 (1991) pp 279 -300.