Electronic Notes in Theoretical Computer Science 140 (2005) 41–54 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Probabilistic Approach to the Lambda Definability for Fourth Order Types [1](#_bookmark0)

# Marek Zaionc[2](#_bookmark0)

*Computer Science Department Jagiellonian University Krakow, Poland*

Abstract

It has been proved by Loader [[3](#_bookmark14)] that Statman-Plotkin conjecture (see [[7](#_bookmark18)] and [[5](#_bookmark16)]) fails. The Loader proof was done by encoding the word problem in the full type hierarchy based on the domain with 7 elements. The aim of this paper is to show that the lambda definability problem limited to regular fourth order types is decidable in any finite domain. Obviously λ definability is decidable for 1, 2 and 3 order types. As an additional effect of the result described we may observe that for certain types there is no finite context free grammar generating all closed terms. We prove also that probability that randomly chosen fourth order type (or type of the order not grater then 4 ) admits decidable lambda definability problem is zero.

*Keywords:* probabilistic distribution, lambda calculus, higher order types.

# Syntax of simple typed lambda calculus

We shall consider a simple typed lambda calculus with a single ground type

*O*. The set T of types is defined as follows: *O* is a type and if *τ* and *μ* are types then *τ* → *μ* is a type. We will use the following notation: if *μ, τ*1*, τ*2*, ..., τn* are types then by *τ*1 → *τ*2 → *...* → *τn* → *μ* we mean the type *τ*1 → (*τ*2 → *...* → (*τn* → *μ*)*...*). Therefore, every type *τ* has the form *τ*1 → *...* → *τn* → *O*. By a *τk* → *μ* we mean the type *τ...* → *τ* → *O* with *k* occurrences of type *τ* (with *τ* 0 → *μ* = *μ*). If *τ* has the form

1 Supported by the State Committee for Scientific Research in Poland (KBN), research grant 7T11C 022 21

2 Email: [zaionc@ii.uj.edu.pl](mailto:zaionc@ii.uj.edu.pl)

1571-0661 © 2005 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2005.06.025

*τ*1 → *...* → *τn* → *O*, then *τi* for *i* ≤ *n* are called components of type *τ* and are denoted by *τ* [*i*]. For any type *τ* we define *rank*(*τ* ) and *arg*(*τ* ) as follows: *arg*(*O*) = 0, *rank*(*O*) = 1 and *arg*(*τ*1 → *...* → *τn* → *O*) = *n* and *rank*(*τ*1 → *...* → *τn* → *O*)= max*i*=1*...n rank*(*τ* [*i*]) + 1. We define inductively types *τ* [*i*1*, ..., ik*] by (*τ* [*i*1*, ..., ik*−1])[*ik*] for 1 ≤ *ik* ≤ *arg*(*τ* [*i*1*, ..., ik*−1]).

Definition 1.1 *Type τ is called regular if rank*(*τ* ) ≤ 4 *and every component of τ has arg* ≤ 1*. This implies that only components allowed for regular types are O, O* → *O and* (*Ok* → *O*) → *O for any k.*

For any type *τ* there is given a denumerable set of variables *V* (*τ* ). Any type *τ* variable is a type *τ* term . If *T* is a term having type *τ* → *μ* and *S* is a type *τ* term , then TS is a term which has type *μ*. If *T* is a type *μ* term and *x* is a type *τ* variable , then *λx.T* is a term having type *τ* → *μ*. The axioms of equality between terms have the form of *βη* conversions and the convertible terms will be written as *T* =*βη S*. Term *T* is in a long normal form if *T* = *λx*1*...xn.yT*1*...Tk*, where *y* is an *xi* for some *i* ≤ *n* or *y* is a free variable, *Tj* for *j* ≤ *k* are in a long normal form and *yT*1...*Tk* is a type *O* term. Long normal forms exist and are unique for *βη* conversions (compare [[6](#_bookmark17)]). A closed term is a term without free variables. By *Cl*(*τ* ) we mean a set of all closed type *τ* terms. Let *Y* be a set of typed variables. By *Cl*(*τ, Y* ) we mean a set of all type *τ* terms such that any free variable of such terms is in *Y* . Let us introduce a complexity measure *π* for closed terms. If *T* is a closed term written in a long normal form and *T* = *λx*1*...xn.xi* then *π*(*T* ) = 1. If *T* = *λx*1*...xn.xiT*1*...Tk*, then *π*(*T* )= max*j*=1*...k*(*π*(*λx*1*...xn.Tj*)) + 1. Complexity *π* is just the depth of the B¨ohm tree of a given term. If *y*1*, ..., yn* are some occurrences of free variables in term *T* then *T* [*y*1*/t*1*, ..., yk/tk*] is a term obtained from *T* by replacement of each *yi* be a term *ti* respectively. For more detailed treatment of typed lambda calculus see [[8](#_bookmark19)].

# Model of simple typed lambda calculus

A *full type hierarchy* {*Dτ* }*τ*∈T is a collection of finite domains, one for each type. The whole hierarchy is determined by *DO*. We assume that the set *DO* is a given finite set and *Dτ* →*μ* is a collection of all functions from *Dτ* to *Dμ*. Therefore all *Dτ* are finite. Since any function *f* ∈ *Dτ* →*μ* is a finite set, we assume some sensible encoding of those functions as Go¨del numbers for their graphs. An assignment *φ* is a function associating an object from one of the domains with each variable of typed lambda calculus in such a way that the type *τ* variable is assigned an element from the domain *Dτ* . Any assignment *φ* can be extended to an interpretation *V φ* of all terms in *full type hierarchy*. A function *V φ* is defined by:

1. *V φ*(*x*)= *φ*(*x*)
2. *V φ*(*TS*)= *V φ*(*T* )(*V φ*(*S*))
3. *V φ*(*λxτ .Tμ*) is such a function from *Dτ* to *Dμ* that for each element *X* ∈ *Dτ* the value of this function is *V ψ*(*T* ), where *ψ*, is an assignment which is identical to *φ* except that *ψ*(*x*)= *X*

It can be noted that any type *τ* term is interpreted as an objects from *Dτ* . It is also easy to notice that interpretation *V φ* on term *T* depends only on the assignment *φ* on free variables in *T* . Hence, closed terms have a fixed interpretation in the *full type hierarchy*. Function *f* ∈ *Dτ* is called lambda definable if there is a closed type *τ* term *T* that is interpreted as *f* .

The *lambda deﬁnability problem* is a decision problem to determine whether or not the given object *f* ∈ *τ* ∈T *Dτ* is lambda definable. It was proved by Loader that the problem is undecidable. For the particular type *τ* the *τ- lambda deﬁnability problem* is the decision problem to determine whether or not the given object *f* ∈ *Dτ* is lambda definable. In this case the type *τ* is not a part of the problem.

Definition 2.1 *A function α* : *Dτ n* → *Dτ is called λ preserving function if the following implication holds: if x*1*, ..., xn are lambda deﬁnable objects in Dτ then α*(*x*1*, ..., xn*) *is also a lambda deﬁnable object in Dτ .*

Definition 2.2 *We call* {*a*1*, ..., ap, α*1*, ..., αk*} *a λ system in Dτ if a*1*, ..., ap are some λ deﬁnable objects in Dτ and α*1*, ..., αk are some λ preserving* *functions in Dτ of various arities.*

Definition 2.3 *By the closure of the λ system* {*a*1*, ..., ap, α*1*, ..., αk*} *in Dτ we mean the minimal set of objects in the domain Dτ containing elements a*1*, ..., ap and closed for all functions α*1*, ..., αk. Obviously all objects in the closure are λ deﬁnable functions. The λ system* {*a*1*, ..., ap, α*1*, ..., αk*} *in Dτ is called λ complete if the closure of the system consists of all λ deﬁnable objects in Dτ .*

Lemma 2.4 *For λ system L* = {*a*1*, ..., ap, α*1*, ..., αk*} *the closure is effectively constructible.*

Proof. We are going to construct the closure just by iteration of *λ* preserving functions. Let us define recursively sets *D*0 *, D*1 *, ...* by

*L L*

1. *D*0 = {*a*1*, ..., ap*}

*L*

1. *Di*+1 = *Di* ∪ {*αj*(*x*1*, ..., xp* ): *x*1*, ..., xp*

*L*

*L*

1≤*j*≤*k*

j

∈ *Di* }

j

*L*

We are having a monotonic chain of sets of *λ* definable objects in *Dτ* .

0 ⊆ *D*1

*D*

*L*

*L*

⊆ *...* ⊆ *Dτ* . Since *Dτ* is finite, then there will be an index *i*0 such

that *Di*0 = *Di*0+1. Obviously *Di*0 is a closure of the system *L*. In the case

*L L L*

when the system is complete, the closure *Di*0 consists of all *λ* definable objects in *Dτ* .

*L*

# Term grammars

Term grammars have been introduced in [[9](#_bookmark20)]. A more detailed treatment of term grammars can be found in [[8](#_bookmark19)] or [[10](#_bookmark21)]. A term grammar *G* is a triple (*V, P, S*) where *V* is a finite set of nonterminals, each of which is a typed variable, *P* is a finite set of productions, each of which is a pair (*y, t*) denoted also as *y* → *t*, where *y* ∈ *V* and *t* ∈ *Cl*(*τ, V* ), where *τ* is a common type for *y* and *t*. *S* is a special nonterminal symbol from *V* . A closed term *T* is obtained from closed terms *t*1 ∈ *τ*1*, ..., tk* ∈ *τk* by production *y* → *t* if there is exactly *k* occurrences of nonterminal variables *y*1*, ..., yk* in *t* which have types *τ*1*, ..., τk* respectively, such that *T* =*βη t*[*y*1*/t*1*, ..., yk/tk*]. By a derivation of the closed term *T* we mean a finite sequence of closed terms *t*0*, t*1*, ..., tn* = *T* , such that for every term *ti* in the sequence either there is production *y* → *ti* in grammar *G* ( which means that *ti* is a closed term ) or *ti* is obtained from

previous terms in the sequence by some production. Let us write *y* →∗

*T* to

denote that *T* is obtained from some closed terms *t*1*, ..., tk* derivable in *G* by production *y* → *t*. By *L*(*G, y*) where *y* ∈ *V* we mean the set of all closed terms which are generated from *y* by grammar *G*, i.e. if *y* is type *τ* variable

then *L*(*G, y*)= {*t* ∈ *Cl*(*τ* ): *y* →∗ *t*}. Let us assume that *y* → *t* is production

and *y*1*, ..., yn* are all free occurrences of nonterminal variables in the term *t*. Let *y* ∈ *τ , y*1 ∈ *τ*1 *, ..., yn* ∈ *τn*. This production determines the function *α* : *Cl*(*τ*1) × *...* × *Cl*(*τn*) → *Cl*(*τ* ) defined by: *α*(*t*1*, ..., tn*) = *t*[*y*1*/t*1*, ..., yk/tk*] for every closed terms *t*1 ∈ *Cl*(*τ*1)*, ..., tn* ∈ *Cl*(*τn*). If there are no nonterminal variables in the term *t*, then the production *y* → *t* determines a 0-ary function (constant) *t* which belongs to *Cl*(*τ* ).

Example 3.1 Let *τ* be a following type ((*O* → *O* → *O*) → *O*) → (*O* → *O*). Let us consider the following grammar ({*y*}*, P, y*). Types of auxiliary variables used are the following *p* ∈ (*O, O* → *O*) → *O* and *x, v, z* ∈ *O*. There are 4 productions in the set *P* .

1. *y* → *λpx.x*
2. *y* → *λpx.p*(*λvz.ypx*)
3. *y* → *λpx.p*(*λvz.ypv*)
4. *y* → *λpx.p*(*λvz.ypz*)

Let *α, β, γ, δ* be the functions determined by these productions respec- tively. The closed term *λpx*1*.p*(*λx*2*x*3*.p*(*λx*4*x*5*.p*(*λx*6*x*7*.x*4))) of type *τ* has the

following derivation sequence

* 1. *λpx.x*
  2. *λpx.p*(*λvz.x*)
  3. *λpx.p*(*λvz.p*(*λx*6*x*7*.v*)
  4. *λpx*1*.p*(*λx*2*x*3*.p*(*λx*4*x*5*.p*(*λx*6*x*7*.x*4)))

or the term can be presented as *β*(*γ*(*β*(*α*))). It is easy to prove that this grammar generates all closed terms of type *τ* (compare [[9](#_bookmark20)] or theorem [3.3](#_bookmark5)).

Theorem 3.2 *For every type rank 3 type τ there is a ﬁnite grammar with one type τ nonterminal variable which generates all closed type τ terms.*

Proof. Let *τ* = *τ* [1]*,τ* [2]*, ..., τ* [*n*] → *O*. It will be proved that the following grammar ({*y*}*, P, y*) produces all closed type *τ* terms. The grammar contains exactly *n* productions which have the form:

1. *y* → *λx*1*...xn.xi* if *arg*(*τ* [*i*]) = 0
2. *y* → *λx*1*...xn.xi*(*yx*1*...xn*)*...*(*yx*1*...xn*) if *arg*(*τ* [*i*]) = *k >* 0

It is easy to see that any term produced by the grammar has the type *τ* . The reverse implication is by induction of the complexity *π* of closed terms. Let us suppose that *T* is a closed type *τ* term in a long normal form . If *T* is a projection *λx*1*...xn.xi*, then obviously it can be obtained directly by i-th production. Suppose *T* has a long normal form *λx*1*...xn.xiK*1*...Kk* for some *τ* [*i*] such that *arg*(*τ* [*i*]) = *k* where *Kj* are type *O* terms. Let us define *T*1*, ..., Tk* by *Tj* = *λx*1*...xn.Kj*. Each term *Tj* is a closed type *τ* term and is simpler with respect to complexity *π* then the original term *T* . Therefore by induction closed terms *T*1*, ..., Tk* are to be generated by the grammar *G*. The term *T* can be obtained from *T*1*, ..., Tk* by production (ii).

Theorem 3.3 *[see Zaionc [*[*9*](#_bookmark20)*]] For every regular rank 4 type, such that there is i* ≤ *arg*(*τ* ) *such τ* [*i*] = *O, there is a ﬁnite grammar with one nonterminal variable generating all closed type τ terms. An illustration of this case is presented in the example* [*3.1*](#_bookmark3)

Proof. Let *τ* = *τ* [1]*,τ* [2]*, ..., τ* [*n*] → *O*. Every *τ* [*i*] is one of the following types: *O*, *O* → *O* or (*O, ..., O* → *O*) → *O*. Let us assume that *τ* [*r*] = *O* for some *r* ≤ *n*. We will prove that the following grammar ({*y*}*, P, y*) produces all closed type *τ* term. The grammar contains productions, one for each type *O* and type *O* → *O* components and 1 + *arg*(*τ* [*i*]) productions for each second order component *τ* [*i*]= (*O, ..., O* → *O*) → *O*. The productions have the form:

*pi y* → *λx*1*...xn.xi* if *τ* [*i*]) = *O*

*qi y* → *λx*1*...xn.xi*(*yx*1*...xn*) if *τ* [*i*]= *O* → *O*

*li y* → *λx*1*...xn.xi*(*λz*1*...zp.yx*1*...xn*) if *τ* [*i*]= (*Op* → *O*) → *O*

*r i,*1

*l*

*r*

*l*

*i,arg*(*τ* [*i*])

*y* → *λx*1*...xn.xi*(*λz*1*...zp.yx*1*...xr*−1*z*1*xr*+1*...xn*) if *τ* [*i*]= (*Op* → *O*) → *O*

.

.

.

*y* → *λx*1*...xn.xi*(*λz*1*...zp.yx*1*...xr*−1*zpxr*+1*...xn*) if *τ* [*i*]= (*Op* → *O*) → *O*

We can see that for any term *λx*1*...xn.T* in a long normal form where *T* ∈ *Cl*(*O,* {*x*1*...xn*}) which has a regular type *τ* , there is at the most one occurrence of type *O* variable among all *x*1*...xn* which occurs free in *T* . We are going to prove that every closed type *τ* term can be obtained by the grammar. Let *λx*1*...xn.T* be a closed type *τ* term.

(case 1) If *T* = *xi* for some type *O* variable *xi* then *λx*1*...xn.T* can be obtained by production *pi*.

(case 2) If *T* has the form *xiS* for some term *S* ∈ *Cl*(*O,* {*x*1*...xn*}) where *xi* has type *O* → *O*, then the term *λx*1*...xn.S* has also type *τ* and is simpler than *λx*1*...xn.T* with respect to the complexity measure *π*. Therefore by induction *λx*1*...xn.S* can be obtained by the grammar. Hence our term *λx*1*...xn.T* can be obtained by production *qi* from *λx*1*...xn.S*.

(case 3) If *T* has the form *xi*(*λz*1*...zp.S*) for some term *S* ∈ *Cl*(*O,* {*x*1*...xn, z*1*...zp*}) where *xi* has type (*Op* → *O*) → *O* then *λx*1*...xnz*1*...zp.S* is the closed term with regular type *τ* [1]*,τ* [2]*, ..., τ* [*n*]*, Op* → *O*. According to what we note there is at the most one occurrence of one type *O* variable among *x*1*...xn, z*1*...zp* which occurs free in *S*.

(case 3a) If the only type *O* variable which occurs free in *S* is *xs* for some *s* ≤ *n*, then *λx*1*...xn.S* is a closed type *τ* term and is simpler than *T* with respect to the complexity measure *π*. Therefore by induction *λx*1*...xn.S* can be obtained by the grammar. Hence our term *λx*1*...xn.T* can be obtained by production *li* from *λx*1*...xn.S*.

(case 3b) If the only type *O* variable which occurs free in *S* is *zs* for some 1 ≤ *s* ≤ *p*, then *λx*1*...xrzsxr*+1*...xn.S* is closed type *τ* term and is simpler then *T* with respect to the complexity measure *π*. There- fore by induction *λx*1*...xrzsxr*+1*...xn.S* can be obtained by the grammar. Hence our term *λx*1*...xn.T* can be obtained by produc-

tion *lr* from the term *λx*1*...xrzsxr*+1*...xn.S*.

*i,s*

Theorem 3.4 *[see Zaionc [*[*9*](#_bookmark20)*]] For every regular rank 4 type, such that there is no such i* ≤ *arg*(*τ* ) *that τ* [*i*] = *O, there is a ﬁnite grammar with two nonterminal variables generating all type τ closed terms.*

Proof. Let *τ* = *τ*1*,... , τn* → *O*. Let us define *τ* ' = *τ*1*,... , τn,O* → *O.* The new type *τ* ' satisfies the assumption of theorem [3.3](#_bookmark5), therefore there exists a grammar with one nonterminal type *τ* ' variable *y*', generating all closed terms of *τ* '. Let us add to the existing grammar the following additional productions:

*y* → *λx*1*...xn.xi*(*yx*1*...xn*) if *τ* [*i*]= *O* → *O*

*y* → *λx*1*...xn.xi*(*λz*1*...zp.y*'*x*1*...xn, z*1) if *τ* [*i*]= (*Op* → *O*) → *O*

.

.

.

*y* → *λx*1*...xn.xi*(*λz*1*...zp.y*'*x*1*...xn, zp*) if *τ* [*i*]= (*Op* → *O*) → *O*

The starting symbol for the new grammar is *y*. The proof of correctness of this construction is similar to the proof of theorem [3.3](#_bookmark5).

# Main result

Theorem 4.1 *Suppose there is a grammar with exactly one nonterminal, type τ variable y , generating all closed type τ terms. Then there exists effectively a complete λ system in Dτ for any full type hierarchy.*

Proof. Let {*Dτ* }*τ*∈T be a *full type hierarchy* built up from some finite set *DO*. First we identify all productions *y* → *t* such that there is no nonterminal variable in *t*. For each such production *p* : *y* → *t*, term *t* is a closed type *τ* term. Let us evaluate *t* in the *full type hierarchy* to find an object *ap* ∈ *Dτ* . Then let us identify all productions *p* : *y* → *t*[*y, ...y*] such that there are some *k >* 0 occurrences of nonterminal variable *y* in *t*. For each such production we will define a function *αp* : *Dτ k* → *Dτ* . Let *p* : *y* → *t*[*y, ...y*] be a production with exactly *k >* 0 occurrences of nonterminal variable *y* in *t*[*y, ..., y*]. Let

*t*1*, ..., tk* be given objects from *Dτ* . Let us define the term *t*[*y*1*, ..., yk*] obtained by replacement of each occurrence of the free variable *y* by the new type *τ* variables *y*1*, ..., yk* respectively. Now let us define *αp*(*t*1*, ..., tk*) as an object *V φ*(*t*[*y*1*, ..., yk*]) from *Dτ* where *φ* is an assignment associating *yi* with *ti* for all *i* ≤ *k*. Therefore we have defined the function *αp* : *Dτ k* → *Dτ* . All such functions and such objects form the *λ* system in *Dτ* . We need to prove that such *λ* system is complete. Suppose an object *a* ∈ *Dτ* is *λ* definable. We are going to show that *a* belongs to the closure of the just constructed *λ* system. There is a closed *λ* term *Ta* interpreted as *a*. Term *Ta* has a finite derivation in the grammar *G*. If the length of the derivation is 1, it means that *Ta* is obtained from production *y* → *Ta*. In this case interpretation of *Ta* is an *a* ∈ *Dτ* which belongs to the *λ* system. In the case the length of the derivation for *Ta* is longer than 1, *Ta* is obtained from some previous terms in the derivation sequence, let us say from *R*1*, ..., Rk*, by some production *p* : *y* → *t* from the grammar. Let *αp* be a function *αp* : *Dτ k* → *Dτ* associated with the production *p* : *y* → *t*. Let *b*1*, ..., bk* be interpretations of closed terms *R*1*, ..., Rk* in *Dτ* . By induction we assume that *b*1*, ..., bk* belong already to the closure of the *λ* system. From the construction of the *λ* system it follows that *a* = *fp*(*b*1*, ..., bk*). So *a* ∈ *Dτ* also belongs to the closure.

Theorem 4.2 *For any rank 3 type τ or for any regular rank 4 τ for which there is i* ≤ *arg*(*τ* ) *such that τ* [*i*]= *O the λ deﬁnability problem is decidable in full type hierarchy built up from any ﬁnite set DO.*

Proof. Given is an object *f* ∈ *Dτ* . For type *τ* there is effectively a finite grammar with exactly one nonterminal variable generating all closed type *τ* terms . The following procedure applies. First we find the grammar ( theorem [3.2](#_bookmark4) , [3.3](#_bookmark5) ) which generates all closed type *τ* terms . Then having the grammar we construct the *λ* system using procedure described in theorem [4.1](#_bookmark6). The *λ* system formed from such grammar must be complete ( theorem [4.1](#_bookmark6) ). Finally, we find the closure of this system (lemma [2.4](#_bookmark2)) which consists of all *λ* definable objects in *Dτ* . Then we check whether or not the given function *f* ∈ *Dτ* belongs to this closure.

Theorem 4.3 *Let τ be a regular rank 4 type such that there is no such i* ≤ *arg*(*τ* ) *that τ* [*i*]= *O. The λ deﬁnability problem for τ is decidable in full type hierarchy built up from any ﬁnite set DO.*

Proof. Given is an object *f* ∈ *Dτ* . Suppose *τ* = *τ* [1]*, ..., τ* [*n*] → *O*. Let us define a new type *τ* ' as *τ* [1]*, ..., τ* [*n*]*,O* → *O*. Type *τ* ' satisfies assumptions of theorem [3.3](#_bookmark5) and therefore there is a finite grammar with exactly one nonter- minal variable *y*' generating all closed type *τ* ' terms. Using theorem [4.2](#_bookmark7) we

construct effectively the finite set *Dλ* of all *λ* definable objects in *Dτ*' . To

*τ* '

generate closed type *τ* terms we add additional productions

1. *y* → *λx*1*...xn.xi*(*yx*1*...xn*) if *τ* [*i*]= *O* → *O*

(2.1) *y* → *λx*1*...xn.xi*(*λz*1*...zp.y*'*x*1*...xn, z*1) if *τ* [*i*]= (*Op* → *O*) → *O*

.

.

.

(2.p) *y* → *λx*1*...xn.xi*(*λz*1*...zp.y*'*x*1*...xn, zp*) if *τ* [*i*]= (*O, ..., O* → *O*) → *O*

Any type (1) production defines the unary function *α* : *Dτ* → *Dτ* in the usual way (see theorem 4.1). Productions 2.1 - 2.p define *p* unary functions

*β*1 : *Dτ*'

→ *Dτ , ..., βp* : *Dτ*'

→ *Dτ* . The following procedure applies. First find

the image *I* of *Dλ*

*τ* '

by all functions *β*1*, ..., βp*.

*I* = 1≤*i*≤*p βi*(*Dλ* )

*τ* '

Then let us find the closure of the set *I* in *Dτ* with respect to all functions *α* (see lemma 2.4 ). This closure is a set of all *λ* definable functions in *Dτ* . Finally, we check whether the given function *f* ∈ *Dτ* belongs to this closure. The proof of correctness of this procedure is identical with the proof of theorem

[4.2](#_bookmark7).

Theorem 4.4 *λ deﬁnability problem is decidable for all rank 1, 2, 3 types and for regular rank 4 types.*

Proof. For the rank 3 and regular rank 4 types see theorems [4.2](#_bookmark7) and [4.3](#_bookmark8). For the type *O* of rank 0 the *λ* definability problem is trivial since there is no *λ* definable objects in *DO*. For rank 2 types the problem is decidable since for such types there are only finite number of closed terms.

Theorem 4.5 *λ deﬁnability problem is undecidable for any non regular rank 4 type.*

Proof. Proof is based on the observation that the type L = ((*O* → *O*) →

*O*) → ((*O* → (*O* → *O*) → *O*) is the simplest non regular type of rank 4. Therefore by simple lambda definable coding L can be embedded to any non regular type of rank 4. But the *λ* definability problem is undecidable for L (see [[1](#_bookmark12)]) .

Example 4.6 Let us consider the *full type hierarchy* built from the set *DO* =

{0*,* 1}. The problem is to find all *λ* definable elements in the set *Dτ* for type *τ* of Church’s numerals (*O* → *O*) → (*O* → *O*). Let us name 4 elements of the set *DO*→*O* respectively by *a, b, c* and *d*, where those are given by the following

true table:

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| *a* | 0 | 0 |
| *b* | 0 | 1 |
| *c* | 1 | 0 |
| *d* | 1 | 1 |

The set *Dτ* consists of 256 functionals. The grammar for generating all Church’s numerals is the following: (see theorem [3.2](#_bookmark4)).

1. *y* → *λux.x*
2. *y* → *λux.u*(*yux*)

The *λ* system {*a*0*, α*} associated with the grammar consists of one lambda definable object *a*0, which is an interpretation of the term *λux.x*, and one unary function *α* which is an interpretation of the production *y* → *λux.u*(*yux*).

Object *a*0 is given by the true-table.

*a*0 = {*a* → *b, b* → *b, c* → *b, d* → *b*}

It can be found easily that the value of *α* on the object *a*0 is another object

*a*1 given by:

*a*1 = {*a* → *a, b* → *b, c* → *c, d* → *d*}

The value of *α* on *a*1 is a new object, let us say *a*2 given by:

*a*2 = {*a* → *a, b* → *b, c* → *b, d* → *d*}

Since the value *α*(*a*2)= *a*1 we have found all lambda definable functionals

{*a*0*, a*1*, a*2} among all 256 elements of *Dτ* . In fact the object *a*1 is an interpre- tation of all odd Church’s numerals and *a*2 is an interpretation of all positive even Church’s numerals. The object *a*0 is the interpretation of Church’s zero.

Theorem 4.7 *As a trivial side effect of the result described in theorems* [*4.1*](#_bookmark6) *and* [*4.5*](#_bookmark9) *we may observe that there is no ﬁnite grammar generating all closed terms for any of non regular rank 4 types.*

# Probabilistic approach

In this section we will consider probability of the fact that randomly chosen 4 order type has decidable lambda definability problem. So we investigate the size of the fraction of number of types of the given length *n* in which the *λ* definability problem is decidable in *full type hierarchy* against the number of

all types of length *n*. We are specially interested in asymptotic behavior of this fraction. Our interest lays in finding limit of that fraction when *n* → ∞. If the limit exists it represents the real number between 0 and 1 which we may call *the density of decidability*. We prove that rank 4 types with decidable *λ* definability problem are asymptotically empty meaning that the limit of that fraction is 0.

First we have to establish the way the length of types are measured.

Definition 5.1 *By τ * *we mean the length of type τ which we deﬁne as the total number of occurrences of atomic type O in the given type. Parenthesis which are sometimes necessary and arrow sign itself are not included in the length of type. Formally, ai * =1 *and φ* → *ψ * = *φ * + *ψ * *.*

Definition 5.2 *We associate the density μ*(A) *with a subset* A⊂ T *of types as:*

(1)

*if the limit exists.*

*μ*(A) = lim

*n*→∞



#{*τ* ∈A : *τ* = *n*}

#{*τ* ∈ T : *τ* = *n*}

The number *μ*(A) if exists is an asymptotic probability of finding type from the class A among all types or it can be interpreted as the asymptotic density of the set A in the set of types. It can be seen immediately that the density *μ* is finitely additive so if A and B are disjoint classes of formulas such that *μ*(A) and *μ*(B) exist then *μ*(A∪B) also exists and *μ*(A∪B)= *μ*(A)+ *μ*(B)*.* It is straightforward to observe that for any finite set A the density *μ*(A) exists and is 0. Dually for co-finite sets A the density *μ*(A) = 1. Nevertheless the density *μ* is not countably additive.

* 1. *Elementary Counting of Types*

In this section we present some properties of numbers characterizing the amount of types of different ranks. We may observe that many results and methods could be rephrased purely in terms of binary trees with given prop- erties. Obviously any type of size *n* can be seen as a binary tree with *n* leaves. More advanced type counting can be found Moczurad, Tyszkiewicz, Zaionc [[4](#_bookmark15)]. See also Kostrzycka and Zaionc [[2](#_bookmark13)] and [[11](#_bookmark22)].

Definition 5.3 *By Fn and Gn we mean respectively the total number of types*

*k k*

*of rank k and size n and the total number of types of rank* ≤ *k and size n so:*

1. *Fn* =# {*φ* ∈ T : *φ * = *n*} *,*

*k*

1. *Gn* =# {*φ* ∈ T : *φ* ≤ *n*} *.*

*k*

Lemma 5.4 *Fn and Gn are given by the following mutual recursion:*

*k k*

1. *Fn* = *if n* =1 *then* 1 *else* 0

1

1. *Gn* = *if n* =1 *then* 1 *else* 0

1

*n n*

1. *Fn*

= Σ *FiGn*−*i* + Σ *Gi Fn*−*i*

*k*+1

*k k*

*i*=0

*i*=0

*k k*+1

1. *Gn* = *Gn* + *Fn*

*k*+1 *k k*+1

Proof. The equality [4](#_bookmark10) and [5](#_bookmark10) are obvious since the type *O* is the only one of size 1. Recursive equation [7](#_bookmark10) is obtained directly from the definition of rank in section [1](#_bookmark1). The type *τ* → *μ* has rank *k* + 1 either when *τ* is exactly of rank *k* and rank of *μ* is ≤ *k* or when rank of *τ* is ≤ *k* and rank of *μ* is exactly *k* + 1. This leads to the recursive equation [6](#_bookmark10).

* 1. *Generating functions*

We are going to use the generating function technique for proving the asymp- totic behavior of appropriate fractions. For the purpose of counting for every

*k* ≥ 1 we define the pair of generating functions *fk*(*z*) = Σ∞ *Fizi* and

Σ∞ *i i*

*gk*(*z*)=

*i*=0 *Gkz* .

*i*=0 *k*

Lemma 5.5 *Functions fk and gk satisfy the following recursive deﬁnitions:*

1. ​

*fk*+1

(*x*) = *fk*(*x*)*gk*(*x*)

1 − *gk*(*x*)

1. *gk*+1(*x*)= *gk*(*x*)+ *fk*+1(*x*)

Proof. By simple encoding of [4](#_bookmark10), [5](#_bookmark10), [6](#_bookmark10) and [7](#_bookmark10) we get equations

1. *f*1(*x*)= *x*
2. *g*1(*x*)= *x*
3. *fk*+1(*x*)= *fk*(*x*)*gk*(*x*)+ *gk*(*x*)*fk*+1(*x*)
4. *gk*+1(*x*)= *gk*(*x*)+ *fk*+1(*x*)

Solving it proves the lemma

Theorem 5.6 *Let* R ⊂ T *be a set of all regular types of rank 4. Let Rn be the number of elements of* R *of size n. The generating function fR for the sequence Rn is*

*x*3

*fR*(*x*)= (1 − 2*x*)(1 − *x* − *x*2)

Proof. We start with counting regular types of rank not greater then 3. Let *Ln* stands for the number of such types of size *n*. Obviously we get *L*0 = 0*, L*1 = 1 and moreover *Ln* = *Ln*−1 + *Ln*−2 since any regular type of rank not greater then 3 must be of either of the form *O* → *τ* or (*O* → *O*) → *τ* for some

shorter regular type *τ* . As we can see *Ln* forms the sequence of Fibonacci numbers. Therefore the generating function *fL* for the sequence *Ln* is

1

*fL*(*x*)= 1 − *x* − *x*2 *.*

By generic regular fourth order types we mean any type of the form ((*Ok* → *O*) → *O*) → *τ* for some regular type *τ* of rank not greater then 3 and for some *k* ≥ 1. Let *Gn* stands for the number of such types of size *n*. Since for every *k* ≥ 1 there is exactly one type of the form ((*Ok* → *O*) → *O*) the generating function *fG* for the sequence *Gn* must be

*x*3 *x*3

*fG*(*x*)= 1 − *x* · *fL*(*x*)= (1 − *x*)(1 − *x* − *x*2) *.*

Finally we observe, that any regular fourth order type type is either generic regular forth order type or can be obtain from other shorter regular type *τ* of rank 4 by the construction *O* → *τ* or (*O* → *O*) → *τ* or ((*Ok* → *O*) →

*O*) → *τ* . Let *Rn* stands for the number of all regular forth order types of size *n*. Therefore *Rn* has to satisfy the following recursive equation *Rn* = *Gn* + *i*=*n*−1 *Ri*. This can be translated in to equation for the generating function *fR* namely *fR*(*x*)= *fG*(*x*)+ *xf*R(*x*) .

*i*=1

Σ

1−*x*

Theorem 5.7 *The density of rank* 4 *types with decidable λ deﬁnability prob- lem among all rank 4 types is* 0*.*

Proof. It is enough to find the closed form for generating functions involved.

Namely for all rank 4 types the function calculated from equality [8](#_bookmark11) is

*x*4

*f*4(*x*)= 1 − 5*x* + 7*x*2 − 2*x*3

Which can be easily turned into power serious *O* 3+√5 *n* . The closed form term for the function *fR*(*x*) enumerating regular 4 order types returns

2

rate of Fibonacci numbers namely *O* 1+√5 *n* .

2

It doesn’t help much to add all third order types for which we know the lambda definability problem is decidable. Undecidability have again a domi- nating factor.

Theorem 5.8 *The density of types of rank* ≤ 4 *with the decidable λ deﬁn- ability problem among all types of rank* ≤ 4 *is again* 0*.*

Proof. It is enough to find the closed form for generation functions involved. For all rank ≤ 4 types the function is

*x* − 2*x*2

*g*4(*x*)= 1 − 3*x* + *x*2

The respective power serious is again *O* 3+√5 *n* . Function *g*

2

3

obtained

from equality [9](#_bookmark11) is *x*3

(1−2*x*)(1−*x*)

. The closed form term for the function *fR*

(*x*) +

*g*3(*x*) enumerating regular 4 order types plus all third order types returns rate

of *O* (2*n*).

# References

1. Thierry Joly, On the λ definability I, Problem at Fixed Model and Generalizations of the Matching Problem, to appear at Fundamentae Informaticae.
2. Zofia Kostrzycka, Marek Zaionc, Statistics of intuitionistic versus classical logics, Studia Logica 76(3) (2004) 307 -328.
3. Ralph Loader, The Undecidability of λ - Definability, *In Logic, Meaning and Computation: Essays in Memory of Alonzo Church,* 331-342, C.A. Anderson and Zeleny editors, Kluwer Academic Publishers, 2001.
4. Ma-lgorzata Moczurad, Jerzy Tyszkiewicz, Marek Zaionc, Statistical properties of simple types, Mathematical Structures in Computer Science 10 (2000), 575-594.
5. Gordon D. Plotkin. λ definability and logical relations, Memorandum SAI-RM-4, School of Artificial Intelligence, University of Edinburgh, Ocober 1973
6. Richard Statman. On the existence of closed terms in the typed λ-calculus. In: R. Hindley and J. Seldin, eds. Combinatory Logic, Lambda Calculus and Formalism (Academic Press, New York, 1980).
7. Richard. Statman. Equality of functionals revisited, in L.A. Harrington et al. (Eds.), *Harvey* *Friedman’s Research on the Foundations of Mathematics*, North- Holland, Amsterdam, 1985, 331-338.
8. David A. Wolfram. The Clausual Theory of Types, Cambridge Tracts in Theoretical Computer Science 21, Cambridge University Press 1993.
9. Marek Zaionc. The set of unifiers in typed λ calculus as regular expression, Rewriting techniques and Applications, Lecture Notes in Computer Science 202, Springer 1985, 430-440.
10. Marek Zaionc. Word Operations Definable in the Typed λ calculus, Theoretical Computer Science 52, (1987 ) pp 1- 14.
11. Marek Zaionc, On the asymptotic density of tautologies in logic of implication and negation, Reports on Mathematical Logic 39 (2004).