Polynomial tuning of multiparametric combinatorial samplers*

Maciej Bendkowski† Olivier Bodini‡ Sergey Dovgal‡§¶

Abstract

Boltzmann samplers and the recursive method are prominent algorithmic frameworks for the approximate-size and exact-size random generation of large combinatorial structures, such as maps, tilings, RNA sequences or various tree-like structures. In their multiparametric variants, these samplers allow to control the profile of expected values corresponding to multiple combinatorial parameters. One can control, for instance, the number of leaves, profile of node degrees in trees or the number of certain subpatterns in strings. However, such a flexible control requires an additional non-trivial tuning procedure. In this paper, we propose an efficient polynomial-time, with respect to the number of tuned parameters, tuning algorithm based on convex optimisation techniques. Finally, we illustrate the efficiency of our approach using several applications of rational, algebraic and Pólya structures including polyomino tilings with prescribed tile frequencies, planar trees with a given specific node degree distribution, and weighted partitions.

1 Introduction

Uniform random generation of combinatorial structures forms a prominent research area of computer science with multiple important applications ranging from automated software testing techniques, see [19], to complex simulations of large physical statistical models, see [8]. Given a formal specification defining a set of combinatorial structures (for instance graphs, proteins or tree-like data structures) we are interested in their efficient random sampling ensuring the uniform distribution among all structures sharing the same size.

One of the earliest examples of a generic sampling template is Nijenhuis and Wilf's recursive method [35] later systematised by Flajolet, Zimmermann and Van Cutsem [27]. In this approach, the generation scheme is split into two stages – an initial preprocessing phase where recursive branching probabilities dictating subsequent sampler decisions are computed, and the proper sampling phase itself. Alas, in both phases the algorithm manipulates integers of size exponential in the target size n, turning its effective bit complexity to $O(n^{3+\varepsilon})$, compared to $\Theta(n^2)$ arithmetic operations required. Denise and Zimmermann reduced later the average-case bit complexity of the recursive method to $O(n \log n)$ in time and $O(n)$ in space using a certified floating-point arithmetic optimisation [20]. Regardless, worst-case space bit complexity remained $O(n^2)$ as well as bit complexity for non-algebraic languages. Remarkably, for rational languages Bernardi and Giménez [6] recently linked the floating-point optimisation of Denise and Zimmermann with a specialised divide-and-conquer scheme reducing further the worst-case space bit complexity and the average-case time bit complexity to $O(n)$.

A somewhat relaxed, approximate-size setting of the initial generation problem was investigated by Duchon, Flajolet, Louchard and Schaeffer who proposed a universal sampler construction framework of so-called Boltzmann samplers [24]. The key idea in their approach is to embed the generation scheme into the symbolic method of analytic combinatorics [26] and, in consequence, obtain an effective recursive sampling template for a wide range of existing combinatorial classes. In recent years, a series of important improvements was proposed for both unlabelled and Pólya structures. Let us mention for instance linear approximate-size (and quadratic exact-size) Boltzmann samplers for planar graphs [28], general-purpose samplers for unlabelled structures [25], efficient samplers for plane partitions [11] or the cycle pointing operator for Pólya struc-

*Maciej Bendkowski was partially supported within the Polish National Science Center grant 2016/21/N/ST6/01032 and the French Government Scholarship within the French-Polish POLONIUM grant number 34648/2016. Olivier Bodini and Sergey Dovgal were supported by the French project ANR project MetACOnc, ANR-15-CE40-0014.

†Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland. Email: bendkowski@tcs.uj.edu.pl.

‡Institut Galilée, Université Paris 13, 99 Avenue Jean Baptiste Clément, 93430 Villetaneuse, France. Email: {Olivier.Bodini,Dovgal}@lipn.univ-paris13.fr

§Institut de Recherche en Informatique Fondamentale, Université Paris 7, 5 Rue Thomas Mann 75013 Paris, France.

¶Moscow Institute of Physics and Technology, Institutskiy per. 9, Dolgoprudny, Russia 141700

tures [16]. Moreover, the framework was generalised onto differential specifications [15, 10]; linear exact-size samplers for Catalan and Motzkin trees were obtained, exploiting the shape of their holonomic specifications [1].

What was left open since the initial work of Duchon et al., was the development of (i) efficient Boltzmann oracles providing effective means of evaluating combinatorial systems within their disks of convergence and (ii) an automated tuning procedure controlling the expected sizes of parameter values of generated structures. The former problem was finally addressed by Pivoteau, Salvy and Soria [38] who defined a rapidly converging combinatorial variant of the Newton oracle by lifting the combinatorial version of Newton's iteration of Bergeron, Labelle and Leroux [5] to a new numerical level. In principle, using their Newton iteration and an appropriate use of binary search, it became possible to approximate the singularity of a given algebraic combinatorial system with arbitrarily high precision. However, even if the singularity ρ is estimated with precision 10^{-10} its approximation quality does not correspond to an equally accurate approximation of the generating function values at ρ, often not better than 10^{-2}. Precise evaluation at z close to ρ requires an extremely accurate precision of z. Fortunately, it is possible to trade-off the evaluation precision for an additional rejection phase using the idea of analytic samplers [13] retaining the uniformity even with rough evaluation estimates.

Nonetheless, frequently in practical applications including for instance semi-automated software testing techniques, additional control over the internal structure of generated objects is required, see [37]. In [14] Bodini and Ponty proposed a multidimensional Boltzmann sampler model, developing a tuning algorithm meant for the random generation of words from context-free languages with a given target letter frequency vector. However, their algorithm converges only in an *a priori* unknown vicinity of the target tuning variable vector. In practice, it is therefore possible to control no more than a few tuning parameters at the same time.

In the present paper we propose a novel polynomial-time tuning algorithm based on convex optimisation techniques, overcoming the previous convergence difficulties. We demonstrate the effectiveness of our approach with several examples of rational, algebraic and Pólya structures. Remarkably, with our new method, we are easily able to handle large combinatorial systems with thousands of combinatorial classes and tuning parameters.

In order to illustrate the effectiveness of our approach, we have implemented a prototype sampler generator Boltzmann Brain (bb in short). The source code is available at Github[1]. Supplementary scripts used to generate and visualise the presented applications of this paper are available as a separate repository[2].

In § 2 we briefly recall the principles of Boltzmann sampling. Next, in § 3 we describe the tuning procedure. In § 4 we propose four exemplary applications and explain the interface of bb. Finally, in the appendix we give the proofs of the theorems, discuss implementation details and describe a novel exact-size sampling algorithm for strongly connected rational grammars.

2 Sampling from Boltzmann principles

2.1 Specifiable k-parametric combinatorial classes. Let us consider the neutral class $\mathcal{E}$ and its atomic counterpart $\mathcal{Z}$, both equipped with a finite set of admissible operators (i.e. disjoint union $+$, Cartesian product $\times$, sequence SEQ, multiset MSET and cycle CYC), see [26, pp. 24–30]. Combinatorial specifications are finite systems of equations (possibly recursive) built from elementary classes $\mathcal{E}$, $\mathcal{Z}$ and the admissible operators.

Example. Consider the following joint specification for $\mathcal{T}$ and $\mathcal{Q}$. In the combinatorial class $\mathcal{T}$ of trees, nodes of even level (the root starts at level one) have either no or two children and each node at odd level has an arbitrary number of non-planarily ordered children:

$$\begin{cases} \mathcal{T} = \mathcal{Z}\,\mathrm{MSET}(\mathcal{Q})\,, \\ \mathcal{Q} = \mathcal{Z} + \mathcal{Z}\mathcal{T}^2\,. \end{cases} \tag{2.1}$$

In order to distinguish (in other words *mark*) some additional combinatorial parameters we consider the following natural multivariate extension of specifiable classes.

DEFINITION 2.1. *(Specifiable k-parametric combinatorial classes) A specifiable k-parametric combinatorial class is a combinatorial specification built, in a possibly recursive manner, from k distinct atomic classes $\mathcal{Z}_i$ ($i \in \{1, \ldots, k\}$), the neutral class $\mathcal{E}$ and admissible operators $+, \times, \mathrm{SEQ}, \mathrm{MSET}$ and CYC. In particular, a vector $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \ldots, \mathcal{C}_m)$ forms a specifiable k-parametric combinatorial class if its specification can be written down as*

$$\begin{cases} \mathcal{C}_1 = \Phi_1(\boldsymbol{\mathcal{C}}, \mathcal{Z}_1, \ldots, \mathcal{Z}_k)\,, \\ \vdots \\ \mathcal{C}_m = \Phi_m(\boldsymbol{\mathcal{C}}, \mathcal{Z}_1, \ldots, \mathcal{Z}_k) \end{cases} \tag{2.2}$$

[1] https://github.com/maciej-bendkowski/boltzmann-brain

[2] https://github.com/maciej-bendkowski/multiparametric-combinatorial-samplers

where the right-hand side expressions are composed from $\mathcal{C}, \mathcal{Z}_1, \ldots, \mathcal{Z}_k$, admissible operators and the neutral class $\mathcal{E}$. Moreover, we assume that specifiable k-parametric combinatorial specifications form well-founded aperiodic systems, see [5, 38, 23].

Example. Let us continue our running example, see (2.1). Note that we can introduce two additional *marking* classes $\mathcal{U}$ and $\mathcal{V}$ into the system, of weight zero each, turning it in effect to a k-specifiable combinatorial class as follows:

$$\begin{cases} \mathcal{T} = \mathcal{U}\mathcal{Z}\,\mathrm{MSet}(\mathcal{Q}), \\ \mathcal{Q} = \mathcal{V}\mathcal{Z} + \mathcal{Z}\mathcal{T}^2\,. \end{cases} \tag{2.3}$$

In this example, $\mathcal{U}$ is meant to mark the occurrences of nodes at odd levels, whereas $\mathcal{V}$ is meant to mark leaves at even levels. In effect, we *decorate* the univariate specification with explicit information regarding the internal structural patterns of our interest.

Much like in their univariate variants, k-parametric combinatorial specifications are naturally linked to ordinary multivariate generating functions, see e.g [26].

Definition 2.2. *(Multivariate generating functions) The multivariate ordinary generating function in variables $z_1, \ldots, z_k$ associated to a specifiable k-parametric combinatorial class $\mathcal{C}$ is defined as*

$$C(z_1, \ldots, z_k) = \sum_{p_1 \geq 0, \ldots, p_k \geq 0} c_{\boldsymbol{p}} \boldsymbol{z}^{\boldsymbol{p}} \tag{2.4}$$

where $c_{\boldsymbol{p}} = c_{p_1,\ldots,p_k}$ denotes the number of structures with p_i atoms of type $\mathcal{Z}_i$ and $\boldsymbol{z}^{\boldsymbol{p}}$ denotes the product $z_1^{p_1} \cdots z_k^{p_k}$. In the sequel, we call $\boldsymbol{p}$ the (composition) size of the structure.

In this setting, we can easily lift the usual univariate generating function building rules to the realm of multivariate generating functions associated to specifiable k-parametric combinatorial classes. Table 1 summarises these rules.

2.2 Multiparametric Boltzmann samplers. Consider a typical multiparametric Boltzmann sampler workflow [14] on our running example, see (2.3). We start with choosing target expectation quantities (n, k, m) of nodes from atomic classes $(\mathcal{Z}, \mathcal{U}, \mathcal{V})$. Next, using a dedicated tuning procedure we obtain a vector of three real positive numbers $\boldsymbol{z} = (z, u, v)$ depending on (n, k, m). Then, we construct a set of recursive Boltzmann samplers $\Gamma\mathcal{U}(\boldsymbol{z}), \Gamma\,\mathrm{MSet}(\mathcal{Q}(\boldsymbol{z}))$, etc. according to the building rules in Table 1. Finally, we use the so constructed samplers to generate structures with tuned parameters.

In order to sample from either $\mathcal{E}$ or atomic classes, we simply construct the neutral element ε or an appropriate atomic structure $\square_i$, respectively. For union classes we make a Bernoulli choice depending on the quotients of respective generating functions values and continue with sampling from the resulting class. In the case of product classes, we spawn two independent samplers, one for each class, and return a pair of built structures. Finally, for $\textsc{Seq}(\mathcal{A})$ we draw a random value from a geometric distribution with parameter $1 - A(\boldsymbol{z})$ and spawn that many samplers corresponding to the class $\mathcal{A}$. In other words, $\mathbb{P}(\ell \text{ instances}) = A(\boldsymbol{z})^\ell (1 - A(\boldsymbol{z}))$. In the end, we collect the sampler outcomes and return their list. The more involved MSet and Cyc constructions are detailed in Appendix C.

The probability space associated to so constructed Boltzmann samplers takes then the following form. Let $\boldsymbol{z} \in (\mathbb{R}^+)^k$ be a vector inside the ball of convergence of $C(\boldsymbol{z})$ and ω be a structure of composition size $\boldsymbol{p}$ in a k-parametric class $\mathcal{C}$. Then, the probability that ω becomes the output of a multiparametric Boltzmann sampler $\Gamma\mathcal{C}(\boldsymbol{z})$ is given as

$$\mathbb{P}_{\boldsymbol{z}}(\omega) = \frac{\boldsymbol{z}^{\boldsymbol{p}}}{C(\boldsymbol{z})}\ . \tag{2.5}$$

Proposition 2.1. *Let $\boldsymbol{N} = (N_1, \ldots, N_k)$ be the random vector where N_i equals the number of atoms of type $\mathcal{Z}_i$ in a random combinatorial structure returned by the k-parametric Boltzmann sampler $\Gamma\mathcal{C}(\boldsymbol{z})$. Then, the expectation vector $\mathbb{E}_{\boldsymbol{z}}(\boldsymbol{N})$ and the covariance matrix $\mathrm{Cov}_{\boldsymbol{z}}(\boldsymbol{N})$ are given by*

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z}}(N_i) &= \frac{\partial}{\partial \xi_i} \log C(e^{\boldsymbol{\xi}})\Big|_{\boldsymbol{\xi} = \log \boldsymbol{z}} \quad \text{and} \\ \mathrm{Cov}_{\boldsymbol{z}}(\boldsymbol{N}) &= \left[\frac{\partial^2}{\partial \xi_i \partial \xi_j} \log C(e^{\boldsymbol{\xi}})\right]_{i,j=1}^{k}\Bigg|_{\boldsymbol{\xi} = \log \boldsymbol{z}}\ . \end{aligned} \tag{2.6}$$

Hereafter, we use $e^{\boldsymbol{z}}$ to denote coordinatewise exponentiation.

Corollary 2.1. *The function $\gamma(\boldsymbol{z}) := \log C(e^{\boldsymbol{z}})$ is convex because its matrix of second derivatives, as a covariance matrix, is positive semi-definite inside the set of convergence. This crucial assertion will later prove central to the design of our tuning algorithm.*

Remark 2.1. *Uniparametric recursive samplers of Nijenhuis and Wilf take, as well as Boltzmann samplers, a system of generating functions as their input. This system can be modified by putting fixed values of tuning variables, in effect altering the corresponding*

Table 1: Multivariate generating functions and their Boltzmann samplers $\Gamma\mathcal{C}(\boldsymbol{z})$.

Class	Description	$C(\boldsymbol{z})$	$\Gamma\mathcal{C}(\boldsymbol{z})$
Neutral	$\mathcal{C} = \{\varepsilon\}$	$C(\boldsymbol{z}) = 1$	ε
Atom	$\mathcal{C} = \{t_i\}$	$C(\boldsymbol{z}) = z_i$	$\square_i$
Union	$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$A(\boldsymbol{z}) + B(\boldsymbol{z})$	$\text{Bern}\left(\frac{A(\boldsymbol{z})}{C(\boldsymbol{z})}, \frac{B(\boldsymbol{z})}{C(\boldsymbol{z})}\right) \longrightarrow \Gamma\mathcal{A}(\boldsymbol{z}) \mid \Gamma\mathcal{B}(\boldsymbol{z})$
Product	$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$A(\boldsymbol{z}) \times B(\boldsymbol{z})$	$(\Gamma\mathcal{A}(\boldsymbol{z}), \Gamma\mathcal{B}(\boldsymbol{z}))$
Sequence	$\mathcal{C} = \textsc{Seq}(\mathcal{A})$	$(1 - A(\boldsymbol{z}))^{-1}$	$\ell := \text{Geom}(1 - A(\boldsymbol{z})) \longrightarrow (\Gamma\mathcal{A}(\boldsymbol{z}))_{\times \ell \text{ times}}$
MultiSet	$\textsc{MSet}(\mathcal{A})$	$\exp\left(\sum_{m=1}^{\infty} \frac{1}{m} A(\boldsymbol{z}^m)\right)$	see Algorithm 2, Appendix C
Cycle	$\textsc{Cyc}(\mathcal{A})$	$\sum_{m=1}^{\infty} \frac{\varphi(m)}{m} \ln \frac{1}{1 - A(\boldsymbol{z}^m)}$	see Algorithm 1, Appendix C

branching probabilities. The resulting distribution of the random variable corresponding to a weighted recursive sampler coincides with the distribution of the Boltzmann-generated variable conditioned on the structure size. As a corollary, the tuning procedure that we discuss in the following section is also valid for the exact-size approximate-frequency recursive sampling. In Appendix B we describe an algorithm for rational specifications which samples objects of size $n + O(1)$. As a by-product, we show how to convert approximate-size samplers corresponding to rational systems into exact-size samplers.

3 Tuning as a convex optimisation problem

We start with a general result about converting the problem of tuning arbitrary specifiable k-parametric combinatorial specifications into a convex optimisation problem, provided that one has access to an oracle yielding values and derivatives of corresponding generating functions. We note that this general technique can be applied to differential specifications as well. We write $f(\cdot) \to \min_{\boldsymbol{z}}$, $f(\cdot) \to \max_{\boldsymbol{z}}$ to denote the minimisation (maximisation, respectively) problem of the target function $f(\cdot)$ with respect to the vector variable $\boldsymbol{z}$. All proofs are postponed until Appendix A. Throughout this section, we assume that given tuning expectations are *admissible* in the sense that there always exists a target vector $\boldsymbol{z}^*$ corresponding to (2.6). Furthermore, we assume that the combinatorial system is *well-founded* and *strongly connected*. Some non-strongly connected cases fall into the scope of our framework as well, but for the core proof ideas we concentrate only on strongly connected systems.

THEOREM 3.1. *Consider a multiparametric combinatorial class $\mathcal{C}$. Fix the expectations $\mathbb{E}_{\boldsymbol{z}}\boldsymbol{N} = \boldsymbol{\nu}$, see Proposition 2.1. Let $C(\boldsymbol{z})$ be the generating function corresponding to $\mathcal{C}$. Then, the tuning vector $\boldsymbol{z}$, see* (2.6)*, is equal to $e^{\boldsymbol{\xi}}$ where $\boldsymbol{\xi}$ comes from the following minimisation problem:*

$$\log C(e^{\boldsymbol{\xi}}) - \boldsymbol{\nu}^\top \boldsymbol{\xi} \to \min_{\boldsymbol{\xi}} \quad . \tag{3.7}$$

Let us turn to the specific classes of algebraic and rational specification. In those cases, no differential-equation type systems are allowed; however, it is possible to reformulate the problem so that no extra oracles are required.

THEOREM 3.2. *Let $\boldsymbol{\mathcal{C}} = \boldsymbol{\Phi}(\boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{Z}})$ be a multiparametric algebraic system with $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \ldots, \mathcal{C}_m)$. Fix the expectations N_i of the parameters of objects sampled from $\mathcal{C}_1$ to $\mathbb{E}_{\boldsymbol{z}}\boldsymbol{N} = \boldsymbol{\nu}$. Then, the tuning vector $\boldsymbol{z}$ is equal to $e^{\boldsymbol{\xi}}$ where $\boldsymbol{\xi}$ comes from the convex problem:*

$$\begin{cases} c_1 - \boldsymbol{\nu}^\top \boldsymbol{\xi} \to \min_{\boldsymbol{\xi}, \boldsymbol{c}} \ , \\ \log \boldsymbol{\Phi}(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}}) - \boldsymbol{c} \leq 0. \end{cases} \tag{3.8}$$

Hereafter, "$\leq$" and $\log \boldsymbol{\Phi}$ denote a set of inequalities and the coordinatewise logarithm, respectively.

Let us note that the above theorem naturally extends to the case of labelled structures with SET and CYC operators. For unlabelled Pólya operators like MSET or CYC, we have to truncate the specification to bound the number of substitutions. In consequence, it becomes possible to sample corresponding unlabelled structures, including partitions, functional graphs, series-parallel circuits, etc.

Singular Boltzmann samplers (also defined in [24]) are the limiting variant of ordinary Boltzmann samplers with an infinite expected size of generated structures. In their multivariate version, samplers are considered *singular* if their corresponding variable vectors belong to the boundary of the respective convergence sets.

THEOREM 3.3. *Let $\boldsymbol{\mathcal{C}} = \boldsymbol{\Phi}(\boldsymbol{\mathcal{C}}, \mathcal{Z}, \boldsymbol{\mathcal{U}})$ be a multiparametric algebraic system with $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \ldots, \mathcal{C}_m)$, the atomic class $\mathcal{Z}$ marking the corresponding structure size and*

$\boldsymbol{\mathcal{U}} = (\mathcal{U}_1, \ldots, \mathcal{U}_k)$ being a vector (possibly empty) of distinguished atoms. Assume that the target expected frequencies of the atoms $\mathcal{U}_i$ are given by the vector $\boldsymbol{\alpha}$. Then, the variables $(z, \boldsymbol{u})$ that deliver the tuning of the corresponding singular Boltzmann sampler are the result of the following convex optimisation problem, where $z = e^{\xi}$, $\boldsymbol{u} = e^{\boldsymbol{\eta}}$:

$$\begin{cases} \xi + \boldsymbol{\alpha}^{\top}\boldsymbol{\eta} \to \max_{\xi,\boldsymbol{\eta},\boldsymbol{c}}\ , \\ \log \boldsymbol{\Phi}(e^{\boldsymbol{c}}, e^{\xi}, e^{\boldsymbol{\eta}}) - \boldsymbol{c} \le 0\,. \end{cases} \tag{3.9}$$

Finally, let us note that all of the above outlined convex programs can be effectively optimised using the polynomial-time interior-point method optimisation procedure of Nesterov and Nemirovskii [34]. The required precision ε is typically $Poly(n)$, see Appendix A.

THEOREM 3.4. *For multiparametric combinatorial systems with description length L, the tuning problem can be solved with precision ε in time $O\big(L^{3.5} \log \frac{1}{\varepsilon}\big)$.*

Let us complete this section by constructing an optimisation system for (2.1). Let (n, k, m) be the target expectation quantities of $(\mathcal{Z}, \mathcal{U}, \mathcal{V})$. By the rules in Table 1, the system of functional equations and its log-exp transformed optimisation counterpart take the form

$$\begin{cases} T(z,u,v) = uz \exp\left(\displaystyle\sum_{i=1}^{\infty} \frac{Q(z^i,u^i,v^i)}{i}\right), \\ Q(z,u,v) = vz + zT(z,u,v)^2\,. \end{cases} \tag{3.10}$$

Setting $T(z^i,u^i,v^i) = e^{\tau_i}$, $Q(z^i,u^i,v^i) = e^{\kappa_i}$, $z = e^{\zeta}$, $u = e^{\eta}$, $v = e^{\phi}$, we obtain

$$\begin{cases} \tau_1 - n\zeta - k\eta - m\phi \to \min, \\ \tau_j \ge \eta j + \zeta j + \displaystyle\sum_{i=1}^{\infty} \frac{e^{\kappa_{ij}}}{i}, \quad j \in \{1,2,\ldots\} \\ \kappa_j \ge \log(e^{\phi j + \zeta j} + e^{\zeta j + 2\tau_j}), \quad j \in \{1,2,\ldots\}\ . \end{cases} \tag{3.11}$$

For practical purposes, the sum can be truncated with little effect on distribution.

4 Applications

In this section we present several examples illustrating the wide range of applications of our tuning techniques. Afterwards, we briefly discuss our prototype sampler generator and its implementation details.

4.1 Polyomino tilings. We start with a benchmark example of a rational specification defining $n \times 7$ rectangular tilings using up to 126 different tile variants (a toy example of so-called transfer matrix models, cf. [26, Chapter V.6, Transfer matrix models]).

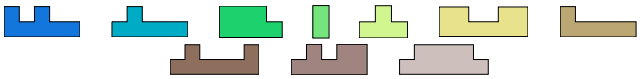


Figure 1: Examples of admissible tiles

We begin the construction with defining the set T of admissible tiles. Each tile $t \in T$ consists of two horizontal layers. The base layer is a single connected block of width $w_t \le 6$. The second layer, placed on top of the base one, is a subset (possibly empty) of w_t blocks, see Figure 1. For presentation purposes each tile is given a unique, distinguishable colour.

Next, we construct the asserted rational specification following the general construction method of defining a deterministic automaton with one state per each possible partial tiling configuration using the set T of available tiles. Tracking the evolution of attainable configurations while new tiles arrive, we connect relevant configurations by suitable transition rules in the automaton. Finally, we (partially) minimise the constructed automaton removing states unreachable from the initial empty configuration. Once the automaton is created, we tune the tiling sampler such that the target colour frequencies are uniform, i.e. each colour occupies, on average, approximately $\frac{1}{126} \approx 0.7936\%$ of the outcome tiling area. Figure 2 depicts an exemplary tiling generated by our sampler.

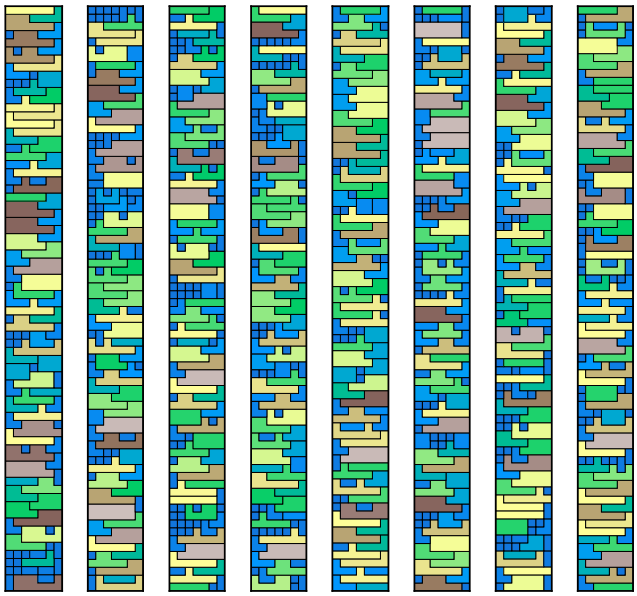


Figure 2: Eight random $n \times 7$ tilings of areas in the interval $[500; 520]$ using in total 95 different tiles.

The automaton corresponding to our tiling sampler

consists of more than 2000 states and 28,000 transitions. We remark that this example is a notable improvement over the work of Bodini and Ponty [14] who were able to sample $n \times 6$ tilings using 7 different tiles (we handle 126) with a corresponding automaton consisting of roughly 1500 states and 3200 transitions.

4.2 Simply-generated trees with node degree constraints. Next, we give an example of simple varieties of plane trees with fixed sets of admissible node degrees, satisfying the general equation

$$y(z) = z\phi(y(z)) \quad \text{for some polynomial} \quad \phi\colon \mathbb{C} \to \mathbb{C}\,.$$

Let us consider the case of plane trees where nodes have degrees in the set $D = \{0, \ldots, 9\}$, i.e. $\phi(y(z)) = a_0 + a_1 y(z) + a_2 y(z)^2 + \cdots + a_9 y(z)^9$. Here, the numbers $a_0, a_1, a_2, \ldots, a_9$ are nonnegative real coefficients. We tune the corresponding algebraic specification so to achieve a target frequency of 1% for all nodes of degrees $d \geq 2$. Frequencies of nodes with degrees $d \leq 1$ are left undistorted. For presentation purposes all nodes with equal degree are given the same unique, distinguishable colour. Figure 3 depicts two exemplary trees generated in this manner.

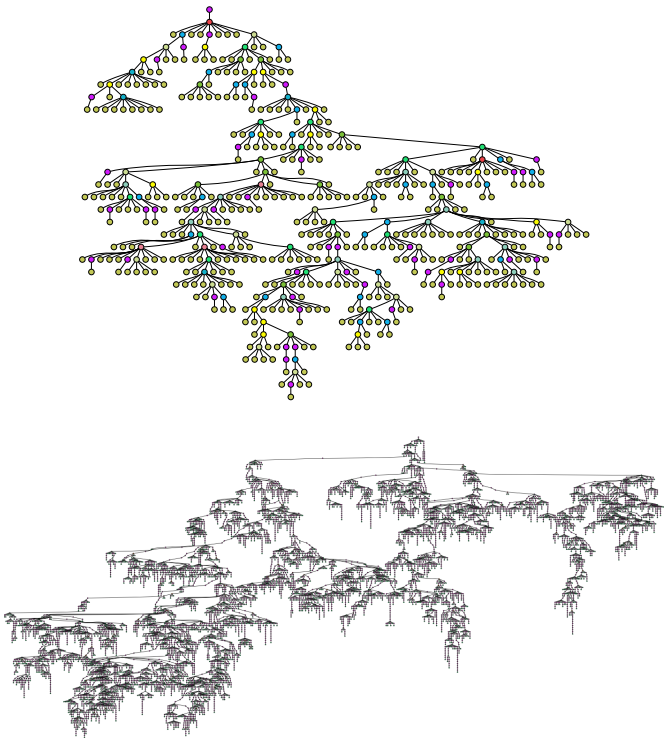


Figure 3: Two random plane trees with degrees in the set $D = \{0, \ldots, 9\}$. On the top, a tree of size in between 500 and 550; in the bottom, a tree of size in the interval $[10,000; 10,050]$.

Empirical frequencies for the bottom tree of Figure 3 and a simply-generated tree of size in between 10,000 and 10,050 with default node degree frequencies are included in Table 2.

We briefly remark that for this particular problem, Bodini, David and Marchal proposed a different, bit-optimal sampling procedure for random trees with given partition of node degrees [9].

4.3 Variable distribution in plain λ-terms. To exhibit the benefits of distorting the intrinsic distribution of various structural patterns in algebraic data types, we present an example specification defining so-called plain λ-terms with explicit control over the distribution of de Bruijn indices.

In their nameless representation due to de Bruijn [17] λ-terms are defined by the formal grammar $L ::= \lambda L \mid (LL) \mid D$ where $D = \{\underline{0}, \underline{1}, \underline{2}, \ldots\}$ is an infinite denumerable set of so-called indices (cf. [4, 29]). Assuming that we encode de Bruijn indices as a sequence of successors of zero (i.e. use a unary base representation), the class $\mathcal{L}$ of plain λ-terms can be specified as $\mathcal{L} = \mathcal{Z}\mathcal{L} + \mathcal{Z}\mathcal{L}^2 + \mathcal{D}$ where $\mathcal{D} = \mathcal{Z}\,\textsc{Seq}(\mathcal{Z})$. In order to control the distribution of de Bruijn indices we need a more explicit specification for de Bruijn indices. For instance:

$$\mathcal{D} = \mathcal{U}_0\mathcal{Z} + \mathcal{U}_1\mathcal{Z}^2 + \cdots + \mathcal{U}_k\mathcal{Z}^{k+1} + \mathcal{Z}^{k+2}\,\textsc{Seq}(\mathcal{Z})\,.$$

Here, we roll out the $k+1$ initial indices and assign distinct marking variables to each one of them, leaving the remainder sequence intact. In doing so, we are in a position to construct a sampler tuned to enforce a uniform distribution of 8% among all marked indices, i.e. indices $\underline{0}, \underline{1}, \ldots, \underline{8}$, distorting in effect their intrinsic geometric distribution.

Figure 4 illustrates two random λ-terms with such a new distribution of indices. For presentation purposes, each index in the left picture is given a distinct colour.

Empirical frequencies for the bottom term of Figure 4 and a plain λ-term of size in between 10,000 and 10,050 with default de Bruijn index frequencies are included in Table 3.

Let us note that algebraic data types, an essential conceptual ingredient of various functional programming languages such as Haskell or OCaml, and the random generation of their inhabitants satisfying additional structural or semantic properties is one of the central problems present in the field of property-based software testing (see, e.g. [19, 37]). In such an approach to software quality assurance, programmer-declared function invariants (so-called properties) are checked using random inputs, generated accordingly to some predetermined, though usually not rigorously controlled, distri-

Table 2: Empirical frequencies of the node degree distribution.

Node degree	0	1	2	3	4	5	6	7	8	9
Tuned frequency	- - -	- - -	1.00%	1.00%	1.00%	1.00%	1.00%	1.00%	1.00%	1.00%
Observed frequency	35.925%	56.168%	0.928%	0.898%	1.098%	0.818%	1.247%	0.938%	1.058%	0.918%
Default frequency	50.004%	24.952%	12.356%	6.322%	2.882%	1.984%	0.877%	0.378%	0.169%	0.069%

Table 3: Empirical frequencies (with respect to the term size) of index distribution.

Index	0	1	2	3	4	5	6	7	8
Tuned frequency	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%	8.00%
Observed frequency	7.50%	7.77%	8.00%	8.23%	8.04%	7.61%	8.53%	7.43%	9.08%
Default frequency	21.91%	12.51%	5.68%	2.31%	0.74%	0.17%	0.20%	0.07%	- - -

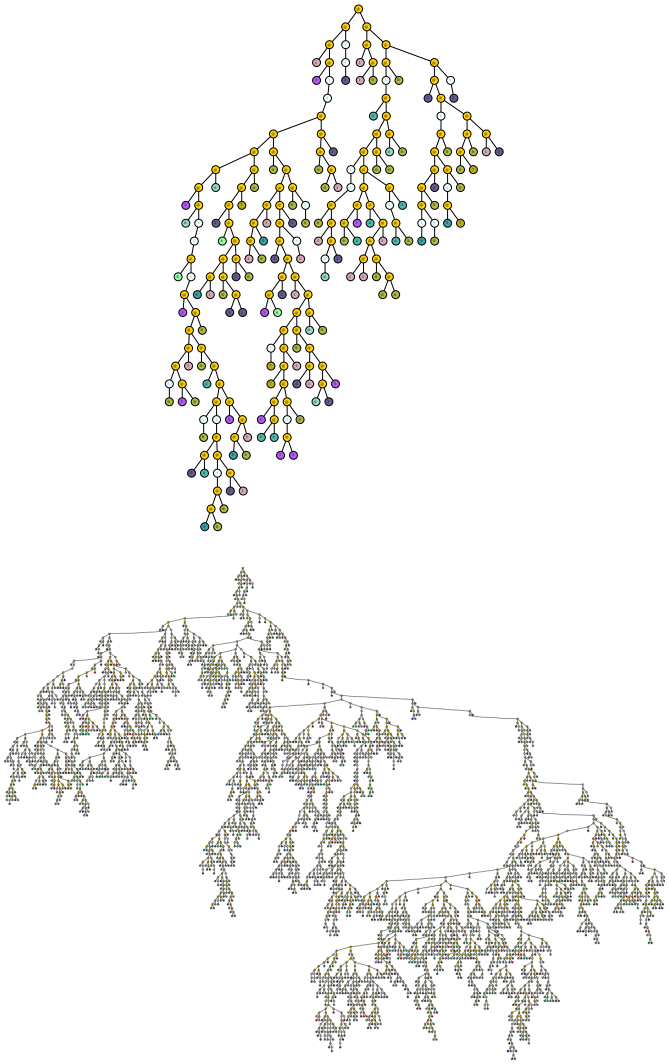


Figure 4: On the top, a random λ-term of size in the interval $[500; 550]$; below, a larger example of a random λ-term of size between $10,000$ and $10,050$.

bution. In this context, our techniques provide a novel and effective approach to generating random algebraic data types with fixed average frequencies of type constructors. In particular, using our methods it is possible to *boost* the intrinsic frequencies of certain desired subpatterns or *diminish* those which are unwanted.

4.4 Weighted partitions. Integer partitions are one of the most intensively studied objects in number theory, algebraic combinatorics and statistical physics. Hardy and Ramanujan obtained the famous asymptotics which has later been refined by Rademacher [26, Chapter VIII]. In his article [39], Vershik considers several combinatorial examples related to statistical mechanics and obtains the limit shape for a random integer partition of size n with $\alpha\sqrt{n}$ parts and summands bounded by $\theta\sqrt{n}$. Let us remark that Bernstein, Fahrbach, and Randall [7] have recently analysed the complexity of exact-size Boltzmann sampler for weighted partitions. In the model of ideal gas, there are several particles (bosons) which form a so-called assembly of particles. The overall energy of the system is the sum of the energies $\Lambda = \sum_{i=1}^{N} \lambda_i$ where λ_i denotes the energy of i-th particle. We assume that energies are positive integers. Depending on the energy level λ there are $j(\lambda)$ possible available states for each particle; the function $j(\lambda)$ depends on the physical model. Since all the particles are indistinguishable, the generating function $P(z)$ for the number of assemblies $p(\Lambda)$ with energy Λ takes the form

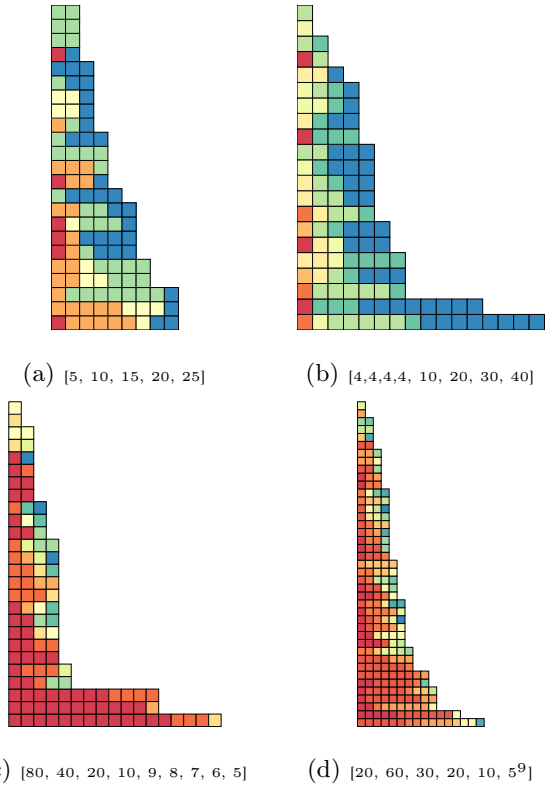
$$P(z) = \sum_{\Lambda=0}^{\infty} p(\Lambda) z^{\Lambda} = \prod_{\lambda>0} \frac{1}{(1-z^{\lambda})^{j(\lambda)}} \quad . \tag{4.12}$$

In the model of d-dimensional harmonic trap (also known as the Bose-Einstein condensation) according to [18, 31, 33] the number of states for a particle with energy λ is $\binom{d+\lambda-1}{\lambda}$ so that each state can be represented as a multiset with λ elements having d different colours.

Accordingly, an assembly is a multiset of particles (since they are bosons and hence indistinguishable) therefore the generating function for the number of assemblies takes the form

$$P(z) = \textsc{MSet}(\textsc{MSet}_{\geq 1}(\mathcal{Z}_1 + \cdots + \mathcal{Z}_d)) \ . \tag{4.13}$$

It is possible to control the expected frequencies of colours using our tuning procedure and sample resulting assemblies as Young tableaux. Each row corresponds to a particle whereas the colouring of the row displays the multiset of included colours, see Figure 5. We also generated weighted partitions of expected size 1000 (which are too large to display) with tuned frequencies of 5 colours, see Table 4.



(a) [5, 10, 15, 20, 25] (b) [4,4,4,4, 10, 20, 30, 40]

(c) [80, 40, 20, 10, 9, 8, 7, 6, 5] (d) [20, 60, 30, 20, 10, 5^9]

Figure 5: Young tableaux corresponding to Bose–Einstein condensates with expected numbers of different colours. Notation $[c_1, c_2, \ldots, c_k]$ provides the expected number c_j of the j-th colour, c_k^m is a shortcut for m occurrences of c_k.

Let us briefly explain our generation procedure. Boltzmann sampling for the outer MSET operator is described in Algorithm 2, Appendix C. The sampling of inner $\textsc{MSet}_{\geq 1}(\mathcal{Z}_1 + \ldots + \mathcal{Z}_d)$ is more delicate. The generating function for this multiset can be written as

$$\textsc{MSet}_{\geq 1}(z_1 + \cdots + z_d) = \prod_{i=1}^{d} \frac{1}{1 - z_i} - 1 \ . \tag{4.14}$$

Table 4: Empirical frequencies of colours observed in random partition.

Colour index	1	2	3	4	5	size
Tuned frequency	0.03	0.07	0.1	0.3	0.5	1000
	0.03	0.08	0.07	0.33	0.49	957
	0.03	0.06	0.09	0.28	0.54	1099
Observed frequency	0.03	0.08	0.09	0.34	0.46	992
	0.04	0.07	0.1	0.31	0.49	932
	0.04	0.09	0.1	0.25	0.52	1067

In order to correctly calculate the branching probabilities, we introduce slack variables $s_1, \ldots, s_d$ satisfying $(1 + s_i) = (1 - z_i)^{-1}$. Boltzmann samplers for the newly determined combinatorial classes $\Gamma\mathcal{S}_i$ are essentially Boltzmann samplers for $\textsc{Seq}_{\geq 1}(\mathcal{Z}_i)$. Let us note that after expanding brackets the expression becomes

$$\begin{aligned}\textsc{MSet}_{\geq 1}(z_1 + \cdots + z_d) &= (s_1 + \cdots + s_d) \\ &+ (s_1 s_2 + \cdots + s_{d-1} s_d) + \cdots + s_1 s_2 \ldots s_d.\end{aligned}$$

The total number of summands is $2^d - 1$ where each summand corresponds to choosing some subset of colours. Finally, let us explain how to precompute all the symmetric polynomials and efficiently handle the branching process in quadratic time using a dynamic programming approach. We can recursively define two arrays of real numbers $p_{k,j}$ and $q_{k,j}$ satisfying

$$\begin{cases} p_{1,j} = s_j, \quad j \in \{1, \ldots, d\}; \\ q_{k,d} = p_{k,d}, \quad k \in \{1, \ldots, d\}; \\ q_{k,j} = p_{k,j} + q_{k,j+1}, \quad j \in \{k, \ldots, d-1\}, \quad k \in \{1, \ldots, d\}; \\ p_{k,j} = s_{j-k+1} \cdot q_{k-1,j}, \quad j \in \{k, \ldots, d-1\}, \quad k \in \{2, \ldots d\}; \end{cases} \tag{4.15}$$

Arrays $(p_{k,j})_{j=k}^{d}$ contain the branching probabilities determining the next colour inside the k-th symmetric polynomial. Arrays $(q_{k,j})_{j=k}^{d}$ contain partial sums for the k-th symmetric polynomial and are required in intermediate steps. Numbers $q_{k,k}$ are equal to the total values of symmetric polynomials $(s_1 + \cdots + s_2), (s_1 s_2 + \cdots + s_{d-1} s_d), \ldots, s_1 s_2, \ldots, s_d$ and they define initial branching probabilities to choose the number of colours.

4.5 Prototype sampler generator. Consider the following example of an input file for Boltzmann Brain:

```
-- Motzkin trees
Motzkin = Leaf (3)
        | Unary Motzkin
        | Binary Motzkin Motzkin (2) [0.3].
```

Here, a `Motzkin` algebraic data type is defined. It consists of three constructors: a constant `Leaf` of weight three, a `Unary` constructor of weight one (default value if not explicitly annotated) and a constructor `Binary` of weight two together with an explicit tuning frequency of 30%. Such a definition corresponds to the combinatorial specification $\mathcal{M} = \mathcal{Z}^3 + \mathcal{Z}\mathcal{M} + \mathcal{U}\mathcal{Z}^2\mathcal{M}^2$ where the objective is to obtain the mean proportion of $\mathcal{U}\mathcal{Z}^2\mathcal{M}^2$ equal 30% of the total structure size. All the terms `Leaf`, `Unary`, `Motzkin`, `Binary` are user-defined keywords. Given such a specification on input, `bb` builds a corresponding singular Boltzmann sampler implemented in form of a self-contained Haskell module.

References

[1] A. Bacher, O. Bodini and A. Jacquot. "Exact-size sampling for Motzkin trees in linear time via Boltzmann samplers and holonomic specification". *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*. 2013, pp. 52–61.

[2] C. Banderier, O. Bodini, Y. Ponty and H. T. Bouzid. "On the diversity of pattern distributions in rational language". *Proceedings of the Ninth Workshop on Analytic Alg. and Combinatorics*. 2012, pp. 107–115.

[3] E. A. Bender and L. B. Richmond. "Central and local limit theorems applied to asymptotic enumeration II: Multivariate generating functions". *Journal of Combinatorial Theory, Series A* 34.3 (1983), pp. 255–265.

[4] M. Bendkowski, K. Grygiel, P. Lescanne and M. Zaionc. "Combinatorics of λ-terms: a natural approach". *Journal of Logic and Computation* (2017).

[5] F. Bergeron, G. Labelle and P. Leroux. *Combinatorial species and tree-like structures*. Vol. 67. Cambridge University Press, 1998.

[6] O. Bernardi and O. Giménez. "A linear algorithm for the random sampling from regular languages". *Algorithmica* 62.1 (2012), pp. 130–145.

[7] M. Bernstein, M. Fahrbach and D. Randall. "Analyzing Boltzmann Samplers for Bose-Einstein Condensates with Dirichlet Generating Functions". *arXiv preprint arXiv:1708.02266* (2017).

[8] P. Bhakta, B. Cousins, M. Fahrbach and D. Randall. "Approximately sampling elements with fixed rank in graded posets". *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms SODA*. 2017, pp. 1828–1838.

[9] O. Bodini, J. David and P. Marchal. "Random-bit optimal uniform sampling for rooted planar trees with given sequence of degrees and applications". *Conference on Algorithms and Discrete Applied Mathematics*. 2016, pp. 97–114.

[10] O. Bodini, M. Dien, X. Fontaine, A. Genitrini and H.-K. Hwang. "Increasing Diamonds". *Latin American Symposium on Theoretical Informatics*. 2016, pp. 207–219.

[11] O. Bodini, É. Fusy and C. Pivoteau. "Random sampling of plane partitions". *Combinatorics, Probability and Computing* 19.2 (2010), pp. 201–226.

[12] O. Bodini, A. Genitrini and N. Rolin. "Pointed versus singular Boltzmann samplers: a comparative analysis". *Pure Mathematics and Application* 25.2 (2015), pp. 115–131.

[13] O. Bodini, J. Lumbroso and N. Rolin. "Analytic samplers and the combinatorial rejection method". *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*. 2015, pp. 40–50.

[14] O. Bodini and Y. Ponty. "Multi-dimensional Boltzmann sampling of context-free languages". *21st International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms (AofA'10)*. Vol. AM. 2010.

[15] O. Bodini, O. Roussel and M. Soria. "Boltzmann samplers for first-order differential specifications". *Disc. App. Math.* 160.18 (2012), pp. 2563–2572.

[16] M. Bodirsky, É. Fusy, M. Kang and S. Vigerske. "Boltzmann samplers, Pólya theory, and cycle pointing". *SIAM J. Comp.* 40.3 (2011), pp. 721–769.

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

[18] K. C. Chase, A. Z. Mekjian and L. Zamick. "Canonical and microcanonical ensemble approaches to Bose-Einstein condensation: The thermodynamics of particles in harmonic traps". *The European Physical Journal B-Condensed Matter and Complex Systems* 8.2 (1999), pp. 281–285.

[19] K. Claessen and J. Hughes. "QuickCheck: a lightweight tool for random testing of Haskell programs". *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*. 2000, pp. 268–279.

[20] A. Denise and P. Zimmermann. "Uniform random generation of decomposable structures using floating-point arithmetic". *Theoretical Computer Science* 218.2 (1999), pp. 233–248.

[21] S. Diamond and S. Boyd. "CVXPY: A Python-embedded bodeling language for convex optimization". *J. Mach. Learn. Research* 17.83 (2016), pp. 1–5.

[22] A. Domahidi, E. Chu and S. Boyd. "ECOS: An SOCP solver for embedded systems". *Control Conference (ECC), 2013 European*. IEEE. 2013, pp. 3071–3076.

[23] M. Drmota. "Systems of functional equations". *Rand. Struct. & Alg.* 10.1-2 (1997), pp. 103–124.

[24] P. Duchon, P. Flajolet, G. Louchard and G. Schaeffer. "Boltzmann samplers for the random generation of combinatorial structures". *Combinatorics, Probability & Computing* 13.4-5 (2004), pp. 577–625.

[25] P. Flajolet, É. Fusy and C. Pivoteau. "Boltzmann sampling of unlabelled structures". *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*. 2007, pp. 201–211.

[26] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN: 978-0-521-89806-5.

[27] P. Flajolet, P. Zimmermann and B. V. Cutsem. "A calculus for the random generation of labelled combinatorial structures". *Theoretical Computer Science* 132.1 (1994), pp. 1–35.

[28] É. Fusy. "Quadratic exact size and linear approximate size random generation of planar graphs". *Discr. Math. & Theor. Comp. Sc.* 2005, pp. 125–138.

[29] B. Gittenberger and Z. Gołębiewski. "On the number of lambda terms with prescribed size of their de Bruijn representation". *33rd Symposium on Theoretical Aspects of Computer Science, STACS*. 2016, 40:1–40:13.

[30] M. Grant, S. Boyd and Y. Ye. "Disciplined convex programming". *Global optimization*. 2006, pp. 155–210.

[31] T. Haugset, H. Haugerud and J. O. Andersen. "Bose-Einstein condensation in anisotropic harmonic traps". *Physical Review A* 55.4 (1997), p. 2922.

[32] J. Kemeny and J. Snell. *Finite Markov chains*. Springer, 1960.

[33] J. Lucietti and M. Rangamani. "Asymptotic counting of BPS operators in superconformal field theories". *J. Math. Phys.* 49.8 (2008), p. 082301.

[34] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[35] A. Nijenhuis and H. S. Wilf. *Combinatorial Algorithms*. 2nd ed. Academic Press, 1978.

[36] B. O'Donoghue, E. Chu, N. Parikh and S. Boyd. "Conic optimization via operator splitting and homogeneous self-dual embedding". *J. Opt. Th. & App.* 169.3 (2016), pp. 1042–1068.

[37] M. H. Pałka. "Random structured test data generation for black-box testing". PhD thesis. Chalmers University of Technology, 2012.

[38] C. Pivoteau, B. Salvy and M. Soria. "Algorithms for combinatorial structures: well-founded systems and Newton iterations". *J. Comb. Th.* A 119.8 (2012), pp. 1711–1773.

[39] A. M. Vershik. "Statistical mechanics of combinatorial partitions, and their limit shapes". *Functional Analysis and Its Applications* 30.2 (1996), pp. 90–105.

A Convex optimisation: proofs and algorithms

Until now, we have left several important questions unanswered. Firstly, what is the required precision ε for multiparametric tuning? Secondly, what is its precise computational complexity? In order to determine the time and space complexity of our tuning procedure we need to explain some technical decisions regarding the choice of particular optimisation methods. In this section we prove that the optimisation procedures described in § 3 give the correct solution to the tuning problem.

A.1 Proofs of the theorems.

Proof of Theorem 3.1. Let the following *nabla-notation* denote the vector of derivatives (so-called gradient vector) with respect to the variable vector $\boldsymbol{z} = (z_1, \ldots, z_k)$:

$$\nabla_{\boldsymbol{z}} f(\boldsymbol{z}) = \left(\frac{\partial}{\partial z_1} f(\boldsymbol{z}), \ldots, \frac{\partial}{\partial z_k} f(\boldsymbol{z}) \right)^{\top} . \tag{A.1}$$

We start with noticing that tuning the expected number of atom occurrences is equivalent to solving the equation $\nabla_{\boldsymbol{\xi}} \log C(e^{\boldsymbol{\xi}}) = \boldsymbol{\nu}$, see Proposition 2.1. Here, the right-hand side is equal to $\nabla_{\boldsymbol{\xi}}(\boldsymbol{\nu}^{\top}\boldsymbol{\xi})$ so tuning is further equivalent to $\nabla_{\boldsymbol{\xi}} \left(\log C(e^{\boldsymbol{\xi}}) - \boldsymbol{\nu}^{\top}\boldsymbol{\xi} \right) = 0$. The function under the gradient is convex as it is a sum of a convex and linear function. In consequence, the problem of minimising the function is equivalent to finding the root of the derivative

$$\log C(e^{\boldsymbol{\xi}}) - \boldsymbol{\nu}^{\top}\boldsymbol{\xi} \to \min_{\boldsymbol{\xi}} . \tag{A.2}$$

Definition A.1. *(Feasible points) In the optimisation problem*

$$\begin{cases} f(\boldsymbol{z}) \to \min, \\ \boldsymbol{z} \in \Omega \end{cases} \tag{A.3}$$

a point $\boldsymbol{z}$ is called feasible if it belongs to the set Ω.

Proof of Theorem 3.2. Let $\boldsymbol{N} = (N_1, \ldots, N_k)$ be the vector of atom occurrences of each type. Consider the vector $\boldsymbol{z}^*$ such that $\mathbb{E}_{\boldsymbol{z}^*}(\boldsymbol{N}) = \boldsymbol{\nu}$. Let $\boldsymbol{c}$ denote the logarithms of the values of generating functions at point $\boldsymbol{z}^* = e^{\boldsymbol{\xi}^*}$. Clearly, in such a case all inequalities in (3.8) become equalities and the point $(\boldsymbol{c}, \boldsymbol{\xi}^*)$ is feasible.

Let us show that if the point $(\boldsymbol{c}, \boldsymbol{\xi})$ is optimal, then all the inequalities in (3.8) become equalities. Firstly, suppose that the inequality

$$c_1 \geq \log \Phi_1(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}}) \tag{A.4}$$

does not turn to an equality. Certainly, there is a *gap* and the value c_1 can be decreased. In doing so, the target function value is decreased as well. Hence, the point $(\boldsymbol{c}, \boldsymbol{\xi})$ cannot be optimal.

Now, suppose that the initial inequality does turn to equality, however $c_k > \log \Phi_k(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}})$ for some $k \neq 1$. Since the system is strongly connected, there exists a path $P = c_1 \to c_2 \to \cdots \to c_k$ (indices are chosen without loss of generality) in the corresponding dependency graph. Note that for pairs of consecutive variables (c_i, c_{i+1}) in P, the function $\log \Phi_i(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}})$ is strictly monotonic in c_{i+1} (as its monotonic and references c_{i+1}). In such a case we can decrease c_{i+1} so to assure that $c_i > \log \Phi_i(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}})$ while the point $(\boldsymbol{c}, \boldsymbol{\xi})$ remains feasible. Decreasing $c_{i+1}, c_i, \ldots, c_1$ in order, we finally arrive at a feasible point with a decreased target function value. In consequence, $(\boldsymbol{c}, \boldsymbol{\xi})$ could not have been optimal to begin with.

So, eventually, the optimisation problem reduces to minimising the expression subject to the system of equations $\boldsymbol{c} = \log \boldsymbol{\Phi}(e^{\boldsymbol{c}}, e^{\boldsymbol{\xi}})$ or, equivalently, $\boldsymbol{C}(\boldsymbol{z}) = \boldsymbol{\Phi}(\boldsymbol{C}(\boldsymbol{z}), \boldsymbol{z})$ and can be therefore further reduced to Theorem 3.1.

Proof of Theorem 3.3. By similar reasoning as in the previous proof, we can show that the maximum is attained when all the inequalities turn to equalities. Indeed, suppose that at least one inequality is strict, say $c_j > \log \Phi_j(e^{\boldsymbol{c}}, e^{\xi}, e^{\boldsymbol{\eta}})$. The value c_j can be slightly decreased by ε by choosing a sufficiently small distortion ε to turn all the equalities containing c_j in the right-hand side $\log \Phi_i(e^{\boldsymbol{c}}, e^{\xi}, e^{\boldsymbol{\eta}})$ to strict inequalities, because the right-hand sides of each of the inequalities are monotonic functions with respect to c_j. This procedure can be repeated until all the equalities turn into inequalities. Finally, we slightly decrease the value ξ to increase the target function while still staying inside the feasible set, because of the monotonicity of the right-hand side with respect to ξ.

Let us fix $\boldsymbol{u} = e^{\boldsymbol{\eta}}$. For rational and algebraic grammars, within the Drmota–Lalley–Woods framework, see for instance [23], the corresponding generating function singular approximation takes the form

$$C(z, \boldsymbol{u}) \sim a_0(\boldsymbol{u}) - b_0(\boldsymbol{u}) \left(1 - \frac{z}{\rho(\boldsymbol{u})}\right)^t \quad . \tag{A.5}$$

If $t < 0$, then the asymptotically dominant term becomes $-b_0 \left(1 - \frac{z}{\rho(\boldsymbol{u})}\right)^t$. In this case, tuning the target expected frequencies corresponds to solving the following equation as $z \to \rho(u)$:

$$\operatorname{diag}(\boldsymbol{u}) \frac{[z^n] \nabla_{\boldsymbol{u}} C(z, \boldsymbol{u})}{[z^n] C(z, \boldsymbol{u})} = n\boldsymbol{\alpha} \quad . \tag{A.6}$$

Let us substitute the asymptotic expansion (A.5) into (A.6) to track how $\boldsymbol{u}$ depends on $\boldsymbol{\alpha}$:

$$\operatorname{diag}(\boldsymbol{u}) \frac{[z^n] t b_0(\boldsymbol{u}) \left(1 - \frac{z}{\rho(\boldsymbol{u})}\right)^{t-1} z \frac{\nabla_{\boldsymbol{u}} \rho(\boldsymbol{u})}{\rho^2(\boldsymbol{u})}}{[z^n] b_0(\boldsymbol{u}) \left(1 - \frac{z}{\rho(\boldsymbol{u})}\right)^t} = -n\boldsymbol{\alpha} \quad . \tag{A.7}$$

Only dominant terms are accounted for. Then, by the binomial theorem

$$\operatorname{diag}(\boldsymbol{u}) b_0(\boldsymbol{u}) \frac{t}{n} \binom{t-1}{n} \frac{z \nabla_{\boldsymbol{u}} \rho(\boldsymbol{u})}{\rho^2(\boldsymbol{u})} b_0(\boldsymbol{u})^{-1} \binom{t}{n}^{-1} = -\boldsymbol{\alpha} \quad , \tag{A.8}$$

With $z = \rho(\boldsymbol{u})$, as $n \to \infty$, we obtain after cancellations

$$\operatorname{diag}(\boldsymbol{u}) \frac{\nabla_{\boldsymbol{u}} \rho(\boldsymbol{u})}{\rho(\boldsymbol{u})} = -\boldsymbol{\alpha} \tag{A.9}$$

which can be rewritten as

$$\nabla_{\boldsymbol{\eta}} \log \rho(e^{\boldsymbol{\eta}}) = -\boldsymbol{\alpha} \quad . \tag{A.10}$$

Passing to exponential variables (A.10) becomes

$$\nabla_{\boldsymbol{\eta}} (\xi(\boldsymbol{\eta}) + \boldsymbol{\alpha}^\top \boldsymbol{\eta}) = 0 \quad . \tag{A.11}$$

As we already discovered, the dependence $\xi(\boldsymbol{\eta})$ is given by the system of equations because the maximum is achieved only when all inequalities turn to equations. That is, tuning the singular sampler is equivalent to maximising $\xi + \boldsymbol{\alpha}^\top \boldsymbol{\eta}$ over the set of feasible points.

REMARK A.1. *For ordinary and singular samplers, the corresponding feasible set remains the same; what differs is the optimised target function. Singular samplers correspond to imposing an infinite target size. In practice, however, the required singularity is almost never known* exactly *but rather calculated up to some feasible finite precision. The tuned structure size is therefore enormously large, but still, nevertheless, finite. In this context, singular samplers provide a natural* limiting *understanding of the tuning phenomenon and as such, there are several possible ways of proving Theorem 3.3.*

Figure 6 illustrates the feasible set for the class of binary trees and its transition after applying the log-exp transform, turning the set into a convex collection of feasible points. In both figures, the singular point is the rightmost point on the plot. Ordinary sampler tuning corresponds to finding the tangent line which touches the set, given the angle between the line and the abscissa axis.

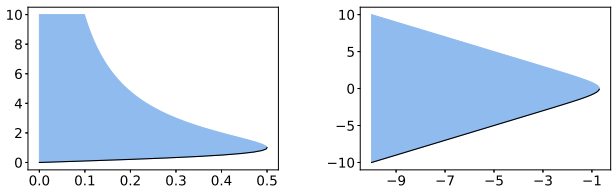


Figure 6: Binary trees $B \geq z + zB^2$ and log-exp transform of the feasible set. The black curve denotes the principal branch of the generating function $B(z)$ corresponding to the class of binary trees.

A.2 Disciplined convex programming and optimisation algorithms. In the subsequent proofs, we present the framework of Disciplined Convex Programming (DCP in short) and show how to incorporate elementary combinatorial constructions into this framework. Nesterov and Nemirovskii [34] developed a seminal polynomial-time optimisation algorithm for convex programming which involves the construction of certain *self-concordant barriers* related to the feasible set of points. The arithmetic complexity of their method is

$$O\left(\log \frac{1}{\varepsilon}\sqrt{\vartheta}\mathcal{N}\right) \tag{A.12}$$

where $\mathcal{N}$ is the arithmetic complexity of a single Newton iteration step, ϑ is the so-called constant of self-concordeness of the barriers and ε is the target precision. Before we go into each of the terms, we mention that for sparse matrix representations, it is possible to accelerate the speed of the Newton iteration, i.e. the step of solving the system of linear equations

$$A\boldsymbol{x} = \boldsymbol{b} \quad \text{where} \quad A \in \mathbb{R}^{m\times m} \text{ and } \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^m \tag{A.13}$$

from $O(m^3)$ to $O(m^2)$.

Unfortunately, for general convex programming problems there is no constructive general-purpose barrier construction method, merely existence proofs. Fortunately, Grant, Boyd, and Ye [30] developed the DCP framework which automatically constructs suitable barriers for the user. Moreover, DCP also automatically provides the starting feasible point which is itself a non-trivial problem in general. As its price, the user is obliged to provide a certificate that the constructed problem is convex, i.e. express all convex functions in terms of a predefined set of elementary convex functions.

In our implementation, we rely on two particular solvers, a second-order (i.e. using second-order derivatives) Embedded Conic Solver (`ECOS`) [22] and recently developed first-order (i.e. using only first-order derivatives) Splitting Conic Solver (`SCS`) algorithm [36]. The conversion of the DCP problem into its standard form is done using `cvxpy`, a Python-embedded modelling language for disciplined convex programming [21].

Proof of Theorem 3.4. We start with showing that the tuning procedure can be effectively represented in the framework of DCP.

In our case, every inequality takes the form

$$c_i \geq \log\left(\sum_{i=1}^{m} e^{\ell_i(\boldsymbol{c},\boldsymbol{z})}\right) \tag{A.14}$$

where $\ell_i(\boldsymbol{c}, \boldsymbol{z})$ are some linear functions. Appreciably, the log-sum-exp function belongs to the set of admissible constructions of the DCP framework.

Converting the tuning problem into DCP involves creating some slack variables. For each product of two terms $X \times Y$ we create slack variables for X and Y which are represented by the variables ξ and η in the log-exp realm as

$$e^{\xi} = X \qquad \text{and} \qquad e^{\eta} = Y \ . \tag{A.15}$$

Next, we replace $X \times Y$ by $e^{\xi+\eta}$ as composition of addition and exponentiation is a valid DCP program. Since every expression in systems corresponding to considered combinatorial classes is a sum of products, the corresponding restriction (A.14) is converted to a valid DCP constraint using the elementary log-sum-exp function.

The sequence operator $\textsc{Seq}(\mathcal{A})$ which converts a generating function $A(\boldsymbol{z})$ into $(1 - A(\boldsymbol{z}))^{-1}$ is *unfolded* by adding an extra equation into the system in form of

$$D := \textsc{Seq}\, A(\boldsymbol{z}) \quad \text{whereas} \quad D = 1 + AD\,. \tag{A.16}$$

Two additional constructions, MSET and CYC are treated in a similar way. Infinite sums are replaced by finite ones because the difference in the distribution of truncated variables is a negative exponent in the truncation length, and hence negligible.

Using the DCP method, the constant of self-concordness of the barriers is equal to $\vartheta = O(L)$, where L is the length of the problem description. This includes the number of combinatorial classes, number of atoms for which we control the frequency and the sum of lengths of descriptions of each specification, i.e. their overall length. In total, the complexity of optimisation can be therefore crudely estimated as

$$O\left(L^{3.5} \log \frac{1}{\varepsilon}\right) . \tag{A.17}$$

Certainly, the complexity of tuning is polynomial, as stated. We emphasise that in practice, using sparse matrices this can be further reduced to $O(L^{2.5} \log(1/\varepsilon))$.

REMARK A.2. *Weighted partitions, one of our previous applications, involves a multiset operator* $\mathrm{MSET}_{\geq 1}(\mathcal{Z}_1 + \cdots + \mathcal{Z}_d)$ *which generalises to* $\mathrm{SEQ}(\mathcal{C}_1)\,\mathrm{SEQ}(\mathcal{C}_2)\cdots\mathrm{SEQ}(\mathcal{C}_d) - 1$ *and does not immediately fall into the category of admissible operators as it involves subtraction. This is a general* weak point *of Boltzmann sampling involving usually a huge amount of rejections, in consequence substantially slowing down the generation process. Moreover, it also disables our convex optimisation tuning procedure because the constructions involving the minus sign cease to be convex and therefore do not fit the DCP framework.*

We present the following change of variables for this operator, involving a quadratic number of slack variables. The $\mathrm{SEQ}(\mathcal{C}_i)$ *operator yielding the generating function* $(1 - C_i(\boldsymbol{z}))^{-1}$ *is replaced by* $(1 + \mathcal{S}_i)$ *where* $\mathcal{S}_i$ *satisfies*

$$\mathcal{S}_i = \mathcal{C}_i + \mathcal{S}_i\mathcal{C}_i \ . \tag{A.18}$$

Next, we expand all of the brackets in the product $\prod_{i=1}^{d}(1+\mathcal{S}_i)-1$. *Consequently, we define the following arrays* $\mathcal{P}_{i,j}$ *and* $\mathcal{Q}_{i,j}$*:*

(A.19)
$$\begin{cases} \mathcal{P}_{1,j} = \mathcal{C}_j, \quad j \in \{1,\ldots,d\} \\ \mathcal{Q}_{k,d} = \mathcal{P}_{k,d}, \quad k \in \{1,\ldots,d\} \\ \mathcal{Q}_{k,j} = \mathcal{P}_{k,j} + \mathcal{Q}_{k,j+1}, \quad j \in \{k,\ldots,d-1\},\ k \in \{1,\ldots,d\} \\ \mathcal{P}_{k,j} = \mathcal{C}_{j-k+1} \cdot \mathcal{Q}_{k-1,j}, \quad j \in \{k,\ldots,d-1\},\ k \in \{2,\ldots,d\} \ . \end{cases}$$

Semantically, as in § 4.4, $(\mathcal{P}_{i,j})_{j=k}^{d}$ *and* $(\mathcal{Q}_{i,j})_{j=k}^{d}$ *denote the summands inside symmetric polynomials*

(A.20)
$$\begin{cases} \mathcal{Q}_{1,1} = \mathcal{S}_1 + \mathcal{S}_2 + \ldots + \mathcal{S}_d \ , \\ \mathcal{Q}_{2,2} = \mathcal{S}_1(\mathcal{S}_2 + \ldots + \mathcal{S}_d) + \mathcal{S}_2(\mathcal{S}_3 + \ldots + \mathcal{S}_d) + \ldots + \mathcal{S}_{d-1}\mathcal{S}_d \ , \\ \mathcal{Q}_{3,3} = \mathcal{S}_1(\mathcal{S}_2\mathcal{S}_3 + \ldots + \mathcal{S}_{d-1}\mathcal{S}_d) + \ldots + \mathcal{S}_{d-2}\mathcal{S}_{d-1}\mathcal{S}_d \end{cases}$$

and the auxiliary partial sums used to recompute the consequent expressions, respectively. So for instance when $d = 5$ *we obtain*

(A.21)
$$\mathcal{P} = \begin{pmatrix} \mathcal{S}_1 & \mathcal{S}_2 & \mathcal{S}_3 & \mathcal{S}_4 & \mathcal{S}_5 \\ 0 & \mathcal{S}_1(\mathcal{S}_2 + \ldots + \mathcal{S}_5) & \ldots & \mathcal{S}_3(\mathcal{S}_4 + \mathcal{S}_5) & \mathcal{S}_4\mathcal{S}_5 \\ 0 & 0 & \mathcal{S}_1(\ldots) & \mathcal{S}_2(\mathcal{S}_3(\mathcal{S}_4 + \mathcal{S}_5) + \mathcal{S}_4\mathcal{S}_5) & \mathcal{S}_3\mathcal{S}_4\mathcal{S}_5 \\ 0 & 0 & 0 & \mathcal{S}_1(\ldots) & \mathcal{S}_2\ldots\mathcal{S}_5 \\ 0 & 0 & 0 & 0 & \mathcal{S}_1\ldots\mathcal{S}_5 \end{pmatrix}.$$

The union of classes in each row gives corresponding symmetric polynomial $\mathcal{Q}_{k,k}$*, and the partial sum of elements in the row gives the elements of* $\mathcal{Q}$.

Finally, the expression $\prod_{i=1}^{d}(1+\mathcal{S}_i)-1$ *is replaced by the sum of elementary symmetric polynomials* $\mathcal{Q}_{1,1} + \mathcal{Q}_{2,2} + \cdots + \mathcal{Q}_{d,d}$ *where we have (combinatorially)*

$$\mathcal{Q}_{j,j} = \sum_{1 \leq i_1 < \ldots < i_j \leq d} \mathcal{S}_{i_1}\cdots\mathcal{S}_{i_j} \ . \tag{A.22}$$

We emphasise that the last sum is not meant to be implemented in practice in a naïve way as it would take an exponential amount of time to be computed.

A.3 Tuning precision. In this section, we only work with algebraic systems that meet the certain regularity conditions from Drmota–Lalley–Woods Theorem [23].

PROPOSITION A.1. *Consider a multiparametric combinatorial specification*

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\Phi}(\boldsymbol{\mathcal{Y}}, \mathcal{Z}, \boldsymbol{\mathcal{U}}) \tag{A.23}$$

whose corresponding system of equations is either rational or algebraic. Suppose that we sample objects from the class $\mathcal{F} = \mathcal{Y}_1$ *with target expected sizes* $(n, \nu_1 n, \ldots, \nu_d n)$*, where* ν_i *are constants,* $n \to \infty$*. Let* $F(z, \boldsymbol{u})$ *be the multivariate generating function corresponding to the class* $\mathcal{F}$*, and let* $(z^*, \boldsymbol{u}^*)$ *be the target tuning vector. Then, there exists* $\varepsilon = \Theta(1/Poly(n))$ *such that the points* $(z, \boldsymbol{u})$ *from the* ε*-ball centered at* $(z^*, \boldsymbol{u}^*)$ *intersected with the set of feasible points*

$$\Big\{(z,\boldsymbol{u}) \in \mathbb{R}^{1+d} \mid \boldsymbol{Y}(z,\boldsymbol{u}) \geq \boldsymbol{\Phi}(\boldsymbol{Y}(z,\boldsymbol{u}), z, \boldsymbol{u}), \\ \|(z^* - z, \boldsymbol{u}^* - \boldsymbol{u})\| \leq \varepsilon\Big\}$$

yield expectations within $O(1)$ *of target expectations:*

$$\begin{aligned} zF'_z(z,\boldsymbol{u})/F(z,\boldsymbol{u}) &= n + O(1) \ , \\ u_i F'_{u_i}(z,\boldsymbol{u})/F(z,\boldsymbol{u}) &= \nu_i n + O(1),\ i \in \{1,\ldots,d\} \ . \end{aligned}$$

Proof. Let us show that z^*, as a function of n, satisfies

$$\begin{cases} z^*(n) \sim \rho(1 - \alpha/n), & \mathcal{F} \text{ is rational;} \\ z^*(n) \sim \rho(1 - C/n^2), & \mathcal{F} \text{ is algebraic.} \end{cases} \tag{A.24}$$

Here, α is a positive integer depending on the rational system, C is a generic constant. We also note that the same asymptotics is valid for each coordinate of the vector $\boldsymbol{u}^*(n)$, up to multiplicative constants depending on ν_i and the values of α and C.

For rational systems, there exist analytic functions $\beta(z, \boldsymbol{u})$, $\rho(\boldsymbol{u})$ and a positive integer α such that

$$F(z,\boldsymbol{u}) \sim \beta(z,\boldsymbol{u})(1 - z/\rho(\boldsymbol{u}))^{-\alpha} \ , \quad z \to \rho(\boldsymbol{u}) \ . \tag{A.25}$$

After substituting the asymptotic expansion (A.25) into (2.6), we obtain the first part of (A.24).

For algebraic systems, according to Drmota–Lalley–Woods Theorem [23], there exist analytic functions $\alpha(z, \boldsymbol{u}), \beta(z, \boldsymbol{u}), \rho(\boldsymbol{u})$ such that as $z \to \rho(\boldsymbol{u})$,

$$F(z,\boldsymbol{u}) \sim \alpha(z,\boldsymbol{u}) - \beta(z,\boldsymbol{u})(1 - z/\rho(\boldsymbol{u}))^{1/2} \ . \tag{A.26}$$

Again, substituting this asymptotic expansion into (2.6), we obtain

$$z^*(n)\frac{\beta}{2\rho\alpha}\left(1-\frac{z^*(n)}{\rho}\right)^{-1/2} \sim n \ . \tag{A.27}$$

Taking into account that $z^*(n) = \rho + o(1)$, this implies the second part of (A.24). Similarly, this can be applied to each coordinate of $\boldsymbol{u}$, not only to z.

Let us handle the tuning precision. We use the mean value theorem to bound ε. Let $m = n + O(1)$. Then,

$$\begin{aligned}\varepsilon^2 &\geq \|(z^*(n) - z^*(m), \boldsymbol{u}^*(n) - \boldsymbol{u}^*(m))\|^2 \\ &= (z^*(n) - z^*(m)^2 + \sum_{i=1}^{d}(u_i^*(n) - u_i^*(m))^2 \ .\end{aligned}$$

By the mean value theorem, there exist numbers $(n_i')_{i=0}^d$ from the interval $[n, m]$ such that

$$\begin{aligned}z^*(n) - z^*(m) &= (n-m)\frac{dz^*}{dn}(n_0') \ , \\ u_i^*(n) - u_i^*(m) &= (n-m)\frac{du_i^*}{dn}(n_i'),\ i \in \{1,\dots,d\} \ .\end{aligned}$$

Thus, as $n - m = O(1)$, we obtain

$$\varepsilon^2 \geq O(1)\left[\left(\frac{dz^*}{dn}(n_0')\right)^2 + \sum_{i=1}^{d}\left(\frac{du_i^*}{dn}(n_i')\right)^2\right] \ .$$

Since $n_i' = n + O(1)$, after substituting (A.24) and expressing the derivatives, we obtain the bound $\varepsilon = O(n^{-2})$ for rational grammars and $\varepsilon = O(n^{-3})$ for algebraic specifications.

Remark A.3. *If one uses the* anticipated rejection *principle for sampling the objects of approximate size $n+O(1)$, in effect rejecting objects smaller than $n-O(1)$ and "killing" the generation of objects whose size exceeds $n + O(1)$, it is possible to have a more relaxed bound $\varepsilon = O(n^{-2})$ for the case of algebraic specifications. Even though the expected size of generated objects will be smaller than n, so that we will need a large number of restarts, the total amount of generated atoms will be nevertheless linear in n. We refer to [12, Theorem 4.1] for further discussion.*

Remark A.4. *Under an extra frequency rejection (independently of the structure size)* it is not possible *to get rid of the assumption of strong connectivity and get a general estimate on the complexity of rejection-based sampling for arbitrary combinatorial specifications. Let us recall that Banderier, Bodini, Ponty and Bouzid give combinatorial classes with non-continuous parameter distributions [2]. For instance, consider the combinatorial class*

$$\mathcal{F} = \textsc{Seq}(\mathcal{Z}^3)\,\textsc{Seq}(\mathcal{U}\mathcal{Z}^3) + \textsc{Seq}(\mathcal{U}^2\mathcal{Z}^3)\,\textsc{Seq}(\mathcal{U}^3\mathcal{Z}^3) \tag{A.28}$$

in which all the structures have parameter frequencies in the intervals $(0, \frac{1}{3})$ and $(\frac{2}{3}, 1)$. Certainly, tuning the sampler for a target frequency inside the interval $(\frac{1}{3}, \frac{2}{3})$ yields a rejection sampler which never stops as there is no structures of demanded frequency.

For this reason we restrict our attention on two important subclasses of combinatorial specifications, i.e. strongly connected rational and algebraic languages. Due to Bender and Richmond [3] both classes follow a multivariate Gaussian law with linear expectation and standard deviation. In consequence, corresponding multiparametric Boltzmann samplers work in linear time if we accept a linear tolerance for the size $[(1-\epsilon)n, (1+\epsilon)n]$ and a square root tolerance for the parameters $[f - \kappa/\sqrt{n}, f + \kappa/\sqrt{n}]$.

B Samplers for rational grammars

Recall that a strongly connected rational grammar

$$\boldsymbol{F} = \boldsymbol{\Phi}(\boldsymbol{F}, \boldsymbol{z}) \tag{B.29}$$

is a specification corresponding to a rational language whose dependency graph is strongly connected. State and transitions of the associated automaton correspond to classes $\boldsymbol{F} = (F_1, \dots, F_m)$ and to appropriate monomials in the system (B.29), respectively.

For rational samplers, we decide to implement the strategy of *interruptible sampling*, introduced in [1] as the so-called *Hand of God* principle. The idea of anticipated rejection is also discussed in [12]. We start with fixing two distinguished states of the automaton, a starting one and a final (terminal) one. The starting and the terminal states may coincide. Next, we construct a tuned variant of the corresponding singular sampler. Specifically, tune it with arbitrarily high, yet still feasible precision. In essence, it is enough to tune to expected quantities exceeding the target ones. While tuning, we add a constraint $\|\boldsymbol{v}\| \leq M$ where $\boldsymbol{v}$ contains all the variables $\boldsymbol{F}$ and $\boldsymbol{z}$, and M is a logarithm of a large number, say $M = 40$. This constraint is required because otherwise the value of associated generating functions tends to infinity as $\boldsymbol{z}$ approaches the singular point. Moreover, by doing so we will compute branching probabilities with an error no more than $O(e^{-M})$. Under these conditions, the resulting sampler is unlikely to stop with output size less than the target one. Finally, we run the sampler from its initial state and continue sampling until the target structure size is attained. From that moment on, we wait for the

sampler to naturally reach its final state at which point the process is interrupted. In the following proposition we show that such a sampling procedure is actually an efficient generation scheme.

PROPOSITION B.1. *Let n be the target size of an interruptible sampler $\Gamma\mathcal{S}$ associated with a strongly connected rational system $\mathcal{S}$. Then, the following assertions hold:*

1. *structures are sampled from a uniform, conditioned on the (composition) size, distribution;*
2. *the size of the generated structures is $n + O(1)$ in probability where the constant error term depends solely on $\mathcal{S}$.*

Proof. Let ω be a structure generated by the interruptible sampler $\Gamma\mathcal{S}$. Assume w.l.o.g. that $\mathcal{S} = (S_1, \ldots, S_m)$ and moreover S_1 and S_m correspond to the associated automaton's starting and final state, respectively. We split the proof into two parts.

Firstly, let us focus on the uniformity (1). We show that conditioned on the vector of quantities $\boldsymbol{n}$, the probability of a structure ω with given number of atomic classes is proportional to $\boldsymbol{z}^{\boldsymbol{n}}$. According to the underlying Boltzmann model, each transition $S_i \to S_j$ taken by $\Gamma\mathcal{S}$ happens with probability

$$\mathbb{P}_{S_i \to S_j} = \boldsymbol{z}^{\Delta \boldsymbol{n}} \frac{S_j(\boldsymbol{z})}{S_i(\boldsymbol{z})} \tag{B.30}$$

where $\Delta\boldsymbol{n}$ denotes the change in the size of ω following transition $S_i \to S_j$.

Note however that while we trace the interruptible sampler generating ω, the ratios of generating functions in (B.30) cancel out (with the exception of the final $S_m(\boldsymbol{z})$). In consequence, the probability $\mathbb{P}_\omega$ that $\Gamma\mathcal{S}$ generated the structure ω becomes

$$\mathbb{P}_\omega = \boldsymbol{z}^{\sum \Delta \boldsymbol{n}} S_m(\boldsymbol{z}) = \boldsymbol{z}^{\boldsymbol{n}} S_m(\boldsymbol{z}) \tag{B.31}$$

where the latter equality follows from the fact that the sum $\sum \Delta\boldsymbol{n}$ of the increments in size is equal to the final size $\boldsymbol{n}$. And so, if we condition on the composite size, i.e. the vector of numbers of atoms, the distribution is indeed uniform.

Let us turn to assertion (2). Once the sampler passes the target size, it becomes a Markov chain with a single absorbing state S_m. The chain is irreducible, as the associated system is strongly connected, whereas all of the states $S_1, \ldots, S_{m-1}$ are not absorbing. Moreover, we can assume that once the target size is reached, the sampler starts a random walk in state S_i where $i \neq m$ as otherwise our claim holds trivially.

In consequence, the expected excess outcome size is proportional to the expected absorption time starting in the transient state S_i. This time, however, is known to be finite, see [32, Chapter III]. In conclusion, the expected outcome excess size is necessarily finite. An application of Markov's inequality finishes the proof.

C Sampling Pólya structures

In the sequel, X denotes the generating function of $\mathcal{X}$, ϵ is an empty sequence. We present the algorithms from [25] in order to make the paper more self-contained.

Algorithm 1 $\Gamma\,\textsc{Cyc}(\mathcal{A})(\boldsymbol{z})$

Input: Parameters $\boldsymbol{z}$.
Output: A cycle $\textsc{Cyc}(\mathcal{A})$.

Let K be a random variable in $\mathbb{Z}_{>0}$ satisfying $\mathbb{P}(K = k) = -\frac{1}{F_{Cyc(\mathcal{A})}} \frac{\varphi(k)}{k} \ln\left(1 - A(\boldsymbol{z}^k)\right)$.
Draw k following the law of K.
Let L be a random variable in $\mathbb{Z}_{>0}$ satisfying $\mathbb{P}(L = \ell) = -\frac{(A(\boldsymbol{z}^k))^\ell}{\ell} \frac{1}{\ln(1 - A(\boldsymbol{z}^k))}$.
Draw ℓ following the law of L, $M \leftarrow \epsilon$.
for i ***from*** 1 ***to*** ℓ **do**
  $A_i \leftarrow \Gamma\mathcal{A}(\boldsymbol{z}^k)$
  $M \leftarrow M \cdot A_i$
end for
return $[M \ldots M]_{k \text{ times}}$

Algorithm 2 $\Gamma\,\textsc{MSet}(\mathcal{A})(\boldsymbol{z})$

Input: Parameters $\boldsymbol{z}$.
Output: A multi-set $\textsc{MSet}(\mathcal{A})$.

Let K be a random variable in $\mathbb{Z}_{\geq 0}$ satisfying $\mathbb{P}(K \leq k) = \prod_{j>k} \exp\left(-\frac{1}{j} A(\boldsymbol{z}^j)\right)$.
Draw k following the law of K, $S \leftarrow \epsilon$.
if $k > 0$ **then**
  for j ***from*** 1 ***to*** $k-1$ **do**
    Draw $q \sim \text{Poiss}\left(\frac{1}{j} A(\boldsymbol{z}^j)\right)$.
    for i ***from*** 1 ***to*** q **do**
      $A_i \leftarrow j$ copies of $\Gamma\mathcal{A}(\boldsymbol{z}^j)$
      $S \leftarrow S \cdot A_i$
    end for
  end for
  Draw $q \sim \text{Poiss}_{\geq 1}\left(\frac{1}{k} A(\boldsymbol{z}^k)\right)$.
  for i ***from*** 1 ***to*** q **do**
    $A_i \leftarrow k$ copies of $\Gamma\mathcal{A}(\boldsymbol{z}^k)$
    $S \leftarrow S \cdot A_i$
  end for
  return S
end if