# QUANTITATIVE ASPECTS OF LINEAR AND AFFINE CLOSED LAMBDA TERMS

PIERRE LESCANNE

ABSTRACT. Affine $\lambda$-terms are $\lambda$-terms in which each bound variable occurs at most once and linear $\lambda$-terms are $\lambda$-terms in which each bound variables occurs once. and only once. In this paper we count the number of closed affine $\lambda$-terms of size $n$, closed linear $\lambda$-terms of size $n$, affine $\beta$-normal forms of size $n$ and linear $\beta$-normal forms of sizese $n$, for different ways of measuring the size of $\lambda$-terms. From these formulas, we show how we can derive programs for generating all the terms of size $n$ for each class. The foundation of all of this is specific data structures, which are contexts in which one counts all the holes at each level of abstractions by $\lambda$'s.

**Keywords:**

## 1. INTRODUCTION

The $\lambda$-calculus [1] is a well known formal system designed by Alonzo Church [7] for studying the concept of function. It has three kinds of basic operations: variables, application and abstraction (with an operator $\lambda$ which is a binder of variables). We assume the reader familiar with the $\lambda$-calculus and with de Bruijn indices.[1]

In this paper we are interested in terms in which bound variables occur once. A *closed $\lambda$-term* is a $\lambda$-term in which there is no free variable. An *affine $\lambda$-term* is a $\lambda-$term in which bound variables occur at most once. A *linear* $\lambda$-term is a $\lambda$-term in which bound variables occur once and only once

In this paper we propose a method for counting and generating (including random generation) linear and affine closed $\lambda$-terms based on a data structure which we call *SwissCheese* because of its holes. Actually we count those $\lambda$-terms up-to $\alpha$-conversion. Therefore it is adequate to use de Bruijn indices [9], because a term with de Bruijn indices represents an $\alpha$-equivalence class. An interesting aspect of these terms is the fact that they are simply typed [14, 13]. For instance, generated by the program of Section 5, there are 16 linear terms of natural size 8:

$$(\lambda 0\ (\lambda 0\ \lambda 0))\quad (\lambda 0\ \lambda(\lambda 0\ 0))\quad (\lambda 0\ \lambda(0\ \lambda 0))\quad ((\lambda 0\ \lambda 0)\ \lambda 0)\quad (\lambda(\lambda 0\ 0)\ \lambda 0)\quad (\lambda(0\ \lambda 0)\ \lambda 0)\quad \lambda(\lambda 0\ (\lambda 0\ 0))\quad \lambda(\lambda 0\ (0\ \lambda 0))$$

$$\lambda((\lambda 0\ \lambda 0)\ 0)\quad \lambda(\lambda(\lambda 0\ 0)\ 0)\quad \lambda(\lambda(0\ \lambda 0)\ 0)\quad \lambda(0\ (\lambda 0\ \lambda 0))\quad \lambda(0\ \lambda(\lambda 0\ 0))\quad \lambda(0\ \lambda(0\ \lambda 0))\quad \lambda((\lambda 0\ 0)\ \lambda 0)\quad \lambda((0\ \lambda 0)\ \lambda 0)$$

written with explicit variables

$$\lambda x.x\ (\lambda x.x\ \lambda x.x)\quad \lambda x.x\ \lambda y.(\lambda x.x\ y)\quad \lambda x.x\ \lambda y.(y\ \lambda x.x)\quad (\lambda x.x\ \lambda x.x)\ \lambda x.x$$

$$\lambda y.(\lambda x.x\ y)\ \lambda x.x\quad \lambda y.(y\ \lambda x.x)\ \lambda x.x\quad \lambda y.(\lambda x.x\ (\lambda x.x\ y))\quad \lambda y.(\lambda x.x\ (y\ \lambda x.x))$$

$$\lambda y.((\lambda x.x\ \lambda x.x)\ y)\quad \lambda y.(\lambda z.(\lambda x.x\ z)\ y)\quad \lambda y.(\lambda z.(z\ \lambda x.x)\ y)\quad \lambda y.(y\ (\lambda x.x\ \lambda x.x))$$

$$\lambda y.(y\ \lambda z.(\lambda x.x\ z))\quad \lambda y.(y\ \lambda z.(z\ \lambda x.x))\quad \lambda y.((\lambda x.x\ y)\ \lambda x.x)\quad \lambda y.((y\ \lambda x.x)\ \lambda x.x)$$

and there are 25 affine terms of natural size 7:

$$(\lambda 0\ \lambda\lambda 1)\quad (\lambda 0\ \lambda\lambda\lambda 0)\quad (\lambda\lambda 0\ \lambda\lambda 0)\quad (\lambda\lambda 1\ \lambda 0)\quad (\lambda\lambda\lambda 0\ \lambda 0)\quad \lambda(\lambda\lambda 1\ 0)\quad \lambda(\lambda\lambda\lambda 0\ 0)\quad \lambda(0\ \lambda\lambda 1)$$

$$\lambda(0\ \lambda\lambda\lambda 0)\quad \lambda(\lambda 0\ \lambda 1)\quad \lambda(\lambda 1\ \lambda 0)\quad \lambda\lambda(\lambda 0\ 1)\quad \lambda\lambda(1\ \lambda 0)\quad \lambda(\lambda 0\ \lambda\lambda 0)\quad \lambda(\lambda\lambda 0\ \lambda 0)\quad \lambda\lambda(\lambda\lambda 0\ 0)$$

$$\lambda\lambda(0\ \lambda\lambda 0)\quad \lambda\lambda\lambda(0\ 1)\quad \lambda\lambda\lambda(1\ 0)\quad \lambda\lambda\lambda\lambda 2\quad \lambda\lambda(\lambda 0\ \lambda 0)\quad \lambda\lambda\lambda(\lambda 0\ 0)\quad \lambda\lambda\lambda(0\ \lambda 0)\quad \lambda\lambda\lambda\lambda\lambda 1\quad \lambda\lambda\lambda\lambda\lambda\lambda 0$$

The Haskell programs of this development are on GitHub: `https://github.com/PierreLescanne/CountingGeneratingAfffineLinearClosedLambdaterms`.

*Date*: February 23, 2017.

[1]If the reader is not familiar with the $\lambda$-calculus, we advise him to read the introduction of [12].

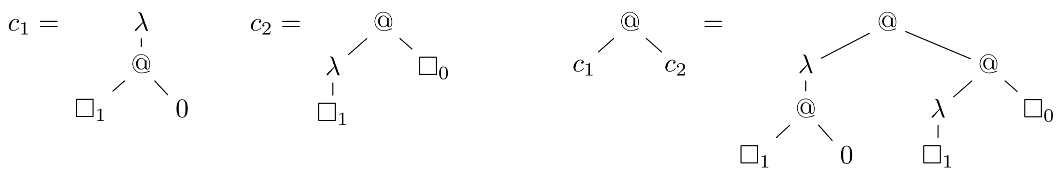

FIGURE 1. Building a SwissCheese by application

**Notations.** In this paper we use specific notations.

Given a predicate $p$, the Iverson notation written $[p(x)]$ is the function taking natural values which is 1 if $p(x)$ is true and which is 0 if $p(x)$ is false.

Let $\mathbf{m} \in \mathbb{N}^p$ be the $p$-tuple $(m_0, ..., m_{p-1})$. In Section 5, we consider infinite tuples. Thus $\mathbf{m} \in \mathbb{N}^\omega$ is the sequence $(m_0, m_1, ...)$.

- $p$ is the *length* of $\mathbf{m}$, which we write also $\mathsf{length}\ \mathbf{m}$
- The $p$-tuple $(0, ..., 0)$ is written $0^p$. $0^\omega$ is the infinite sequence made of 0's.
- The *increment* of a $p$-tuple at $i$ is:

$$\mathbf{m}^{\uparrow i} = \mathbf{n} \in \mathbb{N}^p \text{ where } n_j = m_j \text{ if } j \neq i \text{ and } n_i = m_i + 1$$

- Putting an element $x$ as *head* of a tuple is written

$$x : \mathbf{m} = x : (m_0, ...) = (x, m_0, ...)$$

  $\mathsf{tail}$ removes the head of an tuple:

$$\mathsf{tail}(x : \mathbf{m}) = \mathbf{m}.$$

- $\oplus$ is the componentwise addition on tuples.

## 2. SwissCheese

The basic concept is this of **m-SwissCheese** or **SwissCheese** if there is no ambiguity on $\mathbf{m}$. That is a $\lambda$-term with holes of $p$ levels, which are all counted, using $\mathbf{m}$. The $p$ levels of holes are $\Box_0$,... $\Box_{p-1}$. A hole $\Box_i$ is meant to be a location for a variable at level $i$, that is under $i$ $\lambda$'s. According to the way bound variables are inserted when creating abstractions (see below), we consider linear or affine SwissCheeses. The holes have size 0. An $\mathbf{m}$-SwissCheese has $m_0$ holes at level 0, $m_1$ holes at level 1, ... $m_{p-1}$ holes at level $p$. Let $l_{n,\mathbf{m}}$ (resp. $a_{n,\mathbf{m}}$) count the linear (resp. the affine) $\mathbf{m}$-SwissCheese of size $n$. $l_{n,\mathbf{m}} = l_{n,\mathbf{m}'}$ and $a_{n,\mathbf{m}} = a_{n,\mathbf{m}'}$ if $\mathbf{m}$ is finite, $\mathsf{length}\ \mathbf{m} \geq n$, $m_i = m'_i$ for $i \leq \mathsf{length}\ \mathbf{m}$, and $m'_i = 0$ for $i > \mathsf{length}\ \mathbf{m}$. $l_{n,0^n}$ (resp. $a_{n,0^n}$) counts the closed linear (resp. the closed affine) $\lambda$-terms.

### 2.1. Growing a SwissCheese.

Given two SwissCheeses, we can build a SwissCheese by application like in Fig 1. In Fig. 1, $c_1$ is a $(0, 1, 0, 0, 0)$-SwissCheese, $c_2$ is a $(1, 1, 0, 0, 0)$-SwissCheese and $c_1@c_2$ is a $(1, 2, 0, 0, 0)$-SwissCheese.

Given a SwissCheese, there are two ways to grow a SwissCheese to make another SwissCheese by abstraction.

(1) We put a $\lambda$ on the top of a $\mathbf{m}$-SwissCheese $c$. This increases the levels of the holes: a hole $\Box_i$ becomes a hole $\Box_{i+1}$. $\lambda c$ is a $(0 : \mathbf{m})$-SwissCheese. See Fig 2 on the left. This way, no index is bound by the top $\lambda$, therefore this does not preserve linearity (it preserves affinity however). Therefore this construction is only for building affine SwissCheeses, not for building linear SwissCheeses. In Figure 2 (left), we colour the added $\lambda$ in blue and we call it *abstraction with no binding.*

(2) In the second method for growing a SwissCheese by abstraction, we select first a hole $\Box_i$, we top the SwissCheese by a $\lambda$, we increment the levels of the other holes and we replace the chosen box by $S^i 0$. In Figure 2 (right), we colour the added $\lambda$ in green and we call it *abstraction with binding.*

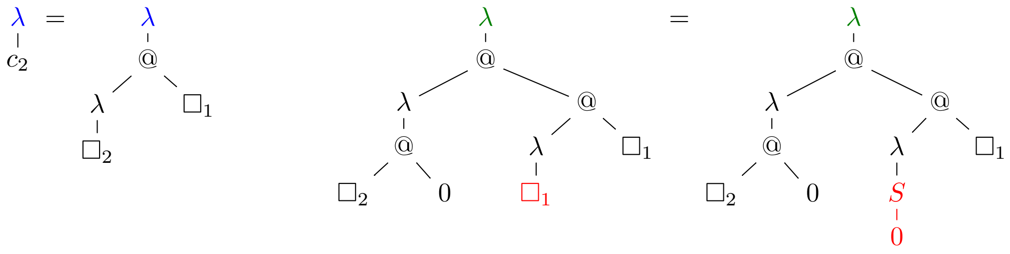

FIGURE 2. Abstracting SwissCheeses without and with binding

2.2. **Measuring SwissCheese.** We considers several ways of measuring the size of a SwissCheese derived from what is done on $\lambda$-terms. In all these sizes, applications @ and abstractions $\lambda$ have size 1 and holes have size 0. The differences are in the way variables are measured.

- Variables have size 0, we call this **variable size** 0.
- Variables have size 1, we call this **variable size** 1 .
- Variables (or de Bruijn indices) $S^i 0$ have size $i+1$, we call this **natural size**.

## 3. Counting linear closed terms

We start with counting linear terms since they are slightly simpler. We will give recursive formulas first for the numbers $l^{\nu}_{n,\mathbf{m}}$ of linear SwissCheeses of natural size $n$ with holes set by $\mathbf{m}$, then for the numbers $l^{0}_{n,\mathbf{m}}$ of linear SwissCheeses of size $n$, for variable size 0, with holes set by $\mathbf{m}$, eventually for the numbers $l^{1}_{n,\mathbf{m}}$ of linear SwissCheeses of size $n$, for variable size 1, with holes set by $\mathbf{m}$. When we do not want to specify a chosen size, we write only $l_{n,\mathbf{m}}$ without superscript.

3.1. **Natural size.** First let us count linear SwissCheeses with natural size. This is given by the coefficient $l^{\nu}$ which has two arguments: the size $n$ of the SwissCheese and a tuple $\mathbf{m}$ which specifies the number of holes of each level. In other words we are interested by the quantity $l_{n,\mathbf{m}}$. We assume that the length of $\mathbf{m}$ is $p$, greater than $n$.

**$\mathbf{n = 0}$:** whatever size is considered, there is only one SwissCheese of size 0 namely $\Box_0$. This means that the number of SwissCheeses of size 0 is 1 if and only if $\mathbf{m} = (1, 0, 0, ...)$:

$$l_{0,\mathbf{m}} \quad = \quad [m_0 = 1 \wedge \bigwedge_{j=1}^{p} m_j = 0]$$

**$\mathbf{n \neq 0}$ and application:** if a $\lambda$-term of size $n$ has holes set by $\mathbf{m}$ and is an application, then it is obtained from a $\lambda$ term of size $k$ with holes set by $\mathbf{q}$ and a $\lambda$ term of size $n-k-1$ with holes set by $\mathbf{r}$, with $\mathbf{m} = \mathbf{q} \oplus \mathbf{r}$:

$$\sum_{\mathbf{q} \oplus \mathbf{r} = \mathbf{m}} \sum_{k=0}^{n} l_{k,\mathbf{q}}\, l_{n-1-k,\mathbf{r}}$$

**$\mathbf{n \neq 0}$ and abstraction with binding:** consider a level $i$, that is a level of hole $\Box_i$. In this hole we put a term $S^{i-1}0$ of size $i$. There are $m_i$ ways to choose a hole $\Box_i$. Therefore there are $m_i\, l^{\nu}_{n-i-1,\mathbf{m}}$ SwissCheeses which are abstractions with binding in which a $\Box_i$ has been replaced by the de Bruijn index $S^{i-1}0$ among $l^{\nu}_{n,0:\mathbf{m}^{\downarrow \mathbf{i}}}$ SwissCheeses, where $\mathbf{m}^{\downarrow \mathbf{i}}$ is $\mathbf{m}$ in which $m_i$ is decremented. We notice that this refers only to an $\mathbf{m}$ starting with 0. Hence by summing over $i$ and adjusting $\mathbf{m}$, this part contributes as:

$$\sum_{i=0}^{p} (m_i + 1)\; l^{\nu}_{n-i,\mathbf{m}^{\uparrow i}}$$

to $l^{\nu}_{n+1,0:\mathbf{m}}$.

We have the following recursive definitions of $l^{\nu}_{n,\mathbf{m}}$:

$$
\begin{aligned}
l^{\nu}_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} l^{\nu}_{k,\mathbf{q}}\, l^{\nu}_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\ l^{\nu}_{n-i,\mathbf{m}\uparrow i}\\
l^{\nu}_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} l^{\nu}_{k,\mathbf{q}}\, l^{\nu}_{n-k,\mathbf{r}}
\end{aligned}
$$

3.2. **Variable size 0.** The only difference is that the inserted de Bruijn index has size 0. Therefore we have $m_i\, l^{0}_{n-1,\mathbf{m}}$ where we had $m_i\, l^{\nu}_{n-i-1,\mathbf{m}}$ for natural size. Hence the formulas:

$$
\begin{aligned}
l^{0}_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} l^{0}_{k,\mathbf{q}}\, l^{0}_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\ l^{0}_{n,\mathbf{m}\uparrow i}\\
l^{0}_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} l^{0}_{k,\mathbf{q}}\, l^{0}_{n-k,\mathbf{r}}
\end{aligned}
$$

The sequence $l^{0}_{n,0^n}$ of the numbers of closed linear terms is $0,1,0,5,0,60,0,1105,0,27120,0,828250$, which is sequence A062980 in the *On-line Encyclopedia of Integer Sequences* with 0's at even indices.

3.3. **Variable size 1.** The inserted de Bruijn index has size 1. We have $m_i\, l^{1}_{n-2,\mathbf{m}}$ where we had $m_i\, l^{\nu}_{n-i-1,\mathbf{m}}$ for natural size. Moreover we have to give the value of $l^{1}_{1,\mathbf{m}}$ which is

$$
l^{1}_{1,\mathbf{m}} \quad=\quad [m_0 = 2 \wedge \bigwedge_{j=1}^{p} m_j = 0]
$$

since there is only one SwissCheese of size 1, namely $\square_0\,\square_0$, which corresponds to the tuple $[2,0,...]$.

$$
\begin{aligned}
l^{1}_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} l^{1}_{k,\mathbf{q}}\, l^{1}_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\ l^{1}_{n-1,\mathbf{m}\uparrow i}\\
l^{1}_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} l^{1}_{k,\mathbf{q}}\, l^{1}_{n-k,\mathbf{r}}
\end{aligned}
$$

There are no linear closed $\lambda$-terms of size $3k$ and $3k+1$. However for the values $3k+2$ we get the sequence: $1,5,60,1105,27120,\ldots$ which is again sequence A062980 of the *On-line Encyclopedia of Integer Sequences*. Zeilberger [16] notices that this sequence counts the closed linear $\lambda$-terms when the size of a term is the number of applications. The resurgence of sequence A062980 can be explained as follows: closed linear $\lambda$-terms have the same number of variables (de Bruijn indices) as the number of abstractions and as the number of applications minus 1. Variable size 1 counts the number of applications + the number of abstractions + the number of variables, wherever variable size 0 counts the number of applications + the number of abstractions. Due to the above restrictions on closed linear $\lambda$-terms, there cannot exist closed linear $\lambda$-terms of odd variable size 0 and there cannot exist closed linear $\lambda$-terms of variable size 1 equal to $3k$ and $3k+1$. And the other non 0 values are counted by sequence A062980.

## 4. Counting affine closed terms

We have just to add the case $n \neq 0$ *and abstraction without binding*. Since no index is added, the size increases by 1. The numbers are written $a^{\nu}_{n,\mathbf{m}}$, $a^{0}_{n,\mathbf{m}}$, $a^{1}_{n,\mathbf{m}}$, and $a_{n,\mathbf{m}}$ when the size does not matter. There are $(0:\mathbf{m})$-SwissCheeses of size $n$ that are abstraction without binding. We get the recursive formulas:

**Natural size:**

$$
\begin{aligned}
a^{\nu}_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} a^{\nu}_{k,\mathbf{q}}\, a^{\nu}_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\ a^{\nu}_{n-i,\mathbf{m}\uparrow i} + a^{\nu}_{n,\mathbf{m}}\\
a^{\nu}_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} a^{\nu}_{k,\mathbf{q}}\, a^{\nu}_{n-k,\mathbf{r}}
\end{aligned}
$$

**Variable size** $0$**:**

$$\begin{aligned}
a^0_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} a^0_{k,\mathbf{q}}\, a^0_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\; a^0_{n,\mathbf{m}\uparrow i} + a^0_{n,\mathbf{m}}\\
a^0_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} a^0_{k,\mathbf{q}}\, a^0_{n-k,\mathbf{r}}
\end{aligned}$$

**Variable size** $1$**:**

$$\begin{aligned}
a^1_{1,\mathbf{m}} &= [m_0 = 2 \wedge \bigwedge_{j=1}^{p} m_j = 0]\\
a^1_{n+1,0:\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} a^1_{k,\mathbf{q}}\, a^1_{n-k,\mathbf{r}} + \sum_{i=0}^{p}(m_i+1)\; a^1_{n-1,\mathbf{m}\uparrow i} + a^1_{n,\mathbf{m}}\\
a^1_{n+1,(h+1):\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}=(h+1):\mathbf{m}} \sum_{k=0}^{n} a^1_{k,\mathbf{q}}\, a^1_{n-k,\mathbf{r}}
\end{aligned}$$

## 5. Generating functions

Consider families $F_{\mathbf{m}}(z)$ of generating functions indexed by $\mathbf{m}$, where $\mathbf{m}$ is an infinite tuple of naturals. In fact, we are interested in the infinite tuples $\mathbf{m}$ that are always 0, except a finite number of indices, in order to compute $F_{0^\omega}(z)$, which corresponds to closed $\lambda$-terms. Let $\mathbf{u}$ stands for the infinite sequences of variables $(u_0, u_1, ...)$ and $\mathbf{u}^{\mathbf{m}}$ stands for $(u_0^{m_0}, u_1^{m_1}, ..., u_n^{m_n}, ...)$ and $\mathsf{tail}\,(\mathbf{u})$ stand for $(u_1, ...)$. We consider the series of two variables $z$ and $\mathbf{u}$ or double series associated with $F_{\mathbf{m}}(z)$:

$$\mathcal{F}(z,\mathbf{u}) = \sum_{\mathbf{m}\in\mathbb{N}^\omega} F_{\mathbf{m}}(z)\,\mathbf{u}^{\mathbf{m}}.$$

**Natural size.** $L^\nu_{\mathbf{m}}(z)$ is associated with the numbers of *closed linear SwissCheeses* for natural size:

$$\begin{aligned}
L^\nu_{0:\mathbf{m}}(z) &= z \sum_{\mathbf{m}'\oplus\mathbf{m}''=0:\mathbf{m}} L^\nu_{\mathbf{m}'}(z) L^\nu_{\mathbf{m}''}(z) + z\sum_{i=0}^{\infty}(m_i+1)z^i L^\nu_{\mathbf{m}\uparrow i}(z)\\
L^\nu_{(h+1):\mathbf{m}}(z) &= [h = 0 + \bigwedge_{i=0}^{\infty} m_i = 0] + z \sum_{\mathbf{m}'\oplus\mathbf{m}''=(h+1):\mathbf{m}} L^\nu_{\mathbf{m}'}(z) L^\nu_{\mathbf{m}''}(z)
\end{aligned}$$

$L^\nu_{0^\omega}$ is the generating function for the closed linear $\lambda$-terms. $\mathcal{L}^\nu(z,\mathbf{u})$ is the double series associated with $L^\nu_{\mathbf{m}}(z)$ and is solution of the equation:

$$\mathcal{L}^\nu(z,\mathbf{u}) = u_0 + z(\mathcal{L}^\nu(z,\mathbf{u}))^2 + z\,u_0 \sum_{i=1}^{\infty} z^i \frac{\partial \mathcal{L}^\nu(z,(\mathsf{tail}\,(\mathbf{u}))}{\partial u^i}$$

$\mathcal{L}^\nu(z, 0^\omega)$ is the generating function of closed linear $\lambda$-terms.

For *closed affine SwissCheeses* we get:

$$\begin{aligned}
A^\nu_{0:\mathbf{m}}(z) &= z \sum_{\mathbf{m}'\oplus\mathbf{m}''=0:\mathbf{m}} A^\nu_{\mathbf{m}'}(z) A^\nu_{\mathbf{m}''}(z) + z\sum_{i=0}^{\infty}(m_i+1)z^i A^\nu_{\mathbf{m}\uparrow i}(z) + z\,A^\nu_{\mathbf{m}}(z)\\
A^\nu_{(h+1):\mathbf{m}}(z) &= [h = 0 + \bigwedge_{i=0}^{\infty} m_i = 0] + z \sum_{\mathbf{m}'\oplus\mathbf{m}''=(h+1):\mathbf{m}} A^\nu_{\mathbf{m}'}(z) A^\nu_{\mathbf{m}''}(z)
\end{aligned}$$

$A^\nu_{0^\omega}$ is the generating function for the affine linear $\lambda$-terms. $\mathcal{A}^\nu(z,\mathbf{u})$ is the double series associated with $A^\nu_{\mathbf{m}}(z)$ and is solution of the equation:

$$\mathcal{A}^\nu(z,\mathbf{u}) = u_0 + z(\mathcal{A}^\nu(z,\mathbf{u}))^2 + zu_0 \sum_{i=1}^{\infty} z^i \frac{\partial \mathcal{A}^\nu(z,\mathsf{tail}\,(\mathbf{u}))}{\partial u^i} + z\mathcal{A}^\nu(z,\mathsf{tail}\,(\mathbf{u}))$$

$\mathcal{A}^\nu(z, 0^\omega)$ is the generating function of closed linear $\lambda$-terms.

**Variable size** $0$**.** $L^0_{\mathbf{m}}$ is associated with the numbers of *closed linear SwissCheeses* for variable size 0:

$$\begin{aligned} L^0_{0:\mathbf{m}}(z) &= z \sum_{\mathbf{m}'\oplus\mathbf{m}''=\mathbf{m}} L^0_{\mathbf{m}'}(z) L^0_{\mathbf{m}''}(z) + z \sum_{i=0}^{\infty} (m_i+1) L^0_{\mathbf{m}^{\uparrow i}}(z) \\ L^0_{(h+1):\mathbf{m}}(z) &= [h=0+\bigwedge_{i=0}^{\infty} m_i = 0] + \sum_{\mathbf{m}'\oplus\mathbf{m}''=\mathbf{m}} z L^0_{\mathbf{m}'}(z) L^0_{\mathbf{m}''}(z) \end{aligned}$$

$L^0_{0^\omega}$ is the generating function for the closed linear $\lambda$-terms. $\mathcal{L}^0(z,\mathbf{u})$ is the double series associated with $L^0_{\mathbf{m}}(z)$ and is solution of the equation:

$$\mathcal{L}^0(z,\mathbf{u}) = u_0 + z(\mathcal{L}^0(z,\mathbf{u}))^2 + z\, u_0 \sum_{i=1}^{\infty} \frac{\partial \mathcal{L}^0(z, (\mathsf{tail}\,(\mathbf{u}))}{\partial u^i}$$

$\mathcal{L}^0(z, 0^\omega)$ is the generating function of closed linear $\lambda$-terms.

For *closed affine SwissCheeses* we get:

$$\begin{aligned} A^0_{0:\mathbf{m}}(z) &= z \sum_{\mathbf{m}'\oplus\mathbf{m}''=0:\mathbf{m}} A^0_{\mathbf{m}'}(z) A^0_{\mathbf{m}''}(z) + z \sum_{i=0}^{\infty} (m_i+1) A^0_{\mathbf{m}^{\uparrow i}}(z) + z\, A^0_{\mathbf{m}}(z) \\ A^0_{(h+1):\mathbf{m}}(z) &= [h=0+\bigwedge_{i=0}^{\infty} m_i = 0] + \sum_{\mathbf{m}'\oplus\mathbf{m}''=(h+1):\mathbf{m}} z A^0_{\mathbf{m}'}(z) A^0_{\mathbf{m}''}(z) \end{aligned}$$

$A^0_{0^\omega}$ is the generating function for the affine linear $\lambda$-terms. $\mathcal{A}^0(z,\mathbf{u})$ is the double series associated with $A^0_{\mathbf{m}}(z)$ and is solution of the equation:

$$\mathcal{A}^0(z,\mathbf{u}) = u_0 + z(\mathcal{A}^0(z,\mathbf{u}))^2 + z\, u_0 \sum_{i=1}^{\infty} \frac{\partial \mathcal{A}^0(z, \mathsf{tail}\,(\mathbf{u}))}{\partial u^i} + z\mathcal{A}^0(z, \mathsf{tail}\,(\mathbf{u}))$$

$\mathcal{A}^0(z, 0^\omega)$ is the generating function of closed linear $\lambda$-terms. We do not present variable size 1, since it goes exactly the same way.

**Variable size** $1$**.** The generating functions for $l^1_{n,\mathbf{m}}$ are:

$$\begin{aligned} L^1_{0:\mathbf{m}}(z) &= z \sum_{\mathbf{m}'\oplus\mathbf{m}''=\mathbf{m}} L^1_{\mathbf{m}'}(z) L^1_{\mathbf{m}''}(z) + z^2 \sum_{i=0}^{\infty} (m_i+1) L^1_{\mathbf{m}^{\uparrow i}}(z) \\ L^1_{(h+1):\mathbf{m}}(z) &= [h=0+\bigwedge_{i=0}^{\infty} m_i = 0] + z[h=1+\bigwedge_{i=0}^{\infty} m_i = 0] + \sum_{\mathbf{m}'\oplus\mathbf{m}''=\mathbf{m}} z L^1_{\mathbf{m}'}(z) L^1_{\mathbf{m}''}(z) \end{aligned}$$

Then we get as associated double series :

$$\mathcal{L}^1(z,\mathbf{u}) = u_0 + z u_0^2 + z(\mathcal{L}^1(z,\mathbf{u}))^2 + z^2\, u_0 \sum_{i=1}^{\infty} \frac{\partial \mathcal{L}^1(z, (\mathsf{tail}\,(\mathbf{u}))}{\partial u^i}$$

## 6. Effective computations

The definition of the coefficients $a^\nu_{\mathbf{m}}$ and others is highly recursive and requires a mechanism of memoization. In Haskell, this can be done by using the call by need which is at the core of this language. Assume we want to compute the values of $a^\nu_{\mathbf{m}}$ until a value `bound` for $n$. We use a recursive data structure:

```
data Mem = Mem [Mem] | Load [Integer]
```

in which we store the computed values of a function

```
am :: Int -> [Int] -> Integer
```

In our implementation the depth of the recursion of `Mem` is limited by `bound`, which is also the longest tuple $\mathbf{m}$ for which we will compute $a^\nu_{\mathbf{m}}$. Associated with `Mem` there is a function

```
access :: Mem -> Int -> [Int] -> Integer access (Load l) n [] = l !! n
access (Mem listM) n (k:m) = access (listM !! k) n m
```

The leaves of the tree memory, corresponding to `Load`, contains the values of the function:

```
memory :: Int -> [Int] -> Mem
memory 0 m = Load [am n (reverse m) | n<-[0..]]
memory k m = Mem [memory (k-1) (j:m) | j<-[0..]]
```

The memory relative to the problem we are interested in is

```
theMemory = memory (bound) []
```

and the access to `theMemory` is given by a specific function:

```
acc :: Int -> [Int] -> Integer acc n m = access theMemory n m
```

Notice that `am` and `acc` have the same signature. This is not a coincidence, since `acc` accesses values of `am` already computed. Now we are ready to express `am`:

```
am 0 m = iv (head m == 1 && all ((==) 0) (tail m))
am n m = amAPP n m + amABSwB n m + amABSnB n m
```

`amAPP` *counts affine terms that are applications:*

```
amAPP n m = sum (map (\((q,r),(k,nk))->(acc k q)*(acc nk r))
                     (allCombinations m (n-1)))
```

where `allCombinations` returns a list of all the pairs of pairs $(\mathbf{m}', \mathbf{m}'')$ such $\mathbf{m} = \mathbf{m}' \oplus \mathbf{m}''$ and of pairs $(k, nk)$ such that $k + nk = n$. `amABSwB` *counts affine terms that are abstractions with binding.*

```
amABSwB n m
   | head m == 0 = sum [amABSAtD n m i |i<-[1..(n-1)]]
   | otherwise = 0
```

`amABSAtD` *counts affine terms that are abstractions with binding at level* $i$:

```
amABSAtD n m i = (fromIntegral (1 + m!!i))*(acc (n-i-1) (tail (inc i m) ++ [0]))
```

`amABSnB` *counts affine terms that are abstractions with no binding*:

```
amABSnB n m
     | head m == 0 = (acc (n-1) (tail m ++ [0]))
     | otherwise = 0
```

Anyway the efficiency of this program is limited by the size of the memory, since for computing $a^{\nu}_{n,0^n}$, for instance, we need to compute $a^{\nu}_{\mathbf{r}}$ for about $n!$ values.

## 7. Generating affine and linear terms

By relatively small changes it is possible to build programs which generate linear and affine terms. For instance for generating affine terms we get.

```
amg :: Int -> [Int] -> [SwissCheese]
amg 0 m =  if (head m == 1 && all ((==) 0) (tail m)) then [Box 0] else []
amg n m = allAPP n m ++ allABSwB n m ++ allABSnB n m

allAPP  :: Int -> [Int] -> [SwissCheese]
allAPP n m = foldr (++) [] (map (\((q,r),(k,nk))-> appSC (cartesian (accAG k q)
                                                                    (accAG nk r))
                            (allCombinations m (n-1)))

allABSAtD :: Int -> [Int] -> Int -> [SwissCheese]
allABSAtD n m i = foldr (++) [] (map (abstract (i-1)) (accAG (n - i - 1)
                                                       (tail (inc i m) ++ [0])))

allABSwB  :: Int -> [Int] -> [SwissCheese]
allABSwB n m
  | head m == 0 = foldr (++) [] [allABSAtD n m i |i<-[1..(n-1)]]
  | otherwise = []

allABSnB  :: Int -> [Int] -> [SwissCheese]
allABSnB n m
  | head m == 0 = map (AbsSC . raise) (accAG (n-1) (tail m ++ [0]))
  | otherwise = []

memoryAG :: Int -> [Int] -> MemSC
memoryAG 0 m = LoadSC [amg n (reverse m) | n<-[0..]]
memoryAG k m = MemSC [memoryAG (k-1) (j:m) | j<-[0..]]
```

```
theMemoryAG = memoryAG (upBound) []

accAG :: Int -> [Int] -> [SwissCheese]
accAG n m = accessSC theMemoryAG n m
```

There is similar programs for generating all the terms of size $n$ for variable size 0 and variable size 1. From this, we get programs for generating random affine terms or random linear terms.

## 8. Normal forms

From the method used for counting affine and linear closed terms, it is easy to deduce method for counting affine and linear closed normal forms. Like before, we use SwissCheeses. In this section we consider only natural size.

### 8.1. Natural size.

*Affine closed normal forms.* Let us call $anf^{\nu}_{n,\mathbf{m}}$ the numbers of affine SwissCheeses with no $\beta$-redex and $ane^{\nu}_{n,\mathbf{m}}$ the numbers of neutral affine SwissCheeses, i.e., affine SwissCheeses with no $\beta$-redexes that are sequences of applications starting with a de Bruijn index. In addition we count:

- $anf^{\nu}\lambda w_{n,m}$ the number of affine SwissCheeses with no $\beta$-redex which are abstraction with a binding of a de Bruijn index,
- $anf^{\nu}\lambda n_{n,m}$ the number of affine SwissCheeses with no $\beta$-redex which are abstraction with no binding.

$$
\begin{aligned}
anf^{\nu}_{0,\mathbf{m}} &= ane^{\nu}_{0,\mathbf{m}}\\
anf^{\nu}_{n+1,\mathbf{m}} &= ane^{\nu}_{n+1,\mathbf{m}} + anf^{\nu}\lambda w_{n+1,m} + anf^{\nu}\lambda n_{n+1,m}
\end{aligned}
$$

where

$$
\begin{aligned}
ane^{\nu}_{0,\mathbf{m}} &= m_0 = 1 \wedge \bigwedge_{j=1}^{p} m_j = 0\\
ane^{\nu}_{n+1,\mathbf{m}} &= \sum_{\mathbf{q}\oplus\mathbf{r}\,=\,0:\mathbf{m}} \sum_{k=0}^{n} ane^{\nu}_{k,\mathbf{q}}\, anf^{\nu}_{n-k,\mathbf{r}}
\end{aligned}
$$

and

$$
anf^{\nu}\lambda w_{n,m} = \sum_{i=0}^{p}(m_i+1)\; anf^{\nu}_{n-i,\mathbf{m}^{\uparrow i}}
$$

and

$$
anf^{\nu}\lambda n_{n+1,m} = anf^{\nu}_{n,m}
$$

There are two generating functions, $\mathcal{A}^{nf}$ and $\mathcal{A}^{ne}$, which are associated to $anf^{\nu}_{n,\mathbf{m}}$ and $anf^{\nu}_{n,\mathbf{m}}$:

$$
\begin{aligned}
\mathcal{A}^{nf}(z,\mathbf{u}) &= \mathcal{A}^{ne}(z,\mathbf{u}) + z\,u_0 \sum_{i=1}^{\infty} z^i \frac{\partial \mathcal{A}^{nf}(z,(\mathsf{tail}\,(\mathbf{u}))}{\partial u^i} + z\mathcal{A}^{nf}(z,(\mathsf{tail}\,(\mathbf{u}))\\
\mathcal{A}^{ne}(z,\mathbf{u}) &= u_0 + z\mathcal{A}^{ne}(z,\mathbf{u})\mathcal{A}^{nf}(z,\mathbf{u})
\end{aligned}
$$

*Linear closed normal forms.* Let us call $lnf^{\nu}_{n,\mathbf{m}}$ the numbers of linear SwissCheeses with no $\beta$-redex and $lne^{\nu}_{n,\mathbf{m}}$ the numbers of neutral linear SwissCheeses, linear SwissCheeses with no $\beta$-redexes that are sequences of applications starting with a de Bruijn index. In addition we count $lnf^{\nu}\lambda w_{n,m}$ the number of linear SwissCheeses with no $\beta$-redex which are abstraction with a binding of a de Bruijn index.

$$
\begin{aligned}
lnf^{\nu}_{0,\mathbf{m}} &= lne^{\nu}_{0,\mathbf{m}}\\
lnf^{\nu}_{n+1,\mathbf{m}} &= lne^{\nu}_{n+1,\mathbf{m}} + lnf^{\nu}\lambda w_{n+1,m}
\end{aligned}
$$

where

$$
\begin{aligned}
lne^{\nu}_{0,\mathbf{m}} &= m_0 = 1 \wedge \bigwedge_{j=1}^{p} m_j = 0 \\
lne^{\nu}_{n+1,\mathbf{m}} &= \sum_{\mathbf{q} \oplus \mathbf{r} \,=\, 0:\mathbf{m}} \sum_{k=0}^{n} lne^{\nu}_{k,\mathbf{q}}\, lnf^{\nu}_{n-k,\mathbf{r}}
\end{aligned}
$$

and

$$
lnf^{\nu}\lambda w_{n,m} = \sum_{i=0}^{p} (m_i + 1)\; lnf^{\nu}_{n-i,\mathbf{m}\uparrow i}
$$

with the two generating functions:

$$
\begin{aligned}
\mathcal{L}^{nf,\nu}(z,\mathbf{u}) &= \mathcal{L}^{ne,\nu}(z,\mathbf{u}) + z\, u_0 \sum_{i=1}^{\infty} z^i \frac{\partial \mathcal{L}^{nf,\nu}(z, (\mathsf{tail}\,(\mathbf{u}))}{\partial u^i} \\
\mathcal{L}^{ne,\nu}(z,\mathbf{u}) &= u_0 + z\mathcal{L}^{ne,\nu}(z,\mathbf{u})\mathcal{L}^{nf,\nu}(z,\mathbf{u})
\end{aligned}
$$

We also deduce programs for generating all the closed affine or linear normal forms of a given size from which we deduce programs for generating random closed affine or linear normal forms of a given size. For instance, here are three randoms linear closed normal forms (using de Bruijn indices) of natural size 28:

λλλλ(2 λ((1 2) λ(0 (5 1))))   λ(0 λλ(1 λλ((0 (2 λλ((1 λ0) 0))) 1)))   λ((0 λ0) λλ((0((1 λ0)λλ(1 (0 λ0))))λ0))

## 8.2. Variable size 0.

*Linear closed normal forms.* A little like previously, let us call $lnf^0_{n,\mathbf{m}}$ the numbers of linear SwissCheeses with no $\beta$-redex and $lne^0_{n,\mathbf{m}}$ the numbers of neutral linear SwissCheeses, linear SwissCheeses with no $\beta$-redexes that are sequences of applications starting with a de Bruijn index. In addition we count $lnf^0\lambda w_{n,m}$ the number of linear SwissCheeses with no $\beta$-redex which are abstraction with a binding of a de Bruijn index. We assume that the reader knows now how to proceed.

$$
\begin{aligned}
lnf^0_{0,\mathbf{m}} &= lne^0_{0,\mathbf{m}} \\
lnf^0_{n+1,\mathbf{m}} &= lne^0_{n+1,\mathbf{m}} + lnf^0\lambda w_{n+1,m}
\end{aligned}
$$

where

$$
\begin{aligned}
lne^{0}_{0,\mathbf{m}} &= m_0 = 1 \wedge \bigwedge_{j=1}^{p} m_j = 0 \\
lne^{0}_{n+1,\mathbf{m}} &= \sum_{\mathbf{q} \oplus \mathbf{r} \,=\, 0:\mathbf{m}} \sum_{k=0}^{n} lne^{0}_{k,\mathbf{q}}\, lnf^{0}_{n-k,\mathbf{r}}
\end{aligned}
$$

$$
lnf^{0}\lambda w_{n,m} = \sum_{i=0}^{p} (m_i + 1)\; lnf^{0}_{n,\mathbf{m}\uparrow i}
$$

and the two generating functions:

$$
\begin{aligned}
\mathcal{L}^{nf,0}(z,\mathbf{u}) &= \mathcal{L}^{ne,0}(z,\mathbf{u}) + z\, u_0 \sum_{i=1}^{\infty} \frac{\partial \mathcal{L}^{nf,0}(z, (\mathsf{tail}\,(\mathbf{u}))}{\partial u^i} \\
\mathcal{L}^{ne,0}(z,\mathbf{u}) &= u_0 + z\mathcal{L}^{ne,0}(z,\mathbf{u})\mathcal{L}^{nf,0}(z,\mathbf{u})
\end{aligned}
$$

With no surprise we get for $lnf^0_{n,0^n}$ the sequence:

$$0, 1, 0, 3, 0, 26, 0, 367, 0, 7142, 0, 176766, 0, 5304356, ...$$

mentioned by Zeilberger in [16] which are the coefficients of the generating function $\mathcal{L}^{nf,0}(z, 0^{\omega})$.

We let the reader deduce how to count closed affine normal forms for variable size 0 and closed linear and affine normal forms for variable size 1 alike. Notice that the Haskell programs are on the GitHub site.

## 9. Related works and Acknowledgement

There are several works on counting $\lambda$-terms, for instance on natural size [3, 2], on variable size 1 [8, 15], on variable size 0 [11], on affine terms with variable size 1 [6, 5], on linear $\lambda$-terms [18, 16, 17], also on a size based binary representation of the $\lambda$-calculus [12] (see [10] for a synthetic view of both natural size and binary size).

We would like to thank Olivier Bodini, Maciej Bendkowski, Katarzyna Grygiel and Noam Zeilberger for stimulating discussions.

## 10. Conclusion

This work on counting opens new perspective on the generation, for instance the random generation of closed lambda terms in the line of [12, 4].

## References


[1] Henk P. Barendregt. *The Lambda-Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[2] Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne, and Marek Zaionc. Combinatorics of $\lambda$-terms: a natural approach. *CoRR*, abs/1609.07593, 2016.

[3] Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne, and Marek Zaionc. A natural counting of lambda terms. In Rusins Martins Freivalds, Gregor Engels, and Barbara Catania, editors, *SOFSEM 2016: Theory and Practice of Computer Science - 42nd International Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 23-28, 2016, Proceedings*, volume 9587 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2016.

[4] Maciej Bendkowski, Katarzyna Grygiel, and Paul Tarau. Boltzmann samplers for closed simply-typed lambda terms. In Yuliya Lierler and Walid Taha, editors, *Practical Aspects of Declarative Languages - 19th International Symposium, PADL 2017, Paris, France, January 16-17, 2017, Proceedings*, volume 10137 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2017.

[5] Olivier Bodini, Danièle Gardy, Bernhard Gittenberger, and Alice Jacquot. Enumeration of generalized BCI lambda-terms. *Electr. J. Comb.*, 20(4):P30, 2013.

[6] Olivier Bodini, Danièle Gardy, and Alice Jacquot. Asymptotics and random sampling for BCI and BCK lambda terms. *Theor. Comput. Sci.*, 502:227–238, 2013.

[7] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[8] René David, Christophe Raffalli, Guillaume Theyssier, Katarzyna Grygiel, Jakub Kozik, and Marek Zaionc. Some properties of random lambda terms. *Logical Methods in Computer Science*, 9(1), 2009.

[9] Nicolaas G. de Bruijn. Lambda calculus with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Proc. Koninkl. Nederl. Akademie van Wetenschappen*, 75(5):381–392, 1972.

[10] Bernhard Gittenberger and Zbigniew Golebiewski. On the number of lambda terms with prescribed size of their de Bruijn representation. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 40:1–40:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[11] Katarzyna Grygiel and Pierre Lescanne. Counting and generating lambda terms. *J. Funct. Program.*, 23(5):594–628, 2013.

[12] Katarzyna Grygiel and Pierre Lescanne. Counting and generating terms in the binary lambda calculus. *J. Funct. Program.*, 25, 2015.

[13] J. Roger Hindley. BCK-combinators and linear lambda-terms have types. *Theor. Comput. Sci.*, 64(1):97–105, 1989.

[14] J. Roger Hindley. *Basic Simple Type Theory*. Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[15] Pierre Lescanne. On counting untyped lambda terms. *Theor. Comput. Sci.*, 474:80–97, 2013.

[16] Noam Zeilberger. Counting isomorphism classes of $\beta$-normal linear lambda terms. *CoRR*, abs/1509.07596, 2015.

[17] Noam Zeilberger. Linear lambda terms as invariants of rooted trivalent maps. *J. Funct. Program.*, 26:e21, 2016.

[18] Noam Zeilberger and Alain Giorgetti. A correspondence between rooted planar maps and normal planar lambda terms. *Logical Methods in Computer Science*, 11(3), 2015.


## Data

In the appendix, we give the first values of $l^{\nu}_{n,0^n}$, $a^{\nu}_{n,0^n}$, and $anf^{\nu}_{n,0^n}$.

University of Lyon, École normale supérieure de Lyon, LIP (UMR 5668 CNRS ENS Lyon UCBL INRIA), 46 allée d'Italie, 69364 Lyon, France

*E-mail address*: pierre.lescanne@ens-lyon.fr

| | |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5 | 3 |
| 6 | 2 |
| 7 | 0 |
| 8 | 16 |
| 9 | 24 |
| 10 | 8 |
| 11 | 117 |
| 12 | 252 |
| 13 | 180 |
| 14 | 1024 |
| 15 | 2680 |
| 16 | 2952 |
| 17 | 10350 |
| 18 | 29420 |
| 19 | 42776 |
| 20 | 116768 |
| 21 | 335520 |
| 22 | 587424 |
| 23 | 1420053 |
| 24 | 3976424 |
| 25 | 7880376 |
| 26 | 18103936 |
| 27 | 48816576 |
| 28 | 104890704 |
| 29 | 237500826 |
| 30 | 617733708 |
| 31 | 1396750576 |
| 32 | 3171222464 |
| 33 | 8014199360 |
| 34 | 18688490336 |
| 35 | 42840683418 |
| 36 | 106063081288 |
| 37 | 251769197688 |
| 38 | 583690110208 |
| 39 | 1425834260080 |
| 40 | 3417671496432 |
| 41 | 8007221710652 |
| 42 | 19404994897976 |
| 43 | 46747189542384 |
| 44 | 110498345360800 |
| 45 | 266679286291872 |
| 46 | 644021392071840 |
| 47 | 1533054190557133 |
| 48 | 3693823999533360 |
| 49 | 8931109667692464 |
| 50 | 21375091547312128 |

| | |
|---|---|
| 51 | 51496022711337536 |
| 52 | 124591137939086496 |
| 53 | 299402908258405410 |
| 54 | 721839933329222924 |
| 55 | 1747307145272084192 |
| 56 | 4211741383777966592 |
| 57 | 10165998012602469888 |
| 58 | 24620618729658655936 |
| 59 | 59482734150603634286 |
| 60 | 143764591607556354344 |
| 61 | 348379929166234350008 |
| 62 | 843169238563254723200 |
| 63 | 2040572920613086128400 |
| 64 | 4948102905207104837424 |
| 65 | 11992521016286173712196 |
| 66 | 29059897435554891991144 |
| 67 | 70516464312280927105392 |
| 68 | 171105110698292441423968 |
| 69 | 415095704639682396539232 |
| 70 | 1008016383720573882885792 |
| 71 | 2448305474519849567597826 |
| 72 | 5945721872300885649415632 |
| 73 | 14449388516068567845838736 |
| 74 | 35125352062243788817753856 |
| 75 | 85382289240293493116120064 |
| 76 | 207650379931166057815603296 |
| 77 | 505172267243918348155299780 |
| 78 | 1229005880128485245247395000 |
| 79 | 2991079243470267667831893408 |
| 80 | 7281852742753184123608419712 |
| 81 | 17729171587798767750815341440 |
| 82 | 43177454620325445122944305984 |
| 83 | 105185452787117035266315446868 |
| 84 | 256273862465425158211948020048 |
| 85 | 624527413292252904584121980208 |
| 86 | 1522355057007327280427270436480 |
| 87 | 3711429775030704772089070886624 |
| 88 | 9050041253711022076275958636128 |
| 89 | 22073150301758857110072042919800 |
| 90 | 53844910909398928990641101351664 |
| 91 | 131371135544173914537076774932576 |
| 92 | 320588677238085642820920910555968 |
| 93 | 782465218885869813183863213231424 |
| 94 | 1910077425906069707804966102543936 |
| 95 | 4663586586924802791117231052636349 |
| 96 | 11388259565942452837717688743953504 |
| 97 | 27813754361897984543467478917223008 |
| 98 | 67941781284113201998645699501746176 |
| 99 | 165989485724048964272023600773271424 |
| 100 | 405588809305168453963137377442321728 |

FIGURE 3. *Natural size:* numbers of closed linear terms of size $n$ from 0 to 100

| | |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 1 |
| 3 | 1 |
| 4 | 2 |
| 5 | 5 |
| 6 | 12 |
| 7 | 25 |
| 8 | 64 |
| 9 | 166 |
| 10 | 405 |
| 11 | 1050 |
| 12 | 2763 |
| 13 | 7239 |
| 14 | 19190 |
| 15 | 51457 |
| 16 | 138538 |
| 17 | 374972 |
| 18 | 1020943 |
| 19 | 2792183 |
| 20 | 7666358 |
| 21 | 21126905 |
| 22 | 58422650 |
| 23 | 162052566 |
| 24 | 450742451 |
| 25 | 1256974690 |
| 26 | 3513731861 |
| 27 | 9843728012 |
| 28 | 27633400879 |
| 29 | 77721141911 |
| 30 | 218984204904 |
| 31 | 618021576627 |
| 32 | 1746906189740 |
| 33 | 4945026080426 |
| 34 | 14017220713131 |
| 35 | 39784695610433 |
| 36 | 113057573020242 |
| 37 | 321649935953313 |
| 38 | 916096006168770 |
| 39 | 2611847503880831 |
| 40 | 7453859187221508 |
| 41 | 21292177500898858 |
| 42 | 60875851617670699 |
| 43 | 174195916730975850 |
| 44 | 498863759031591507 |
| 45 | 1429753835635525063 |
| 46 | 4100730353324163138 |
| 47 | 11769771167532816128 |
| 48 | 33804054749367200891 |
| 49 | 97151933333668422006 |
| 50 | 279385977720772581435 |
| 51 | 803928779462727941247 |
| 52 | 2314623127904669382002 |
| 53 | 6667810436356967142481 |
| 54 | 19218411059885449257096 |
| 55 | 55421020161661024650870 |
| 56 | 159899218321197381984561 |
| 57 | 461557020400062903560120 |
| 58 | 1332920908954281811200519 |
| 59 | 3851027068336583693412910 |
| 60 | 11131032444503136571789527 |
| 61 | 32186581221116996967632029 |
| 62 | 93108410048006285466998584 |
| 63 | 269446191702411420790402033 |
| 64 | 780043726186403167392453886 |
| 65 | 2259043189995515315930349650 |
| 66 | 6544612955390252336187266873 |
| 67 | 18966737218108971681014445025 |
| 68 | 54985236298270057405776629352 |
| 69 | 159455737350384637847783055311 |
| 70 | 462562848624435724964181323484 |
| 71 | 1342251884451664733064283251627 |
| 72 | 3896065622127200625653134100538 |
| 73 | 11312117748805772104795220337816 |
| 74 | 32853646116456632492645965741531 |
| 75 | 95442534633482460553801961967438 |
| 76 | 277342191547330839640289978813667 |
| 77 | 806125189457291902863848267463755 |
| 78 | 2343682130911232279285707290604156 |
| 79 | 6815564023736534208079367816340359 |
| 80 | 19824812322145727566417303371819466 |
| 81 | 57679033022808238913186144092831856 |
| 82 | 167851787082561392384648248846390041 |
| 83 | 488574368670832093243802790464796207 |
| 84 | 1422426342380883254459783410845365006 |
| 85 | 4142104564089044203901190817275864665 |
| 86 | 12064305885705003967881526911560653106 |
| 87 | 35145647815239737143373764367447378676 |
| 88 | 102406303052123097062053564818109468705 |
| 89 | 298446029598661205216170897850336550644 |
| 90 | 869935452705023302189031644932803990417 |
| 91 | 2536229492704354513309696228592784181158 |
| 92 | 7395518143425160073537967606298755947391 |
| 93 | 21568776408467701927134211542478146593789 |
| 94 | 62915493935623036562559989770249004382816 |
| 95 | 183553775888862113259168150130266362416356 |
| 96 | 535600661621556969155453544692826625532079 |
| 97 | 1563109720672526919899689366626240867515144 |
| 98 | 4562542818801138452310024131223304186909233 |
| 99 | 13319630286623965617386598746472280781972745 |
| 100 | 38890520391341859449843201188612375394153776 |

FIGURE 4. *Natural size:* numbers of closed affine terms of size $n$ from 0 to 100

| | |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 1 |
| 3 | 1 |
| 4 | 2 |
| 5 | 3 |
| 6 | 7 |
| 7 | 10 |
| 8 | 20 |
| 9 | 40 |
| 10 | 77 |
| 11 | 160 |
| 12 | 318 |
| 13 | 671 |
| 14 | 1405 |
| 15 | 2981 |
| 16 | 6312 |
| 17 | 13672 |
| 18 | 29399 |
| 19 | 63697 |
| 20 | 139104 |
| 21 | 304153 |
| 22 | 667219 |
| 23 | 1469241 |
| 24 | 3247176 |
| 25 | 7184288 |
| 26 | 15949179 |
| 27 | 35480426 |
| 28 | 79083472 |
| 29 | 176607519 |
| 30 | 395119875 |
| 31 | 885450388 |
| 32 | 1987289740 |
| 33 | 4466760570 |
| 34 | 10053371987 |
| 35 | 22656801617 |
| 36 | 51121124910 |
| 37 | 115478296639 |
| 38 | 261139629999 |
| 39 | 591138386440 |
| 40 | 1339447594768 |
| 41 | 3037843646560 |
| 42 | 6895841598615 |
| 43 | 15666498585568 |
| 44 | 35620848278448 |
| 45 | 81052838239593 |
| 46 | 184564847153821 |
| 47 | 420564871255118 |
| 48 | 958975854646984 |
| 49 | 2188068392529104 |
| 50 | 4995528560788451 |
| 51 | 11411921511827547 |
| 52 | 26084524952754538 |
| 53 | 59654682828889245 |
| 54 | 136500653558490261 |
| 55 | 312496493161999851 |
| 56 | 715760763686417314 |
| 57 | 1640194881084692664 |
| 58 | 3760284787917366081 |
| 59 | 8624561382605096780 |
| 60 | 19789639944299656346 |
| 61 | 45427337308377290201 |
| 62 | 104320438668034814453 |
| 63 | 239656248361374562433 |
| 64 | 550769764273325683828 |
| 65 | 1266217774600330829940 |
| 66 | 2912050679107531357883 |
| 67 | 6699418399886008666265 |
| 68 | 15417663698156810292010 |
| 69 | 35492710197462925262295 |
| 70 | 81732521943462960197057 |
| 71 | 188270363628099910161436 |
| 72 | 433807135012774797924026 |
| 73 | 999851681931974600766994 |
| 74 | 2305129188866501774481545 |
| 75 | 5315847675735178072941600 |
| 76 | 12262083079763320881047944 |
| 77 | 28292248892584567512609357 |
| 78 | 65294907440089718078048829 |
| 79 | 150729070403767032817820543 |
| 80 | 348031015577337732605480908 |

FIGURE 5. *Natural size:* numbers of closed affine normal forms of size $n$ from 0 to 80