A NATURAL COUNTING OF LAMBDA TERMS

Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne, Marek Zaionc

A NATURAL COUNTING OF LAMBDA TERMS

MACIEJ BENDKOWSKI[†], KATARZYNA GRYGIEL[†], PIERRE LESCANNE[†,‡]
AND
MAREK ZAIONC[†]

[†]JAGIELLONIAN UNIVERSITY,
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE,
THEORETICAL COMPUTER SCIENCE DEPARTMENT,
UL. PROF. ŁOJASIEWICZA 6, 30-348 KRAKÓW, POLAND

[‡]UNIVERSITY OF LYON,
ÉCOLE NORMALE SUPÉRIEURE DE LYON,
LIP (UMR 5668 CNRS ENS LYON UCBL INRIA)
46 ALLÉE D'ITALIE, 69364 LYON, FRANCE

Abstract. We study the sequences of numbers corresponding to lambda terms of given sizes, where the size is this of lambda terms with de Bruijn indices in a very natural model where all the operators have size 1. For plain lambda terms, the sequence corresponds to two families of binary trees for which we exhibit bijections. We study also the distribution of normal forms, head normal forms and strongly normalizing terms. In particular we show that strongly normalizing terms are of density 0 among plain terms.

Keywords:lambda calculus, combinatorics, functional programming, test, random generator, bijection, binary tree, asymptotic

1. Introduction

In this paper we consider a natural way of counting the size of λ-terms, namely λ-terms presented by de Bruijn indices[1] in which all the operators are counted with size 1. This means that abstractions, applications, successors and zeros have all size 1. Formally

$$\begin{array}{rcl} |\lambda M| & = & |M| + 1 \\ |M_1\, M_2| & = & |M_1| + |M_2| + 1 \\ |Sn| & = & |n| + 1 \\ |0| & = & 1. \end{array}$$

For instance the term for K which is written traditionally $\lambda x.\lambda y.x$ in the lambda calculus is written $\lambda\lambda S0$ using de Bruijn indices and we have:

$$|\lambda\lambda S0| = 4.$$

The first author was supported by the National Science Center of Poland, grant number 2011/01/B/HS1/00944, when the author hold a post-doc position at the Jagiellonian University within the SET project co-financed by the European Union.

[1]Readers not familiar with de Bruijn indices are invited to read Appendix A.

since there are two λ abstractions, one successor S and one $\mathtt{0}$. The term for S (which should not be confused with the successor symbol) is written $\lambda x.\lambda y.\lambda z.(xz)(yz)$ which is written $\lambda\lambda\lambda(((SS\mathtt{0})\mathtt{0})((S\mathtt{0})\mathtt{0}))$ using de Bruijn indices and its size is:

$$|\lambda\lambda\lambda(((SS\mathtt{0})\mathtt{0})((S\mathtt{0})\mathtt{0}))| = 13.$$

since there are three λ abstractions, three applications, three successors S's, and four $\mathtt{0}$'s. The term $\lambda x.xx$ which corresponds to the term $\lambda(\mathtt{0}\,\mathtt{0})$ has size 4 and the term $(\lambda x.xx)(\lambda x.xx)$ which corresponds to the term ω is written $(\lambda(\mathtt{0}\,\mathtt{0}))\,\lambda(\mathtt{0}\,\mathtt{0})$ and has size 9. The term $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ which corresponds to the fixpoint Y is written $\lambda((\lambda((S\mathtt{0})\,(\mathtt{0}\,\mathtt{0})))\ \lambda((S\mathtt{0})\,(\mathtt{0}\,\mathtt{0})))$ and has size 16.

2. Lambda terms

2.1. Counting plain terms with a natural size: L_∞. Since the terms are either applications, abstractions or de Bruijn indices, the set $\mathcal{L}_\infty$ of lambda terms is solution of the combinatorial equation:

$$\mathcal{L}_\infty = \mathcal{L}_\infty\,\mathcal{L}_\infty \ \oplus\ \lambda\mathcal{L}_\infty \ \oplus\ \mathcal{D}$$

where $\mathcal{D}$ is the set of de Bruijn indices which is solution of

$$\mathcal{D} = S\mathcal{D} \oplus \mathtt{0}$$

Let us call L_∞ the generating function for counting the numbers of the plain terms. It is solution of the functional equation:

$$L_\infty \quad = \quad zL_\infty^2 + zL_\infty + \frac{z}{1-z},$$

which yields the equation:

$$zL_\infty^2 - (1-z)L_\infty + \frac{z}{1-z} = 0 \tag{1}$$

which has discriminant

$$\begin{aligned}\Delta_{L_\infty} \quad &= \quad (1-z)^2 - 4\frac{z^2}{1-z} \quad = \quad \frac{(1-z)^3-4z^2}{1-z}\\ &= \quad \frac{1-3z-z^2-z^3}{1-z}\end{aligned}$$

This gives the solution

$$\begin{aligned}L_\infty \quad &= \quad \frac{(1-z)-\sqrt{\Delta_{L_\infty}}}{2z}\\ &= \quad \frac{(1-z)^{3/2}-\sqrt{1-3z-z^2-z^3}}{2z\sqrt{1-z}}\end{aligned}$$

which has $\rho_{L_\infty} \doteq 0.29559774252208393$ as pole closest to 0. The 18 first values of $[z^n]L_\infty$ are:

0, 1, 2, 4, 9, 22, 57, 154, 429, 1223, 3550, 10455, 31160, 93802, 284789, 871008, 2681019

This sequence is **A105633** in the *Online Encyclopedia of Integer Sequences*.

Theorem 1. *Assume* $C \doteq 0.60676...$ *and* $\rho_{L_\infty} \doteq 0.29559...$
that is $1/\rho_{L_\infty} \doteq 3.38297...$

$$[z^n]L_\infty \sim \left(\frac{1}{\rho_{L_\infty}}\right)^n \frac{C}{n^{\frac{3}{2}}}$$

Proof. The proof mimics this of Theorem 1 in [3]. Let us write L_∞ as

$$\begin{aligned} L_\infty &= \frac{(1-z)-\sqrt{\frac{1-3z-z^2-z^3}{1-z}}}{2z} \\ &= \frac{(1-z)-\sqrt{\rho_{L_\infty}(1-\frac{z}{\rho_{L_\infty}})\frac{Q(z)}{1-z}}}{2z} \end{aligned}$$

where

$$\begin{aligned} R(z) &= z^3+z^2+3z-1 \\ Q(z) &= \frac{R(z)}{\rho_{L_\infty}-z} \end{aligned}$$

Applying Theorem VI.1 of [2], we get:

$$[z^n]L_\infty \sim \left(\frac{1}{\rho_{L_\infty}}\right)^n \cdot \frac{n^{-3/2}}{\Gamma(-\frac{1}{2})}\,\widetilde{C}$$

with

$$\widetilde{C} = \frac{-\sqrt{\rho_{L_\infty}\frac{Q(\rho_{L_\infty})}{1-\rho_{L_\infty}}}}{2\rho_{L_\infty}}$$

Notice that $Q(\rho_{L_\infty}) = R'(\rho_{L_\infty}) = 3\rho_{L_\infty}^2 + 2\rho_{L_\infty} + 3$. From this we get

$$C = \frac{\widetilde{C}}{\Gamma(-\frac{1}{2})} \doteq 0.60676...$$

□

Figure 1 shows approximations of $[x^n]L_\infty$.

2.2. **An holonomic presentation.** Using the Maple package gfun [6] we were able to build a holonomic equation of which L_∞ is the solution namely

$$z^3+z^2-2z+(z^3+3z^2-3z+1)L_\infty+(z^5+2z^3-4z^2+z)L'_\infty = 0.$$

From this equation it is possible to derive the following recursive and linear definition for the coefficients:

$$L_{\infty,0} = 0 \qquad L_{\infty,1} = 1 \qquad L_{\infty,2} = 2 \qquad L_{\infty,3} = 4$$

$$L_{\infty,n} = \frac{(4n-1)L_{\infty,n-1}-(2n-1)L_{\infty,n-2}-L_{\infty,n-3}-(n-4)L_{\infty,n-4}}{n+1}$$

2.3. **Counting terms with at most m indices: L_m.** The set $\mathcal{L}_m$ of terms with free indices $0, .., m-1$ is described as

$$\mathcal{L}_m = \mathcal{L}_m\mathcal{L}_m \oplus \lambda\mathcal{L}_{m+1} \bigoplus_{i=0}^{m-1} S^i(0).$$

The set $\mathcal{L}_0$ is the set of closed lambda terms. If we consider the λ-terms with at most m free indices, we get:

$$L_m = zL_m^2 - zL_{m+1} + \frac{z(1-z^m)}{1-z}$$

$\mathbf{n}$	$[\mathbf{x^n}]\mathbf{L}_\infty$
10	3550
20	253106837
30	27328990723991
40	3503758934959966001
50	493839291745701673090756
60	73920774614279746859303111580
70	11535317831253359292868402823579507
80	1855899670106913269845444317474927546423
90	305649725186484753579669948042728038245882292
100	51274965000307280025396615989999357497440689837989
$\mathbf{n}$	$\lfloor (\mathbf{1}/\rho_{\mathbf{L}_\infty})^{\mathbf{n}} \mathbf{C}/\mathbf{n^{3/2}} \rfloor$
10	3767
20	261489930
30	27945182509468
40	3563589864915927683
50	500623883981281516056181
60	74770204056757299054875868847
70	11649230835743409545961872906078995
80	1871967051054756263272240387385909197928
90	308005368563187477433148735955649926279818246
100	51631045600653143846184406311963448514677624135086

FIGURE 1. Approximation of $[x^n]L_\infty$.

which yields:

$$zL_m^2 - L_m + z\left(L_{m+1} + \frac{1-z^m}{1-z}\right) \quad = \quad 0.$$

Let us state

$$\Delta_{L_m} = 1 - 4z^2\left(L_{m+1} + \frac{1-z^m}{1-z}\right)$$

we have

$$L_m = \frac{1-\sqrt{\Delta_{L_m}}}{2z} = \frac{1-\sqrt{1-4z^2\left(L_{m+1}+\frac{1-z^m}{1-z}\right)}}{2z}.$$

Notice that L_m is defined using L_{m+1}. If this definition is developed, then L_m is defined by an infinite sequence of nested radicals. The sequences $([z^n]L_m)_{n\in\mathbb{N}}$ do not occur in the *Online Encyclopedia of Integer Sequences*.

2.4. **Counting λ-terms with another notion of size.** Assume we take another notion of size in which 0 has size 0 and applications have size 2, whereas abstraction and succession keep their size 1. In other words:

$$\begin{aligned}
|\lambda M| &= |M| + 1\\
|M_1\, M_2| &= |M_1| + |M_2| + 2\\
|Sn| &= |n| + 1\\
|0| &= 0.
\end{aligned}$$

The generating function[2] A_1 fulfills the identity:

$$z^2 A_1^2 - (1-z)A_1 + \frac{1}{1-z}.$$

The reader may check that

$$L_\infty = z\,A_1 \quad \text{and} \quad [z^n]A_1 = [z^{n+1}]L_\infty.$$

Hence both notions of size correspond to sequence **A105633**. In Appendix B we consider the case where all the operators (application, abstraction and succession) have size 1 and 0 has size 0.

3. Typable Terms

A difficult open problem is to count simply typable terms. In this section, we give empiric results we obtain by an implementation on counting closed typed terms.

size	**typables**	**all**
0	0	0
1	0	0
2	1	1
3	1	1
4	2	3
5	5	6
6	13	17
7	27	41
8	74	116
9	198	313
10	508	895
11	1371	2550
12	3809	7450
13	10477	21881
14	29116	65168
15	82419	195370
16	233748	591007
17	666201	1798718
18	1914668	5510023
19	5528622	16966529
20	16019330	52506837
21	46642245	163200904
22	136326126	509323732
23	399652720	1595311747
24	1175422931	5013746254

Figure 2. Numbers of typable closed terms vs numbers of closed terms

[2]We write this function A_1 as a reference to the function $A(x,1)$ described in sequence **A105632** of the *Online Encyclopedia of Integer Sequences*.

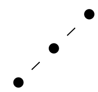
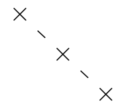
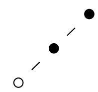
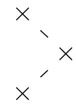
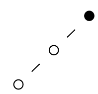
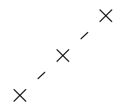
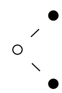
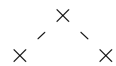
$\lambda-$**terms**	**black-white trees**	**zigzag free trees**	**neutral hnf**
$S^2 0$			$S^3 0$
$\lambda S 0$			$0\,(S0)$
$\lambda\lambda 0$			$0\,(\lambda 0)$
$0\,0$			$(S0)\,0$

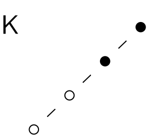
FIGURE 3. Bijection between λ-terms, E_1-free black-white binary trees, zigzag-free trees of size 3 ($L_3 = 4$) and neutral head normal forms (Section 8) of size 4 ($K_4 = 4$).

4. E-FREE BLACK-WHITE BINARY TREES

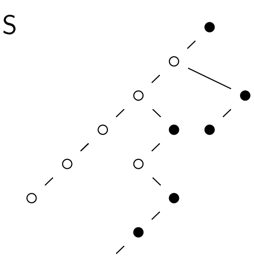
A black-white binary tree is a binary tree with colored nodes using two colors, *black* $\bullet$ and *white* $\circ$. *The root of a black-white binary tree is* $\bullet$*, by convention.* A E-free black-white binary tree is a black-white binary tree in which edges from a set E are forbidden. For instance if the set of forbidden edges is $E_1 = \{ {}_{\bullet}\!/^{\circ}, {}^{\bullet}\backslash_{\circ}, {}^{\bullet}\backslash_{\bullet}, {}^{\circ}\backslash_{\circ} \}$, this means that only edges in $A_1 = \{ {}_{\circ}\!/^{\bullet}, {}_{\bullet}\!/^{\bullet}, {}_{\circ}\!/^{\circ}, {}^{\circ}\backslash_{\bullet} \}$ are allowed. The E_1-free black-white binary trees of size 3 and 4 are as many as lambda terms of size 3 and 4. They are listed in Fig. 3 and Fig. 4 second column. For $E_1 = \{ {}_{\bullet}\!/^{\circ}, {}^{\bullet}\backslash_{\circ}, {}^{\bullet}\backslash_{\bullet}, {}^{\circ}\backslash_{\circ} \}$, like for $E_2 = \{ {}_{\circ}\!/^{\bullet}, {}_{\bullet}\!/^{\bullet}, {}_{\circ}\!/^{\circ}, {}^{\circ}\backslash_{\bullet} \}$, which is obtained by left-right symmetry, the E-free black-white binary trees are counted by A105633 [4]. In what follows we will consider E_1 and we will rather speak in terms of an allowed set of pattern namely A_1. For simplicity, we will call in this paper *black-white trees*, the binary black-white trees with allowed pattern set A_1.

Before giving the bijection, let us give the trees corresponding to $\mathsf{K} = \lambda\lambda S(0)$, to $\mathsf{S} = \lambda\lambda\lambda(SS0\,0)\,(S0\,0)$, to $\omega = (\lambda(0\,0))\,\lambda(0\,0)$, and to $Y = \lambda(\lambda(S\,0\,(0\,0))\,\lambda(S\,0\,(0\,0)))$:

for K



for S



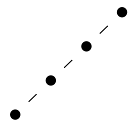
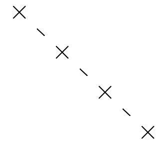
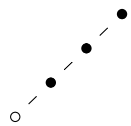
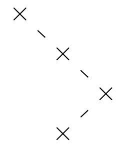
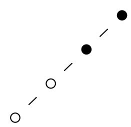
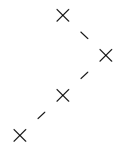
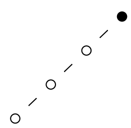
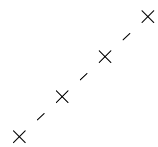
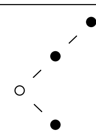
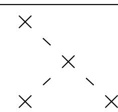
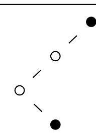
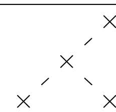
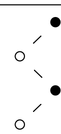
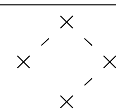
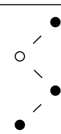
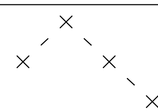
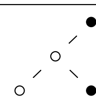
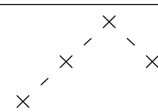
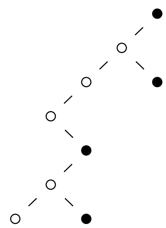
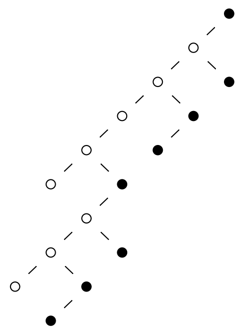
$\lambda -$ **terms**	**black-white trees**	**zigzag free trees**	**neutral hnf**
$S^3\mathtt{0}$			$S^4\mathtt{0}$
$\lambda S^2\mathtt{0}$			$\mathtt{0}\,(S^2\mathtt{0})$
$\lambda\lambda S\mathtt{0}$			$\mathtt{0}\,(\lambda S\mathtt{0})$
$\lambda\lambda\lambda\mathtt{0}$			$\mathtt{0}\,(\lambda\lambda\mathtt{0})$
$\mathtt{0}\,(S\mathtt{0})$			$(S\mathtt{0})\,(S\mathtt{0})$
$\mathtt{0}\,(\lambda\mathtt{0})$			$(S\mathtt{0})\,(\lambda\mathtt{0})$
$(\lambda\mathtt{0})\,\mathtt{0}$			$\mathtt{0}\,\mathtt{0}\,\mathtt{0}$
$(S\mathtt{0})\,\mathtt{0}$			$(S^2\mathtt{0})\,\mathtt{0}$
$\lambda(\mathtt{0}\,\mathtt{0})$			$\mathtt{0}\,(\mathtt{0}\,\mathtt{0})$

FIGURE 4. Bijection between λ-terms, E_1-free black-white binary trees and zigzag free trees of size 4 ($L_4 = 9$) and neutral head normal forms (Section 8) of size 5 ($K_5 = 9$).

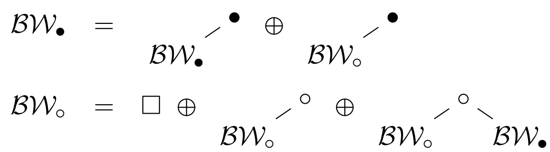
for ω



for Y



4.1. **Recursive description.** Assume $\Box$ is the empty tree which is usually not represented in drawing. The E_1-free black-white binary trees are described by the following combinatorial equation:



which yields the following functional equations:

$$\begin{aligned} BW_\bullet &= zBW_\bullet + zBW_\circ \\ BW_\circ &= 1 + zBW_\circ + zBW_\circ\, BW_\bullet \end{aligned}$$

hence

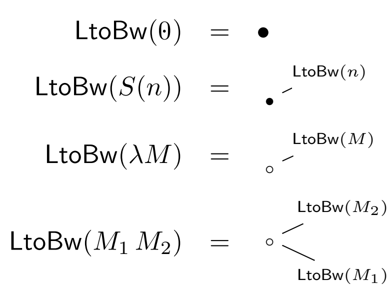
$$BW_\circ = \frac{1-z}{z}\, BW_\bullet$$

and

$$z(1-z)BW_\bullet^2 + (1-z)^2 BW_\bullet + z = 0.$$

which is the same equation up to a multiplication by $1-z$ as (1) namely the equation defining L_∞

4.2. **The bijection.** Let us define the function LtoBw from λ-terms to black-white trees:



In other words a new node is added on the leftmost node of the tree. from black-white trees to λ-terms Let us now define the function BwtoL

$$\begin{array}{rcl}
\mathsf{BwtoL}(\bullet) & = & \theta \\
\mathsf{BwtoL}\left(\begin{array}{ll} & {\scriptstyle T} \\ \bullet & \!\!\diagup \end{array}\right) & = & S(\mathsf{BwtoL}(T)) \\
\mathsf{BwtoL}\left(\begin{array}{ll} & {\scriptstyle T} \\ \circ & \!\!\diagup \end{array}\right) & = & \lambda\, \mathsf{BwtoL}(T) \\
\mathsf{BwtoL}\left(\begin{array}{ll} & {\scriptstyle T_2} \\ & \!\diagup \\ \circ & \\ & \!\diagdown \\ & {\scriptstyle T_1} \end{array}\right) & = & \mathsf{BwtoL}(T_1)\, \mathsf{BwtoL}(T_2)
\end{array}$$

In other words, to decompose a binary tree which is not the node $\bullet$, we look for the left most node.

- If the leftmost node is $\bullet$, then the λ-term is a de Bruijn index. Actually there are only $\bullet$'s (indeed $\begin{array}{ll} & \circ \\ \bullet & \!\!\diagup \end{array}$ is forbidden) and the tree is linear. If this linear tree has n $\bullet$'s it represents $S^{n-1}(\theta)$.
- If the leftmost node is $\circ$ and has no child, then the λ-term is an abstraction of the bijection of the rest.
- If the leftmost node is $\circ$ and has a right child, then the λ-term is an application of the bijection of the right subtree on the bijection of the above tree .

Proposition 1. *$\mathsf{LtoBw} \circ \mathsf{BwtoL} = id_\Lambda$ and $\mathsf{BwtoL} \circ \mathsf{LtoBw} = id_{\mathcal{BW}_\bullet}$*

4.3. **The bijection in Haskell.** In this section we describe Haskell programs for the bijections. First we define black-white trees. We consider three kinds of trees: leafs (of arity zero and size zero) corresponding to $\Box$ and not represented in drawing.

$\boxed{\textit{Haskell program}}$

```
-- DeBruijn index datatype.
data DeBruijn = S DeBruijn
              | Z

-- Lambda-term datatype.
data LTerm = App LTerm LTerm
           | Abs LTerm
           | Nat DeBruijn

-- Black-White binary tree datatype.
data BWTree = Black BWTree BWTree
            | White BWTree BWTree
            | Leaf

-- Substitutes the given Black-White tree bwt
-- for the leftmost node in the second tree.
sub :: BWTree -> BWTree -> BWTree
sub bwt (Black t t') = Black (bwt `sub` t) t'
sub bwt (White t t') = White (bwt `sub` t) t'
```

```
sub bwt Leaf = bwt

-- Translates the given DeBruijn index
-- to a corresponding Black-White tree.
dToBw :: DeBruijn -> BWTree
dToBw Z = Black Leaf Leaf
dToBw (S n) = Black Leaf Leaf `sub` dToBw n

-- Translates the given Lambda-term to
-- a corresponding Black-White tree.
lToBw :: LTerm -> BWTree
lToBw (Nat n) = dToBw n
lToBw (Abs e) = White Leaf Leaf `sub` lToBw e
lToBw (App e e') = White Leaf (lToBw e) `sub` lToBw e'

-- Cuts the leftmost subtree out from the given tree
-- returning a pair (leftmost subtree, pruned tree).
prune :: BWTree -> (BWTree, BWTree)
prune p @ (White Leaf _) = (p, Leaf)
prune (White l r) = case prune l of
    (lm, p) -> (lm, White p r)
prune p @ (Black Leaf _) = (p, Leaf)
prune (Black l r) = case prune l of
    (lm, p) -> (lm, Black p r)

-- Translates the given black rooted Black-White
-- tree to a corresponding DeBruijn index.
bToD :: BWTree -> DeBruijn
bToD (Black Leaf Leaf) = Z
bToD (Black t Leaf) = S $ bToD t

-- Translates the given Black-White
-- tree to a corresponding Lambda-term.
bwToL :: BWTree -> LTerm
bwToL bwt = case prune bwt of
    (Black Leaf Leaf, pt) -> Nat $ bToD bwt
    (White Leaf Leaf, pt) -> Abs $ bwToL pt
    (White Leaf t, pt) -> App (bwToL t) (bwToL pt)
```

End of Haskell program

In order to translate λ-terms to corresponding black-white trees we carry out a rather unusual induction, where after the recursive step we attach a new subtree to the leftmost node in one of the previously obtained black-white trees. Similarly, in the inverse translation from black-white trees to λ-terms, we have to cut out the leftmost node of the current black-white tree and pattern match against the result. This unusual recursion is a result of our natural top-down representation of black-white trees, where children are drawn below their parents. Note that if we change this convention so that children are drawn on the right to their parents, the previously leftmost node becomes the root of the black-white tree. The data type for black-white trees does not change, but instead of top-down trees, we are working with left-right ones. Such a representation simplifies the overall implementation as the algorithm is no longer required to look for the leftmost node.

5. Binary trees without zigzags

5.1. Non empty zigzag free binary trees. Consider $\mathcal{BZ}_1$ the set of binary trees with no zigzag i.e., with no subtree like

$$\begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ \times & & & & \\ & \backslash & & & \\ & & \times & & \\ & / & & \backslash & \end{array}$$

$\mathcal{BZ}_1$ is described by the combinatorial equations:

$$\begin{array}{rcl} \mathcal{BZ}_1 & = & \begin{array}{cc} \times & \\ & \backslash \; \mathcal{BZ}_1 \end{array} \;\oplus\; \mathcal{BZ}_2 \\ \mathcal{BZ}_2 & = & \times \;\oplus\; \begin{array}{cc} & \times \\ \mathcal{BZ}_2 \; / & \end{array} \;\oplus\; \begin{array}{ccc} & \times & \\ \mathcal{BZ}_2 \; / & & \backslash \; \mathcal{BZ}_1 \end{array} \end{array}$$

Like L_∞ and $BW_\bullet$, BZ_1 is solution of the functional equation:

$$z(1-z)BZ_1^2 + (1-z)^2 BZ_1 + z = 0.$$

5.2. A formula. Sapounakis et al. [7] consider a similar sequence defined in term of avoiding Dyck paths and give the formula:

$$[z^n]BZ_1 = [z^n]L_\infty = \sum_{k=0}^{(n-1)\div 2} \frac{(-1)^k}{n-k} \binom{n-k}{k} \binom{2n-3k}{n-2k-1}$$

6. The bijections between black white trees and zigzag free trees

6.1. From black white trees to zigzag free trees. Let us call BwToBz the bijection from black white trees to zigzag free trees. Notice that the fourth equation removes a $\bullet$ and the last equation adds a $\times$, keeping a balance between $\bullet$ nodes and $\times$ nodes on the leftmost branch.

$$\begin{aligned}
\mathsf{BwToBz}(\square) &= \square\\
\mathsf{BwToBz}(\bullet) &= \times\\
\mathsf{BwToBz}\left(\begin{array}{ccc} & & \bullet \\ & / & \\ t & & \end{array}\right) &= \begin{array}{ccc}\times & & \\ & \backslash & \\ & & \mathsf{BwToBz}(t)\end{array} \qquad \text{when } t = \begin{array}{ccc} & & \bullet\\ & / & \\ u & & \end{array}\\
\mathsf{BwToBz}\left(\begin{array}{ccc} & & \bullet \\ & / & \\ t & & \end{array}\right) &= \mathsf{BwToBz}(t) \qquad \text{when } t = \begin{array}{ccc} & & \circ\\ & / & \\ u & & \end{array}\\
\mathsf{BwToBz}\left(\begin{array}{ccccc} & & \circ & & \\ & / & & \backslash & \\ t & & & & t' \end{array}\right) &= \begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ \mathsf{BwToBz}(t) & & & & \mathsf{BwToBz}(t') \end{array} \qquad \text{when } t = \begin{array}{ccccc} & & \circ & & \\ & / & & \backslash & \\ u_1 & & & & u_2 \end{array}\\
\mathsf{BwToBz}\left(\begin{array}{ccc} \circ & & \\ & \backslash & \\ & & t \end{array}\right) &= \begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ \times & & & & \mathsf{BwToBz}(t) \end{array}
\end{aligned}$$

6.2. **From zigzag free trees to black white trees.** We use two functions $\mathsf{BzToBw}_\bullet$ and $\mathsf{BzToBw}_\circ$. Notice also that on the leftmost branch a $\bullet$ is added and a $\times$ is removed;

$$\begin{aligned}
\mathsf{BzToBw}_\bullet(\square) &= \square\\
\mathsf{BzToBw}_\bullet(\times) &= \bullet\\
\mathsf{BzToBw}_\bullet\left(\begin{array}{ccc} \times & & \\ & \backslash & \\ & & t \end{array}\right) &= \begin{array}{ccc} & & \bullet \\ & / & \\ \mathsf{BzToBw}_\bullet(t) & & \end{array} \qquad \text{when } t = \begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ u_1 & & & & u_2 \end{array}\\
\mathsf{BzToBw}_\bullet\left(\begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ t & & & & t' \end{array}\right) &= \begin{array}{ccccccc} & & & & & & \bullet \\ & & & & & / & \\ & & \circ & & & & \\ & / & & \backslash & & & \\ \mathsf{BzToBw}_\circ(t) & & & & \mathsf{BzToBw}_\bullet(t') & & \end{array} \qquad \text{when } t = \begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ u_1 & & & & u_2 \end{array}\\
\mathsf{BzToBw}_\circ(\times) &= \square\\
\mathsf{BzToBw}_\circ\left(\begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ t & & & & t' \end{array}\right) &= \begin{array}{ccccc} & & \circ & & \\ & / & & \backslash & \\ \mathsf{BzToBw}_\circ(t) & & & & \mathsf{BzToBw}_\bullet(t') \end{array} \qquad \text{when } t = \begin{array}{ccccc} & & \times & & \\ & / & & \backslash & \\ u_1 & & & & u_2 \end{array}
\end{aligned}$$

Proposition 2. $\mathsf{BzToBw}_\bullet \circ \mathsf{BwToBz} = id_{\mathcal{BW}_\bullet}$ *and* $\mathsf{BwToBz} \circ \mathsf{BzToBw}_\bullet = id_{\mathcal{BZ}}$.

6.3. **Haskell code.**

Haskell program

```
-- Black-White binary tree datatype.
data BWTree = Black BWTree BWTree
            | White BWTree BWTree
            | BWLeaf

-- Zigzag free tree datatype.
data BZTree = Node BZTree BZTree
            | BZLeaf

-- Useful shorthand.
blNode :: BZTree
```

```
blNode = Node BZLeaf BZLeaf

-- Translates the given Black-White tree
-- to a corresponding Zigzag free tree.
bwToBz :: BWTree -> BZTree
bwToBz BWLeaf = BZLeaf
bwToBz (Black BWLeaf BWLeaf) = blNode
bwToBz (Black t @ (Black _ BWLeaf) BWLeaf) = Node BZLeaf $ bwToBz t
bwToBz (Black t @ (White _ BWLeaf) BWLeaf) = bwToBz t
bwToBz (White t @ (White _ _) t') = Node (bwToBz t) (bwToBz t')
bwToBz (White BWLeaf t) = Node blNode (bwToBz t)

-- Translates the given Zigzag free tree to a
-- corresponding black rooted Black-White tree.
bzToBwB :: BZTree -> BWTree
bzToBwB BZLeaf = BWLeaf
bzToBwB (Node BZLeaf BZLeaf) = Black BWLeaf BWLeaf
bzToBwB (Node BZLeaf t @ (Node _ _)) = Black (bzToBwB t) BWLeaf
bzToBwB (Node t @ (Node _ _) t') = Black u BWLeaf
    where
        u = White (bzToBwW t) (bzToBwB t')

-- Translates the given Zigzag free tree to a
-- corresponding white rooted Black-White tree.
bzToBwW :: BZTree -> BWTree
bzToBwW (Node BZLeaf BZLeaf) = BWLeaf
bzToBwW (Node t @ (Node _ _) t') = White (bzToBwW t) (bzToBwB t')
```

End of Haskell program

7. The bijections between lambda terms and zigzag free trees

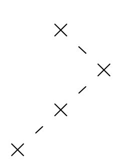
7.1. **From lambda terms to zigzag free trees.** Lest us call LToBz this bijection. It is described in Figure 5

7.2. **From zigzag free terms to lambda terms.** The bijection called BzToL is defined in Figure 6.

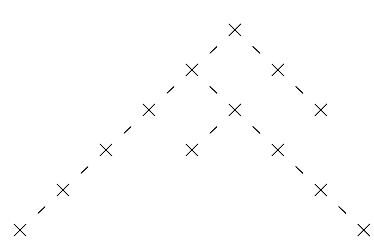
Proposition 3. $\mathsf{LToBz} \circ \mathsf{BzToL} = id_{\mathcal{BZ}}$ *and* $\mathsf{BzToL} \circ \mathsf{LToBz} = id_{\Lambda}$.

7.3. **Examples.** Let us look at the bijection on classical examples, namely K, S, ω and Y:

for K



for S



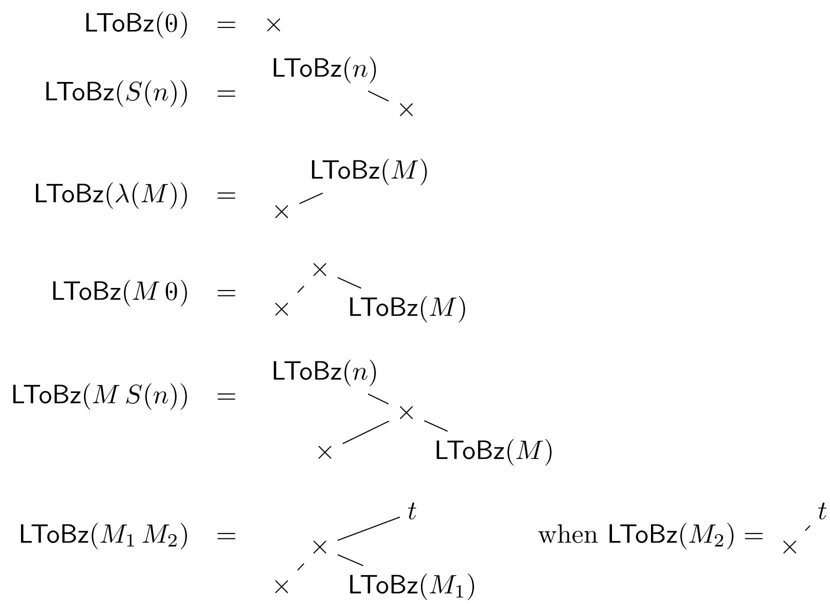
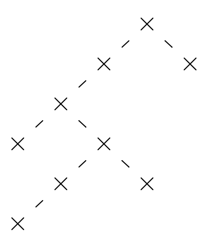
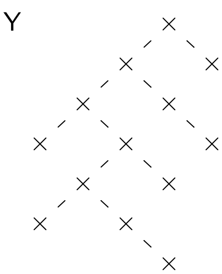


FIGURE 5. The bijection LToBz from lambda terms to zigzag free trees

for ω



for Y



7.4. **Haskell code.**

Haskell program

```
-- DeBruijn index datatype.
data DeBruijn = S DeBruijn
              | Z

-- Lambda-term datatype.
data LTerm = App LTerm LTerm
           | Abs LTerm
           | Nat DeBruijn

-- Zigzag free tree datatype.
data BZTree = Node BZTree BZTree
            | BZLeaf
```

$$
\begin{aligned}
\mathsf{BzToL}(\times) &= \theta \\
\mathsf{BzToL}\left(\begin{array}{ccc} n & & \\ & \diagdown & \\ & & \times \end{array}\right) &= S(\mathsf{BzToL}\,(n)) \\
\mathsf{BzToL}\left(\begin{array}{ccc} & & \times \\ & \diagup & \\ \times & & \end{array}\right) &= \lambda\theta \\
\mathsf{BzToL}\left(\begin{array}{ccccc} & & \times & & \\ & \diagup & & \diagdown & \\ \times & & & & T \end{array}\right) &= \mathsf{BzToL}(T)\,\theta \\
\mathsf{BzToL}\left(\begin{array}{ccc} n & & \\ & \diagdown & \\ & & \times \\ & \diagup & \\ \times & & \end{array}\right) &= \lambda\,\mathsf{BzToL}\left(\begin{array}{ccc} n & & \\ & \diagdown & \\ & & \times \end{array}\right) \\
\mathsf{BzToL}\left(\begin{array}{ccccc} n & & & & \\ & \diagdown & & & \\ & & \times & & \\ & \diagup & & \diagdown & \\ \times & & & & T \end{array}\right) &= \mathsf{BzToL}(T)\ \lambda\,\mathsf{BzToL}\left(\begin{array}{ccc} n & & \\ & \diagdown & \\ & & \times \end{array}\right) \\
\mathsf{BzToL}\left(\begin{array}{ccccc} & & & & T \\ & & & \diagup & \\ & & \times & & \\ & \diagup & & & \\ \times & & & & \end{array}\right) &= \lambda\,\mathsf{BzToL}\left(\begin{array}{ccc} & & T \\ & \diagup & \\ \times & & \end{array}\right) \\
\mathsf{BzToL}\left(\begin{array}{ccccc} & & & & T_2 \\ & & & \diagup & \\ & & \times & & \\ & \diagup & & \diagdown & \\ \times & & & & T_1 \end{array}\right) &= \mathsf{BzToL}(T_1)\ \mathsf{BzToL}\left(\begin{array}{ccc} & & T_2 \\ & \diagup & \\ \times & & \end{array}\right)
\end{aligned}
$$

FIGURE 6. The bijection BzToL

```
-- Useful shorthand.
blNode :: BZTree
blNode = Node BZLeaf BZLeaf

-- Substitutes the given Zigzag free tree zg
-- for the leftmost node in the second tree.
subL :: BZTree -> BZTree -> BZTree
subL zg (Node t t') = Node (zg `subL` t) t
subL zg BZLeaf = zg

-- Substitutes the given Zigzag free tree zg
-- for the rightmost node in the second tree.
subR :: BZTree -> BZTree -> BZTree
subR zg (Node t t') = Node t (zg `subR` t')
subR zg BZLeaf = zg

-- Cuts the leftmost subtree out from the given tree
```

```
-- returning a pair (leftmost subtree, pruned tree).
prune :: BZTree -> (BZTree, BZTree)
prune p @ (Node BZLeaf _) = (p, BZLeaf)
prune (Node l r) = case prune l of
    (lm, p) -> (lm, Node p r)

-- Translates the given DeBruijn index
-- to a corresponding Zigzag free tree.
dToBz :: DeBruijn -> BZTree
dToBz Z = Node BZLeaf BZLeaf
dToBz (S n) = blNode `subR` dToBz n

-- Translates the given Lambda-term
-- to a corresponding Zigzag free tree.
lToBz :: LTerm -> BZTree
lToBz (Nat n) = dToBz n
lToBz (Abs e) = blNode `subL` lToBz e
lToBz (App e (Nat k)) = case k of
    Z -> Node blNode $ lToBz e
    S n -> Node blNode (lToBz e) `subR` dToBz n
lToBz (App e e') = case prune $ lToBz e' of
    (Node BZLeaf BZLeaf, t) -> Node blNode (lToBz e) `subL` t
```

End of Haskell program

We leave the straightforward implementation of BzToL from λ-terms to Zigzag free trees to the reader.

8. Normal forms

We are now interested in normal forms, that are terms irreducible by β reduction that are also terms which do not have subterms of the form $(\lambda \mathsf{M})\,\mathsf{N}$.

There are three associated classes: $\mathcal{N}$ (the normal forms), $\mathcal{M}$ (the neutral terms, which are the normal forms without head abstractions) and $\mathcal{D}$ (the de Bruijn indices) :

$$\begin{aligned} \mathcal{N} &= \mathcal{M} + \lambda\mathcal{N} \\ \mathcal{M} &= \mathcal{M}\mathcal{N} + \mathcal{D} \\ \mathcal{D} &= S\mathcal{D} + \mathtt{0}. \end{aligned}$$

Let us call N the generating function of $\mathcal{N}$, M the generating function for $\mathcal{M}$ and D the generating function for $\mathcal{D}$. The above equations yield the equations for the generating functions:

$$\begin{aligned} N &= M + zN \\ M &= zMN + D \\ D &= zD + z \end{aligned}$$

One shows that

$$
\begin{aligned}
M &= \frac{1 - z - \sqrt{(1+z)(1-3z)}}{2z} \\
N &= \frac{M}{1-z}
\end{aligned}
$$

M is the generating function of Motzkin trees (see [2] p. 396).

9. The bijections between Motzkin trees and neutral normal forms

In this section we exhibit a bijection between Motzkin trees and neutral normal forms as suggested by the identity between their genrating functions. Let u_n denote the unary Motzkin path of height n. We start with defining two auxiliary operations UnToL and UnToD, translating unary Motzkin paths into λ-paths and DeBruijn indices, respectively.

$$
\begin{aligned}
\mathsf{UnToL}\,(\bullet) &= \lambda \\
\mathsf{UnToL}\left(\begin{array}{c} \bullet \\ \downarrow \\ u_n \end{array}\right) &= \begin{array}{c} \lambda \\ \downarrow \\ \mathsf{UnToL}\,(u_n) \end{array}
\end{aligned}
$$

Figure 7. Operation UnToL

$$
\begin{aligned}
\mathsf{UnToD}\,(\bullet) &= 0 \\
\mathsf{UnToD}\left(\begin{array}{c} \bullet \\ \downarrow \\ u_n \end{array}\right) &= \begin{array}{c} S \\ \downarrow \\ \mathsf{UnToD}\,(u_n) \end{array}
\end{aligned}
$$

Figure 8. Operation UnToD

Using UnToL and UnToD we can now define (Figure 9) the translation MoToNe from Motzkin trees into corresponding neutral terms.

$$
\mathsf{MoToNe}\left(\begin{tikzcd} & u_n \arrow[d] & \\ & \bullet \arrow[dl] \arrow[dr] & \\ t & & t' \end{tikzcd}\right) = \begin{tikzcd} & @ \arrow[dl] \arrow[dr] & \\ \mathsf{MoToNe}\,(t) & & \mathsf{UnToL}\,(u_n) \arrow[d] \\ & & \mathsf{MoToNe}\,(t') \end{tikzcd}
$$

$$
\mathsf{MoToNe}\left(\begin{tikzcd} & \bullet \arrow[dl] \arrow[dr] & \\ t & & t' \end{tikzcd}\right) = \begin{tikzcd} & @ \arrow[dl] \arrow[dr] & \\ \mathsf{MoToNe}\,(t) & & \mathsf{MoToNe}\,(t') \end{tikzcd}
$$

$$
\mathsf{MoToNe}\,(u_n) = \mathsf{UnToD}\,(u_n)
$$

Figure 9. Translation MoToNe

Proposition 4. MoToNe *is injective.*

Proof. The proposition is an easy consequence of the fact that MoToNe preserves the exact number of unary and binary nodes. □

What remains is to give the inverse translation NeToMo from neutral terms to Motzkin trees (Figure 10). Let LToUn and DToUn denote the inverse functions of UnToL and UnToD respectively. Let l_n denote the unary λ-path of height n and d_n denote the n-th DeBruijn index. The translation NeToMo is given by:

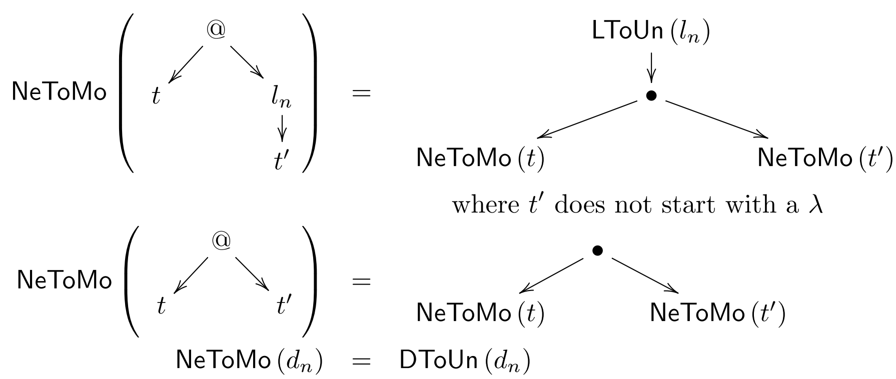


FIGURE 10. Translation NeToMo

Proposition 5. MoToNe $\circ$ NeToMo $= id_{\mathcal{M}}$ *and* NeToMo $\circ$ MoToNe $= id_{\mathcal{T}}$.

9.1. **Haskell code.**

Haskell program

```
-- DeBruijn index datatype.
data DeBruijn = S DeBruijn
              | Z

-- Motzkin tree datatype.
data MotzkinTree = BNode MotzkinTree MotzkinTree
                 | UNode MotzkinTree
                 | Leaf

-- Normal form datatype.
data NormalF = Lambda NormalF
             | NF NeutralT

-- Neutral term datatype.
data NeutralT = App NeutralT NormalF
              | Nat DeBruijn

-- Transforms the given unary Motzkin path
-- into a corresponding DeBruijn index.
unToD :: MotzkinTree -> DeBruijn
```

```
unToD (UNode mt) = S $ unToD mt
unToD Leaf = Z

-- Transforms the given unary Motzkin path into
-- a chain of lambda abstractions attached
-- to the root of the given normal form.
pushL :: MotzkinTree -> NormalF -> NormalF
pushL (UNode mt) lt = pushL mt $ Lambda lt
pushL (BNode _ _) lt = lt

-- Finds the splitting node of the given Motzkin tree.
splittingNode :: MotzkinTree -> Maybe MotzkinTree
splittingNode (UNode mt) = splittingNode' mt
    where
        splittingNode' Leaf = Nothing
        splittingNode' sn @ (BNode _ _) = Just sn
        splittingNode' (UNode mt) = splittingNode' mt

-- Syntactic type sugar.
moToNf :: MotzkinTree -> NormalF
moToNf = NF . moToNe

-- Translates the given Motzkin tree
-- to a corresponding neutral term.
moToNe :: MotzkinTree -> NeutralT
moToNe Leaf = Nat Z
moToNe (BNode l r) = App (moToNe l) (moToNf r)
moToNe root = case splittingNode root of
    Nothing -> Nat $ unToD root
    Just sn -> case moToNe sn of
         App lt rt -> App lt $ pushL root rt
```

End of Haskell program

In order to translate Motzkin trees to corresponding neutral terms we have to consider two cases. Either we are given a Motzkin tree starting with a unary node or a binary one. The later case is straightforward due to the fact that binary nodes correspond to neutral term application. Assume we are given a Motzkin tree starting with a unary path u_n of size n. We have to decide whether the path corresponds to a DeBruijn index or a chain of λ-abstractions. This distinction is uniquely determined by the existence of the path's *splitting node* – the binary node directly below u_n. If u_n has a splitting node then it corresponds to a chain of n λ-abstractions which will be placed on top of the corresponding right neutral term constructed recursively from u_n's splitting node. Otherwise, u_n corresponds to the n-th DeBruijn index.

We leave the straightforward implementation of NeToMo from neutral terms to Motzkin trees to the reader.

10. Head normal forms

We are now interested in the set of head normal forms

$$\begin{aligned}\mathcal{H} &= \mathcal{K} + \lambda\mathcal{H}\\ \mathcal{K} &= \mathcal{K}\mathcal{L}_\infty + \mathcal{D}\end{aligned}$$

which yields the equations

$$\begin{aligned}H &= K + zH\\ K &= zKL_\infty + D\end{aligned}$$

and

$$\begin{aligned}K &= \frac{D}{1 - zL_\infty}\\ H &= \frac{K}{1-z}\end{aligned}$$

From which we draw

$$K = z + zL_\infty.$$

This can be explained by the following bijection (see Figure 3 and Figure 4):

Proposition 6. *If P is a neutral head normal form, it is of the form:*

- $P = 0\, N_1 N_2 \ldots N_p$ *with* $p \geq 1$ *(of size $k+1$) then it is in bijection with* $(\lambda\, N_1)N_2 \ldots N_p$ *(of size k),*
- $P = (Sn)\, N_1 \ldots N_p$ *(of size $k+1$) then it is in bijection with* $n\, N_1 \ldots N_p$ *(of size k),*
- $P = 0$ *(of size 1), treated by the case z.*

From Theorem 1 we get:

Proposition 7.

$$[z^{n+1}]K \sim \left(\frac{1}{\rho_{L_\infty}}\right)^n \frac{C}{n^{\frac{3}{2}}}$$

with $C \doteq 0.60676...$ *and* $\rho_{L_\infty} \doteq 0.29559....$

The density of a set $\mathcal{A}$ in a set $\mathcal{B}$ is

$$\lim_{n\to\infty} \frac{A_n}{B_n}$$

where A_n (respectively B_n) are the numbers of elements of $\mathcal{A}$ (respectively of $\mathcal{B}$) of size n. For instance the density of $\mathcal{K}$ in $\mathcal{L}_\infty$ is

$$\lim_{n\to\infty} \frac{[z^n]K}{[z^n]L_\infty};$$

Hence the proposition.

Proposition 8. *The density of $\mathcal{K}$ in $\mathcal{L}_\infty$ (i.e., the density of neutral head normal forms among plain terms) is ρ_{L_∞}.*

Proposition 9.

$$[z^n]H \sim \left(\frac{1}{\rho_{L_\infty}}\right)^n \frac{C_H}{n^{\frac{3}{2}}}$$

with $C_H \doteq 0.254625911836762946...$

Proof. The proof is like this of Theorem 1 with

$$C_H = \frac{-\sqrt{\rho_{L_\infty} \frac{Q(\rho_{L_\infty})}{1-\rho_{L_\infty}}}}{2(1-\rho_{L_\infty})\Gamma(-\frac{1}{2})} \doteq 0.254625911836762946...$$

□

Figure 11 compares the coefficients of H with its approximation.

n	$[\mathbf{x^n}]\mathbf{H}$
10	1902
20	118768916
30	12338289374047
40	1552505356757052270
50	216408050593408223194666
60	32156818736630052190010494575
70	4992016749940033843389032870415375
80	800041142163881275363093897487465240590
90	131362728872240507612558556757894820073668254
100	21984069003048322712483528437236630547685953755064
n	$\lfloor(\mathbf{1}/\rho_{\mathbf{L}_\infty})^{\mathbf{n}}\mathbf{C_H}/\mathbf{n^{3/2}}\rfloor$
10	1581
20	109732518
30	11727010776119
40	1495436887319673848
50	210083497584679365571791
60	31376820974748144171493861802
70	4888522574435898663355075650509052
80	785558576073780985739070920824898277393
90	129252413184969184232722751628403772087829182
100	21666626365243195881127917362969390314273901016408

FIGURE 11. Approximation of $[x^n]H$.

Proposition 10. *The density of $\mathcal{H}$ in $\mathcal{L}_\infty$ (i.e., the density of head normal forms among plain terms) is $\rho_{L_\infty}/(1-\rho_{L_\infty}) \doteq 0.41964337760707887...$*

Proof. Actually $\frac{C_H}{C} = \frac{\rho_{L_\infty}}{(1-\rho_{L_\infty})}$. □

11. Terms containing specific subterms

Consider a term M of size p and the set $\mathcal{T}$ of terms that contain M as subterm.

$$\mathcal{T} \quad = \quad t + \lambda\mathcal{T} + \mathcal{T}\mathcal{L}_\infty + \mathcal{L}_\infty\mathcal{T} - \mathcal{T}\mathcal{T}$$

which yields

$$T = z^p + zT + 2zTL_\infty - zT^2$$

and

$$zT^2 + (1 - 2zL_\infty - z)T - z^p \quad = \quad 0.$$

Notice that

$$1 - 2zL_\infty - z \quad = \quad \sqrt{\Delta_{L_\infty}}$$

Then the discriminant is

$$\begin{aligned} \Delta_T &= \Delta_{L_\infty} + 4z^{p+1} \\ (1-z)\Delta_T &= (1-z)\Delta_{L_\infty} + 4z^{p+1}(1-z). \end{aligned}$$

In the interval $(0,1)$, Δ_∞ is decreasing (its derivative is negative) and $(1-z)\Delta_T > (1-z)\Delta_{L_\infty}$. Hence the root ρ_T of Δ_T is larger than the root ρ_{L_∞} of Δ_∞, that is $\rho_T > \rho_{L_\infty}$. Beside:

$$T = \frac{\sqrt{\Delta_T} - \sqrt{\Delta_{L_\infty}}}{2z}.$$

Hence the number of terms that do not have M as subterm is given by

$$L_\infty - T = \frac{(1-z) - \sqrt{\Delta_T}}{2z}.$$

Theorem 2. *The density in $\mathcal{L}_\infty$ of terms that do not have M as subterm is 0.*

Proof. Indeed the smallest pole of $L_\infty - T$ is ρ_T and the smallest pole of L_∞ is ρ_{L_∞}. Therefore,

$$\begin{aligned} [z^n](L_\infty - T) &\bowtie \left(\frac{1}{\rho_T}\right)^n \\ [z^n]L_\infty &\bowtie \left(\frac{1}{\rho_{L_\infty}}\right)^n \end{aligned}$$

Hence, since $\rho_T > \rho_{L_\infty}$

$$\lim_{n\to\infty} \frac{[z^n](L_\infty - T)}{[z^n]L_\infty} = \left(\frac{\rho_{L_\infty}}{\rho_T}\right)^n = 0.$$

□

For instance if $|t| = 9$, that is for instance if $t = \omega = (\lambda(\theta\,\theta))\,\lambda(\theta\,\theta)$, then

$$\rho_T \doteq 0.2956014673597697$$

and

$$\frac{\rho_{L_\infty}}{\rho_T} \doteq 0.9999873991231537.$$

Corollary 1. *The density in $\mathcal{L}_\infty$ of terms that contain M as subterm is 1.*

Corollary 2. *Asymptotically almost no λ-term is strongly normalizing.*

Proof. In other words, *the density of strongly normalizing terms is* 0. Indeed, the density in $\mathcal{L}_\infty$ of terms that contain $(\lambda(\theta\,\theta))\,\lambda(\theta\,\theta)$ is 1. Hence the density of non strongly normalizing terms is 1. Hence the density of strongly normalizing terms is 0. □

12. Conclusion

Figure 12 summarizes what we obtained on densities of terms.

Moreover, this research opens many issues, among others about generating random terms and random normal forms using Boltzmann samplers [5].

nf	nhdnf		terms with M
sn		hdnf	$\overline{\mathsf{sn}}$
0	0.295...	0.419...	1

nf = normal forms

nhdnf = neutral head normal forms $\quad$ hdnf = head normal forms

$\mathsf{terms\ with\ M}$ = terms containing subterm M

sn = strongly normalizing terms $\quad$ $\overline{\mathsf{sn}}$ = non strongly normalizing terms

Figure 12. Summary of densities

References

[1] Nicolas Bourbaki. *Theory of Sets*. Elements of Mathematics. Springer-Verlag, 2004.

[2] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[3] Katarzyna Grygiel and Pierre Lescanne. Counting terms in the binary lambda calculus. *CoRR*, abs/1401.0379, 2014. Published in the Proceedings of *25th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms* 2014, `https://hal.inria.fr/hal-01077251`.

[4] Nancy S. S. Gu, Nelson Y. Li, and Toufik Mansour. 2-binary trees: Bijections and related issues. *Discrete Mathematics*, 308(7):1209–1221, 2008.

[5] Pierre Lescanne. Boltzmann samplers for random generation of lambda terms. Technical report, ENS de Lyon, 2014. `http://hal-ens-lyon.archives-ouvertes.fr/docs/00/97/90/74/PDF/boltzmann.pdf`.

[6] Bruno Salvy and Paul Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[7] Aristidis Sapounakis, Ioannis Tasoulas, and Panagiotis Tsikouras. Ordered trees and the in-order traversal. *Discrete Mathematics*, 306(15):1732–1741, 2006.

Appendix A. De Bruijn notations

De Bruijn indices are a system of notations for bound variables due to Nikolaas de Bruijn and somewhat connected to those proposed by Bourbaki [1]. The goal is to replace bound variables by placeholders and to link each bound variable to its binder. For instance (see Figure 13) Bourbaki ([1] p. 20) proposes to represent placeholders by boxes $\Box$ and to represent binds by drawn lines. This requires a two dimensional notation. For example, he considers the formula:

$$(\tau x)\,\neg(x \in A') \vee x \in A''$$

Notice that we use an infix notation whereas he uses a prefix notation which gives $\tau \vee \neg \in xA' \in xA$. The formula contains the binder τ (a binder that Bourbaki introduces) and two occurrences of the bound variable x, this involves two $\Box$'s and two drawn lines from τ, namely to the first $\Box$ and to the second $\Box$. De Bruijn proposes to represent the placeholders (in other words the variables) by natural numbers which represent the length of the link, that is the number of binders crossed when reaching the actual binder of the variables. In our proposal, we write natural numbers using the functions *zero* $\underline{0}$ and *successor* S. For instance, 3 is written $SSS\underline{0}$. With de Bruijn notations, Bourbaki's formula is written:

$$\tau\,(\neg\underline{0} \in A') \vee \underline{0} \in A''$$

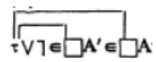


FIGURE 13. Bourbaki's notations for formula $\tau \vee \neg \in xA' \in xA$.

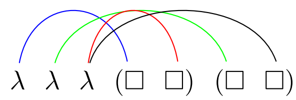


FIGURE 14. S in Bourbaki style

and the lambda terms $\lambda x.\lambda y.\lambda z.(xz)(yz)$ is written $\lambda\lambda\lambda(((SS\mathtt{0})\mathtt{0})((S\mathtt{0})\mathtt{0}))$ which would correspond to the drawing of Figure 14 in Bourbaki style.

APPENDIX B. ANOTHER NATURAL COUNTING OF LAMBDA TERMS

Another natural counting is a counting where:

$$\begin{aligned} |\lambda M| &= |M| + 1 \\ |M_1\, M_2| &= |M_1| + |M_2| + 1 \\ |Sn| &= |n| + 1 \\ |\mathtt{0}| &= 0. \end{aligned}$$

The generating function is solution of

$$zM_\infty^2 - (1-z)M_\infty + \frac{1}{1-z} = 0$$

with discriminant

$$\begin{aligned} \Delta_{M_\infty} &= (1-z)^2 - 4\frac{z}{1-z} \\ &= \frac{(1-z)^3 - 4z}{1-z} \\ &= \frac{1 - 7z + 3z^2 - z^3}{1-z} \end{aligned}$$

and with root closest to 0: $\rho_{M_\infty} \doteq 0.152292401860433$ and $1/\rho_{M_\infty} = 6.5663157700831193$. The first values are:

1, 3, 10, 40, 181, 884, 4539, 24142, 131821, 734577, 4160626

This sequence is **A258973** in the *Online Encyclopedia of Integer Sequences* and grows significantly faster than **A105633**.

E-mail address: GRYGIEL@TCS.UJ.EDU.PL,PIERRE.LESCANNE@ENS-LYON.FR