COMBINATORICS OF EXPLICIT SUBSTITUTIONS (EXTENDED ABSTRACT)

MACIEJ BENDKOWSKI[†] AND PIERRE LESCANNE[‡]

> *It is clear that, while de Bruijn indices certainly work great as a model for variables as used by abstract machines constrained to a single evaluation strategy, in our opinion it remains unclear when variable names or de Bruijn indices are best suited for* reasoning *about reduction.* K. Rose et al. *in* [RBL12]

1. Introduction

We would like to strengthen Kris Rose and his co-authors statement, by saying that variable names are better suited for *reasoning* about reduction; however, based on the very simple calculus $\lambda\upsilon$, we show how de Bruijn indices are well suited for *counting* certain aspects of reduction.

Despite their apparent practical utility, quantitative aspects of term rewriting systems are not well studied. In [CKS89] 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 normalisation strategy. Following a similar, analytic approach, Dershowitz and Lindenstrauss provide an average-time analysis of inference parallelisation in logic programming [DL89]. 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 [BGZ17; Ben17]. 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. A full version of this paper is in [BL18].

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 [Les94]. Our quantitative analysis of $\lambda\upsilon$-terms is based on techniques borrowed from analytic combinatorics, in particular singularity analysis developed by Flajolet and Odlyzko [FO90]. Using these techniques, we argue that $\lambda\upsilon$ is, in a strong sense, an intrinsically non-strict calculus of explicit substitutions. Typically $\lambda\upsilon$-terms represent non-strict computations and almost all substitutions are in fact suspended. Finally, we investigate the distribution of various redexes governing the substitution resolution in $\lambda\upsilon$ and investigate the quantitative contribution of various substitution primitives. Figure 1 summarises the $\lambda\upsilon$ rewriting system.

Remark. We choose to outline $\lambda\upsilon$-calculus following the presentation of [Les96] where indices start with $\underline{0}$ instead of [Les94; Ben+96] where de Bruijn indices start with $\underline{1}$, as introduced by de Bruijn himself, cf. [Bru72]. 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. [GL13; GL15; Ben+17; GG16].

[†]Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, ul. Prof. Łojasiewicza 6, 30-348 Kraków, Poland.

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

E-mail addresses: maciej.bendkowski@tcs.uj.edu.pl, pierre.lescanne@ens-lyon.fr.

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

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

(A) Rewriting rules.

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

(B) Terms of $\lambda\upsilon$-calculus. Note that de Bruijn indices are encoded in unary base using a successor operator S.

FIGURE 1. The $\lambda\upsilon$-calculus rewriting system.

2. Counting $\lambda\upsilon$-terms

In order to count terms, we impose a *size notion* such that the size of a $\lambda\upsilon$-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.

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

FIGURE 2. Natural size notion for $\lambda\upsilon$-terms.

Such a size notion, in which all building constructors contribute equal weight one to the overall term size was introduced in [Ben+16] as the so-called *natural size notion*. Certainly, our choice is arbitrary and, in principle, different size measures can be assumed, cf. [GL15; Ben+16; GG16]. 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. [GG16].

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

Proposition 2.1. Let T_n and S_n denote the number of $\lambda\upsilon$-terms and substitutions of size n, respectively. Then

$$
T_n = \begin{cases} 0, & \text{for } n = 0 \\ \dfrac{1}{n+1}\dbinom{2n}{n}, & \text{otherwise} \end{cases} \quad \text{and} \quad S_n = \begin{cases} 0, & \text{for } n = 0 \\ \displaystyle\sum_{k=0}^{n-1} \frac{1}{k+1}\binom{2k}{k} & \text{otherwise.} \end{cases} \tag{2}
$$

hence also

$$
T_n \sim \frac{4^n}{\sqrt{\pi}n^{3/2}} \quad \text{whereas} \quad S_n \sim \frac{4^{n+1}}{3\sqrt{\pi}n^{3/2}}. \tag{3}
$$

The correspondence exhibited in Proposition 2.1 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.

2.1. **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 leafs. Consequently, nodes in Figure 3 with a single or no subtrees are to be understood as having implicit leafs attached to vacuous branches.

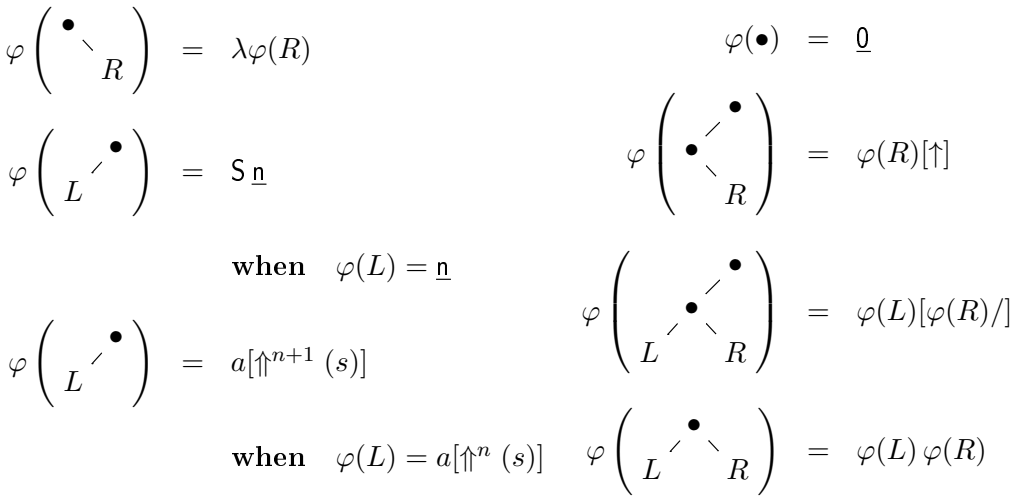

FIGURE 3. Pictorial representation of the size-preserving bijection φ between $\lambda\upsilon$-terms and plane binary trees.

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 on third rule of the left-hand size of Figure 3.

Proposition 2.2. 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.

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, in the size of the binary tree, time. Composing φ with effective samplers (i.e. computable functions constructing random, conditioned on size, structures) for the former, we readily obtain effective samplers for random $\lambda\upsilon$-terms.

Remark. Using Rémy's elegant sampling algorithm [Rém85] 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 [Knu06; BBJ13]. 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 [BBD18] developed within the general framework of Boltzmann samplers [Duc+04] and the exact-size sampling framework of the so-called recursive method [NW78].

3. 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.

3.1. **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 3.1. 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

$$a[b/] \to t_1 \to \cdots \to t_n = t \tag{4}$$

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 what follows we show that the proportion of terms representing the indivisible, classic resolution tactic tends to zero with the term size tending to infinity.

Proposition 3.2. Asymptotically almost all $\lambda\upsilon$-terms are in lazy substitution form.

3.2. **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 3.3. 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 3.4. 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

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{316}{3}. \tag{5}$$

3.3. **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.

Proposition 3.5. 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

$$\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{3}{64}n \quad \text{and} \quad \mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{153}{4096}n \tag{6}$$

Likewise, we obtain similar results for the other rewriting rules.

- **(App)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{32}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{45}{2048}n$
- **(Lambda)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{32}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{53}{2048}n$
- **(FVar)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{3}{256}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{729}{65536}n$
- **(RVar)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{256}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{249}{65536}n$

- **(FVarLift)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{128}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{241}{32768}n$
- **(RVarLift)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{384}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{377}{147456}n$
- For **(VarShift)-redexes** $\mathbb{E}(X_n) \xrightarrow[n\to\infty]{} \frac{1}{64}n$ and $\mathbb{V}(X_n) \xrightarrow[n\to\infty]{} \frac{57}{4096}n$

The following table outlines the obtained means and variances.

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)	$0.002604n$	$0.002557n$

4. Conclusions

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

References

[BBD18] Maciej Bendkowski, Olivier Bodini and Sergey Dovgal. "Polynomial tuning of multiparametric combinatorial samplers". In: *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. 2018, pp. 92–106. DOI: 10.1137/1.9781611975062.9.

[BBJ13] A. Bacher, O. Bodini and A. Jacquot. "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)*. 2013, pp. 52–61. DOI: 10.1137/1.9781611973037.7. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611973037.7.

[Ben+16] Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne and Marek Zaionc. "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*. Springer Berlin Heidelberg, 2016, pp. 183–194.

[Ben+17] Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne and Marek Zaionc. "Combinatorics of λ-terms: a natural approach". In: *Journal of Logic and Computation* 27.8 (2017), pp. 2611–2630. DOI: 10.1093/logcom/exx018.

[Ben+96] Zine-El-Abidine Benaissa, Daniel Briaud, Pierre Lescanne and Jocelyne Rouyer-Degli. "$\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation". In: *Journal of Functional Programming* 6.5 (1996), pp. 699–722. DOI: 10.1017/S0956796800001945.

[Ben17] Maciej Bendkowski. "Normal-order reduction grammars". In: *Journal of Functional Programming* 27 (2017). DOI: 10.1017/S0956796816000332.

[BGZ17] Maciej Bendkowski, Katarzyna Grygiel and Marek Zaionc. "On the likelihood of normalization in combinatory logic". In: *Journal of Logic and Computation* (2017). DOI: 10.1093/logcom/exx005.

[BL18] Maciej Bendkowski and Pierre Lescanne. *Combinatorics of explicit substitutions*. ArXiV. https://arxiv.org/abs/1804.03862. Apr. 2018.

[Bru72] Nicolaas G. de Bruijn. "Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem". In: *Indagationes Mathematicae (Proceedings)* 75.5 (1972), pp. 381–392.

[CKS89] Christine Choppy, Stéphane Kaplan and Michèle Soria. "Complexity Analysis of Term-Rewriting Systems". In: *Theor. Comput. Sci.* 67.2&3 (1989), pp. 261–282.

[DL89] Nachum Dershowitz and Naomi Lindenstrauss. "Average Time Analyses Related to Logic Programming". In: *Logic Programming, Proceedings of the Sixth International Conference, Lisbon, Portugal, June 19-23, 1989*. Ed. by Giorgio Levi and Maurizio Martelli. MIT Press, 1989, pp. 369–381. ISBN: 0-262-62065-0.

[Duc+04] Philippe Duchon, Philippe Flajolet, Guy Louchard and Gilles Schaeffer. "Boltzmann Samplers for the Random Generation of Combinatorial Structures". In: *Combinatorics, Probability and Computing* 13.4-5 (2004), pp. 577–625.

[FO90] Philippe Flajolet and Andrew M. Odlyzko. "Singularity Analysis of Generating Functions". In: *SIAM Journal on Discrete Mathematics* 3.2 (1990), pp. 216–240.

[GG16] Bernhard Gittenberger and Zbigniew Gołębiewski. "On the Number of Lambda Terms With Prescribed Size of Their De Bruijn Representation". In: *33rd Symposium on Theoretical Aspects of Computer Science, STACS*. 2016, 40:1–40:13.

[GL13] Katarzyna Grygiel and Pierre Lescanne. "Counting and generating lambda terms". In: *Journal of Functional Programming* 23.5 (2013), pp. 594–628.

[GL15] Katarzyna Grygiel and Pierre Lescanne. "Counting and generating terms in the binary lambda calculus". In: *Journal of Functional Programming* 25 (2015). DOI: 10.1017/S0956796815000271.

[Knu06] Donald Knuth. *The Art of Computer Programming: Generating All Trees–History of Combinatorial Generation*. Vol. 4. Addison-Wesley Professional, 2006. ISBN: 0321335708.

[Les94] Pierre Lescanne. "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*. ACM, 1994, pp. 60–69.

[Les96] Pierre Lescanne. "The lambda calculus as an abstract data type". In: *Recent Trends in Data Type Specification*. Ed. by Magne Haveraaen, Olaf Owe and Ole-Johan Dahl. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 74–80. ISBN: 978-3-540-70642-7.

[NW78] Albert Nijenhuis and Herbert S. Wilf. *Combinatorial Algorithms*. 2nd ed. Academic Press, 1978.

[RBL12] Kristoffer Høgsbro Rose, Roel Bloo and Frédéric Lang. "On Explicit Substitution with Names". In: *J. Autom. Reasoning* 49.2 (2012), pp. 275–300.

[Rém85] Jean-Luc Rémy. "Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire". In: *ITA* 19.2 (1985), pp. 179–195.