Combinatorics of Explicit Substitutions

Maciej Bendkowski
Theoretical Computer Science Department
Faculty of Mathematics and Computer Science
Jagiellonian University
ul. Prof. Łojasiewicza 6, 30-348 Kraków, Poland.
maciej.bendkowski@tcs.uj.edu.pl

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

ABSTRACT

λv is an extension of the λ-calculus which internalises the calculus of substitutions. In the current paper, we investigate the combinatorial properties of λv focusing on the quantitative aspects of substitution resolution. We exhibit an unexpected correspondence between the counting sequence for λv-terms and famous Catalan numbers. As a by-product, we establish effective sampling schemes for random λv-terms. We show that typical λv-terms represent, in a strong sense, non-strict computations in the classic λ-calculus. Moreover, typically almost all substitutions are in fact suspended, i.e. unevaluated, under closures. Consequently, we argue that λv is an intrinsically non-strict calculus of explicit substitutions. Finally, we investigate the distribution of various redexes governing the substitution resolution in λv and investigate the quantitative contribution of various substitution primitives.

CCS CONCEPTS

• **Theory of computation → Equational logic and rewriting**; *Abstract machines*; • **Mathematics of computing** → *Combinatorics*;

ACM Reference Format:
Maciej Bendkowski and Pierre Lescanne. 2018. Combinatorics of Explicit Substitutions. In *The 20th International Symposium on Principles and Practice of Declarative Programming (PPDP '18), September 3–5, 2018, Frankfurt am Main, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3236950.3236951

1 INTRODUCTION

Substitution of terms for variables forms a central concept in various formal calculi with qualifiers, such as predicate logic or different variants of λ-calculus. Though substitution supports the computational character of β-reduction in λ-calculus, it is usually specified as an external meta-level formalism, see [5]. Such an epitheoretic presentation of substitution masks its execution as a single, indivisible calculation step, even though it requires a considerable computational effort to carry out, see [37]. In consequence, the number of β-reduction steps required to normalise a λ-term does not reflect the actual operational cost of normalisation. In order to effectuate *substitution resolution*, i.e. the substitution execution, its process needs to be decomposed into a series of fine-grained atomic rewriting steps included as part of the considered calculus.

An early example of such a calculus, internalising the evaluation of substitution, is combinatory logic [18]; alas, bearing the price of loosing the intuitive, high-level structure of encoded functions, mirrored in the polynomial blow-up of their representation, see [30, 31]. Focusing on retaining the basic intuitions behind substitution, various calculi of explicit substitutions highlighting multiple implementation principles of substitution resolution in λ-calculus were proposed throughout the years, cf. [1, 20, 34, 40]. The formalisation of substitution evaluation as a rewriting process provides a formal platform for operational semantics of reduction in λ-calculus using abstract machines, such as, for instance, the Krivine machine [17]. Moreover, with the internalisation of substitution, reduction cost reflects more closely the true computational complexity of executing modern functional programs.

Nonetheless, due to the numerous nuances regarding the evaluation cost of functional programs (e.g. assumed reduction or strictness strategies) supported by a general tradition of considering computational complexity in the framework of Turing machines or RAM models rather than formal calculi, the evaluation cost in term rewriting systems, such as λ-calculus or combinatory logic, gains increasing attention only quite recently, see [2, 3, 33]. The continuing development of automated termination and complexity analysers for both first- and higher-order term rewriting systems echoes the immense practical, and hence also theoretical, demand for complexity analysis frameworks of declarative programming languages, see e.g. [25, 43]. In this context, the computational analysis of first-order term rewriting systems seems to most accurately reflect the practical evaluation cost of declarative programs [14, 16]. Consequently, the average-case performance analysis of abstract rewriting machines executing the declared computations requires a quantitative analysis of their internal calculi. Such investigations provide not only key insight into the quantitative aspects of basic rewriting principles, but also allow to optimise abstract rewriting machines so to reflect the quantitative contribution of various rewriting primitives.

Despite their apparent practical utility, quantitative aspects of term rewriting systems are not well studied. In [15] Choppy, Kaplan and Soria provide a quantitative evaluation of a general class of confluent, terminating term rewriting systems in which the term reduction cost (i.e. the number of rewriting steps required to reach the final normal form) is independent of the assumed

Maciej Bendkowski was partially supported within the Polish National Science Center grant 2016/21/N/ST6/01032.

PPDP'18, September 2018, Frankfurt am Main, Germany

ACM ISBN 978-1-4503-6441-6/18/09…$15.00
https://doi.org/10.1145/3236950.3236951

normalisation strategy. Following a similar, analytic approach, Dershowitz and Lindenstrauss provide an average-time analysis of inference parallelisation in logic programming [21]. More recently, Bendkowski, Grygiel and Zaionc analyse quantitative aspects of normal-order reduction of combinatory logic terms and estimate the asymptotic density of normalising combinators [7, 11]. Alas, due to the intractable, epitheoretic formalisation of substitution in untyped λ-calculus, its quantitative rewriting aspects have, to our best knowledge, not yet been investigated. Some asymptotic aspects of β-reduction in the simply-typed λ-calculus, however, are considered in [41] where it is shown that typically almost all λ-terms of order k have a β-reduction sequence of length $(k-2)$-fold exponential in the term size.

In the following paper we offer a combinatorial perspective on substitution resolution in λ-calculus and propose a combinatorial analysis of explicit substitutions in $\lambda\upsilon$-calculus [34]. The paper is structured as follows. In Section 2 we draft the basic characteristics of $\lambda\upsilon$ required for the reminder of the current paper. Next, we introduce the necessary analytic toolbox used in the quantitative analysis, see Section 3. In Section 4 we enumerate $\lambda\upsilon$-terms and exhibit the declared correspondence between their corresponding counting sequence and Catalan numbers. Some statistical properties of random $\lambda\upsilon$-terms are investigated in Section 5. In 5.1 we relate the typical form of $\lambda\upsilon$-terms with the classic, epitheoretic substitution tactic of λ-calculus. The quantitative impact of substitution suspension is investigated in 5.2. In 5.3 we discuss the contribution of various substitution resolution primitives. The final Section 6 concludes the paper.

2 PRELIMINARIES

2.1 $\lambda\upsilon$-calculus

In the current subsection we outline the main characteristics of $\lambda\upsilon$-calculus (lambda-upsilon calculus) required for the reminder of the paper. We refer the curious reader to [6, 34] for a more detailed exposition.

Remark. We choose to outline $\lambda\upsilon$-calculus following the presentation of [35] where indices start with $\underline{0}$ instead of [6, 34] where de Bruijn indices start with $\underline{1}$, as introduced by de Bruijn himself, cf. [19]. Although both conventions are assumed in the context of static, quantitative aspects of λ-calculus, the former convention seems to be the most recent standard, cf. [8, 10, 13, 26–28].

The computational mechanism of β-reduction is usually defined as $(\lambda x.a)b \to_\beta a[x := b]$ where the right-hand side $a[x := b]$ denotes the epitheoretic, capture-avoiding substitution of term b for variable x in a. $\lambda\upsilon$-calculus [34] is a simple, first-order rewriting system internalising substitution resolution of classic λ-calculus in de Bruijn notation [19]. Its expressions, called *($\lambda\upsilon$)-terms*, consist of indices $\underline{0}, \underline{1}, \ldots$ (for convenience also referred to as *variables*), abstractions and term application. Terms are also equipped with a new *closure* operator $[s]$ denoting the ongoing resolution of substitution s. Explicit substitutions are fragmented into atomic primitives of *shift*, denoted as $\uparrow$, a unary operator *lift*, written as $\Uparrow$, mapping substitutions onto substitutions, and finally a unary *slash* operator, denoted as $/$, mapping terms onto substitutions. Terms containing closures are called *impure* whereas terms without them are said to be *pure*. De Bruijn indices are encoded using a unary base expansion. In other words, $\underline{n}$ is represented as an n-fold application of the successor operator S to zero. Figure 1b summarises the specification of $\lambda\upsilon$-terms.

The rewriting rules of $\lambda\upsilon$-calculus consist of the usual β-rule, specified in this framework as $(\lambda a)b \to a[b/]$ and an additional set of (seven) rules governing the resolution of explicit substitutions, see Figure 1a. Remarkably, these few rewriting rules are sufficient to correctly model β-reduction and also preserve strong normalisation of closed λ-terms, see [6]. The simple syntax and rewriting rules of $\lambda\upsilon$-calculus are not only of theoretical importance, but also of practical interest, used as the foundation of various reduction engines. Let us mention that $\lambda\upsilon$-calculus and its abstract U-machine executing (strong) β-normalisation was successfully used as the main reduction engine in Pollack's implementation of LEGO, a proof checker of the Calculus of Constructions, the Edinburgh Logical Framework, and also for the Extended Calculus of Constructions [38].

Example 2.1. Consider the term $S = \lambda x.\lambda y.\lambda z.xz(yz)$. Note that in the de Bruijn notation, S is written as $\lambda\lambda\lambda\underline{2}\underline{0}(\underline{1}\underline{0})$. Likewise, the term $K = \lambda x.\lambda y.x$ is denoted as $\lambda\lambda\underline{1}$. Certainly, $Kab \to_\beta^+ a$ for each term a. Note however, that with explicit substitution resolution in $\lambda\upsilon$, this reduction is fragmented into several reduction steps, as follows:

$$\begin{aligned}(\lambda\lambda\underline{1})a &\to (\lambda\underline{1})[a/]\\ &\to \lambda(\underline{1}[\Uparrow(a/)])\\ &\to \lambda\left(\underline{0}[a/][\uparrow]\right)\\ &\to \lambda\left(a[\uparrow]\right).\end{aligned}$$

Note that in the final reduction step of the above we obtain $a[\uparrow]$. The additional shift operator guarantees that (potential) free indices are aptly incremented so to avoid potential variable captures. If a is closed, i.e. each variable in a is bound, then $a[\uparrow]$ resolves simply to a, as intended.

3 ANALYTIC TOOLBOX

We base our quantitative analysis of $\lambda\upsilon$-terms on techniques borrowed from analytic combinatorics, in particular singularity analysis developed by Flajolet and Odlyzko [23]. We refer the unfamiliar reader to [24, 42] for a thorough introduction to (multivariate) generating functions and analytic combinatorics.

Remark. Our arguments follow standard applications of singularity analysis to (multivariate) system of generating functions corresponding to algebraic structures. For the reader's convenience we offer a high-level, though limited to the subject of our interest, outline of this process in the following section.

3.1 Singularity analysis

Interested in the quantitative properties of $\lambda\upsilon$-terms, for instance their asymptotic enumeration or parameter analysis, we typically take the following general approach. We start the analysis with establishing a formal, unambiguous context-free specification describing the structures of our interest. Next, using symbolic

$$
\begin{aligned}
(\lambda a)b &\to a[b/] && \text{(Beta)}\\
(ab)[s] &\to a[s](b[s]) && \text{(App)}\\
(\lambda a)[s] &\to \lambda(a[\Uparrow(s)]) && \text{(Lambda)}\\
\underline{0}[a/] &\to a && \text{(FVar)}\\
(\mathrm{S}\,\underline{n})[a/] &\to \underline{n} && \text{(RVar)}\\
\underline{0}[\Uparrow(s)] &\to \underline{0} && \text{(FVarLift)}\\
(\mathrm{S}\,\underline{n})[\Uparrow(s)] &\to \underline{n}[s][\uparrow] && \text{(RVarLift)}\\
\underline{n}[\uparrow] &\to \mathrm{S}\,\underline{n}. && \text{(VarShift)}
\end{aligned}
$$

(a) Rewriting rules.

$$
\begin{aligned}
\mathcal{T} &::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \mathcal{T}[\mathcal{S}]\\
\mathcal{S} &::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow\\
\mathcal{N} &::= \underline{0} \mid \mathrm{S}\,\mathcal{N}.
\end{aligned}
\tag{1}
$$

(b) Terms of $\lambda\upsilon$-calculus.[a]

[a]Notice that de Bruijn indices are encoded in unary base, using a successor operator S.

Figure 1: The $\lambda\upsilon$-calculus rewriting system.

methods [24, Part A. Symbolic Methods] we convert the specification into a system of generating functions, i.e. formal power series $F(z) = \sum_{n\geq 0} a_n z^n$ in which the coefficient a_n standing by z^n, written as $[z^n]F(z)$, denotes the number of structures (objects) of size n. Interested in parameter analysis, so obtained generating functions become bivariate and take the form $F(z,u) = \sum_{n,k\geq 0} a_{n,k} z^n u^k$ where $a_{n,k}$, also written as $[z^n u^k]F(z,u)$, stands for the number of structures of size n for which the investigated parameter takes value k; for instance, $a_{n,k}$ denotes the number of terms of size n with exactly k occurrences of a specific redex pattern. In this context, variable z corresponds to the size of specified structures whereas u is said to *mark* the investigated parameter quantities.

When the obtained system of generating functions admits an analytic solution (i.e. obtained formal power series are also analytic at the complex plane origin) we can investigate the quantitative properties of respective coefficient sequences, and so also enumerated combinatorial structures, by examining the analytic properties of associated generating functions. For that purpose, we focus on the location of their singularities, i.e. complex points onto which the respective generating functions cannot be analytically continued. The location of their nearest singularities (sometimes referred to as *dominant singularities*) dictates the main, exponential growth rate factor of the investigated coefficient sequence.

Theorem 3.1 (Exponential growth formula [24, Theorem IV.7])**.** If $A(z)$ is analytic at the origin and R is the modulus of a singularity nearest to the origin in the sense that

$$R = \sup\{r \geq 0 \,:\, A(z) \text{ is analytic in } |z| < r\},$$

then the coefficient $a_n = [z^n]A(z)$ satisfies

$$a_n = R^{-n}\theta(n) \quad \text{with} \quad \limsup |\theta(n)|^{\frac{1}{n}} = 1.$$

Generating functions considered in the current paper are algebraic, i.e. are branches of polynomial equations in form of $P(z, F(z)) = 0$. Since $\sqrt{z}$ cannot be unambiguously defined as an analytic function near the origin, the main source of singularities encountered during our analysis are roots of radicand expressions involved in the closed-form, analytic formulae defining studied generating functions. The following classic result due to Pringsheim facilities the inspection of such singularities.

Theorem 3.2 (Pringsheim [24, Theorem IV.6])**.** If $A(z)$ is representable at the origin by a series expansion that has non-negative coefficients and radius of convergence R, then the point $z = R$ is a singularity of $A(z)$.

A detailed singularity analysis of algebraic generating functions, involving an examination of the type of dominant singularities follows as a consequence of the Puiseux series expansion for algebraic generating functions.

Theorem 3.3 (Newton, Puiseux [24, Theorem VII.7])**.** Let $F(z)$ be a branch of an algebraic function $P(z, F(z)) = 0$. Then in a circular neighbourhood of a singularity ρ slit along a ray emanating from ρ, $F(z)$ admits a fractional Newton-Puiseux series expansion that is locally convergent and of the form

$$F(z) = \sum_{k\geq k_0} c_k (z-\rho)^{k/\kappa},$$

where $k_0 \in \mathbb{Z}$ and $\kappa \geq 1$.

With available Puiseux series, the complete asymptotic expansion of sub-exponential growth rate factors associated with coefficient sequences of investigated algebraic generating functions can be accessed using the following standard function scale.

Theorem 3.4 (Standard function scale [24, Theorem VI.1])**.** Let $\alpha \in \mathbb{C} \setminus \mathbb{Z}_{\leq 0}$. Then, $f(z) = (1-z)^{-\alpha}$ admits for large n a complete asymptotic expansion in form of

$$[z^n]f(z) = \tfrac{n^{\alpha-1}}{\Gamma(\alpha)}\left(1 + \tfrac{\alpha(\alpha-1)}{2n} + \tfrac{\alpha(\alpha-1)(\alpha-2)(3\alpha-1)}{24n^2} + O\left(\tfrac{1}{n^3}\right)\right)$$

where $\Gamma\colon \mathbb{C} \setminus \mathbb{Z}_{\leq 0} \to \mathbb{C}$ is the Euler Gamma function defined as

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx.$$

3.2 Parameter analysis

Consider a random variable X_n denoting a certain parameter quantity of a (uniformly) random $\lambda\upsilon$-term of size n. In order to analyse the limit behaviour of X_n as n tends to infinity, we utilise the moment techniques of multivariate generating functions [24, Chapter 3]. In particular, if $F(z,u)$ is a bivariate generating function associated with X_n where u marks the considered parameter quantities, then the expectation $\mathbb{E}(X_n)$ takes the form

$$\mathbb{E}(X_n) = \frac{[z^n]\frac{\partial}{\partial u}F(z,u)|_{u=1}}{[z^n]F(z,1)}.$$

Consequently, the limit mean and, similarly, all higher moments can be accessed using techniques of singularity analysis. Although such a direct approach allows to investigate all the limit moments of X_n (in particular its mean and variance) it is usually more convenient to study the associated *probability generating function* $p_n(u)$ instead, defined as

$$p_n(u) = \sum_{k\geq 0} \mathbb{P}(X_n = k)u^k = \frac{[z^n]F(z,u)}{[z^n]F(z,1)}.$$

With $p_n(u)$ at hand, it is possible to readily access the limit distribution of X_n. In the current paper we focus primarily on continuous, Gaussian limit distributions associated with various redexes in λv-calculus. The following Quasi-powers theorem due to Hwang [29] provides means to obtain a limit Gaussian distribution and establishes the rate at which intermediate distributions converge to the final limit distribution.

Theorem 3.5 (Quasi-powers theorem, see [24, Theorem IX.8])**.** Let $(X_n)_{n=1}^{\infty}$ be a sequence of non-negative discrete random variables (supported by $\mathbb{Z}_{\geq 0}$) with probability generating functions $p_n(u)$. Assume that, uniformly in a fixed complex neighbourhood of $u = 1$, for sequences $\beta_n, \kappa_n \to \infty$, there holds

$$p_n(u) = A(u) \cdot B(u)^{\beta_n} \left(1 + O\left(\frac{1}{\kappa_n}\right)\right)$$

where $A(u)$ and $B(u)$ are analytic at $u = 1$ and $A(1) = B(1) = 1$.

Assume finally that $B(u)$ satisfies the following *variability condition*:

$$B''(1) + B'(1) - B'(1)^2 \quad \neq \quad 0. \tag{2}$$

Then, the distribution X_n is, after standardisation, asymptotically Gaussian with speed of convergence of order $O\left(\frac{1}{\kappa_n} + \frac{1}{\sqrt{\beta_n}}\right)$:

$$\mathbb{P}\left(\frac{X_n - \mathbb{E}(X_n)}{\sqrt{\mathbb{V}(X_n)}} \leq x\right) = \Phi(x) + O\left(\frac{1}{\kappa_n} + \frac{1}{\sqrt{\beta_n}}\right)$$

where $\Phi(x)$ is the standard normal distribution function

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\omega^2/2} d\omega\,.$$

The limit expectation and variance satisfy

$$\begin{aligned} \mathbb{E}(X_n) &\sim B'(1)n \\ \mathbb{V}(X_n) &\sim \left(B''(1) + B'(1) - B'(1)^2\right) n \end{aligned} \tag{3}$$

4 COUNTING λv-TERMS

In the current section we begin the enumeration of λv-terms. For that purpose, we impose on them a *size notion* such that the size of a λv-term, denoted as $|\cdot|$, is equal to the total number of constructors (in the associated term algebra, see Figure 1b) of which it is built. Figure 2 provides the recursive definition of term size.

$$\begin{aligned} |\underline{n}| &= n + 1 & |a/| &= 1 + |a| \\ |\lambda a| &= 1 + |a| & |\Uparrow(s)| &= 1 + |s| \\ |ab| &= 1 + |a| + |b| & |\uparrow| &= 1. \\ |a[s]| &= 1 + |a| + |s| \end{aligned}$$

Figure 2: Natural size notion for λv-terms.

Remark. Such a size notion, in which all building constructors contribute equal weight one to the overall term size was introduced in [9] as the so-called *natural size notion*. Likewise, we also refer to the size notion assumed in the current paper as natural.

Certainly, our choice is arbitrary and, in principle, different size measures can be assumed, cf. [9, 26, 28]. For convenience, we choose the natural size notion thus avoiding the obfuscating (though still manageable) technical difficulties arising in the analysis of general size model frameworks, see e.g. [26]. Moreover, our particular choice exhibits unexpected consequences and hence is, arguably, interesting on its own, see Proposition 4.1.

Equipped with a size notion ensuring that for each $n \geq 0$ the total number of λv-terms of size n is finite, we can proceed with our enumerative analysis. Surprisingly, the counting sequence corresponding to λv-terms in the natural size notion corresponds also to the celebrated sequence of Catalan numbers[1].

Proposition 4.1. Let $T(z)$ and $S(z)$ denote the generating functions corresponding to λv-terms
and substitutions, respectively. Then,

$$T(z) = \frac{1 - \sqrt{1 - 4z}}{2z} - 1 \tag{4}$$

whereas

$$S(z) = \frac{1 - \sqrt{1 - 4z}}{2z} \left(\frac{z}{1 - z}\right). \tag{5}$$

In consequence

$$[z^n]T(z) = \begin{cases} 0, & \text{for } n = 0 \\ \dfrac{1}{n+1}\dbinom{2n}{n}, & \text{otherwise} \end{cases} \tag{6}$$

and

$$[z^n]S(z) = \begin{cases} 0, & \text{for } n = 0 \\ \displaystyle\sum_{k=0}^{n-1} \frac{1}{k+1}\binom{2k}{k} & \text{otherwise.} \end{cases}$$

hence also

$$[z^n]T(z) \sim \frac{4^n}{\sqrt{\pi} n^{3/2}}$$

whereas

$$[z^n]S(z) \sim \frac{4^{n+1}}{3\sqrt{\pi} n^{3/2}}.$$

Proof. Consider the formal specification (1) for λv-terms. Let $N(z)$ be the generating function corresponding to de Bruijn indices. Note that following symbolic methods, the generating functions $T(z)$, $S(z)$, and $N(z)$ give rise to the system

$$\begin{aligned} T(z) &= N(z) + zT(z) + zT(z)^2 + zT(z)S(z) \\ S(z) &= zT(z) + zS(z) + z \\ N(z) &= z + zN(z). \end{aligned} \tag{7}$$

Note that $N(z)$ is an independent variable in (7). We can therefore solve the equation $N(z) = z + zN(z)$ and find that $N(z) = \frac{z}{1-z}$. Substituting this expression for $N(z)$ in the equations defining $T(z)$ and $S(z)$ we obtain

$$T(z) = \frac{z}{1 - z} + zT(z) + zT(z)^2 + zT(z)S(z) \tag{8}$$

whereas

$$S(z) = zT(z) + zS(z) + z.$$

[1] see https://oeis.org/A000108.

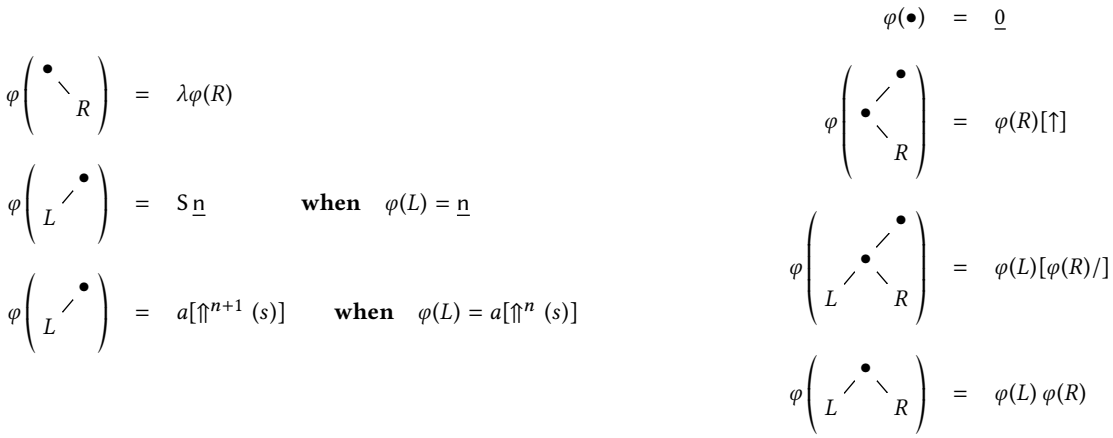


Figure 3: Pictorial representation of the size-preserving bijection φ between $\lambda\upsilon$-terms and plane binary trees.

System (8) admits two solutions, i.e.

$$T(z) = \frac{1 \pm \sqrt{1-4z} - 2z}{2z} \quad \text{and} \quad S(z) = \frac{1 \pm \sqrt{1-4z}}{2(1-z)}, \tag{9}$$

both with agreeing signs.

In order to determine the correct pair of generating functions we invoke the fact that, by their construction, both $[z^n]T(z)$ and $[z^n]S(z)$ are non-negative integers for all $n \geq 0$. Consequently, the declared pair (4) and (5) is the analytic solution of (9). At this point, we notice that both the generating functions in (4) and (5) resemble the famous generating function $C(z) = \frac{1-\sqrt{1-4z}}{2z}$ corresponding to Catalan numbers, see e.g. [42, Section 2.3]. Indeed

$$T(z) = \frac{1-\sqrt{1-4z}}{2z} - 1 \tag{10}$$

whereas

$$S(z) = \frac{1-\sqrt{1-4z}}{2z}\left(\frac{z}{1-z}\right).$$

In this form, we can readily relate Catalan numbers with respective coefficients of $T(z)$ and $S(z)$, see (6). From (10) we obtain $T(z) = C(z) - 1$. The number $[z^n]T(z)$ corresponds thus to $[z^n]C(z)$ for all $n \geq 1$ with the initial $[z^0]T(z) = 0$. Furthermore, given $S(z) = C(z)\frac{z}{1-z}$ we note that $[z^n]S(z)$ corresponds to the partial sum of Catalan numbers[2] up to n (exclusively). □

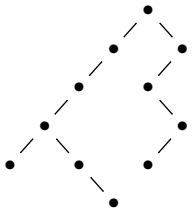
The correspondence exhibited in Proposition 4.1 witnesses the existence of a bijection between $\lambda\upsilon$-terms of size n and, for instance, plane binary trees with n inner nodes. In what follows we provide an alternative, constructive proof of this fact.

4.1 Bijection between $\lambda\upsilon$-terms and plane binary trees

Let $\mathcal{B}$ denote the set of plane binary trees (i.e. binary trees in which we distinguish the order of subtrees). Consider the map $\varphi\colon \mathcal{B} \to \mathcal{T}$ defined as in Figure 3. Note that, for convenience, we omit drawing leaves. Consequently, nodes in Figure 3 with a single or no subtrees are to be understood as having implicit leaves attached to vacuous branches.

Given a tree T as input, φ translates it to a corresponding $\lambda\upsilon$-term $\varphi(T)$ based on the shape of T (performing a so-called pattern matching). This shape, however, might be determined through a recursive call to φ, see the second and third rule of the left-hand size of Figure 3.

Example 4.2. For instance the tree corresponding to the $\lambda\upsilon$-term $\underline{0}[\Uparrow(\lambda\underline{0}/)]\ \underline{1}[\uparrow]$ is:



Proposition 4.3. The map $\varphi\colon \mathcal{B} \to \mathcal{T}$ is a bijection preserving the size of translated structures. In other words, given a tree T with n inner nodes $\varphi(T)$ is a $\lambda\upsilon$-term of size n.

Proof. The fact that φ is size-preserving follows as a direct examination of the rules defining φ, see Figure 3. A straightforward structural induction certifies that all translation rules keep the size of both sides equal.

In order to prove that φ is one-to-one and onto, we note that each maximal (in the sense that it cannot be further continued) sequence of successive left branches has to terminate in either a single leave, hence corresponding to a de Bruijn index through φ, a single right turn, see the second top rule of the right-hand side of Figure 3, or a branching point consisting of a left and right turn, see the third rule of the right-hand side of Figure 3. A final structural induction finishes the proof. □

With a computable map $\varphi\colon \mathcal{B} \to \mathcal{T}$ it is now possible to translate plane binary trees to corresponding $\lambda\upsilon$-terms in linear time in the size of the binary tree. Composing φ with effective samplers (i.e. computable functions constructing random, conditioned on

[2] see https://oeis.org/A014137.

size, structures) for the former, we readily obtain effective samplers for random $\lambda\upsilon$-terms.

We offer a Coq proof of Proposition 4.3 together with a certified Haskell implementation of φ in an external repository[3] with supplementary materials to the current paper.

Remark. Using Rémy's elegant sampling algorithm [39] constructing uniformly random, conditioned on size, plane binary trees of given size n with φ provides a linear time, exact-size sampler for $\lambda\upsilon$-terms. For a detailed presentation of Rémy's algorithm, we refer the curious reader to [4, 32]. Additional combinatorial parameters, such as for instance the number of specific redex sub-patterns in sampled terms, can be controlled using the tuning techniques of [12] developed within the general framework of Boltzmann samplers [22] and the exact-size sampling framework of the so-called recursive method [36].

5 STATISTICAL PROPERTIES OF RANDOM $\lambda\upsilon$-TERMS

In the current section we focus on quantitative properties of random terms. We start our quest with properties of explicit substitutions within $\lambda\upsilon$-calculus. In what follows, we investigate the proportion of $\lambda\upsilon$-terms representing intermediate steps of substitution in classic λ-calculus.

5.1 Strict substitution forms

When a β-rule is applied and $(\lambda x.a)b$ is rewritten to $a[x := b]$ the meta-level substitution of b for variable x in a is executed somewhat outside of the calculus. In operational terms, the substitution $a[x := b]$ is meant to be resolved ceaselessly and cannot be, for instance, suspended or even (partially) omitted if it produces a dispensable result. Such a resolution tactic is reflected in $\lambda\upsilon$-calculus in terms of the following notion of strict substitution forms.

Definition 5.1. A $\lambda\upsilon$-term t is in *strict substitution form* if there exist two pure (i.e. without explicit substitutions) terms a, b and a sequence $t_1, \ldots, t_n$ of $\lambda\upsilon$-terms such that

$$a[b/] \to t_1 \to \cdots \to t_n = t$$

and none of the above reductions is (Beta).

Otherwise, t is said to be in *lazy substitution form*.

In other words, strict substitution forms represent the intermediate computations of resolving substitutions in the classic λ-calculus. Certainly, by design $\lambda\upsilon$-calculus permits more involved resolution tactics, mixing for instance *Beta*-reduction and υ-reductions. In the following proposition we show that the proportion of terms representing the indivisible, classic resolution tactic tends to zero with the term size tending to infinity. Therefore, typical terms in $\lambda\upsilon$-calculus do not match the strict, epitheoretic substitution in the classical λ-calculus, but rather represent an ongoing, non-strict substitution tactic.

Proposition 5.2. Asymptotically almost all $\lambda\upsilon$-terms are in lazy substitution form.

Proof. We argue that the set of strict substitution forms is asymptotically negligible in the set of all $\lambda\upsilon$-terms. Consequently, its complement, i.e. lazy substitution forms, admits an asymptotic density one, as claimed.

Consider the class of $\lambda\upsilon$-terms containing nested substitutions, i.e. subterms in form of $a[b/]$ where b is impure. Note that if a term contains nested substitutions, then it cannot be in strict substitution form. Let us therefore estimate the asymptotic density of terms without nested substitutions. Following the combinatorial specification (1) for $\lambda\upsilon$-terms we can write down the following specification for $\overline{\mathcal{T}}$ using the auxiliary classes $\overline{\mathcal{S}}$ of (restricted) substitutions, and $\mathcal{P}$ of pure $\lambda\upsilon$-terms:

$$\begin{aligned} \overline{\mathcal{T}} &::= \mathcal{N} \mid \lambda\overline{\mathcal{T}} \mid \overline{\mathcal{T}}\,\overline{\mathcal{T}} \mid \overline{\mathcal{T}}[\overline{\mathcal{S}}] \\ \overline{\mathcal{S}} &::= \mathcal{P}/ \mid \Uparrow(\overline{\mathcal{S}}) \mid \uparrow \\ \mathcal{P} &::= \mathcal{N} \mid \lambda\mathcal{P} \mid \mathcal{P}\mathcal{P}. \end{aligned} \tag{11}$$

Note that (11) is almost identical to (1) except for the fact that we permit only pure terms under the slash operator in the definition of $\overline{\mathcal{S}}$. We can now apply symbolic methods and establish a corresponding system of generating functions:

$$\begin{aligned} \overline{T}(z) &= N(z) + z\overline{T}(z) + z\overline{T}(z)^2 + z\overline{T}(z)\overline{S}(z) \\ \overline{S}(z) &= zP(z) + z\overline{S}(z) + z \\ P(z) &= N(z) + zP(z) + zP(z)^2. \end{aligned} \tag{12}$$

Solving (12) for $\overline{T}(z)$ we find that

$$\overline{T}(z) = \frac{1 - z - z\overline{S}(z) - \sqrt{\left(1 - z - z\overline{S}(z)\right)^2 - \frac{4z^2}{1-z}}}{2z} \tag{13}$$

whereas

$$\overline{S}(z) = \frac{z + zP(z)}{1 - z} \quad \text{and} \quad P(z) = \frac{1 - z - \sqrt{(1-z)^2 - \frac{4z^2}{1-z}}}{2z}. \tag{14}$$

Note that both $\overline{S}(z)$ and $P(z)$ share a common, unique dominant singularity $\rho \doteq 0.295598$ being the smallest positive root of the radicand expression $(1-z)^2 - \frac{4z^2}{1-z}$ in the defining formula of $P(z)$, see (14). Moreover, due to the presence of the expression $z\overline{S}(z)$ in the numerator of (13) ρ is also a singularity of $\overline{T}(z)$. Denote the radicand expression of (13) as $R(z)$. Note that

$$\frac{d}{dz}R(z) = -2\left(1 - z - z\overline{S}(z)\right)\left(1 + \overline{S}(z) + z\frac{d}{dz}\overline{S}(z)\right) - \frac{4z^2}{(1-z)^2} - \frac{8z}{1-z}. \tag{15}$$

Since $\overline{S}(z)$ is a generating function with non-negative integer coefficients both $\overline{S}(z)$ and its derivative $\frac{d}{dz}\overline{S}(z)$ are positive in the interval $z \in (0, \rho)$. Therefore, the derivative $\frac{d}{dz}R(z)$ is negative for values $z \in (0, 1)$ satisfying $1 - z - z\overline{S}(z) \geq 0$. A direct computation verifies that $0 < z \leq \rho$ satisfy this condition. Consequently, $R(z)$ is decreasing in the interval $z \in (0, \rho)$.

At this point we note that $R(0) = 1$ whereas $R(\frac{1}{4}) > 0$. It follows therefore that $R(z)$ has no roots in the interval $(0, \frac{1}{4})$. Since the generating function $T(z)$ corresponding to all (unrestricted) $\lambda\upsilon$-terms has a single dominant singularity $\zeta = \frac{1}{4}$, see (4) and (5), a straightforward application of the exponential growth formula (see Theorem 3.1) reveals that $\lambda\upsilon$-terms without nested substitutions

[3] see https://github.com/maciej-bendkowski/combinatorics-of-explicit-substitutions.

are asymptotically negligible in the set of all $\lambda\upsilon$-terms. So is, as well, its subset of strict substitution forms. □

5.2 Suspended substitutions

Closures in $\lambda\upsilon$-terms are intended to represent suspended, unevaluated substitutions in the classic λ-calculus. In other words, substitutions whose resolution is meant to be carried out in a non-strict manner. It the current section we investigate the quantitative impact of this suspension on random terms.

Definition 5.3. Let s be a substitution and t be a $\lambda\upsilon$-term. Then, s, all its subterms and, all the constructors it contains are said to be *suspended in* t if t contains a subterm in form of $[s]$; in other words, when s occurs under a closure in t.

In the following proposition we show that, in expectation, almost all of the term content (i.e. represented computation) is suspended under closures.

Proposition 5.4. Let X_n be a random variable denoting the number of constructors not suspended under a closure in a random $\lambda\upsilon$-term of size n. Then, the expectation $\mathbb{E}(X_n)$ satisfies

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{316}{3}. \tag{16}$$

Proof. Let $T(z,u)$ be a bivariate generating function where $[z^n u^k]T(z,u)$ denotes the number of $\lambda\upsilon$-terms of size n with k constructors not suspended under a closure. In other words, the number of $\lambda\upsilon$-terms of size n in which suspended substitutions are of total size $n-k$.

Given the specification (1) for $\mathcal{T}$ we introduce a new variable u and mark with it constructors which do not occur under closures. Note that, in doing so, we no not mark $\mathcal{T}$ recursively in $\mathcal{S}$. Following symbolic methods we note that $T(z,u)$ satisfies

$$T(z,u) = \frac{zu}{1-zu} + zuT(z,u) + zuT(z,u)^2 + zuT(z,u)S(z). \tag{17}$$

Taking the derivative ∂u at $u=1$ of both sides of (17) we arrive at

$$\begin{aligned}\frac{\partial}{\partial u}T(z,u)|_{u=1} \quad = \quad & \frac{z}{(1-z)^2} + T(z,1)\Big(z + zT(z,1) + zS(z)\Big) \\ & + \frac{\partial}{\partial u}T(z,u)|_{u=1}\Big(z + 2zT(z,1) + zS(z)\Big)\end{aligned}$$

and hence

$$\frac{\partial}{\partial u}T(z,u)|_{u=1} = \frac{\frac{z}{(1-z)^2} + T(z,1)\Big(z + zT(z,1) + zS(z)\Big)}{1 - z - 2zT(z,1) - zS(z)}. \tag{18}$$

From (4) and (5) we note that both $T(z)$ and $S(z)$ admit Puiseux expansions in form of $\alpha - \beta\sqrt{1-4z} + O\left(\left|1-4z\right|\right)$ for some appropriate (different for $T(z)$ and $S(z)$) constants $\alpha, \beta > 0$. Consequently, both the numerator and denominator of (18) admit Puiseux expansions of similar form. Furthermore, as

$$\frac{a - b\sqrt{1-4z} + O\left(\left|1-4z\right|\right)}{c - d\sqrt{1-4z} + O\left(\left|1-4z\right|\right)} = \left(\frac{ad}{c^2 - 4d^2z + d^2} - \frac{bc}{c^2 - 4d^2z + d^2}\right)\sqrt{1-4z} + O\left(\left|1-4z\right|\right)$$

we conclude that

$$\frac{\partial}{\partial u}T(z,u)|_{u=1} = \gamma - \delta\sqrt{1-4z} + O\left(\left|1-4z\right|\right)$$

near $z = \frac{1}{4}$ for some (computable) constants $\gamma, \delta > 0$.

An application of the standard function scale (see Theorem 3.4) provides now the asymptotic estimate

$$\mathbb{E}(X_n) = \frac{[z^n]\frac{\partial}{\partial u}T(z,u)|_{u=1}}{[z^n]T(z,1)} \xrightarrow[n\to\infty]{} C. \tag{19}$$

A direct calculation gives the specific quantity (16) of C. □

Let us note that the above result marks yet another crucial difference between λ-calculus and $\lambda\upsilon$-calculus. In the former, substitutions are carried out somewhat outside of the language whereas in the latter formalism they are not only internalised, but typically constitute almost all of the term content.

5.3 Substitution resolution primitives

The internalisation of substitution in $\lambda\upsilon$-calculus introduces several new types of redexes governing the resolution of closures, see Figure 1a. Instead of a single β-redex, specific implementations of the $\lambda\upsilon$-calculus rewriting system, such as for instance the abstract U-machine, have to handle eight rewriting rules together with their intricate interaction.

In the current section we investigate the distribution of specific redexes in random $\lambda\upsilon$-terms, providing insight in the quantitative contribution of various substitution resolution primitives. Since all redexes share virtually the same proof scheme, for convenience, we provide detailed arguments only for the (Beta) rule. Remaining proofs are merely sketched.

5.3.1 *(Beta) redexes.*

Proposition 5.5. Let X_n be a random variable denoting the number of β-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{3}{64}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{153}{4096}n \tag{20}$$

Proof. Routinely, we begin our considerations with establishing a combinatorial specification for corresponding β-redexes. Note that given the specification $\mathcal{T}$ for general $\lambda\upsilon$-terms (1) we can rewrite the left operand in the $(\mathcal{T}\mathcal{T})$ production using the recursive specification for $\mathcal{T}$. Consequently, we obtain the following modified combinatorial specification:

$$\begin{aligned}\mathcal{T} &::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}[\mathcal{S}] \mid \overbrace{\mathcal{N}\mathcal{T} \mid \underline{(\lambda\mathcal{T})\mathcal{T}} \mid (\mathcal{T}[\mathcal{S}])\mathcal{T} \mid (\mathcal{T}\mathcal{T})\mathcal{T}}^{(\mathcal{T}\mathcal{T})} \\ \mathcal{S} &::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow \\ \mathcal{N} &::= \underline{0} \mid \mathrm{S}\,\mathcal{N}.\end{aligned} \tag{21}$$

Note that in this form, specification (21) explicitly uses the production $(\lambda\mathcal{T})\mathcal{T}$ associated with β-redexes. Since the above specification is unambiguous, i.e. each $\lambda\upsilon$-term has precisely one derivation

starting from $\mathcal{T}$, we can further convert it into the following system of corresponding bivariate generating functions marking β-redexes:

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)S(z,u) + zT(z,u)^2 + \\ &\quad (u-1)z^2T(z,u)^2 \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

Note that instead of a direct transformation, we use the, somewhat indirect, $z^2T(z,u)^2\,(u-1)$ expression to denote λv-terms in application form where each β-redex is marked with variable u. It should be understood as follows. First, we count all applications $(\mathcal{T}\mathcal{T})$ by $zT(z,u)^2$. Next, we over-count marked β-redexes by $uz^2T(z,u)^2$. Finally, we remove the over-counted (and unmarked by $zT(z,u)^2$) β-redexes by $-z^2T(z,u)^2$. At this point, we solve the above system and find that the generating function $T(z,u)$ satisfies

$$T(z,u) = \frac{1-z-\frac{z^2}{1-z} - \sqrt{\left(1-z-\frac{z^2}{1-z}\right)^2 - \frac{4z^2\left(1+(u-1)z+\frac{z}{1-z}\right)}{1-z}}}{2z\left(1+(u-1)z+\frac{z}{1-z}\right)}. \tag{22}$$

In this form it is clear that the dominant singularity $\rho(u)$ is carried by the radicand expression of $T(z,u)$, see (22). Furthermore, the singularity $\rho(u)$ is non-constant and *moving*, i.e. varies smoothly with u. Its specific form can be readily accessed by equating the above radicand expression to zero and noting that $\rho(u)$ (see Figure 4) is the only solution satisfying $\lim_{u\to 1}\rho(u) = \frac{1}{4}$ corresponding to the dominant singularity of $T(z,1)$. Consequently, $\rho(u)$ can be analytically continued onto a larger domain containing the point $u = 1$. Let $\rho(u)$ denote, by a slight abuse of notation, this continuation of the definition of the function in Figure 4. It follows that $T(z,u)$ can be uniquely represented as

$$T(z,u) = \alpha(z,u) + \beta(z,u)\sqrt{1-\frac{z}{\rho(u)}}$$

where both $\alpha(z,u)$ and $\beta(z,u)$ are non-vanishing near $(z,u) = (\frac{1}{4},1)$. With u fixed sufficiently close to one, we can now apply the standard function scale (see Theorem 3.4) and obtain the estimate

$$[z^n]T(z,u) = \gamma\left(\frac{1}{\rho(u)}\right)^n \quad \text{with} \quad \gamma = \frac{\beta\left(\rho(u),u\right)}{2\sqrt{\pi}n^{3/2}}.$$

Consequently, the probability generating function $p_n(u)$ satisfies

$$p_n(u) = \frac{[z^n]T(z,u)}{[z^n]T(z,1)} = \overline{\gamma}\left(\frac{\rho(1)}{\rho(u)}\right)^n\left(1+O\left(\frac{1}{n}\right)\right)$$

$$\text{where} \quad \overline{\gamma} = \frac{\beta\left(\rho(u),u\right)}{\beta\left(\rho,1\right)}.$$

Such a form of $p_n(u)$ matches the premises of the Quasi-powers theorem (see Theorem 3.5) taking

$$A(u) = \frac{\beta(\rho(u),u)}{\beta(\rho,1)}, \quad B(u) = \frac{\rho(1)}{\rho(u)} \quad \text{and} \quad \beta_n = \kappa_n = n.$$

Given the explicit formula of Figure 4 for $\rho(u)$ a routine calculation verifies the requested variability condition (2). Consequently, an application of the Quasi-powers theorem finishes the proof. The limit expectation and variance (20) associated with X_n can be computed using formulas (3). □

5.3.2 (App) redexes. In order to mark occurrence of (App) redexes, we take the specification (1) of $\mathcal{T}$ and rewrite the production $\mathcal{T}[\mathcal{S}]$ into four, more detailed ones, including an explicit production for (App) redexes.

$$\begin{aligned} \mathcal{T} &::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\mathcal{N}[\mathcal{S}] \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid \underline{(\mathcal{T}\mathcal{T})[\mathcal{S}]}}^{\mathcal{T}[\mathcal{S}]} \\ \mathcal{S} &::= \mathcal{T} / \mid \Uparrow(\mathcal{S}) \mid \uparrow \\ \mathcal{N} &::= \underline{0} \mid \mathrm{S}\,\mathcal{N}. \end{aligned}$$

A direct translation onto the level of corresponding generation functions gives

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)z^2T(z,u)^2S(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

A detailed analysis provides then the following result.

Proposition 5.6. Let X_n be a random variable denoting the number of (App)-redexes in a random λv-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{32}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{45}{2048}n$$

5.3.3 (Lambda) redexes. The case of (Lambda) redexes is virtually identical to (App) redexes. This time, however, a different production is marked:

$$\begin{aligned} \mathcal{T} &::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\mathcal{N}[\mathcal{S}] \mid \underline{(\lambda\mathcal{T})[\mathcal{S}]} \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]} \\ \mathcal{S} &::= \mathcal{T} / \mid \Uparrow(\mathcal{S}) \mid \uparrow \\ \mathcal{N} &::= \underline{0} \mid \mathrm{S}\,\mathcal{N}. \end{aligned}$$

Accordingly,

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)z^2T(z,u)S(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

Proposition 5.7. Let X_n be a random variable denoting the number of (Lambda)-redexes in a random λv-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{32}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{53}{2048}n$$

5.3.4 (FVar) redexes. The case of (FVar) redexes is a bit more involved and requires three layers of production substitution in order to reach an explicit (FVar) production (see Figure 5).

The corresponding system of generating functions takes then the form

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)z^3T(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

$$\rho(u) = \frac{1}{4}+\frac{1}{2}\sqrt{\frac{\sqrt[3]{u+3}}{2^{2/3}(u-1)^{2/3}}+\frac{1}{4}}-\frac{1}{2}\sqrt{\frac{u+7}{4(u-1)\sqrt{\frac{\sqrt[3]{u+3}}{2^{2/3}(u-1)^{2/3}}+\frac{1}{4}}}-\frac{\sqrt[3]{u+3}}{2^{2/3}(u-1)^{2/3}}+\frac{1}{2}}$$

Figure 4: $\rho(u)$

$$\mathcal{T} ::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\underbrace{\underline{0}[\mathcal{T}/] \mid \underline{0}[\Uparrow(\mathcal{S})] \mid \underline{0}[\uparrow]}_{\underline{0}[\mathcal{S}]} \mid (\mathsf{S}\,\mathcal{N})[\mathcal{S}]}^{\mathcal{N}[\mathcal{S}]} \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]}$$

$$\mathcal{S} ::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow$$

$$\mathcal{N} ::= \underline{0} \mid \mathsf{S}\,\mathcal{N}.$$

Figure 5: The specification associated with $FVar$**.**

Proposition 5.8. Let X_n be a random variable denoting the number of (FVar)-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{3}{256}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{729}{65536}n$$

5.3.5 (RVar) redexes. Similarly to (FVar) redexes, (RVar) redexes require three layers of substitution. The final layer, however, involves now $(\mathsf{S}\,\mathcal{N})[\mathcal{S}]$ (see Figure 6).

When transformed, we obtain the following system of generating functions:

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)\frac{z^4}{1-z}T(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

Proposition 5.9. Let X_n be a random variable denoting the number of (RVar)-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{256}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{249}{65536}n$$

5.3.6 (FVarLift) redexes. The (FVarLift) redex follows the same successive transformation of the initial specification $\mathcal{T}$. When (FVarLift) redexes are obtained and marked, we get the outcome specification given in Figure 7.

Consequently

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)z^3S(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

Proposition 5.10. Let X_n be a random variable denoting the number of (FVarLift)-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{128}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{241}{32768}n$$

5.3.7 (RVarLift) redexes. (RVarLift) redexes are specified analogously to (FVarLift) redexes. The deepest level of transformation involved now $(\mathsf{S}\,\mathcal{N})[\mathcal{S}]$ instead of $\underline{0}[\mathcal{S}]$ as in the case of (FVarLift) redexes. The resulting specification takes the form given in Figure 8. And so, the associated system of generating function becomes

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)\frac{z^4}{1-z}S(z,u) \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

Proposition 5.11. Let X_n be a random variable denoting the number of (RVarLift)-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{384}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{377}{147456}n$$

5.3.8 (VarShift) redexes. The final case of (VarShift) redexes can be approached as before. Marking $\mathcal{N}[\uparrow]$ in the associated specification we obtain the specification of Figure 9.

And so

$$\begin{aligned} T(z,u) &= \frac{z}{1-z} + zT(z,u) + zT(z,u)^2 + zT(z,u)S(z,u) + \\ &\quad (u-1)\frac{z^3}{1-z} \\ S(z,u) &= zT(z,u) + zS(z,u) + z. \end{aligned}$$

$$\mathcal{T} ::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\underbrace{\underline{0}[\mathcal{S}] \mid \underbrace{(\mathsf{S}\,\mathcal{N})[\mathcal{T}/] \mid (\mathsf{S}\,\mathcal{N})[\Uparrow(\mathcal{S})] \mid (\mathsf{S}\,\mathcal{N})[\uparrow]}_{(\mathsf{S}\,\mathcal{N})[\mathcal{S}]}}^{\mathcal{N}[\mathcal{S}]} \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]}$$
$$\mathcal{S} ::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow$$
$$\mathcal{N} ::= \underline{0} \mid \mathsf{S}\,\mathcal{N}.$$

Figure 6: The specification associated with $RVar$.

$$\mathcal{T} ::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\underbrace{\underline{0}[\mathcal{T}/] \mid \underline{0}[\Uparrow(\mathcal{S})] \mid \underline{0}[\uparrow]}_{\underline{0}[\mathcal{S}]} \mid (\mathsf{S}\,\mathcal{N})[\mathcal{S}]}^{\mathcal{N}[\mathcal{S}]} \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]}$$
$$\mathcal{S} ::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow$$
$$\mathcal{N} ::= \underline{0} \mid \mathsf{S}\,\mathcal{N}.$$

Figure 7: The specification associated with $FVarLift$.

$$\mathcal{T} ::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\underbrace{\underline{0}[\mathcal{S}] \mid \underbrace{(\mathsf{S}\,\mathcal{N})[\mathcal{T}/] \mid \underline{(\mathsf{S}\,\mathcal{N})[\Uparrow(\mathcal{S})]} \mid (\mathsf{S}\,\mathcal{N})[\uparrow]}_{(\mathsf{S}\,\mathcal{N})[\mathcal{S}]}}^{\mathcal{N}[\mathcal{S}]} \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]}$$
$$\mathcal{S} ::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow$$
$$\mathcal{N} ::= \underline{0} \mid \mathsf{S}\,\mathcal{N}.$$

Figure 8: The specification associated with $RVarLift$.

$$\mathcal{T} ::= \mathcal{N} \mid \lambda\mathcal{T} \mid \mathcal{T}\mathcal{T} \mid \overbrace{\underbrace{\mathcal{N}[\mathcal{T}/] \mid \mathcal{N}[\Uparrow(\mathcal{S})] \mid \underline{\mathcal{N}[\uparrow]}}_{\mathcal{N}[\mathcal{S}]} \mid (\lambda\mathcal{T})[\mathcal{S}] \mid (\mathcal{T}[\mathcal{S}])[\mathcal{S}] \mid (\mathcal{T}\mathcal{T})[\mathcal{S}]}^{\mathcal{T}[\mathcal{S}]}$$
$$\mathcal{S} ::= \mathcal{T}/ \mid \Uparrow(\mathcal{S}) \mid \uparrow$$
$$\mathcal{N} ::= \underline{0} \mid \mathsf{S}\,\mathcal{N}.$$

Figure 9: The specification associated with $VarShift$.

Proposition 5.12. Let X_n be a random variable denoting the number of (VarShift)-redexes in a random $\lambda\upsilon$-term of size n. Then, after standardisation, X_n converges in law to a Gaussian distribution with speed of convergence of order $O\left(\frac{1}{\sqrt{n}}\right)$. The limit expectation $\mathbb{E}(X_n)$ and variance $\mathbb{V}(X_n)$ satisfy

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{64}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{57}{4096}n$$

Remark. In order to facilitate the intensive calculations involved in obtaining redex distributions, one can use formulas (3) and the implicit form of the singularity $\rho(u)$ defined as a root of an appropriate radicand expression $P(z,u)$ of the corresponding bivariate generating function $T(z,u)$. The quantities $\rho'(1)$ and $\rho''(1)$ can be extracted using implicit derivatives of the equation $P(\rho(u),u) = 0$.

Table 1 outlines the obtained means and variances for all considered $\lambda\upsilon$-redexes.

6 CONCLUSIONS

Our contribution is a step towards the quantitative analysis of substitution resolution and, in particular, the average-case analysis

Table 1: List of redex distributions in descending order of limit means. Specific values are approximated to the sixth decimal point.

Redex	Mean	Variance
(Beta)	$0.046875n$	$0.037354n$
(App)	$0.031250n$	$0.021973n$
(Lambda)	$0.031250n$	$0.025879n$
(VarShift)	$0.015625n$	$0.013916n$
(FVar)	$0.011719n$	$0.011124n$
(FVarLift)	$0.007812n$	$0.007355n$
(RVar)	$0.003906n$	$0.003799n$
(RVarLift)	$0.002604n$	$0.002557n$

of abstract machines associated with calculi of explicit substitutions. Although we focused on $\lambda\upsilon$-calculus, other calculi are readily amenable to similar analysis. Our particular choice is motivated by the relative, compared to other calculi of explicit substitutions, simple syntax of $\lambda\upsilon$. With merely eight rewriting rules, $\lambda\upsilon$ is one of the conceptually simplest calculi of explicit substitutions. Notably, rewriting rules contribute just to the technical part of the quantitative analysis, not its general scheme. Consequently, we expect that investigations into more complex calculi might be more technically challenging, however should not pose significantly more involved issues.

Our quantitative analysis exhibited that typical $\lambda\upsilon$-terms represent, in a strong sense, intrinsically non-strict computations of the classic λ-calculus. Typically, substitutions are not ceaselessly evaluated, but rather suspended in their entirety; almost all of the encoded computation is suspended under closures. Not unexpectedly, on average, the most frequent redex is (Beta). In the υ fragment of $\lambda\upsilon$, however, the most recurrent redexes are, in order, (App) and (Lambda). The least frequent, and at the same time the most intricate redex, is (RVarLift). Let us note that such a diversity of redex frequencies might be exploited in practical implementations. For instance, knowing that specific redexes are more frequent than others, abstract machines might be aptly optimised.

Finally, as an unexpected by-product of our analysis, we exhibited a size-preserving bijection between $\lambda\upsilon$-terms and plane binary trees, enumerated by the famous Catalan numbers. Notably, such a correspondence has practical implications. Specifically, we established an exact-size sampling scheme for random $\lambda\upsilon$-terms based on known samplers for the latter structures. Consequently, it is possible to effectively generate random $\lambda\upsilon$-terms of size n in $O(n)$ time.

ACKNOWLEDGEMENTS

We would like to thank Sergey Dovgal for fruitful discussions and valuable comments, as well as the annonymous reviewers for their suggestions and remarks.

REFERENCES

[1] Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J. (1991). Explicit substitutions. *Journal of Functional Programming*, **1**(4), 375–416.
[2] Accattoli, B. and Lago, U. D. (2016). (leftmost-outermost) beta reduction is invariant, indeed. *Logical Methods in Computer Science*, **12**(1).
[3] Avanzini, M., Lago, U. D., and Moser, G. (2015). Analysing the complexity of functional programs: higher-order meets first-order. In K. Fisher and J. H. Reppy, editors, *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015, Vancouver, BC, Canada.*, pages 152–164. ACM.
[4] Bacher, A., Bodini, O., and Jacquot, A. (2013). Exact-size sampling for motzkin trees in linear time via boltzmann samplers and holonomic specification. In *2013 Proceedings of the Tenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 52–61.
[5] Barendregt, H. P. (1984). *The Lambda Calculus: Its Syntax and Semantics*, volume 103. North Holland, revised edition.
[6] Benaissa, Z.-E.-A., Briaud, D., Lescanne, P., and Rouyer-Degli, J. (1996). $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, **6**(5), 699–722.
[7] Bendkowski, M. (2017). Normal-order reduction grammars. *Journal of Functional Programming*, **27**.
[8] Bendkowski, M. and Lescanne, P. (2018). Counting environments and closures. In H. Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction*, LIPIcs. To appear.
[9] Bendkowski, M., Grygiel, K., Lescanne, P., and Zaionc, M. (2016). A natural counting of lambda terms. In *Theory and Practice of Computer Science: 42nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM*, pages 183–194. Springer Berlin Heidelberg.
[10] Bendkowski, M., Grygiel, K., Lescanne, P., and Zaionc, M. (2017a). Combinatorics of λ-terms: a natural approach. *Journal of Logic and Computation*, **27**(8), 2611–2630.
[11] Bendkowski, M., Grygiel, K., and Zaionc, M. (2017b). On the likelihood of normalization in combinatory logic. *Journal of Logic and Computation*.
[12] Bendkowski, M., Bodini, O., and Dovgal, S. (2018). Polynomial tuning of multiparametric combinatorial samplers. In *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 92–106.
[13] Bodini, O., Gittenberger, B., and Gołębiewski, Z. (2018). Enumerating lambda terms by weighted length of their de Bruijn representation. *Discrete Applied Mathematics*, **239**, 45–61.
[14] Bonfante, G., Cichon, A., Marion, J., and Touzet, H. (2001). Algorithms with polynomial interpretation termination proof. *J. Funct. Program.*, **11**(1), 33–53.
[15] Choppy, C., Kaplan, S., and Soria, M. (1989). Complexity analysis of term-rewriting systems. *Theor. Comput. Sci.*, **67**(2&3), 261–282.
[16] Cichon, A. and Lescanne, P. (1992). Polynomial interpretations and the complexity of algorithms. In D. Kapur, editor, *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*, volume 607 of *Lecture Notes in Computer Science*, pages 139–147. Springer.
[17] Curien, P.-L. (1993). *Categorical Combinators, Sequential Algorithms and Functional*. Birkhaüser, 2nd edition.
[18] Curry, H. B. and Feys, R. (1958). *Combinatory Logic*, volume 1. North-Holland. Second printing 1968.
[19] de Bruijn, N. G. (1972). Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)*, **75**(5), 381–392.
[20] de Bruijn, N. G. (1978). A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report 78-WSK-03, Technological University Eindhoven, Netherlands, Department of Mathematics.
[21] Dershowitz, N. and Lindenstrauss, N. (1989). Average time analyses related to logic programming. In G. Levi and M. Martelli, editors, *Logic Programming, Proceedings of the Sixth International Conference, Lisbon, Portugal, June 19-23, 1989*, pages 369–381. MIT Press.
[22] Duchon, P., Flajolet, P., Louchard, G., and Schaeffer, G. (2004). Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, **13**(4-5), 577–625.
[23] Flajolet, P. and Odlyzko, A. M. (1990). Singularity analysis of generating functions. *SIAM Journal on Discrete Mathematics*, **3**(2), 216–240.
[24] Flajolet, P. and Sedgewick, R. (2009). *Analytic Combinatorics*. Cambridge University Press, 1 edition.
[25] Giesl, J., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., and Thiemann, R. (2014). Proving termination of programs automatically with AProVE. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning*, pages 184–191, Cham. Springer International Publishing.
[26] Gittenberger, B. and Gołębiewski, Z. (2016). On the number of lambda terms with prescribed size of their de Bruijn representation. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS*, pages 40:1–40:13.
[27] Grygiel, K. and Lescanne, P. (2013). Counting and generating lambda terms. *Journal of Functional Programming*, **23**(5), 594–628.
[28] Grygiel, K. and Lescanne, P. (2015). Counting and generating terms in the binary lambda calculus. *Journal of Functional Programming*, **25**.
[29] Hwang, H.-K. (1998). On convergence rates in the central limit theorems for combinatorial structures. *European Journal of Combinatorics*, **19**(3), 329–343.
[30] Joy, M. S. (1984). *On the efficient implementation of combinators as an object code for functional programs*. Ph.D. thesis, University of East Anglia.

[31] Joy, M. S., Rayward-Smith, V. J., and Burton, F. W. (1985). Efficient combinator code. *Computer Languages*, **10**(3-4), 211–224.
[32] Knuth, D. (2006). *The Art of Computer Programming: Generating All Trees–History of Combinatorial Generation*, volume 4. Addison-Wesley Professional.
[33] Lago, U. D. and Martini, S. (2012). On constructor rewrite systems and the lambda calculus. *Logical Methods in Computer Science*, **8**(3).
[34] Lescanne, P. (1994). From $\lambda\sigma$ to $\lambda\upsilon$: A journey through calculi of explicit substitutions. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 60–69. ACM.
[35] Lescanne, P. (1996). The lambda calculus as an abstract data type. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specification*, pages 74–80, Berlin, Heidelberg. Springer Berlin Heidelberg.
[36] Nijenhuis, A. and Wilf, H. S. (1978). *Combinatorial Algorithms*. Academic Press, 2 edition.
[37] Peyton Jones, S. L. (1987). *The Implementation of Functional Programming Languages*. Prentice-Hall, Inc.
[38] Pollack, R. (2018). Personal correspondence.
[39] Rémy, J. (1985). Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *ITA*, **19**(2), 179–195.
[40] Rose, K. H., Bloo, R., and Lang, F. (2012). On explicit substitution with names. *J. Autom. Reasoning*, **49**(2), 275–300.
[41] Sin'Ya, R., Asada, K., Kobayashi, N., and Tsukada, T. (2017). Almost every simply typed λ-term has a long β-reduction sequence. In *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures - Volume 10203*, pages 53–68, New York, NY, USA. Springer-Verlag New York, Inc.
[42] Wilf, H. S. (2006). *Generatingfunctionology*. A. K. Peters, Ltd.
[43] Zankl, H. and Korp, M. (2014). Modular complexity analysis for term rewriting. *Logical Methods in Computer Science*, **10**(1).