Theoretical Computer Science Department
Faculty of Mathematics and Computer Science
Jagiellonian University

Quantitative aspects and generation of random lambda and combinatory logic terms

Maciej Bendkowski

Ph.D. Thesis
Advisors: Marek Zaionc and Katarzyna Grygiel
Kraków, May 2017

Abstract

We present a quantitative analysis of λ-calculus in the de Bruijn notation and combinatory logic under various combinator bases. Both classes of computational models are shown to share the fixed subterm property – for an arbitrary fixed term T, asymptotically almost all terms contain T as a subterm. In consequence, both models exhibit similar quantitative properties with respect to normalisation and typeability. Specifically, asymptotically almost no term is either strongly normalising or typeable. Furthermore, asymptotically almost no normalising term is simultaneously strongly normalising.

In the context of λ-calculus, we investigate the average de Bruijn index weight within the general Gittenberger-Gołębiewski size notion framework. We show that its mean value tends to a constant as the term size tends to infinity. Moreover, we focus on the so-called natural size notion for which we state several quantitative results. For instance, we identify the corresponding counting sequence of neutral λ-terms with Motzkin trees giving mutually inverse size-preserving translations. As a result, we obtain an exact-size sampler for the former based on the exact-size sampler for Motzkin trees of Bacher, Bodini and Jacquot.

Concerning combinator-specific results, we provide a complete syntactic characterisation of normalising SK-combinators by means of a constructive hierarchy of unambiguous regular tree grammars. As an application, we present an algorithmic technique of finding asymptotically significant fractions of normalising SK-combinators. Utilising our systematic approach, we show that the asymptotic density of normalising combinators cannot be less than 34%. We discuss the limits of our method and, based on super-computer experimental results, discuss the asymptotic density and average complexity of normalising combinators, arguing that the asymptotic density of normalising combinators is approximately equal to 85%.

Finally, we discuss the effective generation of random λ-terms and combinators, focusing on the set of closed and typeable λ-terms. We provide effective Boltzmann samplers for several classes of interesting λ-terms including the restricted class of so-called closed h-shallow λ-terms, i.e. closed λ-terms in which de Bruijn indices are bounded by h. Combining Boltzmann models and logic programming techniques available in modern Prolog systems, we give a sampling scheme for closed typeable λ-terms and discuss the intriguing challenges blocking effective sampling of large closed typeable λ-terms.

Acknowledgements

First and foremost, I would like to express my gratitude to my advisors Marek Zaionc and Katarzyna Grygiel for introducing me to the subject of quantitative aspects of computational logic and their constant support during my research.

I thank the countless (though countable) group of people with whom I had the privilege to work with, discuss problems and learn from. I'm especially indebted to Pierre Lescanne, Paul Tarau, Olivier Bodini, Bernhard Gittenberger, Zbigniew Gołębiewski, Sergey Dovgal and Matthieu Dien.

Finally, I would like to thank ACC Cyfronet AGH in Kraków for granting me access to the super-computer cluster Prometheus to perform numerical experiments with normalising combinators.

Contents

Chapter 1

Introduction

Quantitative investigations in logic where combinatorial aspects and asymptotic behaviour of large 'typical' logical entities are studied, form a prominent branch of modern computer science. The standard approach to attaching formal meaning to the informal notion of 'typical' structure in an infinite denumerable universe Ω is to consider the limit behaviour of probabilities over sets of structures with bounded size. Quite often, appropriate size notions are naturally derived from the syntax of the considered universe, such as the length of strings representing studied structures or the number of nodes in their tree counterparts. In this setting, the probability p_n that a uniformly random structure of size n admits a fixed property P is well-defined. Considering the limit of probabilities p_n with n tending to infinity leads to the notion of asymptotic density – the central tool of quantitative investigations in logic. If the considered limit exists, P is said to have asymptotic density in Ω. Moreover, if its quantity is positive, then P constitutes a property that asymptotically significant fractions of 'typical' structures in Ω possess.

Perhaps the most well-known results of this quantitative flavour in logic are Fagin's 0–1 laws in finite model theory, stating that for a fixed first-order sentence σ the asymptotic fraction of finite relational models satisfying σ exists and is either equal to zero or one [Fag76]. In the language of graphs, a first-order sentence σ holds for almost all finite graphs if and only if σ holds for the famous Rado graph which is known to be a PSPACE-complete problem due to Grandjean [Gra83].

Concerning open problems of quantitative provenance in propositional logic, arguably the most prominent one is the well-known satisfiability threshold hypothesis for k-SAT formulae postulating the existence of a phase transition between satisfiability and unsatisfiability for formulae with m clauses over n variables as the quotient m/n is varied. The threshold is proven to exists for 2-SAT or 3-XORSAT instances, though the intriguing case of 3-SAT remains open (see e.g. [Puy04]).

Over the last two decades a significant effort has been devoted to the problem of estimating the asymptotic fraction of tautologies in different propositional logics including classical and intuitionistic ones (see e.g. [Fou+10; MTZ00]). While the implicational fragments of both the classical and intuitionistic logics are proven to be asymptotically identical by Fournier et al. [Fou+07], remarkably the full propositional systems escape the usual 0–1 law; specifically, Genitrini and Kozik show that asymptotically 5/8 of classical tautologies are also tautologies in intuitionistic logic [GK12].

Connected to propositional logic, random and/or trees over a fixed number of variables and their corresponding Boolean functions were first studied by Chauvin et al. [Cha+04]. Much attention has been dedicated to the asymptotic density of and/or trees representing a given Boolean function (see e.g. [Koz08]) or the complexity of typical Boolean functions, especially with respect to the so-called Shannon effect [GGM14]. We refer the curious reader to Gardy's survey on probability distributions on Boolean functions and their complexity [Gar05].

More recently, a new active stream of quantitative research in λ-calculus and combinatory logic was initiated when Wang explored the uniform random generation of closed λ-terms up to α-equivalence [Wan05]. In her model, Wang assumed a size notion in which all constructors (i.e. abstractions, applications and variables) contribute one to the overall term size. The problem of giving asymptotically accurate approximations on the number of λ-terms of a given size was left open. Later, David et al. investigated a similar, canonical model in which variables do not contribute to the term size, showing that asymptotically almost all λ-terms are strongly normalising [Dav+13]. Contrary to λ-calculus, in the same paper David et al. proved that asymptotically almost no combinatory logic term is strongly normalising, marking a crucial quantitative difference of both computational models. Although the general counting problem has not been resolved, Bodini et al. obtained asymptotic approximations for some restricted classes of closed linear and affine λ-terms [Bod+13].

Later on, different representations of both λ-calculus and combinatory logic were investigated. Lescanne proposed an alternative representation of λ-terms in the de Bruijn notation where variables are replaced with natural indices intended to denote the distance to their binding abstractions [Les13; GL13]. Such a new representation allowed to utilise techniques of analytic combinatorics and, in consequence, embed the generation of λ-terms in the general framework of Boltzmann samplers [Les14]. Independently, Tromp considered a binary encoding of λ-calculus and combinatory logic meant for the construction of compact and efficient self-interpreters for both languages with applications to Kolmogorov complexity [Tro06]. Quantitative aspects of the binary λ-calculus were later studied by Grygiel and Lescanne [GL15].

Somewhat contrary to the canonical λ-calculus representation, Bendkowski et al. proposed a so-called natural size notion of λ-terms in the de Bruijn notation, exhibiting an intriguing quantitative connection between λ-calculus and combinatory logic; assuming their natural size notion, asymptotically almost no λ-term is strongly normalising, as in the case of combinatory logic [Ben+16a]. Various size notions based on the de Bruijn notation, in particular the natural and Tromp's binary size notions, were later generalised under a common size model framework by Gittenberger and Gołębiewski who provided tight lower and upper asymptotic bounds on the number of closed λ-terms [GG16].

In the current dissertation we continue the quantitative investigations in λ-calculus and combinatory logic. In our endeavour we adopt the methods and techniques of analytic combinatorics – a profound approach to quantitative investigations regarding large combinatorial structures, linking the analytic properties of generating functions corresponding to the investigated class of combinatorial structures with their quantitative properties. Necessary preliminary concepts and notions of analytic combinatorics, λ-calculus and combinatory logic are presented in Chapter 2.

In Chapter 3 we investigate the quantitative properties of λ-calculus in the de Bruijn notation within the Gittenberger-Gołębiewski size model framework for which we state several model-independent results with respect to normalisation and typeability. Specifically, we show that λ-terms in this notation have the fixed subterm property, i.e. for an arbitrary fixed term T, asymptotically almost all λ-terms contain T as a subterm. In consequence, λ-terms satisfying various classic properties such as typeability or strong normalisation are asymptotically negligible in the set of all λ-terms. We prove that the set of normalising λ-terms cannot have a trivial 0–1 asymptotic density, hence among the set of normalising λ-terms asymptotically almost no are at the same time strongly normalising. We investigate the average value of a de Bruijn index in a random λ-term, proving that its mean value tends to a constant depending entirely on the assumed size notion.

Subsequently, we focus on the natural size notion. Somewhat unexpectedly, the counting sequence for λ-terms in the natural size notion corresponds also to two families of binary trees – so-called black-white trees and zigzag-free ones. We provide a constructive proof of the former fact by exhibiting appropriate mutually inverse translations. Moreover, we identify the sequence of Motzkin numbers with the counting sequence for neutral λ-terms, giving a bijection which, in consequence, results in an exact-size sampler for the latter based on the exact-size sampler for Motzkin trees of Bacher et al. [BBJ13]. Results in the natural size notion were obtained in collaboration with Grygiel, Lescanne and Zaionc [Ben+16a; Ben+16b].

Finally, we investigate the practical aspects of generating large uniformly random λ-terms, in particular closed and typeable ones. We provide an effective method of generating so-called closed h-shallow λ-terms (i.e. closed λ-terms with de Bruijn indices bounded by h) within the framework of Boltzmann samplers and discuss the random generation of closed typeable λ-terms combining the methods of rejection sampling and logical programming techniques. Results regarding random generation of closed typeable λ-terms were obtained in collaboration with Grygiel and Tarau [BGT16; BGT17].

In Chapter 4 we study the combinatorial structure of normalising SK-combinators. We present an algorithm which, for given n, constructs an unambiguous regular tree grammar defining the set of SK-combinators normalising in precisely n normal-order reduction steps. In light of the famous Curry and Feys's standardisation theorem, our reduction grammars form a complete syntactic characterisation of normalising SK-combinators. We present a recursive method of constructing corresponding ordinary generating functions which we later utilise to investigate the asymptotic density of normalising combinators. Finally, we discuss the number of productions of generated grammars giving a primitive recursive upper bound. Results regarding normal-order reduction grammars are published in [Ben17].

In Chapter 5 we focus on the quantitative properties of combinatory logic. We start with several basis-independent results exhibiting an intriguing quantitative connection between combinatory logic and λ-calculus in the de Bruijn notation. Similarly to the case of λ-terms, we show that combinators exhibit the fixed subterm property. In consequence, it becomes possible to port normalisation and typeability results of λ-calculus directly to the universe of combinatory logic.

Next, we investigate the asymptotic density of normalising SK-combinators. We prove that for each positive n, the set of SK-combinators reducing in n normal-order reduction steps has positive asymptotic density in the set of all combinators. We present an algorithmic approach of constructing asymptotically significant fractions of normalising combinators. As an application of our method, we show that the density of normalising combinators cannot be less than 34%. Finally, we present some super-computer experimental results, arguing that the density of normalising combinators is close to 85%. Results regarding combinatory logic were obtained in collaboration with Grygiel and Zaionc [BGZ15; BGZ17].

In Chapter 6 we conclude the dissertation and discuss remaining open problems.

Chapter 2

Preliminaries

In the following chapter we present the necessary preliminary notions and results concerning analytic combinatorics, λ-calculus and combinatory logic.

We start with the fundamental notion of combinatorial class.

Definition 2.1 (Combinatorial class)**.** Let $\mathcal{A}$ be a countable set of combinatorial structures. Assume that $\mathcal{A}$ is equipped with a size notion $|\cdot|\colon \mathcal{A} \to \mathbb{N}$ assigning each $\alpha \in \mathcal{A}$ its size $|\alpha|$ in such a way that for each non-negative integer n there are finitely many structures of size n in $\mathcal{A}$. Then, $\mathcal{A}$ together with $|\cdot|$ form a combinatorial class.

Following standard notational conventions we use capital calligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$ to denote combinatorial classes. We write $\mathcal{A}_n$ to denote the set of structures in $\mathcal{A}$ of size n and a_n to denote its cardinality. Finally, we call the sequence $(a_n)_{n\in\mathbb{N}}$ the counting sequence of $\mathcal{A}$.

Definition 2.2 (Asymptotic density)**.** Let $\mathcal{A}$ be a combinatorial class contained in an infinite class $\mathcal{B}$. Assume that there exists a non-negative integer N such that for each $n \geq N$ the number b_n of structures in $\mathcal{B}$ of size n is positive. Then, the asymptotic density of $\mathcal{A}$ in $\mathcal{B}$ is defined as

$$\mu\left(\frac{\mathcal{A}}{\mathcal{B}}\right) = \lim_{n\to\infty} \frac{a_{N+n}}{b_{N+n}}\,. \tag{2.1}$$

For given $\mathcal{A}$ and $\mathcal{B}$ it might not be a priori clear if $\mathcal{A}$ has an asymptotic density in $\mathcal{B}$ or what its precise quantity is. Nonetheless, since $a_{N+n} \leq b_{N+n}$ the above fraction sequence has a lower and upper limit; hence, we can consider the lower and upper asymptotic densities defined as

$$\mu^-\left(\frac{\mathcal{A}}{\mathcal{B}}\right) = \liminf_{n\to\infty} \frac{a_{N+n}}{b_{N+n}} \quad \text{and} \quad \mu^+\left(\frac{\mathcal{A}}{\mathcal{B}}\right) = \limsup_{n\to\infty} \frac{a_{N+n}}{b_{N+n}} \tag{2.2}$$

even if $\mathcal{A}$ has no asymptotic density in $\mathcal{B}$.

It is worth noticing that though it might be tempting to interpret asymptotic density as an asymptotic probability, it is not countably additive. Each finite subset of an infinite class $\mathcal{A}$ has a zero asymptotic density in $\mathcal{A}$; in particular, all singletons $\{\alpha_n\}_{n\in\mathbb{N}}$ of structures in $\mathcal{A}$. However, their countable union forms the entire $\mathcal{A}$ which has asymptotic density one in itself.

Definition 2.3 (Asymptotic equivalence)**.** Two sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ are said to be asymptotically equivalent if there exists a non-negative integer N such that

$$\lim_{n\to\infty} \frac{a_{N+n}}{b_{N+n}} = 1\,. \tag{2.3}$$

In such a case, we write $a_n \sim b_n$.

Definition 2.4 (Asymptotically typical properties)**.** Let $\mathcal{B}$ be a combinatorial class and P be a predicate defining a property of structures in $\mathcal{B}$. Let $\mathcal{A}$ be a subclass of $\mathcal{B}$ consisting of all structures satisfying P, i.e. $\alpha \in \mathcal{A} \iff P(\alpha)$. If both the counting sequences of $\mathcal{A}$ and $\mathcal{B}$ are asymptotically equivalent, then asymptotically almost all structures in $\mathcal{B}$ have property P; in other words, P is typical. On the other hand, if $\mathcal{A}$ has a zero asymptotic density in $\mathcal{B}$, then asymptotically almost no structures in $\mathcal{B}$ have property P; in other words, P is asymptotically negligible.

Let us notice that slightly different (though not always equivalent) definitions of asymptotic density and asymptotic equivalence are used in the literature (cf., e.g. [Dav+13; BGZ15; GK12]). The fractions (2.1), (2.2) and (2.3) are well defined if and only if the corresponding denominators are strictly positive. Since we are interested in the quotients for large sizes, we intend to disregard the initial segment where the denominator might be equal to zero and focus on the limit of the remainder sequence.

2.1 Analytic combinatorics

In this section we outline the main concepts and techniques of analytic combinatorics used to investigate properties of large random λ-calculus and combinatory logic terms. We refer the curious reader to the excellent handbooks [FS09; Wil06] for a detailed exposition.

Let us start with the fundamental notion of generating function.

Definition 2.5 (Generating function)**.** Let $\mathcal{A}$ be a combinatorial class. Then, the formal power series

$$A(z) = \sum_{n \geq 0} a_n z^n \tag{2.4}$$

is the ordinary generating function of $\mathcal{A}$.

Different types of generating functions are studied in the literature, e.g. Dirichlet series, exponential generating functions or their multivariate variants (see, e.g. [FS09, Chapter III]). Since we are exclusively interested in ordinary ones, as a convention we omit the adjective 'ordinary' and simply write generating function, instead.

We write $A(z)$ to denote the generating function corresponding to the combinatorial class $\mathcal{A}$. To denote the coefficient standing by z^n in $A(z)$ we use $[z^n]A(z)$. Finally, if $[z^n]A(z) \leq [z^n]B(z)$ for each $n \geq 0$ (for instance, if $\mathcal{A}$ is a subclass of $\mathcal{B}$), we write $A(z) \preceq B(z)$.

Generating functions, as formal power series, provide a compact representation of counting sequences and, as such, prove useful in solving linear recurrence equations [Wil06]. From this point of view, the question of their convergence is naturally irrelevant. Nonetheless, generating functions representing convergent power series in one complex variable are of special interest as they coincide with the notion of analytic functions at the complex plane origin.

Definition 2.6 (Analytic function)**.** A function f defined over a region Ω of the complex plane is analytic at $z_0 \in \Omega$ if in some open disk centred at z_0 and contained in Ω, f is representable by a convergent power series expansion

$$f(z) = \sum_{n=0}^{\infty} a_n (z - z_0)^n \,. \tag{2.5}$$

We say that f is analytic in a region Ω if f is analytic in each point $z_0 \in \Omega$.

In this analytic perspective, various natural operations on functions analytic at the complex plane origin map to symbolic manipulations on the counting sequences represented by respective generating functions (see [FS09, Part A, Symbolic Methods]). In particular, the addition and multiplication of generating functions analytic at the origin map to the disjoint union and Cartesian product (i.e. the class of ordered pairs of structures) of corresponding combinatorial classes, respectively:

$$\sum_{n\geq 0} a_n z^n + \sum_{n\geq 0} b_n z^n = \sum_{n\geq 0} c_n z^n \quad \text{where} \quad c_n = a_n + b_n\,, \tag{2.6}$$

$$\sum_{n\geq 0} a_n z^n \cdot \sum_{n\geq 0} b_n z^n = \sum_{n\geq 0} c_n z^n \quad \text{where} \quad c_n = \sum_{i=0}^{n} a_i b_{n-i}\,. \tag{2.7}$$

Moreover, the interpretation of generating functions as functions analytic at the origin allows us to relate their analytic properties with the quantitative properties of studied structures; for instance, access the asymptotic form of the corresponding counting sequences using the methods of singularity analysis initiated by Flajolet and Odlyzko [FO90].

Theorem 2.7 ([Wil06, Theorem 2.4.1]). Let $A(z)$ be analytic at the origin. Then, either:

(i) $A(z)$ converges for each value of z, or

(ii) there exists a positive real number R such that

- $A(z)$ converges for all $|z| < R$, and
- $A(z)$ diverges for all $|z| > R$.

In the latter case, the number R is referred to as the convergence radius of $A(z)$.

Generating functions representing converging power series are analytic in the interior of their convergence disks. Furthermore, their convergence radii convey the exponential factors in the growth rate of corresponding counting sequences.

Theorem 2.8 (Exponential growth formula [FS09, 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

$$R = \sup\{r \geq 0 \ : \ A(z) \text{ is analytic in } |z| < r\}\,, \tag{2.8}$$

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

$$a_n = R^{-n}\theta(n) \quad \text{with} \quad \limsup |\theta(n)|^{\frac{1}{n}} = 1\,. \tag{2.9}$$

Definition 2.9 (Complex plane region). A set Ω is called a region if it is a non-empty, open and connected subset of the complex plane.

Definition 2.10 (Analytic continuation). Let f be an analytic function in a region Ω. If there exists an analytic function g in a region $\Omega^\star$ such $\Omega \cap \Omega^\star \neq \varnothing$ and $f(z) = g(z)$ in $\Omega \cap \Omega^\star$, then g is an analytic continuation of f to $\Omega^\star$.

Definition 2.11 (Singularity)**.** Let Ω be a region of the complex plane containing z_0 and f be an analytic function in the region $\Omega \setminus \{z_0\}$. If $f(z)$ has an analytic continuation to Ω, then z_0 is said to be a removable singularity of f. Otherwise, if f cannot be analytically continued to a larger region containing z_0, then z_0 is said to be a singularity of f.

Theorem 2.12 (Boundary singularities [FS09, Theorem IV.5])**.** Let $A(z)$ be a function analytic at the origin, whose expansion has a finite radius of convergence. Then necessarily $A(z)$ has a singularity on the boundary of its disk of convergence.

In other words, generating functions analytic at the origin cease to converge on the boundaries of their disks of convergence due to the existence of so-called *dominant singularities.* Finding their location and type determines the process of singularity analysis.

Theorem 2.13 (Pringsheim [FS09, 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 $A(z)$.

In our case, the investigated generating functions are algebraic, i.e. are roots of polynomial equations. It is a well known fact that $\sqrt{z}$ cannot be unambiguously defined as an analytic function in a neighbourhood of $z = 0$; hence, when looking for the radius of convergence of $A(z)$ we are primarily interested in roots of radicands involved in the closed-form expression defining $A(z)$.

Proposition 2.14 (Rescaling rule)**.** Let $A(z)$ be analytic at the origin. Then,

$$[z^n]A(z) = \rho^{-n}[z^n]A(\rho z)\,. \tag{2.10}$$

After a proper rescaling, we can focus on the type of singularities of $A(z)$ on the unit circle. The following analytic tools allow us to determine the sub-exponential factors in the growth rate of the corresponding counting sequence.

Theorem 2.15 (Standard function scale [FS09, 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

$$[z^n]f(z) = \frac{n^{\alpha-1}}{\Gamma(\alpha)}\left(1 + \frac{\alpha(\alpha-1)}{2n} + \frac{\alpha(\alpha-1)(\alpha-2)(3\alpha-1)}{24n^2} + O\left(\frac{1}{n^3}\right)\right) \tag{2.11}$$

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

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx\,. \tag{2.12}$$

Theorem 2.16 (Newton, Puiseux [FS09, 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

$$f(z) = \sum_{k \geq k_0} c_k (z-\rho)^{k/\kappa}\,, \tag{2.13}$$

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

Theorem 2.17 (Algebraic singularity analysis [FS09, Theorem VII.8])**.** Suppose that $\alpha \in \mathbb{C} \setminus \mathbb{Z}_{\leq 0}$. Let $f(z) = (1 - z/\rho)^{-\alpha} g(z) + h(z)$ be an algebraic function, analytic at the origin, which has a unique dominant singularity $z = \rho$. Moreover, assume that $g(z)$ and $h(z)$ are analytic in a larger disk $|z| < \rho + \eta$ for some $\eta > 0$. Then, the coefficients $[z^n]f(z)$ satisfy the following asymptotic approximation

$$[z^n]f(z) \sim \rho^{-n} \frac{Cn^{\alpha-1}}{\Gamma(\alpha)} \tag{2.14}$$

where the constant C is equal to $g(\rho)$, i.e. the coefficient standing by $(1 - z/\rho)^{-\alpha}$ in the Newton-Puiseux expansion of $f(z)$.

In order to analyse the limit mean value of a random variable X_n corresponding to a combinatorial parameter of random λ-terms, we utilise the moment techniques of multivariate generating functions (see e.g. [FS09, Chapter 3]). In particular, we use the following derivation and integration schemes:

$$u \frac{\partial}{\partial u} A(z, u) = \sum_{n,k \geq 0} k a_{n,k} z^n u^k \,, \tag{2.15}$$

$$\int_0^u \frac{dt}{t} A(z, t) = \sum_{n,k \geq 0} \frac{1}{k} a_{n,k} z^n u^k \tag{2.16}$$

which enable the elegant expression of the mean value $\mathbb{E}(X_n)$ associated with the investigated multivariate generating function $A(z, u)$.

2.1.1 Technical lemmas

In the course of using symbolic methods we often obtain an expression $A(z)$ which, as such, is not defined at the origin. In consequence, we cannot carry out the algebraic singularity analysis of $A(z)$ directly; instead, we have to consider its analytic continuation including the complex plane origin. To determine whether $A(z)$ has an analytic continuation including the origin (i.e. whether the singularity at the origin is removable) we use the following classic result.

Theorem 2.18 (Riemann, see e.g. [Kra99])**.** Let f be analytic on the punctured disk $\Omega \setminus \{z_0\}$ of the complex plane. Then, f has an analytic continuation on Ω if and only if

$$\lim_{z \to z_0} (z - z_0) f(z) = 0\,. \tag{2.17}$$

As an immediate consequence, we obtain the following technical lemma.

Lemma 2.19. Let f be analytic on the punctured disk $\Omega \setminus \{z_0\}$. Suppose that f has an analytic continuation on Ω. Then for each $n \geq 2$, the function $f(z)^n$ has an analytic continuation on Ω.

Proof. Straightforward induction. □

In order to simplify the reasoning about the type and location of singularities of generating functions given without explicit closed-form expressions, we utilise the following technical lemma guaranteeing certain natural closure properties of analytic functions with a single square-root type dominating singularity.

Lemma 2.20. Let Ω be the open disk $|z| < \rho + \eta$ for some $0 < \rho < 1$ and $\eta > 0$. Let F denote the set of functions $f\colon (0,\rho) \to \mathbb{C}$ in form of $f(z) = \sqrt{1 - z/\rho}\, P(z) + Q(z)$ for arbitrary $P(z)$ and $Q(z)$ analytic in $\Omega \setminus \{0\}$. Then, F with natural function addition and multiplication forms a commutative ring.

Proof. Note that it suffices to show that F is closed under addition and multiplication, as the commutative ring laws are certainly preserved. Let $(U, +, \times)$ be the commutative ring of functions analytic in $\Omega \setminus \{0\}$. Consider arbitrary $f, g \in F$ given by $f(z) = \sqrt{1 - z/\rho}\, P_f(z) + Q_f(z)$ and $g(z) = \sqrt{1 - z/\rho}\, P_g(z) + Q_g(z)$.

Let us start with $f(z) + g(z)$. Note that

$$f(z) + g(z) = \sqrt{1 - z/\rho}\, \big(P_f(z) + P_g(z)\big) + Q_f(z) + Q_g(z)\,. \tag{2.18}$$

Clearly, $f(z) + g(z) = \sqrt{1 - z/\rho}\, \widetilde{P}(z) + \widetilde{Q}(z)$ where both $\widetilde{P}(z) \in U$ and $\widetilde{Q}(z) \in U$. Hence, $f(z) + g(z) \in F$.

Now, let us consider $f(z) \cdot g(z)$. By rewriting, we obtain

$$\begin{aligned} f(z) \cdot g(z) \quad &= \quad \sqrt{1 - z/\rho}\big(P_g(z)Q_f(z) + P_f(z)Q_g(z)\big) \\ &\quad\ + (1 - z/\rho)P_f(z)P_g(z) + Q_f(z)Q_g(z)\,. \end{aligned} \tag{2.19}$$

Clearly, $f(z) \cdot g(z) \in F$. □

2.1.2 Boltzmann samplers

In their seminal paper, Duchon et al. propose a universal framework meant for uniform random generation of large combinatorial structures [Duc+04]. The central idea in their approach is to embed the generation scheme in the theory of analytic combinatorics and therefore, obtain a recursive sampling template for a wide range of existing combinatorial classes. For our purposes, we are primarily interested in samplers for so-called unlabelled algebraic combinatorial classes which design we briefly excerpt in this subsection.

Suppose we have a combinatorial class $\mathcal{A}$ for which we intend to design a sampler, i.e. an algorithm which, for a given non-negative integer n, constructs a uniformly random structure $\alpha \in \mathcal{A}_n$. Duchon et al. propose the following approach.

Relax the deterministic outcome size restriction and parametrise the sampler with an additional real parameter x in the open interval $(0, \rho)$ where ρ is the modulus of a dominating singularity of $A(z)$. Let us impose a probability space on $\mathcal{A}$ such that $\mathbb{P}_x(\alpha)$, i.e. the intended probability that $\alpha \in \mathcal{A}$ is the sampler's outcome, is equal to

$$\mathbb{P}_x(\alpha) = \frac{x^{|\alpha|}}{A(x)}\,. \tag{2.20}$$

Certainly, any two structures of the same size are given the same probability. Now, let N be the random variable marking the size of the sampler's outcome. Note that the probability $\mathbb{P}_x(N = n)$ that the sampler returns a structure of size n is given by

$$\mathbb{P}_x(N = n) = \frac{a_n x^n}{A(x)}\,. \tag{2.21}$$

This is indeed a proper probability as

$$\sum_{n\geq 0} \mathbb{P}_x(N = n) = \frac{1}{A(x)} \sum_{n\geq 0} a_n x^n = 1\,. \tag{2.22}$$

It is easy to verify that the expected outcome size $\mathbb{E}_x(N)$ and its standard deviation $\sigma_x(N)$ are given by the following formulas:

$$\mathbb{E}_x(N) = x\frac{A'(x)}{A(x)} \quad \text{and} \quad \sigma_x(N) = \sqrt{\frac{x^2A''(x) + xA'(x)}{A(x)} - \left(x\frac{A'(x)}{A(x)}\right)^2}\,. \tag{2.23}$$

Hence, in the Boltzmann model we do not control the exact size of the generated sample, though we can calibrate its expected size and standard deviation by choosing a suitable parameter x.

Let $\mathcal{A}$ be a combinatorial class for which we intend to design a Boltzmann sampler $\Gamma_x(\mathcal{A})$. The process of constructing $\Gamma_x(\mathcal{A})$ follows the recursive specification of $\mathcal{A}$.

Suppose that $\mathcal{A} = \mathcal{B}+\mathcal{C}$. Let $\alpha \in \mathcal{A}$. Since both $\mathcal{B}$ and $\mathcal{C}$ are disjoint, the probabilities $\mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{B})$ that $\alpha \in \mathcal{B}$ and $\mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{C})$ that $\alpha \in \mathcal{C}$ are equal to

$$\mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{B}) = \frac{B(x)}{A(x)} \quad \text{and} \quad \mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{C}) = \frac{C(x)}{A(x)}\,. \tag{2.24}$$

And so, in order to sample a structure from $\mathcal{A}$ we have to make a probabilistic decision whether to continue with sampling a structure from $\mathcal{B}$ or $\mathcal{C}$. To do so, we draw uniformly at random a real in the closed interval $[0, 1]$ and compare it with the branching probabilities $B(x)/A(x)$ and $C(x)/A(x)$. Subsequently, we continue the sampling by calling one of the corresponding samplers $\Gamma_x(\mathcal{B})$ or $\Gamma_x(\mathcal{C})$.

Now, suppose that $\mathcal{A} = \mathcal{B} \times \mathcal{C}$. Let $\alpha = (\beta, \gamma) \in \mathcal{A}$. Note that

$$\mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{A}) = \frac{x^{|\alpha|}}{A(x)} = \frac{x^{|\beta|+|\gamma|}}{B(x)C(x)} = \mathbb{P}_{\Gamma,x}(\beta \in \mathcal{B}) \cdot \mathbb{P}_{\Gamma,x}(\gamma \in \mathcal{C})\,. \tag{2.25}$$

And so, in order to sample a structure from $\mathcal{A}$ we have to independently sample two structures, $\beta \in \mathcal{B}$ and $\gamma \in \mathcal{C}$, which we then assemble into a pair (β, γ).

The recursion stops at the level of singleton classes. In such a case, we simply return the single structure in our class, since

$$\mathbb{P}_{\Gamma,x}(\alpha \in \mathcal{A}) = \frac{x^{|\alpha|}}{A(x)} = \frac{x^{|\alpha|}}{x^{|\alpha|}} = 1\,. \tag{2.26}$$

Once the control parameter x is established, it becomes possible to compile the Boltzmann sampler given its combinatorial specification and the numerical values of involved generating functions at x. In consequence, we obtain an efficient approximate-size sampler.

Theorem 2.21 (Duchon, Flajolet, Louchard and Schaeffer [Duc+04])**.** *Let $\mathcal{A}$ be a combinatorial class specified (in a possibly recursive way) from finite sets by means of disjoint unions and Cartesian products. Assume as given an oracle that provides the finite collection of exact values at a coherent value $x \in (0, \rho)$ of the generating functions intervening in a specification of $\mathcal{A}$. Then, the Boltzmann generator $\Gamma_x(\mathcal{A})$ performs $O(n)$ real-arithmetic operations where n is the size of its output structure.*

2.2 Lambda calculus

In this section we excerpt the key notions of λ-calculus used throughout the dissertation. We refer the curious reader to the classic handbook [Bar84] for a detailed exposition.

Definition 2.22 (λ-terms)**.** Let V be an infinite, denumerable set of variables. Then, the set of λ-terms is defined inductively as follows:

(i) Each variable $x \in V$ is a λ-term;

(ii) If N and M are λ-terms, then (NM) is a λ-term;

(iii) If N is a λ-term and x is a variable, then $(\lambda x.N)$ is a λ-term.

Lambda terms in form of (NM) are called applications. Terms in form of $(\lambda x.N)$ are referred to as abstractions.

Following standard notational conventions we omit outermost parentheses and drop parentheses from left-associated λ-terms. For instance, $(\lambda x.\lambda y.((xy)z))$ is equivalent to $\lambda x \lambda y.xyz$. Moreover, as in the example, whenever there is a sequence of consecutive abstractions, we replace it with a single 'bulk' abstraction. Hence, $\lambda x \lambda y.xyz$ is simply written as $\lambda xy.xyz$. We use lower case letters $x, y, z, \ldots$ to denote variables. To denote arbitrary λ-terms, we use capital letters $N, M, P, \ldots$.

Definition 2.23 (λ-trees)**.** Let N be a λ-term. Then, the λ-tree of N is defined inductively as follows:

(i) The λ-tree of a variable x is a single node labelled 'x';

(ii) If $N = MP$, then the λ-tree of N has a binary root with two subtrees – the λ-tree of M on the left-hand side and the λ-tree of P on the right-hand side;

(iii) If $N = \lambda x.M$, then the λ-tree of N has a unary root followed by the λ-tree of M. The root node in the λ-tree of N is labelled 'λx'.

As an example, consider the following λ-tree of $T = \lambda xy.x(zx)$:

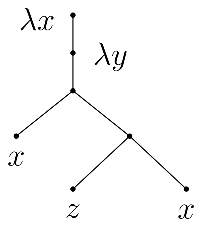


Figure 2.1: Tree-like representation of T.

Henceforth we equate λ-terms with their λ-trees and use both representation interchangeably. This slightly abusive convention allows us to conveniently express certain attributes of λ-terms using common combinatorial notions from graph theory.

Definition 2.24 (Free variables)**.** Let N be a λ-term. Then, the set $\mathrm{FV}(N)$ of free variables in N is defined inductively as follows:

(i) $\mathrm{FV}(x) = \{x\}$;

(ii) $\mathrm{FV}(NM) = \mathrm{FV}(N) \cup \mathrm{FV}(M)$;

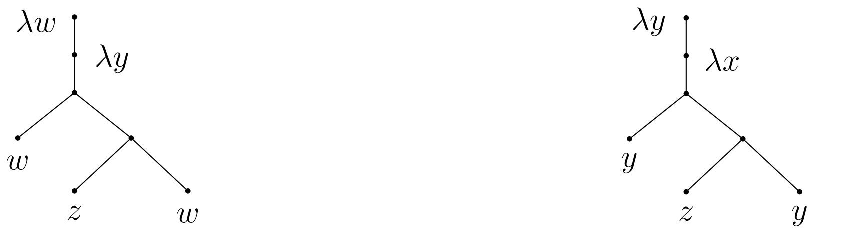
(iii) $\mathrm{FV}(\lambda x.N) = \mathrm{FV}(N) \setminus \{x\}$.

In the context of λ-trees, a variable x is *bound* in N if on its path to the root there exists an abstraction labelled λx. In such a case the closest, in terms of path distance, abstraction λx on the path to the root is said to be the *binder* of x. Otherwise, if the variable x is not bound in N, then x is *free* in N. If all variables in N are bound (equivalently $\mathrm{FV}(N) = \varnothing$), then N is said to be *closed*. Otherwise, if some variable occurs freely in N, then N is said to be *open*.

From abstract functional calculus point of view, some terms in λ-calculus become intrinsically equivalent. Consider $\lambda x.x$ and $\lambda y.y$. Both λ-terms are intended to represent the same syntactic anonymous identity function; the particular argument name is naturally irrelevant. Hence, we wish to identify λ-terms that result from a syntactic change of bound variables. Formally, the λ-term derived by changing the bound variable name x to y in $\lambda x.N$ is equal to $\lambda y.(N[x/y])$ where $N[x/y]$ denotes the result of substituting y for all free occurrences of x in N. This leads us to the following classic notion of α-conversion.

Definition 2.25 (α-conversion)**.** Let N, M be two λ-terms. Then, N and M are said to be α-convertible if M is obtainable from N by a series of bound variables changes.

As an example, consider the following two α-convertible λ-terms:



(a) $T_1 = \lambda wy.w(zw)$ (b) $T_2 = \lambda yx.y(zy)$

Figure 2.2: Two α-convertible λ-terms T_1 and T_2.

Such a conversion induces an equivalence relation on the set of λ-terms; inhabitants of each α-equivalence class are λ-terms identical up to bound variable renaming.

Equipped with α-conversion, we are now allowed to use the following convenient variable name convention due to Barendregt [Bar84]. If $N, M, P, Q, \ldots$ occur in some context, then we assume that all their bound variable names are unique and different from the free variables in $N, M, P, Q, \ldots$. This assumption allows us to express the fundamental notions of substitution and β-reduction in λ-calculus as follows.

Definition 2.26 (Substitution)**.** Let N, M be two λ-terms. Then, the substitution of M for x in N, denoted $N[x := M]$, is defined inductively as follows:

(i) $x[x := M] = M$;

(ii) $y[x := M] = y$;

(iii) $(NP)[x := M] = (N[x := M])(P[x := M])$;

(iv) $(\lambda y.N)[x := M] = \lambda y.(N[x := M])$.

Note that with our variable name convention, in rule (iv) we are safe to substitute M for x in N without the unintended side-effect of capturing the free occurrences of y in M under the scope of the binding abstraction.

Definition 2.27 (β-reduction)**.** The β-reduction relation $\to_\beta$ is the least relation on λ-terms such that:

(i) $(\lambda x.N)M \to_\beta N[x := M]$;

(ii) If $N \to_\beta M$, then for each variable x, $\lambda x.N \to_\beta \lambda x.M$;

(iii) If $N \to_\beta M$, then for each P both $PN \to_\beta TM$ and $NP \to_\beta MP$.

Terms in form of $(\lambda x.N)M$ are called redexes. If N does not contain any redex as subterm, then N is said to be in β-normal form (or simply normal form). If there exists a finite sequence $N_1, N_2, \ldots, N_m$ such that $N = N_1$, consecutive λ-terms form a β-reduction sequence, i.e. $N_i \to_\beta N_{i+1}$, and N_m is in normal form, then N is (weakly) normalising. If all β-reduction sequences starting with N are finite, then N is strongly normalising.

For convenience, we use $\to_\beta^*$ to denote the transitive reflexive closure of $\to_\beta$. To denote the transitive closure of $\to_\beta$ we write $\to_\beta^+$.

Example 2.28. Let $\omega = \lambda x.xx$. Note that $\Omega = \omega\omega$ is not normalising as we have

$$\Omega = (\lambda x.xx)(\lambda x.xx) \to_\beta (xx)[x := \lambda x.xx] = (\lambda x.xx)(\lambda x.xx) = \Omega\,. \tag{2.27}$$

Moreover, any λ-term N containing Ω as a subterm cannot be strongly normalising; rules (ii) and (iii) in the definition of β-reduction imply that we can form a β-reduction sequence proceeding with the reduction of Ω ad infinitum.

However, if N contains Ω as a subterm, then N might still be normalising. Consider the λ-term $N = (\lambda xy.y)\Omega$. Note that we have the following two β-reduction sequences for N:

$$N \;\to_\beta\; (\lambda y.y)[x := \Omega] = \lambda y.y; \tag{2.28}$$

$$N \;\to_\beta\; (\lambda xy.y)\left((zz)[z := \omega]\right) = (\lambda xy.y)\Omega \to_\beta \cdots \tag{2.29}$$

Here, the former one is finite, whereas the latter one is infinite as it tries to reduce Ω first over applying the head abstraction $(\lambda xy.y)$ to Ω.

Term rewriting by means of β-reduction constitutes the computational foundation of λ-calculus – terms represent abstract computations whereas the (potentially infinite) iterative process of β-reduction forms the mechanism of their execution.

Theorem 2.29 (Church and Rosser [CR36])**.** The set of λ-terms is confluent under β-reduction, i.e. if $N \to_\beta^* M$ and $N \to_\beta^* P$, then there exists a λ-term Q such that both $M \to_\beta^* Q$ and $P \to_\beta^* Q$. Pictorially:

$$\begin{tikzcd}
& N \arrow[dl, "*", "\beta"'] \arrow[dr, "*"', "\beta"] & \\
M \arrow[dr, dashed, "*", "\beta"'] & & P \arrow[dl, dashed, "*"', "\beta"] \\
& Q &
\end{tikzcd}$$

In consequence, each normalising λ-term has a unique, modulo α-conversion, normal form.

Finding the normal form of a normalising λ-term poses a considerable difficulty as some reduction strategies fail to find normal forms despite their existence (see Example 2.28). Appreciably, there exists a so-called standard reduction strategy which application is guaranteed to find normal forms of normalising λ-terms.

Theorem 2.30 (Curry and Feys [CF58])**.** Let N be a normalising λ-term. Then, the iterated process of applying β-reduction to the leftmost-outermost redex in N leads to the normal form of N.

The computational expressiveness of λ-calculus originates from Kleene's effective expression of total Herbrand-Gödel recursive functions by means of weakly normalising λ-terms [Kle36]. In light of the famous Church-Turing thesis, it is therefore possible to represent all computable functions within λ-calculus, setting it as a universal theory of computations.

Theorem 2.31 (Church [Chu36])**.** The set of normalising λ-terms is undecidable.

Similarly to Kleene's original construction, it is possible to express total Herbrand-Gödel recursive functions by means of strongly normalising λ-terms (see e.g. [Bar84, Chapter 9] or [Urz03]). In consequence, we have the following classic result.

Theorem 2.32. The set of strongly normalising λ-terms is undecidable.

2.2.1 Simply-typed theory

A prominent part of λ-calculus is its simply-typed variant. Here, we outline some of its basic notions. We refer the curious reader to the excellent handbooks [Hin96; SU06] for a thorough exposition.

Definition 2.33 (Types)**.** Let X be an infinite, denumerable set of type variables, distinct from the set of term variables. Then, the set of types is defined inductively as follows:

(i) Each type variable $a \in X$ is a type;

(ii) If σ and τ are types, then $(\sigma \to \tau)$ is a type.

Types in form of $(\sigma \to \tau)$ are called arrow types.

Following standard notational conventions we omit outermost parentheses and drop parentheses from right-associated types, e.g. $\sigma \to \tau \to \rho$ is equivalent to $(\sigma \to (\tau \to \rho))$. We use lower case letters $a, b, c, \ldots$ to denote type variables. To denote arbitrary types, we use lower case Greek letters $\sigma, \tau, \rho, \ldots$.

Definition 2.34 (Typing context)**.** A typing context Γ is a finite set of typing assignments, i.e. pairs in form of $x \colon \sigma$ (read x is of type σ), such that no term variable x is subject to more than one type assignment.

Definition 2.35 (Typing rules)**.** Simply-typed λ-calculus is equipped with the following typing rules:

$$\frac{x \colon \sigma \in \Gamma}{\Gamma \vdash x \colon \sigma} \tag{2.30}$$

$$\frac{\Gamma \vdash N \colon \sigma \to \tau \qquad \Gamma \vdash M \colon \sigma}{\Gamma \vdash NM \colon \tau} \tag{2.31}$$

$$\frac{\Gamma, x \colon \sigma \vdash N \colon \tau}{\Gamma \vdash \lambda x.N \colon \sigma \to \tau} \tag{2.32}$$

Figure 2.3: Typing rules for simply-typed λ-calculus.

In words,

(i) If $x \colon \sigma$ is in context Γ, then we can infer that x is of type σ in the context Γ;

(ii) If N is of type $\sigma \to \tau$ in the context Γ whereas in the same context M is of type σ, then the result of applying N to M is of type τ;

(iii) If N is of type τ in a context Γ containing the type assignment $x \colon \sigma$, then $\lambda x.N$ is of type $\sigma \to \tau$ in the context $\Gamma \setminus \{x \colon \sigma\}$.

Definition 2.36 (Typeable λ-terms)**.** Let N be a λ-term. Then, N is (simply) typeable if there exists a type σ such that $\varnothing \vdash N \colon \sigma$. To denote the fact that N is of type σ in an empty context we write $N \colon \sigma$.

Example 2.37. Let $I = \lambda x.x$. Note that $I \colon \sigma \to \sigma$ for an arbitrary type σ. Certainly, using rule (2.30) we can infer that $\{x \colon \sigma\} \vdash x \colon \sigma$. As we now abstract the variable x, rule (2.32) asserts that $\lambda x.x \colon \sigma \to \sigma$. Intuitively, $\lambda x.x$ represents an anonymous identity function which maps arguments of type σ to results of identical type σ.

Note however that not all λ-terms are typeable. Consider $\omega = \lambda x.xx$. Suppose to the contrary that ω is typeable, i.e. $\omega \colon \sigma \to \tau$ (recall that abstractions can be assigned only arrow types). Hence, $\{x \colon \sigma\} \vdash (xx) \colon \tau$. However, since (xx) is an application, we know that x is of some type $\rho \to \tau$ in the context $\{x \colon \sigma\}$. And so $\sigma = (\rho \to \tau)$. Moreover, at the same time $\rho = \sigma$ as x is applied to itself in (xx). In consequence, $\sigma = (\sigma \to \tau)$ yielding an infinite type σ – a contradiction. Thus, ω is not typeable.

From a programming point of view, simple types introduce an important safety layer in λ-calculus as they prohibit the construction of non-terminating computations.

Theorem 2.38 (Tait, see e.g. [SU06])**.** Let N be a typeable λ-term. Then, N is strongly normalising.

Moreover, from a logical point of view, simply-typed λ-terms constitute a proof system for the implicational fragment of minimal logic – a variant of intuitionistic logic rejecting the principle of explosion (Latin: *ex falso sequitur quodlibet*).

Definition 2.39 (Minimal logic)**.** The implicational fragment of minimal logic $\mathfrak{I}_{\to}$ is a proof system with an infinite, denumerable set of axioms in form of $(\Pi, \sigma \vdash \sigma)$ for each finite set of types Π (also called premises) containing type σ, and the following two inference rules:

$$(\mathrm{E} \to) \, \frac{\Pi \vdash \sigma \to \tau \qquad \Pi \vdash \sigma}{\Pi \vdash \tau} \tag{2.33}$$

$$(\mathrm{I} \to) \, \frac{\Pi, \sigma \vdash \tau}{\Pi \vdash \sigma \to \tau} \tag{2.34}$$

Figure 2.4: Inference rules of $\mathfrak{I}_{\to}$.

In words,

(i) If from a set Π of types we can infer both the types $\sigma \to \tau$ and σ, then we can infer the type τ from Π as well (the so-called implication elimination);

(ii) If from a set Π of types containing σ we can infer the type τ, then from Π without σ we can infer the type $\sigma \to \tau$ (the so-called implication introduction).

A type σ is said to be a $\mathfrak{I}_{\to}$-theorem if $\varnothing \vdash \sigma$, i.e. σ can be inferred from an empty set of premises.

The apparent analogy between simply-typed λ-terms and $\mathfrak{I}_{\to}$ is known in the literature as the Curry-Howard correspondence.

Theorem 2.40 (Curry and Howard, see e.g. [SU06])**.** A type σ can be inferred from a set $\{\tau_1, \ldots, \tau_n\}$ of premises in $\mathfrak{I}_{\to}$ if and only if there exists a context $\Gamma = \{x_1 \colon \tau_1, \ldots, x_n \colon \tau_n\}$ and a λ-term N such that N is of type σ in the context Γ and moreover $\mathrm{FV}(N) = \{x_1, \ldots, x_n\}$. In particular, σ is an $\mathfrak{I}_{\to}$-theorem if and only if there exists a closed λ-term N such that $N \colon \sigma$.

Definition 2.41 (Substitution)**.** A substitution Δ is a finite collection of pairs of types $[a_1/\tau_1, \ldots, a_n/\tau_n]$ where each a_i is a unique type variable and each τ_i is a type. The result of applying Δ to a type σ, denoted as $\sigma\Delta$, is defined inductively as follows:

$$\sigma\Delta = \begin{cases} \tau\Delta \to \rho\Delta & \text{if } \sigma = (\tau \to \rho)\,, \\ \tau_i & \text{if } \sigma = a_i \text{ and } a_i/\tau_i \in \Delta\,, \\ \sigma & \text{if } \sigma \text{ is a variable such that } \sigma \notin \{a_1, \ldots, a_n\}\,. \end{cases} \tag{2.35}$$

Definition 2.42 (Principal types)**.** Let $N \colon \sigma$. Then, σ is a principal type of N if for each τ such that $N \colon \tau$, σ is more general than τ, i.e. there exists a substitution Δ such that $\sigma\Delta = \tau$. In such a case τ is called an instance of σ.

Example 2.43. Let us consider again $\lambda x.x$. Certainly, we can assign to it one of infinitely many types, for instance:

(i) $\lambda x.x \colon a \to a$;

(ii) $\lambda x.x \colon (a \to a) \to (a \to a)$;

(iii) $\lambda x.x \colon (b \to c) \to (b \to c)$.

Note that $a \to a$ is more general than either $(a \to a) \to (a \to a)$ or $(b \to c) \to (b \to c)$. In the former case we have $(a \to a)[a/(a \to a)] = (a \to a) \to (a \to a)$ whereas in the latter one $(a \to a)[a/(b \to c)] = (b \to c) \to (b \to c)$. In fact, each type σ such that $\lambda x.x \colon \sigma$ must be in form of $\tau \to \tau$ (see Example 2.37). Hence, $a \to a$ is a principal type of $\lambda x.x$. However, $a \to a$ is not its unique principal type; an equally perfect principal type is, for instance $b \to b$.

Principal types play an important role in the implementation of functional programming languages, e.g. due to the prominent Hindley–Milner type system with parametric polymorphism [Hin96] in which it is possible to automatically infer a principal type of a function without explicit programmer-declared type annotations.

2.2.2 De Bruijn notation

The classic representation of λ-terms with variable names, though elegant and succinct for manual manipulations, poses considerable problems with automatic substitution, e.g. in implementations of functional programming languages [Pey87]. De Bruijn proposed the following alternative representation of λ-calculus using natural integers, instead of variable names [Bru72].

Definition 2.44 (λ-terms)**.** Let $I = \{\underline{0}, \underline{1}, \underline{2}, \ldots\}$ be an infinite, denumerable set of indices. Then, the set of λ-terms is defined inductively as follows:

(i) Each index $\underline{n} \in I$ is a λ-term;

(ii) If N and M are λ-terms, then (NM) is a λ-term;

(iii) If N is a λ-term, then (λN) is a λ-term.

As in the classic notation, we follow standard notational conventions; we omit outermost parentheses and drop parentheses from left-associated λ-terms.

It is worth mentioning that originally de Bruijn started indices with $\underline{1}$ instead of $\underline{0}$. Here, we follow the convention started, to our best knowledge, by Lescanne (Lescanne, personal communication, 2016).

In the de Bruijn notation, each variable x is replaced with an appropriate index $\underline{n}$ intended to encode the distance between x and its binding abstraction. Specifically, the index $\underline{n}$ represents a variable occurrence which binder is the $(n+1)$st abstraction on the unique path from the index $\underline{n}$ to the term root. If there are less than $n+1$ abstractions on this path, for instance $h < n+1$, then the index represents a free variable x_{n-h+1} assuming an arbitrary fixed listing $(x_n)_{n\in\mathbb{N}}$ of available variables. In consequence, each

λ-term in the classic variable notation has a natural representation in the de Bruijn notation – remove variable labels in abstractions and replace each variable occurrence with an appropriate index.

As in the classic variable notation, it is convenient to equate λ-terms in the de Bruijn notation with their analogous λ-trees (see Definition 2.23).

Example 2.45. Consider the following λ-tree of $T = \lambda\lambda\underline{1}(\underline{2}\underline{1})$:

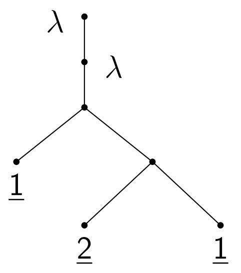


Figure 2.5: λ-tree of T.

Here, T has two bound index occurrences $\underline{1}$ and a free one $\underline{2}$. The index $\underline{2}$ is free in T as there are only two abstractions. Note however that if we add a single abstraction on top of T, then $\underline{2}$ becomes bound.

A prominent feature of the de Bruijn notation is the fact that α-convertible λ-terms in the classic variable notation have an identical de Bruijn representation. In consequence, each λ-term in the de Bruijn sense represents an entire α-equivalence class of classic terms. Moreover, as there are no variable names in this notation, the substitution in an abstraction is significantly simplified; if we substitute M for an index $\underline{n}$ in λN we have to increase $\underline{n}$ by one and continue with $\lambda(N[(\underline{n+1}) := M])$. Variable renaming and capture avoidance is therefore no longer an issue.

2.3 Combinatory logic

In this section we present the main notions of combinatory logic. As in the case of λ-calculus, we refer the curious reader to the classic handbook [Bar84] for a detailed exposition.

Definition 2.46 (Combinators)**.** Let $\mathcal{B}$ be a finite basis of primitive combinators. Then, the set of $\mathcal{B}$-combinators is defined inductively as follows:

(i) Each primitive combinator $X \in \mathcal{B}$ is a $\mathcal{B}$-combinator;

(ii) If N and M are $\mathcal{B}$-combinators, then (NM) is a $\mathcal{B}$-combinator.

In the latter case, we say that (NM) is an application of N to M.

If the underlying basis is clear from the context (or perhaps even irrelevant), we write combinators (terms) instead of $\mathcal{B}$-combinators. For convenience, instead of writing $\{X_1, \ldots, X_n\}$-combinators, we write $X_1 \ldots X_n$-combinators. And so, e.g. instead of

$\{S, K\}$-combinators we simply write SK-combinators. Following standard notational conventions, we omit outermost parentheses and drop parentheses from left-associated combinators, e.g. instead of writing $((MN)(PQ))$ we write $MN(PQ)$. We use $N, M, P, Q, \ldots$ to denote arbitrary combinators. To denote primitive combinators, we use letters $X, Y, \ldots$.

Definition 2.47 (Combinator trees). Let N be a combinator. The combinator tree of N is defined inductively as follows:

(i) The combinator tree of a primitive combinator X is a single node labelled 'X';

(ii) If $N = MP$, then the combinator tree of N has a binary root with two subtrees – the combinator tree of M on the left-hand side and the combinator tree of P on the right-hand side.

As an example, consider the following representation of the SK-combinator $T = SK(KS)$:

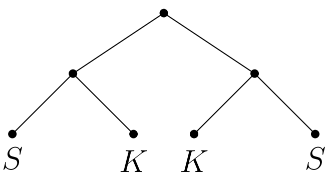


Figure 2.6: Combinator tree representation of T.

As in the case of λ-terms, we equate combinators with their combinatorial representation. And so for instance, N is a *subterm* of M if the combinator tree of N is a subtree of the combinator tree of M.

Definition 2.48 (Reduction). Let $\mathcal{B}$ be a combinator basis. Each primitive combinator $X \in \mathcal{B}$ contributes a reduction rule $\to_X$ in form of

$$XN_1 \ldots N_n \to_X M \tag{2.36}$$

where $n \geq 1$ and M is built from $N_1, \ldots, N_n$ and application.

The reduction relation $\to_w$ is then the least relation on $\mathcal{B}$-combinators such that

(i) If $XN_1 \ldots N_n \to_X M$ for some $X \in \mathcal{B}$, then $XN_1 \ldots N_n \to_w M$;

(ii) If $N \to_w M$, then for each P both $PN \to_w TM$ and $NP \to_w MP$.

Combinators in form of $XN_1 \ldots N_n$ are called redexes whereas their corresponding right-hand side combinators M are called reducts. If N does not contain any redex as subterm, then N is in normal form. If there exists a finite sequence $N_1, N_2, \ldots, N_m$ such that $N = N_1$, consecutive combinators form a reduction sequence, i.e. $N_i \to_w N_{i+1}$, and N_m is in normal form, then N is (weakly) normalising. If all reduction sequences starting with N are finite, then N is strongly normalising.

We use $\to_w^*$ to denote the transitive reflexive closure of $\to_w$. The transitive closure of $\to_w$ is written as $\to_w^+$.

Example 2.49. Throughout the literature different classic combinator bases are considered, such as $\{S, K\}$, $\{S, K, I\}$ or $\{B, C, K, W\}$. The reduction rules for corresponding primitive combinators are defined as follows:

$$\begin{aligned} SNMP &\to_S NP(MP); && (2.37)\\ KNM &\to_K N; && (2.38)\\ IN &\to_I N; && (2.39)\\ BNMP &\to_B N(MP); && (2.40)\\ CNMP &\to_C NPM; && (2.41)\\ WNM &\to_W NMM. && (2.42) \end{aligned}$$

Definition 2.50 (Universal combinator bases)**.** Let $\mathcal{B}$ be a combinator basis. Then, $\mathcal{B}$ is universal if there exist $\mathcal{B}$-combinators $\overline{S}$ and $\overline{K}$ such that for arbitrary $\mathcal{B}$-combinators N, M, P the following conditions hold:

(i) $\overline{S}NMP \to_w^+ NP(MP)$;

(ii) $\overline{K}NM \to_w^+ N$.

In such a case both $\overline{S}$ and $\overline{K}$ are said to implement S and K, respectively.

Example 2.51. Let us show that $\mathcal{B} = \{B, C, K, W\}$ is a universal combinator basis. Certainly, $K \in \mathcal{B}$ hence is suffices to find a $BCKW$-combinator $\overline{S}$ implementing S. Let us consider $\overline{S} = B(BW)(BBC)$. Then, for arbitrary $BCKW$-combinators N, M, P:

$$\begin{aligned} \overline{S}NMP &= B(BW)(BBC)NMP && (2.43)\\ &\overset{(2.40)}{\to_w} BW(BBCN)MP && (2.44)\\ &\overset{(2.40)}{\to_w} W(BBCNM)P && (2.45)\\ &\overset{(2.42)}{\to_w} BBCNMPP && (2.46)\\ &\overset{(2.40)}{\to_w} B(CN)MPP && (2.47)\\ &\overset{(2.40)}{\to_w} CN(MP)P && (2.48)\\ &\overset{(2.41)}{\to_w} NP(MP). && (2.49) \end{aligned}$$

Hence, $\{B, C, K, W\}$ is indeed a universal combinator basis.

On the other hand, not all bases are universal. Consider $\{S\}$. Note that in its reduction rule (2.37) S copies all of its arguments to the reduct. It means that no S-combinator is able to discard its arguments and thus able to implement K (2.38).

Definition 2.52 (Translation to λ-terms)**.** Let $X \in \mathcal{B}$ be a primitive combinator with a corresponding reduction rule $XN_1 \dots N_n \to_X M$. Suppose we assign to X a λ-term $\lambda x_1 \dots x_n.\Psi(M)$ where $\Psi(M)$ is defined inductively as follows:

(i) If $M = N_i$, then $\Psi(M) = x_i$;

(ii) If $M = PQ$, then $\Psi(M) = \Psi(P)\Psi(Q)$.

In words, we replace each occurrence of N_i in M with x_i, translating the application of $\mathcal{B}$-combinators to application of λ-terms. The translation $(\cdot)_\lambda$ of $\mathcal{B}$-combinators to λ-terms is then defined inductively as follows:

(i) $(X)_\lambda = \Psi(X)$;

(ii) $(NM)_\lambda = (N)_\lambda(M)_\lambda$.

Proposition 2.53. If $N \to_w M$, then $(N)_\lambda \to_\beta^+ (M)_\lambda$.

Proof. Note that if $XN_1 \ldots N_n \to_X M$, then $(XN_1 \ldots N_n)_\lambda \to_\beta^+ (M)_\lambda$. □

In consequence, it becomes possible to emulate the computations of $\mathcal{B}$-combinators in λ-calculus. Remarkably, the converse implication is also true, provided that $\mathcal{B}$ is a universal combinator basis. The key idea in proving this fact is the construction of an 'abstraction' operation within the language of $\mathcal{B}$-combinators acting as an abstraction in λ-calculus.

Theorem 2.54 (Translation to $\mathcal{B}$-combinators, see e.g. [Bar84])**.** Let $\mathcal{B}$ be a universal combinator basis. Then, there exists a translation $(\cdot)_\mathcal{B}$ of λ-terms to $\mathcal{B}$-combinators such that for arbitrary closed λ-terms N, M if $N \to_\beta^* M$, then $(N)_\mathcal{B} \to_w^* (M)_\mathcal{B}$.

In consequence, we obtain the following classic result.

Theorem 2.55. Let $\mathcal{B}$ be a universal combinator basis. Then, the set of normalising $\mathcal{B}$-combinators is undecidable.

From a rewriting theory point of view, the set of $\mathcal{B}$-combinators falls under the scope of so-called left-normal orthogonal (also known as left-linear non-ambiguous) term rewriting systems. In consequence, the properties of confluence and standard, leftmost-outermost normalisation strategies follow as direct corollaries (see e.g. [Klo92]).

Theorem 2.56 (Confluence and standardisation)**.** The set of $\mathcal{B}$-combinators is confluent under the reduction relation $\to_w$. In particular, each normalising combinator has a unique normal-form. Moreover, if N is normalising, then the iterated process of reducing the leftmost-outermost redex in N leads to the normal form of N.

2.3.1 Simply-typed combinators

As in the case of λ-calculus, simply-typed variants of combinatory logic are also considered in the literature. In what follows, we related them with simply-typed variants of λ-calculus and formalise the notion of typeability under various combinator bases.

Definition 2.57 (Typing rules)**.** Suppose we equip SK-combinators with the following typing system consisting of two axiom schemes and a single inference rule modus ponens:

$$\frac{}{S\colon (a \to b \to c) \to (a \to b) \to a \to c} \quad \text{(Axiom }\mathbf{S}\text{)}$$

$$\frac{}{K\colon a \to b \to c} \quad \text{(Axiom }\mathbf{K}\text{)}$$

$$\frac{N\colon \sigma \to \tau \qquad M\colon \sigma}{NM\colon \tau} \quad \text{(Modus ponens)}$$

Figure 2.7: Typing rules for simply-typed SK-combinators.

In words,

(i) The S combinator can be assigned any type σ which is an instance of the type $(a \to b \to c) \to (a \to b) \to a \to c$;

(ii) The K combinator can be assigned any type σ which is an instance of $a \to b \to a$;

(iii) If N is of type $\sigma \to \tau$ and M is of type σ, then the result (NM) of applying N to M is of type τ.

A combinator N is typeable if there exists a type σ such that N is of type σ. We denote this fact as $N\colon \sigma$.

Note that using the standard translation to λ-calculus (see Definition 2.52) we obtain $\Psi(S) = \lambda xyz.xz(yz)$ and $\Psi(K) = \lambda xy.x$. It is easy to verify that types for S and K are precisely principal types for $\Psi(S)$ and $\Psi(K)$. By extension, we can therefore consider other combinator bases with different axiom schemes.

Definition 2.58 (Sound combinator bases)**.** Let $\mathcal{B}$ be a combinator basis. We say that $\mathcal{B}$ is sound if the following conditions are satisfied:

(i) Each primitive combinator $X \in \mathcal{B}$ contributes an axiom scheme in form of

$$\frac{}{X\colon \sigma}$$

where σ is a principal type of $\Psi(X)$;

(ii) The set of typeable combinators is closed under (Modus ponens).

Chapter 3

Quantitative aspects of lambda-calculus

In the current chapter we focus on the quantitative properties of λ-terms in the de Bruijn notation within the Gittenberger-Gołębiewski size notion framework [GG16].

Recall that in this framework, the size on a λ-term is defined as follows.

Definition 3.1 (λ-term size)**.** Let a, b, c, d be non-negative integers and N be a λ-term in the de Bruijn notation. Then, the size $|N|$ of N is defined inductively as follows:

(i) If $N = \underline{0}$, then $|N| = a$;

(ii) If $N = \underline{n+1}$, then $|N| = b + |\underline{n}|$;

(iii) If $N = \lambda M$, then $|N| = c + |M|$;

(iv) If $N = MP$, then $|N| = d + |M| + |P|$.

In other words, de Bruijn indices are represented in a unary base using zero `0` and the successor operator `succ`; the weights of zero, successor, abstraction and application are a, b, c, d, respectively, and the size of a λ-term is the weighted sum of its constructors.

Moreover, in this general size notion framework constructor weights are assumed to satisfy the following conditions:

(1) $a + d \geq 1$;

(2) $b, c \geq 1$;

(3) $\gcd(b, c, a + d) = 1$.

Conditions (1) and (2) guarantee that the set of λ-terms forms a combinatorial class, i.e. for each n the set of λ-terms of size n is finite. Note that if a and b are both equal to 0, then it is possible to extend any λ-term with an arbitrary number of applications and zeros without changing the initial term size. Similarly, if b or c are equal to 0, then it is possible to insert arbitrarily long chains of abstractions or successors to a λ-term without changing its size. Finally, condition (3) is a more subtle technical assumption guaranteeing that the generating function corresponding to the set of λ-terms has a unique singularity on the boundary of its disk of convergence.

We remark that the Gittenberger-Gołębiewski size model framework captures a broad class of various size notions, including the natural one [Ben+16a] (take $a = b = c = d = 1$) or Tromp's binary size notion [GL15] (take $b = 1$ whereas $a = c = d = 2$). Furthermore, we have the following common asymptotic approximation.

Proposition 3.2 (Gittenberger and Gołębiewski [GG16]). Let $L_\infty(z)$ denote the generating function corresponding to the set of λ-terms. Then,

$$L_\infty(z) = \frac{1 - z^c - \sqrt{(1-z^c)^2 - \frac{4z^{a+d}}{1-z^b}}}{2z^d}\,. \tag{3.1}$$

Moreover,

$$[z^n]L_\infty(z) \sim \rho^{-n}\frac{Cn^{-3/2}}{\Gamma(-\frac{1}{2})} \tag{3.2}$$

where ρ is the smallest positive root of $(1-z^c)^2(1-z^b) - 4z^{a+d}$ and C is a model-specific (i.e. depending entirely on the assumed size notion) constant.

Furthermore, as the radicand expression in (3.1) can be factored into $\frac{(\rho-z)Q(z)}{1-z^b}$ for some determined polynomial $Q(z)$ we can express $L_\infty(z)$ as

$$L_\infty(z) = \frac{1 - z^c - \sqrt{\frac{(\rho-z)Q(z)}{1-z^b}}}{2z^d} = \frac{1 - z^c - \sqrt{\rho\frac{Q(z)}{1-z^b}\left(1-\frac{z}{\rho}\right)}}{2z^d}\,. \tag{3.3}$$

Hence, by a straightforward algebraic singularity analysis, the specific constant C in (3.2) can be expressed as

$$C = -\frac{1}{2\rho^d}\sqrt{\rho\frac{Q(\rho)}{1-\rho^b}}\,. \tag{3.4}$$

3.1 Model-independent results

In this section we are interested in model-independent properties of large random λ-terms. We start with showing that *plain* (i.e. open or closed) λ-terms have the fixed subterm property – asymptotically almost all λ-terms contain an arbitrary fixed λ-term as subterm.

Proposition 3.3 (Fixed subterm property). Let N be a fixed λ-term of size p and $\mathcal{T}_N$ be the set of λ-terms containing N as a subterm. Then, $[z^n]T_N(z)$ and $[z^n]L_\infty(z)$ are asymptotically equivalent.

Proof. Let $M \in \mathcal{T}_N$. Note that M is either equal to N, or N is a proper subterm of M. Hence, in the case when $M = \lambda P$, then $P \in \mathcal{T}_N$. On the other hand, if $M = PQ$, then one of P and Q are in $\mathcal{T}_N$ whereas the other one is an arbitrary λ-term. We can therefore specify $T_N(z)$ as follows:

$$T_N(z) = z^p + z^c T_N(z) + 2z^d T_N(z) L_\infty(z) - z^d T_N(z)^2\,. \tag{3.5}$$

Note that in the expression $2z^d T_N(z)L_\infty(z)$ we are counting each λ-term PQ containing N as a subterm in both P and Q twice. Hence, to avoid double counting we have to subtract $z^d T_N(z)^2$.

Let $\Delta_{L_\infty(z)} = (1-z^c)^2 - \frac{4z^{a+d}}{1-z^b}$ be the radicand in the closed-form expression (3.1) of $L_\infty(z)$. Note that

$$T_N(z) = \frac{-\sqrt{\Delta_{L_\infty(z)}} \pm \sqrt{\Delta_{T_N(z)}}}{2z^d} \quad \text{where} \quad \Delta_{T_N(z)} = \Delta_{L_\infty(z)} + 4z^{d+p}\,. \tag{3.6}$$

Since $\lim_{z\to 0} \Delta_{L_\infty(z)} = \lim_{z \to 0} \Delta_{T_N(z)} = 1$ and $d \geq 1$, by Riemann's removable singularities theorem (see Theorem 2.18) only

$$T_N(z) = \frac{-\sqrt{\Delta_{L_\infty(z)}} + \sqrt{\Delta_{T_N(z)}}}{2z^d} \tag{3.7}$$

is analytically continuable at the origin. It means that the generating function corresponding to the set of λ-terms avoiding N as a subterm is given by

$$L_\infty(z) - T_N(z) = \frac{1 - z^c - \sqrt{\Delta_{T_N(z)}}}{2z^d}\,. \tag{3.8}$$

And so, the smallest positive root ρ_T of $\Delta_{T_N(z)}$ dictates the exponential growth rate of the sequence of λ-terms avoiding N as a subterm. Combining the facts that $T_N(z)$ corresponds to a subclass of plain λ-terms (hence $T_N(z) \preceq L_\infty(z)$) and $\Delta_{T_N(z)} = \Delta_{L_\infty(z)} + 4z^{d+p}$, we finally note that $\rho < \rho_T$ which by virtue of the exponential growth formula implies our claim. □

In consequence, we obtain the following general corollary.

Corollary 3.4. Let $\mathfrak{A}$ be a non-empty set of λ-terms such that if $N \in \mathfrak{A}$ and M contains N as a subterm, then $M \in \mathfrak{A}$. Then, asymptotically almost all λ-terms are in $\mathfrak{A}$, i.e. it is a set of typical λ-terms.

In other words, all properties of λ-terms spanning to superterms are typical among large random λ-terms. Alas, strong normalisation or typeability are not typical.

Corollary 3.5. Asymptotically almost every λ-term is neither in normal form nor strongly normalising (hence also typeable).

Proof. Fix $\Omega = (\lambda x.xx)(\lambda x.xx)$. By Proposition 3.3 asymptotically each λ-term contains Ω as a subterm; however, Ω is neither in normal form, normalisable or typeable. □

Let us notice the striking discrepancy between the density of strongly normalising terms in the de Bruijn size model and the corresponding density in the canonical model considered in [Dav+13]. In the latter case, variables do not contribute to the term size and, arguably, tend to be arbitrarily far from their binders. In the de Bruijn size model, however, the size of a variable is proportional to the distance to its binding abstraction. Such an additional cost diminishes the average distance forcing indices to be rather shallow.

Proposition 3.6. Let X_n be a random variable denoting the average de Bruijn weight in a random λ-term of size n. Let μ_n denote the mean value of X_n. Then, μ_n tends to a constant (depending entirely on the assumed size model) as n tends to infinity.

Proof. Consider the trivariate generating function $L_\infty(z, v, w)$ in which $[z^n v^k w^l]L_\infty(z, v, w)$ (i.e. the coefficient standing by $z^n v^k w^l$) denotes the number of λ-terms of size n with exactly k indices of total weight l and the related generating function $D(z, v, w)$ of corresponding de Bruijn indices. Let us start with the latter.

Note that each index $\underline{\mathsf{n}}$ corresponds to an n-fold application of succ to $\mathtt{0}$. Hence, in order to mark variable occurrences with v it suffices to mark the occurrences of $\mathtt{0}$.

Moreover, since the weight of an index coincides with its size, we can write the following formula for $D(z, v, w)$:

$$\begin{aligned} D(z,v,w) &= vw^a z^a + w^b z^b D(z,v,w) \\ &= \frac{vw^a z^a}{1 - w^b z^b}\,. \end{aligned} \tag{3.9}$$

In consequence, the defining equation for $L_\infty(z, v, w)$ is given as

$$\begin{aligned} L_\infty(z,v,w) &= z^c L_\infty(z,v,w) + z^d L_\infty(z,v,w)^2 + \frac{vw^a z^a}{1 - w^b z^b} \\ &= \frac{1}{2z^d}\left(1 - z^c - \sqrt{(1-z^c)^2 - \frac{4vw^a z^{a+d}}{1 - w^b z^b}}\right) \end{aligned} \tag{3.10}$$

where (3.10) follows from the fact that, by its construction, $L_\infty(z, 1, 1)$ coincides with the univariate generating function corresponding to the set of plain λ-terms.

Now, utilising the symbolic moment techniques (see Section 2.1) we can express the mean value μ_n as

$$\mu_n = \frac{[z^n] \displaystyle\int_0^v \frac{dt}{t} \frac{\partial}{\partial w} L_\infty(z,t,w)\bigg|_{v=w=1}}{[z^n] L_\infty(z,1,1)}\,. \tag{3.11}$$

Note that

$$\int_0^v \frac{dt}{t} \frac{\partial}{\partial w} L_\infty(z,t,w)\bigg|_{v=w=1} = -\frac{\left(a - az^b + bz^b\right)\sqrt{(1-z^c)^2 - \frac{4z^{a+d}}{1-z^b}}}{2z^d\left(1 - z^b\right)}\,. \tag{3.12}$$

A straightforward application of the algebraic singularity analysis yields an asymptotic expansion in form of

$$[z^n] \int_0^v \frac{dt}{t} \frac{\partial}{\partial w} L_\infty(z,t,w)\bigg|_{v=w=1} \sim \rho^{-n} \frac{\overline{C} n^{-3/2}}{\Gamma(-\frac{1}{2})} \quad \text{where} \quad \overline{C} = -\frac{a - a\rho^b + b\rho^b}{2\rho^d\left(1 - \rho^b\right)}\,. \tag{3.13}$$

Comparing the obtained constant $\overline{C}$ with the corresponding constant in the asymptotic expansion (3.4) of $[z^n]L_\infty(z)$ we note that

$$\mu_n \xrightarrow[n\to\infty]{} \frac{a - a\rho^b + b\rho^b}{1 - \rho^b} \tag{3.14}$$

which finishes the proof. □

And so, the average de Bruijn index weight tends to a model-specific constant. Moreover, using (3.14) we can easily compute its precise quantity given the specific size notion. For instance, for the natural size notion ($a = b = c = d = 1$) we have

$$\mu_n \xrightarrow[n\to\infty]{} \frac{1}{1-\rho} \approx 1.41964 \tag{3.15}$$

which indeed corresponds to remarkably shallow indices.

Let us conclude model-independent results with the observation that the set of normalising λ-terms does not fall under the 0–1 asymptotic density regime in the set $\mathcal{L}_\infty$ of all λ-terms.

Theorem 3.7. Let $\mathcal{WN}$ be the set of normalising λ-terms. Then

$$0 < \mu^{-}\left(\frac{\mathcal{WN}}{\mathcal{L}_\infty}\right) \quad \text{and} \quad \mu^{+}\left(\frac{\mathcal{WN}}{\mathcal{L}_\infty}\right) < 1\,. \tag{3.16}$$

The proof of the above theorem is analogous to its combinatory logic counterpart (see Theorem 5.9) to which we refer the curious reader.

Combining the above result and that fact that the set of strongly normalising λ-terms is asymptotically negligible in the set of plain λ-terms (see Corollary 3.5), we obtain the following corollary.

Corollary 3.8. Asymptotically almost all weakly normalising λ-terms are not strongly normalising.

3.2 Natural size model

In this section we focus on the natural size model [Ben+16b] in which all constructor contribute one to the term size. Recall that by (3.1) the corresponding generating function $L_\infty(z)$ is then given by

$$L_\infty(z) = \frac{1 - z - \sqrt{(1-z)^2 - \frac{4z^2}{1-z}}}{2z}\,. \tag{3.17}$$

And so, the specific constant C in the asymptotic approximation (3.2) of $[z^n]L_\infty(z)$ can be computed by (3.4) yielding

$$C = -\frac{1}{2\rho_{L_\infty}}\sqrt{\rho_{L_\infty}\frac{Q(\rho_{L_\infty})}{1-\rho_{L_\infty}}}\,. \tag{3.18}$$

Interestingly, using the Maple package gfun [SZ94] it is possible to automatically find the following, surprisingly simple, holonomic specification (i.e. differential equation with polynomial coefficients) of $L_\infty(z)$:

$$z^3 + z^2 - 2z + (z^3 + 3z^2 - 3z + 1)L_\infty(z) + (z^5 + 2z^3 - 4z^2 + z)L'_\infty(z) = 0 \tag{3.19}$$

with the initial condition $L_\infty(0) = 0$. Such an implicit form of $L_\infty(z)$ allows us to derive a simpler, compared to the naive combinatorial definition, recursive scheme defining the successive coefficients of $L_\infty(z)$. Let us denote $[z^n]L_\infty(z)$ as $L_{\infty,n}$. Then, $L_{\infty,n}$ satisfies the following recursion:

$$\begin{aligned} &L_{\infty,0} = 0, \qquad L_{\infty,1} = 1, \qquad L_{\infty,2} = 2, \qquad L_{\infty,3} = 4,\\ &(n+1)L_{\infty,n} = (4n-1)L_{\infty,n-1} - (2n-1)L_{\infty,n-2} - L_{\infty,n-3} - (n-4)L_{\infty,n-4}\,. \end{aligned} \tag{3.20}$$

Note that $L_{\infty,n}$ depends on the previous four values $L_{\infty,n-1}$, $L_{\infty,n-2}$, $L_{\infty,n-3}$ and $L_{\infty,n-4}$. Exploiting this fact, the above recursive relation allows us to compute the exact value $L_{\infty,n}$ using only linear number of arithmetic operations.

We remark that Bacher et al. developed an efficient exact-size sampler for Motzkin trees by attaching a suitable combinatorial interpretation to the holonomic specification of considered trees [BBJ13]. Alas, no effective combinatorial interpretation of (3.19) is known.

Nonetheless, (3.20) enables the efficient computation of the initial values of the sequence $([z^n]L_\infty(z))_{n\in\mathbb{N}}$ known as **A105633** in the Online Encyclopedia of Integer Sequences [Slo64]. The sequence starts as follows:

$$0,\ 1,\ 2,\ 4,\ 9,\ 22,\ 57,\ 154,\ 429,\ 1223,\ 3550,\ 10455,\ 31160,\ 93802,\ \ 284789.$$

Somewhat surprisingly, this sequence corresponds not only to plain λ-terms, but also to two classes of trees, so-called black-white binary trees and binary trees without zigzags (see e.g. [GLM08; STT06]). In what follows, we prove the correspondence between plain λ-terms and black-white binary trees by exhibiting appropriate bijective translations. The corresponding translations involving zigzag-free trees can be found in [Ben+16b].

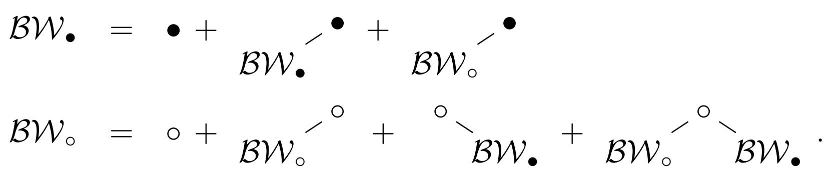
3.2.1 E-free black-white binary trees

Definition 3.9 (Black-white binary trees)**.** A black-white binary tree is a binary tree in which nodes are coloured either black $\bullet$ or white $\circ$. Let E be a finite set of edges with black or white endpoints. An E-free black-white binary tree is a black-white binary tree in which edges from the set E are forbidden. The size of a black-white tree is the total number of its nodes.

In order to show the declared correspondence, we are interested in a particular class of black-white trees with the set $E = \{\, {}_{\bullet}\diagup^{\circ} ,\ {}^{\bullet}\diagdown_{\circ} ,\ {}^{\bullet}\diagdown_{\bullet} ,\ {}^{\circ}\diagdown_{\circ} \,\}$ of forbidden patterns; or dually, the set $A = \{\, {}_{\circ}\diagup^{\bullet} ,\ {}_{\bullet}\diagup^{\bullet} ,\ {}_{\circ}\diagup^{\circ} ,\ {}^{\circ}\diagdown_{\bullet} \,\}$ of allowed edge patterns. For convenience, we simply write black-white trees to denote this class of trees. Moreover, unless otherwise stated, we assume that black-white trees have black roots.

Proposition 3.10. There exists a size-preserving bijection between the set of λ-terms and black-white trees.

Proof. Let $\mathcal{BW}_\bullet$ and $\mathcal{BW}_\circ$ denote the set of black-white trees with a black, respectively white, root. Interpreting the set A of allowed edges combinatorially, we can define both $\mathcal{BW}_\bullet$ and $\mathcal{BW}_\circ$ using the following mutually recursive specifications:



$\mathcal{BW}_\bullet$ = • + $\mathcal{BW}_\bullet$ + $\mathcal{BW}_\circ$
$\mathcal{BW}_\circ$ = ∘ + $\mathcal{BW}_\circ$ + $\mathcal{BW}_\bullet$ + $\mathcal{BW}_\circ$ $\mathcal{BW}_\bullet$.

Such a representation yields the following identities on the corresponding generating functions $BW_\bullet(z)$ and $BW_\circ(z)$:

$$BW_\bullet(z) = z + zBW_\bullet(z) + zBW_\circ(z)\,, \tag{3.21}$$
$$BW_\circ(z) = z + zBW_\circ(z) + zBW_\bullet(z) + zBW_\circ(z)BW_\bullet(z) \tag{3.22}$$

thus

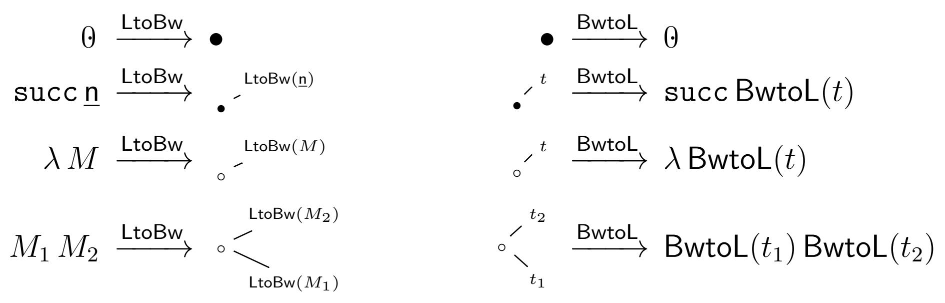
$$(1-z)zBW_\bullet^2(z) - (1-z)^2 BW_\bullet(z) + z = 0\,. \tag{3.23}$$

Note that we can now divide both sides of (3.23) by $(1-z)$ and obtain

$$zBW_\bullet^2(z) - (1-z)BW_\bullet(z) + \frac{z}{1-z} = 0 \tag{3.24}$$

which yields the same solution as for the set of λ-terms (3.17). Both generating functions $BW_\bullet(z)$ and $L_\infty(z)$ are hence equal, so necessarily they represent the same counting sequence. □

The bijective translation LtoBw from λ-terms to black-white trees and the inverse translation BwtoL from black-white trees to λ-terms are given pictorially as follows:

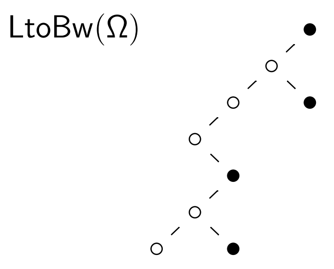


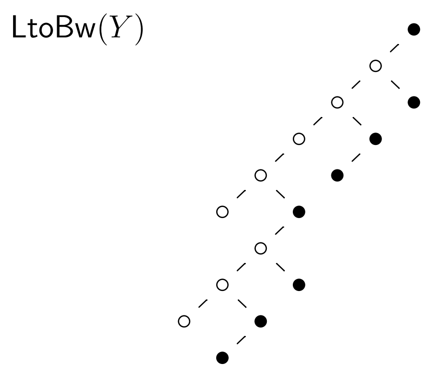
Proposition 3.11. Both LtoBw and BwtoL are mutually inverse bijections.

In order to translate a given black-white tree t into a corresponding λ-term, we decompose t depending on the type of its leftmost node. If t is a single black node $\bullet$, we translate it into $\mathtt{0}$. Otherwise, we have to consider three cases based on the set A of allowed edges and map them into λ-abstraction, successor, or application, respectively.

Example 3.12. Consider the black-white trees corresponding to:

- $\Omega = (\lambda x.xx)(\lambda x.xx) = (\lambda(\underline{0}\underline{0}))\lambda(\underline{0}\underline{0})$, and
- $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) = \lambda(\lambda(\underline{1}\,(\underline{0}\underline{0}))\,\lambda(\underline{1}\,(\underline{0}\underline{0})))$





Our Haskell implementations of LtoBw and BwtoL, tested using QuickCheck [CH00], can be found at [Ben16c].

3.2.2 Neutral λ-terms and β-normal forms

In this subsection we focus on the class $\mathcal{N}$ of β-normal forms and the associated class $\mathcal{M}$ of neutral terms, i.e. normal forms without head abstractions. Specifically, we exhibit a correspondence between both classes of terms and Motzkin numbers (known as **A001006** in the Online Encyclopedia of Integer Sequences [Slo64]) associated with, inter alia, so-called Motzkin trees, i.e. plane unary-binary trees.

Let us start with the following observation.

Proposition 3.13. There exists a size-preserving bijection between the set of λ-terms in normal form and Motzkin trees.

Proof. Note that the combinatorial specification defining normal forms by means of neutral terms can be given as follows:

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

In words, normal forms are either neutral or start with a head abstraction. Neutral terms, in turn, are either de Bruijn indices or in form of an application of a neutral term to a normal form.

Such a specification yields the following system of equations for the corresponding generating functions:

$$\begin{aligned}
N(z) &= M(z) + zN(z)\,,\\
M(z) &= zM(z)N(z) + D(z)\,,\\
D(z) &= zD(z) + z\,.
\end{aligned}$$

Once solved, we obtain the following generating functions:

$$M(z) = \frac{1 - z - \sqrt{(1+z)(1-3z)}}{2z} \quad \text{and} \quad N(z) = \frac{M(z)}{1-z}\,. \tag{3.25}$$

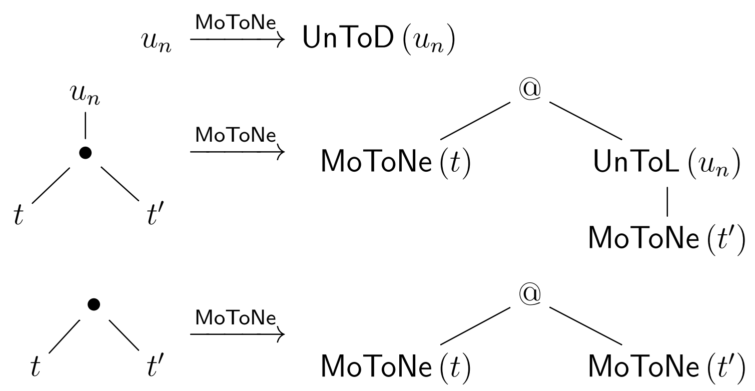
Note that $M(z)$ is in fact the generating function corresponding to the counting sequence of Motzkin numbers (see e.g. [FS09, p. 396]). It means therefore that there exists a size-preserving bijection between Motzkin trees and neutral forms, finishing the proof. $\square$

Similarly to the case of black-white trees, we provide bijective translations between normal forms and Motzkin trees. Let u_n denote the unary path of size $n > 0$. We start by defining two auxiliary operations UnToL and UnToD, translating unary paths into λ-paths (i.e. paths with each vertex labelled by λ) and de Bruijn indices, respectively:

$$\bullet \xrightarrow{\mathsf{UnToL}} \lambda \qquad\qquad \bullet \xrightarrow{\mathsf{UnToD}} 0$$

$$\begin{tikzcd}
\bullet \arrow[d, dash] & & \lambda \arrow[d, dash] & & \bullet \arrow[d, dash] & & \mathtt{succ} \arrow[d, dash] \\
u_n \arrow[rr, "\mathsf{UnToL}"] & & \mathsf{UnToL}\,(u_n) & & u_n \arrow[rr, "\mathsf{UnToD}"] & & \mathsf{UnToD}\,(u_n)
\end{tikzcd}$$

With UnToL and UnToD we are now ready to give the bijective translation MoToNe from Motzkin trees to corresponding neutral terms:



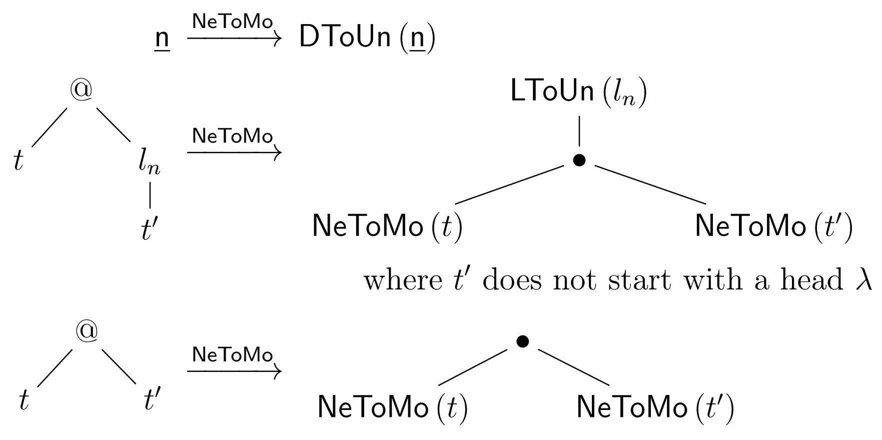
Proposition 3.14. MoToNe is a bijection.

In order to translate a Motzkin tree to a corresponding neutral term we have to consider two cases; either we are given a Motzkin tree starting with a unary node or a Motzkin tree starting with a binary node.

The latter case is straightforward due to the fact that binary nodes correspond to neutral term applications. Suppose that we are given a Motzkin tree starting with a unary path u_n of size n. We have to decide whether the path corresponds to a de Bruijn index or to a chain of λ-abstractions. This distinction is uniquely determined by the existence of the path's *splitting node* – the binary node directly below u_n. If u_n has a splitting node, then it corresponds to a chain of n abstractions which will be placed on top of the corresponding right neutral term constructed recursively from the splitting node of u_n. Otherwise, u_n corresponds to the nth de Bruijn index.

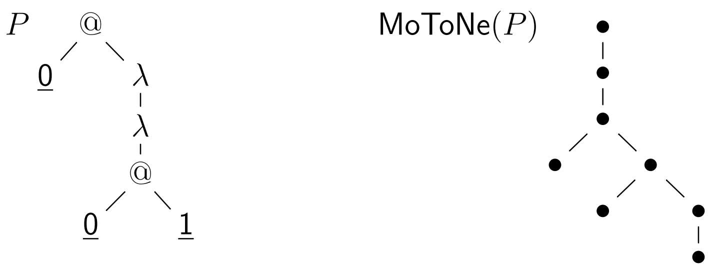
What remains is to give the inverse translation NeToMo from neutral terms to Motzkin trees. Let LToUn and DToUn denote the inverse functions of UnToL and UnToD, respectively. Let l_n denote the unary λ-path of size $n > 0$.

The translation NeToMo is then defined as


$\underline{\mathsf{n}} \xrightarrow{\text{NeToMo}} \text{DToUn}\,(\underline{\mathsf{n}})$
@
t
l_n
t'
NeToMo
LToUn (l_n)
NeToMo (t)
NeToMo (t')
where t' does not start with a head λ
@
t
t'
NeToMo
NeToMo (t)
NeToMo (t')

Proposition 3.15. Both MoToNe and NeToMo are mutually inverse bijections.

Example 3.16. Consider the neutral term $P = \underline{0}\,(\lambda\lambda\underline{0}\underline{1})$. The following figure presents P and its Motzkin tree counterpart through the translation MoToNe.



Note that the simple translation NeToMo allows us to design an effective exact-size sampler for neutral λ-terms in the natural size notion, based on the sampler for Motzkin trees of Bacher et al. [BBJ13]. Given a non-negative integer n, we sample a uniformly random Motzkin tree of size n. Subsequently, we use the NeToMo translation to convert it into a corresponding neutral λ-term. As our translation is linear in time and space, the overall complexity of the described sampler is, on average, linear in both time and space.

3.2.3 Head normal forms

In this subsection we are interested in the class of head normal forms, i.e. λ-terms without head redexes and the associated auxiliary set $\mathcal{K}$ of neutral head normal forms, as defined by the following combinatorial specification:

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

In words, a head normal form either starts with a head λ-abstraction followed by another head normal form or is a neutral head normal form itself. In the latter case, it must be a de Bruijn index or an application of a neutral head normal form to an arbitrary λ-term.

Translating the above specification into a corresponding system of functional equations we obtain:

$$\begin{aligned} H(z) &= K(z) + zH(z)\,, \\ K(z) &= zK(z)L_\infty(z) + D(z) \end{aligned}$$

and hence

$$K(z) = \frac{D(z)}{1 - zL_\infty(z)} \quad \text{whereas} \quad H(z) = \frac{K(z)}{1-z}\,. \tag{3.26}$$

In consequence

$$K(z) = z + zL_\infty(z)\,. \tag{3.27}$$

Note that (3.27) suggests a specific correspondence between the set of neutral head normal forms and the set of plain λ-terms.

Consider the following partial mapping from $\mathcal{K}$ to $\mathcal{L}_\infty$:

$$\begin{aligned} \underline{0}\,N_1N_2\ldots N_m &\mapsto (\lambda\,N_1)N_2\ldots N_m && \text{where } m > 0\,, \\ (\mathsf{succ}\,\underline{\mathsf{n}})\,N_1\ldots N_m &\mapsto \underline{\mathsf{n}}\,N_1\ldots N_m && \text{where } m \geq 0\,. \end{aligned}$$

Note that neutral head normal forms are of size one greater than the size of their plain λ-term counterparts. Since each plain λ-term is either in form of $(\lambda\, N_1)N_2 \ldots N_m$ for some $m > 0$ or $\underline{\mathsf{n}}\, N_1 \ldots N_m$ (in this case m can be equal to 0), the above mapping is surjective which explains the expression $zL_\infty(z)$ in (3.27). The only λ-term in neutral head normal form not covered by the mapping is $\underline{0}$ of size one, hence we have to add the additional z to (3.27).

In consequence, we obtain the following density results.

Proposition 3.17. The asymptotic density of neutral head normal forms in the set of plain terms is equal to $\rho_{L_\infty} \approx 0.29559$.

Proof. Solving (3.27) we obtain the following closed-form expression of $K(z)$:

$$K(z) = z + \frac{1}{2}\left(1 - z - \sqrt{(1-z)^2 - \frac{4z^2}{1-z}}\right). \tag{3.28}$$

A straightforward application of the algebraic singularity analysis gives us the following asymptotic approximation for $[z^n]K(z)$:

$$[z^n]K(z) \sim \rho_{L_\infty}^{-n} \frac{Cn^{-3/2}}{\Gamma(-\frac{1}{2})} \quad \text{where} \quad C = -\frac{1}{2}\sqrt{\rho_{L_\infty} \frac{Q(\rho_{L_\infty})}{1-\rho_{L_\infty}}}. \tag{3.29}$$

Comparing (3.29) with (3.17) we obtain

$$\lim_{n\to\infty} \frac{[z^n]K(z)}{[z^n]L_\infty(z)} = \rho_{L_\infty} \tag{3.30}$$

which finishes the proof. □

Proposition 3.18. The asymptotic density of head normal forms in the set of plain terms is equal to $\frac{\rho_{L_\infty}}{1-\rho_{L_\infty}} \approx 0.41964$.

Proof. From (3.26) we get the following closed-form expression for $H(z)$:

$$H(z) = \frac{1}{2(1-z)}\left(1 - z - \sqrt{(1-z)^2 - \frac{4z^2}{1-z}}\right) + \frac{z}{1-z}. \tag{3.31}$$

As in the case of $K(z)$, a straightforward application of the algebraic singularity analysis gives us the following asymptotic approximation for $[z^n]H(z)$:

$$[z^n]H(z) \sim \rho_{L_\infty}^{-n} \frac{Cn^{-3/2}}{\Gamma(-\frac{1}{2})} \quad \text{where} \quad C = -\frac{1}{2(1-\rho_{L_\infty})}\sqrt{\rho_{L_\infty} \frac{Q(\rho_{L_\infty})}{1-\rho_{L_\infty}}}. \tag{3.32}$$

Comparing (3.32) with (3.17) we get

$$\lim_{n\to\infty} \frac{[z^n]H(z)}{[z^n]L_\infty(z)} = \frac{\rho_{L_\infty}}{1-\rho_{L_\infty}} \tag{3.33}$$

which finishes the proof. □

3.3 Random generation

Standard property-based test generation tools, such as QuickCheck [CH00], provide a set of combinators (i.e. auxiliary higher order functions) facilitating the process of ad-hoc sampler construction. The outcome distribution of generated structures is usually neglected or left for the programmer to control. In consequence, careless sampler design might lead to undesired distributions, favouring certain structures over others or even excluding significant portions of the considered sample space. In this context, the most 'unbiased' distribution is the uniform one, assigning to all structures of equal size the same probability as, e.g. in the rigorous mathematical framework of Boltzmann models.

In the case when we possess a finite combinatorial specification amenable to the techniques of analytic combinatorics and related Boltzmann models, it is relatively straightforward to design an efficient Boltzmann sampler or even have it designed automatically by means of numerical Boltzmann oracles [PSS12] (cf. concrete OCaml implementations [CD09; Dar+12] and Haskell ones [Xia16; Ben16a]). However, if no finite combinatorial specification is available, we can use the method of rejection sampling, i.e. sample random structures from a larger but finitely specifiable class, until we obtain a structure with desired properties. Such an approach has an unfortunate drawback – the main computational cost originates from the number of retrials, rather than the sampling process itself. In this context, positive asymptotic density implies a constant expected number of retrials; however, if this is not the case and the desired property is asymptotically negligible in the larger class of structures, the method of rejection sampling quickly becomes intractable. In consequence, alternative techniques have to be employed.

3.3.1 Closed and typeable λ-terms

From the practical perspective of software verification, in particular compiler testing, the most interesting subclasses of random λ-terms are closed typeable ones [Pał12; CH00]. Unfortunately, no finite admissible combinatorial specification for closed or typeable λ-terms is known. Moreover, due to Corollary 3.5 the set of typeable λ-terms is asymptotically negligible in the set of all λ-terms; hence, rejection sampling becomes intractable already for relatively small term sizes.

Our techniques for random generation of closed typeable λ-terms combine rejection sampling with Prolog systems offering a convenient synergy between logic variables, unification with occurs check and efficient backtracking [BGT16]. Key refinements used in this approach are the following:

(i) Concentrate the expected outcome size around a small finite value;

(ii) Use multiple concurrent threads to speed-up the sampling process;

(iii) Restart the sampling process as soon as it is discovered that the currently constructed λ-term cannot be closed, e.g. due to the occurrence of unbound variables or be typeable, e.g. due to the existence of a non-typeable subterm such as $\lambda x.xx$.

Such an approach allows us to achieve terms of greater size (approximately 120–140 for the natural size notion) compared with naive rejection sampling; nonetheless the intriguing problem of finding direct and efficient methods for random generation of closed typeable λ-terms is still open. Our Prolog sampler implementation is available at [Tar16].

In the case of closed (not necessarily typeable) λ-terms, moderately efficient rejection sampling becomes available as a by-product of the quantitative analysis of so-called m-open λ-terms by Gittenberger and Gołębiewski.

Definition 3.19 (m-open λ-terms)**.** A λ-term N is m-open if $\lambda^m N$, i.e. N with a leading chain of m abstractions, is a closed λ-term.

Naturally, if N is m-open, then it is also $(m+1)$-open. Moreover, the set of 0-open λ-terms is precisely the set of closed λ-terms.

Theorem 3.20 (Gittenberger and Gołębiewski [GG16, Theorem 4])**.** Let ρ be the smallest positive root of $(1-z^b)(1-z^c)^2 - 4z^{a+d}$. Then there exist positive constants $\underline{C}$ and $\overline{C}$ depending on the specific size model and m, such that the number of m-open λ-terms satisfies

$$\liminf_{n\to\infty} \frac{[z^n]L_m(z)}{\underline{C}n^{-3/2}\rho^{-n}} \geq 1 \quad \text{and} \quad \limsup_{n\to\infty} \frac{[z^n]L_m(z)}{\overline{C}n^{-3/2}\rho^{-n}} \leq 1\,. \tag{3.34}$$

In particular, this result asserts the asymptotic density of closed λ-terms in the set of plain ones cannot be equal to zero. Nonetheless, the expected number of retrials is quite significant (roughly 13 in the case of the natural size notion).

An efficient Haskell implementation of a dedicated rejection Boltzmann sampler for closed λ-terms is available as a Cabal package `lambda-sampler` [Ben16b]. Our implementation was tested using QuickCheck [CH00].

3.3.2 Closed h-shallow λ-terms

In large random λ-terms de Bruijn indices are on average bounded by a global constant and so represent variables with small average depth. Hence, the class of closed λ-terms with bounded indices becomes of special interest. In the following subsection we focus on the random generation of this restricted subclasses of so-called closed h-shallow λ-terms.

Definition 3.21 (h-shallow λ-terms)**.** Let $h \geq 1$. A λ-term N is said to be h-shallow if each de Bruijn index in N consists of at most $h-1$ successors. In other words, each index in N is an element of the set $\{\underline{0}, \underline{1}, \ldots, \underline{\mathsf{h}-1}\}$.

Let $\mathcal{L}_m^h$ denote the set of m-open h-shallow λ-terms. Then, the set of all closed λ-terms is equal to $\mathcal{L}_0^\infty$, i.e. the set of 0-open λ-terms with unbounded indices. Let us consider $\mathcal{L}_i^h$ for an arbitrary $i < h$. Note that $\mathcal{L}_i^h$ can be given by means of the following specification:

$$\mathcal{L}_i^h = \lambda\mathcal{L}_{i+1}^h + \mathcal{L}_i^h\mathcal{L}_i^h + \underline{0} + \cdots + \underline{\mathsf{i}-1} \quad \text{for } 0 \leq i < h \tag{3.35}$$

with the defining equation for $\mathcal{L}_h^h$ being

$$\mathcal{L}_h^h = \lambda\mathcal{L}_h^h + \mathcal{L}_h^h\mathcal{L}_h^h + \underline{0} + \cdots + \underline{\mathsf{h}-1}\,. \tag{3.36}$$

In words, the class $\mathcal{L}_i^h$ consists of abstractions in form of $\lambda\mathcal{L}_{i+1}^h$ (note that we have to increase the index i), applications in form of $\mathcal{L}_i^h\mathcal{L}_i^h$ and indices $\underline{0} + \cdots + \underline{\mathsf{i}-1}$. The final equation for $\mathcal{L}_h^h$ is almost identical to the equation for $\mathcal{L}_i^h$ – the only difference is that we do not increase the index for abstractions in form of $\lambda\mathcal{L}_h^h$. Intuitively, such a representation allows us to use the openness index i as a 'counter' for the number of abstraction up to h above in the generated terms.

Let us consider $\mathcal{L}_0^h$, i.e. the set of closed h-shallow λ-terms. Certainly, $\mathcal{L}_0^h$ is defined using a system of $h+1$ equations constituting $\mathcal{L}_0^h, \mathcal{L}_1^h, \ldots, \mathcal{L}_h^h$. Note that in its specification (3.36) $\mathcal{L}_h^h$ depends on itself and no other equation in the system. Therefore, by translating its specification into a functional equation in $L_h^h(z)$ we obtain the following quadratic equation:

$$L_h^h(z) = z^c L_h^h(z) + z^d {L_h^h(z)}^2 + \frac{z^a(1-z^{bh})}{1-z^b} \tag{3.37}$$

which by Riemann's removable singularities theorem (see Theorem 2.18) yields the following solution:

$$L_h^h(z) = \frac{1}{2z^d}\left(1 - z^c - \sqrt{(1-z^c)^2 - \frac{4z^{a+d}\left(1-z^{bh}\right)}{1-z^b}}\right). \tag{3.38}$$

Now, suppose that we have computed the closed-form solution of $L_{i+1}^h(z)$ for some $0 \leq i < h$ and intend to find the closed-form for $L_i^h(z)$, i.e. the generating function corresponding to $\mathcal{L}_i^h$. Note that in its defining equation (3.35) $\mathcal{L}_i^h$ depends on itself and $\mathcal{L}_{i+1}^h$. The corresponding functional equation defining $L_i^h(z)$ is then equal to

$$\begin{aligned} L_i^h(z) &= z^c L_{i+1}^h(z) + z^d L_i^h(z)^2 + \frac{z^a\left(1-z^{bi}\right)}{1-z^b} \\ &= \frac{1}{2z^d}\left(1 - \sqrt{1 - 4z^d\left(z^c L_{i+1}^h(z) + \frac{z^a\left(1-z^{bi}\right)}{1-z^b}\right)}\right) \end{aligned} \tag{3.39}$$

where (3.39) follows again by Riemann's removable singularities theorem. Since we have computed $L_{i+1}^h(z)$, the above equation gives the closed-form solution for $L_i^h(z)$. Iterating this process, it becomes now possible to find the generating function $L_0^h(z)$ and, in particular, evaluate the whole system for each parameter x in the interval $(0, \rho_h)$ where ρ_h denotes the dominating singularity of $L_0^h(z)$.

Gittenberger and Gołębiewski [GG16] prove that ρ_h is in fact the smallest positive root of the innermost radicand of $L_0^h(z)$, i.e.:

$$r_h(z) = (1-z^c)^2 - \frac{4z^{a+d}\left(1-z^{bh}\right)}{1-z^b}. \tag{3.40}$$

Moreover, with h tending to infinity, ρ_h tends to ρ, i.e. the smallest positive root of

$$r(z) = (1-z^c)^2 - \frac{4z^{a+d}}{1-z^b}. \tag{3.41}$$

Since $\overline{r}(z) = (1-z^c)^2\left(1-z^b\right) - 4z^{a+d}$ is concave [GG16, see Proposition 2] and has the same roots as $r(z)$ it is possible to find numerical approximations of ρ using the rapidly converging Newton-Raphson method.

Theorem 3.22 (Thorlund-Petersen [Tho04])**.** Let $f \colon \mathbb{R} \to \mathbb{R}$ be a differentiable concave function on the interval $I = (a, b)$ with a single simple root $\zeta \in I$. Then, the Newton-Raphson sequence $(x_n)_{n \in \mathbb{N}}$ defined as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.42}$$

converges quadratically to ζ for any initial guess $x_0 \in (a, b)$.

Let $\overline{r}_h(z) = \left(1 - z^b\right)(1 - z^c)^2 - 4z^{a+d}\left(1 - z^{bh}\right)$. Certainly, $\overline{r}_h(z)$ has the same roots as $r_h(z)$. Note that

$$\overline{r}_h(z) = \overline{r}(z) + 4z^{a+d+bh}\,. \tag{3.43}$$

In particular $\overline{r}_h(\rho) = 4\rho^{a+d+bh}$ and so ρ_h is relatively close to ρ. We can therefore use the Newton-Raphson method to approximate ρ_h using ρ as the initial guess or, alternatively, use the slower but more robust bisection method in the interval $(\rho, 1)$. Once computed, we can use ρ_h to evaluate $L_0^h(\rho_h)$ and in consequence design an efficient Boltzmann sampler for closed h-shallow λ-terms (see Section 2.1.2).

Our Haskell implementation of a dedicated Boltzmann sampler for closed h-shallow λ-terms, able to generate closed 30-shallow λ-terms of sizes above $100,000$ in a few seconds on a standard PC, is included in the Cabal package `lambda-sampler` [Ben16b].

Chapter 4

Normal-order reduction grammars

In the current chapter we focus on the combinatorial structure of normalising combinators. Specifically, we present an algorithm which, for given n, generates an unambiguous regular tree grammar R_n defining the set of SK-combinators, requiring exactly n normal-order reduction steps to normalise.

Definition 4.1 (Regular tree grammar)**.** A regular tree grammar $G = (A, N, \mathcal{F}, P)$ consists of an axiom A, a set N of non-terminal symbols such that $A \in N$, a set of terminal symbols $\mathcal{F}$ with corresponding arities and a finite set of production rules P of the form $\alpha \to \beta$ where $\alpha \in N$ is a non-terminal and $\beta \in T_{\mathcal{F}}(N)$ is a term in the corresponding term algebra $T_{\mathcal{F}}(N)$, i.e. the set of directed trees built upon terminals $\mathcal{F}$ according to their associated arities.

To build terms of grammar G, we start with the axiom A and use the corresponding derivation relation, denoted by $\to$, as defined through the set of production rules P.

We refer the curious reader to [Com+07] for a detailed exposition on tree grammars.

Example 4.2. Consider the following regular tree grammar defined as $B = (A, N, \mathcal{F}, P)$ where $A := \mathcal{B}$, $N := \{\mathcal{B}\}$, $\mathcal{F} := \{\bullet, \circ(\cdot, \cdot)\}$, and P consists of the two following rules:

$$\begin{aligned} \mathcal{B} &\to \circ(\mathcal{B}, \mathcal{B}); \\ \mathcal{B} &\to \bullet. \end{aligned}$$

Note that B defines the set of terms isomorphic to plane binary trees where leaves correspond to the constant $\bullet$ and inner nodes correspond to the binary terminal $\circ(\cdot, \cdot)$.

In our endeavour, we are recursively constructing regular tree grammars generating sets of SK-combinators. We describe their axioms, terminal and non-terminal symbols here, leaving the algorithm, given in the following section, to define the remaining production rules. And so, the nth grammar R_n will have:

(i) an axiom $A = R_n$;

(ii) a set $\mathcal{F}$ of terminal symbols consisting of two constants S, K and a single binary application operator;

(iii) a set of non-terminal symbols $N_n = \{\mathcal{C}\} \cup \{R_0, \ldots, R_n\}$ where $\mathcal{C}$ denotes the axiom of the set of all combinatory logic terms given by the grammar $\mathcal{C} := S \mid K \mid \mathcal{C}\,\mathcal{C}$.

In other words, the grammar R_n defining terms normalising in n steps, will reference all previous grammars $R_0, \ldots, R_{n-1}$ and the set of all combinatory logic terms $\mathcal{C}$.

Throughout this chapter we adopt the following common definitions and notational conventions. To denote arbitrary combinators, we write $x, y, z, \ldots$. We use lower case letters $\alpha, \beta, \gamma, \delta, \ldots$ to denote *trees*, i.e. elements of the term algebra $T_{\mathcal{F}}(N_n)$ where $N_n =$

$\{\mathcal{C}\} \cup \{R_0, \ldots, R_n\}$ for some n. Capital letters $X, Y, \ldots$ are used to designate one of S or K without specifying which one. We define the *size* of α as the number of applications in α. We say that α is *normal* if either α is of size 0 or $\alpha = X\alpha_1 \ldots \alpha_m$, for some $m \geq 1$ where all $\alpha_1, \ldots, \alpha_m$ are normal. In the latter case we say moreover that α is *complex*. Since we are going to work exclusively with normal trees, we assume that all trees are henceforth normal. We say that a complex α is of *length* m if α is in form of $X\alpha_1 \ldots \alpha_m$. Otherwise, if α is not complex, we say that it is of length 0. The *degree* of α, denoted as $\rho(\alpha)$, is the minimum natural number n such that α does not contain references to any R_i for $i \geq n$. In particular, if α does not reference any reduction grammar, its degree is equal to 0. We use $L_G(\alpha)$ to denote the *language* of α in grammar G, i.e. the set of terms generated by G starting with α. Since R_n does not reference grammars of greater index, we have $L_{R_{\rho(\alpha)-1}}(\alpha) = L_{R_n}(\alpha)$ for arbitrary $n \geq \rho(\alpha)$. And so, for convenience, we use $L(\alpha)$ to denote the language of α in grammar $R_{\rho(\alpha)-1}$ if $\rho(\alpha) > 0$. Otherwise, if $\rho(\alpha) = 0$ we assume that $L(\alpha)$ denotes the language of α in grammar $\mathcal{C}$. Finally, we say that two normal trees are *similar* if both start with the same combinator X and are of equal length.

Example 4.3. Consider the following trees:

(i) $\alpha = S(KR_1)\mathcal{C}$;

(ii) $\beta = K(\mathcal{C}S)R_0$.

Note that both α and β are of size 3 and of equal length 2, although they are not similar since both start with different combinators. Moreover, only α is normal as β has a subtree $\mathcal{C}S$, which is of positive size, but does not start with a combinator. Since α contains a reference to R_1 and no other reduction grammar, its degree is equal to 2, whereas the degree of β is equal to 1.

A crucial observation, which we exploit in our construction, is the fact that normal trees preserve length of generated terms. In other words, if α is of length $m \geq 1$, then any term $x \in L(\alpha)$ is of length m as well, i.e. $x = Xx_1 \ldots x_m$.

4.1 Pseudo-codes and implementation

We state our algorithm using functional pseudo-codes formalising key design subroutines. The adopted syntax echoes basic Haskell notation and built-in primitives, though we allow the use of certain abbreviations making the overall presentation more comprehensible. And so, we use the following data structure representing normal trees.

```
-- | Normal trees.
data Tree = S | K | C | R Int
          | App Tree Tree
```

In our subroutines, we use the following ‘syntactic sugar’ abbreviating the structure of normal trees.

```
  -- | Syntactic sugar.
  X a_1 ... a_m := App X (App a_1 (... App a_{m-1} a_m) ...)
```

Moreover, we allow the use of this abbreviated notation in pattern matching. Specifically, by writing `(X a_1 ... a_m)` we intend to match a complex tree of length m for some $m \in \mathbb{N}$. If multiple arguments are supposed to share the same length, we use the same natural number m, e.g. `(X a_1 ... a_m)` and `(X b_1 ... b_m)`.

Finally, suppose we have a function `f :: a -> [b]` and wish to obtain the set-theoretic union of the image of `[a]` through `f`. In such a case we use the following subroutine.

```
unionMap :: Eq b => (a -> [b]) -> [a] -> [b]
unionMap f xs = nub (concatMap f xs)
```

A working Haskell implementation of our algorithm is available at [Ben16d].

4.2 Algorithm

The key idea used in the construction of reduction grammars is to generate new productions in R_{n+1} based on the productions in R_n. As the base of our inductive construction, we use the set of normal forms R_0 given by

$$R_0 := S \mid K \mid SR_0 \mid KR_0 \mid SR_0R_0\,. \tag{4.1}$$

Clearly, primitive combinators S and K are in normal form. If we take a normal form x, then both $S\,x$ and $K\,x$ are again normal since we did not create any new redex. For the same reason, any term Sx_1x_2 where x_1 and x_2 are normal forms, is itself in normal form. And so, with the above grammar we have captured exactly all redex-free terms.

Let us consider productions of R_0. Note that from both the cases of SR_0 and KR_0 we can abstract a more general rule — if x reduces in n steps, then Sx and Kx reduce in n steps as well, since after reducing x we have no additional redexes left to consider. It follows that any R_n should contain productions SR_n and KR_n. Similarly, from the case of SR_0R_0 we can abstract a more general rule — if Sx_1x_2 reduces in n steps, then both x_1 and x_2 must reduce in total of n steps. The normal-order reduction of Sx_1x_2 proceeds to normalise x_1 and then x_2 in sequence. As there is no head redex, after n steps we obtain a term in normal form. And so, R_n should also contain productions SR_iR_{n-i} for $i \in \{0, \ldots, n\}$.

As we have noticed, all the above productions do not contain head redexes and hence do not increase the total amount of required reduction steps to normalise. Formalising the above observations, we say that α is *short* if either $\alpha = X\alpha_1$ or $\alpha = S\alpha_1\alpha_2$. Otherwise, α is said to be *long*. Hence, we can set a priori the short productions of R_n for $n \geq 1$ and continue to construct the remaining long productions. Naturally, as we consider terms over two primitive combinators S and K, we distinguish two types of long productions, i.e. S- and K-Expansions.

4.2.1 K-Expansions

Let us consider a production $\alpha = X\alpha_1 \ldots \alpha_m$ where $m \geq 0$. The set K-Expansions(α) is defined as

$$\Big\{K(X\alpha_1 \ldots \alpha_k)\mathcal{C}\alpha_{k+1} \ldots \alpha_m \mid k \in \{0, \ldots, m-1\}\Big\}. \tag{4.2}$$

Proposition 4.4. Suppose that $x \in L(K(X\alpha_1 \ldots \alpha_k)\mathcal{C}\alpha_{k+1} \ldots \alpha_m)$. If $x \to_w y$, then $y \in L(X\alpha_1 \ldots \alpha_m)$.

Proof. Let $x = K(Xx_1 \ldots x_k)zx_{k+1} \ldots x_m$. Let us consider its immediate reduct $y = Xx_1 \ldots x_k x_{k+1} \ldots x_m$. Clearly, $x_i \in L(\alpha_i)$ for $i \in \{1, \ldots, m\}$ which finishes the proof. □

In other words, the set K-Expansions(α) has the property that any K-Expansion of α generates terms that reduce in one step to terms generated by α. If we compute the sets K-Expansions(α) for all productions $\alpha \in R_n$, we have almost constructed all of the long K-productions in R_{n+1}. What remains is to include the production $KR_n\mathcal{C}$ as any term $x \in L(KR_n\mathcal{C})$ reduces directly to $y \in L(\alpha)$ for some production $\alpha \in R_n$.

We use the following subroutine computing the set of K-Expansions of a given production.

```haskell
-- | Returns K-Expansions of the given production.
kExpansions :: Tree -> [Tree]
kExpansions p = case p of
    (K a_1 ... a_m) -> kExpansions' K [a_1,...,a_m]
    (S a_1 ... a_m) -> kExpansions' S [a_1,...,a_m]
    where
        kExpansions' _ [] = []
        kExpansions' h [x_1,...,x_k] = K h C x_1 ... x_k
            : kExpansions' (App h x_1) [x_2,...,x_k]
```

4.2.2 S-Expansions

Let us consider a production $\alpha = X\alpha_1 \ldots \alpha_m$ where $m \geq 0$. We would like to define the set S-Expansions(α) similarly to K-Expansions(α), i.e. in such a way that any term generated by an S-Expansion of α reduces in a single step to some $y \in L(\alpha)$. Unfortunately, defining and computing such a set is significantly more complex than the corresponding K-Expansions(α).

Let $q = Xx_1 \ldots x_k z(yz)$. Suppose that $q \in L(\alpha)$ for some production $\alpha \in R_n$. Since $S(Xx_1 \ldots x_k)yz \to_w q$, we would like to guarantee that $S(Xx_1 \ldots x_k)yz \in L(\beta)$ for some $\beta \in$ S-Expansions(α). Assume that $\alpha = X\alpha_1 \ldots \alpha_k \gamma\delta$ where $z \in L(\gamma)$ and $yz \in L(\delta)$. Note that although $z \in L(\gamma)$ and $yz \in L(\delta)$ it might happen that $\delta \neq \delta'\gamma$ for any δ'. Indeed, take $\delta = \mathcal{C}$ and $\gamma = KR_n$. Let $z = Kz_1$ for an arbitrary $z_1 \in L(R_n)$ whereas $y = S$. We have $z = Kz_1 \in L(KR_n)$ and $yz = y(Kz_1) \in L(\mathcal{C})$, however γ is not a suffix of δ.

The above example highlights the main obstacle in finding S-Expansions(α) – given γ, δ such that $z \in L(\gamma)$ and $yz \in L(\delta)$ we cannot decompose δ into $\delta'\gamma$ and set $\beta = S(X\alpha_1 \ldots \alpha_k)\delta'\gamma$ as the resulting S-Expansions(α). Such a set of S-Expansions might generate a language containing additional terms, not reducing in a single step to some $y \in L(\alpha)$. Instead, we have to be more subtle and find $\eta\zeta$ such that $L(\zeta) \subseteq L(\gamma)$ and $L(\eta\zeta) \subseteq L(\delta)$. Using them instead of $\gamma\delta$ we construct a, potentially huge, set $\mathcal{T}$ of trees in form of $S(X\alpha_1 \ldots \alpha_k)\eta\zeta$ constituting S-Expansions(α). Fortunately, it is possible to find a finite set consisting of $\eta\zeta$ such that $L(\mathcal{T})$ captures the indented set of S-Expansions and nothing more.

In order to find such a set, we require an additional 'rewriting' operation on trees that allows us their specialisation, i.e. narrowing the languages they generate. Let us consider the following extension $\rhd$ of the standard derivation relation $\to$:

$$\alpha \rhd \beta \Leftrightarrow \alpha \to \beta \lor (\alpha = \mathcal{C} \land \exists_{n \in \mathbb{N}}\, \beta = R_n) \ . \tag{4.3}$$

In words, $\alpha \rhd \beta$ if β can be derived from α, e.g β is a production of α, or α is a tree representing the set of all combinatory logic terms whereas β represents the set of combinatory logic terms reducing in n steps.

Let $\unrhd$ denote the transitive-reflexive closure of $\rhd$. If $\alpha \unrhd \beta$, we say that α rewrites to β. The important property of $\unrhd$ is the fact that if $\alpha \unrhd \beta$, then $L(\beta) \subseteq L(\alpha)$. Since $\unrhd$ is not a total order on trees, there exist trees incomparable through $\unrhd$, e.g. S and K. To denote the fact that α does not rewrite to β and vice versa, we use the symbol $\alpha \parallel \beta$. In such a case we say that α and β are non-rewritable. Otherwise, if one of them rewrites to the other ($\alpha \unrhd \beta$ or $\beta \unrhd \alpha$), meaning that α and β are rewritable, we use the symbol $\alpha \bowtie \beta$.

Equipped with the rewriting operation $\unrhd$, we have obtained a tool to specialise trees. Now, in order to find appropriate $\eta\zeta$ such that $L(\zeta) \subseteq L(\gamma)$ and $L(\eta\zeta) \subseteq L(\delta)$, it suffices to focus on the rewriting set RewritingSet(γ, δ) of γ and δ, consisting of $\eta\zeta$ such that $\gamma \unrhd \eta$ and $\delta \unrhd \eta\zeta$. In our algorithm, we will guarantee that the constructed rewriting set allows us to find the intended unambiguous set of S-Expansions.

The task of computing RewritingSet(γ, δ) is a fairly straightforward case analysis of δ's structure, except for when $\delta = X\delta_1 \ldots \delta_m$ and $\gamma \parallel \delta_m$. In order to continue our computations, we have to find a set of trees τ such that both $\gamma \unrhd \tau$ and $\delta_m \unrhd \tau$. In order words, the set MeshSet(γ, δ_m) of all trees (also called meshes) τ such that both γ and δ_m rewrite to. In our algorithm, we construct the MeshSet(γ, δ_m) in such a way, that it generates the whole joint sublanguage of both γ and δ_m. Once found MeshSet(γ, δ_m) allows to finish the construction of RewritingSet(γ, δ) and hence also the desired set of S-Expansions.

And so, finding S-Expansions(α) for some $\alpha \in R_{n+1}$ depends on computing appropriate mesh sets which, in turn, might depend on the previously computed grammars $R_0, \ldots, R_n$. For that reason, we start with defining the MeshSet operation. Then, we give the RewritingSet operation, based on the definition of MeshSet. Finally, using RewritingSet we present S-Expansions, which together with K-Expansions constitutes the defining algorithm of reduction grammars $(R_n)_{n \in \mathbb{N}}$.

Mesh Set

In the endeavour of finding appropriate S-Expansions rewritings, we need to find common meshes of given non-rewritable trees $\alpha \parallel \beta$. In other words, a complete partition of $L(\alpha) \cap L(\beta)$ using all possible trees γ such that $\alpha, \beta \unrhd \gamma$. For this purpose, we use the following pseudo-code subroutines.

```
-- | Given X α_1 ... α_m and X β_1 ... β_m computes
-- the family {γ_1, ..., γ_m} of tree meshes.
mesh :: [Tree] -> [Tree] -> [[Tree]]
mesh (x : xs) (y : ys)
     | x `rew` y = [y] : mesh xs ys -- case when x ⊵ y
     | y `rew` x = [x] : mesh xs ys -- case when y ⊵ x
     | otherwise = meshSet x y : mesh xs ys -- case when x ∥ y
mesh [] [] = []
```

The function Mesh, when given two similar productions $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$, constructs a family $\{\gamma_i\}_{i=1}^m$ where each γ_i depends on the comparison of corresponding arguments. In the case when x rewrites to y (denoted as `` x `rew` y `` in the pseudo-code) the singleton $\{y\}$ is constructed. Similarly, when $y \trianglerighteq x$, the singleton $\{x\}$ is constructed. Otherwise, when x and y are both non-rewritable, γ_i is computed using the MeshSet subroutine.

```
-- | Returns the mesh set of given trees.
meshSet :: Tree -> Tree -> [Tree]
meshSet (X a_1 ... a_m) (X b_1 ... b_m) =
        cartesian X [mesh a_i b_i | i <- [1..m]]
meshSet (R k) b @ (X b_1 ... b_m) =
        unionMap (\p -> meshSet p b) (productions $ R k)
meshSet b @ (X b_1 ... b_m) (R k) =
        unionMap (\p -> meshSet b p) (productions $ R k)
meshSet _ _ = []
```

When given two similar trees $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$, MeshSet computes meshes $\gamma_1, \ldots, \gamma_m$ of corresponding arguments α_i and β_i using the subroutine Mesh. Next, argument meshes $\{\gamma_i\}_{i=1}^m$ are used to construct meshes for α and β, using the subroutine Cartesian which computes the Cartesian product $\{X\} \times \gamma_1 \times \cdots \times \gamma_m$ using term application. In the case when one of MeshSet's argument is a reduction grammar R_k and the other α is complex, MeshSet computes recursively mesh sets of α and each production $\delta \in R_k$, outputting their set-theoretic union. In any other case, MeshSet returns the empty set.

Example 4.5. Let $\alpha = KCR_0S$ and $\beta = KS(SR_0\mathcal{C})S$. Consider MeshSet$(\alpha, \beta)$. Both α and β are similar and complex, hence MeshSet proceeds directly to construct mesh sets of corresponding arguments of α and β. Since $\mathcal{C} \trianglerighteq S$, we get $\gamma_1 = \{S\}$. Then, as both R_0 and $SR_0\mathcal{C}$ are non-rewritable, $\gamma_2 =$ MeshSet$(R_0, SR_0\mathcal{C})$. It follows that MeshSet$(R_0, SR_0\mathcal{C})$ is equal to $\bigcup_{\delta \in R_0}$ MeshSet$(\delta, SR_0\mathcal{C})$. Further inspection reveals that MeshSet$(R_0, SR_0\mathcal{C}) = \{SR_0R_0\}$ and thus $\gamma_2 = \{SR_0R_0\}$. Finally, $\gamma_3 = \{S\}$ as S rewrites trivially to itself. Since each γ_i is a singleton, it follows that

$$\text{MeshSet}(\alpha, \beta) = \{KS(SR_0R_0)S\}\,. \tag{4.4}$$

We leave the analysis of MeshSet until we fully define the construction of reduction grammars $(R_n)_{n \in \mathbb{N}}$.

Rewriting Set

Consider our previous example of $q = Xx_1 \ldots x_k z(yz) \in L(\alpha)$ where $\alpha = X\alpha_1 \ldots \alpha_k \gamma\delta$ such that both $z \in L(\gamma)$ and $yz \in L(\delta)$. In order to capture terms reducing to α via an S-redex, we need to find all pairs of trees η, ζ such that $\gamma \trianglerighteq \zeta$ and $\delta \trianglerighteq \eta\zeta$. Since such pairs of trees follow exactly the structure of $z(yz)$ we can use them to define the set S-Expansions(α). And so, to find such rewriting pairs, we use the following RewritingSet pseudo-code subroutine.

```
-- | Given α and β computes their rewriting set.
rewritingSet :: Tree -> Tree -> [Tree]
rewritingSet a S = []
rewritingSet a K = []
rewritingSet a C = [C a]
rewritingSet a (R k) = unionMap
        (\p -> rewritingSet a p) (productions $ R k)
rewritingSet a (X b_1 ... b_m)
        | a `rew` b_m => [X b_1 ... b_m]
        | b_m `rew` a => [X b_1 ... b_{m-1} a]
        | otherwise =>
                cartesian (X b_1 ... b_{m-1}) [meshSet a b_m]
```

The outcome of RewritingSet(α, β) depends on β's structure. If β is a primitive combinator S or K, RewritingSet returns the empty set. If $\beta = \mathcal{C}$, a singleton $\{\mathcal{C}\alpha\}$ is returned. When $\beta = R_k$ for some $k \in \mathbb{N}$, RewritingSet computes recursively the rewriting sets of α and $\gamma \in R_k$, outputting their set-theoretic union. Otherwise when $\beta = X\beta_1 \ldots \beta_m$, RewritingSet determines whether $\alpha \bowtie \beta_m$. If $\alpha \trianglerighteq \beta_m$, a singleton $\{X\beta_1, \ldots, \beta_m\}$ is returned. Conversely, in the case of $\beta_m \trianglerighteq \alpha$, RewritingSet returns $\{X\beta_1, \ldots, \beta_{m-1}\alpha\}$. Finally if α and β_m are non-rewritable, RewritingSet invokes the Cartesian subroutine computing the Cartesian product of $\{X\beta_1, \ldots, \beta_{m-1}\} \times$ MeshSet(α, β_m) using term application, passing afterwards its result as the computed rewriting set.

Example 4.6. Let us consider the rewriting set RewritingSet(S, R_0). Since $\beta = R_0$, we know that RewritingSet$(S, R_0) = \bigcup_{\gamma \in R_0}$ RewritingSet(S, γ). It follows therefore that in order to compute RewritingSet(S, R_0), we have to consider rewriting sets involving productions of R_0. Note that both productions S and K do not contribute new trees. It remains to consider productions SR_0, KR_0 and SR_0R_0. Evidently, each of them is complex and has R_0 as its final argument. Hence, their corresponding rewriting sets are SS, KS and SR_0S, respectively. And so, we obtain that

$$\text{RewritingSet}(S, R_0) = \{SS, KS, SR_0S\}\,. \tag{4.5}$$

Similarly to the case of MeshSet, we postpone the analysis until we define the construction of $(R_n)_{n \in \mathbb{N}}$.

Equipped with the notion of mesh and rewriting sets, we are ready to define the set of S-Expansions. And so, let $\alpha = X\alpha_1 \ldots \alpha_m$ where $m \geq 0$. The set S-Expansions(α) is defined as

$$\Big\{S(X\alpha_1 \ldots \alpha_k)\varphi_l\varphi_r\alpha_{k+3} \ldots \alpha_m \mid k \in \{0, \ldots, m-2\}\Big\} \tag{4.6}$$

where $(\varphi_l\varphi_r) \in$ RewritingSet$(\alpha_{k+1}, \alpha_{k+2})$. We use the following subroutine computing the set of S-Expansions for a given α.

```
-- | Returns S-Expansions of the given production.
sExpansions :: Tree -> [Tree]
sExpansions p = case p of
    (K a_1 ... a_m) -> sExpansions' K [a_1,...,a_m]
    (S a_1 ... a_m) -> sExpansions' S [a_1,...,a_m]
    where
        sExpansions' _ [] = []
        sExpansions' _ [_] = []
        sExpansions' h [x_1,x_2,...,x_k] =
              map (\(App l r) -> S h l r x_3 ... x_m)
              (rewritingSet x_1 x_2) ++
                   sExpansions' (App h x_1) [x_2,...,x_m]
```

Proposition 4.7. Let $x \in L(S(X\alpha_1 \ldots \alpha_k)\varphi_l\varphi_r\alpha_{k+3} \ldots \alpha_m)$. If $x \to_w y$, then $y \in L(X\alpha_1 \ldots \alpha_k\varphi_r(\varphi_l\,\varphi_r)\alpha_{k+3} \ldots \alpha_m)$.

Proof. Let $x = S(Xx_1 \ldots x_k)wzx_{k+3} \ldots x_m$. Let us consider its immediate reduct $y = Xx_1 \ldots x_kz(w\,z)x_{k+3} \ldots x_m$. Clearly, $x_i \in L(\alpha_i)$ for i in proper range. Moreover, both $w \in L(\varphi_l)$ and $z \in L(\varphi_r)$, which finishes the proof. □

4.2.3 Algorithm pseudo-code

With the complete and formal definitions of both S- and K-Expansions we are ready to give the main algorithm Reduction Grammar, which for given $n \in \mathbb{N}$ constructs the grammar R_n.

```
-- | Given $n \in \mathbb{N}$ constructs $R_n$.
reductionGrammar :: Integer -> [Tree]
reductionGrammar 0 = [S, K, S (R 0), K (R 0), S (R 0) (R 0)]
reductionGrammar n = [S (R n), K (R n)]
        ++ [S (R $ n-i) R_i | i <- [0..n]]
        ++ [K (R $ n-1) C]
        ++ concatMap kExpansions (reductionGrammar $ n-1)
        ++ concatMap sExpansions (reductionGrammar $ n-1)
```

The grammar R_0 is given explicitly. Each next grammar R_n consists of five groups of productions where the first two define all the short ones and the last three define the long ones:

(i) $\{ SR_n, KR_n \}$;

(ii) $\{ SR_{n-i}R_i \mid i = 0 \ldots n\}$;

(iii) $\{ KR_{n-i}C \mid i = 0 \ldots n \}$;

(iv) $\bigcup\{$ K-Expansions$(\alpha) \mid \alpha \in R_{n-1} \}$;

(v) $\bigcup\{$ S-Expansions$(\alpha) \mid \alpha \in R_{n-1} \}$.

Example 4.8. Following the construction algorithm, the grammar R_1, defining the set of SK-combinators reducing in a single step to their normal forms, is defined as follows:

$$\begin{aligned} R_1 \;&=\; SR_1 \mid KR_1 \mid SR_0R_1 \mid SR_1R_0 \mid KR_0C \mid KSCR_0 \mid KKCR_0 \mid \\ &\quad KSCR_0R_0 \mid K(SR_0)CR_0 \mid SSSR_0 \mid SSKR_0 \mid SS(SR_0)R_0\,. \end{aligned} \tag{4.7}$$

Let us consider $\alpha = SSSR_0$. Since $\alpha \in$ S-Expansions(SR_0R_0) we get $\alpha \in R_1$. Note that S-Expansions(α) contains $\beta_1 = S(SS)SS$ and $\beta_2 = S(SS)KS$. It follows that $\beta_1, \beta_2 \in R_2$.

4.3 Analysis

Most of our proofs in the following sections are using inductive reasoning on the underlying tree structure. Alas, in certain cases most natural candidates for induction such as tree size fail due to *self-referencing productions*, i.e. productions of R_n which explicitly use the non-terminal symbol R_n. In order to remedy such problems, we introduce the following notion of tree potential.

Definition 4.9 (Tree potential)**.** Let α be a tree. Then, the tree potential $\pi(\alpha)$ of α is defined inductively as follows:

(i) $\pi(S) = \pi(K) = \pi(\mathcal{C}) = 0$;

(ii) $\pi(X\alpha_1 \ldots \alpha_m) = m + \sum_{i=1}^{m} \pi(\alpha_i)$;

(iii) $\pi(R_n) = 1 + \max\limits_{\gamma \in \Phi(R_n)} \pi(\gamma)$.

where $\Phi(R_n)$ denotes the set of productions of R_n which do not use the non-terminal symbol R_n. Such a statement of $\pi(R_n)$ is well-defined due to the fact that R_0 is finite and the algorithm constructing R_{n+1} out of $R_0, \ldots, R_n$ does not use unbounded recursion (see Proposition 4.10).

Moreover, note that such a definition of potential is almost identical to the notion of tree size. The potential of α is the sum of α's size and the weighted sum of all non-terminal grammar symbols occurring in α.

Immediately from the definition we get $\pi(R_0) = 1$. Furthermore, $\pi(R_{n+1}) > \pi(R_n)$ for any $n \in \mathbb{N}$. Indeed, let $\alpha \in R_n$ be the witness of R_n's potential. Now, $(K\alpha\mathcal{C}) \in \Phi(R_{n+1})$ and so R_{n+1} has necessarily greater potential. Moreover, $\pi(\alpha) > \pi(\beta)$ if β is a subtree of α. It follows that the notion of tree potential is a well-suited candidate for the intuitive tree complexity measure.

4.3.1 Soundness

In this section we are interested in the soundness of Reduction Grammar. In particular, we prove that it is computable, terminates on all legal inputs and, for given n, constructs a reduction grammar R_n generating only terms that require exactly n steps to normalise. We start with showing that the rewriting relation is decidable.

Proposition 4.10. It is decidable to check whether $\alpha \trianglerighteq \beta$.

Proof. Induction over $n = \pi(\alpha) + \pi(\beta)$. If $\alpha = X$, then the only tree α rewrites to is X. On the other hand, if $\alpha = \mathcal{C}$, then α rewrites to any β. And so, it is decidable to check whether $\alpha \trianglerighteq \beta$ in case $n = 0$. Now, let us assume that $n > 0$. We have two remaining cases to consider:

(i) If $\alpha = X\alpha_1 \ldots \alpha_m$, then $\alpha \trianglerighteq \beta$ if and only if $\beta = X\beta_1 \ldots \beta_m$ and $\alpha_i \trianglerighteq \beta_i$ for all $i \in \{1, \ldots, m\}$. Since the total potential of $\pi(\alpha_i) + \pi(\beta_i)$ is less than n, we can use the induction hypothesis to decide whether all arguments of α rewrite to the respective arguments of β. It follows that we can decide whether $\alpha \trianglerighteq \beta$;

(ii) If $\alpha = R_k$, then clearly $\alpha \trianglerighteq \beta$ if and only if $\beta = R_k$ or there exists a production $\gamma \in R_k$ such that $\gamma \trianglerighteq \beta$. Let us assume that γ is a production of R_k. Note that if $\gamma \trianglerighteq \beta$, then γ and β are similar. And so, since similarity is decidable, we can rephrase our previous observation as $\alpha \trianglerighteq \beta$ if and only if $\beta = R_k$ or there exists a production $\gamma \in R_k$ such that γ is similar to β and $\gamma \trianglerighteq \beta$. Checking whether $\beta = R_k$ is trivial, so let us assume the other option and start with the case when γ is a short production referencing R_k.

    If $\gamma = XR_k$ is similar to $\beta = X\beta_1$, we know that $\gamma \trianglerighteq \beta$ if and only if $R_k \trianglerighteq \beta_1$. Since $\pi(R_k) + \pi(\beta_1) < n$, we know that checking whether $R_k \trianglerighteq \beta_1$ is decidable, hence so is $\gamma \trianglerighteq \beta$.

    Let us assume w.l.o.g. that $\gamma = SR_kR_0$. Hence, $\beta = S\beta_1\beta_2$. And so, $\gamma \trianglerighteq \beta$ if and only if $R_k \trianglerighteq \beta_1$ and $R_0 \trianglerighteq \beta_2$. Notice that $\pi(R_k) + \pi(\beta_1) < n$ as well as $\pi(R_0) + \pi(\beta_2) < n$. Using the induction hypothesis to both, we get that checking $R_k \trianglerighteq \beta_1$ and $R_0 \trianglerighteq \beta_2$ is decidable, hence so is $\alpha \trianglerighteq \beta$.

    Finally, if γ is a long production we can rewrite it as $\gamma = X\gamma_1 \ldots \gamma_m$, and so reduce this case to the previous one when both trees are complex, as $\pi(\gamma)$ is necessarily smaller than n.

□

Proposition 4.11. Let α, β be two trees. Then, both $\alpha \trianglerighteq \gamma$ and $\beta \trianglerighteq \gamma$ for any $\gamma \in$ MeshSet(α, β).

Proof. Induction over $n = \pi(\alpha) + \pi(\beta)$. Let $M =$ MeshSet(α, β). Clearly, it suffices to consider such α, β that $M \neq \varnothing$.

Let us assume that both $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$. If $\alpha_i \bowtie \beta_i$ for all $i \in \{1, \ldots, m\}$, then M consists of a single tree $\gamma = X\gamma_1 \ldots \gamma_m$ for which $\alpha_i, \beta_i \trianglerighteq \gamma_i$. Evidently, our claim holds. Suppose that there exists an $i \in \{1, \ldots, m\}$ such that $\alpha_i \parallel \beta_i$. Since $\pi(\alpha_i) + \pi(\beta_i) < n$, we can apply the induction hypothesis to MeshSet(α_i, β_i). The set $M' = \textsc{MeshSet}(\alpha_i, \beta_i)$ cannot be empty and so let δ_i be an arbitrary mesh in M'. We know that $\alpha_i, \beta_i \trianglerighteq \delta_i$. And so, if we consider an arbitrary $\gamma = X\gamma_i \ldots \gamma_m \in M$, we get $\alpha_i, \beta_i \trianglerighteq \gamma_i$ for all $i \in \{1, \ldots, m\}$, which implies our claim.

What remains is to consider the case when either $\alpha = R_k$ and β is complex or, symmetrically, $\beta = R_k$ and α is complex. Let us assume w.l.o.g. the former case. From the definition, $\textsc{MeshSet}(R_k, \beta)$ depends on the union of $\textsc{MeshSet}(\gamma, \beta)$ for $\gamma \in R_k$. Clearly, R_k rewrites to any of its productions. Let $\gamma \in R_k$ be a production referencing R_k. We have to consider two cases based on the structure of γ:

(i) Let $\gamma = XR_k$. Then, $\pi(\gamma) = \pi(R_k) + 1$ and so we cannot use the induction hypothesis to $\textsc{MeshSet}(\gamma, \beta)$ directly. Note however, that we can assume that $\beta = X\beta_1$, since otherwise $\textsc{MeshSet}(\gamma, \beta)$ would be empty. Therefore, we know that $\textsc{MeshSet}(R_k, \beta_1) \neq \varnothing$ to which we can now use the induction hypothesis, as $\pi(R_k) + \pi(\beta_1) < n$. Immediately, we get that $R_k, \beta \trianglerighteq \gamma$;

(ii) W.l.o.g. let $\gamma = SR_kR_0$. Then, $\pi(\gamma) = 3 + \pi(R_k)$. Again, we cannot directly use the induction hypothesis. Note however, that we can assume that $\beta = S\beta_1\beta_2$. And so we get $\pi(R_k) + \pi(\beta_1) < n$ and $\pi(R_0) + \pi(\beta_2) < n$. Using the induction hypothesis to both parts we conclude that $R_k, \beta \trianglerighteq \gamma$ in this case as well.

To finish the proof we need to show that our claim holds for all $\gamma \in R_k$ which do not reference R_k. Indeed, any such production has necessarily smaller potential than R_k, and so, we can use the induction hypothesis directly to the resulting mesh set. Evidently, our claim holds. □

In other words, $\textsc{MeshSet}(\alpha, \beta)$ is in fact a set of meshes, i.e. trees generating a joint portion of $L(\alpha)$ and $L(\beta)$. Note, that along the lines of proving the above proposition, we have also showed that indeed $\textsc{MeshSet}(\alpha, \beta)$ terminates on all legal inputs, as the number of recursive calls cannot exceed $2(\pi(\alpha) + \pi(\beta))$; in the worst case, every second recursive call decreases the total potential sum of its inputs.

Proposition 4.12. Let α, β be two trees. Then, $\alpha \trianglerighteq \varphi_r$ and $\beta \trianglerighteq \varphi_l\varphi_r$ for any $\varphi_l\varphi_r \in \textsc{RewritingSet}(\alpha, \beta)$.

Proof. We can assume that $\textsc{RewritingSet}(\alpha, \beta) \neq \varnothing$, as otherwise our claim trivially holds. Let $\varphi_l\varphi_r \in \textsc{RewritingSet}(\alpha, \beta)$. Based on the structure of β, we have to three cases to consider:

(i) If $\beta = \mathcal{C}$, then $\varphi_l\varphi_r = \mathcal{C}\alpha$. Immediately, $\alpha \trianglerighteq \alpha$ and $\mathcal{C} \trianglerighteq \mathcal{C}\,\alpha$;

(ii) If $\beta = X\beta_1 \ldots \beta_m$, then we have again exactly three possibilities. Both cases when $\alpha \bowtie \beta_m$ are trivial, so let us assume that $\alpha \parallel \beta_m$. It follows that there exists such a $\gamma \in \textsc{MeshSet}(\alpha, \beta_m)$ that $\varphi_l\varphi_r = X\beta_1 \ldots \beta_{m-1}\gamma$. Due to Proposition 4.11, we know that $\alpha, \beta_m \trianglerighteq \gamma$ and so directly that $\alpha \trianglerighteq \varphi_r$ and $\beta \trianglerighteq \varphi_l\varphi_r$;

(iii) Finally, suppose that $\beta = R_n$. Then, there exists a production $\gamma \in R_n$ such that $\varphi_l\varphi_r \in \textsc{RewritingSet}(\alpha, \gamma)$. Note however, that in this situation $\gamma = X\gamma_1 \ldots \gamma_m$ and so we can reduce this case to the already considered case above.

□

Now we are ready to give the anticipated soundness theorem.

Theorem 4.13 (Soundness)**.** If $x \in L(R_n)$, then x reduces in n steps.

Proof. Induction over pairs (n, m) where m denotes the length of a minimal, in terms of length, derivation Σ of $x \in L(R_n)$. Let $n = 0$ and so $x \in L(R_0)$. If $m = 1$, then $x \in \{S, K\}$ hence x is already in normal form. Suppose that $m > 1$. Clearly, $x \notin \{S, K\}$. Let $R_0 \to \alpha$ be the first production rule used in derivation Σ. Using the induction hypothesis to the remainder of the derivation, we know that x does not contain any nested redexes. Moreover, α avoids any head redexes and so we get that x is in normal form.

Let $n > 0$. We have to consider several cases based on the choice of the first production rule $R_n \to \alpha$ used in the derivation Σ:

(i) $\alpha = SR_n$ or $\alpha = KR_n$. Using the induction hypothesis we know that $x = Xy$ where y reduces in n steps. Naturally, so does x;

(ii) $\alpha = SR_{n-i}R_i$ for some $i \in \{0, \ldots, n\}$. Then, $x = Syz$ where $y \in L(R_{n-i})$ and $z \in L(R_i)$. Note that both their derivations are in fact shorter than the derivation of x and thus applying the induction hypothesis to both y and z we know that they reduce in $n - i$ and i steps, respectively. Following the normal-order reduction strategy, we note that y and z reduce sequentially in x. Since x does not contain a head redex itself, we reduce it in total of n reductions;

(iii) $\alpha = KR_{n-1}\mathcal{C}$. Directly from the induction hypothesis we know that $x = Kyz$ where y reduces in $n - 1$ steps. And so $x \to_w y$, implying that x reduces in n steps;

(iv) $\alpha = K(X\alpha_1 \ldots \alpha_k)\mathcal{C}\alpha_{k+1} \ldots \alpha_m$. Let $x \in L(\alpha)$. Clearly, x has a head redex and so let $x \to_w y$. Using Proposition 4.4, we know that $y \in L(X\alpha_1 \ldots \alpha_m)$. Moreover, by the construction of R_n we get $\alpha \in$ K-Expansions$(X\alpha_1 \ldots \alpha_m)$ and therefore $y \in L(R_{n-1})$. It follows that y reduces in $n - 1$ steps and so x in n steps;

(v) $\alpha = S(X\alpha_1 \ldots \alpha_k)\varphi_l\varphi_r\alpha_{k+3} \ldots \alpha_m$. Let $x \in L(\alpha)$. Clearly, x has a head redex and so let $x \to_w y$. Due to Proposition 4.7 we get that $y \in L(X\alpha_1 \ldots \alpha_k\varphi_r(\varphi_l\,\varphi_r)\alpha_{k+3} \ldots \alpha_m)$. In order to show that x reduces in n steps it suffices to show that $y \in L(R_{n-1})$. Let us consider β such that $\alpha \in$ S-Expansions(β). From the structure of α we can rewrite it as $\beta = X\alpha_1 \ldots \alpha_k\alpha_{k+1}\alpha_{k+2} \ldots \alpha_m$. Moreover, from Proposition 4.12 we know that $\alpha_{k+1} \trianglerighteq \varphi_r$ and $\alpha_{k+2} \trianglerighteq \varphi_l\,\varphi_r$. Clearly, $y \in L(\beta)$, which finishes the proof.

□

Combining the above result with the fact that each normalising combinatory logic term reduces in a determined number of normal-order reduction steps, gives us the following corollary.

Corollary 4.14. If $L(R_n) \cap L(R_m) \neq \varnothing$, then $n = m$.

4.3.2 Completeness

In this section we are interested in the completeness of Reduction Grammar. Specifically, we show that every term normalising in exactly n steps is generated by R_n.

W start with some auxiliary lemmas showing the completeness of MeshSet and, in consequence, RewritingSet.

Lemma 4.15. Let α, β be two non-rewritable trees and x be a term. Then, $x \in L(\alpha) \cap L(\beta)$ if and only if there exists a mesh $\gamma \in \textsc{MeshSet}(\alpha, \beta)$ such that $x \in L(\gamma)$.

Proof. It suffices to show the sufficiency part, the necessity is clear from Proposition 4.11. We show this result using induction over the size $|x|$ of x. Let $x \in L(\alpha) \cap L(\beta)$. Let us start with noticing that $|\alpha| + |\beta| > 0$. Moreover, there are only two cases where $x \in L(\alpha) \cap L(\beta)$, i.e. when either $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$ or when exactly one of them is equal to some R_n and the other is complex. And so, let us consider these cases separately:

(i) Suppose that $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$. It follows that we can rewrite x as $Xx_1 \ldots x_m$ such that $x_i \in L(\alpha_i) \cap L(\beta_i)$. Clearly, if all $\alpha_i \bowtie \beta_i$, then there exists a mesh γ such that $x \in L(\gamma)$. Let us assume that some α_i and β_i are non-rewritable. Then, using the induction hypothesis we find a mesh $\gamma_i \in \textsc{MeshSet}(\alpha_i, \beta_i)$ such that $x_i \in L(\gamma_i)$. Immediately, we get that there exists a mesh in $\textsc{MeshSet}(\alpha, \beta)$ which generates x;

(ii) Let us assume w.l.o.g. that $\alpha = R_n$ and $\beta = X\beta_1 \ldots \beta_m$. Since $x \in L(R_n)$, there must be such a production $\gamma \in R_n$ that $x \in L(\gamma)$. Although the size of x does not decrease, note that we can reduce this case to the one considered above since both γ and β are complex. Clearly, it follows that we can find a suiting mesh $\delta \in \textsc{MeshSet}(\gamma, \beta)$ such that $x \in L(\delta)$. Immediately, we get $\delta \in \textsc{MeshSet}(\alpha, \beta)$ which finishes the proof.

□

Lemma 4.16. Let α, β be two trees and x, yx be two terms. Then, $x \in L(\alpha)$ and $yx \in L(\beta)$ if and only if there exists such a $\varphi_l\varphi_r \in \textsc{RewritingSet}(\alpha, \beta)$ that $x \in L(\varphi_r)$ and $yx \in L(\varphi_l\varphi_r)$.

Proof. Due to Proposition 4.12 the necessity is clear. What remains is to show the sufficiency part. Let $x \in L(\alpha)$ and $yx \in L(\beta)$. Consider the structure of β. If $\beta = \mathcal{C}$, then $\mathcal{C}\alpha \in \textsc{RewritingSet}(\alpha, \beta)$ and so $\varphi_l = \mathcal{C}, \varphi_r = \alpha$. Clearly, our claim holds. Now, consider the case when $\beta = X\beta_1 \ldots \beta_m$. Based on the rewritability of α and β_m we distinguish three subcases:

(i) If $\alpha \trianglerighteq \beta_m$, then $X\beta_1 \ldots \beta_m \in \textsc{RewritingSet}(\alpha, \beta)$. Since $yx \in L(\beta)$, we get $x \in L(\beta_m)$ and in consequence $x \in L(\varphi_r)$;

(ii) If $\beta_m \trianglerighteq \alpha$, then $X\beta_1 \ldots \beta_{m-1}\alpha \in \textsc{RewritingSet}(\alpha, \beta)$. Since $\beta_m \trianglerighteq \alpha$, we know that $L(\alpha) \subseteq L(\beta_m)$ and so $yx \in L(X\beta_1 \ldots \beta_{m-1}\alpha)$;

(iii) If $\alpha \parallel \beta_m$, then we know that $x \in L(\alpha) \cap L(\beta_m)$. If not, then yx could not be a term of $L(\beta)$. And so, using Lemma 4.15 we find a mesh $\gamma \in \textsc{MeshSet}(\alpha, \beta_m)$ such that $x \in L(\gamma)$. We know that $X\beta_1 \ldots \beta_{m-1}\gamma \in \textsc{RewritingSet}(\alpha, \beta)$. Clearly, it is the tree we were looking for.

It remains to consider the case when $\beta = R_k$. Note however, that it can be reduced to the case when $\beta = X\beta_1 \ldots \beta_m$. Indeed, since $x \in L(R_k)$, then there exists a production $\gamma \in R_k$ such that $x \in L(\gamma)$. From the previous arguments we know that we can find a tree satisfying our claim. $\square$

Using the above completeness results for MeshSet and RewritingSet, we are ready to give the anticipated completeness result of $(R_n)_{n \in \mathbb{N}}$.

Theorem 4.17 (Completeness)**.** If x reduces in n steps, then $x \in L(R_n)$.

Proof. Induction over pairs (n, s) where s denotes the size of x. The base case $n = 0$ is clear due to the completeness of R_0. Let $n > 0$.

Let us start with considering short terms. Let $x = Xy$ be a term of size s. Since x has no head redex, y must reduce in n steps as well. Now, we can apply the induction hypothesis to y and deduce that $y \in L(R_n)$. It follows that $x \in L(XR_n)$. Clearly, XR_n is a production of R_n and so $x \in L(R_n)$. Now, assume that $x = Syz$. Since x reduces in n steps and does not contain a head redex, there exists such an $i \in \{0, \ldots, n\}$ that y reduces in i steps and z reduces in $n - i$ steps. Applying the induction hypothesis to both y and z, we get that $y \in L(R_i)$ whereas $z \in L(R_{n-i})$. Immediately, we get that $x \in L(R_n)$ as $SR_iR_{n-i} \in R_n$.

What remains is to consider long terms. Let $x = Kx_1x_2$. Note that x_1 must reduce in $n-1$ steps, as $x \to_w x_1$. And so, from the induction hypothesis we get that $x_1 \in L(R_{n-1})$. Now we have $x \in L(KR_{n-1}\mathcal{C})$ and hence $x \in L(R_n)$ as $KR_{n-1}\mathcal{C}$ is a production of R_n.

Now, let $x = Kx_1 \ldots x_m$ for $m \geq 3$. Since x has a head redex, we know that $x \to_w y = x_1x_3 \ldots x_m$, which itself reduces in $n-1$ steps. Let us rewrite y as $Xy_1 \ldots y_kx_3 \ldots x_m$ where $x_1 = Xy_1 \ldots y_k$. We know that there exists a production $\alpha \in R_{n-1}$ such that $y \in L(\alpha)$. Let $\alpha = X\overline{\alpha_1} \ldots \overline{\alpha_k}\alpha_3 \ldots \alpha_m$. Clearly, there exists a $\beta = K(X\overline{\alpha_1} \ldots \overline{\alpha_k})\mathcal{C}\alpha_3 \ldots \alpha_m \in$ K-Expansions(α). We claim that $x \in L(\beta)$. Indeed, $y \in L(\alpha)$ implies that $y_i \in L(\overline{\alpha_i})$ and $x_j \in L(\alpha_j)$ for any i and j in proper ranges. Since $x_2 \in L(\mathcal{C})$, we conclude that $x \in L(\beta)$ and hence $x \in L(R_n)$.

Let $x = Sx_1 \ldots x_m$ for $m \geq 3$. Since x has a head redex $x \to_w y = x_1x_3(x_2x_3)x_4 \ldots x_m$ which reduces in $n - 1$ steps. Again, let us rewrite y as $Xy_1 \ldots y_kx_3(x_2x_3)x_4 \ldots x_m$ where $x_1 = Xy_1 \ldots y_k$. Now, since $y \in L(R_{n-1})$, there must exist a production $\alpha = X\overline{\alpha_1} \ldots \overline{\alpha_k}\alpha_3\gamma\alpha_4 \ldots \alpha_m \in R_{n-1}$ such that $y \in L(\alpha)$. We claim that there exists a production $\beta \in$ S-Expansions(α) such that $x \in L(R_n)$. If so, the proof would be complete. Notice that $x_3 \in L(\alpha_3)$ and $x_2x_3 \in L(\gamma)$. Using Lemma 4.16 we know that there exists a tree $\varphi_l\varphi_r \in$ RewritingSet(α_3, γ) such that $x_3 \in L(\varphi_r)$ and $(x_2x_3) \in L(\varphi_l\varphi_r)$. And so $y \in L(X\overline{\alpha_1} \ldots \overline{\alpha_k}\varphi_r(\varphi_l\varphi_r)\alpha_4 \ldots \alpha_m)$. Moreover, due to the fact that $\varphi_l\varphi_r \in$ RewritingSet(α_3, γ), we know that the tree $\beta = S(X\overline{\alpha_1} \ldots \overline{\alpha_k})\varphi_l\varphi_r\alpha_4 \ldots \alpha_m \in$ S-Expansions(α) and so also $\beta \in R_n$. Since $x_2 \in L(\varphi_l)$, we get that $x \in L(\beta)$. $\square$

4.3.3 Unambiguity

In this section we show that reduction grammars are in fact unambiguous, i.e. every term $x \in L(R_n)$ has exactly one derivation. Due to the mutual recursive nature of MeshSet, RewritingSet and ReductionGrammar, we split the proof into two separate parts. In the following lemma, we show that MeshSet returns unambiguous meshes under the assumption that $R_0, \ldots, R_n$ up to some n are themselves unambiguous. In the corresponding theorem we use inductive reasoning which supplies the aforementioned assumption and thus, as a consequence, allows us to prove the main result.

Lemma 4.18. Let α, β be two trees such that $\gamma, \overline{\gamma} \in \textsc{MeshSet}(\alpha, \beta)$ where in addition $\rho(\alpha), \rho(\beta) \leq r + 1$. If $R_0, \ldots, R_r$ are unambiguous and $L(\gamma) \cap L(\overline{\gamma}) \neq \varnothing$, then $\gamma = \overline{\gamma}$.

Proof. Induction over $n = \pi(\alpha) + \pi(\beta)$. Let $x \in L(\gamma) \cap L(\overline{\gamma})$. We can assume that $|\textsc{MeshSet}(\alpha, \beta)|$ is greater than 1 as the case for $|\textsc{MeshSet}(\alpha, \beta)| = 1$ is trivial. In consequence, the base case $n = 0$ is clear as the resulting MeshSet for two trees of potential 0 has to be necessarily empty. Hence, we have to consider two cases based on the structure of α and β:

(i) Let $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$. Clearly, x is in form of $x = Xx_1 \ldots x_m$. Let $\alpha_i \parallel \beta_i$ be an arbitrary non-rewritable pair of arguments in α, β. It follows that $x_i \in L(\alpha_i) \cap L(\beta_i)$ and so, due to Lemma 4.15, there exists a mesh $\delta \in \textsc{MeshSet}(\alpha_i, \beta_i)$ such that $x_i \in L(\delta)$. Let $M_i = \textsc{MeshSet}(\alpha_i, \beta_i)$. Since $\pi(\alpha_i) + \pi(\beta_i) < n$ we can use the induction hypothesis to M_i and immediately conclude that δ is the only mesh in M_i generating x_i. And so, we know that γ and $\overline{\gamma}$ are equal on the non-rewritable arguments of α, β. Note that if $\alpha_i \bowtie \beta_i$, then both contribute a single mesh at position i. Immediately, we get that both γ and $\overline{\gamma}$ are also equal on the rewritable arguments of α and β, hence finally $\gamma = \overline{\gamma}$;

(ii) W.l.o.g. let $\alpha = R_k$ and $\beta = X\beta_1 \ldots \beta_m$. Clearly, as $\rho(\alpha) \leq r + 1$, we know that R_k is unambiguous. From the definition of MeshSet there exist productions $\delta, \overline{\delta} \in R_k$ such that $\gamma \in \textsc{MeshSet}(\delta, \beta)$ and $\overline{\gamma} \in \textsc{MeshSet}(\overline{\delta}, \beta)$. We claim that $\gamma = \overline{\gamma}$ as otherwise $\delta, \overline{\delta}$ would generate a common term. Suppose that $\gamma \neq \overline{\gamma}$. From Lemma 4.15 we know that $L(\gamma) \subseteq L(\delta)$ and $L(\overline{\gamma}) \subseteq L(\overline{\delta})$. Since $x \in L(\gamma) \cap L(\overline{\gamma})$, we get that $x \in L(\delta) \cap L(\overline{\delta})$ and therefore a contradiction with the fact that R_k is unambiguous. It follows that $\gamma = \overline{\gamma}$, which finishes the proof.

□

Theorem 4.19 (Unambiguity)**.** Let $\alpha, \beta \in R_n$. If $L(\alpha) \cap L(\beta) \neq \varnothing$, then $\alpha = \beta$.

Proof. Induction over n. Let $x \in L(\alpha) \cap L(\beta)$. Note that if $x \in L(\alpha) \cap L(\beta)$, then both α, β must be similar. We can therefore focus on similar productions of R_n. For that reason, we immediately notice that R_0 satisfies our claim.

Let $n > 0$. Since R_n does not contain combinators as productions, we can rewrite both α as $X\alpha_1 \ldots \alpha_m$ and β as $X\beta_1 \ldots \beta_m$. Let us consider several cases based on their common structure:

(i) Let $X = K$. If $m = 1$, then α and β are equal as there is exactly one short K-production in R_n. If $m = 2$, then again $\alpha = \beta$, since there is a unique K-production $KR_{n-1}\mathcal{C}$ of length two in R_n. If $m > 2$, then both are K-Expansions of some productions in R_{n-1}. And so

$$\begin{aligned} \alpha &= K(X\overline{\alpha_1} \ldots \overline{\alpha_k})\mathcal{C}\alpha_3 \ldots \alpha_m \in \textsc{K-Expansions}(\gamma)\,, && (4.8) \\ \beta &= K(X\overline{\beta_1} \ldots \overline{\beta_k})\mathcal{C}\beta_3 \ldots \beta_m \in \textsc{K-Expansions}(\delta) && (4.9) \end{aligned}$$

where

$$\begin{aligned}\gamma &= X\overline{\alpha_1}\ldots\overline{\alpha_k}\alpha_3\ldots\alpha_m\,, &(4.10)\\ \delta &= X\overline{\beta_1}\ldots\overline{\beta_k}\beta_3\ldots\beta_m\,. &(4.11)\end{aligned}$$

Since $x \in L(\alpha) \cap L(\beta)$, we can assume that x is in form of $K(Xy_1 \ldots y_k)x_2x_3 \ldots x_m$ where $y_i \in L(\overline{\alpha_i}) \cap L(\overline{\beta_i})$ and $x_j \in L(\alpha_j) \cap L(\beta_j)$. It follows that we can use the induction hypothesis to $\gamma, \delta \in R_{n-1}$ obtaining $\overline{\alpha_i} = \overline{\beta_i}$ and $\alpha_j = \beta_j$. Immediately, we get $\alpha = \beta$;

(ii) Let $X = S$. If $m = 1$, then α and β are equal due to the fact that there is exactly one S-production of length one in R_n. If $m = 2$, then α, β are in form of $\alpha = SR_iR_{n-i}$ and $\beta = SR_jR_{n-j}$. Hence, $x = Sx_1x_2$ for some terms x_1, x_2. Since $x_1 \in L(R_i) \cap L(R_j)$ and $x_2 \in L(R_{n-i}) \cap L(R_{n-j})$, we know that $i = j$ due to Corollary 4.14 and thus $\alpha = \beta$. It remains to consider long S-productions. Let

$$\begin{aligned}\alpha &= S(X\overline{\alpha_1}\ldots\overline{\alpha_k})\varphi_l\varphi_r\alpha_4\ldots\alpha_m \in \text{S-E}\textsc{xpansions}(\gamma)\,, &(4.12)\\ \beta &= S(X\overline{\beta_1}\ldots\overline{\beta_k})\overline{\varphi_l\varphi_r}\beta_4\ldots\beta_m \in \text{S-E}\textsc{xpansions}(\delta) &(4.13)\end{aligned}$$

where

$$\begin{aligned}\gamma &= X\overline{\alpha_1}\ldots\overline{\alpha_k}\alpha_2\alpha_3\alpha_4\ldots\alpha_m\,, &(4.14)\\ \delta &= X\overline{\beta_1}\ldots\overline{\beta_k}\beta_2\beta_3\beta_4\ldots\beta_m\,. &(4.15)\end{aligned}$$

It follows that we can rewrite x as $S(Xy_1 \ldots y_k)wzx_4 \ldots x_m$. Let us focus on the reduct $x \to_w y = Xy_1 \ldots y_kz(wz)x_4 \ldots x_m$. Evidently, $y \in L(\gamma) \cap L(\delta)$ and so according to the induction hypothesis we know that $\gamma = \delta$, in particular $\alpha_2 = \beta_2$ and $\alpha_3 = \beta_3$. Hence, both $\varphi_l\varphi_r$ and $\overline{\varphi_l\varphi_r}$ are elements of the same RewritingSet. If we could guarantee that $\varphi_l\varphi_r = \overline{\varphi_l\varphi_r}$, then immediately $\alpha = \beta$ and the proof is finished. From the construction of the RewritingSet we have two cases left to consider:

(i) If $\alpha_3 = X\gamma_1 \ldots \gamma_m$, then both $\varphi_l\varphi_r$ and $\overline{\varphi_l\varphi_r}$ are either in form of $X\gamma_1 \ldots \gamma_{m-1}\,\varphi_r$ or $X\gamma_1 \ldots \gamma_{m-1}\,\overline{\varphi_r}$. It follows that $\varphi_l = \overline{\varphi_l}$. It remains to show that $\varphi_r = \overline{\varphi_r}$. Note that $\rho(\alpha_2), \rho(\alpha_3) \leq n$ since both $\gamma, \delta \in R_{n-1}$. Moreover, from the induction hypothesis we know that $R_0, \ldots, R_{n-1}$ are unambiguous. And so, since $z \in L(\varphi_r) \cap L(\overline{\varphi_r})$, we can use Lemma 4.18 to conclude that $\varphi_r = \overline{\varphi_r}$;

(ii) If $\alpha_3 = R_k$, then necessarily there exist such productions $\eta, \overline{\eta} \in R_k$ that $\varphi_l\varphi_r \in \text{R}\textsc{ewriting}\text{S}\textsc{et}(\alpha_2, \eta)$ whereas $\overline{\varphi_l\varphi_r} \in \text{R}\textsc{ewriting}\text{S}\textsc{et}(\alpha_2, \overline{\eta})$. Due to Proposition 4.12, we know that $L(\varphi_l\varphi_r) \subseteq L(\eta)$ and $L(\overline{\varphi_l\varphi_r}) \subseteq L(\overline{\eta})$. It implies that $wz \in L(\eta) \cap L(\overline{\eta})$, however, since $k < n$, we know from the induction hypothesis that R_k is unambiguous. Hence $\eta = \overline{\eta}$. Finally, it means that we can reduce this case to one of the previous cases when α_3 is complex, concluding that $\varphi_l\varphi_r = \overline{\varphi_l\varphi_r}$.

□

4.3.4 Generating functions

Fix an arbitrary normal-order reduction grammar R_n. Let us consider the counting sequence $(r_{n,k})_{k\in\mathbb{N}}$ where $r_{n,k}$ denotes the number of SK-combinators of size k reducing in n normal-order reduction steps. Suppose we associate with it a formal power series $R_n(z)$ defined as

$$R_n(z) = \sum_{k=0}^{\infty} r_{n,k}\, z^k\,. \tag{4.16}$$

In the following theorem we present a recursive method of computing the closed-form solution (i.e. finite representation using elementary functions) of $R_n(z)$ using the regular tree grammars $R_0, \ldots, R_n$ and the inductive use of the symbolic method (see Section 2.1).

Let us start with the following proposition.

Proposition 4.20 (Bendkowski, Grygiel and Zaionc [BGZ15])**.** The generating function $C(z)$ associated with the set of all SK-combinators and its corresponding dominating singularity ρ_C are given by

$$C(z) = \frac{1-\sqrt{1-8z}}{2z} \quad \text{and} \quad \rho_C = \frac{1}{8}\,. \tag{4.17}$$

Furthermore, the generating function $R_0(z)$ enumerating SK-combinators in normal form and its corresponding dominating singularity ρ_0 are given by

$$R_0(z) = \frac{1-2z-\sqrt{1-4z-4z^2}}{2z^2} \quad \text{and} \quad \rho_0 = \frac{1}{2}\left(\sqrt{2}-1\right) \approx 0.207107\,. \tag{4.18}$$

Theorem 4.21 (Generating function construction)**.** For each $n \geq 0$, the generating function $R_n(z)$ corresponding to the sequence $(r_{n,k})_{k\in\mathbb{N}}$ has a computable (in terms of elementary functions) closed-form solution.

Proof. Induction over n. Certainly, the base case $n = 0$ is clear due to (4.18).

Suppose that $n \geq 1$. Recall that in its construction, R_n might depend on previous reduction grammars $R_0, \ldots, R_{n-1}$, the set $\mathcal{C}$ of all SK-combinators and itself, via self-referencing productions. Due to Theorem 4.19, R_n is unambiguous and so we can express its generating function $R_n(z)$ as the unique solution of

$$R_n(z) = \sum_{\alpha\in R_n} z^{k(\alpha)} C(z)^{c(\alpha)} \prod_{i=0}^{n} R_i(z)^{r_i(\alpha)} \tag{4.19}$$

where $k(\alpha)$, $c(\alpha)$ and $r_i(\alpha)$ denote the number of applications, the number of non-terminal symbols C and the number of non-terminal symbols R_i in α, respectively.

Note that R_n has exactly four self-referencing productions, i.e. SR_n, KR_n, SR_0R_n and SR_nR_0. It means that by converting them into appropriate functional equations, we can further rewrite (4.19) as

$$R_n(z) = 2zR_n(z) + 2z^2R_0(z)R_n(z) + \sum_{\alpha\in\Phi(R_n)} z^{k(\alpha)} C(z)^{c(\alpha)} \prod_{i=0}^{n-1} R_i(z)^{r_i(\alpha)} \tag{4.20}$$

where $\Phi(R_n)$ denotes the set of productions $\alpha \in R_n$ which do not reference R_n. By the induction hypothesis, we can compute the closed-form solutions for $R_0(z), \ldots, R_{n-1}(z)$ turning (4.20) into a linear equation in $R_n(z)$. Simplifying (4.18) for $R_0(z)$ we derive the final closed-form solution

$$R_n(z) = \frac{1}{\sqrt{1-4z-4z^2}} \sum_{\alpha \in \Phi(R_n)} z^{k(\alpha)} C(z)^{c(\alpha)} \prod_{i=0}^{n-1} R_i(z)^{r_i(\alpha)} \,. \tag{4.21}$$

□

4.3.5 Other applications

In this section we highlight some interesting consequences of the existence of normal-order reduction grammars. In particular, we prove that terms reducing in n steps have necessarily bounded length. Moreover, we show that the problem of deciding whether a given term reduces in n steps can be done in memory independent of the size of the term.

Proposition 4.22. If $\alpha \in R_n$, then α has length at most $2n+2$.

Proof. Induction over n. The base case $n = 0$ is clear from the shape of R_0. Fix $n > 0$. Let us consider long productions in R_n. If β is a K-Expansion of some $X\alpha_1 \ldots \alpha_m \in R_{n-1}$, then

$$\beta = K(X\alpha_1 \ldots \alpha_k)\mathcal{C}\alpha_{k+1} \ldots \alpha_m \qquad \text{for} \quad 0 \leq k \leq m-1 \,. \tag{4.22}$$

In particular, the longest K-Expansion of $X\alpha_1 \ldots \alpha_m$ is in form of

$$\overline{\beta} = KX\alpha_1 \ldots \alpha_k \mathcal{C}\alpha_{k+1} \ldots \alpha_m \,. \tag{4.23}$$

Note that $\overline{\beta}$ is of length $m+2$ and so by the induction hypothesis at most $2n+2$.

Now, let us consider the case when β is a S-Expansion of some $X\alpha_1 \ldots \alpha_m \in R_{n-1}$. Then,

$$\beta = S(X\alpha_1 \ldots \alpha_k)\varphi_l\varphi_r\alpha_{k+3} \ldots \alpha_m \qquad \text{for} \quad 0 \leq k \leq m-2 \tag{4.24}$$

where in addition $(\varphi_l\, \varphi_r) \in \text{RewritingSet}(\alpha_{k+1}, \alpha_{k+2})$. Again, the longest S-Expansion of $X\alpha_1 \ldots \alpha_m$ is in form of

$$\overline{\beta} = SX\alpha_1 \ldots \alpha_k \varphi_l\varphi_r\alpha_{k+3} \ldots \alpha_m \,. \tag{4.25}$$

It follows that $\overline{\beta}$ is of length at most $m+1$ and so also at most $2n+1$. □

In other words, terms reducing in n steps cannot be too long as their length is tightly bounded by $2n+2$. Now, let us consider the following two problems:

Problem: FIXED-STEP-REDUCIBILITY
Input: A combinatory logic term $x \in L(\mathcal{C})$.
Output: YES if and only if x reduces in n steps.

Problem: REDUCIBILITY
Input: A combinatory logic term $x \in L(\mathcal{C})$ and a number $n \in \mathbb{N}$.
Output: YES if and only if x reduces in n steps.

Let us start with the FIXED-STEP-REDUCIBILITY problem. Since n in not a part of the input, we can compute R_n in constant time and memory. Using R_n we build a bottom-up tree automaton [Com+07] recognising $L(R_n)$ and use it to check whether $x \in L(R_n)$ in time $O(|x|)$, without using additional memory. On the other hand, the NAIVE algorithm, performing up to n normal-order reduction steps, requires $O(|x|)$ time and additional memory. At each step, the considered term doubles at most in size, as $Sxyz \to_w xz(yz)$. In order to find the next redex we spend up to linear time in the current size of x, therefore both size and time are bounded by

$$\begin{aligned} |x| + 2|x| + 4|x| + \cdots + 2^n|x| &= |x|\Big(1 + 2 + 4 + \cdots + 2^n\Big) \\ &= |x|\Big(2^{n+1} - 1\Big) = O(|x|)\,. \end{aligned} \tag{4.26}$$

As a natural extension of the above discussion, we get the following corollary.

Corollary 4.23. The REDUCIBILITY problem is decidable in space depending exclusively on n, independently of $|x|$.

4.3.6 Upper bound

In this section we focus on the upper bound on the number $|R_n|$ of productions in R_n. We show that there exists a primitive recursive function $f : \mathbb{N} \to \mathbb{N}$ such that $|R_n| \leq f(n)$.

Following the scheme of the soundness proofs in Section 4.3.1, we construct suitable upper bounds using the notions of tree potential and degree. In the end of this section, we show that these values are in fact bounded in each R_n thus giving the desired upper bound.

Lemma 4.24. Let α, β be two trees of degree at most n such that their total potential $\pi(\alpha) + \pi(\beta)$ is equal to p. Then, the number of distinct meshes in MESHSET(α, β) is bounded by $|R_n|^{e\,p!}$.

Proof. Induction over the total potential p. Consider the following primitive recursive function $f_n : \mathbb{N} \to \mathbb{N}$:

$$f_n(k) = \begin{cases} 1 & \text{if } \ k = 0\,, \\ (|R_n| \cdot f_n(k-1))^k & \text{otherwise.} \end{cases} \tag{4.27}$$

We claim that $|\text{MESHSET}(\alpha, \beta)| \leq f_n(p)$. Note that it suffices to consider such α, β that $|\text{MESHSET}(\alpha, \beta)| > 1$ since f_n is an increasing function attaining positive values for any given input. It follows that the base case $p = 0$ is clear, as if $\pi(\alpha) + \pi(\beta) = 0$, then MESHSET$(\alpha, \beta)$ is necessarily empty. Now, let us assume that $p > 0$. From the construction of the common mesh set M of α and β, we can distinguish two cases left to consider:

(i) Suppose that $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$. In order to maximise the size of M, we can furthermore assume that none of the pairs α_i, β_i are rewritable. And so, the total number of meshes in M is equal to the product of all meshes in corresponding mesh sets for α_i and β_i. The degree of α_i and β_i is still at most n, however $\pi(\alpha_i) + \pi(\beta_i) \leq p - 2$. Hence, using the induction hypothesis we get $|\textsc{MeshSet}(\alpha_i, \beta_i)| \leq f_n(p-2)$. Since both α, β are of length $m \leq p$ we can furthermore state that

$$\begin{aligned} |M| &\leq (f_n(p-2))^m \leq (f_n(p-2))^p \\ &\leq (f_n(p-1))^p \leq (|R_n| \cdot f_n(p-1))^p \\ &= f_n(p)\,; \end{aligned} \tag{4.28}$$

(ii) Let us assume w.l.o.g. that $\alpha = R_i$ and β is complex. In order to maximise the total number of meshes in M, we can moreover assume that all productions $\gamma \in R_i$ are similar to β and generate disjoint sets of meshes. We claim that $\textsc{MeshSet}(\gamma, \beta) \leq f_n(p-1)$. And so, if γ does not reference R_i, then our claim is trivially true. Suppose that γ is a self-referencing production. If $\gamma = XR_i$, then β is in form of $X\beta_1$. From the construction of M, we get that

$$|\textsc{MeshSet}(\gamma, \beta)| = |\textsc{MeshSet}(R_i, \beta_1)|\,. \tag{4.29}$$

As $\pi(R_i)+\pi(\beta_1) \leq p-1$, we can apply the induction hypothesis to $\textsc{MeshSet}(R_i, \beta_1)$ and immediately obtain $|\textsc{MeshSet}(\gamma, \beta)| \leq f_n(p-1)$. Now, suppose w.l.o.g. that $\gamma = SR_iR_0$ and hence $\beta = S\beta_1\beta_2$. Again, from the construction of M we know that

$$|\textsc{MeshSet}(\gamma, \beta)| = |\textsc{MeshSet}(R_i, \beta_1)| \cdot |\textsc{MeshSet}(R_0, \beta_2)|\,. \tag{4.30}$$

Due to the fact that both $\pi(R_i) + \pi(\beta_1) \leq p - 2$ and $\pi(R_0) + \pi(\beta_2) \leq p - 2$, we can use the induction hypothesis and immediately get that

$$\begin{aligned} |\textsc{MeshSet}(\gamma, \beta)| &= |\textsc{MeshSet}(R_i, \beta_1)| \cdot |\textsc{MeshSet}(R_0, \beta_2)| \\ &\leq f_n(p-2)\, f_n(p-2). \end{aligned} \tag{4.31}$$

Note that $(f_n(p-2))^2 \leq f_n(p-1)$ for $p \geq 2$ and, in consequence, $|\textsc{MeshSet}(\gamma, \beta)| \leq f_n(p-1)$. Indeed, if $p = 2$, then $(f_n(p-2))^2 = 1 \leq f_n(1) = |R_n|$. Otherwise if $p > 2$, then

$$\begin{aligned} f_n(p-1) &= (|R_n| \cdot f_n(p-2))^{p-1} \\ &= \left(|R_n|^{p-1}(f_n(p-3))^{p-2}\right)^{p-1} \\ &\geq \left(|R_n|^{p-2}(f_n(p-3))^{p-2}\right)^{p-1} \\ &= (|R_n| \cdot f_n(p-3))^{(p-1)(p-2)}\,. \end{aligned} \tag{4.32}$$

As $2(p-2) \leq (p-1)(p-2)$ for $p > 2$, we finally obtain

$$\begin{aligned} (|R_n| \cdot f_n(p-3))^{(p-1)(p-2)} &\geq (|R_n| \cdot f_n(p-3))^{2(p-2)} \\ &= (f_n(p-2))^2\,. \end{aligned} \tag{4.33}$$

We know therefore that $\textsc{MeshSet}(\gamma, \beta) \leq f_n(p-1)$ for each $\gamma \in R_i$. Finally, using the fact that $|R_i| \leq |R_n|$, we get

$$\begin{aligned} |M| &\leq |R_n| \cdot f_n(p-1) \\ &\leq (|R_n| \cdot f_n(p-1))^p \\ &= f_n(p)\,. \end{aligned} \tag{4.34}$$

And so, we know that $|\textsc{MeshSet}(\alpha, \beta)| \leq f_n(p)$. Solving the recurrence for $f_n(p)$, using e.g. Mathematica [Wol15], we obtain the following closed-form expression:

$$f_n(p) = |R_n|^{e\,p\,\Gamma(p,1)} \tag{4.35}$$

where

$$\Gamma(s, x) = (s-1)!\, e^{-x} \sum_{k=0}^{s-1} \frac{x^k}{k!} \tag{4.36}$$

is the upper incomplete gamma function (see e.g. [AS72]). Simplifying the above expression in the case $x = 1$ and using the fact that $\sum_{k=0}^{s-1} \frac{1}{k!} \leq e$ for arbitrary s, we finally obtain the anticipated upper bound

$$f_n(p) \leq |R_n|^{e\,p!}\,. \tag{4.37}$$

□

Lemma 4.25. Let α, β be two trees of degree at most n such that their total potential $\pi(\alpha) + \pi(\beta)$ is equal to p. Then, the number of distinct trees in $\textsc{RewritingSet}(\alpha, \beta)$ is bounded by $|R_n|^{1+e\,p!}$.

Proof. If $|\textsc{RewritingSet}(\alpha, \beta)| \leq 1$, then our claim is trivially true. Let us focus therefore on the remaining cases when either $\beta = X\beta_1 \ldots \beta_m$ and both β_m and α are non-rewritable, or $\beta = R_i$.

First, consider the former case. Note that the resulting rewriting set is of equal size as $\textsc{MeshSet}(\alpha, \beta_m)$. Since $\pi(\alpha) + \pi(\beta_m) \leq p - 1$, we can use Lemma 4.24 to deduce that

$$|\textsc{RewritingSet}(\alpha, \beta)| = |\textsc{MeshSet}(\alpha, \beta_m)| \leq |R_n|^{e\,(p-1)!} < |R_n|^{1+e\,p!}\,. \tag{4.38}$$

Now, let us consider the latter case. In order to maximise the resulting rewriting set we assume that each production $\gamma \in R_i$ generates a disjoint set of trees. We claim that each production γ contributes at most $|R_n|^{e\,p!}$ new trees to the resulting rewriting set and therefore $|\textsc{RewritingSet}(\alpha, \beta)| \leq |R_n|^{1+e\,p!}$, as there are at most $|R_n|$ productions in R_i. Indeed, consider an arbitrary $\gamma \in R_i$. Evidently, if $|\textsc{RewritingSet}(\alpha, \gamma)| \leq 1$, then our claim is true. Hence, let us assume that $|\textsc{RewritingSet}(\alpha, \gamma)| > 1$. It follows that γ is complex. Let us rewrite it as $X\gamma_1 \ldots \gamma_m$. Note that as in the previous case, the resulting rewriting set is of equal size as $\textsc{MeshSet}(\alpha, \gamma_m)$. Since $\pi(\alpha) + \pi(\gamma_m) \leq p - 1$ we use Lemma 4.24 and get

$$|\textsc{RewritingSet}(\alpha, \gamma)| = |\textsc{MeshSet}(\alpha, \gamma_m)| \leq |R_n|^{e\,(p-1)!} < |R_n|^{e\,p!}\,. \tag{4.39}$$

□

Lemma 4.26. Let α, β be two trees of total potential $\pi(\alpha) + \pi(\beta)$ equal to p. Then, each mesh in $\textsc{MeshSet}(\alpha, \beta)$ has potential bounded by $p!(1 + e)$.

Proof. Induction over total potential p. Again, it suffices to consider such α, β that $\textsc{MeshSet}(\alpha, \beta)$ is not empty. Immediately, the base case $p = 0$ is clear. Let us assume that $p > 0$. Consider the following primitive recursive function $f : \mathbb{N} \to \mathbb{N}$:

$$f(k) = \begin{cases} 1 & \text{if } k = 0\,, \\ k \cdot (f(k-1) + 1) & \text{otherwise.} \end{cases} \tag{4.40}$$

Let $\gamma \in \textsc{MeshSet}(\alpha, \beta)$. We claim that $\pi(\gamma) \leq f(p)$. Note that f is an increasing function attaining positive values for any input. We have two cases to consider:

(i) Suppose that $\alpha = X\alpha_1 \ldots \alpha_m$ and $\beta = X\beta_1 \ldots \beta_m$. Note that $\pi(\alpha_i) + \pi(\beta_i) \leq p - 2$ for each pair of corresponding arguments α_i, β_i. Using the induction hypothesis to pairs α_i, β_i and the fact that $\gamma \in \textsc{MeshSet}(\alpha, \beta)$ is similar to both α and β, we bound γ's potential by

$$\pi(\gamma) \leq m \cdot f(p-2) + m \leq p \cdot (f(p-2) + 1) \leq f(p)\,; \tag{4.41}$$

(ii) Assume w.l.o.g. that $\alpha = R_i$ and β is complex. It follows that $\gamma \in \textsc{MeshSet}(\delta, \beta)$ for some $\delta \in R_i$. If δ does not reference R_i, then clearly $\pi(\delta) \leq \pi(R_i) - 1$ and therefore $\pi(\gamma) \leq f(p-1)$.

Now, suppose that δ is a self-referencing production of R_i. If $\delta = XR_i$, then β is in form of $X\beta_1$ and similarly $\gamma = X\gamma_1$. It follows that $\pi(\delta) = \pi(R_i) + 1$ and therefore $\pi(\delta) + \pi(\beta) = p + 1$. Note however that $\pi(\gamma_1) \leq f(p-1)$ as $\pi(R_i) + \pi(\beta_1) \leq p - 1$. Due to that, $\pi(\gamma) = 1 + f(p-1) \leq f(p)$.

Let us assume w.l.o.g. that $\delta = SR_iR_0$. Immediately, β is in form of $S\beta_1\beta_2$ whereas $\gamma = S\gamma_1\gamma_2$. Moreover, $\pi(\delta) = \pi(R_i)+3$. Note however that both $\pi(R_i)+\pi(\beta_1) \leq p-2$ and $\pi(R_0) + \pi(\beta_2) \leq p - 2$. We can therefore use the induction hypothesis and conclude that

$$\pi(\gamma) = 2 + \pi(\gamma_1) + \pi(\gamma_2) \leq 2 + 2 \cdot f(p-2)\,. \tag{4.42}$$

Since $\pi(\delta) \geq 4$, we know that $p \geq 3$ and so we can further bound $\pi(\gamma)$ by

$$\begin{aligned} \pi(\gamma) &= 2\,(1 + f(p-2)) \\ &\leq (p-1)\,(1 + f(p-2)) \\ &= f(p-1) \leq f(p)\,. \end{aligned} \tag{4.43}$$

Finally, we know that $\pi(\gamma) \leq f(p)$. What remains is to solve the recursion, using e.g. Mathematica [Wol15], for f and give its closed-form solution. It follows that

$$\begin{aligned} f(p) &= \Gamma(1+p) + e\,p\,\Gamma(p,1) \\ &\leq p! + e\,p! \\ &= p!(1+e) \end{aligned} \tag{4.44}$$

where

$$\Gamma(n) = (n-1)! \tag{4.45}$$

□

Lemma 4.27. Let α, β be two trees of potential $\pi(\alpha) + \pi(\beta) = p$. Then, each tree in $\textsc{RewritingSet}(\alpha, \beta)$ has potential bounded by $p!(1+e) + p$.

Proof. Let γ be an arbitrary tree in $\textsc{RewritingSet}(\alpha, \beta)$. Based on the structure of β we have several cases to consider. If $\beta = \mathcal{C}$, then $\gamma = \mathcal{C}\alpha$ and so $\pi(\gamma) = \pi(\alpha) + 1 = p + 1$. Note that $1 < p!(1+e)$ for any p and thus our bound holds.

If $\beta = X\beta_1 \dots \beta_m$, then $\pi(\alpha) + \pi(\beta_m) \leq p - 1$. In both cases when $\alpha \bowtie \beta_m$ the resulting tree has potential bounded by p and so also by $p!(1 + e) + p$. Let us assume that $\alpha \parallel \beta_m$. We can therefore rewrite γ as $X\gamma_1 \dots \gamma_m$. Using Lemma 4.26, we know that $\pi(\gamma_m) \leq (p - 1)!(1 + e)$. Moreover, both α and β are similar to γ. Let us rewrite them as $X\alpha_1 \dots \alpha_m$ and $X\beta_1, \dots, \beta_m$, respectively. Note that for each $i < m$, γ_i is equal to α_i or β_i. It follows that we can bound the potential of $X\gamma_1 \dots \gamma_{m-1}$ by $p - 1$ and hence γ's potential by $(p - 1)!(1 + e) + p$.

Now, if $\beta = R_i$, then $\gamma \in \textsc{RewritingSet}(\alpha, \delta)$ for some $\delta \in R_i$. If δ does not reference R_i, we know that $\pi(\delta) \leq \pi(R_i) - 1 \leq p - 1$. Moreover, δ is complex, as otherwise $\textsc{RewritingSet}(\alpha, \delta) = \varnothing$. Using our previous argumentation, we can therefore conclude that $\pi(\gamma) \leq (p - 1)!(1 + e) + p$. Suppose that δ is a self-referencing production of R_i. If $\delta = XR_i$, then α is in form of $X\alpha_1$ and $\gamma = X\gamma_1$. Immediately, $\pi(\alpha) + \pi(\delta) = p + 1$. If $R_i \bowtie \alpha_1$, then γ has potential bounded by p. Therefore, let us assume that $R_i \parallel \alpha_1$. Since $\pi(R_i) + \pi(\alpha_1) = p - 1$, we know from Lemma 4.26 that $\pi(\gamma_1) \leq (p - 1)!(1 + e)$. It follows immediately that $\pi(\gamma) \leq (p - 1)!(1 + e) + 1 \leq p!(1 + e) + p$.

Finally, suppose that $\delta = S\delta_1\delta_2$ and so $\alpha = S\alpha_1\alpha_2$. Immediately, $\gamma = S\gamma_1\gamma_2$. Again, if $\delta_2 \bowtie \alpha_2$, we can bound γ's potential by p. Hence, let us assume that $\delta_2 \parallel \alpha_2$. Clearly, $\pi(\alpha) + \pi(\delta) = p + 3$. Note however that $\pi(\alpha_1) + \pi(\delta_1) \leq p - 2$ and $\pi(\alpha_2) + \pi(\delta_2) \leq p - 2$, as both δ_1 and δ_2 are non-terminal reduction grammar symbols of positive potential. Using Lemma 4.26 to $\textsc{MeshSet}(\alpha_2, \delta_2)$ we conclude that $\pi(\gamma_2) \leq (p - 2)!(1 + e)$. It follows that $\pi(\gamma) \leq (p - 2)!(1 + e) + p \leq p!(1 + e) + p$. □

Lemma 4.28 (Upper bound for $\pi(R_n)$)**.** There exists a primitive recursive function $\psi : \mathbb{N} \to \mathbb{N}$ such that $\pi(R_n) \leq \psi(n)$.

Proof. Consider the following function $\psi : \mathbb{N} \to \mathbb{N}$:

$$\psi(k) = \begin{cases} 1 & \text{if } k = 0\,, \\ 4\,(\psi(k-1) + 2)! + 2\psi(k-1) + 5 & \text{otherwise.} \end{cases} \tag{4.46}$$

Evidently, ψ is an increasing primitive recursive function. We show that $\psi(n)$ bounds the potential of R_n using induction over n. Since $\pi(R_0) = \psi(0) = 1$, the base case is clear. Let $n > 0$. In order to prove our claim, we have to check that $\pi(\alpha) \leq \psi(n) - 1$ for all productions $\alpha \in R_n$ which do not reference R_n:

(i) Suppose that $\alpha = SR_{n-i}R_i$. The potential of α is equal to $2 + \pi(R_{n-i}) + \pi(R_i)$. Using the induction hypothesis, we know moreover that

$$\begin{aligned} \pi(\alpha) &\leq 2 + \psi(n-i) + \psi(i) \\ &\leq 2 + 2\psi(n-1) \\ &\leq \psi(n) - 1\,; \end{aligned} \tag{4.47}$$

(ii) Let $\alpha = KR_{n-1}\mathcal{C}$. Due to the fact that $\pi(\alpha) = 2 + \pi(R_{n-1})$, we use the induction hypothesis and immediately obtain

$$\pi(\alpha) \leq 2 + \psi(n-1) \leq \psi(n) - 1\,; \tag{4.48}$$

(iii) Suppose that $\alpha \in$ K-EXPANSIONS(β) for some $\beta \in R_{n-1}$. Note that $\pi(\beta) \leq \psi(n-1)+3$ as the productions of greatest potential in R_{n-1} are exactly $SR_{n-1}R_0$ and SR_0R_{n-1}. Since $\pi(\alpha) = 2 + \pi(\beta)$, we get

$$\pi(\alpha) \leq 5 + \psi(n-1) \leq \psi(n) - 1\,; \tag{4.49}$$

(iv) Finally, let $\alpha \in$ S-EXPANSIONS(β) for some $\beta \in R_{n-1}$. Again, $\pi(\beta) \leq \pi(R_{n-1}) + 3$ and hence from the induction hypothesis $\pi(\beta) \leq \psi(n-1) + 3$. Let us rewrite α as $S(X\beta_1 \ldots \beta_k)\varphi_l\varphi_r\beta_{k+3}\ldots\beta_m$ where $\beta = X\beta_1 \ldots \beta_m$. Note that $\pi(\alpha) \leq \pi(\beta) + \pi(\varphi_l) + \pi(\varphi_r) + 1$. Moreover, as $\pi(\varphi_l\varphi_r) = 1 + \pi(\varphi_l) + \pi(\varphi_r)$, we get $\pi(\alpha) \leq \pi(\beta) + \pi(\varphi_l\varphi_r)$. Since $\pi(\beta_{k+1}\beta_{k+2}) \leq \pi(\beta) - 1$ and thus, $\pi(\beta_{k+1}\beta_{k+2}) \leq \psi(n-1)+2$, we can use Lemma 4.27 to obtain

$$\pi(\varphi_l\varphi_r) \leq (\psi(n-1)+2)!(1+e) + \psi(n-1) + 2\,. \tag{4.50}$$

It follows therefore that

$$\begin{aligned} \pi(\alpha) &\leq \pi(\beta) + \pi(\varphi_l\varphi_r) \\ &\leq (\psi(n-1)+2)!(1+e) + 2\psi(n-1) + 5 \\ &\leq \psi(n) - 1 \end{aligned} \tag{4.51}$$

where the last inequality follows from the fact that

$$(3-e)\,(\psi(n-1)+2)! \geq \frac{1}{5}(\psi(n-1)+2)! \geq \frac{6}{5} > 0\,. \tag{4.52}$$

□

Theorem 4.29 (Upper bound for $|R_n|$)**.** There exists a primitive recursive function $\chi : \mathbb{N} \to \mathbb{N}$ such that the number $|R_n|$ of productions in R_n is bounded by $\chi(n)$.

Proof. Consider R_n for some $n > 0$. Recall that R_n consists of:

(i) two productions SR_n and KR_n;

(ii) $n+1$ short S-productions in form of $SR_{n-i}R_i$;

(iii) an additional K-production $KR_{n-1}\mathcal{C}$;

(iv) K-EXPANSIONS(α) for each $\alpha \in R_{n-1}$;

(v) S-EXPANSIONS(α) for each $\alpha \in R_{n-1}$.

It suffices therefore to bound the number of K- and S-EXPANSIONS, as the number of other productions in R_n is clear. Let us start with K-EXPANSIONS. Suppose that α is of length m. Then, we have $|$K-EXPANSIONS$(\alpha)| = m$. Using Proposition 4.22, we know that that each production $\alpha \in R_{n-1}$ is of length at most $2n$. It follows that there are at most $2n \cdot |R_{n-1}|$ K-EXPANSIONS in R_n. Now, let us consider S-EXPANSIONS. In order to bound the number of S-EXPANSIONS in R_n, we assume that each production $\alpha \in R_{n-1}$ is of length $2n$ and moreover each REWRITINGSET of appropriate portions of

α generates a worst-case set of trees. And so, assuming that α is of length $2n$ we can rewrite it as $X\alpha_1 \dots \alpha_{2n}$. Let ψ denote the upper bound function on the potential of R_{n-1} from Lemma 4.28. Evidently, $\pi(\alpha) \leq \psi(n-1) + 3$. Now, using Lemma 4.25 we know that each RewritingSet(α_i, α_{i+1}) contributes at most

$$|R_{n-1}|^{1+e\left(\psi(n-1)+3\right)!} \tag{4.53}$$

new S-Expansions. As there are at most $2n-1$ pairs of indices $(i, i+1)$ yielding RewritingSets, we get that the number of S-Expansions in R_n is bounded by

$$(2n-1) \cdot |R_{n-1}| \cdot |R_{n-1}|^{1+e\left(\psi(n-1)+3\right)!} \leq (2n-1) \cdot |R_{n-1}|^{2+3\left(\psi(n-1)+3\right)!}. \tag{4.54}$$

Finally, since $|R_0| = 5$, we combine the above observations and get the following primitive recursive upper bound on $|R_n|$.

$$\chi(k) = \begin{cases} 5 & \text{if } k = 0\,, \\ 4 + k + 2k \cdot \chi(k-1) & \\ +\ (2k-1) \cdot \chi(k-1)^{2+3\left(\psi(k-1)+3\right)!} & \text{otherwise.} \end{cases} \tag{4.55}$$

□

We emphasise the fact that although the size of R_n is bounded by a primitive recursive function of n, it seems to be enormously overestimated. Our computer implementation of the Reduction Grammar algorithm [Ben16d] suggests that the initial numbers in the sequence $(|R_n|)_{n\in\mathbb{N}}$ are in fact:

$$5, 12, 75, 625, 5673, 53164, 508199, \dots.$$

The upper bound $\chi(1)$ on the size of R_1 is already of order $6 \cdot 10^{84549}$ whereas the actual size of R_1 is equal to 12. Naturally, we conjecture that $(|R_n|)_{n\in\mathbb{N}}$ grows much slower than $(\chi(n))_{n\in\mathbb{N}}$, although the problem of giving better approximations on the size of R_n for large n is still open.

Chapter 5

Quantitative aspects of combinatory logic

In the current chapter we investigate the quantitative properties of combinatory logic. We start with several basis-independent results. Subsequently, we focus on the classic set of SK-combinators studying the quantitative aspects of normalising combinators.

5.1 Basis independent results

We start with certain general results about classes of plane trees with labelled leaves, deriving the universal, basis-independent combinatory logic results as immediate corollaries.

Definition 5.1 (L-trees)**.** Suppose that L is a finite set of d distinct labels. Then, the set of L-trees consists of plane binary trees where each leave has a corresponding label in the set L. We use T_L to denote the set of L-trees.

Let us notice that the asymptotic growth rate of L-trees with n inner nodes directly depends on the asymptotic approximation of the Catalan numbers Cat_n counting the number of plane binary trees with n inner nodes (see, e.g. [FS09]). It is well known that

$$\mathsf{Cat}_n = \frac{1}{n+1}\binom{2n}{n} \qquad \text{and} \qquad \mathsf{Cat}_n \sim \frac{4^n}{\sqrt{\pi}n^{3/2}}\,. \tag{5.1}$$

Certainly, since each plane binary tree with n inner nodes has exactly $n+1$ leaves, the number $T_{L,n}$ of L-terms of size n is given as

$$T_{L,n} = d^{n+1}\cdot \mathsf{Cat}_n = \frac{d^{n+1}}{n+1}\binom{2n}{n}\,. \tag{5.2}$$

Suppose that $t \in T_L$. Let $\overline{T_L}(z)$ denote the generating function associated with the class of L-trees containing t as a subtree. In the following series of propositions, we derive the closed-form solution for $\overline{T_L}(z)$ and check the conditions of the algebraic singularity analysis used subsequently to show that in fact both $[z^n]\overline{T_L}(z)$ and $[z^n]T_L(z)$ are asymptotically equivalent, independently of L.

Proposition 5.2. Let T_L be the set of L-trees where $|L| = d$. Then, its counting sequence $(T_{L,n})_{n\in\mathbb{N}}$ has a corresponding generating function $T_L(z)$ given by

$$T_L(z) = \frac{1-\sqrt{1-4dz}}{2z}\,. \tag{5.3}$$

Proof. Note that T_L can be defined as $T_L = L + {T_L}^2$, which by virtue of the symbolic method translates into the following functional equation defining $T_L(z)$:

$$T_L(z) = d + zT_L(z)^2 \tag{5.4}$$

with the following two possible solutions:

$$T_L(z) = \frac{1 \pm \sqrt{1-4dz}}{2z}\,. \tag{5.5}$$

An easy application of Riemann's removable singularities theorem (see Theorem 2.18) yields that (5.3) is the only solution analytically continuable at the origin, finishing the proof. □

Proposition 5.3. Let L be a set of d distinct labels. Assume that $t \in T_L$ is an L-tree of size $p \geq 1$. Then the set of L-trees containing t as a subtree, denoted as $\overline{T_L}$, has the following generating function:

$$\overline{T_L}(z) = \frac{-\sqrt{1-4dz} + \sqrt{1-4dz+4z^{p+1}}}{2z}\,. \tag{5.6}$$

Proof. Let us start with noticing that any L-tree containing t as a subtree is either equal to t, or one of its left or right subtrees contains t whereas the other one is a tree in T_L. However, since trees in T_L may contain t as a subtree, we have to subtract trees containing t in both branches to avoid double counting. Such a specification yields the following functional equation defining $\overline{T_L}(z)$:

$$\overline{T_L}(z) = z^p + 2zT_L(z)\overline{T_L}(z) - z\overline{T_L}(z)^2\,. \tag{5.7}$$

Solving (5.7) for $\overline{T_L}(z)$ we obtain two possible solutions:

$$\frac{-\sqrt{1-4dz} \pm \sqrt{1-4dz+4z^{p+1}}}{2z}\,. \tag{5.8}$$

Since $p \geq 1$, there are no L-trees of size 0 containing t as a subterm. In consequence $\lim_{z\to 0} \overline{T_L}(z) = 0$ and so we obtain the claimed solution. □

Proposition 5.4. Let $\rho = 1/4d$. Then, ρ is the only singularity on both the circles of convergence of $T_L(z)$ and $\overline{T_L}(z)$.

Proof. From (5.3) it is clear that ρ is the only singularity of $T_L(z)$ on the circle $|z| = \rho$. Now, let us focus on $\overline{T_L}(z)$. Due to the fact that $\overline{T_L}(z) \preceq T_L(z)$, the exponential growth formula guarantees that $\overline{T_L}(z)$ has no singularities in the disk $|z| < \rho$. Moreover, since $\sqrt{1-4dz}$ is a part of the closed-form expression (5.6) of $\overline{T_L}(z)$, it suffices to check that $F(z) = 1 - 4dz + 4z^{p+1}$ has no complex roots of modulus ρ. Let us rewrite (5.6) as

$$\begin{aligned}\overline{T_L}(z) &= \frac{1-\sqrt{1-4dz} - \left(1-\sqrt{1-4dz+4z^{p+1}}\right)}{2z} \\ &= T_L(z) - \frac{1-\sqrt{1-4dz+4z^{p+1}}}{2z}\,. \end{aligned} \tag{5.9}$$

Both $T_L(z)$ and $\overline{T_L}(z)$ are generating functions corresponding to sequences of non-negative integers; hence the coefficients in the Maclaurin series of the latter expression are non-negative integers as well. To finish the proof, we note that

$$F(\rho) = 4^{-p}\left(\frac{1}{d}\right)^{p+1} > 0 \tag{5.10}$$

and so due to Pringsheim's theorem, $F(z)$ cannot have complex roots of modulus ρ. □

Proposition 5.5. Both $[z^n]T_L(z)$ and $[z^n]\overline{T_L}(z)$ asymptotically equivalent. Specifically:

$$[z^n]T_L(z) \sim [z^n]\overline{T_L}(z) \sim \frac{4^n d^{n+1}}{\sqrt{\pi} n^{3/2}} \,. \tag{5.11}$$

Proof. By (5.3) and (5.6) we can rewrite the closed-form solutions of $T_L(z)$ and $\overline{T_L}(z)$ as

$$T_L(z) = \sqrt{1-4dz}\left(-\frac{1}{2z}\right) + \frac{1}{2z} \quad \text{and} \tag{5.12}$$

$$\overline{T_L}(z) = \sqrt{1-4dz}\left(-\frac{1}{2z}\right) + \frac{\sqrt{1-4dz+4z^{p+1}}}{2z} \,. \tag{5.13}$$

Due to Proposition 5.4 both have a unique dominating singularity hence are amenable to the algebraic singularity analysis. □

Consequently, we are in the position to prove that L-trees admit the fixed subterm property, i.e. asymptotically almost all L-trees contain an arbitrary fixed L-tree as a subtree.

Proposition 5.6 (Fixed subterm property)**.** Let $t \in T_L$. Then asymptotically almost all L-trees contain t as a subtree.

Proof. In the case when $|t| \geq 1$ our claim follows directly from the asymptotic approximation of $\overline{T_L}(z)$. Now, suppose that $|t| = 0$ (i.e. t is a single leave). We can safely assume that $|L| > 1$ as otherwise our claim is trivial. Note that $[z^n]T_{L\setminus\{t\}}(z) = \Theta(4^n(d-1)^{n+1})$ whereas $[z^n]T_L(z) = \Theta(4^n d^{n+1})$; hence, the set of L-trees avoiding t is asymptotically negligible in the set of all L-trees, which finishes the proof. □

Furthermore, as in the case of λ-calculus, we obtain the following corollary.

Corollary 5.7. Let $\mathfrak{A}$ be a non-empty set of L-trees closed under taking supertrees. In other words, if $N \in \mathfrak{A}$ and N is a subtree of M, then $M \in \mathfrak{A}$. Then, asymptotically almost all L-trees are in the set $\mathfrak{A}$.

The above general observation asserts that 'local' properties of L-trees propagating to supertrees span asymptotically almost the whole set of L-trees. In light of the natural correspondence between $\mathcal{B}$-combinators and $\mathcal{B}$-trees, we can state that each 'local' property of $\mathcal{B}$-combinators closed under taking superterms is typical, i.e. has asymptotic probability one in the set of all $\mathcal{B}$-combinators. In particular, we obtain the following result.

Corollary 5.8. For each universal basis $\mathcal{B}$, asymptotically almost every $\mathcal{B}$-combinator is neither in normal form nor strongly normalising. Moreover, for each sound universal basis $\mathcal{B}$ (see Definition 2.58), asymptotically almost no $\mathcal{B}$-combinator is typeable.

Theorem 5.9. Let $\mathcal{B}$ be a universal basis of primitive combinators. Then

$$0 < \mu^-\left(\frac{\mathcal{WN}_\mathcal{B}}{\mathcal{C}_\mathcal{B}}\right) \quad \text{and} \quad \mu^+\left(\frac{\mathcal{WN}_\mathcal{B}}{\mathcal{C}_\mathcal{B}}\right) < 1 \,. \tag{5.14}$$

Proof. Let $\overline{S}$ and $\overline{K}$ be two $\mathcal{B}$-combinators implementing S and K, respectively.

We start with the lower limit. Let us consider the set $\mathcal{G}$ of combinators in form of $\overline{K}XM$ where X is a primitive combinator and $M \in \mathcal{C}_\mathcal{B}$. Certainly, $\mathcal{G}$ consists entirely of normalising combinators; hence $\mathcal{G} \subseteq \mathcal{WN}_\mathcal{B}$. Let us fix $p := |\overline{K}|$. Then

$$\mu\left(\frac{\mathcal{G}}{\mathcal{C}_\mathcal{B}}\right) = \lim_{n\to\infty} \frac{|\mathcal{G}_n|}{|\mathcal{C}_{\mathcal{B},n}|} = \lim_{n\to\infty} \frac{d \cdot |\mathcal{C}_{\mathcal{B},n-p-2}|}{|\mathcal{C}_{\mathcal{B},n}|} \tag{5.15}$$

$$= \lim_{n\to\infty} \frac{d^{n-p} \cdot \mathsf{Cat}_{n-p-2}}{d^{n+1} \cdot \mathsf{Cat}_n} = \frac{1}{d^{p+1} \cdot 4^{p+2}} > 0\,. \tag{5.16}$$

And so

$$\mu\left(\frac{\mathcal{G}}{\mathcal{C}_\mathcal{B}}\right) \leq \mu^-\left(\frac{\mathcal{WN}_\mathcal{B}}{\mathcal{C}_\mathcal{B}}\right) \tag{5.17}$$

hence the lower bound indeed holds.

Now, let us focus on the upper limit. Let $\omega = \overline{SII}$. Note that $SIIx \to_w^+ xx$ and so $SII(SII)$ has no normal form. In consequence, nor does $\Omega = \omega\omega$. Consider the map $\Phi\colon \mathcal{B} \to \mathcal{B}$ which for a given $\mathcal{B}$-combinator substitutes Ω for its leftmost primitive combinator X. Let $\mathcal{U}$ be the image of $\mathcal{C}_\mathcal{B}$ through Φ. Since Ω has no normal form, by virtue of the standardisation theorem $\mathcal{U}$ consists entirely of non-normalising combinators. In other words, we have $\mathcal{WN}_\mathcal{B} \subseteq \mathcal{C}_\mathcal{B} \setminus \mathcal{U}$. Let M be an arbitrary combinator in $\mathcal{U}$. Note that since there are d primitive combinators in $\mathcal{B}$, the map Φ sends exactly d distinct combinators to M. Fix $p := |\Omega|$. Then

$$\mu\left(\frac{\mathcal{U}}{\mathcal{C}_\mathcal{B}}\right) = \lim_{n\to\infty} \frac{|\mathcal{U}_n|}{|\mathcal{C}_{\mathcal{B},n}|} = \lim_{n\to\infty} \frac{d \cdot |\mathcal{C}_{\mathcal{B},n-p}|}{|\mathcal{C}_{\mathcal{B},n}|} \tag{5.18}$$

$$= \lim_{n\to\infty} \frac{d^{n-p+2} \cdot \mathsf{Cat}_{n-p}}{d^{n+1} \cdot \mathsf{Cat}_n} = \frac{d}{(4d)^p} > 0\,. \tag{5.19}$$

Since $\mathcal{WN}_\mathcal{B} \subseteq \mathcal{C}_\mathcal{B} \setminus \mathcal{U}$, we finally get

$$\mu^+\left(\frac{\mathcal{WN}_\mathcal{B}}{\mathcal{C}_\mathcal{B}}\right) \leq 1 - \mu\left(\frac{\mathcal{U}}{\mathcal{C}_\mathcal{B}}\right) \tag{5.20}$$

and so the upper bound holds as well. □

Using the fact that asymptotically no $\mathcal{B}$-combinator is strongly normalising (see Corollary 5.8), we obtain the following result.

Corollary 5.10. For each universal basis $\mathcal{B}$ of primitive combinators, asymptotically almost every weakly normalising $\mathcal{B}$-combinator is at the same time not strongly normalising.

5.2 Normalising SK-combinators

In this section we address the problem of estimating the asymptotic density of normalising SK-combinators in the set of all combinators. Specifically, we prove that for each positive integer n, the set of all SK-combinators reducing in n normal-order reduction steps has positive asymptotic density in the set of all SK-combinators.

Let us start with a few technical lemmas and propositions regarding the previously introduced generating functions $C(z)$ and $R_0(z)$ (see Proposition 4.20).

Lemma 5.11. Let $n \geq 1$. Then, $C(z)^n = \sqrt{1-8z}\,P(z)+Q(z)$ for some rational functions $P(z)$ and $Q(z)$ analytic in $\mathbb{C}\setminus\{0\}$. Moreover, $C(z)^n$ has a single removable singularity at $z=0$ in the disk $|z| < \rho_C$.

Proof. From equation (4.17), $C(z)$ can be rewritten as

$$C(z) = \sqrt{1-8z}\,P(z) + Q(z) \quad \text{where} \quad P(z) = -\frac{1}{2z} \quad \text{and} \quad Q(z) = \frac{1}{2z}\,. \tag{5.21}$$

Both $P(z)$ and $Q(z)$ are rational and hence also analytic in the complex plane except the origin. From Lemma 2.20, $C(z)^n$ can be expressed as

$$C(z)^n = \sqrt{1-8z}\,\overline{P}(z) + \overline{Q}(z) \tag{5.22}$$

for some $\overline{P}(z)$ and $\overline{Q}(z)$ analytic in $\mathbb{C}\setminus\{0\}$. Furthermore, following (2.19) and the closure properties of rational functions, it is clear that $\overline{P}(z)$ and $\overline{Q}(z)$ are also rational. Riemann's removable singularities theorem (see Theorem 2.18) guarantees that $C(z)$ has an analytic continuation including $z=0$ and, by virtue of Lemma 2.19, so does $C(z)^n$, finishing the proof. □

Lemma 5.12. Let $n \geq 1$. Then $R_0(z)^n = \sqrt{1-4z-4z^2}\,P(z) + Q(z)$ for some rational functions $P(z)$ and $Q(z)$ analytic in $\mathbb{C}\setminus\{0\}$. Moreover, $R_0(z)^n$ has a single removable singularity at $z=0$ in the disk $|z| < \rho_0$.

Proof. From the shape of equation (4.18) we have

$$R_0(z) = \sqrt{1-4z-4z^2}\,P(z) + Q(z) \tag{5.23}$$

where

$$P(z) = -\frac{1}{2z^2} \quad \text{and} \quad Q(z) = \frac{1-2z}{2z^2}\,. \tag{5.24}$$

Certainly, both $P(z)$ and $Q(z)$ are rational and analytic in $\mathbb{C}\setminus\{0\}$. The proof follows now easily from the same arguments as in Lemma 5.11. □

Through the remainder of this section, we exploit the structure of the normal-order reduction grammars (see Chapter 4), showing the following main result.

Theorem 5.13. Let $k \geq 1$. Then the asymptotic growth rate of $[z^n]R_k(z)$ is given by

$$[z^n]R_k(z) \sim 8^n \frac{\overline{C}_k n^{-3/2}}{\Gamma(-\frac{1}{2})} \tag{5.25}$$

where $\overline{C}_k$ is a constant depending entirely on k.

Recall that due to Theorem 4.21, the generating function $R_n(z)$ exhibits the following implicit form:

$$R_n(z) = \frac{1}{\sqrt{1-4z-4z^2}} \sum_{\alpha\in\Phi(R_n)} R_\alpha(z) \tag{5.26}$$

where

$$R_\alpha(z) = z^{k(\alpha)} C(z)^{c(\alpha)} \prod_{i=0}^{n-1} R_i(z)^{r_i(\alpha)}\,. \tag{5.27}$$

Let us start with some technical lemmas and propositions regarding the generating functions corresponding to normal-order reduction grammars.

Lemma 5.14. Let $n \geq 0$. Then, each $R_n(z)$ has a removable singularity at $z = 0$.

Proof. Induction over n. Following Riemann's removable singularities theorem (see Theorem 2.18), $R_n(z)$ has a removable singularity at $z = 0$ if and only if the limit $\lim_{z\to 0} zR_n(z)$ exists and is equal to 0. In particular, from (5.26) and (5.27)

$$\lim_{z\to 0} \frac{z}{\sqrt{1-4z-4z^2}} \left(z^{k(\alpha)} C(z)^{c(\alpha)} \prod_{i=0}^{n-1} R_i(z)^{r_i(\alpha)} \right) = 0 \tag{5.28}$$

for each $\alpha \in \Phi(R_n)$.

Let us start with $n = 0$. In this case, the product $\prod_{i=0}^{n-1} R_i(z)^{r_i(\alpha)}$ vanishes, simplifying (5.28) to

$$\lim_{z\to 0} \frac{z^{k(\alpha)+1} C(z)^{c(\alpha)}}{\sqrt{1-4z-4z^2}} = 0\,. \tag{5.29}$$

Due to Lemma 5.11, $C(z)^{c(\alpha)}$ has an analytic continuation including $z = 0$. In consequence, $\lim_{z\to 0} C(z)^{c(\alpha)}$ exists, indeed satisfying equation (5.29).

Now, suppose that $n > 0$. By the induction hypothesis all $R_0(z), \ldots, R_{n-1}(z)$ have removable singularities at $z = 0$. Using Lemma 2.19, we can moreover state that so do their powers $R_0(z)^{r_0(\alpha)}, \ldots, R_{n-1}(z)^{r_{n-1}(\alpha)}$. Together with our previous observation that $C(z)^{c(\alpha)}$ has an analytic continuation including the origin, we conclude that (5.28) is satisfied, finishing the proof. □

Definition 5.15 (Major and minor productions)**.** Let $\alpha \in \Phi(R_n)$ for some $n \geq 1$. The production α is said to be major if α references either $\mathcal{C}_{SK}$ or some R_i for $i \in \{1, \ldots, n-1\}$. Otherwise, α is said to be minor.

In the following lemma we use the notions of major and minor productions, showing that major productions contribute to the asymptotic growth rate of the underlying counting sequence of $R_n(z)$ whereas minor ones are asymptotically negligible.

Lemma 5.16. Let $n \geq 1$. Then each $R_n(z)$ is in form of $\sqrt{1-8z}\,P(z) + Q(z)$ where both $P(z)$ and $Q(z)$ are analytic in the disk $|z| < \rho_0$ excluding the origin.

Proof. Induction over n. Consider the base case $n = 1$. Let us divide $\Phi(R_1)$ into two groups, i.e. major and minor productions. Suppose that $\alpha \in R_1$ is a major production. Since $\alpha \in \Phi(R_1)$, its corresponding generating function $R_\alpha(z)$ is in form of

$$R_\alpha(z) = z^{k(\alpha)} C(z)^{c(\alpha)} R_0(z)^{r_0(\alpha)} \tag{5.30}$$

where in addition $c(\alpha) \geq 1$. Utilising Lemmas 5.11 and 5.12, we can further rewrite the formula (5.30) as

$$R_\alpha(z) = \sqrt{1-8z}\,\overline{P}(z) + \overline{Q}(z) \tag{5.31}$$

for functions $\overline{P}(z)$ and $\overline{Q}(z)$ analytic in the disk $|z| < \rho_0$ excluding the origin. Similarly, if $\alpha \in \Phi(R_1)$ is minor, we can rewrite its generating function as

$$R_\alpha(z) = \sqrt{1-4z-4z^2}\,\widehat{P}(z) + \widehat{Q}(z) \tag{5.32}$$

where $\widehat{P}(z)$ and $\widehat{Q}(z)$ are analytic in some disk $|z| < \rho_0 + \varepsilon$ for $\varepsilon > 0$ excluding the origin. The requested form of $R_1(z)$ follows now from Lemma 2.20 and the fact that $\rho_1 < \rho_0$.

Now, suppose that $n > 1$. Again, let us consider an arbitrary major production $\alpha \in \Phi(R_n)$. By the induction hypothesis and (5.27), we can rewrite $R_\alpha(z)$ as

$$R_\alpha(z) = z^{k(\alpha)} C(z)^{c(\alpha)} R_0(z)^{r_0(\alpha)} \prod_{i=1}^{n-1} \left(\sqrt{1-8z}\, \overline{P_i}(z) + \overline{Q_i}(z) \right)^{r_i(\alpha)}. \tag{5.33}$$

Applying Lemmas 5.11 and 5.12, we can further rewrite (5.33) as

$$R_\alpha(z) = \left(\sqrt{1-8z}\, \overline{P}(z) + \overline{Q}(z) \right) \prod_{i=1}^{n-1} \left(\sqrt{1-8z}\, \overline{\overline{P_i}}(z) + \overline{\overline{Q_i}}(z) \right)^{r_i(\alpha)}. \tag{5.34}$$

The result follows now easily from Lemma 2.20. □

Finally, we are in the position to prove Theorem 5.13.

Proof. (Theorem 5.13) Let $k > 0$. Due to Lemma 5.16, every function $R_k(z)$ is in form of $\sqrt{1-8z} P_k(z) + Q_k(z)$ for some algebraic functions $P_k(z)$ and $Q_k(z)$ that are analytic in the disk $|z| < \frac{\sqrt{2}-1}{2}$ excluding the origin. Moreover, by Lemma 5.14, every $R_k(z)$ has a removable singularity at the origin. Therefore, all functions $R_k(z)$ are amenable to the algebraic singularity analysis and admit the following asymptotic approximation:

$$[z^n] R_k(z) \sim 8^n \frac{\overline{C}_k n^{-3/2}}{\Gamma(-\frac{1}{2})} \tag{5.35}$$

where $\overline{C}_k$ is a constant depending entirely on k. □

Since $\rho_m = 1/8$ for every $m \geq 1$, we can easily compute the coefficients $\overline{C}_m$ in the asymptotic approximation of $[z^n] R_m(z)$ utilising available computer algebra systems, e.g. Mathematica [Wol15]. Note that the quotient $-\overline{C}_m/4$ (5.11) yields the desired asymptotic density of SK-combinators normalising in m normal-order reduction steps in the set of all combinators. In consequence, we obtain the following corollary.

Corollary 5.17. For each natural integer m, the asymptotic density of combinators normalising in m steps in the set of all SK-combinators is computable.

Combining our normal-order reduction grammar algorithm implementation with Mathematica we were able to compute the densities of combinators reducing in m normal-order reduction steps R_m in $\mathcal{C}_{SK}$ for $m = 1, \ldots, 7$. For convenience, let us denote $\mu_m = \mu\left(\frac{R_m}{\mathcal{C}_{SK}}\right)$. The results are summarised in Figure 5.1.

Exploiting the finite additivity of asymptotic density we obtain the following lower bound for normalising SK-combinators:

$$0.3401040259 \leq \mu^{-}\left(\frac{\mathcal{WN}_{SK}}{\mathcal{C}_{SK}}\right). \tag{5.36}$$

Clearly, the above lower bound can be further improved if we compute the next asymptotic densities for $m \geq 8$. Alas, due to the sheer amount of major productions this process

m	μ_m
1	0.0896123329
2	0.0641737440
3	0.0501056553
4	0.0413196741
5	0.0357099692
6	0.0311952570
7	0.0279873932

Figure 5.1: Density of R_m in $\mathcal{C}_{SK}$ for $m = 1, \ldots, 7$

is immensely time and memory consuming, quickly requiring resources exceeding available desktop computer capabilities.

Nevertheless, $\mu_m > 0$ for each m. Necessarily, $\sum_{m \geq 0} \mu_m \leq 1$ is convergent to some value $0 < \zeta < 1$. Moreover, if $\mu\left(\frac{\mathcal{WN}_{SK}}{\mathcal{C}_{SK}}\right)$ exists, then we have $\zeta \leq \mu\left(\frac{\mathcal{WN}_{SK}}{\mathcal{C}_{SK}}\right)$. Alas, the intriguing problem of determining whether the inequality can be replaced by an equality remains open.

5.3 Experimental results

Our method developed in Section 5.2 allows us to improve the lower bound (5.36) provided we have enough computational resources to find and manipulate generating functions $R_m(z)$. Unfortunately, the current gap between the lower and upper bound on the density of normalising combinators is still quite significant. In this section we present some experimental results regarding the aforementioned density as well as numerical evaluations of the obtained approximation error.

5.3.1 Super-computer results

Let us consider an experiment scheme $G(s, n, r)$ parametrised with three positive integer parameters s, n, r. We start with drawing s uniformly random SK-combinators of size n, using an exact-size sampler based on Rémy's algorithm [Rém85; Knu06]. Next, we reduce each of the s samples using up to r normal-order reduction steps. We record then the number of normalised samples, with their corresponding reduction lengths. Samples not normalising in r reduction steps are recorded with an artificially reduction length of -1. Finally, we collect the reduction lengths plotting the obtained function mapping reduction lengths to the number of samples attaining the specific reduction length.

Our experiments were performed on the Prometheus super-computer cluster granted by ACC Cyfronet AGH in Kraków, Poland [1]. Figure 5.2 summarises the experiment result for $G(s = 1200, n = 50,000,000, r = 1000)$.

[1]`http://www.cyfronet.krakow.pl/`, Accessed: 15.03.2017

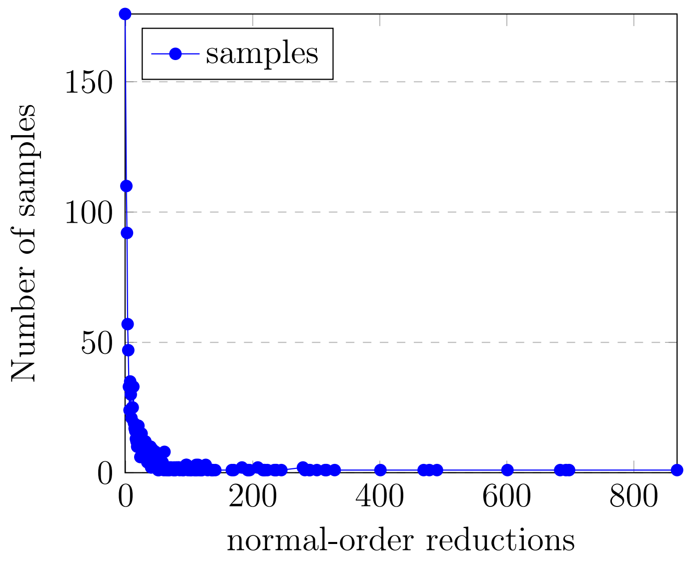


Figure 5.2: $G(s = 1200, n = 50,000,000, r = 1000)$

Even though the number $r = 1000$ bounding the number of reductions is significantly smaller than the size of considered samples, the experiment revealed that normalising combinators have short reduction lengths. In fact, the mean reduction length over all normalising samples is approximately equal to 31.5810 whereas $\log_2 n \approx 25.5754$. Out of 1200 samples, only 176 did not normalise in 1000 steps, yielding approximately 14.6% of all considered samples. Similar results were obtained with different parameters, suggesting that the ratio of normalising SK-combinators is approximately equal to 85% whereas the mean reduction length of normalising terms is $\Theta(\log_2 n)$. Our Haskell implementation of the program, as well as all the obtained data sets are available at [Ben16e].

5.3.2 Approximation error

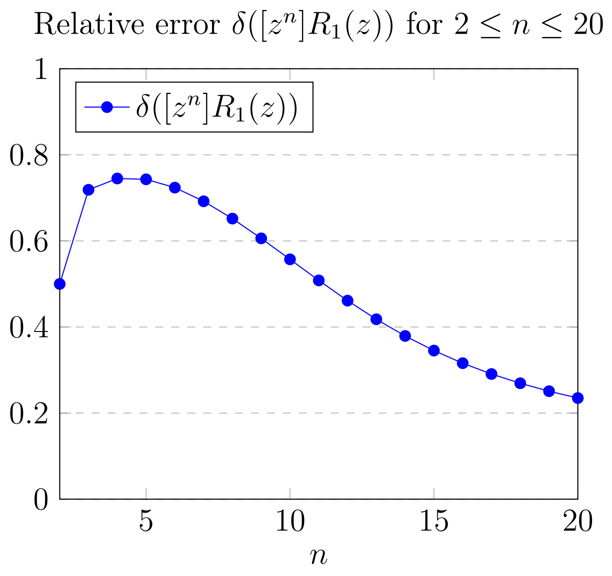
In Section 5.2 we proved that for positive m, the set of combinators reducing in m normal-order reduction steps has positive asymptotic density in the set of all combinators. However, techniques used to obtain this result do not provide direct access to the convergence rate of sequences in question. Using Mathematica we compared $[z^n]R_m(z)$ with its approximation $8^n\widetilde{C}_m n^{-3/2}$. Figures 5.3 and 5.4 summarise values for $[z^n]R_1(z)$ and their relative error $\delta([z^n]R_1(z))$.

Since $[z^n]R_1(z) \sim 8^n\widetilde{C}_1 n^{-3/2}$ the relative error $\delta([z^n]R_1(z))$ is inevitably tending to 0 as $n \to \infty$. Remarkably, the error $\delta([z^n]R_1(z))$ converges much more slowly than expected. With $n = 300$ the error is just of order 10^{-2}. We observed similar results in the relative errors for higher n, where the convergence rate is even slower than in the case of $[z^n]R_1(z)$.

Our Mathematica scripts and an implementation of the algorithm computing $R_m(z)$ are available at [Ben16d].

n	$[z^n]C(z)$	$[z^n]R_1(z)$	$\lfloor 8^n\widetilde{C}_1 n^{-3/2}\rfloor$	$\delta([z^n]R_1(z))$
2	16	4	2	0.50000
3	80	32	9	0.71875
4	448	200	51	0.74500
5	2688	1152	296	0.74305
6	16896	6528	1803	0.72380
7	109824	37184	11450	0.69207
8	732160	215328	74973	0.65181
9	4978688	1275520	502653	0.60592
10	34398208	7753472	3433386	0.55718
11	240787456	48412416	23808041	0.50822
12	1704034304	310294272	167159405	0.46128
13	12171673600	2037696512	1185980764	0.41797
14	87636049920	13675532288	8489666053	0.37920
15	635361361920	93532264448	61240081391	0.34525
16	4634400522240	650108973568	444715903783	0.31593
17	33985603829760	4580578080768	3248472837654	0.29081
18	250420238745600	32644683026432	23852497067944	0.26932
19	1853109766717440	234890688573440	175955235773882	0.25090
20	13765958267043840	1703833526784000	1303399617705108	0.23501

Figure 5.3: $[z^n]R_1(z) \sim 8^n\widetilde{C}_1 n^{-3/2}$ with $\widetilde{C}_1 \approx 0.10111668957132425$.



n	$\delta([z^n]R_1(z))$
20	0.23501
40	0.10914
60	0.07194
80	0.05369
100	0.04284
120	0.03564
140	0.03051
160	0.02667
180	0.02369
200	0.02131
220	0.01936
240	0.01774
260	0.01637
280	0.01520
300	0.01418

Figure 5.4: Relative error $\delta([z^n]R_1(z))$.

Chapter 6

Conclusions and open problems

We presented a quantitative analysis of λ-calculus in the de Bruijn notation and combinatory logic under various combinator bases, marking crucial similarities between both computational models. Notably, with respect to quantitative properties of normalisation, λ-calculus in the representation using de Bruijn indices stands in sharp contrast to the canonical representation of David et al. [Dav+13]. In the latter representation, variables do not contribute to the term size; hence, random λ-terms tend to avoid any fixed closed λ-term as a subterm. On the other hand, in the former representation, the average distance between a variable and its binding abstraction is constant which, arguably, leads to the precisely opposite result – a random λ-term contains any fixed λ-term as a subterm.

Such a striking disparity, where different computational models yield opposite, sometimes counterintuitive quantitative results with respect to undecidable properties is not entirely unexpected. Let us notice that similar questions regarding the existence of the asymptotic density of terminating computations were considered in the context of Turing machines (see e.g. [BDS15; BDS16; HM06]). Remarkably, depending on the assumed machine model the density of terminating Turing machine computations may or may not exist. Hamkins and Miasnikov consider the model with a single semi-infinite tape showing that asymptotically almost all Turing machines terminate their computations, quite rapidly falling of the tape [HM06]. On the other hand, Bienvenu et al. consider a general framework of algorithmic information theory optimal machines, showing that the fraction sequence of terminating computations cannot have a limit [BDS16]. With Tromp's introduction of Kolmogorov complexity to λ-calculus and combinatory logic [Tro06] it is quite natural to ask whether similar discrepancies hold in the universes of λ-calculus or combinatory logic.

Presented results regarding the asymptotic density of normalising combinatory logic terms provide a tentative answer to the general asymptotic density problem of terminating computations in combinatory logic. With our effective characterisation of normalising combinators, it became possible to utilise techniques of analytic combinatorics and give an algorithmic scheme of systematically improving the lower bound of the corresponding lower asymptotic density. Alas, our method requires extensive amounts of resources, both in terms of time and memory. Based on the initial numbers in the counting sequence corresponding to normal-order reduction grammar productions, the presented primitive recursive upper bound seems to be a vast overestimation. Nonetheless, the intriguing problem of finding better asymptotic approximations is left open.

Problem 6.1. Establish the asymptotic growth rate of the counting sequence corresponding to normal-order reduction grammar productions.

With available computing resources, we found that combinators normalising in up to seven normal-order reduction steps form a set of asymptotic density over 34% whereas super-computer experiments suggest that the actual asymptotic density of normalising combinators is close to 85%. The sheer amount of asymptotically significant fractions of normalising combinators reveals the intrinsic difficulty of establishing the existence (less alone the precise quantity) of the asymptotic density of normalising combinators.

Problem 6.2. Determine whether the set of normalising combinators has an asymptotic density in the set of all combinators. If so, determine its quantity.

In the case of λ-calculus, the problem of determining the asymptotic density of normalising λ-terms seems to be even more difficult. Here, the arguably main obstacle is the context-sensitive nature of the substitution operation, unlike the 'local' contraction in combinatory logic. In consequence, the corresponding reduction grammars for λ-calculus ought be, at least to some degree, context-sensitive as well. Alternatively, it is possible to investigate the same problem for some variant of λ-calculus with so-called explicit substitution, including substitution as a first-class citizen in the language (see e.g. [Les94]).

Problem 6.3. Determine whether the set of normalising λ-terms has an asymptotic density in the set of all λ-terms. If so, determine its quantity.

Incorporating substitution into the language of λ-calculus raises hopes for an effective combinatorial characterisation of normalising λ-terms in some variant with explicit substitution. Various λ-calculi with explicit substitution model it in diverse ways, yielding contrasting properties in terms of confluence on open terms or strong normalisation. In consequence different, quite natural problems arise, including the following.

Problem 6.4. Investigate the quantitative properties of λ-calculi with explicit substitution. Which substitution primitives contribute the most to the execution cost in the typical case?

Concerning more outright practical aspects of quantitative nature in λ-calculus and combinatory logic, the perhaps most interesting open problem is the effective uniform generation of large terms with predetermined properties, such as typeability or (strong) normalisation. Utilising techniques of Boltzmann sampling, particularly numerical oracles [PSS12], the problem of approximate-size generation of finitely specifiable systems of interesting terms is virtually closed (see e.g. [CD09; Dar+12; Xia16; Ben16a]). In some cases, e.g. neutral terms in the natural size notion, effective exact-size sampling is available due to effective bijections with known combinatorial structures. In particular, due to the neat binary tree sampling algorithm of Rémy [Rém85] it is possible to sample combinators of given size in arbitrary combinator bases. Finally, let us notice that for a broad variety of so-called admissible classes of terms, the less effective recursive method of Nijenhuis and Wilf involving large integer arithmetic is also available [NW78].

Alas, the arguably most interesting classes of λ-terms, including closed, typeable or normalising ones, are not known to be finitely specifiable in terms of admissible constructions; hence, do not fall directly under the methodological scope of Boltzmann samplers. Presented sampling methods for closed h-shallow λ-terms provide a partial framework meant to approximate the infinite system of closed λ-terms and, as such, are able to generate large λ-terms in a restricted class of closed ones. We note also that rejection methods for the full class of closed λ-terms are available due to the results of Gittenberger and Gołębiewski [GG16].

In the case of closed typeable λ-terms, no finite specification approximating an interesting non-trivial fragment of all typeable terms is known. We resort therefore to rejection sampling combined with logical programming techniques. Although the resulting sampler improves over previous approaches, the achievable term sizes of around 140 in the case of the natural size notion are still somewhat unsatisfactory.

Problem 6.5. Develop an efficient method of exact- or approximate-size generation of large λ-terms in an interesting, non-trivial fragment of closed typeable λ-terms. In particular, the whole set of closed typeable ones.

Let us remark that a natural candidate class of non-trivial typeable λ-terms is the class of so-called affine λ-terms, i.e. λ-terms in which each abstraction binds at most one variable (see e.g. [Hin96]).

With respect to normalisation, presented results regarding the effective construction of normal-order reduction grammars as well as their corresponding generating functions provide a novel approach to the generation of large normalisable combinators. Using Boltzmann sampler techniques, it becomes possible to construct random combinators reducing in any fixed number of normal-order reductions. Unfortunately, due to the quickly intractable size of reduction grammars, such tools are able to operate on just the few initial grammars. Nonetheless, under the standard translation to λ-calculus, combinators reducing under a few normal-order reduction steps constitute a novel, non-trivial source of large random normalising λ-terms.

With the growing popularity of property-based testing techniques involving, for instance, large random λ-terms, it becomes necessary to understand the statistical properties of large uniformly random structures. Presented results regarding the average de Bruijn index weight in a large random λ-term are one of various intriguing statistics worth exploring. In the case of plain λ-terms, our result was obtained utilising the moment techniques of analytic combinatorics; however, for most interesting classes, such as closed λ-terms, the application of moment techniques to infinite combinatorial systems becomes a challenging endeavour.

Problem 6.6. Investigate the distribution of various intriguing combinatorial parameters for plain and closed λ-terms. For instance, the average number of unbound indices in a random open λ-term or the average number of binding abstractions in a random closed λ-term.

Bibliography

[AS72] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1972. ISBN: 978-0-486-61272-0.

[Bar84] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Revised. Vol. 103. North Holland, 1984.

[BBJ13] Axel Bacher, Olivier Bodini and Alice Jacquot. "Exact-size Sampling for Motzkin Trees in Linear Time via Boltzmann Samplers and Holonomic Specification". In: *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*. SIAM, 2013, pp. 52–61.

[BDS15] Laurent Bienvenu, Damien Desfontaines and Alexander Shen. "What Percentage of Programs Halt?" In: *Automata, Languages, and Programming: 42nd International Colloquium, ICALP*. Springer Berlin Heidelberg, 2015, pp. 219–230.

[BDS16] Laurent Bienvenu, Damien Desfontaines and Alexander Shen. "Generic algorithms for halting problem and optimal machines revisited". In: *Logical Methods in Computer Science* 12.2 (2016), pp. 1–29.

[Ben+16a] 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+16b] Maciej Bendkowski, Katarzyna Grygiel, Pierre Lescanne and Marek Zaionc. "Combinatorics of λ-terms: a natural approach". In: *CoRR* abs/1609.07593 (2016). Accepted in Journal of Logic and Computation. URL: `http://arxiv.org/abs/1609.07593`.

[Ben16a] Maciej Bendkowski. *Boltzmann Brain: Boltzmann sampler compiler for combinatorial systems.* `https://github.com/maciej-bendkowski/boltzmann-brain`. Accessed: 15.03.2017. 2016.

[Ben16b] Maciej Bendkowski. *Lambda-sampler: Boltzmann sampler utilities for lambda calculus.* `https://hackage.haskell.org/package/lambda-sampler`. Accessed: 15.03.2017. 2016.

[Ben16c] Maciej Bendkowski. *Natural Counting of Lambda Terms - Haskell implementations.* `https://github.com/maciej-bendkowski/natural-counting-of-lambda-terms`. Accessed: 15.03.2017. 2016.

[Ben16d] Maciej Bendkowski. *Normal-order reduction grammars – Haskell implementation.* `https://github.com/maciej-bendkowski/normal-order-reduction-grammars`. Accessed: 15.03.2017. 2016.

[Ben16e] Maciej Bendkowski. *SK-sampler – Haskell implementation.* `https://github.com/maciej-bendkowski/sk-sampler`. Accessed: 15.03.2017. 2016.

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

[BGT16] Maciej Bendkowski, Katarzyna Grygiel and Paul Tarau. "Random generation of closed simply-typed λ-terms: a synergy between logic programming and Boltzmann samplers". In: *CoRR* abs/1612.07682 (2016). Submitted. URL: `http://arxiv.org/abs/1612.07682`.

[BGT17] Maciej Bendkowski, Katarzyna Grygiel and Paul Tarau. "Boltzmann Samplers for Closed Simply-Typed Lambda Terms". In: *19th International Symposium on Practical Aspects of Declarative Languages, PADL*. Springer International Publishing, 2017, pp. 120–135.

[BGZ15] Maciej Bendkowski, Katarzyna Grygiel and Marek Zaionc. "Asymptotic Properties of Combinatory Logic". In: *Theory and Applications of Models of Computation, TAMC*. Springer International Publishing, 2015, pp. 62–72.

[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`.

[Bod+13] Olivier Bodini, Danièle Gardy, Bernhard Gittenberger and Alice Jacquot. "Enumeration of Generalized BCI Lambda-terms". In: *Electronic Journal of Combinatorics* 20.4 (2013).

[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.

[CD09] Benjamin Canou and Alexis Darrasse. "Fast and Sound Random Generation for Automated Testing and Benchmarking in Objective Caml". In: *Proceedings of the 2009 ACM SIGPLAN Workshop on ML*. ACM, 2009, pp. 61–70.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic*. Vol. 1. Second printing 1968. North-Holland, 1958.

[CH00] Koen Claessen and John Hughes. "QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs". In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*. ACM, 2000, pp. 268–279.

[Cha+04] Brigitte Chauvin, Philippe Flajolet, Danièle Gardy and Bernhard Gittenberger. "And/Or Trees Revisited". In: *Combinatorics, Probability and Computing* 13.4-5 (2004), pp. 475–497.

[Chu36] Alonzo Church. "An Unsolvable Problem of Elementary Number Theory". In: *American Journal of Mathematics* 58.2 (1936), pp. 345–363.

[Com+07] Hubert Comon, Max Dauchet, Remi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison and Marc Tommasi. *Tree Automata Techniques and Applications*. Release October, 12th 2007. 2007. URL: `http://www.grappa.univ-lille3.fr/tata`.

[CR36] Alonzo Church and John B. Rosser. "Some properties of conversion". In: *Transactions of the American Mathematical Society* 39 (1936), pp. 472–482.

[Dar+12] Alexis Darrasse, Mathieu Dien, Antoine Genitrini, Marwan Ghanem, Frédéric Peschanski and Xuming Zhan. *Arbogen: a fast uniform random tree generator*. Accessed: 15.03.2017. 2012. URL: `https://github.com/fredokun/arbogen`.

[Dav+13] René David, Katarzyna Grygiel, Jakub Kozik, Christophe Raffalli, Guillaume Theyssier and Marek Zaionc. "Asymptotically almost all λ-terms are strongly normalizing". In: *Logical Methods in Computer Science* 9 (2013), pp. 1–30.

[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.

[Fag76] Ronald Fagin. "Probabilities on Finite Models". In: *The Journal of Symbolic Logic* 41.1 (1976), pp. 50–58.

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

[Fou+07] Hervé Fournier, Danièle Gardy, Antoine Genitrini and Marek Zaionc. "Classical and intuitionistic logic are asymptotically identical". In: *Computer Science Logic*. Springer Berlin Heidelberg, 2007, pp. 177–193.

[Fou+10] Hervé Fournier, Danièle Gardy, Antoine Genitrini and Marek Zaionc. "Tautologies over implication with negative literals". In: *Mathematical Logic Quarterly* 56.4 (2010), pp. 388–396.

[FS09] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. 1st ed. Cambridge University Press, 2009. ISBN: 0521898064, 9780521898065.

[Gar05] Danièle Gardy. "Random Boolean expressions". In: *Discrete Mathematics and Theoretical Computer Science, proceedings AF* (2005), pp. 1–36.

[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.

[GGM14] Bernhard Gittenberger, Antoine Genitrini and Cécile Mailler. "No Shannon Effect Induced by And/Or Trees". In: *25th International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*. 2014.

[GK12] Antoine Genitrini and Jakub Kozik. "In the full propositional logic, 5/8 of classical tautologies are intuitionistically valid". In: *Annals of Pure and Applied Logic* 163.7 (2012), pp. 875–887.

[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.

[GLM08] Nancy S. S. Gu, Nelson Y. Li and Toufik Mansour. “2-Binary trees: Bijections and related issues”. In: *Discrete Mathematics* 308.7 (2008), pp. 1209–1221.

[Gra83] Etienne Grandjean. “Complexity of the first-order theory of almost all finite structures”. In: *Information and Control* 57.2–3 (1983), pp. 180–204.

[Hin96] J. Roger Hindley. *Basic simple type theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1996. ISBN: 0-521-46518-4.

[HM06] Joel D. Hamkins and Alexei Miasnikov. “The Halting Problem is decidable on a set of asymptotic probability one”. In: *Notre Dame Journal of Formal Logic* 47.4 (2006), pp. 515–524.

[Kle36] Stephen C. Kleene. “λ-definability and recursiveness”. In: *Duke Mathematical Journal* 2.2 (1936), pp. 340–353.

[Klo92] Jan W. Klop. “Handbook of Logic in Computer Science”. In: ed. by S. Abramsky, Dov M. Gabbay and S. E. Maibaum. Vol. 2. Oxford University Press, Inc., 1992. Chap. Term Rewriting Systems, pp. 1–116. ISBN: 0-19-853761-1.

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

[Koz08] Jakub Kozik. “Subcritical pattern languages for and/or trees”. In: *Fifth Colloquium on Mathematics and Computer Science.* DMTCS, 2008, pp. 437–448.

[Kra99] Steven Krantz. *Handbook of Complex Variables.* 1st ed. Birkhäuser Basel, 1999. ISBN: 978-0-8176-4011-8, 978-1-4612-7206-9.

[Les13] Pierre Lescanne. “On counting untyped lambda terms”. In: *Theoretical Computer Science* 474 (2013), pp. 80–97.

[Les14] Pierre Lescanne. “Boltzmann samplers for random generation of lambda terms”. In: *CoRR* abs/1404.3875 (2014). URL: http://arxiv.org/abs/1404.3875.

[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.

[MTZ00] Małgorzata Moczurad, Jerzy Tyszkiewicz and Marek Zaionc. “Statistical properties of simple types”. In: *Mathematical Structures in Computer Science* 10.05 (2000), pp. 575–594.

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

[Pał12] Michał H. Pałka. "Random Structured Test Data Generation for Black-Box Testing". PhD thesis. Chalmers University of Technology, 2012.

[Pey87] Simon L. Peyton Jones. *The Implementation of Functional Programming Languages.* Prentice-Hall, Inc., 1987. ISBN: 013453333X.

[PSS12] Carine Pivoteau, Bruno Salvy and Michèle Soria. "Algorithms for combinatorial structures: Well-founded systems and Newton iterations". In: *Journal of Combinatorial Theory.* A 119.8 (2012), pp. 1711–1773. ISSN: 0097-3165.

[Puy04] Vincent Puyhaubert. "Generating functions and the satisfiability threshold". In: *Discrete Mathematics and Theoretical Computer Science* 6 (2004), pp. 425–436.

[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.

[Slo64] Neil J. A. Slone. *Online Encyclopedia of Integer Sequences.* Accessed: 15.03.2017. 1964. URL: `http://oeis.org/`.

[STT06] Aristidis Sapounakis, Ioannis Tasoulas and Panagiotis Tsikouras. "Ordered trees and the inorder traversal". In: *Discrete Mathematics* 306.15 (2006), pp. 1732–1741.

[SU06] Morten H. Sørensen and Paweł Urzyczyn. *Lectures on the Curry-Howard Isomorphism.* Vol. 149. Studies in Logic and the Foundations of Mathematics. Elsevier Science Inc., 2006. ISBN: 0444520775.

[SZ94] Bruno Salvy and Paul Zimmermann. "Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable". In: *ACM Transactions on Mathematical Software* 20.2 (1994), pp. 163–177.

[Tar16] Paul Tarau. *Prolog generators and samplers for lambda terms.* `http://www.cse.unt.edu/~tarau/research/2016/bol.pro`. Accessed: 15.03.2017. 2016.

[Tho04] Lars Thorlund-Petersen. "Global convergence of Newton's method on an interval". In: *Mathematical Methods of Operations Research* 59.1 (2004), pp. 91–110.

[Tro06] John Tromp. "Binary Lambda Calculus and Combinatory Logic". In: *Kolmogorov Complexity and Applications.* Vol. 06051. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[Urz03] Paweł Urzyczyn. "A Simple Proof of the Undecidability of Strong Normalisation". In: *Mathematical Structures in Computer Science* 13.1 (2003), pp. 5–13.

[Wan05] Jue Wang. *Generating random lambda calculus terms.* Tech. rep. Boston University, 2005.

[Wil06] Herbert S. Wilf. *Generatingfunctionology.* A. K. Peters, Ltd., 2006. ISBN: 1568812795.

[Wol15] Inc. Wolfram Research. *Mathematica Version 10.3.* Champaign, Illinois. 2015.

[Xia16] Li-yao Xia. *Generic random*. Accessed: 15.03.2017. 2016. URL: `https://github.com/Lysxia/generic-random`.