Theoretical Computer Science 52 (1987) 1-14
North-Holland

WORD OPERATION DEFINABLE IN THE TYPED λ-CALCULUS

Marek ZAIONC *
Department of Computer and Information Sciences, University of Alabama at Birmingham, University Station, Birmingham, AL 35294, U.S.A.

Communicated by G. Mirkowska
Received March 1985
Revised February 1987

Abstract. A λ-language over a simple type structure is considered. Type $B=(O\to O)\to((O\to O)\to(O\to O))$ is called a binary word type because of the isomorphism between words over a binary alphabet and closed terms of this type. Therefore, any term of type $B\to(B\to\cdots\to(B\to B)\cdots)$ represents an n-ary word function. The problem is: what class of word functions are represented by the closed terms of the examined type. It is proved that there exists a finite base of word functions such that any λ-definable word function is some composition of functions from the base. The algorithm which, for every closed term, returns the function in the form of a composition of basic operations is given. The main result is proved for a binary alphabet only, but can be easily extended to any finite alphabet. This result is a natural extension of the Schwichtenberg theorem (see Schwichtenberg (1975) and Statman (1979)) which solves the same problem for the natural number type $N=(O\to O)\to(O\to O)$.

Notations

We denote by $\mathbb{N}$ the set of nonnegative integers. The interval $\{1,2,\ldots,n\}$ is denoted by $[n]$ and the interval $\{0,1,\ldots,n\}$ by $\overline{[n]}$. Σ is a binary alphabet $\{a,b\}$. By Σ^* we mean the set of all words over Σ. The empty word is denoted by Λ. We will use the following notation: if $n\in\mathbb{N}$, then by (n) we mean the word $a\ldots a$ with n occurrences of the letter a (with $(0)=\Lambda$). By $c(n,w)$ we denote the n-ary function $(\Sigma^*)^n\to\Sigma^*$ which maps onto the word w constantly. If w is a word, then $\lfloor w\rfloor$ is the number of letters in w (with $\lfloor\Lambda\rfloor=0$). We have $\lfloor(n)\rfloor=n$ for any number $n\in\mathbb{N}$.

1. Typed λ-calculus

Our language is based on the Church's [13] simple theory of types. The set of types is introduced as follows; O is a type and if τ and μ are types, then $\tau\to\mu$ is a type. We will use the following notation; if $\tau_1,\ldots,\tau_n$, τ are types, then by

* The research described in this paper was done while author was at Jagiellonian University, Krakow (Poland).

$\tau_1, \ldots, \tau_n \to \tau$ we understand the type $\tau_1 \to (\tau_2 \to \cdots \to (\tau_n \to \tau) \cdots)$. By $\tau^n \to \mu$ we mean the type $\tau, \ldots, \tau \to \mu$ with n occurrences of τ (with $\tau^0 \to \mu = \mu$). Therefore, every type τ has a form $\tau_1, \ldots, \tau_n \to O$. The type τ_i is called the component of τ and is denoted by $\tau[i]$. By $\tau[i_1, \ldots, i_k]$ we mean $\tau[i_1] \ldots [i_k]$. For any type τ we define numbers $\arg(\tau)$ and $\mathrm{rank}(\tau)$ as follows: $\arg(O) = \mathrm{rank}(O) = 0$ and if $\tau = \tau[1], \ldots, \tau[n] \to O$, then $\arg(\tau) = n$ and $\mathrm{rank}(\tau) = \max_{i=1,\ldots,n} (\mathrm{rank}(\tau[i])) + 1$. For any type τ a denumerable set of variables $V(\tau)$ is given. A set of terms is a minimal set containing variables and which is closed for application and abstraction rules; i.e., if T is a term of type $\tau \to \mu$ and S is a term of type τ, then TS is a term of type μ; and if x is variable of type τ and T is a term of type μ, then $\lambda x . T$ is a term of type $\tau \to \mu$. If T is a term of type τ, we write $T \in \tau$. We shall use the notation $\lambda x_1 x_2 \ldots x_n . T$ for term $\lambda x_1 . (\lambda x_2 \ldots (\lambda x_n . T) \ldots)$ and $TS_1 S_2 \ldots S_n$ for $(\ldots((TS_1)S_2) \ldots S_n)$. If T is a term and x is a variable of the same type as a term S, then $T[x/S]$ denotes the term obtained by substitution of the term S for each free occurrence of x in T.

The axioms of equality between terms have the form of α, β, and η conversions (see [1, 3]) and the convertible terms will be written as $T =_{\beta\eta} S$. All terms are considered modulo α, β, and η conversions. By $\mathrm{Cl}(\tau)$ we mean the set of all closed (without free variables) terms of type τ. If Y is a set of variables, then $\mathrm{Cl}(\tau, Y)$ is the set of all terms of type τ with only free variables from Y. Obviously, $\mathrm{Cl}(\tau, \emptyset) = \mathrm{Cl}(\tau)$ and $\mathrm{Cl}(\tau, \emptyset) \subseteq \mathrm{Cl}(\tau, Y)$. Term T is in long normal form iff $T = \lambda x_1 \ldots x_n . y T_1 \ldots T_k$, where y is an x_i for $i \in [n]$ or y is a free variable, and T_j, for each $j \in [k]$, is in long normal form and $y T_1 \ldots T_k$ is a term of type O. It is easy to prove that long normal forms exist and are unique for $\beta\eta$ conversions (compare [10] or Φ-normal form in [2]). Let us introduce a complexity measure π for closed terms. If T is a closed term written in normal form and $T = \lambda x_1 \ldots x_n . x_i$, then $\pi(T) = 0$. If $T = \lambda x_1 \ldots x_n . x_i T_1 \ldots T_k$, then $\pi(T) = \max_{j=1,\ldots,k}(\pi(\lambda x_1 \ldots x_n . T_j)) + 1$. For a closed term S, $\pi(S)$ is defined as $\pi(T)$ for T in long normal form such that $S =_{\beta\eta} T$.

2. Term grammars

Let NT be a finite or denumerable set of variables (the elements of NT correspond to nonterminal elements in the classical grammars). A production is a pair (y, T) also denoted by $y \Rightarrow T$, where variable $y \in \mathrm{NT}$, y and T have the common type τ, and $T \in \mathrm{Cl}(\tau, \mathrm{NT})$. A grammar is a finite or denumerable set of productions. The relation of the indirect production $\to$ in the grammar G is defined by induction as follows:

if $y \Rightarrow T \in G$, then $y \to T$ holds; if $y \to T$ and $z \to S$ hold, then $y \to R$,

where R is any term obtained from T by substitution of at most one free occurrence of z by S. By $L(G, y)$ we mean the set of all closed terms which are generated from

y by grammar G; i.e., if $y \in \tau$, then $L(G, y) = \{T \in \mathrm{Cl}(\tau) \mid y \to T\}$. It is easy to notice that if $y \to T$ and $z \to S$ hold, then $y \to T[z/S]$ holds. Let us assume that $y \Rightarrow T$ is a production and $y_1, \ldots, y_n$ are all free occurrences of nonterminal variables in the term T. Let $y \in \tau, y_1 \in \tau_1, \ldots, y_n \in \tau_n$. We say that this production determines a function $\alpha : \mathrm{Cl}(\tau_1) \times \mathrm{Cl}(\tau_2) \times \cdots \times \mathrm{Cl}(\tau_n) \to \mathrm{Cl}(\tau)$ defined by

$$\alpha(T_1, \ldots, T_n) = T[y_1/T_1, \ldots, y_n/T_n] \quad \text{for every closed term } T_1 \in \mathrm{Cl}(\tau_1), \ldots, T_n \in \mathrm{Cl}(\tau_n).$$

If there are no nonterminal variables in the term T, then the production $y \Rightarrow T$ determines a 0-ary function (constant) T which belongs to $\mathrm{Cl}(\tau)$.

We will use the lower case Greek letters α, β, γ, δ for the names of such functions. Let us define a grammar $G(\tau)$ for a given type τ. The construction of this grammar is analogous with the construction of the Huet matching-tree for the unification problem (see [4, Chapter 3.4, p. 37] with simplification for the $\beta\eta$ λ-calculus in Chapter 4.5, p. 51). Let y be a nonterminal variable of type τ. For the type O, grammar $G(O)$ is $y \Rightarrow y$. If $\tau = \tau[1], \ldots, \tau[n] \to 0$, then the grammar contains all productions which are of the form:

(i) $y \Rightarrow \lambda x_1 \ldots x_n . x_i$ if $\arg(\tau[i]) = 0$,

(ii) $y \Rightarrow \lambda x_1 \ldots x_n . x_i T_1 \ldots T_k$ if $\arg(\tau[i]) = k > 0$,

where $T_j \in \tau[i, j]$ for $j \leqslant k$ are as follows:

(ii1) $T_j = y x_1 \ldots x_n$ iff $\arg(\tau[i, j]) = 0$,

(ii2) $T_j = \lambda z_1 \ldots z_p . y' x_1 \ldots x_n z_1 \ldots z_p$ iff $\arg(\tau[i, j]) = p > 0$,

where y' is a new nonterminal variable of type $\tau[1], \ldots, \tau[n] \to \tau[i, j]$ and $z_s \in \tau[i, j, s]$ for $s \leqslant p$.

This construction is repeated for all new nonterminal variables introduced in this step.

Example 2.1. Let τ be a following type $((O, O \to O) \to O) \to (O \to O)$. Let, as considered, the following grammar be over $\mathrm{NT} = \{y\}$. Types of auxiliary variables are the following $p \in (O, O \to O) \to O$ and $x, v, z \in O$:

(1): $y \Rightarrow \lambda px.x$, (2): $y \Rightarrow \lambda px.p(\lambda vz.ypx)$,

(3): $y \Rightarrow \lambda px.p(\lambda vz.ypv)$, (4): $y \Rightarrow \lambda px.p(\lambda vz.ypz)$.

Let α, β, γ, δ be the functions determined by these productions respectively. The closed term of type τ

$$\lambda p x_1 . p(\lambda x_2 x_3 . p(\lambda x_4 x_5 . p(\lambda x_6 x_7 . x_4)))$$

can be obtained by means of productions (2), (3), (2), (1). This closed term can be presented as $\beta \circ \gamma \circ \beta \circ \alpha$. It is easy to prove that this grammar generates all closed terms of type τ (cf. [12]).

Theorem 2.2. *For every type τ the grammar $G(\tau)$ generates all closed terms of type τ.*

Proof. By induction on the complexity measure: Let T be a closed term of type τ. If T is a projection written in normal form, then T can be obtained by means of production (i). Let $T = \lambda x_1 \ldots x_n.x_i T_1 \ldots T_k$ where $\arg(\tau[i]) = k$ and $T_j \in \mathrm{Cl}(\tau[i,j], \{x_1, \ldots, x_n\})$ for $j \leqslant k$. Let S_j be the term $\lambda x_1 \ldots x_n.T_j$ for $j \leqslant k$. Every term S_j belongs to $\mathrm{Cl}(\tau[1], \ldots, \tau[n] \to \tau[i,j])$ for $j \leqslant k$. The complexity measure $\pi(S_j)$ is less than $\pi(T)$ for every $j \leqslant k$. So, from the inductive assumption, every term S_j can be obtained by this grammar from the following nonterminal variables:

if $\tau[i,j] = 0$, then $y \to S_j$ (case (ii1)),

if $\tau[i,j] = \tau[i,j,1], \ldots, \tau[i,j,p] \to \mu$, then $y' \to S_j$ (case (ii2)).

Let α be the function determined by production (ii). Therefore, the term T can be obtained by means of production (ii) from the terms $S_1, \ldots, S_k$ and the condition $\alpha(S_1, \ldots, S_k) = T$ holds. $\square$

Lemma 2.3. *For every type τ such that* $\mathrm{rank}(\tau) \leqslant 2$ *there is a finite grammar which generates all closed terms of type τ.*

Proof. Let $\tau = \tau[1], \tau[2], \ldots, \tau[n] \to 0$. We will prove that the grammar $G(\tau)$ is finite and according to Theorem 2.2 produces all closed term of type τ. The grammar contains all productions which are of the form:

$y \Rightarrow \lambda x_1 \ldots x_n.x_n$ if $\arg(\tau[i]) = 0$,

$y \Rightarrow \lambda x_1 \ldots x_n.x_i T_1 \ldots T_k$ if $\arg(\tau[i]) = k > 0$, where $T_j = y x_1 \ldots x_n$ for $j \leqslant k$.

Grammars described here can produce any free structure. $\square$

Theorem 2.4 (Zaionc [12]). *For every type τ such that* $\mathrm{rank}(\tau) \leqslant 3$ *and* $\arg(\tau[i]) \leqslant 1$ *for every $i \leqslant \arg(\tau)$, there is a finite grammar which generates all closed terms of this type (compare the notion of regular grammar introduced in* [12]).

An illustration of this case is presented in Example 2.1.

3. Finitely generated sets

For a given type τ the set $\mathrm{FUN}(\tau)$ is defined as $\bigcup_{i=1}^{\infty} \mathrm{Cl}(\tau^i \to \tau)$. Two terms R and T from the set $\mathrm{Cl}(\tau^i \to \tau)$ are strongly equivalent, $R \equiv T$, if, for every closed term $S_1, \ldots, S_i \in \mathrm{Cl}(\tau)$, $RS_1 \ldots S_i =_{\beta\eta} TS_1 \ldots S_i$. If F is a subset of $\mathrm{FUN}(\tau)$, then by $\mathrm{App}(F)$ we denote the set of all compositions of members of F; i.e., $\mathrm{App}(F)$ is

a minimal subset of FUN(τ) containing F, all projections $\lambda x_1 \dots x_n.x_i$ such that $x_j \in \tau$ for $j \in [n]$, and containing all constant functions from FUN(τ) of the form $\lambda x_1 \dots x_n.T$ where $x_i \in \tau$ and $T \in \mathrm{Cl}(\tau)$ and closed for compositions. That is, if $T \in \mathrm{App}(F)$ such that $T \in \mathrm{Cl}(\tau^n \to \tau)$ and if $S_1, \dots, S_n \in \mathrm{App}(F)$ such that $S_i \in \mathrm{Cl}(\tau^{k_i} \to \tau)$ for $i \in [n]$ and $k_i \in \mathbb{N}$, then term $\lambda x_1 \dots x_n.T(S_1 x_{j_{1,1}} \dots x_{j_{1,k_1}}) \dots (S_n x_{j_{n,1}} \dots x_{j_{n,k_n}})$ belongs to App(F) for every $j_{p,q} \in [n]$ such that $p \in [n]$ and $q \in [k_p]$.

The set FUN(τ) is finitely generated iff there is a finite set $F \subset \mathrm{FUN}(\tau)$ such that, for every $T \in \mathrm{FUN}(\tau)$, there is an $S \in \mathrm{App}(F)$ such that $T \equiv S$.

Theorem 3.1 (Schwichtenberg [8] and Statman [9]). *Set* FUN(N) *is finitely generated where* $N = (O \to O) \to (O \to O)$. *The set of closed terms of type* N *(Church's numerals) can be naturally interpreted as numbers; the set of generators consists of terms which represents addition, multiplication,* sq *and* $\overline{\mathrm{sq}}$ *(see* [5, p. 223]). *Therefore, the set* FUN(N) *represents the extended polynomials.*

Theorem 3.2 (cf. Statman [11, p. 24]). *Set* FUN(τ_n), *where* τ_n *is the type* $O^n \to O$, *is finitely generated for every* $n \in \mathbb{N}$.

Proof. From the functional completeness of n-valued propositional logic (cf., for example, [6] or [7]) and from the fact that there are exactly n closed terms of type τ_n, the above theorem easily follows; Let the number $i \in [n]$ be represented by the ith projection in the following way: $\underline{i} = \lambda x_1 \dots x_k.x_i$. The term $T \in \mathrm{Cl}(\tau_n^k \to \tau)$ represents the function $f:[n]^k \to [n]$ if, for all $n_1, \dots, n_i \in [n]$, $T\underline{n_i} \dots \underline{n_i} =_{\beta\eta} \underline{f(n_1, \dots, n_i)}$. By induction on k we will prove that every function $f:[n]^k \to [n]$ is represented.

For $k = 1$, the term $\lambda c x_1 \dots x_n.c x_{f(1)} \dots x_{f(n)}$ represents the function $f:[n] \to [n]$. Now, suppose any k-ary function is represented. Let f be a $(k+1)$-ary function. By $f_1, \dots f_n$ we denote the k-ary functions which are defined by

$$f_j(x_1, \dots, x_k) = f(x_1, \dots, x_k, j) \quad \text{for } j \in [n].$$

So there exist terms $T_1, \dots, T_n \in \mathrm{Cl}(\tau_n^k \to \tau)$ which represent $f_1, \dots, f_n$ respectively. Therefore, function f is represented by the term $\lambda c_1 \dots c_{k+1} x_1 \dots x_n.c_{k+1}(T_1 c_1 \dots c_k x_1 \dots x_n) \dots (T_n c_1 \dots c_k x_1 \dots x_n)$.

For every $n \geqslant 2$, n-valued propositional logic is functionally complete, i.e., there is a finite number of functions which generate any function. The set of representatives of those functions form the base for FUN(τ_n). For $n = 1$, the possible set of generators for FUN(τ_1) is $\lambda ux.x \in \mathrm{Cl}(\tau_1 \to \tau_1)$. For FUN($\tau_0$), the set of generators is empty. □

Example 3.3. FUN($O, O \to O$) is finitely generated. A possible set of generators is $\lambda pqxy.q(pyx)(pyy)$ and $\lambda pxy.pyx$ ($p, q \in (O, O \to O)$ and $x, y \in O$) if term $\lambda xy.x$ represent falsity and $\lambda.xy.y$ truth. The first term represents implication and the second negation in classical 2-valued propositional calculus.

4. Word functions

In this section the set of word functions $(\Sigma^*)^n \to \Sigma^*$ for $n \geqslant 1$ is investigated. Let us distinguish the following functions recursively definable in Manna manner [14]. The function $\text{app}: (\Sigma^*)^2 \to \Sigma$ is the usual concatenation inductively defined by

$$\text{app}(\Lambda, y) = y. \qquad \text{app}(a\,x, y) = a\,\text{app}(x, y), \qquad \text{app}(b\,x, y) = b\,\text{app}(x, y).$$

The function $\text{sub}: (\Sigma^*)^3 \to \Sigma$ is called 'substitution'. The word $\text{sub}(x, y, z)$ is obtained from x by substituting for all occurrences of the letter a the word y and for all occurrences of the letter b the word z. The definition is as follows:

$$\text{sub}(\Lambda, y, z) = \Lambda, \qquad \text{sub}(a\,x, y, z) = \text{app}(y, \text{sub}(x, y, z)),$$
$$\text{sub}(b\,x, y, z) = \text{app}(z, \text{sub}(x, y, z)).$$

The functions cut_a and cut_b extract maximal prefixes of the form $a \ldots a$ and $b \ldots b$ respectively and are defined by

$$\text{cut}_a(\Lambda) = \Lambda, \qquad \text{cut}_a(a\,x) = \text{app}(a, \text{cut}_a(x)), \qquad \text{cut}_a(b\,x) = \Lambda,$$
$$\text{cut}_b(\Lambda) = \Lambda, \qquad \text{cut}_b(a\,x) = \Lambda,$$
$$\text{cut}_b(b\,x) = \text{app}(b, \text{cut}_b(x)).$$

The functions $\text{sq}, \overline{\text{sq}}: \Sigma^* \to \Sigma^*$ are emptiness and nonemptiness tests:

$$\text{sq}(\Lambda) = (0), \quad \text{sq}(a\,x) = (1), \quad \text{sq}(b\,x) = (1),$$
$$\overline{\text{sq}}(\Lambda) = (1), \quad \overline{\text{sq}}(a\,x) = (0), \quad \overline{\text{sq}}(b\,x) = (0).$$

The functions $\text{occ}_a, \text{occ}_b: \Sigma^* \to \Sigma$ check if the letter a or b respectively occur in a given word x and are defined by

$$\text{occ}_a(\Lambda) = (0), \qquad \text{occ}_a(a\,x) = (1), \qquad \text{occ}_a(b\,x) = \text{occ}_a(x),$$
$$\text{occ}_b(\Lambda) = (0), \qquad \text{occ}_b(a\,x) = \text{occ}_b(x), \qquad \text{occ}_b(b\,x) = (1).$$

The functions $\text{beg}_a, \text{beg}_b: \Sigma^* \to \Sigma^*$ which check if a given word begins with a or b respectively can be defined as

$$\text{beg}_a = \text{sq} \circ \text{cut}_a, \qquad \text{beg}_b = \text{sq} \circ \text{cut}_b.$$

Let us now define the set λdef as a minimal set of word functions containing app, sub, cut_a, cut_b, sq, $\overline{\text{sq}}$, occ_a, occ_b, all projections, and all constant functions which are closed for compositions. By $\lambda\text{def}(n)$ we denote a subset of λdef which consists of all n-ary functions from λdef. The set TEST is a subset of $\lambda\text{def}(1)$ containing sq, $\overline{\text{sq}}$, beg_a, $\overline{\text{sq}} \circ \text{beg}_a$, beg_b, $\overline{\text{sq}} \circ \text{beg}_b$, occ_a, $\overline{\text{sq}} \circ \text{occ}_a$, occ_b, $\overline{\text{sq}} \circ \text{occ}_b$.

If $n, m \in \mathbb{N}$, then by $(n) \oplus (m)$ and $(n) \otimes (m)$ we understand the words $\text{app}((n), (m))$ and $\text{sub}((n), (m), \Lambda)$ respectively. Let us assume that $x_i \in \Sigma^*$ for $i \in \mathbb{N}$. The word $\Theta_{i=1}^{k} x_i$ is defined by induction as

$$\mathop{\Theta}_{i=1}^{1} x_i = x_1, \qquad \mathop{\Theta}_{i=1}^{k+1} x_i = \text{app}\left(x_{k+1}, \mathop{\Theta}_{i=1}^{k} x_i \right).$$

For all $n, m, n_1, \ldots, n_k \in \mathbb{N}$ the following equalities hold:

$$(n)\oplus(m)=(n+m), \qquad (n)\otimes(m)=(nm), \qquad \bigoplus_{i=1}^{k}(n_i)=\left(\sum_{i=1}^{k} n_i\right).$$

The class $P(n,k)$ for $n,k\geqslant 1$ of n-ary functions $(\Sigma^*)^n\to\{(0),(1),\ldots,(k-1)\}$ is defined as a minimal class which contains all constant functions $c(n,(j))$ for $j\in\overline{[k-1]}$ and is closed for the conditional choice rule; i.e., if $p, q\in P(n,k)$, $i\in[n]$ and $f\in\mathrm{TEST}$, then the function s defined below belongs to $P(n,k)$.

$$s(x_1,\ldots,x_n)=\begin{cases} p(x_1,\ldots,x_n) & \text{iff } f(x_i)=(1),\\ q(x_1,\ldots,x_n) & \text{iff } \overline{\mathrm{sq}}(f(x_i))=(1).\end{cases}$$

A function obtained by the conditional choice rule can also be defined as

$$s(x_1,\ldots x_n)=f(x_i)\otimes p(x_1,\ldots,x_n)\oplus\overline{\mathrm{sq}}(f(x_i))\otimes q(x_1,\ldots,x_n).$$

Lemma 4.1. $P(n,k)\subset\lambda\mathrm{def}(n)$.

Proof. Inductively for the construction of $p\in P(n,k)$. For a constant function p it is obvious. If s is obtained from p, q by the conditional choice rule, then

$$s(x_1,\ldots x_n)=f(x_i)\otimes p(x_1,\ldots,x_n)\oplus\overline{\mathrm{sq}}(f(x_i))\otimes q(x_1,\ldots,x_n).$$

Therefore, s is a composition of basic functions and $s\in\lambda\mathrm{def}(n)$. □

Lemma 4.2. *If $p, q, r\in P(n,k)$, then the function s defined below belongs to $P(n,k)$.*

$$s(x_1,\ldots,x_n)=\begin{cases} p(x_1,\ldots,x_n) & \textit{iff } \mathrm{sq}(r(x_1,\ldots,x_n))=(1),\\ q(x_1,\ldots,x_n) & \textit{iff } \overline{\mathrm{sq}}(r(x_1,\ldots,x_n))=(1).\end{cases}$$

Proof. By induction on the construction of the function r. If r is a constant function, then $s=p$ or $s=q$. Let us assume that the function r is constructed by means of r_1, r_2 from $P(n,k)$ and f from TEST. Therefore, r has a form

$$r(x_1,\ldots x_n)=f(x_i)\otimes r_1(x_1,\ldots,x_n)\oplus\overline{\mathrm{sq}}(f(x_i))\otimes r_2(x_1,\ldots,x_n).$$

Because of the inductive assumption we know that the functions s_1, s_2 defined by

$$s_1(x_1,\ldots,x_n)=\begin{cases} p(x_1,\ldots,x_n) & \text{iff } \mathrm{sq}(r_1(x_1,\ldots,x_n))=(1),\\ q(x_1,\ldots,x_n) & \text{iff } \overline{\mathrm{sq}}(r_1(x_1,\ldots,x_n))=(1);\end{cases}$$

$$s_2(x_1,\ldots,x_n)=\begin{cases} p(x_1,\ldots,x_n) & \text{iff } \mathrm{sq}(r_2(x_1,\ldots,x_n))=(1),\\ q(x_1,\ldots,x_n) & \text{iff } \overline{\mathrm{sq}}(r_2(x_1,\ldots,x_n))=(1)\end{cases}$$

belong to $P(n,k)$. It is straightforward to verify that

$$s(x_1,\ldots,x_n)=f(x_i)\otimes s_1(x_1,\ldots,x_n)\oplus\overline{\mathrm{sq}}(f(x_i))\otimes s_2(x_1,\ldots,x_n).$$

Then s belongs to $P(n,k)$. □

Lemma 4.3. *Let* $s \in P(n, k)$. *The function* s' *defined by* $s'(x_1, \ldots, x_n) = (i \dot{-} 1)$ *iff* $s(x_1, \ldots, x_n) = (i)$ (*where* $i \dot{-} 1 = i - 1$ *for* $i > 0$ *and* 0 *for* $i = 0$) *belongs to* $P(n, k \dot{-} 1)$.

Proof. Inductively for the construction of s. It is easy to notice this fact in the case when s is constant. Let

$$s(x_1, \ldots x_n) = f(x_i) \otimes p(x_1, \ldots, x_n) \oplus \overline{\mathrm{sq}}(f(x_i)) \otimes q(x_1, \ldots, x_n).$$

Therefore,

$$s'(x_1, \ldots, x_n) = f(x_i) \otimes p'(x_1, \ldots, x_n) \oplus \overline{\mathrm{sq}}(f(x_i)) \otimes q'(x_1, \ldots, x_n) \quad \square$$

5. Representability

Type $\boldsymbol{B} = (O \to O) \to ((O \to O) \to (O \to O))$ is called a binary word type because of the isomorphism between $\mathrm{Cl}(\boldsymbol{B})$ and Σ^*. We define that a closed term of type $\boldsymbol{B}$ represents a word $w \in \Sigma^*$ by induction in the following way; Λ is represented by the term $\lambda uvx.x$. If $w \in \Sigma^*$ is represented by the term $W \in \mathrm{Cl}(\boldsymbol{B})$, then words aw and bw are represented by terms $\lambda uvx.u(Wuvx)$ and $\lambda uvx.v(Wuvx)$ respectively. This constitutes a 1-1 correspondence between $\mathrm{Cl}(\boldsymbol{B})$ and Σ^*. The term which represents the word w is denoted by $\underline{w}$. If H is a closed term of type $\boldsymbol{B}^n \to \boldsymbol{B}$, then we call H a λ-word theoretic function. The function $h : (\Sigma^*)^n \to \Sigma^*$ is represented by the term $H \in \mathrm{Cl}(\boldsymbol{B}^n \to \boldsymbol{B})$ iff, for all $x_1, \ldots, x_n \in \Sigma^*$, $H\underline{x_1} \ldots \underline{x_n} = \underline{h(x_1, \ldots, x_n)}$. The term which represents the function h is denoted by $\underline{h}$.

Let us define the following terms with $c, d, e \in \boldsymbol{B}$, $u, v \in (O \to O)$, $x, y \in O$:

$$\mathrm{APP} = \lambda cduvx.cuv(duvx), \qquad \mathrm{SUB} = \lambda cdeuvx.c(\lambda y.duvy)(\lambda y.euvy)x,$$

$$\mathrm{CUT}_a = \lambda cuvx.cu(\lambda y.x)x, \qquad \mathrm{CUT}_b = \lambda cuvx.c(\lambda y.x)vx,$$

$$\mathrm{SQ} = \lambda cuvx.c(\lambda y.ux)(\lambda y.ux)x, \qquad \overline{\mathrm{SQ}} = \lambda cuvx.(\lambda y.x)(\lambda y.x)(ux),$$

$$\mathrm{BEG}_a = \lambda cuvx.c(\lambda y.ux)(\lambda y.x)x, \qquad \mathrm{BEG}_b = \lambda cuvx.c(\lambda y.x)(\lambda y.ux)x,$$

$$\mathrm{OCC}_a = \lambda cuvx.c(\lambda y.ux)(\lambda y.y)x, \qquad \mathrm{OCC}_b = \lambda cuvx.c(\lambda y.y)(\lambda y.ux)x.$$

It is easy to verify that these terms represent the functions app, sub, cut_a, cut_b, sq, $\overline{\mathrm{sq}}$, beg_a, beg_b, occ_a and occ_b respectively.

Lemma 5.1. *Every function* $h \in \lambda\mathrm{def}(n)$ *is represented by some* λ*-word theoretic function* $H \in \mathrm{Cl}(\boldsymbol{B}^n \to \boldsymbol{B})$ *and, for* $x_1, \ldots x_n \in \Sigma^*$, *the following condition holds*: $\underline{h}\underline{x_1} \ldots \underline{x_n} = \underline{h(x_1, \ldots, x_n)}$.

Proof. Basical functions are represented. If a function $g : (\Sigma^*)^n \to \Sigma^*$ is represented by a term $G \in \mathrm{Cl}(\boldsymbol{B}^n \to \boldsymbol{B})$ and functions $f_1, \ldots, f_n : (\Sigma^*)^p \to \Sigma$ are represented by terms $F_1, \ldots, F_n$ respectively, then the composition, i.e. the function $(x_1, \ldots, x_p) \to g(f_1(x_1, \ldots, x_p), \ldots, f_n(x_1, \ldots, x_p))$ is represented by the term $\lambda c_1 \ldots c_p.G(F_1 c_1 \ldots c_p) \ldots (F_n c_1 \ldots c_p)$. Therefore, all functions from $\lambda\mathrm{def}$ are represented. $\square$

Let us define a type $\tau(n, k)$ where $n, k \geq 1$ by $B^n, (O \to O), (O \to O), O^k \to O$. Let f be a function $(\Sigma^*)^n \to \Sigma^*$ and let p be a function $(\Sigma^*)^n \to \{(0), \ldots, (k-1)\}$. We say that the pair (f, p) is represented by a term $T \in \mathrm{Cl}(\tau(n, k))$ if, for all $w_1, \ldots, w_n$, $w \in \Sigma^*$ and for every $i \in \overline{[k-1]}$, the following condition holds:

$$f(w_1, \ldots, w_n) = w \text{ and } p(w_1, \ldots, w_n) = (i)$$

$$\text{iff } T\underline{w_1} \ldots \underline{w_n} =_{\beta\eta} \lambda uvx_{k-1} \ldots x_0.\underline{w}uvx_i.$$

This condition can also be written as

$$T\underline{w_1} \ldots \underline{w_n} =_{\beta\eta} \lambda uvx_{k-1} \ldots x_0.\underline{(f(w_1, \ldots, w_n))}uvx_{\lfloor p(w_1, \ldots, w_n) \rfloor}.$$

For $k = 1$, the type $\tau(n, k)$ is equal to $B^n \to B$. Therefore, this definition of representability is a generalization of the previous one in the type $B^n \to B$.

Lemma 5.2. *For every function $p \in P(n, k)$ there is a term $T \in \mathrm{Cl}(\tau(n, k))$ which represents the pair $(c(n, \Lambda), p)$.*

Proof. Inductively for the construction of p. If p is a constant function such that $p(w_1, \ldots, w_n) = (i)$ for every $w_1, \ldots, w_n \in \Sigma^*$ and $i \in \overline{[k-1]}$, then the pair $(c(n, \Lambda), p)$ is represented by the term $\lambda w_1 \ldots w_n uvx_{k-1} \ldots x_0.x_i$. Let us assume that pairs $(c(n, \Lambda), p)$ and $(c(n, \Lambda), q)$ are represented by terms $P, Q \in \mathrm{Cl}(\tau(n, k))$ respectively. Let $i \in [n]$ and $f \in \mathrm{TEST}$. The function f is represented by $F \in \mathrm{Cl}(B \to B)$ (see Lemma 5.1). Then the pair $(c(n, \Lambda), s)$, where s is defined by the conditional choice rule

$$s(x_1, \ldots x, x_n) = f(x_i) \otimes p(x_1, \ldots, x_n) \oplus \overline{\mathrm{sq}}(f(x_i)) \otimes q(x_1, \ldots, x_n)$$

is represented by the term

$$\lambda w_1 \ldots w_n uvx_{k-1} \ldots x_0.Fw_i(\lambda y.Pw_1 \ldots w_n uvx_{k-1} \ldots x_0)$$
$$(\lambda y.Pw_1 \ldots w_n uvx_{k-1} \ldots x_0)\ (Qw_1 \ldots w_n uvx_{k-1} \ldots x_0). \qquad \square$$

Lemma 5.3. *For every $p \in P(n, k)$ there is a term $P \in \mathrm{Cl}(\tau(n, 1))$ such that the pair $(p, c(1, \Lambda))$ is represented by term P.*

Proof. Since function p belongs to $\lambda\mathrm{def}(n)$ (see Lemma 4.1), p has a representation (see Lemma 5.1). $\square$

Theorem 5.4 (soundness). *For every pair (w, p) such that $w \in \lambda\mathrm{def}(n)$ and $p \in P(n, k)$ there is a term $T \in \mathrm{Cl}(\tau(n, k))$ which represents (w, p).*

Proof. Let $\bar{p}$ be a representative of the pair $(c(n, \Lambda), p)$ (see Lemma 5.2) and $\underline{w}$ be a representative of the function w in type $B^n \to B$ (see Lemma 5.1). The pair (w, p)

is represented by the term

$$\lambda c_1 \ldots c_n uvx_{k-1} \ldots x_0 . \underline{w} c_1 \ldots c_n uv(\bar{p} c_1 \ldots c_n uvx_{k-1} \ldots x_0). \qquad \square$$

Theorem 5.5 (completeness). *Every closed term* $T \in \mathrm{Cl}(\tau(n, k))$ *represents some pair* (w, p) *where* $w \in \lambda\ \mathrm{def}(n)$ *and* $p \in P(n, k)$.

Proof. First we will construct the grammar $G(\tau(n, k))$ which generates all closed terms of type $\tau(n, k)$ (see Theorem 2.2) and then we will prove, by induction on the grammar construction of the term T, that T represents some pair. Let n be fixed. By y^k we understand the variable of type $\tau(n, k)$. Let $\mathrm{NT} = \{y^k \mid k \geq 1\}$. We build up the productions in accordance with Theorem 2.2. Such a grammar consists of a denumerable set of productions which can be assembled in the four production schemas. We will prove by induction that if T represents some pair, then a new term obtained from T by some production also represents another pair. The grammar for type $\tau(n, k)$ is as follows:

$$\alpha_i^k \qquad y^k \Rightarrow \lambda c_1 \ldots c_n uvx_{k-1} \ldots x_0 . x_i,$$

$$\beta^k \qquad y^k \Rightarrow \lambda c_1 \ldots c_n uvx_{k-1} \ldots x_0 . u(y^k c_1 \ldots c_n uvx_{k-1} \ldots x_0),$$

$$\gamma^k \qquad y^k \Rightarrow \lambda c_1 \ldots c_n uvx_{k-1} \ldots x_0 . v(y^k c_1 \ldots c_n uvx_{k-1} \ldots x_0),$$

$$\delta_j^k \qquad y^k \Rightarrow \lambda c_1 \ldots c_n uvx_{k-1} \ldots x_0 . c_j(\lambda z . y^{k+1} c_1 \ldots c_n uvx_{k-1} \ldots x_0 z)$$
$$(\lambda z . y^{k+1} c_1 \ldots c_n uvx_{k-1} \ldots x_0 z)(y^k c_1 \ldots c_n uvx_{k-1} \ldots x_0).$$

α_i^k is an element of $\mathrm{Cl}(\tau(n, k))$. β^k is a function from $\mathrm{Cl}(\tau(n, k))$ to $\mathrm{Cl}(\tau(n, k))$, γ^k is a function from $\mathrm{Cl}(\tau(n, k))$ to $\mathrm{Cl}(\tau(n, k))$, and δ_j^k is a function from $\mathrm{Cl}(\tau(n, k+1)) \times \mathrm{Cl}(\tau(n, k+1)) \times \mathrm{Cl}(\tau(n, k))$ to $\mathrm{Cl}(\tau(n, k))$. Now we will show that every term from $\mathrm{Cl}(\tau(n, k))$ represents some pair (w, p). Element α_i^k represents a pair $(c(n, \Lambda), c(n, (i)))$. If the term $T \in \mathrm{Cl}(\tau(n, k))$ represents a pair (w, p), then $\beta^k(T)$ and $\gamma^k(T)$ represent pairs $(\mathrm{app}(a, w), p)$ and $(\mathrm{app}(b, w), p)$ respectively.

The main part of this theorem will be the proof of the following fact: If E, $F \in \mathrm{Cl}(\tau(n, k+1))$ and $G \in \mathrm{Cl}(\tau(n, k))$ represent some pairs, then the term $\delta_j^k(E, F, G)$ is also representative of a certain pair. The next part of this proof will be a construction of a pair for $\delta_j^k(E, F, G)$ by means of word functions which are represented by E, F and G. Suppose we have three pairs of functions represented respectively by E, F and G. E represents pair (w_e, e), F represents pair (w_f, f) and G represents (w_g, g). Functions w_e, w_f, and w_g are elements of the space $\lambda\mathrm{def}(n)$ and functions $e, f \in P(n, k+1)$ and $g \in P(n, k)$. Let us define a pair (w, p) as follows:

$$w(c_1, \ldots, c_n) = \bigodot_{i=1}^{8} \mathrm{sub}(A_i(c_1, \ldots, c_n), w_i(c_1, \ldots, c_n), \Lambda),$$

$$p_i(c_1, \ldots, c_n) = \bigodot_{i=1}^{8} [A_i(c_1, \ldots, c_n) \otimes p_i(c_1, \ldots, c_n)],$$

where $p_1=g$, $p_2=g$, $p_3=g$, $p_4=f'$, $p_5=g$, $p_6=e'$, $p_7=e'$, and $p_8=f'$. Further,

$$w_1(c_1,\ldots,c_n)=w_g(c_1,\ldots,c_n),$$

$$w_2(c_1,\ldots,c_n)=\mathrm{app}(\mathrm{sub}(c_j,w_e(c_1,\ldots,c_n),w_f(c_1,\ldots,c_n)),\; w_g(c_1,\ldots,c_n)),$$

$$w_3(c_1,\ldots,c_n)=\mathrm{app}(\mathrm{sub}(c_j,w_e(c_1,\ldots,c_n),\Lambda),w_g(c_1,\ldots,c_n)),$$

$$w_4(c_1,\ldots,c_n)=\mathrm{app}(\mathrm{sub}(\mathrm{cut}_a(c_j),w_e(c_1,\ldots,c_n),\Lambda),w_f(c_1,\ldots,c_n)),$$

$$w_5(c_1,\ldots,c_n)=\mathrm{app}(\mathrm{sub}(c_j,\Lambda,w_f(c_1,\ldots,c_n)),w_g(c_1,\ldots,c_n)),$$

$$w_6(c_1,\ldots,c_n)=\mathrm{app}(\mathrm{sub}(\mathrm{cut}_b(c_j),\Lambda,w_f(c_1,\ldots,c_n)),w_e(c_1,\ldots,c_n)),$$

$$w_7(c_1,\ldots,c_n)=w_e(c_1,\ldots,c_n),\qquad w_8(c_1,\ldots,c_n)=w_f(c_1,\ldots,c_n).$$

The A_i's are defined as follows:

$$A_1(c_1,\ldots,c_n)=\overline{\mathrm{sq}}(c_j),$$

$$A_2(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\overline{\mathrm{sq}}(e(c_1,\ldots,c_n))\otimes\overline{\mathrm{sq}}(f(c_1,\ldots,c_n)),$$

$$A_3(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\overline{\mathrm{sq}}(e(c_1,\ldots,c_n))\otimes\mathrm{sq}(f(c_1,\ldots,c_n))\otimes\overline{\mathrm{sq}}(\mathrm{occ}_b(c_j)),$$

$$A_4(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\overline{\mathrm{sq}}(e(c_1,\ldots,c_n))\otimes\mathrm{sq}(f(c_1,\ldots,c_n))\otimes\mathrm{occ}_b(c_j),$$

$$A_5(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\mathrm{sq}(e(c_1,\ldots,c_n))\otimes\overline{\mathrm{sq}}(f(c_1,\ldots,c_n))\otimes\overline{\mathrm{sq}}(\mathrm{occ}_a(c_j)),$$

$$A_6(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\mathrm{sq}(e(c_1,\ldots,c_n))\otimes\overline{\mathrm{sq}}(f(c_1,\ldots,c_n))\otimes\mathrm{occ}_a(c_j),$$

$$A_7(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\mathrm{sq}(e(c_1,\ldots,c_n))\otimes\mathrm{sq}(f(c_1,\ldots,c_n))\otimes\mathrm{beg}_a(c_j),$$

$$A_8(c_1,\ldots,c_n)=\mathrm{sq}(c_j)\otimes\mathrm{sq}(e(c_1,\ldots,c_n))\otimes\mathrm{sq}(f(c_1,\ldots,c_n))\otimes\mathrm{beg}_b(c_j).$$

The functions A_i for $i\in[8]$ describe complete and consistent set of conditions. It means that, for every $c_1,\ldots,c_n$, there is exactly one i such that $A_i(c_1,\ldots,c_n)=(1)$ and, for $j\neq i$, $A_j(c_1,\ldots,c_n)=(0)$. For example, $A_6(c_1,\ldots,c_n)=(1)$ means that c_j is not empty, word $e(c_1,\ldots,c_n)$ is also not empty, word $f(c_1,\ldots,c_n)$ is empty, and a occurs in c_j. Functions $p_1,\ldots,p_8$ belong to $P(n,k)$ (see Lemma 4.3). Function p is obtained from functions in $P(n,k)$ by multiple application of Lemma 4.2; therefore, $p\in P(n,k)$. It is easy to notice that $w\in\lambda\mathrm{def}(n)$ (see Lemma 5.1).

Now, let us check that the term $\delta_j^k(E,F,G)$ represents the pair (w,p). Let $c_1,\ldots,c_n$ be fixed words of Σ^*. We count out the application of the term $\delta_j^k(E,F,G)$ to the arguments $\underline{c_1},\ldots,\underline{c_n}$:

$$\begin{aligned}&\delta_j^k(E,F,G)\underline{c_1}\ldots\underline{c_n}\\ &\quad=_{\beta\eta}\lambda uvx_{k-1}\ldots x_0.\underline{c_j}(\lambda z.E\underline{c_1}\ldots\underline{c_n}uvx_{k-1}\ldots x_0z)\\ &\qquad\qquad(\lambda z.F\underline{c_1}\ldots\underline{c_n}uvx_{k-1}\ldots x_0z)(G\underline{c_1}\ldots\underline{c_n}uvx_{k-1}\ldots x_0)\end{aligned}$$

now we make an α-conversion which changes x_i to x_{i+1} for $i\in\overline{[k-1]}$ and variable z to x_0, and we obtain

$$=_{\beta\eta} \lambda uvx_k\ldots x_1.\underline{c_j}(\lambda x_0.E\underline{c_1}\ldots\underline{c_n}uvx_k\ldots x_1x_0)(\lambda x_0.F\underline{c_1}\ldots\underline{c_n}uvx_k\ldots x_1x_0)(G\underline{c_1}\ldots\underline{c_n}uvx_k\ldots x_1)$$

from the inductive assumption that E, F, G represents pairs (w_e, e), (w_f, f), and (w_g, g) respectively we obtain

$$=_{\beta\eta} \lambda uvx_k\ldots x_1.\underline{c_j}(\lambda x_0.\underline{w_e(c_1,\ldots,c_n)}uvx_{\lfloor e(c_1,\ldots,c_n)\rfloor})(\lambda x_0.\underline{w_f(c_1,\ldots,c_n)}uvx_{\lfloor f(c_1,\ldots,c_n)\rfloor})(\underline{w_g(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor+1})$$

then, according to the conditions described by the A_i functions, we have

$$=_{\beta\eta}\begin{cases}\lambda uvx_k\ldots x_1.\underline{w_1(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor+1} & \text{iff } A_1(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_2(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor+1} & \text{iff } A_2(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_3(c_1,\ldots,c_n)}\,uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor+1} & \text{iff } A_3(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_4(c_1,\ldots,c_n)}uvx_{\lfloor f(c_1,\ldots,c_n)\rfloor} & \text{iff } A_4(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_5(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor+1} & \text{iff } A_5(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_6(c_1,\ldots,c_n)}uvx_{\lfloor e(c_1,\ldots,c_n)\rfloor} & \text{iff } A_6(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_7(c_1,\ldots,c_n)}uvx_{\lfloor e(c_1,\ldots,c_n)\rfloor} & \text{iff } A_7(c_1,\ldots,c_n)=(1),\\ \lambda uvx_k\ldots x_1.\underline{w_8(c_1,\ldots,c_n)}uvx_{\lfloor f(c_1,\ldots,c_n)\rfloor} & \text{iff } A_8(c_1,\ldots,c_n)=(1)\end{cases}$$

after parallel α-conversion which changes x_{i+1} to x_i for $i\in[k]$, we obtain functions w_i, p_i and A_i such as defined above

$$=_{\beta\eta}\begin{cases}\lambda uvx_{k-1}\ldots x_0.\underline{w_1(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor} & \text{iff } A_1(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_2(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor} & \text{iff } A_2(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_3(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor} & \text{iff } A_3(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_4(c_1,\ldots,c_n)}uvx_{\lfloor f'(c_1,\ldots,c_n)\rfloor} & \text{iff } A_4(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_5(c_1,\ldots,c_n)}uvx_{\lfloor g(c_1,\ldots,c_n)\rfloor} & \text{iff } A_5(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_6(c_1,\ldots,c_n)}uvx_{\lfloor e'(c_1,\ldots,c_n)\rfloor} & \text{iff } A_6(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_7(c_1,\ldots,c_n)}uvx_{\lfloor e'(c_1,\ldots,c_n)\rfloor} & \text{iff } A_7(c_1,\ldots,c_n)=(1),\\ \lambda uvx_{k-1}\ldots x_0.\underline{w_8(c_1,\ldots,c_n)}uvx_{\lfloor f'(c_1,\ldots,c_n)\rfloor} & \text{iff } A_8(c_1,\ldots,c_n)=(1)\end{cases}$$

finally, according to the definitions of w and p, we get

$$=_{\beta\eta} \lambda uvx_{k-1}\ldots x_1.\underline{w(c_1,\ldots,c_n)}uvx_{\lfloor p(c_1,\ldots,c_n)\rfloor}. \qquad \square$$

Theorem 5.6. *Every term $T\in \mathrm{Cl}(\boldsymbol{B}^n\to\boldsymbol{B})$ represents some function from the set* λdef.

Proof. This theorem is only a special case of Theorem 5.5. Type $\boldsymbol{B}^n\to\boldsymbol{B}$ is equal to $\tau(n, 1)$ so that term T represents some pair (w, p), where $w\in\lambda\mathrm{def}(n)$ and $p\in P(n, 1)$ (see Theorem 5.5). The set $P(n, 1)$ consists of one function only, namely $c(n, \Lambda)$. Therefore, T represents a pair $(w, c(n, \Lambda))$ in type $\tau(n, 1)$, but this is equivalent with T representing w in $\boldsymbol{B}^n\to\boldsymbol{B}$. $\square$

Theorem 5.7. FUN($\boldsymbol{B}$) *is finitely generated.*

Proof. Let us show that the set of representatives of the distinguished word functions (see Section 4) is a base for FUN($\boldsymbol{B}$). Suppose $T \in \mathrm{FUN}(\boldsymbol{B})$. Term T represents some $w \in \lambda\mathrm{def}$ (Theorem 5.6). If function w is a composition of functions from the base, then $\underline{w}$ is a combination of terms from the base. It is easy to check by induction on the construction of w that $T \equiv \underline{w}$. □

Example 5.8. This example is designed to show the algorithm introduced in Theorem 5.5 which, for a given term of type $\tau(n, k)$, returns the pair of word functions represented by this term. The special case of type $\tau(n, k)$ is the type $\boldsymbol{B}^n \to \boldsymbol{B}$ when $k = 1$. Let $T \in \boldsymbol{B} \to \boldsymbol{B}$ be the term $\lambda cuvx.c(\lambda z.ux)(\lambda z.z)(vx)$ where $c \in \boldsymbol{B}$, $u, v \in (O \to O)$ and $x, z \in O$. The problem is to find the function $\Sigma^* \to \Sigma^*$ represented by this term. We can decompose the term using the grammar technique and obtain $T = \delta_1^1(\beta^2(\alpha_1^2),\ \alpha_0^2,\ \gamma^1(\alpha_0^1))$. Term α_1^2 is $\lambda cuvx_1x_0.x_1$ which represents pair $(c(1, \Lambda), c(1, (1)))$. Term $\alpha_0^2 = \lambda cuvx_1x_0.x_0$ represents pair $(c(1, \Lambda), c(1, (0)))$. Term $\alpha_0^1 = \lambda cuvx_0.x_0$ represents pair $(c(1, \Lambda), c(1, (0)))$. Using Theorem 5.5 we can easily find the representation for $\beta^2(\alpha_1^2)$. This term represents pair $(c(1, a), c(1, (1)))$. Term $\gamma^1(\alpha_0^1)$ represents pair $(c(1, b), c(1, (0)))$.

The main part is the construction of a representative for $\delta_1^1(\beta^2(\alpha_1^2),\ \alpha_0^2,\ \gamma^1(\alpha_0^1))$ by means of representatives of previous terms. We have three pairs $w_f = c(1, \Lambda)$, $f = c(1, (0))$; $w_e = c(1, a)$, $e = c(1, (1))$; $w_g = c(1, b)$, $g = c(1, (0))$ such that (w_e, e) is represented by $\beta^2(\alpha_1^2)$, (w_f, f) is represented by α_0^2, and (w_g, g) is represented by $\gamma^1(\alpha_0^1)$. According to the construction in Theorem 5.5, the pair (w, p) is defined as

$$w(c) = \bigodot_{i=1}^{8} \mathrm{sub}(A_i(c), w_i(c), \Lambda), \qquad p(c) = \bigodot_{i=1}^{8} [A_i(c) \otimes p_i(c)],$$

where the functions

$$A_1(c) = \overline{\mathrm{sq}}(c), \qquad A_2(c) = A_3(c) = A_4(c) = A_7(c) = A_8(c) = (0),$$

$$A_5(c) = \mathrm{sq}(c) \otimes \overline{\mathrm{sq}}(\mathrm{occ}_a(c)) \quad \text{and} \quad A_6(c) = \mathrm{sq}(c) \otimes \mathrm{occ}_a(c).$$

It is sufficient to find p_i and w_i only for this i which has $A_i(c) \neq (0)$. Therefore,

$$w_1(c) = w_g(c) = b,$$

$$w_5(c) = \mathrm{app}(\mathrm{sub}(c, \Lambda, w_f(c)), w_g(c)) = \mathrm{app}(\mathrm{sub}(c, \Lambda, \Lambda), b) = b,$$

$$w_6(c) = \mathrm{app}(\mathrm{sub}(\mathrm{cut}_b(c), \Lambda, w_f(c)), w_e(c)) = \mathrm{app}(\mathrm{sub}(\mathrm{cut}_b(c), \Lambda, \Lambda), a) = a.$$

Functions p_i are

$$p_1(c) = g(c) = \Lambda, \qquad p_5(c) = g(c) = (0), \qquad p_6(c) = e'(c) = (0)$$

so that the function w is as follows;

$$w(c) = \mathrm{app}(\mathrm{sub}(\overline{\mathrm{sq}}(c), b, \Lambda), \mathrm{app}(\mathrm{sub}(\mathrm{sq}(c) \otimes \overline{\mathrm{sq}}(\mathrm{occ}_a(c)), b, \Lambda),$$
$$\mathrm{sub}(\mathrm{sq}(c) \otimes \mathrm{occ}_a(c), a, \Lambda))).$$

Roughly speaking, the function w can be described as:

$$w(c)=\begin{cases} b \text{ iff } c=\Lambda, \\ a \text{ iff } c\neq\Lambda \text{ and } c \text{ includes the letter } a, \\ b \text{ iff } c\neq\Lambda \text{ and } c \text{ does not include the letter } a. \end{cases}$$

Acknowledgment

I would like to thank the anonymous referee for many helpful suggestions and valuable comments.

References

[1] H.P. Barendregt, *The Lambda Calculus*, Studies in Logic and the Foundations of Mathematics (North-Holland, Amsterdam, 1981).
[2] H.B. Curry and R. Feys, *Combinatory Logic Vol 1* (North-Holland Amsterdam 1968).
[3] H. Friedman, *Equality between Functionals*, in: Lecture Notes in Mathematics **453** (Springer, Berlin, 1975) 22-37.
[4] G. Huet, A unification algorithm for typed λ-calculus, *Theoret. Comput. Sci.* **1** (1975) 27-58.
[5] S.L. Kleene, *Introduction to Mathematics* (Van Nostrand, New York, 1952).
[6] I. Rosenberg, The ramification of Slupecki criterion in many-valued logic, *Proc. 24th Conf. on the History of Logic*, Cracow, April 1978.
[7] J.B. Rosser and A.R. Turquette, *Many-valued Logics* (North-Holland, Amsterdam, 1958).
[8] H. Schwichtenberg, Definierbare Funktionen in λ-Kalkuli mit Typen, *Arch. Math. Logik Grundlagenforsch.* **17** (1975/76) 113-114.
[9] R. Statman, The typed λ-calculus is not elementary recursive, *Theoret. Comput. Sci.* **9** (1979) 73-81.
[10] R. Statman, On the existence of closed terms in the typed λ-calculus I, in: R. Hindley and J. Seldin, eds, *Combinatory Logic, Lambda Calculus, and Formalism* (Academic Press, New York, 1980).
[11] R. Statman, λ-Definable functionals and $\beta\eta$ conversion, *Arch. Math. Logik Grundlagenforsch.* **23** (1983) 21-26.
[12] M. Zaionc, The set of unifiers in typed λ-calculus as regular expression, in: Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 430-440.
[13] A. Church, *The Calculi of Lambda-Conversion* (Princeton University Press, Princeton, NY, 1941).
[14] Z. Manna, *Mathematical Theory of Computation* (McGraw-Hill, New York, 1974).