HAL
open science

Rule-weighted and terminal-weighted context-free grammars have identical expressivity

Yann Ponty

To cite this version:

Yann Ponty. Rule-weighted and terminal-weighted context-free grammars have identical expressivity. [Research Report] 2012. hal-00693600

HAL Id: hal-00693600
https://inria.hal.science/hal-00693600

Submitted on 2 May 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RULE-WEIGHTED AND TERMINAL-WEIGHTED CONTEXT-FREE GRAMMARS HAVE IDENTICAL EXPRESSIVITY

YANN PONTY

Abstract. Two formalisms have recently been proposed to perform a non-uniform random generation of combinatorial objects based on context-free grammars. The former, introduced by Denise *et al*, associates weights with letters, while the latter, recently explored by Weinberg *et al* in the context of random generation, associates weights to transitions. In this short note, we use a trivial modification of the Greibach Normal Form transformation algorithm, due to Blum and Koch, to show the equivalent expressivities of these two formalisms.

1. Introduction

The random generation of combinatorial objects is one of the natural applications of enumerative combinatorics. Following general principles outlined by Wilf [12], Flajolet *et al* [8] proposed a fully-automated algebraic approach for the extensive class of decomposable combinatorial objects, a large class of objects that includes context-free languages. This pioneering work was later completed by the introduction of Boltzmann samplers, an alternative family of random generation algorithms based on analytical properties of the underlying generating functions [7]. However, these works only addressed the uniform distribution, while many applications of random generation (e.g. in RNA bioinformatics [6]) require non-uniform distributions to be modeled.

To that purpose, Denise *et al* [4] introduced (terminal)-weighted grammars, a non-uniform framework where the terminal symbols (letters) are associated with a real positive value, inherited multiplicatively by words in the language. Such weights were then used, through a trivial renormalization, to induce a probability distribution on the finite set of words of a given length. Generic random generation algorithms were proposed [4] and implemented within a general random generation toolbox [10]. Analytic and numerical approaches were proposed for figuring out suitable set of weights that would mimic a given, observed, distribution [3]. Finally, a multidimensional rejection scheme was explored to sample words of a given composition, yielding efficient algorithms by generalizing the principles of Boltzmann sampling [2].

More recently, Weinberg *et al* [11] proposed an alternative definition for weighted grammars, associating positive real-values to rules instead of terminal letters. The

Key words and phrases. Weighted Context-Free Grammars; Random Generation; Normal forms.

authors proposed a random generation procedure based on formal grammar manipulations, followed by a call to an unranking algorithm due to Martinez and Molinero [9]. However, the relative expressivities, in term of the distribution induced by the respective weighting schemes, of the two formalisms were not compared.

In this short note, we establish the equivalence of the two formalisms with respect to their induced distributions. After this short introduction, we remind in Section 2 the definitions of terminal-weighted and rule-weighted grammars. Then we turn to an analysis of the relative expressivities of the two formalisms, and establish in Section 3 that any terminal-weighted grammar can be *simulated* by a rule-weighted grammar. Furthermore, we use a Greibach Normal Form transformation to prove, in Section 4, that any rule-weighted grammar can be transformed into a terminal-weighted grammar inducing the same probability distribution, from which one concludes on the equivalence of the two formalisms. We conclude in Section 5 with some closing remarks and perspectives.

2. Definitions

A **context-free grammar** is a 4-tuple $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ where

- Σ is the alphabet, i.e. a finite set of terminal symbols, also called **letters**.
- $\mathcal{N}$ is a finite set of **non-terminal symbols**.
- $\mathcal{P}$ is the finite set of production **rules** of the form $N \to X$, where $N \in \mathcal{N}$ is a non-terminal and $X \in \{\Sigma \cup \mathcal{N}\}^*$ is a sequence of letters and non-terminals.
- $\mathcal{S}$ is the **axiom** of the grammar, i. e. the initial non-terminal.

We will denote by $\mathcal{L}(\mathcal{G})_n$ the set of all words of length n generated by $\mathcal{G}$. This set is generated by iteratively applying production rules to non-terminals until a word in Σ^* is obtained.

Note that the non-terminals on the right-hand side of a production rule can be independently derived. It follows that the derivation process, starting from the initial axiom and ending with a word w over the terminal alphabet, can be represented by a **parse tree** d_w. This (ordered directed) tree associates production rules to each internal node and terminal letters to each leaf, such that the i-th child of a node labeled with $N \to x_1. \cdots .x_k$ is either a terminal letter $x_i \in \Sigma$ or a further derivation of $x_i \in \mathcal{P}$, starting from a root node that derives the axiom $\mathcal{S}$.

Assumptions: Let us assume, for the sake of simplicity, that the grammars considered in the following are **unambiguous**, i.e. that any word in $\mathcal{L}(\mathcal{G})_n$ has exactly one associated parse tree. Moreover, let us assume, without loss of generality, that the grammar is given using a **binary variant** of the **Chomsky Normal Form (CNF)**, which partitions the non-terminals into four classes, restricting their production rules to:

- **Axiom:** $\mathcal{S} \to N$, $N \in \mathcal{N}$, and/or $\mathcal{S} \to \varepsilon$.
- **Unions:** $N \to N' \mid N''$, such that $N, N', N'' \in \mathcal{N}/\{\mathcal{S}\}$.
- **Products:** $N \to N'.N''$, such that $N, N', N'' \in \mathcal{N}/\{\mathcal{S}\}$.
- **Terminals:** $N \to t,\ t \in \Sigma$.

Finally, we will postulate the absence of non-productive terminals, e.g. having rules of the form $N \to N.N'$.

2.1. Terminal-Weighted Grammars.

A non-uniform distribution can be postulated on the language generated by the grammar. To that purpose, two formalisms have been independently proposed, reminded here for the sake of completeness.

Definition 2.1 ((Terminal)-Weighted Grammar [4])**.** A terminal-weighted grammar $\mathcal{G}_\pi$ is a 5-tuple $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ where:

- $(\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ defines a context-free grammar,
- $\pi : \Sigma \to \mathbb{R}^+$ is a **terminal-weighting function** that associates a non-null positive real-valued weight π_t to each terminal symbol t.

The weight of a word $w \in \mathcal{L}(\mathcal{G}_\pi)$ is then given by

$$\pi(w) = \prod_{t \in \Sigma} \pi_t^{|w|_t}$$

and extended into a probability distribution over $\mathcal{L}(\mathcal{G})_n$ by

$$p_{\pi,n}(w) = \frac{\pi(w)}{\sum_{w \in \mathcal{L}(\mathcal{G})_n} \pi(w)}.$$

2.2. **Rule-Weighted Grammars.**

Definition 2.2 ((Rule)-Weighted Grammar [11])**.** A rule-weighted grammar $\mathcal{G}_\lambda$ is a 5-tuple $\mathcal{G}_\lambda = (\lambda, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ where:

- $(\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ defines a context-free grammar,
- $\lambda : \mathcal{P} \to \mathbb{R}^+$ is a **rule-weighting function** that associates a positive non-null real-valued[1] weight λ_r to each derivation $r \in \mathcal{P}$, using the notation $N \to_y X$ to indicate the association of a weight $\lambda_r = y$ to a rule $r = (N \to X)$.

The weight function λ can then be extended multiplicatively over $\mathcal{L}(\mathcal{G}_\lambda)$ through

$$\lambda(w) = \prod_{\substack{r \in d_w \\ r = (N \to_{\lambda_r} X)}} \lambda_r, \ \forall w \in \mathcal{L}(\mathcal{G}_\lambda)$$

where d_w is the (unique) parse tree of w in $\mathcal{G}_\lambda$. This induces a probability distribution over $\mathcal{L}(\mathcal{G}_\lambda)_n$ such that

$$p_{\lambda,n}(w) = \frac{\lambda(w)}{\sum_{w \in \mathcal{L}(\mathcal{G}_\lambda)_n} \lambda(w)}.$$

3. Any terminal-weighted distribution can be obtained using a rule-weighted grammar

Theorem 3.1. *For any terminal-weighted grammar $\mathcal{G}_\pi$, there exists a rule-weighted grammar $\mathcal{G}_\lambda$, $\mathcal{L}(\mathcal{G}_\pi) = \mathcal{L}(\mathcal{G}_\lambda)$, inducing an identical probability distribution.*

Proof. We give a constructive proof of this theorem. For any grammar $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$, let us consider the rule-weighted grammar defined by $\mathcal{G}_\lambda := (\lambda, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$, such that $\lambda(N \to t) = \pi_t$ and $\lambda(\cdot) = 1$ otherwise.

Clearly, the production rules and axioms of $\mathcal{G}_\lambda$ and $\mathcal{G}_\pi$ are identical, therefore one has $\mathcal{L}(\mathcal{G}_\lambda) = \mathcal{L}(\mathcal{G}_\pi)$ and, in particular,

$$\mathcal{L}(\mathcal{G}_\lambda)_n = \mathcal{L}(\mathcal{G}_\pi)_n, \ \forall n \geq 0.$$

[1]More precisely, Weinberg *et al* restrict their formalism to rational weights, based on the rationale that real-numbers would lead to unstable computations. However their framework could easily be extended to any computable real numbers without loss of precision, e.g. by implementing a *confidence intervals* approach described in Denise and Zimmermann [5], therefore we consider a trivial extension of this formalism here.

Let us now remark that any terminal letter t in a produced word w results from the application of a rule of the form $N \to t$, and that the parse trees in $\mathcal{G}_\pi$ and $\mathcal{G}_\lambda$ of any word $w \in \mathcal{L}(\mathcal{G}_\pi) = \mathcal{L}(\mathcal{G}_\lambda)$ are identical. It follows that the occurrences of the terminal letter t in w are in bijection with the occurrences of the $N \to t$ rule in its parse tree d_w, and therefore

$$\lambda(w) = \prod_{r=(N \to_{\lambda_r} t) \in d_w} \lambda_r = \prod_{(N \to_{\lambda_r} t) \in d_w} \pi_t = \pi(w), \ \forall w \in \mathcal{L}(\mathcal{G}_\pi).$$

Since $\mathcal{L}(\mathcal{G}_\lambda)_n = \mathcal{L}(\mathcal{G}_\pi)_n$, then one has

$$\sum_{w \in \mathcal{L}(\mathcal{G}_\lambda)_n} \lambda(w) = \sum_{w \in \mathcal{L}(\mathcal{G}_\pi)_n} \pi(w),$$

and we conclude that, for any length $n \geq 0$, one has

$$p_{\pi,n}(w) = \frac{\lambda(w)}{\sum_{w \in \mathcal{L}(\mathcal{G}_\lambda)_n} \lambda(w)} = \frac{\pi(w)}{\sum_{w \in \mathcal{L}(\mathcal{G}_\pi)_n} \pi(w)} = p_{\lambda,n}(w)$$

which proves our claim. □

4. Any rule-weighted distribution can be obtained using a terminal-weighted grammar

Theorem 4.1. *For any rule-weighted grammar $\mathcal{G}_\lambda$, there exists a terminal-weighted grammar $\mathcal{G}_\pi$, $\mathcal{L}(\mathcal{G}_\lambda) = \mathcal{L}(\mathcal{G}_\pi)$, inducing an identical probability distribution.*

Proof. Let us first remind the definition of the **Greibach Normal Form** (GNF), which requires each production rule to be of the form:

- $\mathcal{S} \to \varepsilon$, where S is the axiom,
- $N \to t.X$, where $t \in \Sigma$ and $X \in \{\Sigma \cup \mathcal{N}/\{\mathcal{S}\}\}^*$.

Based on Lemma 4.2 proven below, we know that any rule-weighted grammar in Chomsky-Normal Form can be transformed into a GNF grammar that generates the same language and induces the same distribution. Let us then assume, without loss of generality, that the input grammar $\mathcal{G}_\lambda = (\lambda, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ is in GNF.

By duplicating the vocabulary, one easily builds a terminal-weighted grammar that induces the same probability distribution as $\mathcal{G}_\lambda$. Namely, let us define $\mathcal{G}_\pi = (\pi, \Sigma_\pi, \mathcal{N}, \mathcal{P}_\pi, \mathcal{S})$ such that $\Sigma_\pi := \{t_r\}_{r \in \mathcal{P}}$, $\pi(t_r) := \lambda(r)$, and

$$\mathcal{P}_\pi := \{N \to t_r.X \mid r = (N \to_x t.X) \in \mathcal{P}\} \cup \{S \to \varepsilon \mid S \to_x \varepsilon \in \mathcal{P}\}.$$

Clearly, each terminal letter in a word produced by $\mathcal{G}_\pi$ can be unambiguously associated with a rule of $\mathcal{G}_\lambda$, therefore the weight of any non-empty word is preserved. Furthermore, the generated languages of $\mathcal{G}_\lambda$ and $\mathcal{G}_\pi$ are identical, so the distribution is preserved. Finally, the weight of the empty word ε, implicitly set to 1 in the new grammar, may generally differ from its original value $\lambda(S \to \varepsilon)$ in $\mathcal{G}_\lambda$. However, ε is the only word of length 0, and therefore has probability 1 in both grammars. We conclude that the probability distribution induced by $\mathcal{G}_\lambda$ is the same as that of $\mathcal{G}_\pi$.

Lemma 4.2. *For any rule-weighted grammar $\mathcal{G}_\lambda = (\lambda, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$, there exists a grammar $\mathcal{H}_{\lambda'}$ in Greibach Normal Form inducing the same distribution.*

Proof. Again, we use a constructive proof, showing that the weight distribution can be preserved during the transformation of the grammar performed by the Blum and Koch normalisation algorithm [1]. Let us state the algorithm:

(1) Renumber non-terminals in any order, starting with the Axiom $\mathcal{S} \Rightarrow N_1$.
(2) For $k = 1$ to $|\mathcal{N}|$, consider the non-terminal N_k:
   (a) For each $r = (N_k \to_x N_j.X) \in \mathcal{P}$, such that $j < k$ and

$$N_j \to_{x_1} X_1, N_j \to_{x_2} X_2, \cdots, N_j \to_{x_m} X_m,$$

   replace r in $\mathcal{P}$ as follows

Former rule(s)	New rule(s)
$N_k \to_x N_j.X$	$N_k \to_{x \cdot x_1} X_1.X$ $N_k \to_{x \cdot x_2} X_2.X$ $\vdots$ $N_k \to_{x \cdot x_m} X_m.X.$

   (b) Fix any left-recursive non-terminal N_k by replacing its rules as follows, using an alternative *chain-rule* construct:

Former rule(s)	New rule(s)
$N_k \to_{x_1} N_k.X_1$ $\vdots$ $N_k \to_{x_m} N_k.X_m$	$N'_k \to_{x_1} X_1.N'_k$ $N'_k \to_{x_1} X_1$ $\vdots$ $N'_k \to_{x_m} X_m.N'_k$ $N'_k \to_{x_m} X_m$
$N_k \to_{y_1} Y_1$ $\vdots$ $N_k \to_{y_{m'}} Y_{m'}$	$N'_k \to_{y_1} Y_1.N'_k$ $N'_k \to_{y_1} Y_1$ $\vdots$ $N'_k \to_{y_{m'}} Y_m.N'_k$ $N'_k \to_{y_{m'}} Y_{m'}$

(3) For $k = |\mathcal{N}|$ down to 1, consider the non-terminal N_k:
   (a) For each $r = (N_k \to_x N_j.X) \in \mathcal{P}$, such that $j > k$ and

$$N_j \to_{x_1} X_1, N_j \to_{x_2} X_2, \cdots, N_j \to_{x_m} X_m,$$

   replace r in $\mathcal{P}$ as follows

Former rule(s)	New rule(s)
$N_k \to_x N_j.X$	$N_k \to_{x \cdot x_1} X_1.X$ $N_k \to_{x \cdot x_2} X_2.X$ $\vdots$ $N_k \to_{x \cdot x_m} X_m.X.$

One easily verifies that, after any iteration of step (2), the grammar no longer contains any rule $N_j \to N_l.X$ such that $l \leq j \leq k$. This holds for N_k which, after the full execution of step (2), does not depend from any non-terminal, and is therefore in GNF. Furthermore, one may assume that, anytime a non-terminal N_k is considered during step (3), every N_j such that $k < j$ is in GNF. Consequently, the expansion of N_j only creates rules that are GNF-compliant, thus N_k is in GNF at the end of the iteration.

Let us denote by $\mathcal{H}_{\lambda'} = (\lambda', \Sigma, \mathcal{N}', \mathcal{P}', \mathcal{S})$ the rule-weighted grammar obtained at the end of the execution. One first remarks that both the expansions (Steps (2.a) and (3.a)) and the chain-rule reversal (Steps (2.b)) preserve the generated language,

so that the language generated by a non-terminal in $\mathcal{G}_\lambda$ is also the language generated by its corresponding non-terminal in $\mathcal{H}_{\lambda'}$. Furthermore, one can prove, by induction on the number of derivations required to generate a word, that the induced probability distribution is kept invariant by rules substitutions operated by the algorithm.

To that purpose, let us first extend the definition of a rule-weighting function to include a partial derivation instead of a single non-terminal. Namely, $\lambda_X(w)$ will represent the weight of w, as derived from $X \in \{\mathcal{P} \cup \Sigma\}^*$ and, in particular, one has $\lambda_{\mathcal{S}} \equiv \lambda$. Let us now consider the rule-weighting functions $\lambda^{\blacktriangleright}$ and $\lambda^{\blacktriangleleft}$, respectively induced by the grammar before and after a modification:

- *Induction hypothesis:* Any word w generated from any $X\{\mathcal{P} \cup \Sigma\}^*$ using d derivations, $1 \leq d < n$, is such that $\lambda'_X(w) = \lambda''_X(w)$.
- *Rule expansion (Steps (2.a) and (3.a)):* Consider a word w, generated using n derivations from some non-terminal N. Clearly, if $N \neq N_k$ or if the first derivation used is $N \to X' \neq N_j.X$, then the rule used to generated w is not affected by the modification. The induction hypothesis applies and one trivially gets $\lambda^{\blacktriangleright}_{N_k}(w) = \lambda^{\blacktriangleleft}_{N_k}(w)$.

  Consider the initial state of the grammar. When w results from a derivation $N \to_x N_j.X$, then there exists a (unique) decomposition $w = w'.w''$, where w' is produced by the application of some rule $N_j \to_{x_i} X_i$, and w'' is derived from X. The weight of w is then given by $\lambda^{\blacktriangleright}_{N_k}(w) = x \cdot x_i \cdot \lambda^{\blacktriangleright}_{X_i}(w') \cdot \lambda^{\blacktriangleright}_{X}(w'')$.

  In the modified version of the grammar, $w = w'.w''$ unambiguous derives from an application of the new rule $N_k \to_{x \cdot x_i} X_i.X$ ($w' \in \mathcal{L}(X_i)$ and $w'' \in \mathcal{L}(X)$), with associated weight $\lambda^{\blacktriangleleft}_k(w) = x \cdot x_i \cdot \lambda''_k(w') \cdot \lambda^{\blacktriangleleft}_k(w'')$. Since both w' and w'' are generated using less than n derivations, then the induction hypothesis applies and one gets

$$\lambda^{\blacktriangleleft}_{N_k}(w) = x \cdot x_i \cdot \lambda^{\blacktriangleleft}_{X_i}(w') \cdot \lambda^{\blacktriangleleft}_{X}(w'') = x \cdot x_i \cdot \lambda^{\blacktriangleright}_{X_i}(w') \cdot \lambda^{\blacktriangleright}_{X}(w'') = \lambda^{\blacktriangleright}_{N_k}(w).$$

- *Chain-rule reversal (Steps (2.b)):* Any word produced using the initial *left-recursive* chain-rule can be uniquely decomposed as $w = w'.w''_1.\cdots.w''_p$, where w' is generated from some $N_k \to_{y_q} Y_q, q \in [1, m']$, and each w''_i is generated by some rule $N_k \to_{x_{q_i}} N_k.X_{q_i}, q_i \in [1, m]$. Its weight is therefore given by $\lambda^{\blacktriangleright}_{N_k}(w) = y_q \cdot (\prod_{i=1}^m x_{q_i}) \cdot \lambda^{\blacktriangleright}_{Y_q}(w') \cdot \left(\prod_{i=1}^m \lambda^{\blacktriangleright}_{X_{q_i}}(w''_i)\right)$.

  After chain-rule reversal, the same decomposition $w = w'.w''_1.\cdots.w''_p$ holds, but the sequence of derivation is now either $N_k \to_{y_q} Y_q$ ($w = w'$), or

$$\begin{aligned} N_k &\to_{y_q} Y_q.N'_k \to_{x_{q_1}} Y_q.X_{q_1}.N'_k \\ &\rightsquigarrow Y_q.X_{q_1}.\cdots.X_{q_{m-1}}.N'_k \\ &\to_{x_{q_m}} Y_q.X_{q_1}.\cdots.X_{q_m} \rightsquigarrow w'.w''_1.\cdots.w''_p. \end{aligned}$$

In both cases, the induction hypothesis applies for each element of the decomposition, and the weight of w in the new decomposition is given by

$$\begin{aligned}\lambda^{\blacktriangleleft}_{N_k}(w) &= y_q \cdot \left(\prod_{i=1}^{m} x_{q_i}\right) \cdot \lambda^{\blacktriangleleft}_{Y_q}(w') \cdot \left(\prod_{i=1}^{m} \lambda^{\blacktriangleleft}_{X_{q_i}}(w''_i)\right) \\ &= y_q \cdot \left(\prod_{i=1}^{m} x_{q_i}\right) \cdot \lambda^{\blacktriangleright}_{Y_q}(w') \cdot \left(\prod_{i=1}^{m} \lambda^{\blacktriangleright}_{X_{q_i}}(w''_i)\right) = \lambda^{\blacktriangleright}_{N_k}(w).\end{aligned}$$

It follows that the weight of any word is left unchanged by the substitutions performed in the algorithm. Since the generated language is also preserved, then such a preservation of the weights implies a preservation of the probabilities. We conclude that the returned grammar, in addition to being in GNF, also induces the same probability distribution as $\mathcal{G}_\lambda$. □

□

5. Conclusion

Using a trivial modification of the Blum and Koch algorithm [1], we showed that weighting terminal or weighting rules have equal expressive power, i.e. that any distribution captured by the former formalism is also captured by the other and vice-versa.

While both proofs are relatively trivial, going from rule-weighted grammars to terminal-weighted grammars turned out to be more involved than the alternative, leading to an increase of the number of rules. However, this observation might be deceptive, as the choice of the Greibach Normal Form as an intermediate form is only one out of possibly many alternatives, and one could devise more efficient grammar transforms capturing the same distributions. Moreover, it is noteworthy that, even if one chooses to use GNF grammars, there still seems to be a gap between the $O(|\mathcal{P}|^4)$ size of the grammar returned by the Blum and Koch algorithm, and the minimum $O(|\mathcal{P}|^2)$ increase observed for some infinite family of grammars, motivating the search for better GNF transformation algorithms.

References

1. Norbert Blum and Robert Koch, *Greibach normal form transformation revisited*, Information and Computation **150** (1999), no. 1, 112 – 118.
2. O. Bodini and Y. Ponty, *Multi-dimensional Boltzmann sampling of languages*, Proceedings of AOFA'10 (Vienna), Discrete Mathematics and Theoretical Computer Science Proceedings, no. 113, June 2010, pp. 49–64.
3. A. Denise, Y. Ponty, and M. Termier, *Controlled non uniform random generation of decomposable structures*, Theoretical Computer Science **411** (2010), no. 40-42, 3527–3552.
4. A. Denise, O. Roques, and M. Termier, *Random generation of words of context-free languagesaccording to the frequencies of letters*, Mathematics and Computer Science: Algorithms,Trees, Combinatorics and probabilities (D. Gardy and A. Mokkadem, eds.), Trends in Mathematics, Birkhaüser, 2000, pp. 113–125.
5. A. Denise and P. Zimmermann, *Uniform random generation of decomposable structures using floating-point arithmetic*, Theor. Comput. Sci. **218** (1999), no. 2, 233–248.
6. Y. Ding and E. Lawrence, *A statistical sampling algorithm for RNA secondary structure prediction*, Nucleic Acids Research **31** (2003), no. 24, 7280–7301.
7. P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer, *Random sampling from Boltzmann principles*, Automata, Languages, and Programming (P. Widmayer et al., ed.), Lecture Notes in Computer Science, no. 2380, Springer Verlag, 2002, Proceedings of the 29th ICALP Conference, Malaga,July 2002., pp. 501–513.

8. P. Flajolet, P. Zimmermann, and B. Van Cutsem, *Calculus for the random generation of labelled combinatorial structures*, Theoretical Computer Science **132** (1994), 1–35.
9. C. Martinez and X. Molinero, *A generic approach for the unranking of labeled combinatorial classes*, Random Structures & Algorithms, vol. 19, 2001, pp. 472–497.
10. Y. Ponty, M. Termier, and A. Denise, *GenRGenS: Software for generating random genomic sequences and structures*, Bioinformatics **22** (2006), no. 12, 1534–1535.
11. F. Weinberg and M. E. Nebel, *Non uniform generation of combinatorial objects*, Tech. report, Technical Report University of Kaiserslauter, 2010.
12. H. S. Wilf, *A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects*, Advances in Mathematics **24** (1977), 281–291.

CNRS/LIX – Ecole Polytechnique – Palaiseau, France, AMIB Project – INRIA – Saclay, France,

E-mail address: `yann.ponty@lix.polytechnique.fr`