Non-redundant random generation from weighted context-free languages

Yann Ponty

Non-redundant random generation from weighted context-free languages

Yann Ponty
Department of biology
Boston College
Chestnut Hill, MA 02467, USA
ponty@bc.edu

December 20, 2010

Abstract

We address the non-redundant random generation of k words of length n from a context-free language. Additionally, we want to avoid a predefined set of words. We study the limits of a rejection-based approach, whose time complexity is shown to grow exponentially in k in some cases. We propose an alternative recursive algorithm, whose careful implementation allows for a non-redundant generation of k words of size n in $\mathcal{O}(kn \log n)$ arithmetic operations after the precomputation of $\Theta(n)$ numbers. The overall complexity is therefore dominated by the generation of k words, and the non-redundancy comes at a negligible cost.

1 Introduction

The random generation of combinatorial objects has many direct applications in areas ranging from software engineering [1] to bioinformatics [12]. It can help formulate conjectures on the average-case complexity of algorithms, raises new fundamental mathematical questions, and directly benefits from new discoveries of its fundamental objects. These include, but are not limited to, generating functionology, arbitrary precision arithmetics and bijective combinatorics. Following the so-called *recursive* framework introduced by Wilf [15], very elegant and general algorithms for the uniform random generation have been designed [8] and implemented. Many optimizations of this approach been developed, using specificities of certain classes of combinatorial structures [10], or floating-point arithmetics [3]. More recently a probabilistic approach to this problem, the so-called Boltzmann generation [6], has drawn much attention both because its very low memory complexity and its underlying theoretical beauty.

For many applications, it is necessary to drift away from the *uniform* models. A striking example lies in the most recent paradigm for the *in silico* analysis of the RNA molecule's folding. Instead of trying to predict a structure of minimal free-energy, current approaches tend to focus on the *ensemble properties* of achievable conformations, assuming a Boltzmann probability distribution [4]. Random generation is then performed, and complex structural features are evaluated in a statistical manner. In order to capture such features, a general non-uniform scheme was proposed by Denise *et al* [2], based on the concept of *weighted context-free grammars*. Recursive random generation algorithms were derived, with time and space complexities similar to that of the uniform ones [8].

In the weighted probability distribution, the probability ratio between the most and least frequent words sometimes grows exponentially on the size of the generated objects. Therefore it is a natural question to address the **non-redundant random generation** of combinatorial objects, that is the generation of a set of **distinct** objects. By contrast to the general case, this aspect of random generation has, to our best knowledge, only been addressed through the introduction of the `PowerSet` construct by Zimmermann [17]. An algorithm in $\Theta(n^2)$ arithmetic operations, or a practical $\Theta(n^4)$ complexity in this case, was derived for recursive decomposable structures. The absence of redundancy in the set of generated structures was achieved respectively through *rejection* or an *unranking* algorithms. While the former is discussed later on in the document, the latter cannot be transposed directly to the case of weighted languages, since the assumption that different integral ranks correspond to different objects does not hold. Furthermore, the algorithm cannot *immediately* account for an initial set of forbidden words, short of computing an intersection grammar, a computationally intensive process that would further induce large time and memory constants.

In this paper, we address the non-redundant generation of words from a context-free language, generated while avoiding a pre-defined inclusive set $\mathcal{F}$. First, we define some concepts and notations, which allows us to rephrase the random generation process as a *step-by-step* walk. Then, we investigate the efficiency of a rejection-based approach to our problem. We show that, although well-suited for the uniform distribution, the rejection approach can yield high average-case complexities for large sets of forbidden words. In the weighted case, we show that the complexity of the rejection approach can grow exponentially on the number of desired sequences. Finally, we introduce a new algorithm, based on a recursive approach, which generates k sequences of length n while avoiding a set $\mathcal{F}$ at the cost of $\mathcal{O}(kn\log(n))$ arithmetic operations after a precomputation in $\Theta((n+|\mathcal{F}|)n)$ arithmetic operations.

2 Concepts and notations

2.1 Context-free grammars

We remind some formal definitions on context-free grammars and Chomsky Normal Form (CNF). 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.
- $\mathcal{N}$ is a finite set of non-terminal symbols.
- $\mathcal{P}$ is the finite set of production rules, each of the form $N \to X$, for $N \in \mathcal{N}$ any non-terminal and $X \in \{\Sigma \cup \mathcal{N}\}^*$.
- $\mathcal{S}$ is the **axiom** of the grammar, i. e. the initial non-terminal.

A grammar $\mathcal{G}$ is then said to be in **Chomsky Normal Form** (CNF) iff the rules associated to each non-terminal $N \in \mathcal{N}$ are either:

- Product case: $N \to N' \,.\, N''$
- Union case: $N \to N' \mid N''$
- Terminal case: $N \to t$

for $N, N', N'' \in \mathcal{N}$ non-terminal symbols and $t \in \Sigma$ a terminal symbol. In addition, the axiom $\mathcal{S} \in \mathcal{N}$ is allowed to derive the empty word ε only if $\mathcal{S}$ does not appear in the right-hand side of any production.

Let $\mathcal{L}(N)$ be the language associated to $N \in \mathcal{N}$, that is the set of words on terminal symbols, accessible through a sequence of derivations starting from N. Then the language $\mathcal{L}(\mathcal{G})$ generated by a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ is defined to be the language $\mathcal{L}(\mathcal{S})$ associated with its axiom $\mathcal{S}$. It is a classic result that any context-free grammar $\mathcal{G}$ can be transformed into a grammar $\mathcal{G}'$ in Chomsky Normal Form (CNF) such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$. Therefore, we will define our algorithm on CNF grammars, but will nevertheless illustrate its behavior on a compact, non-CNF, grammar for Motzkin words. Indeed the normalization process introduces numerous non-terminals even for the most trivial grammars.

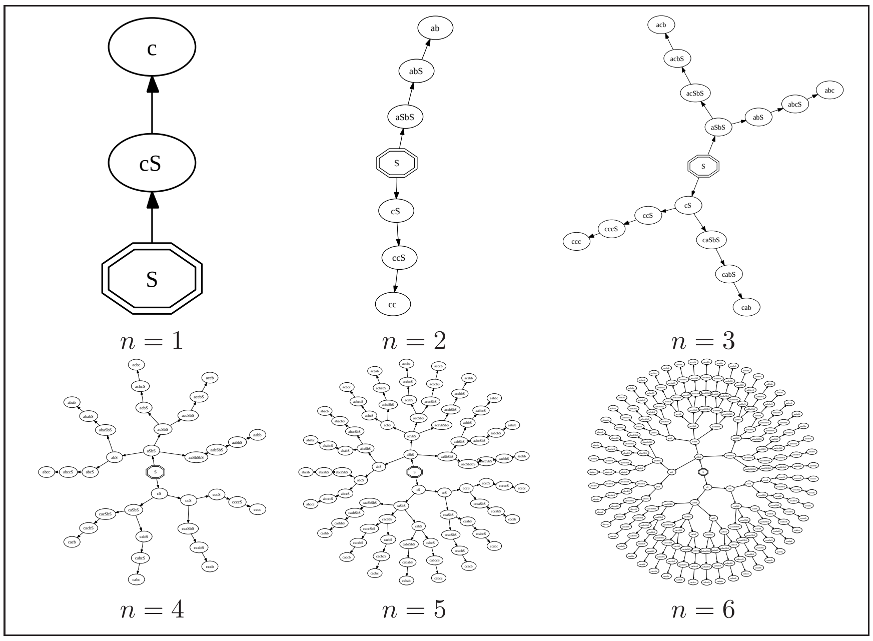


Figure 1: Trees of all walks associated with Motzkin words of size $n \in [1, 6]$ generated by the grammar $S \to a\ S\ b\ S \mid c\ S \mid \varepsilon$ under a *leftmost first* derivation policy.

2.2 Fixed-length languages descriptions: Immature words

We call **mature** word a sequence of terminal symbols. More generally, we will call **immature** a word that contains both terminal and non-terminal symbols, thus potentially requiring some further processing before becoming a mature word. We will denote $\mathcal{L}^{\lhd}(N)$ the set of immature words accessible from a non-terminal symbol N and extend this notion to $\mathcal{L}^{\lhd}(\mathcal{G})$ the immature words accessible from the axiom of a grammar $\mathcal{G}$. It is noteworthy that $\mathcal{L}(\mathcal{G}) \subset \mathcal{L}^{\lhd}(\mathcal{G})$.

We can then **attach required lengths** to the symbols of an immature word. For instance, in the grammar for Motzkin words from figure 1, $c\ a\ S_4\ b\ S_0$ will be a specialization of the immature word $c\ a\ S\ b\ S$, where the words generated from the first (resp. second) instance of the non-terminal S are required to have length 4 (resp. 0). Formally, this amounts to taking into consideration couples of the form $(\omega, \mathbf{n})$ where ω is an immature word, and $\mathbf{n} \in \mathbb{N}^{|\omega|}$ is a vector of sizes for words generated from the different symbols of ω. We naturally extend the notion of language associated with an immature word to these couples in the following way:

$$\mathcal{L}(\omega, \mathbf{n}) = \prod_{i \geq 1}^{|\omega|} \mathcal{L}(\omega_i, \mathbf{n}_i)$$

The length vector $\mathbf{n}$ associated with an immature word ω may be omitted in the rest of the document for the sake of simplicity.

2.3 Random generation as a random walk in language space

An **atomic derivation**, starting from a word $\omega = \omega' \,.\, N \,.\, \omega'' \in \{\Sigma \cup \mathcal{N}\}^*$, is the application of a production $N \to X$ to ω that replaces N by the right-hand side X of the production, which yields $\omega \Rightarrow \omega'.X.\omega''$. Let us call **derivation policy** a deterministic strategy that points, in an immature word, the first non-terminal to be rewritten through an atomic derivation. Formally, in the context of a grammar $\mathcal{G}$, a derivation policy is a function $\phi : \mathcal{L}(\mathcal{G}) \cup \mathcal{L}^{\lhd}(\mathcal{G}) \to \mathbb{N} \cup \{\emptyset\}$ such that

$$\begin{array}{rrcl} \phi : & \omega \in \mathcal{L}(\mathcal{G}) & \rightarrow & \emptyset \\ & \omega' \in \mathcal{L}^{\lhd}(\mathcal{G}) & \rightarrow & i \in [1, |\omega'|] \end{array}$$

A sequence of atomic derivations is then said to be **consistent with a given derivation policy** if the non-terminal rewritten at each step is the one pointed by the policy. A side effect of this somewhat verbose notion is that it provides a convenient framework for defining the **unambiguity** of a grammar without any reference to parse trees.

Definition 2.1. Let $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ be a context-free grammar and ϕ a derivation policy acting on $\mathcal{G}$. The grammar $\mathcal{G}$ is said unambiguous if and only, for each $\omega \in \mathcal{L}(\mathcal{G})$, there exists only one sequence of atomic derivations that is consistent with ϕ and produces ω from $\mathcal{S}$.

The derivation leading to a mature word $\omega \in \mathcal{L}(\mathcal{G})$ in a grammar $\mathcal{G}$ can then be associated in a one-to-one fashion with a walk in the space of languages associated with immature words, or **parse walk**, taking steps consistent with a given derivation policy ϕ. More precisely, such walks starts from the axiom $\mathcal{S}$ of the grammar. From a given immature word $X \in \mathcal{L}^{\lhd}(\mathcal{G})$, the derivation policy ϕ points at a position $k := \phi(X)$, where a non-terminal X_k can be found. The parse walk can then be prolonged using one of the derivations acting on X_k (See Figures 1 and 2).

2.4 Weighted context-free grammars

Definition 2.2 (Weighted Grammar [2])**.** A weighted grammar $\mathcal{G}_\pi$ is a 5-tuple $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$ where Σ, $\mathcal{N}$, $\mathcal{P}$ and $\mathcal{S}$ are the members of a context-free grammar, and $\pi : \Sigma \rightarrow \mathbb{R}$ is a weighting function, that associates a real-valued weight π_t to each terminal symbol t.

This notion of weight naturally extends to a mature word w in a multiplicative fashion, i.e. such that $\pi(w) = \prod_{i=1}^{|w|} \pi_{w_i}$. From this, we can define a **π-weighted probability distribution** over a set of words $\mathcal{L}$, such that the probability associated with $\omega \in \mathcal{L}$ is

$$\mathbb{P}(w \mid \pi) = \frac{\pi(w)}{\displaystyle\sum_{w' \in \mathcal{L}} \pi(w')}$$

In the rest of the document, we may use $\pi(\omega)$ instead of $\pi(\mathcal{L}(\omega, \mathbf{n}))$ to denote the total weight of all words derivable from an immature word ω with associated lengths $\mathbf{n}$.

3 Efficiency of a rejection approach for the non-redundant generation

We address the uniform generation of a non-redundant set of words from a language $\mathcal{L}$ with **forbidden words** $\mathcal{F} \subset \mathcal{L}$. A **rejection approach** for this problem consists in drawing words at random in an unconstrained way, rejecting those previously sampled until k distinct words are generated, as prescribed by Zimmermann [17] in the case of recursive specifications. For the generation of words from context-free languages, we refer to previous works of Flajolet *et al* [8, 6], or Denise *et al* [3] that achieves a $\mathcal{O}(n^{1+\varepsilon})$ complexity through highly non-trivial floating-point arithmetics.

3.1 The uniform case

Theorem 3.1. *Let $\mathcal{L}$ be a context-free language, $n \in \mathbb{N}^+$ a positive integer and $\mathcal{F} \subset \mathcal{L}_n$ a set of forbidden words. Then the* **rejection approach** *for the non-redundant uniform random generation of k words of size n from $\mathcal{L}$ has average-case complexity in $\mathcal{O}\left(\left(\frac{|\mathcal{L}_n|}{|\mathcal{L}_n| - |\mathcal{F}|}\right) n^{1+\varepsilon} k \log k\right)$.*

Proof. In the uniform model when $\mathcal{F} = \emptyset$, the number of attempts necessary to the generation of the i-th word only depends on i and is independent from prior events. Thus the random variable $X_{n,k}$ that contains the total number of trials for the generation of k words of size n is such that

$$\mathbb{E}(X_{n,k}) = \sum_{i=0}^{k-1} \frac{l_n}{l_n - i} = l_n(\mathcal{H}_{l_n} - \mathcal{H}_{l_n-k})$$

where $l_n := |\mathcal{L}_n|$ is the number of words of size n in the language and $\mathcal{H}_i$ the harmonic number of order i, as pointed out by Flajolet *et al* [9]. It follows that $\mathbb{E}(X_{n,k})$ is trivially increasing with k, while remaining upper bounded by $k\mathcal{H}_k \in \Theta(k\log(k))$ when $k = l_n$ (Coupon collector problem). Since the expected number of rejections due to a non-empty forbidden set $\mathcal{F}$ remains the same throughout the generation, and does not have any influence over the generated sequences, it can be considered independently and contributes to a factor $\frac{|\mathcal{L}_n|}{|\mathcal{L}_n|-|\mathcal{F}|}$. Finally, each generation takes time $\mathcal{O}(n^{1+\varepsilon})$, independently from both the generated sequence and the cardinality of $\mathcal{L}_n$. □

The complexity of a rejection approach to this problem is then mainly linear, unless the set $\mathcal{F}$ *overwhelms* the language generated by the grammar. In this case, the generation can become linear in the cardinality of $\mathcal{L}_n$, that is exponential in n for most languages. Furthermore, the worst-case time complexity of this approach remains unbounded.

3.2 Weighted context-free languages

By contrast with the uniform case, the rejection approach to the non-redundant random generation for **weighted context-free languages** can yield an **exponential complexity**, even when starting from an empty set of forbidden words $\mathcal{F} = \emptyset$. Indeed, the weighted distribution can specialize into a power-law distribution on the number of occurrences of the terminal symbol having highest weight, leading to an exponentially-growing number of rejections.
Example: Consider the following grammar, generating the language a^*b^* of words starting with an arbitrary number of a, followed by any number of b:

$$\begin{array}{rcl} S & \to & a\,.\,S \mid T \\ T & \to & b\,.\,T \mid \varepsilon \end{array}$$

We adjoin a weight function π to this grammar, such that $\pi(b) := \alpha > 1$ and $\pi(a) := 1$. The probability of the word $\omega_m := a^{n-m}b^m$ among $\mathcal{S}_n$ is then

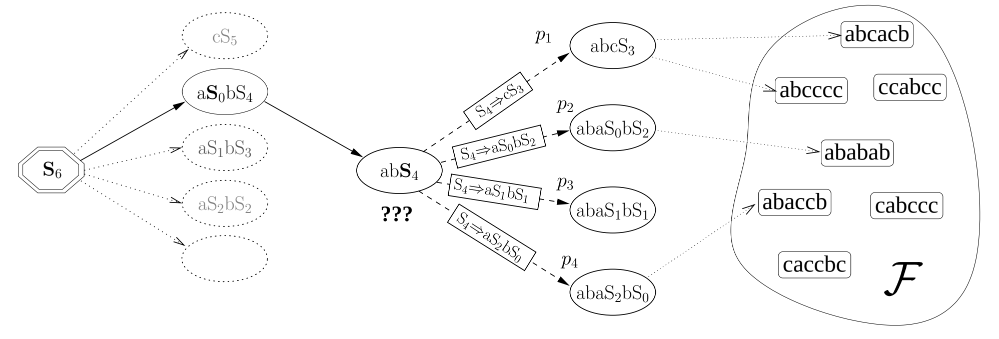
$$\mathbb{P}(\omega_m) = \frac{\pi(\omega_m)}{\sum_{\substack{\omega\in\mathcal{L}(S)\\|\omega|=n}} \pi(\omega)} = \frac{\alpha^m}{\sum_{i=0}^{n}\alpha^i} = \frac{\alpha^{m+1}-\alpha^m}{\alpha^{n+1}-1} < \alpha^{m-n}.$$

Now consider the set $\mathcal{V}_{n,k} \subset \mathcal{S}_n$ of words having less than $n-k$ occurrences of the symbol b. The probability of generating a word from $\mathcal{V}_{n,k}$ is then

$$\mathbb{P}(\mathcal{V}_{n,k}) = \sum_{i=0}^{n-k} \mathbb{P}(\omega_{n-k-i}) = \frac{\alpha^{n-k+1}-1}{\alpha^{n+1}-1} < \alpha^{-k}$$

The expected number of generations before a sequence from $\mathcal{V}_{n,k}$ is generated is then lower-bounded by α^k. Since any non-redundant set of k sequences issued from $\mathcal{S}_n$ must contain at least one sequence from $\mathcal{V}_{n,k}$, then the average-case time complexity of the rejection approach is in $\Omega(n\alpha^k)$, that is exponential in k the number of words.

One may argue that the previous example is not very typical of the rejection algorithm's behavior on a general context-free language, since the grammar is left linear and consequently defines a rational language. By contrast, it can be shown that, under a natural assumption, no word can asymptotically contribute up to a significant proportion of the distribution in simple type grammars.



$$p_1 = \frac{\pi(abcS_3) - \pi(abcacb) - \pi(abcccc)}{\kappa} \qquad p_2 = \frac{\pi(abaS_0bS_2) - \pi(ababab)}{\kappa}$$

$$p_3 = \frac{\pi(abaS_1bS_1)}{\kappa} \qquad p_4 = \frac{\pi(abaS_2bS_0) - \pi(abaccb)}{\kappa}$$

$$\kappa := \pi(abS_4) - \pi(abcacb) - \pi(abcccc) - \pi(ababab) - \pi(abaccb)$$

Figure 2: Snapshot of a *step-by-step* random scenario for a Motzkin word of length 6, generated while avoiding $\mathcal{F}$. From abS_4, the recursive approach prescribes that the probabilities associated with candidate derivations must be proportional to the overall weights for resulting immature words. Additionally, we need to subtract the contributions of $\mathcal{F}$ to these immature words, which yields probabilities (p_1, p_2, p_3, p_4).

Theorem 3.2. *Let $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{S}, \mathcal{P})$ be a weighted grammar of simple type*[1]*. Let ω^0_n be the word of length n generated from $\mathcal{G}_\pi$ with highest probability (i.e. weight) in the π-weighted distribution. Additionally, let us assume that there exists $\alpha, \kappa \in \mathbb{R}^+$ positive constants such that $\pi(\omega^0_n) \underset{n\to\infty}{\longrightarrow} \kappa\alpha^n$.*
Then the probability of ω^0 tends to 0 when $n \to \infty$:

$$\mathbb{P}(\omega^0 \mid \pi) = \frac{\pi(\omega^0)}{\pi(\mathcal{L}(\mathcal{G}_\pi)_n)} \underset{n\to\infty}{\longrightarrow} 0$$

Proof. From the Drmota-Lalley-Woods theorem [5, 11, 16], we know that the generating function of a simple type grammar has a *square-root type* singularity, and its coefficients admits an expansion of the form $\frac{\kappa\beta^n}{n\sqrt{n}}(1 + \mathcal{O}(1/n))$. This property holds in the case of weighted context-free grammars, where the coefficients are now the overall weights $W_n := \pi(\mathcal{L}(\mathcal{G}_\pi)_n)$. Since ω^0_n is contributing to W_n, then $\pi(\omega^0_n) \leq \pi(\mathcal{L}(\mathcal{G}_\pi)_n)$ and therefore $\beta > \alpha$. □

Lastly it can be shown that, for any fixed length n, the set of words $\mathcal{M}$ having maximal number of occurrences of a given symbol t have total probability 1 when $\pi(t) \to \infty$. It follows that sampling more than $|\mathcal{M}|$ words can be extremely time-consuming if one of the weights dominates the others by several orders of magnitude.

4 Step-by-step approach to the non-redundant random generation

Instead of approaching the random generation from context-free languages by considering non-terminal symbols as **independent generators** [8, 2], we consider the random generation sce-

[1]A grammar of simple type is mainly a grammar whose dependency graph is strongly-connected and whose number of words follow an aperiodic progression (See [7] for a more complete definition). Such a grammar can easily be found for the avatars of the algebraic class of combinatorial structures (Dyck words, Motzkin paths, trees of fixed degree,...), all of which can be interpreted as trees.

narios as random (parse) walks. This allows to determine to which of the local alternatives to subtract the contributions of forbidden (or already generated) words.

4.1 The algorithm

We propose the following Algorithm $\mathcal{A}$, which from

- a weighted grammar $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$,
- an immature word $(\omega, \mathbf{n})$,
- and a set of forbidden words $\mathcal{F} \in \mathcal{L}((\omega, \mathbf{n}))$,

draws at random a word $\mathcal{L}(\omega, \mathbf{n})/\mathcal{F}$ with respect to a π-weighted distribution:

1. If ω is a mature word
   then if $\mathbf{n} = (1, \dots, 1)$ and $\omega \notin \mathcal{F}$ then return ω else `Error`.
2. Let N^*_m be the non-terminal pointed by ϕ in ω, such as $\omega = \omega'.N^*_m.\omega''$ and k^π be the total weight of words from $\mathcal{F}$ generated from ω.
3. Choose a derivation $\omega \Rightarrow \omega' X \omega''$ with the following probabilities, depending on the type of N^*:
   - $N^* \to N' \mid N''$: Let $k^\pi_{N'}$ be the sum of weights for words from $\mathcal{F}$ generated from $\omega'.N'_m.\omega''$, then
     $$\mathbb{P}(X = N'_m) = \frac{\pi(\mathcal{L}(\omega'.N'_m.\omega'')) - k^\pi_{N'}}{\pi(\mathcal{L}(\omega)) - k^\pi} = 1 - \mathbb{P}(X = N''_m)$$
   - $N^* \to N' \;.\; N''$: Let k^π_i be the sum of weights for words from $\mathcal{F}$ generated from $\omega'.N'_i.N''_{m-i}.\omega''$.
     $$\mathbb{P}(X = N'_i.N''_{m-i}) = \frac{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')) - k^\pi_i}{\pi(\mathcal{L}(\omega)) - k^\pi},\ i \in [1, m-1]$$
   - $N^* \to t$: If $m = 1$ then $\mathbb{P}(X = t) = 1$ else `Error`.
4. Iterate from step 1.

Proposition 4.1. *Let $\mathcal{G}_\pi = (\pi, \Sigma, \mathcal{N}, \mathcal{S}, \mathcal{P})$ be a weighted grammar, $\omega \in \{\Sigma \cup \mathcal{N}\}^*$ be an immature word and $\mathbf{n} \in \mathbb{N}^{|\omega|}$ be a vector of sizes associated with positions of ω.*
Then Algorithm $\mathcal{A}$ draws a mature word at random according to the π-weighted distribution from $\mathcal{L}(\omega, \mathbf{n})\backslash\mathcal{F}$ or throws an Error if $\mathcal{L}(\omega, \mathbf{n}) = \emptyset$.

Proof. The previous claim can be proved very quickly by induction on the number k of required executions of line 1 before a mature word is issued:
Base: The $k = 0$ case corresponds to an already mature word ω, for which the associated language is limited to $\{\omega\}$. As ω is generated by line 1 iff $\omega \notin \mathcal{F}$ then the claimed result holds in that case.
Inductive step: Assuming that the theorem holds for $k \geq n$, we investigate the probabilities of emission for words that require $k = n + 1$ derivations. Let N^*_m be the non-terminal pointed by ϕ, then:

- $N^* \to N' \mid N''$: Assume that the derivation $N^*_m \Rightarrow N'_m$ is chosen w.p. $\frac{\pi(\mathcal{L}(\omega'.N'_m.\omega''))-k^\pi_{N'}}{\pi(\mathcal{L}(\omega))-k^\pi}$. Then the induction hypothesis applies and a word x is generated from $\mathcal{L}(\omega'.N'_m.\omega'')\backslash\mathcal{F}$ in the π-weighted distribution. In this distribution applied to $\mathcal{L}(\omega'.N'_m.\omega'')\backslash\mathcal{F}$, the probability of x is then

$$\begin{aligned}\mathbb{P}(x \mid \omega'.N'_m.\omega'') &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_m.\omega'')\backslash\mathcal{F})}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_m.\omega'')) - \pi(\mathcal{L}(\omega'.N'_m.\omega'')\cap\mathcal{F})}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_m.\omega'')) - k^\pi_{N'}}\end{aligned}$$

The overall probability of x starting from N^*_m is then

$$\begin{aligned}\mathbb{P}(x) &= \frac{\pi(\mathcal{L}(\omega'.N'_m.\omega'')) - k^\pi_{N'}}{\pi(\mathcal{L}(\omega)) - k^\pi}\cdot\frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_m.\omega'')) - k^\pi_{N'}}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega)) - k^\pi} = \frac{\pi(x)}{\pi(\mathcal{L}(\omega)\backslash\mathcal{F})}\end{aligned}$$

This property also holds if N''_m is chosen, after pointing out that

$$(k^\pi_{N''} = k^\pi - k^\pi_{N'}) \Rightarrow \left(1 - \mathbb{P}(X = N'_m) = \frac{\pi(\mathcal{L}(\omega'.N''_m.\omega'')) - k^\pi_{N'}}{\pi(\mathcal{L}(\omega)) - k^\pi}\right)$$

- $N^* \to N'\ .\ N''$: For any $i \in [1, m-1]$, a partition $N^*_m \Rightarrow N'_i\ .\ N''_{m-i}$ is chosen w.p. $\frac{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega''))-k^\pi_i}{\pi(\mathcal{L}(\omega))-k^\pi}$. Then the induction hypothesis applies and a word x is generated from $\mathcal{L}(\omega'.N'_i.N''_{m-i}\omega'')\backslash\mathcal{F}$ in the π-weighted distribution. In this distribution applied to $\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')\backslash\mathcal{F}$, the probability of x is then

$$\begin{aligned}\mathbb{P}(x \mid \omega'.N'_i.N''_{m-i}.\omega'') &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')\backslash\mathcal{F})}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')) - \pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')\cap\mathcal{F})}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')) - k^\pi_i}\end{aligned}$$

The overall probability of x starting from N^*_m is then

$$\begin{aligned}\mathbb{P}(x) &= \frac{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')) - k^\pi_i}{\pi(\mathcal{L}(\omega)) - k^\pi}\cdot\frac{\pi(x)}{\pi(\mathcal{L}(\omega'.N'_i.N''_{m-i}.\omega'')) - k^\pi_i}\\ &= \frac{\pi(x)}{\pi(\mathcal{L}(\omega)) - k^\pi} = \frac{\pi(x)}{\pi(\mathcal{L}(\omega)\backslash\mathcal{F})}\end{aligned}$$

- $N^* \to t$: The probability of any word x issued from ω is that of the word issued from $\omega'.t.\omega''$, that is $\frac{\pi(x)}{\pi(\mathcal{L}(\omega'.t.\omega'')\backslash\mathcal{F})} = \frac{\pi(x)}{\pi(\mathcal{L}(\omega)\backslash\mathcal{F})}$ by the induction hypothesis.

□

This algorithm then performs random generation of k distinct words from a (weighted) context-free language by setting the initial immature word to $\mathcal{S}_n$ the axiom of $\mathcal{G}_\pi$, adding the freshly generated sequence to $\mathcal{F}$ at each step.

5 Complexities and data structures

The algorithm's complexity depends critically on efficient strategies and data structures for:

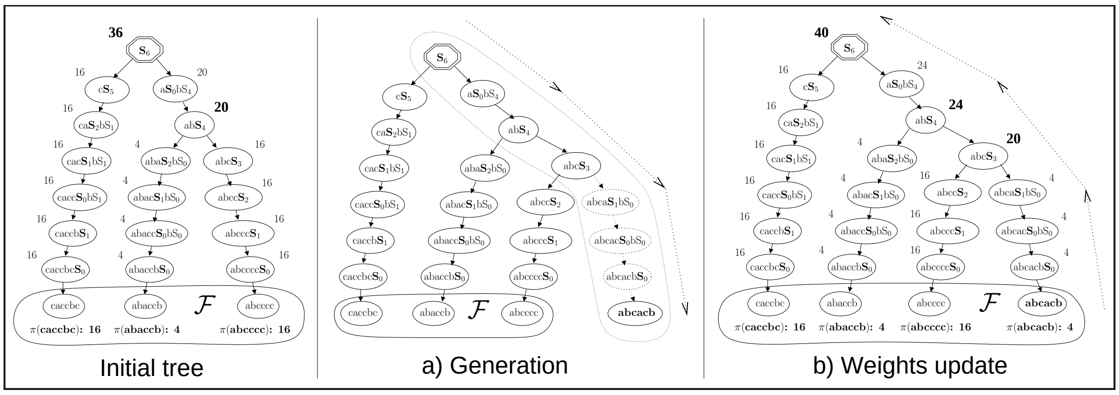


Figure 3: Prefix tree built from the subset $\mathcal{F} = \{\mathbf{caccbc}, \mathbf{abaccb}, \mathbf{abcccc}\}$ of Motzkin words of size 6, using weights $\pi(a) = \pi(b) = 1$ and $\pi(c) = 2$. Generation (**a**) of a new word **abcacb** and update (**b**) of the contributions of $\mathcal{F}$ to the immature words involved in the generation.

1. The weights of languages associated with immature words.
2. The contributions k_π of forbidden (or already generated) words associated with each immature word $(\omega, \mathbf{n})$
3. The investigation of the different *partitions* $N^*_m \Rightarrow N'_i \,.\, N''_{m-i}$ in the case of *product rules*.
4. Big numbers arithmetics.

5.1 Weights of immature languages

Proposition 5.1. *Let* $(\omega, \mathbf{n})$ *be an immature word and its associated length vector, whose weight* $\pi(\omega, \mathbf{n})$ *is known.*
Then a pre-computation involving $\Theta(n)$ *arithmetic operations makes it possible to compute in* $\mathcal{O}(1)$ *arithmetic operations the weight of any word* $(\omega', \mathbf{n}')$ *atomically derived from* $(\omega, \mathbf{n})$.

Proof. In order to compute $\pi(\omega, \mathbf{n})$, we first compute the total weights of languages associated with each non-terminal N for all sizes up to $n = \sum_{n_i \in \mathbf{n}} n_i$, as done by the traditional approach [2]. This can be done in $\Theta(n)$ arithmetic operations, thanks to the holonomic nature of the generating functions at stake. Indeed, the coefficients of an holonomic function obey to a linear recurrence with polynomial coefficients, which can be algorithmically determined (Using the `Maple` package `GFun` [13], for instance).

We can in turn use these values to compute *on the fly* the weights of immature words of interest. Namely, while rewriting an immature word $\omega := \alpha.N^*_k.\beta$ into $\omega' := \alpha.X.\beta$ through a derivation $N^*_k \Rightarrow X$, the new weight $\pi(\omega')$ is given by

$$\pi(\omega') = \pi(\alpha.X.\beta) = \frac{\pi(\alpha)\pi(X)\pi(\beta)\pi(N^*_k)}{\pi(N^*_k)} = \frac{\pi(\omega)\pi(X)}{\pi(N^*_k)}$$

where X contains at most two terms (CNF) and therefore a constant number of arithmetic operations is involved. □

5.2 A prefix tree for forbidden words

Proposition 5.2. *Let* $\mathcal{F}$ *be a set of words characterized by their parse walks*[2]*. Let* $\mathcal{T}$ *be a prefix tree built from the parse walks from* $\mathcal{F}$ *in* $\Theta(|\mathcal{F}|.n)$ *arithmetic operations. Then for any immature*

[2]Starting from an unparsed *raw* set of forbidden words $\mathcal{F}$ will require some additional parsing, which can be performed in $\Theta(|\mathcal{F}|.n^3)$ by a CYK-type algorithm.

word ω reached during a random scenario, the contribution $k_\pi := \pi(\mathcal{L}(\omega) \cap \mathcal{F})$ of forbidden words can be computed in $\mathcal{O}(1)$ arithmetic operations. Furthermore, updating $\mathcal{T}$ through the addition of a generated word can be performed in $\Theta(n)$ arithmetic operations.

Proof. Thanks to the unambiguity of the grammar, a mature word $v \in \mathcal{L}$ belongs to the language generated from an immature one $\omega \in \mathcal{L}^{\triangleleft}$ iff ω is found on the parse walk of v. Therefore we gather the parse walks associated with forbidden words $\mathcal{F}$ into a **prefix tree**, additionally storing at each node the total weight of all words accessible from it. It is then possible to traverse the prefix tree during the generation, read the values of k_π for candidate derivations on the children of the current node.

The update of the prefix tree after a generation can be performed in $\Theta(n)$ arithmetic operations in two phases, as illustrated by Figure 3: First, a *top-down* stage adds nodes for each immature word traversed during the generation (**a**); Then, a *bottom-up* stage will propagate the weight of the sampled mature word to his ancestors (**b**). $\square$

Finally, it is remarkable that many of the internal nodes in the trees have degree 1, which means that their value for k_π is just a copy of their child's own. Since in any tree the number of internal nodes of degree at least two is strictly smaller than the total number of leaves, then the number of *different* values for k_π is bounded by $2 * |\mathcal{F}|$. It follows that the memory needed to store the prefix tree will scale like $\mathcal{O}(n|\mathcal{F}|)$, even if the encoding of each k_π requires $\Theta(n)$ bits of storage.

5.3 Miscellaneous

For point 3, we can use the so-called Boustrophedon strategy, which allows for an $\mathcal{O}(n\log(n))$ arithmetic operations generation in the worst case scenario. Since we only restrict the generation set to authorized (or not previously generated) words, such a property should hold in our case.

For point 4, it is reasonable, for all practical purpose, to assume that the weights are going to be expressed as rational numbers. Multiplying these weights by the least common multiple of their denominators yields a new set of integral weights inducing the same probability distribution, thus arbitrary precision integers can be used. The numbers will scale like $\mathcal{O}(\alpha^n)$ for some explicit α, since the resulting language is context-free, and operations performed on such numbers will take time $\mathcal{O}(n\log(n)\log\log(n))$ [14], while the space occupied by their encoding is in $\mathcal{O}(n)$.

5.4 Summary

Let $n \in \mathbb{N}^+$ be the overall length for generated words, $k \in \mathbb{N}^+$ the number of distinct generated words and $\mathcal{F}$ the initial set of forbidden parse walks:

- The **time-complexity** of Algorithm $\mathcal{A}$ is in $\Theta(kn\log(n))$ **arithmetic operations** in the worst case scenario, after a pre-processing in $\Theta(|\mathcal{F}|n + n)$ **arithmetic operations**.
- The **memory complexity** is in $\Theta(n)$ **numbers** for the pre-processing, plus $\Theta((|\mathcal{F}| + k)n)$ **bits** for the storage of the prefix tree.
- For rational-valued weights, using arbitrary arithmetics, the associated bit-complexities are in respectively $\Theta(kn^2\log(n))$ for time and $\Theta((|\mathcal{F}| + k + n)n)$ for memory.
- Lastly, starting from an empty forbidden set $\mathcal{F} = \emptyset$ yields a generation in $\Theta(kn^2\log(n))$ for time and $\Theta(kn + n^2)$ for memory, complexities similar to that of the **possibly redundant** traditional approach [2, 8].

6 Conclusion and perspectives

We addressed the random generation of non-redundant sets of sequences from context-free languages, while avoiding a predefined set of words. We first investigated the efficiency of a rejection approach. Such an approach was found to be acceptable in the uniform case. By contrast, for weighted languages, we showed that for some languages the expected number of rejections would grow exponentially on the desired number of generated sequences. Furthermore, we showed that in typical context-free languages and for fixed length, the probability distribution can be dominated by a small number of sequences. We proposed an alternative algorithm solution for this problem, based on the so-called recursive approach. The correctness of the algorithm was demonstrated, and its efficient implementation discussed. This algorithm was showed to achieve the generation of a non-redundant set of k structures with a time-complexity in $\mathcal{O}(kn^2\log(n))$, while using $\mathcal{O}(kn+n^2)$ bits of storage. These complexities hold in the worst-case scenario, are almost unaffected by to the weights function used, and are equivalent to that of the traditional, possibly redundant, generation of k words using previous approaches.

One natural extension of the current work concerns the random generation of the more general class of decomposable structures [8]. Indeed, such aspects like the *pointing* and *unpointing* operator are not explicitly accounted for in the current work. Furthermore, the generation of labeled structures might be amenable to similar techniques in order to avoid a redundant generation. It is unclear however how to extend the notion of parse tree in this context. Isomorphism issues might arise, for instance while using the *unranking* operator.

Following the remark that the random generation from reasonable specifications is a *numerically stable problem* [3], we could envision using arbitrary precision arithmetics to achieve a $\mathcal{O}(kn^{1+\varepsilon})$ complexity. Such a feature could accelerate an implementation of this algorithm, for instance in the software `GenRGenS` [12] that already supports the formalism of weighted grammars. Another direction for an efficient implementation of this approach would be to investigate the use of Boltzmann samplers [6].

Moreover, the influence of the number of desired sequences, the length and the weights over the complexity of a rejection based approach deserves to be further characterized. Namely, are there *simple-type* grammars giving rise to an exponential complexity on k ? Can *phase transition*-like phenomena be observed for varying weights ?

Acknowledgements

The author would like to thank A. Denise, for his many helpful comments and suggestions, and C. Herrbach, for asserting that some parts of the document actually made some sense.

References

[1] A. Denise, M.-C. Gaudel, S.-D. Gouraud, R. Lassaigne, and S. Peyronnet, *Uniform random sampling of traces in very large models*, First ACM International Workshop on Random Testing (ISSTA), 2006, pp. 10–19.

[2] A. Denise, O. Roques, and M. Termier, *Random generation of words of context-free languages according 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.

[3] A. Denise and P. Zimmermann, *Uniform random generation of decomposable structures using floating-point arithmetic*, Theor. Comput. Sci. **218** (1999), no. 2, 233–248.

[4] Y. Ding and E. Lawrence, *A statistical sampling algorithm for RNA secondary structure prediction*, Nucleic Acids Research **31** (2003), no. 24, 7280–7301.

[5] M. Drmota, *Systems of functional equations*, Random Struct. Alg. **10** (1997), 103–124.

[6] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer, *Boltzmann samplers for the random generation of combinatorial structures*, Combinatorics, Probablity, and Computing **13** (2004), no. 4–5, 577–625, Special issue on Analysis of Algorithms.

[7] P. Flajolet, E. Fusy, and C. Pivoteau, *Boltzmann sampling of unlabelled structures*, Proceedings of ANALCO'07 (SIAM Press, ed.), January 2007.

[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] Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier, *Birthday paradox, coupon collectors, caching algorithms and self-organizing search*, Discrete Appl. Math. **39** (1992), no. 3, 207–229.

[10] M. Goldwurm, *Random generation of words in an algebraic language in linear binary space*, Information Processing Letters **54** (1995), 229–233.

[11] S. P. Lalley, *Finite range random walk on free groups and homogeneous trees*, Ann. Probab. **21** (1993), 2087–2130.

[12] Y. Ponty, M. Termier, and A. Denise, *GenRGenS: Software for generating random genomic sequences and structures*, Bioinformatics **22** (2006), no. 12, 1534–1535.

[13] B. Salvy and P. Zimmerman, *Gfun: a maple package for the manipulation of generating and holonomic functions in one variable*, ACM Transactions on Mathematical Softwares **20** (1994), no. 2, 163–177.

[14] J. van der Hoeven, *Relax, but don't be too lazy*, Journal of Symbolic Computation **34** (2002), 479–542.

[15] H. S. Wilf, *A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects*, Advances in Mathematics **24** (1977), 281–291.

[16] A. R. Woods, *Coloring rules for finite trees, and probabilities of monadic second order sentences*, Random Struct. Alg. **10** (1997), 453–485.

[17] P. Zimmermann, *Uniform random generation for the powerset construction*, Proceedings of the 7th conference on Formal Power Series and Algebraic Combinatorics, 1995, pp. 589–600.