# Quantitative aspects of linear and affine closed lambda terms

PIERRE LESCANNE, University of Lyon, École normale supérieure de Lyon, LIP (UMR 5668, CNRS, ENS Lyon, UCBL, INRIA), France

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 variable occurs once and only once. In this paper we count the number of affine closed $\lambda$-terms of size $n$, linear closed $\lambda$-terms of size $n$, affine closed $\beta$-normal forms of size $n$ and linear closed $\beta$-normal forms of size $n$, for several measures of 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 a specific data structure, made of contexts in which one counts all the holes at each level of abstractions by $\lambda$'s.

CCS Concepts: • **Theory of computation** → Linear logic; Type theory; Generating random combinatorial structures;

Additional Key Words and Phrases: lambda calculus, combinatorics, functional programming

**ACM Reference Format:**

Pierre Lescanne. 2017. Quantitative aspects of linear and affine closed lambda terms. *ACM Trans. Comput. Logic* 1, 1, Article 1 (January 2017), 18 pages. https://doi.org/10.1145/3173547

## 1 INTRODUCTION

The $\lambda$-calculus [1] is a well known formal system designed by Alonzo Church [9] 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).[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 are no free variables, i.e., only bound variables. An *affine $\lambda$-term* (or BCK term) is a $\lambda$-term in which bound variables occur at most once. A *linear* $\lambda$-term (or BCI 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 [12], 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 as shown by Hindley [18, 19]. For instance, generated by the program of Section 9, here are the 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

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

[1] If the reader is not familiar with the $\lambda$-calculus, we advise her (him) to read the introduction of [16], for instance.

Author's address: Pierre Lescanne, University of Lyon, École normale supérieure de Lyon, LIP (UMR 5668, CNRS, ENS Lyon, UCBL, INRIA), 46 allée d'Italie, 69364, Lyon, France, pierre.lescanne@ens-lyon.fr.

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

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.

## 2 RELATED WORKS

The idea of counting structures in logic started from works of Marek Zaionc and his co-authors who studied quantitative aspects of propositions in several logics [20, 24, 25]. For instance, they got the amazing result that asymptotically almost all classical propositions are actually intuitionistic. In other words, at the limit the proportion of truly classical propositions among all the propositions is negligible. Since counting propositions yields interesting results, this suggested that proofs (i.e., $\lambda$-terms in the perspective of Curry-Howard correspondence) should also be considered quantitatively and this led David, Raffalli, Theyssier, Grygiel, Kozik and Zaionc [11] to address asymptotic behaviour using *variable size* 0 measure (see below), where only the tree structure of the $\lambda$-terms (abstractions and applications) matters for the size. Gittenberger et al. [5] proposed *variable size* 1 measure where also variables are counted for one, with no consideration on how far they are bound. For this, they count 1-2-trees (Motzkin trees) enriched by adding directed edges (pointers). The idea that terms should be counted using de Bruijn indices was proposed by the author in [21] in the same variable size 1 framework. Then counting with de Bruijn indices in the variable size 0 framework was considered by Grygiel and Lescanne in [15]. Actually, despite they seemed to fit well with affine and linear terms, it appeared that variable size 0 and variable size 1 measures loose interesting features of the $\lambda$-terms in general, especially they do not account for the distance between the bound variables and their binder. Too many terms are not discriminated by their size and have the same size. For this reason, the asymptotic growth of the number of terms w.r.t. their size is super-exponential and the nice theory of analytic function cannot apply and the efficient method of random generation, called Boltzmann sampler as well, see Lescanne [22] and Bendkowski, Grygiel and Tarau [4]. The first approach departing from variable size 0 and variable size 1 was an idea of John Tromp [23] based on a representation of $\lambda$-terms as bit strings, the so called *binary $\lambda$-calculus* of Grygiel and Lescanne [16]. Then it appears that the measure can be simplified and made more natural, yielding the so called *natural size* of Bendkowski et al. [2, 3]. See Gittenberger and Gołębiewski [14] for a synthetic view of both natural size and binary size. Unlike counting linear and affine closed $\lambda$-terms, counting general closed $\lambda$-terms is rather complex and indirect. Closed general $\lambda$-terms are the $p = 0$ case of $p$-open terms, where $p$-open terms are $\lambda$-terms that require $p$ $\lambda$'s to be closed. The equation defining generating function for counting the $p$-open terms uses the generating function for counting the $p + 1$-open terms which is an unusual induction. Therefore specific techniques of analytic combinatorics have been devised by Bodini, Gittenberger and Gołębiewski [8].

Meanwhile, works started on counting, with variable size 1 or variable size 0, linear closed $\lambda$-terms (called BCI) by Bodini et al. [7], and affine closed $\lambda$-terms (called BCK) by Zeilberger [26–28]. Bodini, Gardy, and Jacquot [6] and Grygiel et al. [17] study both closed and affine $\lambda$-terms. To express the integer sequences of numbers counting terms of the same size, those approaches use generating functions computer algebra software computations and none proposes explicit inductive formulas on the coefficients. Moreover, the natural size is not addressed.

Notice that, from the results of this paper, new sequences A287141, A281270 and A287045 have been entered in the *On-line Encyclopedia of Integer Sequences.*

## 3 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 $(m_0, \ldots, m_{p-1})$. In Section 7, we consider also infinite tuples. Thus $\mathbf{m} \in \mathbb{N}^\omega$ is the sequence $(m_0, m_1, \ldots)$. Notice in the case of infinite tuples, we are only interested in infinite tuples equal to 0 after some index.

- $p$ is the *length* of $\mathbf{m}$, which we write also $\mathsf{length}\ \mathbf{m}$.
- The $p$-tuple $(0, \ldots, 0)$ is written $0^p$. $0^\omega$ is the infinite tuple 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

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

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

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

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

## 4 SWISSCHEESE

The basic concept is that of **m-SwissCheese** or **Swisscheese of characteristic m** or simply **SwissCheese** if there is no ambiguity on $\mathbf{m}$. An $\mathbf{m}$-SwissCheese or a SwissCheese of characteristic $\mathbf{m}$, where $\mathbf{m}$ is of length $p$, is a $\lambda$-term with holes at $p$ levels, which are all counted, using $\mathbf{m}$. Holes at level $i$ are written $\square_i$. An $\mathbf{m}$-SwissCheese contains holes $\square_0$,... $\square_{p-1}$. A hole $\square_i$ is meant to be a location for a variable at level $i$, that is under $i$ $\lambda$'s. Beside holes a SwissCheese contains variables, represented by de Bruijn indices. De Bruijn indices are written $\underline{n}$ (a now traditional notation) or $S^n 0$. In this paper, we prefer the notation $S^n 0$ because it shows all the symbols ($S$ and 0) which may contribute to the size, as it is the case for the natural size. The variables (or the de Bruijn indices) are bound and each binder binds at most one variable, like in closed and affine $\lambda$-terms. A Swisscheese can be seen as an affine or a linear closed $\lambda$-terms to which holes have been added. According to the way bound variables are inserted when creating abstractions (see below), we create linear or affine SwissCheeses. The holes have size 0. An $\mathbf{m}$-SwissCheese or a SwissCheese of characteristic $\mathbf{m}$ has $m_0$ holes at level 0, $m_1$ holes at level 1, ... $m_{p-1}$ holes at level $p-1$. 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, $\mathsf{length}\ \mathbf{m} \leq \mathsf{length}\ \mathbf{m}'$, $m_i = m'_i$ for $i \leq \mathsf{length}\ \mathbf{m}$, and $m'_i = 0$ for $i > \mathsf{length}\ \mathbf{m}$. This statement holds also for $\mathbf{m}' \in \mathbb{N}^\omega$. $l_{n,0^n}$ (resp. $a_{n,0^n}$) counts the linear closed (resp. the closed affine) $\lambda$-terms of size $n$, since it counts SwissCheeses with no hole.

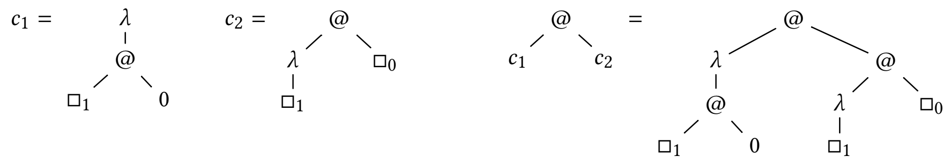


Fig. 1. Building a SwissCheese by application

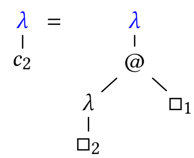


Fig. 2. Abstracting a SwissCheese with no binding

### 4.1 Growing a SwissCheese

Given two SwissCheeses, we can build a SwissCheese *by application* like in Figure 1. In Figure 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. Said otherwise, $c_1$ has characteristic $(0, 1, 0, 0, 0)$, $c_2$ has characteristic $(1, 1, 0, 0, 0)$ and $c_1@c_2$ has characteristic $(1, 2, 0, 0, 0)$. According to what we said, $c_1@c_2$ has characteristic $(1, 2)$ as well as characteristic $(1, 2, 0, 0, \ldots)$ (a tuple starting with 1, followed by 2, followed by infinitely many 0's). We could also say that $c_1$ has characteristic $(0, 1)$ and $c_2$ has characteristic $(1, 1)$ making @ a binary operation on SwissCheeses of length 2 whereas previously we have made @ a binary operation on SwissCheeses of length 5. In other words, when counting SwissCheeses of characteristic $\mathbf{m}$, the trailing 0's are irrelevant. In actual computations, we make the lengths of characteristics consistent by adding trailing 0's to too short ones.

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 $\square_i$ becomes a hole $\square_{i+1}$. $\lambda c$ is a $(0 : \mathbf{m})$-SwissCheese. See Figure 2. 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, we colour the added $\lambda$ in blue and we call it *abstraction with <u>no</u> binding*.

(2) In the second method for growing a SwissCheese by abstraction, we select first a hole $\square_i$, we top the SwissCheese by a $\lambda$, we increment the levels of the other holes and we replace the chosen hole by $S^i 0$, i.e., by the de Bruijn index $\underline{i}$. In Figure 3 we colour the added $\lambda$ in green and we call it *abstraction <u>with</u> binding*.

### 4.2 Measuring SwissCheese

We consider 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.

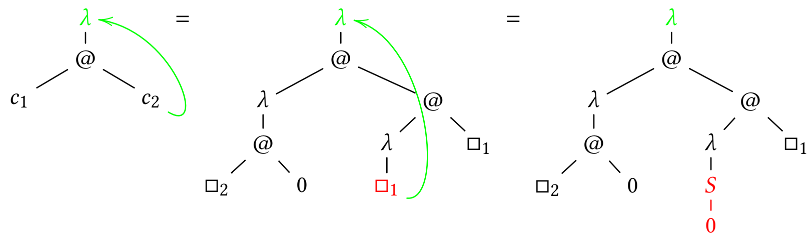

Fig. 3. Abstracting a SwissCheese with binding

- 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**.

## 5 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. This is for specific cases when the part of the formula we describe does not depend of the measure of the size.

### 5.1 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, i.e, which specifies the characteristics of the SwissCheese. In other words we are interested in the quantity $l^{\nu}_{n,\mathbf{m}}$. We assume that the length of $\mathbf{m}$ is $p$, greater than $n$.

**Size is 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 $\mathbf{m} = (1, 0, 0, \ldots)$:

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

**Size is n + 1 and application** If a SwissCheese of size $n + 1$ has holes set by $\mathbf{m}$ and is an application, then it is obtained from a SwissCheese of size $k$ with holes set by $\mathbf{q}$ and a SwissCheese of size $n - k$ with holes set by $\mathbf{r}$, with $\mathbf{m} = \mathbf{q} \oplus \mathbf{r}$:

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

**Size is n + 1 and abstraction with binding** Consider a level $i$, that is a level of hole $\Box_i$. If one wants to get a SwissCheese of size $n + 1$ by abstraction with binding, first this SwissCheese must be a 0:$\mathbf{m}$-SwissCheese (there is no hole at level 0 in this SwissCheese), second the SwissCheese in which one chooses the hole $\Box_i$ is a $\mathbf{m}^{\uparrow i}$-SwissCheese, since one removes a hole $\Box_i$. Therefore, there are $m_i + 1$ ways to choose a hole $\Box_i$. In this hole we put a term $S^{i-1}\,0$ of size $i$. Hence, among $l^{\nu}_{n,0:\mathbf{m}}$ SwissCheeses, there are $(m_i + 1)\, l^{\nu}_{n-i,\mathbf{m}^{\uparrow i}}$ 0:**m**-SwissCheeses which are abstractions with binding in which a $\square_i$ has been replaced by the de Bruijn index $S^{i-1}\,0$. Hence by summing over $i$, the part of abstraction with binding contributes to $l^{\nu}_{n+1,0:\mathbf{m}}$ as:

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

The subtle case of abstraction with binding is pictured in Figure 3. It works as follows. Consider the case where the $(0,1,1)$-SwissCheese $\lambda(\lambda(\square_2\, 0)\,((\lambda S0)\,\square_1))$ is obtained from the $(1,2)$-SwissCheese $\lambda(\square_1\, 0)\,((\lambda \square_1)\,\square_0)$ of Figure 1 (right) by abstraction with binding. Notice that $(0,1,1) = 0 : (1,1)$ and that $(1,2) = (1,1)^{\uparrow 1}$. Focus on level 1 in $\lambda(\square_1\, 0)\,((\lambda \square_1)\,\square_0)$. There are 2 holes at level 1, then 2 ways to choose a hole $\square_1$ at level 1, because 2 is the second index of $(1,2)$, which corresponds to level 1. Assume we choose the second hole from the left, the one in red. Put a (green) lambda on the top. Because of this lambda on the top, raise the levels of the other holes (the leftmost one becomes $\square_2$, the rightmost one becomes $\square_1$). Then replace the chosen hole $\square_1$ by $S\,0$. We get $\lambda(\lambda(\square_2\, 0)\,((\lambda S0)\,\square_1))$.

We have the following recursive definitions of $l^{\nu}_{n+1,\mathbf{m}}$:

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

Numbers of linear closed $\lambda$-terms with natural size are given in Figure 4.

### 5.2 Variable size 0

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

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

The sequence $l^{0}_{n,0^n}$ of the numbers of linear closed $\lambda$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.

### 5.3 Variable size 1

The inserted de Bruijn index has size 1. We have $(m_i + 1)\, l^{1}_{n-1,\mathbf{m}}$ where we had $(m_i + 1)\, l^{\nu}_{n-i,\mathbf{m}}$ for natural size.

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

As noticed by Grygiel et al. [17] (Section 6.1), 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*.

## 6 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 $a^{\nu}_{n,\mathbf{m}}$, $a^{0}_{n,\mathbf{m}}$ and $a^{1}_{n,\mathbf{m}}$, and $a_{n,\mathbf{m}}$ when the size does not matter. There are $a_{n,m}$ $(0 : \mathbf{m})$-SwissCheeses of size $n$ that are abstractions with no binding. We get the recursive formulas:

### 6.1 Natural size

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

The numbers of affine closed size with natural size are given in Figure 5. Since the sequence was unknown in the *On-line Encyclopedia of Integer Sequences* we entered it under the number A287141.

### 6.2 Variable size 0

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

The sequence $a^{0}_{n,0^{\omega}}$ of the numbers of affine closed terms for variable size 0 is

$$0, 1, 2, 8, 29, 140, 661, 3622, 19993, 120909, 744890, 4887401, 32795272, \ldots$$

From our work, the sequence as been entered by Gheorghe Coserea as A287045 in the *On-line Encyclopedia of Integer Sequences*. It corresponds to the coefficients of the generating function $\mathcal{A}(z,0)$ where

$$\mathcal{A}(z,u) = u + z(\mathcal{A}(z,u))^2 + z\frac{\partial \mathcal{A}(z,u)}{\partial u} + z\mathcal{A}(z,u).$$

### 6.3 Variable size 1

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

The sequence $a^{1}_{n,0^{\omega}}$ of the numbers of affine closed terms for variable size 1 is

$$0, 0, 1, 2, 3, 9, 30, 81, 242, 838, 2799, 9365, 33616, 122937, 449698, 1696724, 6558855, \ldots$$

From our work, the sequence as been entered by Gheorghe Coserea as A281270 in the *On-line Encyclopedia of Integer Sequences*. It corresponds to the coefficient of the generating function $\hat{\mathcal{A}}(z,0)$ where $\hat{\mathcal{A}}(z,u)$ is the solution of the functional equation:

$$\hat{\mathcal{A}}(z,u) = zu + z(\hat{\mathcal{A}}(z,u))^2 + z\frac{\partial\hat{\mathcal{A}}(z,u)}{\partial u} + z\hat{\mathcal{A}}(z,u).$$

Notice that this corrects the wrong assumptions of [17] (Section 6.2), which refers actually to sequence A073950 which starts with $1,2,3,9,30,81$, but the $7^{\text{th}}$ of A073950 is 225 instead of 242.

## 7 GENERATING FUNCTIONS

Flajolet and Sedgewick start the preface of their famous book [13] by the following sentence:

> *ANALYTIC COMBINATORICS aims at predicting precisely the properties of large structured combinatorial configurations, through an approach based extensively on analytic methods. Generating functions are the central objects of study of the theory.*

Recall (see [13] Appendix A.5) that generating functions or formal power series extend the notion of polynomials to infinite series of the form:

$$f = \sum_{n\geq 0} f_n z^n.$$

They are used for studying integer sequences counting structures. The idea of considering functions like $\mathcal{F}(z,\mathbf{u})$ for linear and affine $\lambda$-terms is due to Bendkowski, Bodini, Dovgal and Grygiel (private communication) and is not really familiar in this framework.

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 $\mathbf{u}$ stand for the infinite sequences of variables $(u_0, u_1, \ldots)$ and $\mathbf{u}^{\mathbf{m}}$ stands for $(u_0^{m_0}, u_1^{m_1}, \ldots, u_n^{m_n}, \ldots)$ and $\mathsf{tail}\,(\mathbf{u})$ stand for $(u_1, \ldots)$. 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 *linear SwissCheeses* for natural size:

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

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

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

$\mathcal{L}^\nu(z,\mathbf{u})$ is solution of the equation:

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

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

For *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 linear closed $\lambda$-terms. $\mathcal{A}^\nu(z, \mathbf{u})$ is the double series associated with $A^\nu_{\mathbf{m}}(z)$ and is solution of the equation:

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

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

**Variable size** 0

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

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

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

For *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:

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

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

**Variable size** 1

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

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

Then we get as associated double series :

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

## 8 EFFECTIVE COMPUTATIONS

In this section we present Haskell programs for effectively computing the values of the sequences counting affine and linear $\lambda$-terms of size $n$. We are able to compute values for natural size up to 100 on a desk computer with a Pentium(R) Dual-Core at $2.8GHz$.

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 upBound for $n$. We use a recursive data structure:

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

in which we store the computed values of a function

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

In our implementation the depth of the recursion of `Mem` is limited by upBound, 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 [a 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 `a` and `acc` have the same signature. This is not a coincidence, since `acc` accesses values of `a` already computed. Now we are ready to express `a`:

```
a 0 m = iv (head m == 1 && all ((==) 0) (tail m))
a n m = aAPP n m + aABSwB n m + aABSnB n m
```

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

```
aAPP n m = sum (map (\textbackslash((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 $k + nk = n$. `aABSwB` *counts affine terms that are abstractions with binding.*

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

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

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

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

```
aABSnB 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.

## 9 GENERATING AFFINE AND LINEAR TERMS

In this section we present Haskell programs for effectively generating all affine $\lambda$-terms and all linear $\lambda$-terms of size $n$. We can use those programs to generate affine or linear $\lambda$-terms of a given size.

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 (\textbackslash((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
```

From this, we get programs for generating random affine closed $\lambda$-terms or random linear closed $\lambda$-terms as follows: if we want a random linear closed $\lambda$-term of a given size $n$, we throw a random number, say $p$, between 1 and $l_{n,0^n}$ and we look for the $p^{th}$ in the list of all the linear closed $\lambda$-terms of size $n$. Haskell laziness mimics the unranking. Due to high requests in space, we cannot go further than the random generation of linear closed $\lambda$-terms of size 23 and affine closed $\lambda$-terms of size 19 . There are similar programs for generating all the terms of size $n$ for variable size 0 and variable size 1.

## 10 NORMAL FORMS

Normal forms are $\lambda$-terms with no $\beta$-redex, i.e., with no $\lambda$-term of the form $(\lambda M)\,P$. In this section, we are interested with counting and generating normal forms among affine or linear $\lambda$-terms.

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.

### 10.1 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

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

and

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

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 with $anf^{\nu}_{n,\mathbf{m}}$ and $anf^{\nu}_{n,\mathbf{m}}$:

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

*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

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

and

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

with the two generating functions:

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

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

$$\lambda\lambda\lambda\lambda(2\ \lambda((1\ 2)\ \lambda(0\ (5\ 1))))\quad \lambda(0\ \lambda\lambda(1\ \lambda\lambda((0\ (2\ \lambda\lambda((1\ \lambda 0)\ 0)))\ 1)))\quad \lambda((0\ \lambda 0)\ \lambda\lambda((0((1\ \lambda 0)\lambda\lambda(1\ (0\ \lambda 0))))\lambda 0))$$

### 10.2 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

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

and the two generating functions:

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

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

$$0, 1, 0, 3, 0, 26, 0, 367, 0, 7142, 0, 176766, 0, 5304356, \ldots$$

mentioned by Zeilberger in [26] and listing the coefficients of the generating function $\mathcal{L}^{nf,0}(z, 0^\omega)$.

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

### DATA

In this Figure 4, 5 and 6 we give the first values of $l^\nu_{n,0^n}$, $a^\nu_{n,0^n}$, and $anf^\nu_{n,0^n}$.

| | | | |
|---|---|---|---|
| 0 | 0 | 51 | 51496022711337536 |
| 1 | 0 | 52 | 124591137939086496 |
| 2 | 1 | 53 | 299402908258405410 |
| 3 | 0 | 54 | 721839933329222924 |
| 4 | 0 | 55 | 1747307145272084192 |
| 5 | 3 | 56 | 4211741383777966592 |
| 6 | 2 | 57 | 10165998012602469888 |
| 7 | 0 | 58 | 24620618729658655936 |
| 8 | 16 | 59 | 59482734150603634286 |
| 9 | 24 | 60 | 143764591607556354344 |
| 10 | 8 | 61 | 348379929166234350008 |
| 11 | 117 | 62 | 843169238563254723200 |
| 12 | 252 | 63 | 2040572920613086128400 |
| 13 | 180 | 64 | 4948102905207104837424 |
| 14 | 1024 | 65 | 11992521016286173712196 |
| 15 | 2680 | 66 | 29059897435554891991144 |
| 16 | 2952 | 67 | 70516464312280927105392 |
| 17 | 10350 | 68 | 171105110698292441423968 |
| 18 | 29420 | 69 | 415095704639682396539232 |
| 19 | 42776 | 70 | 1008016383720573882885792 |
| 20 | 116768 | 71 | 2448305474519849567597826 |
| 21 | 335520 | 72 | 5945721872300885649415632 |
| 22 | 587424 | 73 | 14449388516068567845838736 |
| 23 | 1420053 | 74 | 35125352062243788817753856 |
| 24 | 3976424 | 75 | 85382289240293493116120064 |
| 25 | 7880376 | 76 | 207650379931166057815603296 |
| 26 | 18103936 | 77 | 505172267243918348155299780 |
| 27 | 48816576 | 78 | 1229005880128485245247395000 |
| 28 | 104890704 | 79 | 2991079243470267667831893408 |
| 29 | 237500826 | 80 | 7281852742753184123608419712 |
| 30 | 617733708 | 81 | 17729171587798767750815341440 |
| 31 | 1396750576 | 82 | 43177454620325445122944305984 |
| 32 | 3171222464 | 83 | 105185452787117035266315446868 |
| 33 | 8014199360 | 84 | 256273862465425158211948020048 |
| 34 | 18688490336 | 85 | 624527413292252904584121980208 |
| 35 | 42840683418 | 86 | 1522355057007327280427270436480 |
| 36 | 106063081288 | 87 | 3711429775030704772089070886624 |
| 37 | 251769197688 | 88 | 9050041253711022076275958636128 |
| 38 | 583690110208 | 89 | 22073150301758857110072042919800 |
| 39 | 1425834260080 | 90 | 53844910909398928990641101351664 |
| 40 | 3417671496432 | 91 | 131371135544173914537076774932576 |
| 41 | 8007221710652 | 92 | 320588677238085642820920910555968 |
| 42 | 19404994897976 | 93 | 782465218885869813183863213231424 |
| 43 | 46747189542384 | 94 | 1910077425906069707804966102543936 |
| 44 | 110498345360800 | 95 | 4663586586924802791117231052636349 |
| 45 | 266679286291872 | 96 | 11388259565942452837717688743953504 |
| 46 | 644021392071840 | 97 | 27813754361897984543467478917223008 |
| 47 | 1533054190557133 | 98 | 67941781284113201998645699501746176 |
| 48 | 3693823999533360 | 99 | 165989485724048964272023600773271424 |
| 49 | 8931109667692464 | 100 | 405588809305168453963137377442321728 |
| 50 | 21375091547312128 | | |

Fig. 4. *Natural size:* numbers of linear closed $\lambda$-terms of size $n$ from 0 to 100


## 11 ACKNOWLEDGEMENT

The basic idea of this work comes from a discussion with Maciej Bendkowski, Olivier Bodini, Sergey Dovgal and Katarzyna Grygiel, I thank them as I thank Noam Zeilberger for interactions, Marek Zaionc for constant support and for initiating the field and the referees for a careful reading and useful suggestions.

| | |
|---|---|
| 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 |

Fig. 5. *Natural size:* numbers of affine closed $\lambda$-terms of size $n$ from 0 to 100

## 12 CONCLUSION

This presentation shares similarities with those of [4, 15, 16]. Instead of considering the size $n$ and the bound $m$ of free indices like in expressions of the form:

$$T_{n+1,m} = T_{n,m+1} + \sum_{i=0}^{n} T_{i,m} T_{n-i,m}$$

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

Fig. 6. *Natural size:* numbers of affine closed normal forms of size $n$ from 0 to 80

here we replace $m$ by the characteristic $\mathbf{m}$. As suggested by Dan Dougherty, we can imagine a common framework. On the other hand, as noticed by Paul Tarau, this approach has features of dynamic programming [10], which makes it somewhat efficient.

## REFERENCES


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

[2] Bendkowski, M., Grygiel, K., Lescanne, P., and Zaionc, M. (2016a). Combinatorics of $\lambda$-terms: a natural approach. *ArXiV*, **abs/1609.07593**.

[3] Bendkowski, M., Grygiel, K., Lescanne, P., and Zaionc, M. (2016b). A natural counting of lambda terms. In R. M. Freivalds, G. Engels, and B. 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.

[4] Bendkowski, M., Grygiel, K., and Tarau, P. (2017). Boltzmann samplers for closed simply-typed lambda terms. In Y. Lierler and W. 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.

[5] Bodini, O., Gardy, D., and Gittenberger, B. (2011). Lambda terms of bounded unary height. In *Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics*, pages 23–32.

[6] Bodini, O., Gardy, D., and Jacquot, A. (2013a). Asymptotics and random sampling for BCI and BCK lambda terms. *Theor. Comput. Sci.*, **502**, 227–238.

[7] Bodini, O., Gardy, D., Gittenberger, B., and Jacquot, A. (2013b). Enumeration of generalized BCI lambda-terms. *Electr. J. Comb.*, **20**(4), P30.

[8] Bodini, O., Gittenberger, B., and Gołębiewski, Z. (2017). Enumerating lambda terms by weighted length of their de bruijn representation. *ArXiV*, **abs/1707.02101**.

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

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition.

[11] David, R., Raffalli, C., Theyssier, G., Grygiel, K., Kozik, J., and Zaionc, M. (2009). Some properties of random lambda terms. *Logical Methods in Computer Science*, **9**(1).

[12] de Bruijn, N. G. (1972). 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.

[13] Flajolet, P. and Sedgewick, R. (2008). *Analytic Combinatorics.* Cambridge University Press.

[14] Gittenberger, B. and Gołębiewski, Z. (2016). On the number of lambda terms with prescribed size of their de Bruijn representation. In N. Ollinger and H. 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.

[15] Grygiel, K. and Lescanne, P. (2013). Counting and generating lambda terms. *J. Funct. Program.*, **23**(5), 594–628.

[16] Grygiel, K. and Lescanne, P. (2015). Counting and generating terms in the binary lambda calculus. *J. Funct. Program.*, **25**.

[17] Grygiel, K., Idziak, P. M., and Zaionc, M. (2013). How big is BCI fragment of BCK logic. *J. Log. Comput.*, **23**(3), 673–691.

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

[19] Hindley, J. R. (1997). *Basic Simple Type Theory.* Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

[20] Kostrzycka, Z. and Zaionc, M. (2004). Statistics of intuitionistic versus classical logics. *Studia Logica*, **76**(3), 307–328.

[21] Lescanne, P. (2013). On counting untyped lambda terms. *Theor. Comput. Sci.*, **474**, 80–97.

[22] Lescanne, P. (2014). Boltzmann samplers for random generation of lambda terms. *ArXiV*, **abs/1404.3875**.

[23] Tromp, J. (2006). Binary lambda calculus and combinatory logic. In M. Hutter, W. Merkle, and P. M. B. Vitányi, editors, *Kolmogorov Complexity and Applications*, volume 06051 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

[24] Zaionc, M. (2005). On the asymptotic density of tautologies in logic of implication and negation. *Reports on Mathematical Logic*, **39**, 67–87.

[25] Zaionc, M. (2006). Probability distribution for simple tautologies. *Theor. Comput. Sci.*, **355**(2), 243–260.

[26] Zeilberger, N. (2015). Counting isomorphism classes of $\beta$-normal linear lambda terms. *ArXiV*, **abs/1509.07596**.

[27] Zeilberger, N. (2016). Linear lambda terms as invariants of rooted trivalent maps. *J. Funct. Program.*, **26**, e21.

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