Non-redundant random generation from weighted context-free languages

Yann Ponty

Boston College → Paris 6 (Dept de biologie)

June 23, 2008

Context-free grammars

Definition (Context-free grammar)

Context-free grammar = 4-tuple $(\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S})$:

- Σ: Alphabet.
- $\mathcal{N}$: Non-terminal symbols.
- $\mathcal{P}$: Set of production rules $N \to X \in \mathcal{N} \times \{\Sigma \cup \mathcal{N}\}^*$.
- $\mathcal{S}$: Axiom, or initial non-terminal.

Alt.: Context-free grammar = **admissible specification** using:

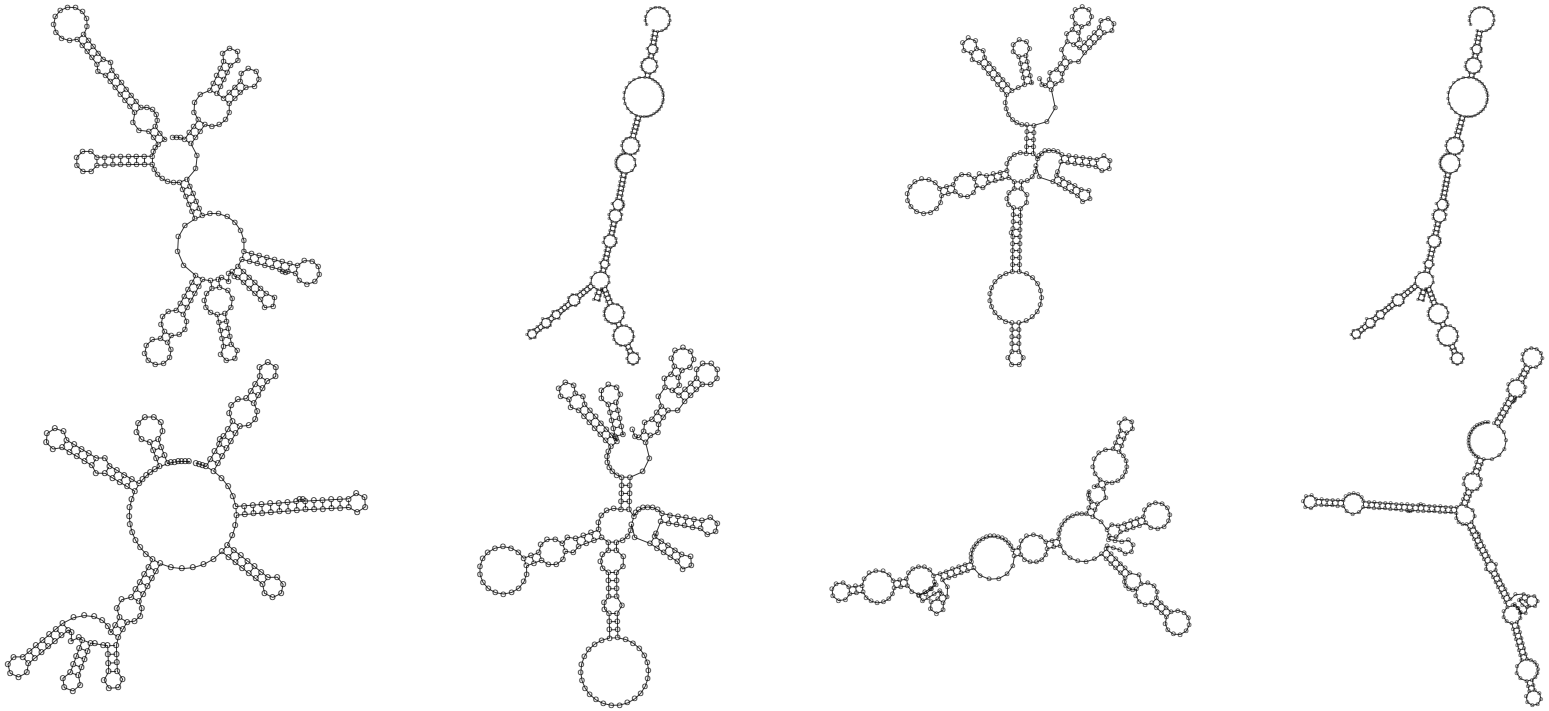
- Operators $\{\times, +\}$
- Finite set of atoms $\{Z_1, Z_2, \ldots, Z_k\}$
- Empty structure 1

Uniform random generation

Rationale

Nature **dislikes** *uniformity (S. Brlek 05)*

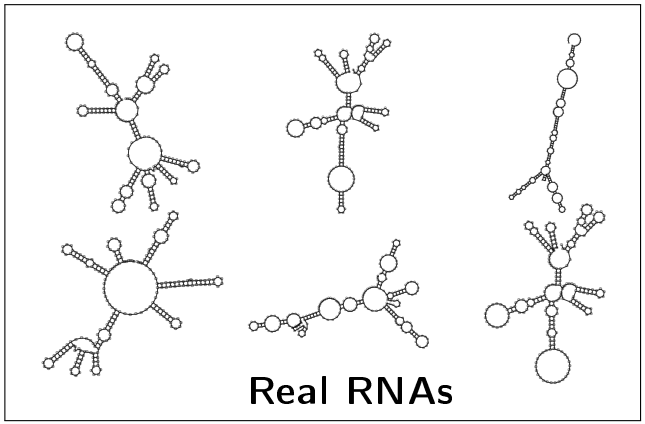
Example: RNA secondary structures

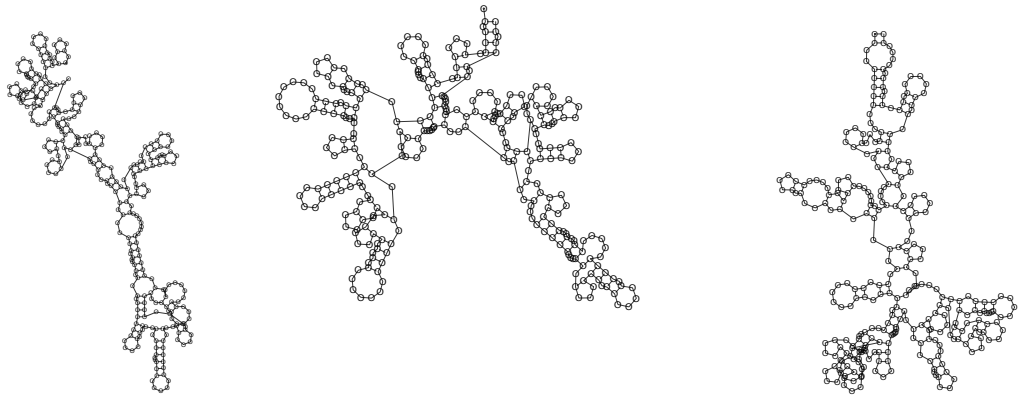


Rationale

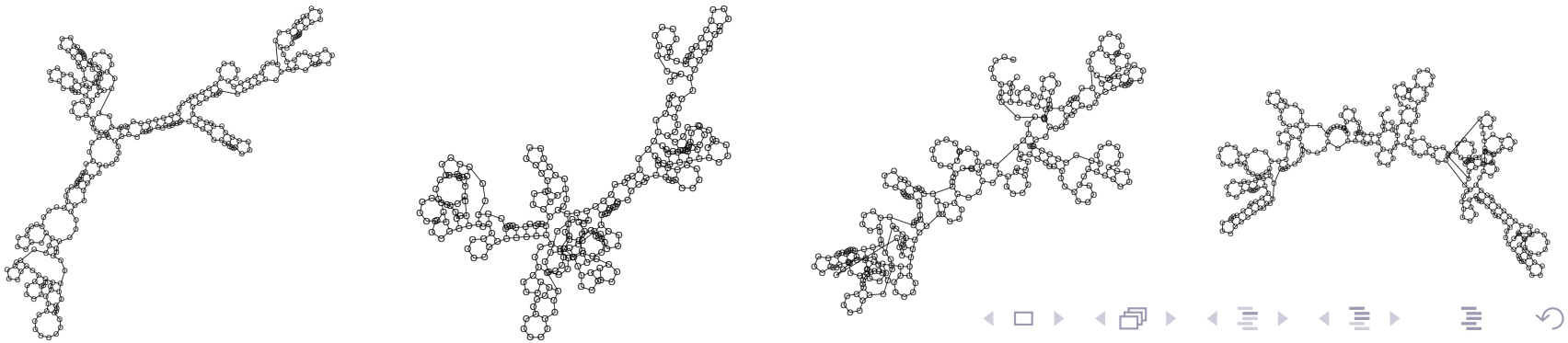
Nature **dislikes** *uniformity (S. Brlek 05)*

Example: RNA secondary structures = *Peakless* Motzkin words





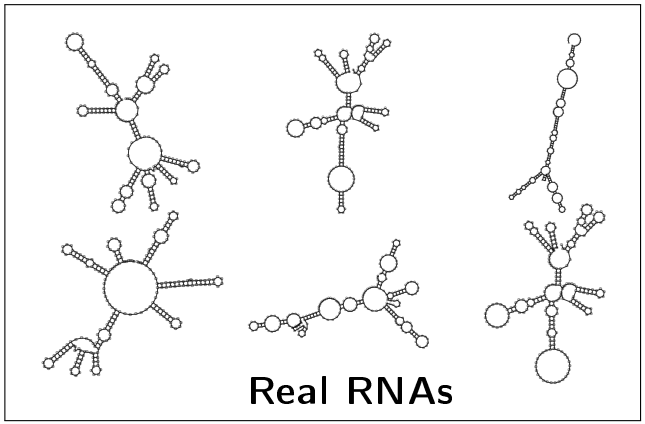
Random uniform RNAs

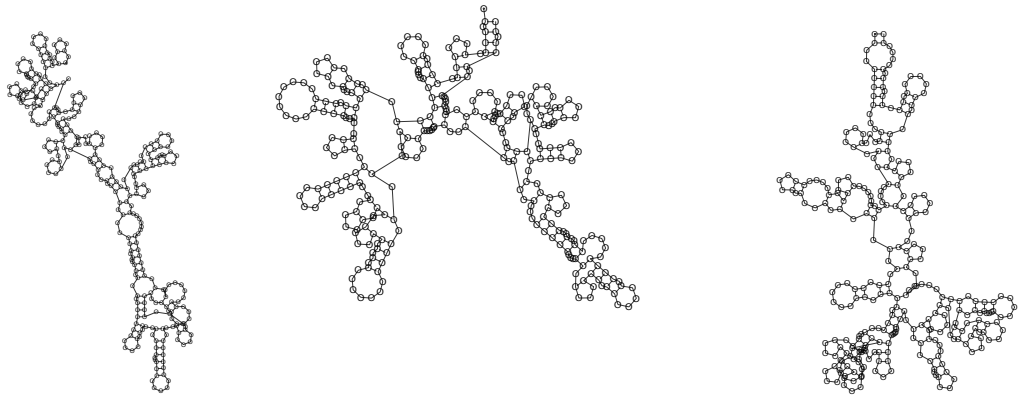


Rationale

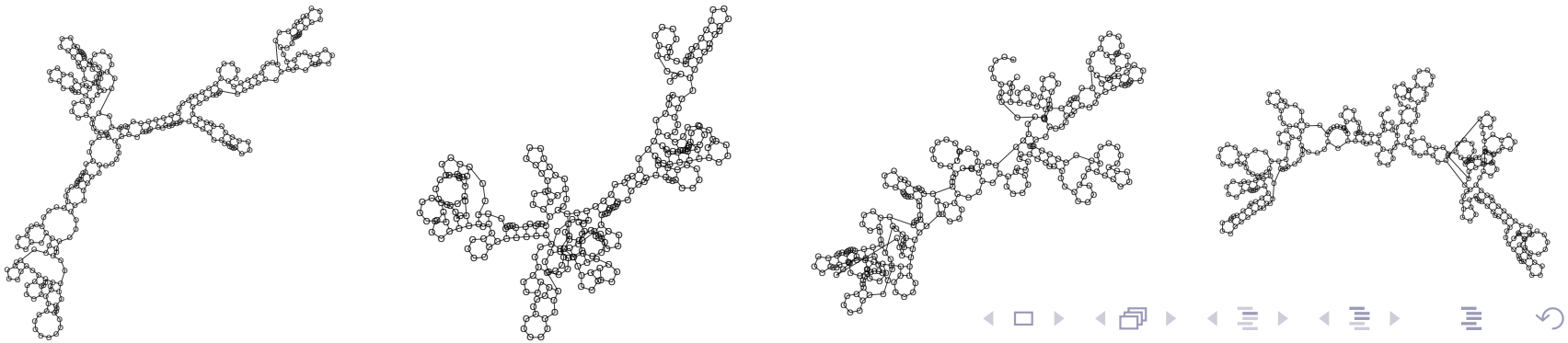
Nature **dislikes** *uniformity (S. Brlek 05)*

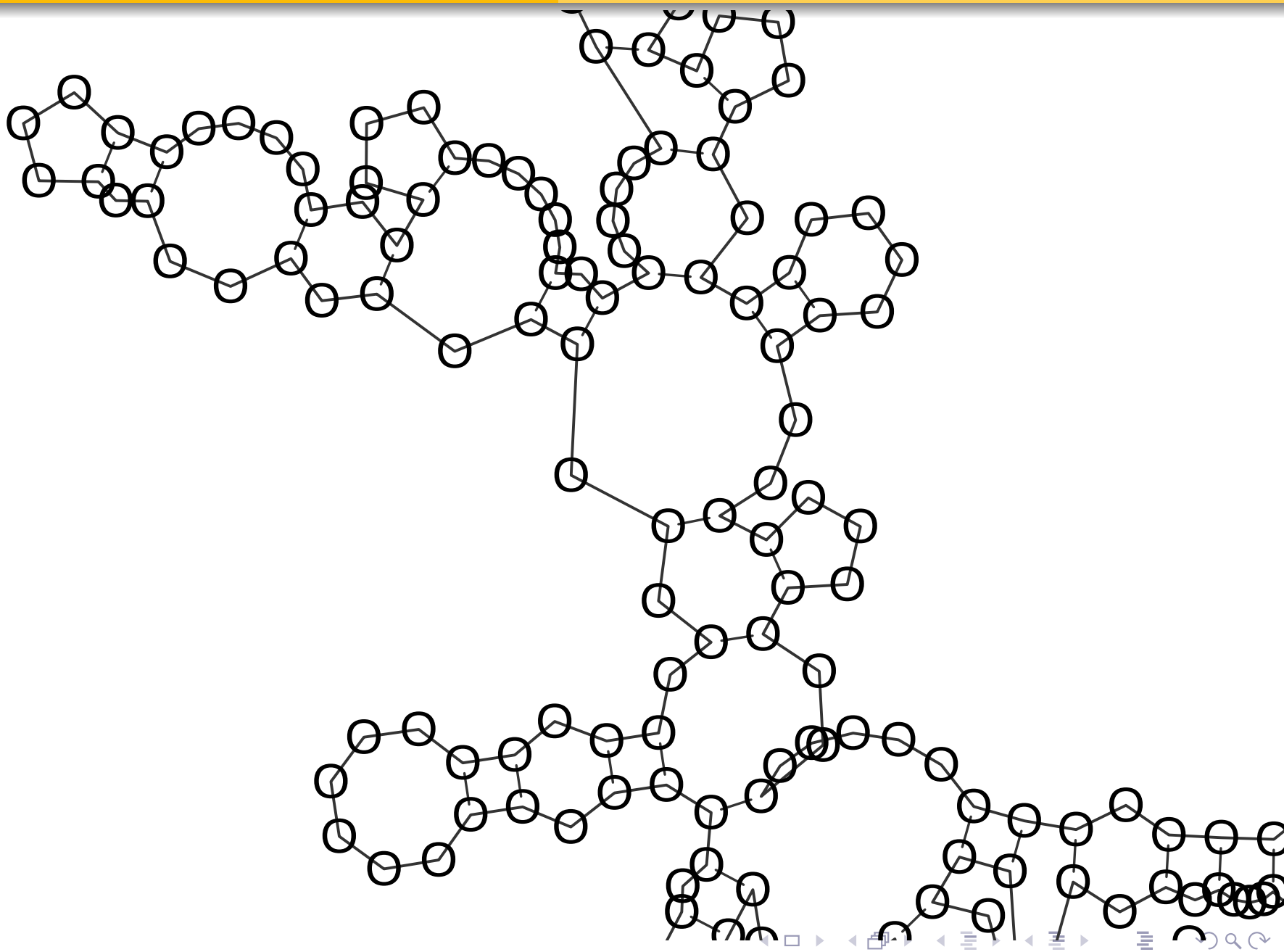
Example: RNA secondary structures = *Peakless* Motzkin words





Random uniform RNAs

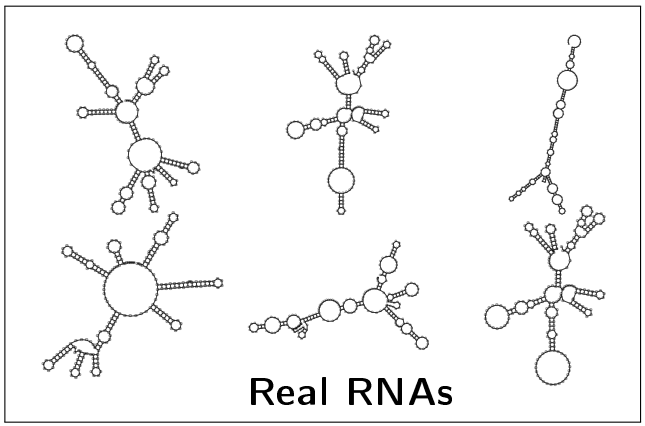


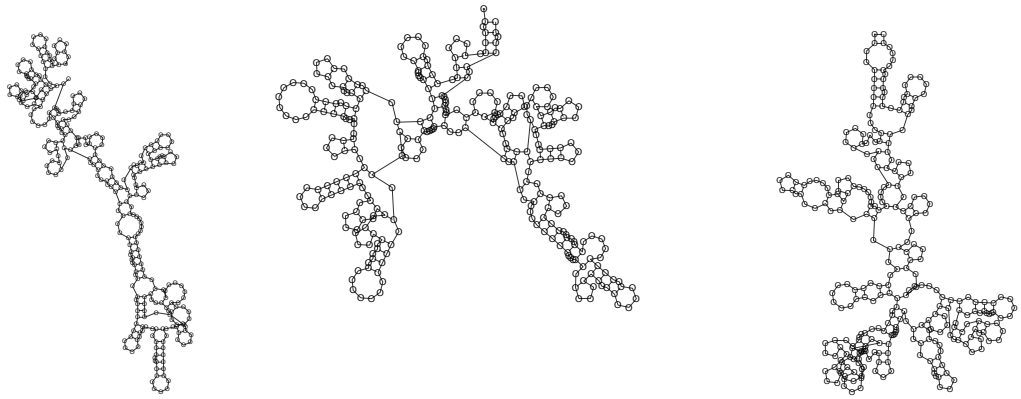


Rationale

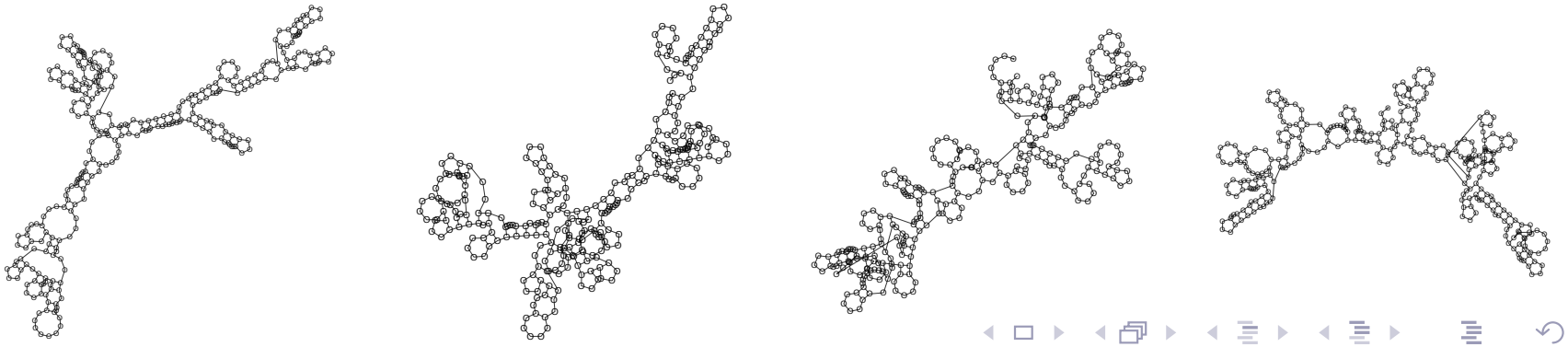
Nature **dislikes** *uniformity (S. Brlek 05)*

Example: RNA secondary structures





Random uniform RNAs



Weighted grammars

Definition (**Weighted** context-free grammar [Denise *et al.*, 2000])

A **weighted** context-free grammar is a **5**-tuple $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S}, \boldsymbol{\pi})$:

- Σ, $\mathcal{N}$, $\mathcal{P}$, $\mathcal{S}$: Same as previously.
- π: **Weight function** $\pi : \Sigma \to \mathbb{R}$.

Definition (Weighted probability distribution)

A WCFG $\mathcal{G}$ implicitly defines a weighted probability distribution $\mathcal{W}$:

$$\forall \omega \in \mathcal{L}(\mathcal{G}),\ \mathbb{P}(\omega) = \frac{\pi(\omega)}{\pi\left(\mathcal{L}(\mathcal{G})\right)}.$$

Definition (Weighted context-free grammar [Denise *et al.*, 2000])

A **weighted** context-free grammar is a **5**-tuple $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{P}, \mathcal{S}, \boldsymbol{\pi})$:

- Σ, $\mathcal{N}$, $\mathcal{P}$, $\mathcal{S}$: Same as previously.
- π: **Weight function** $\pi : \Sigma \to \mathbb{R}$.

Definition (Weighted probability distribution)

A WCFG $\mathcal{G}$ implicitly defines a weighted probability distribution $\mathcal{W}$:

$$\forall \omega \in \mathcal{L}(\mathcal{G}),\ \mathbb{P}(\omega) = \frac{\pi(\omega)}{\pi\left(\mathcal{L}(\mathcal{G})\right)}.$$

Generating k words of size n is in $\mathcal{O}(n^2 + n\log(n).k)^*$.

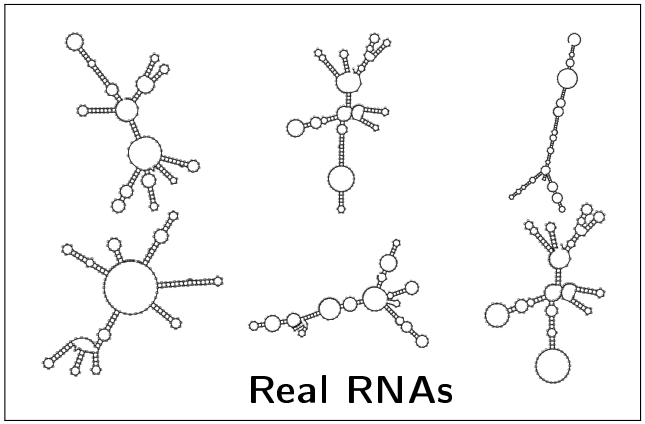
Furthermore, aiming at **observed** terminal frequencies:

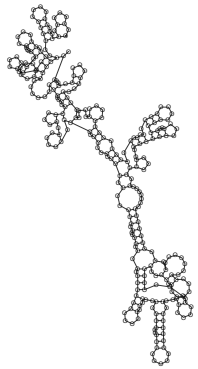
$\Rightarrow$ Asymptotic weights can *sometimes* be computed [Denise *et al.*, 2000]

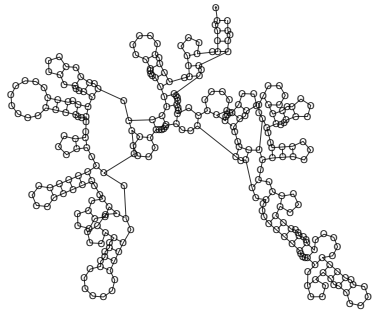
$\Rightarrow$ Weights can be heuristically determined

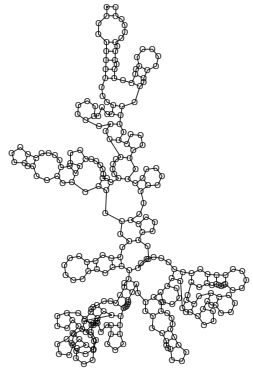
Example

RNA secondary structures

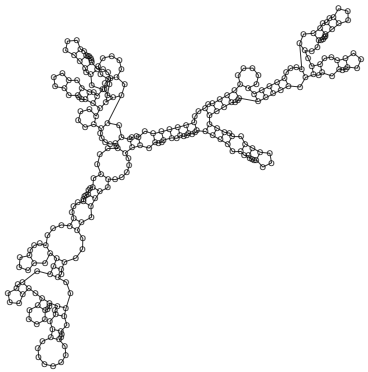


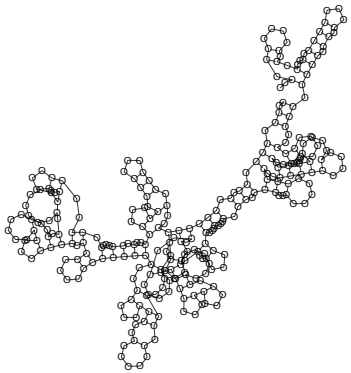


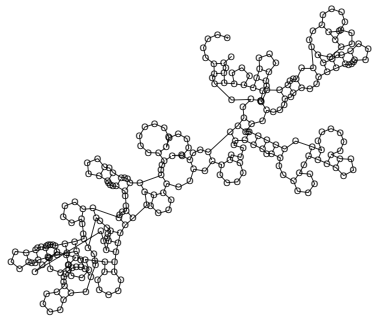


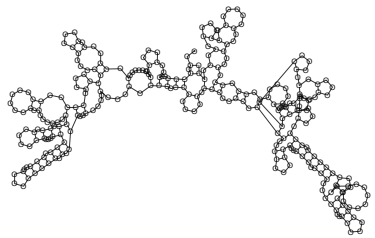


Random **uniform** RNAs



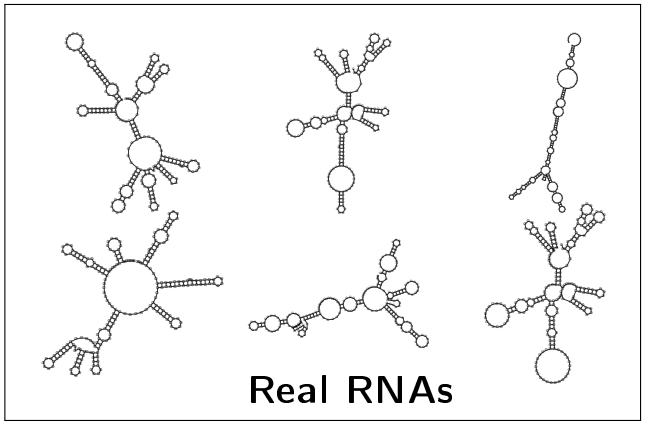


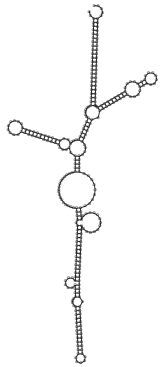


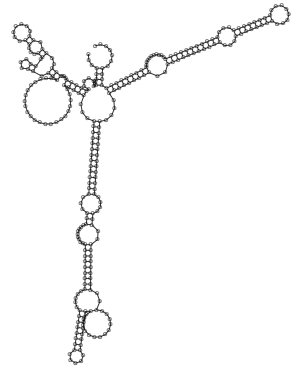


Example

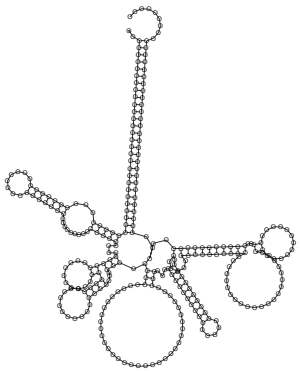
RNA secondary structures

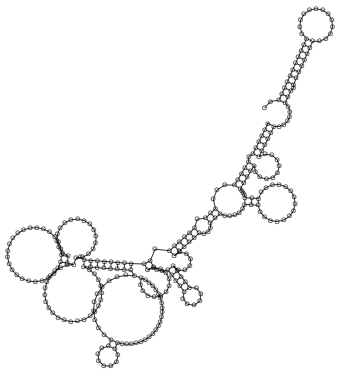


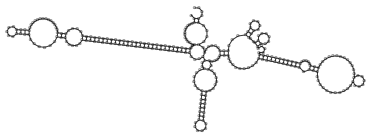


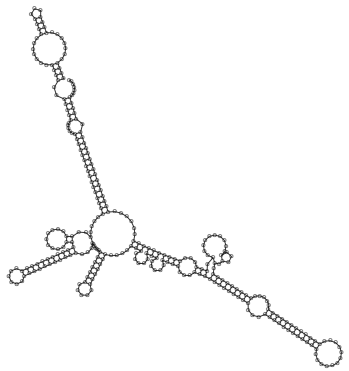


Random **weighted** RNAs









Motivation

In biology: Use random sampling to estimate features of interest.

Example: RNA secondary structures

- Depth/Radius?
- Probability of observing same substructure twice?
- Entropy of the Boltzmann ensemble of low energy?
- ...

Need to eliminate **redundancy** in the recursive generation:

1. No additional information.
2. Mixed performances for generating k **distinct** words with a rejection approach.

Motivation

In biology: Use random sampling to estimate features of interest.

Example: RNA secondary structures

- Depth/Radius?
- Probability of observing same substructure twice?
- Entropy of the Boltzmann ensemble of low energy?
- ...

Need to eliminate **redundancy** in the recursive generation:

1. No additional information.
2. Mixed performances for generating k **distinct** words with a rejection approach.

Motivation

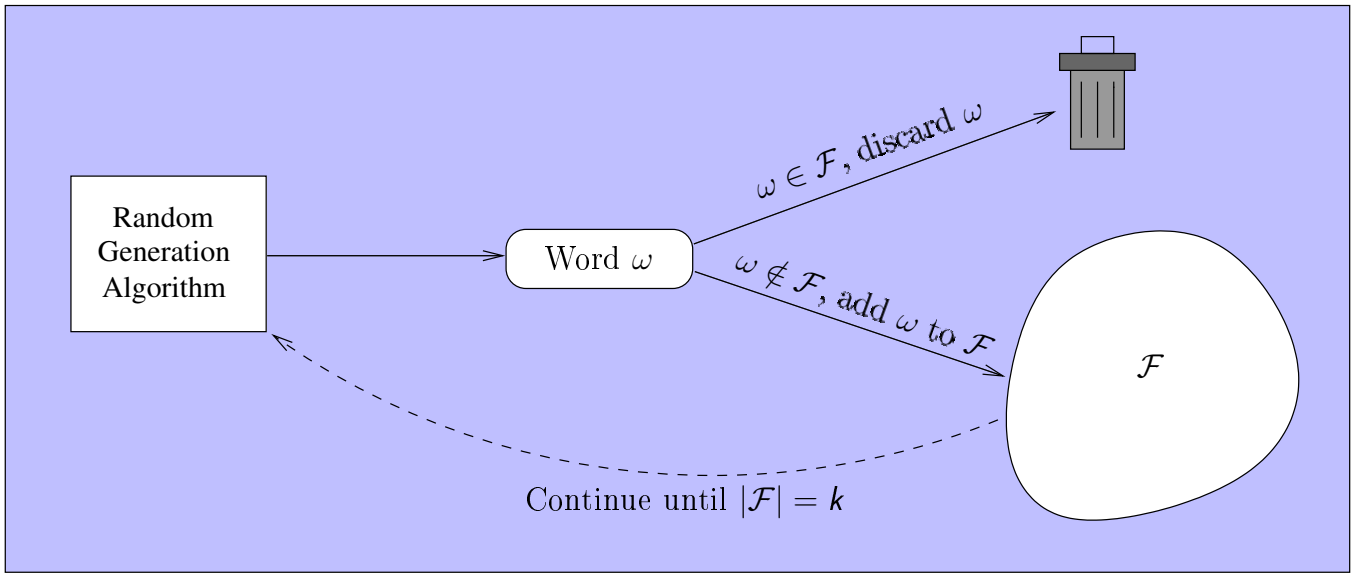
In biology: Use random sampling to estimate features of interest.

Example: RNA secondary structures

- Depth/Radius?
- Probability of observing same substructure twice?
- Entropy of the Boltzmann ensemble of low energy?
- ...

Need to eliminate **redundancy** in the recursive generation:

1. No additional information.
2. Mixed performances for generating k **distinct** words with a rejection approach.



The uniform case

Generating k distinct samples (or `PowerSet` of size k) takes an expected number of attempts in $\mathcal{O}(k \log k)$ with a rejection approach.

Argument: Expected number of attempts is strictly increasing with k.
+ Bounded by $\Theta(k \log k)$ (Coupon collector)
(Better algorithms found in [Zimmermann, 1995])

Weighted distribution: Claim #1

Rejecting for generating of k distinct words can be **exponential** in k.

$$\begin{array}{rcl} \mathcal{P}: S & \to & a\,S \mid T \\ T & \to & b\,T \mid \varepsilon \end{array} \qquad \boxed{\begin{array}{rcl} \pi(a) & = & 2 \\ \pi(b) & = & 1 \end{array}}$$

Among words of length n:

$$\pi(a^n) = 2^n \qquad \pi(a^{n-1}b) = 2^{n-1} \qquad \dots$$
$$\pi\left(\mathcal{L}(S)\right) = 2^{n+1} - 1$$

Sampling k distinct words implies sampling at least a word from

$$\mathcal{R} = \{a^{n-(k-1)-i}b^{k-1+i}\}_{i\in[0,n-(k-1)]}.$$

Since $\pi(\mathcal{R}) = 2^{n-k+2} - 1$, then a word from $\mathcal{R}$ is drawn after $\Theta(2^k)$ attempts on the average.

Simple type grammars

Weighted distribution: Claim #2

However, for simple type grammars, the probabilities associated with every words are exponentially decreasing on n.

Assume that $\mathcal{G}$ is a *simple type* grammar (aperiodic, strongly connected), whose heaviest word ω_n^* of length n is such that:

$$\pi(\omega_n^*) \sim \kappa\alpha^n$$

Then, $\pi(\omega_n^*)$ is **exponentially lower** than the weight of the whole language:

$$\pi(\mathcal{L}(\mathcal{G})) \sim \kappa'\alpha'^n n^{-3/2}(1+\mathcal{O}(1/n)),\ \alpha' > \alpha$$

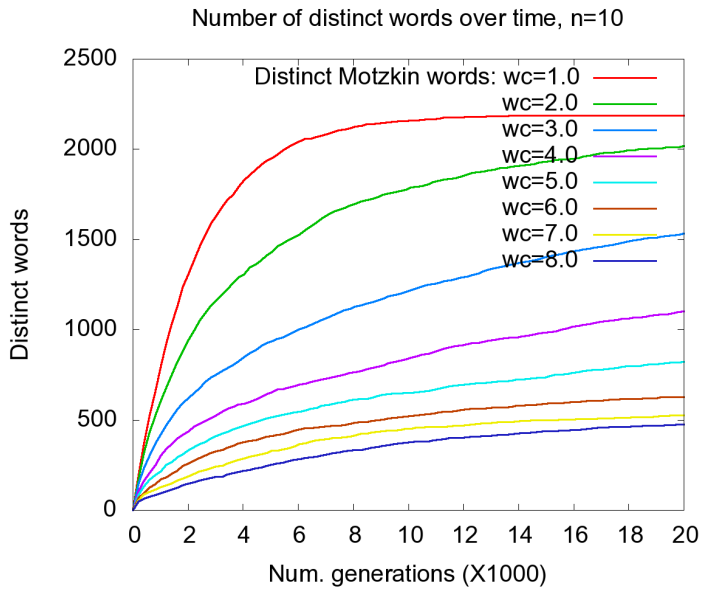
Therefore, generating a polynomial set of words can be performed in an asymptotically linear number of attempts.

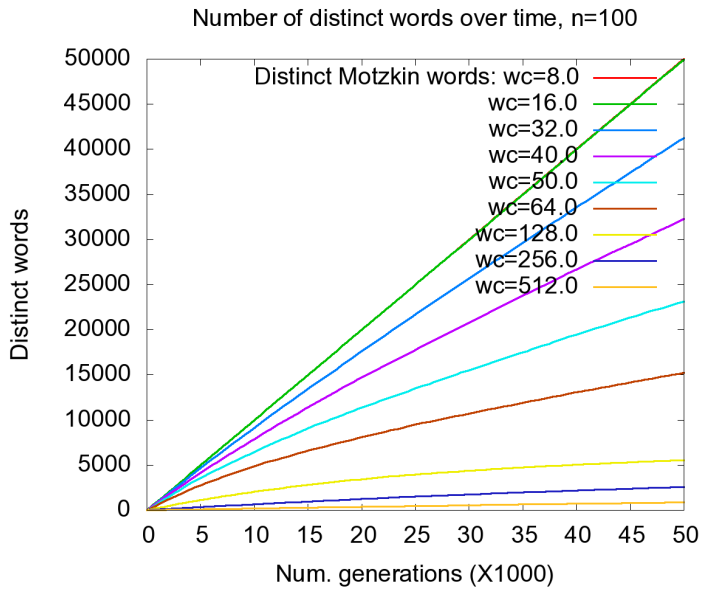
Weight dependency

Weighted distribution: Claim #3

Weights involve non-negligible constant factors in the weighted generation of k distinct words.

Example: Motzkin words





Recursive approach [Wilf, 1977]:

- Perform local probabilistic choices with probabilities proportional to numbers (resp. weights) of accessible words.

▸ Example

- Cardinalities can be precomputed recursively.

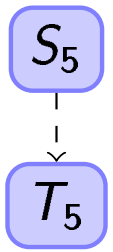
▸ Example

$S \to \varepsilon$	$s_n = \begin{cases} 1 & \text{If } n = 0 \\ 0 & \text{Otherwise} \end{cases}$
$S \to t$	$s_n = \begin{cases} 1 & \text{If } n = 1 \\ 0 & \text{Otherwise} \end{cases}$
$S \to T \mid U$	$s_n = t_n + u_n$
$S \to T\ U$	$s_n = \sum_{i=0}^{n} t_i . u_{n-i}$

Example (Weighted Motzkin words):

Return

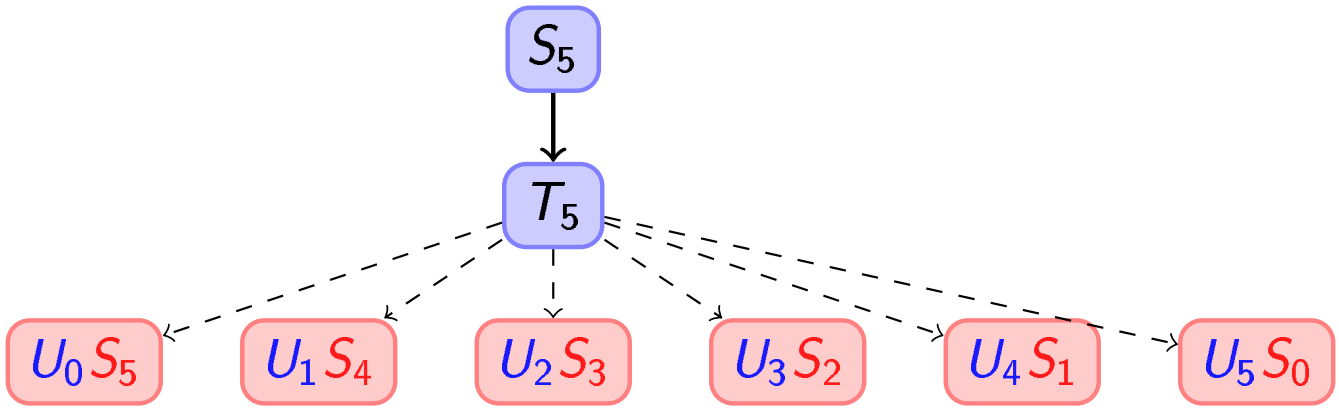
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

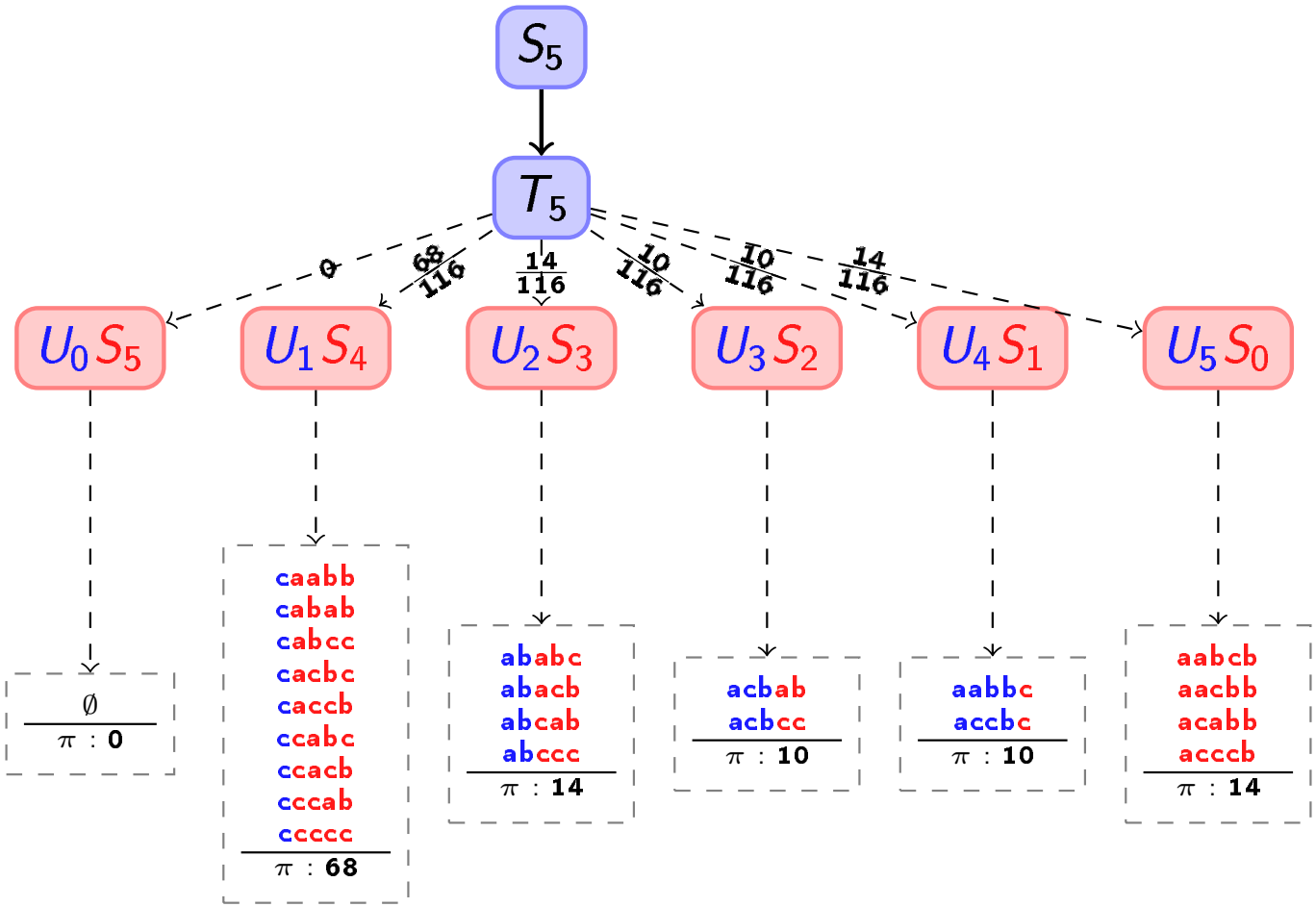
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

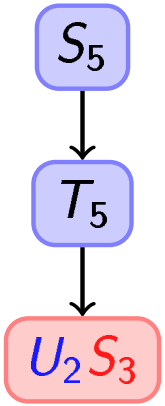
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

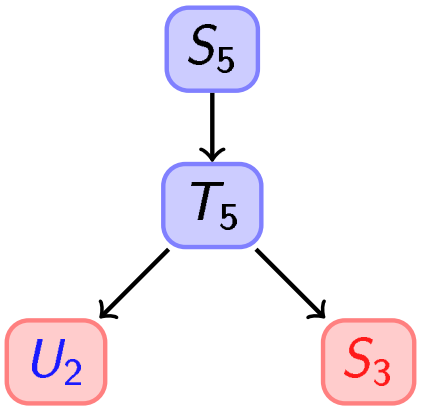
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

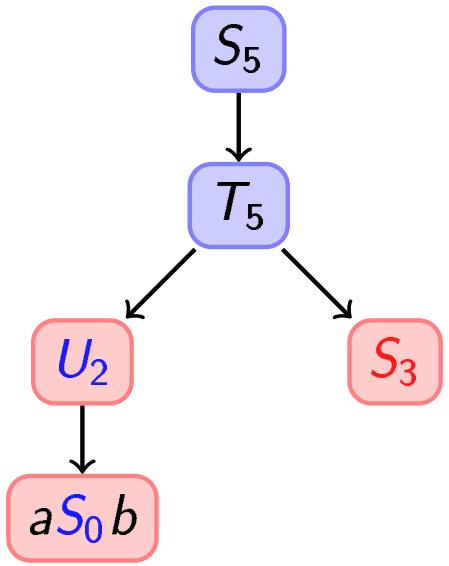
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

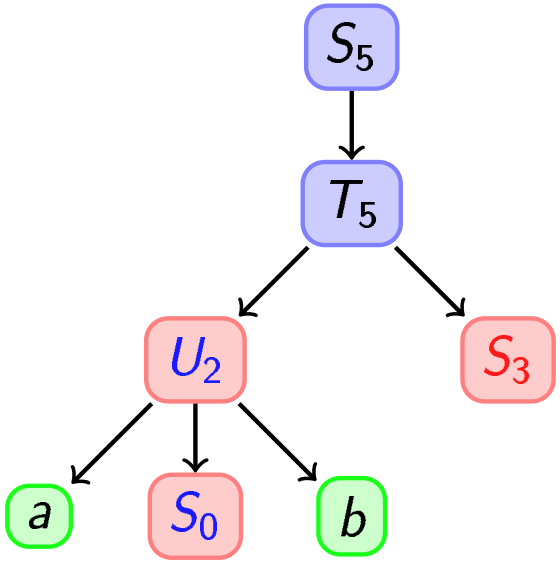
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

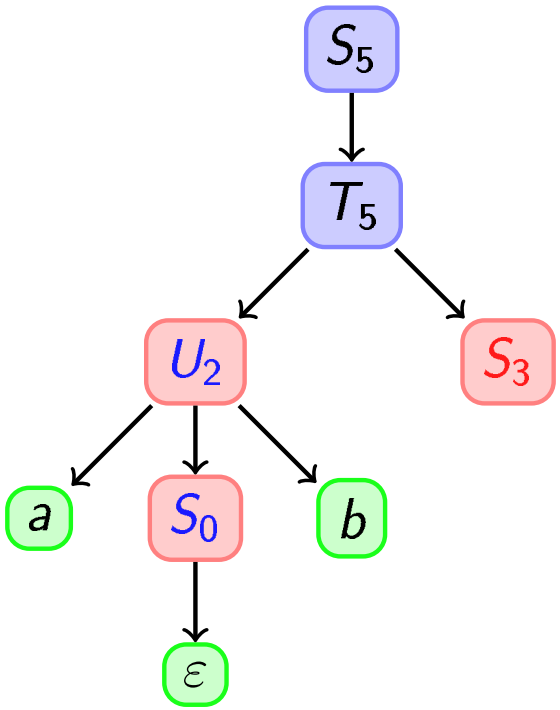
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

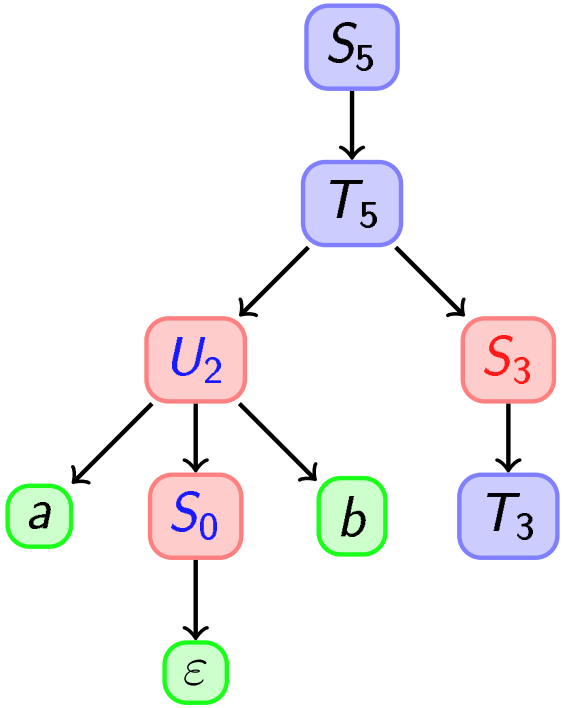
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

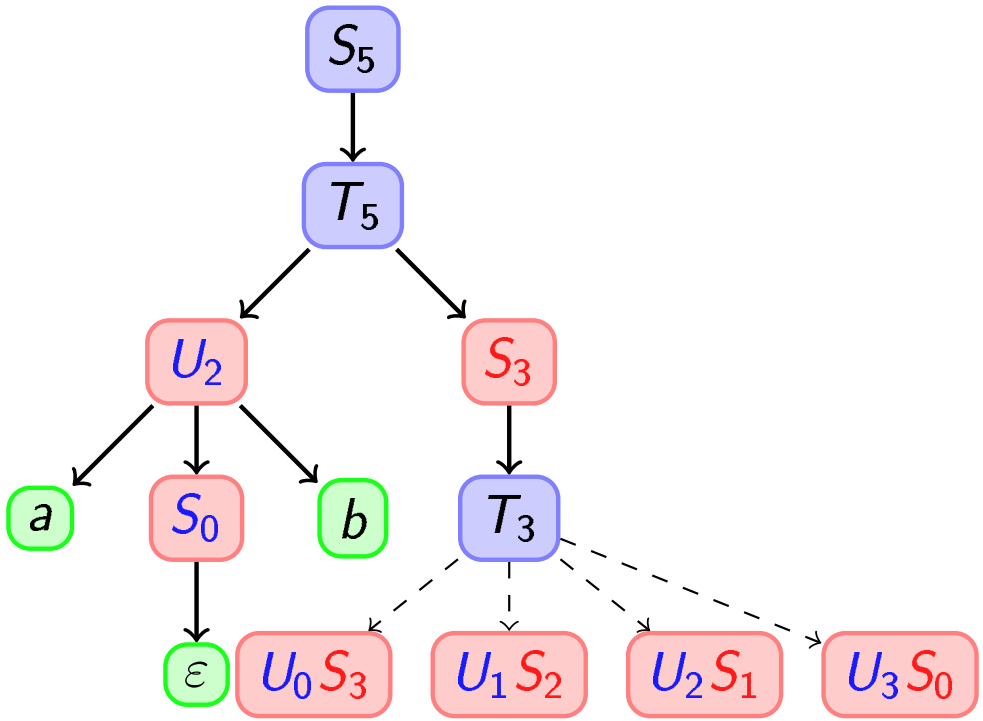
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

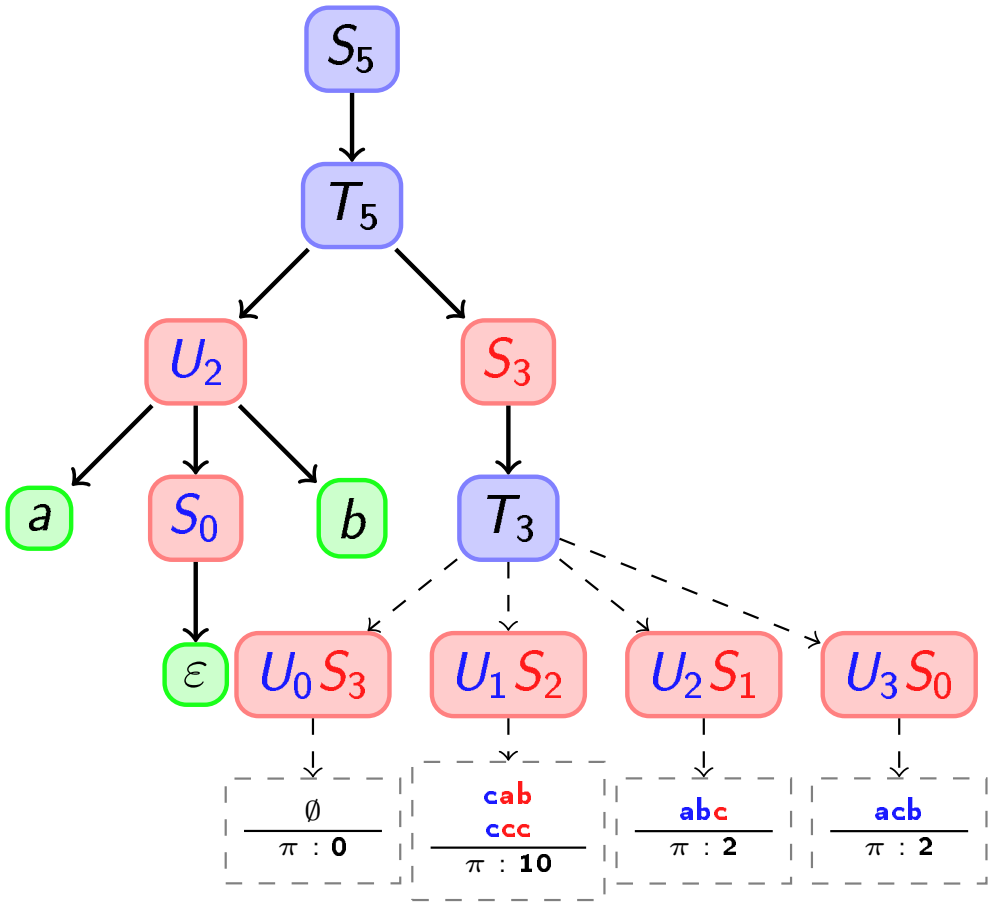
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

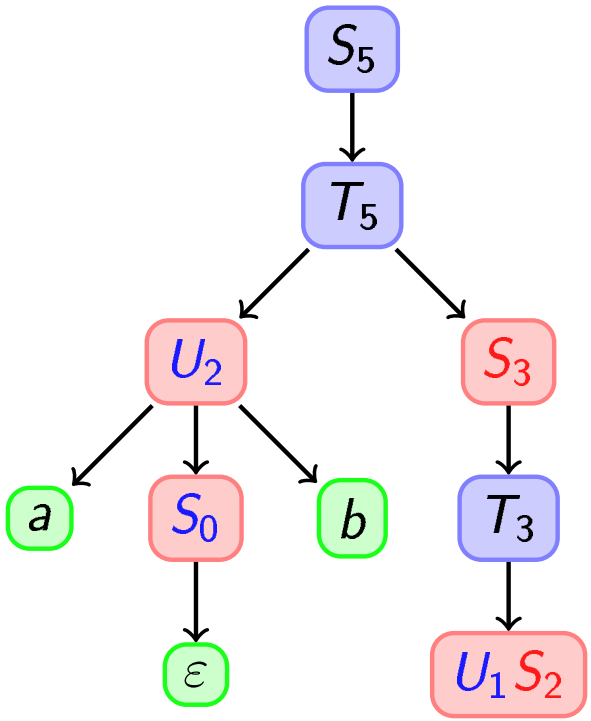
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

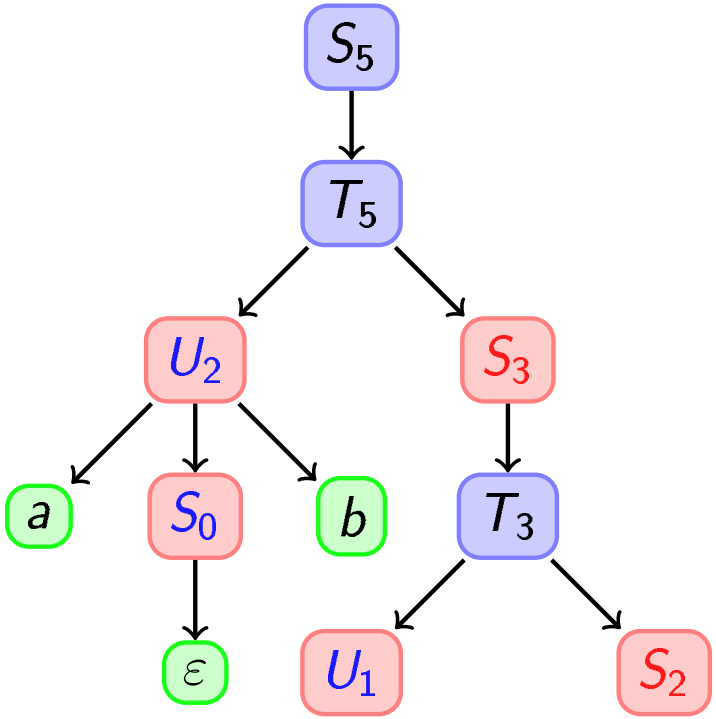
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

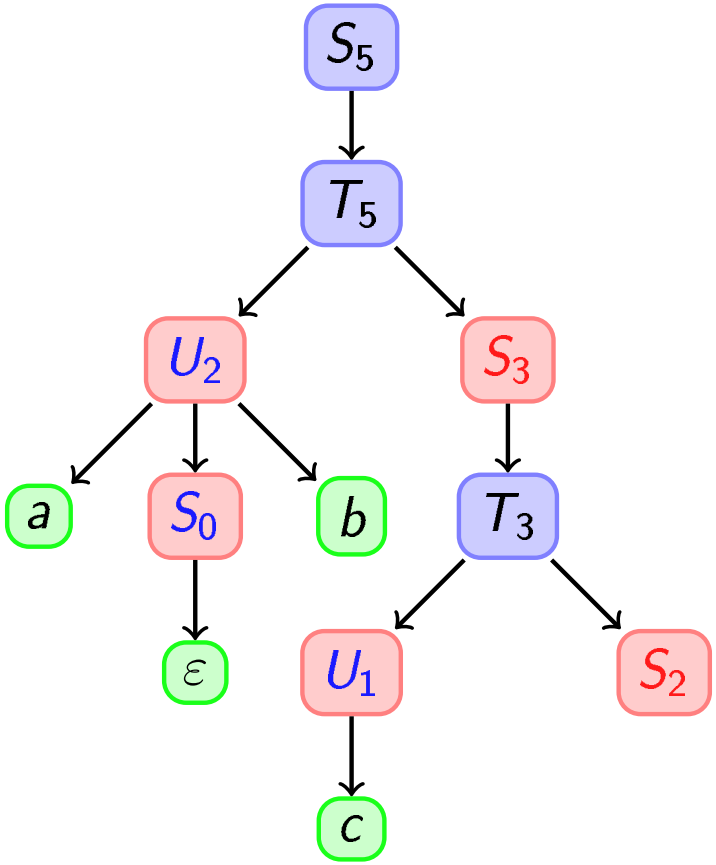
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

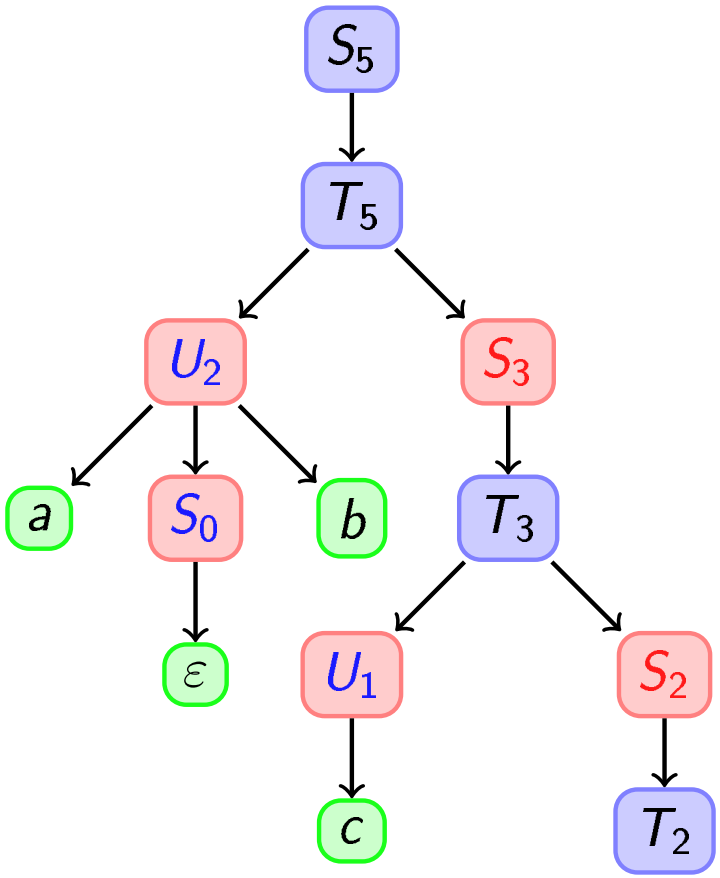
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

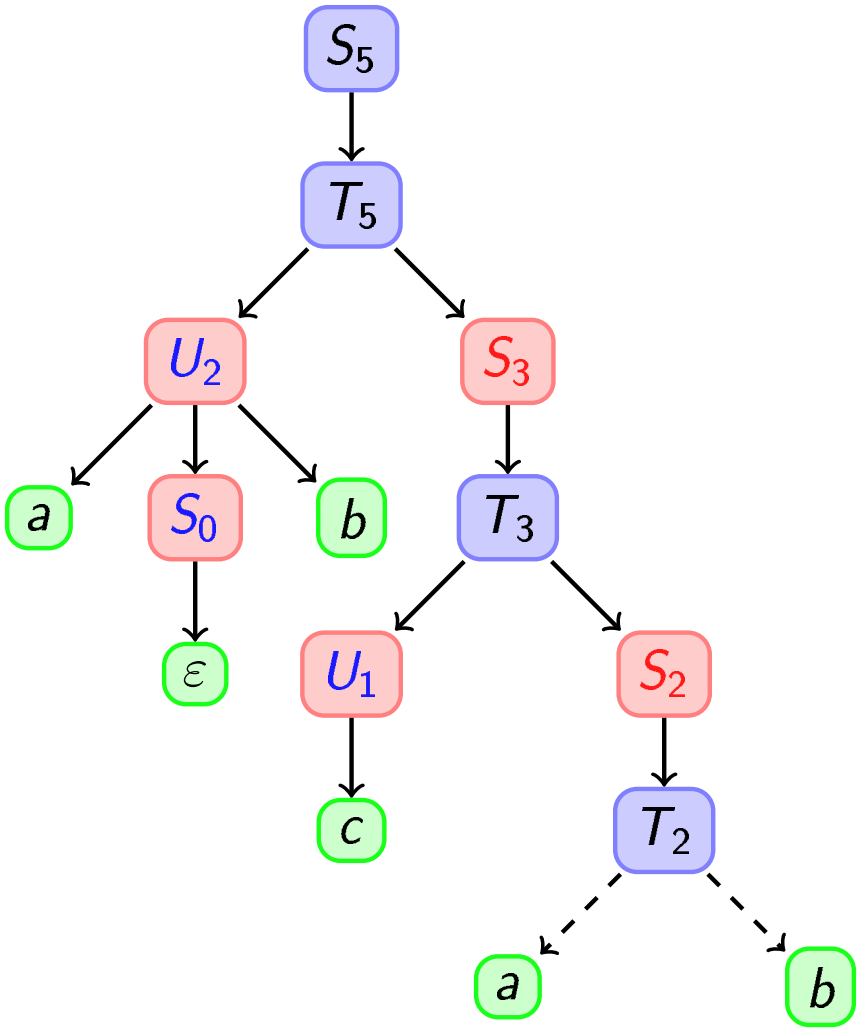
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Recursive approach [Wilf, 1977]:

- Perform local probabilistic choices with probabilities proportional to numbers (resp. weights) of accessible words.

▸ Example

- (Weighted) Cardinalities can be precomputed recursively.

▸ Example

$S \to \varepsilon$	$s_n = \begin{cases} 1 & \text{If } n = 0 \\ 0 & \text{Otherwise} \end{cases}$
$S \to t$	$s_n = \begin{cases} 1 & \text{If } n = 1 \\ 0 & \text{Otherwise} \end{cases}$
$S \to T \mid U$	$s_n = t_n + u_n$
$S \to T\ U$	$s_n = \sum_{i=0}^{n} t_i . u_{n-i}$

Recursive approach [Wilf, 1977]:

- Perform local probabilistic choices with probabilities proportional to numbers (resp. weights) of accessible words.

▸ Example

- (Weighted) Cardinalities can be precomputed recursively.

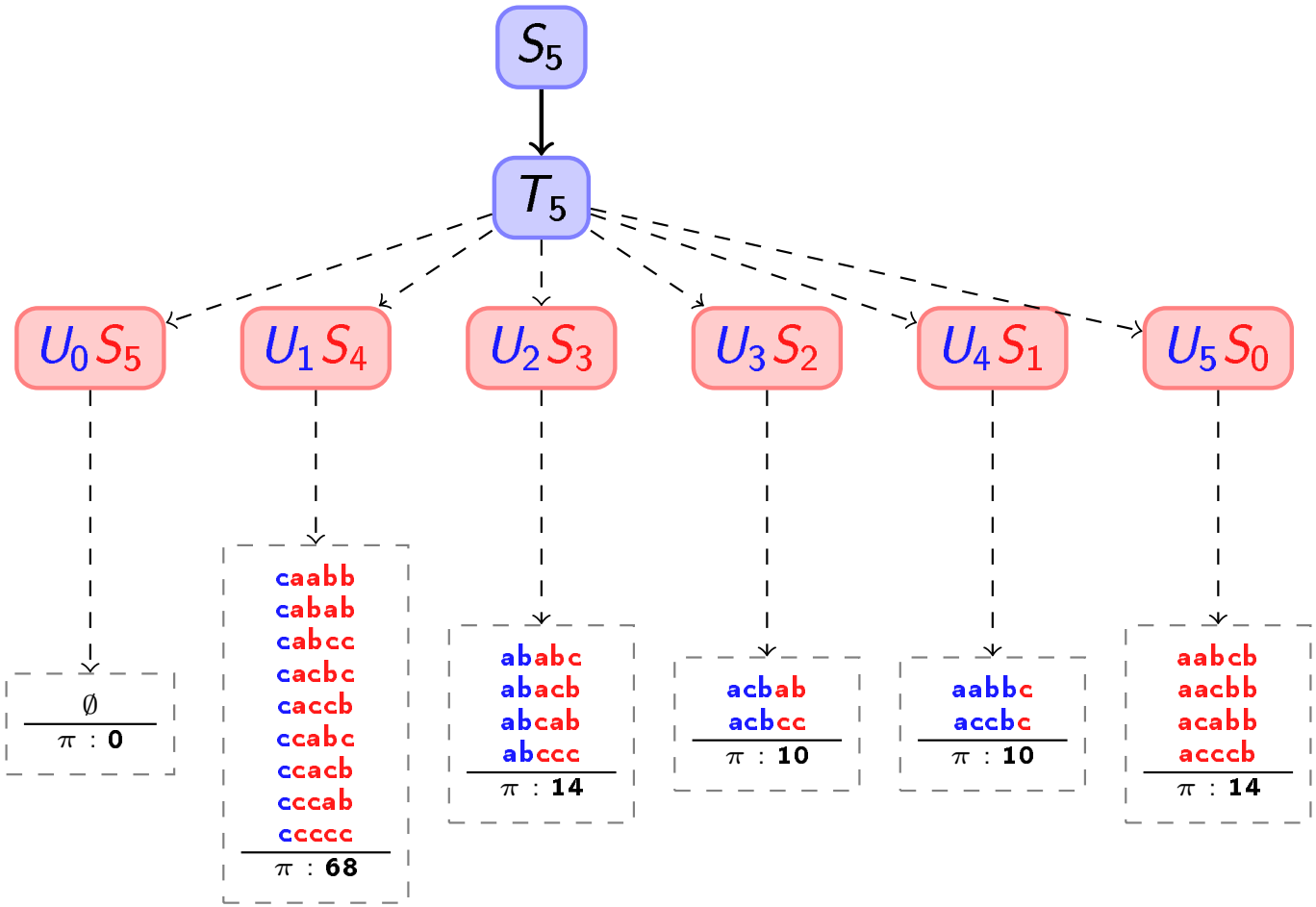
▸ Example

$S \to \varepsilon$	$s_n = \begin{cases} 1 & \text{If } n = 0 \\ 0 & \text{Otherwise} \end{cases}$
$S \to t$	$s_n = \begin{cases} \pi(t) & \text{If } n = 1 \\ 0 & \text{Otherwise} \end{cases}$
$S \to T \mid U$	$s_n = t_n + u_n$
$S \to T\ U$	$s_n = \sum_{i=0}^{n} t_i . u_{n-i}$

Example (Weighted Motzkin words):

Return

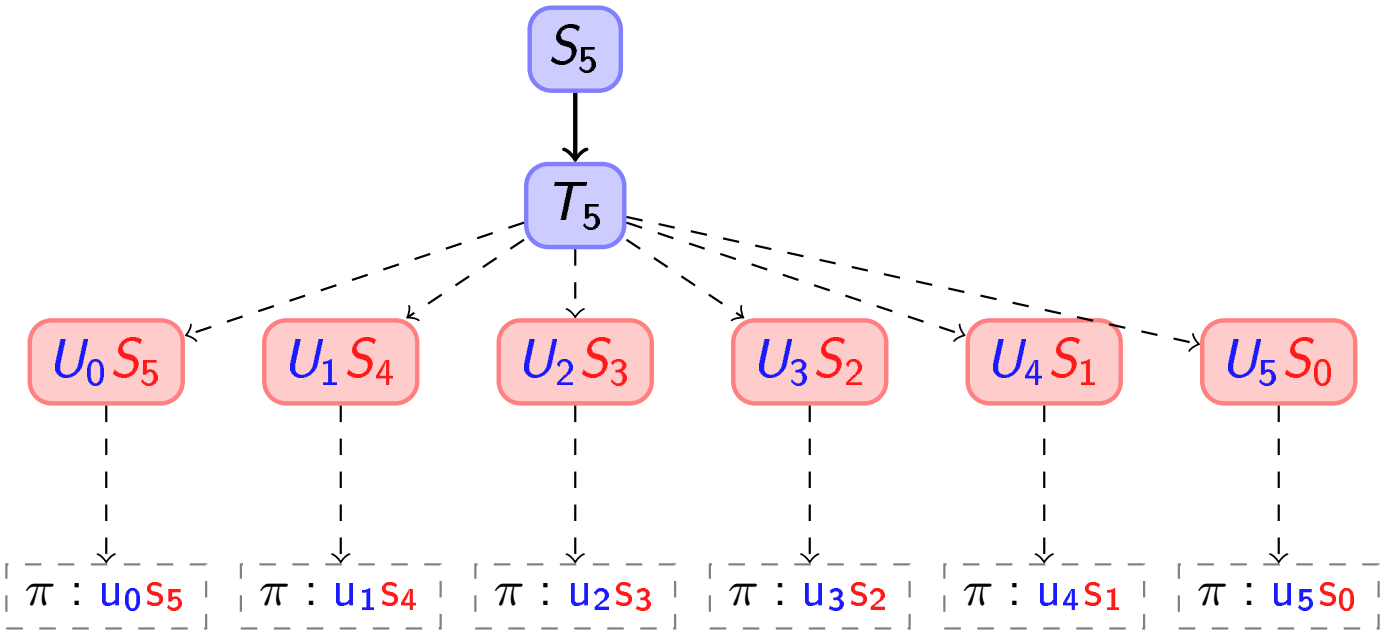
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Main principle

Let $\mathcal{F} \subset \mathcal{L}(\mathcal{G})$ be the set of **forbidden** words.

Goal: Generate from $\mathcal{L}(\mathcal{G})/\mathcal{F}$ in the weighted distribution. ▸ Example

Problem: We cannot simply modify the s_i's !!!
(Same non-terminals occur in different contexts)

Idea

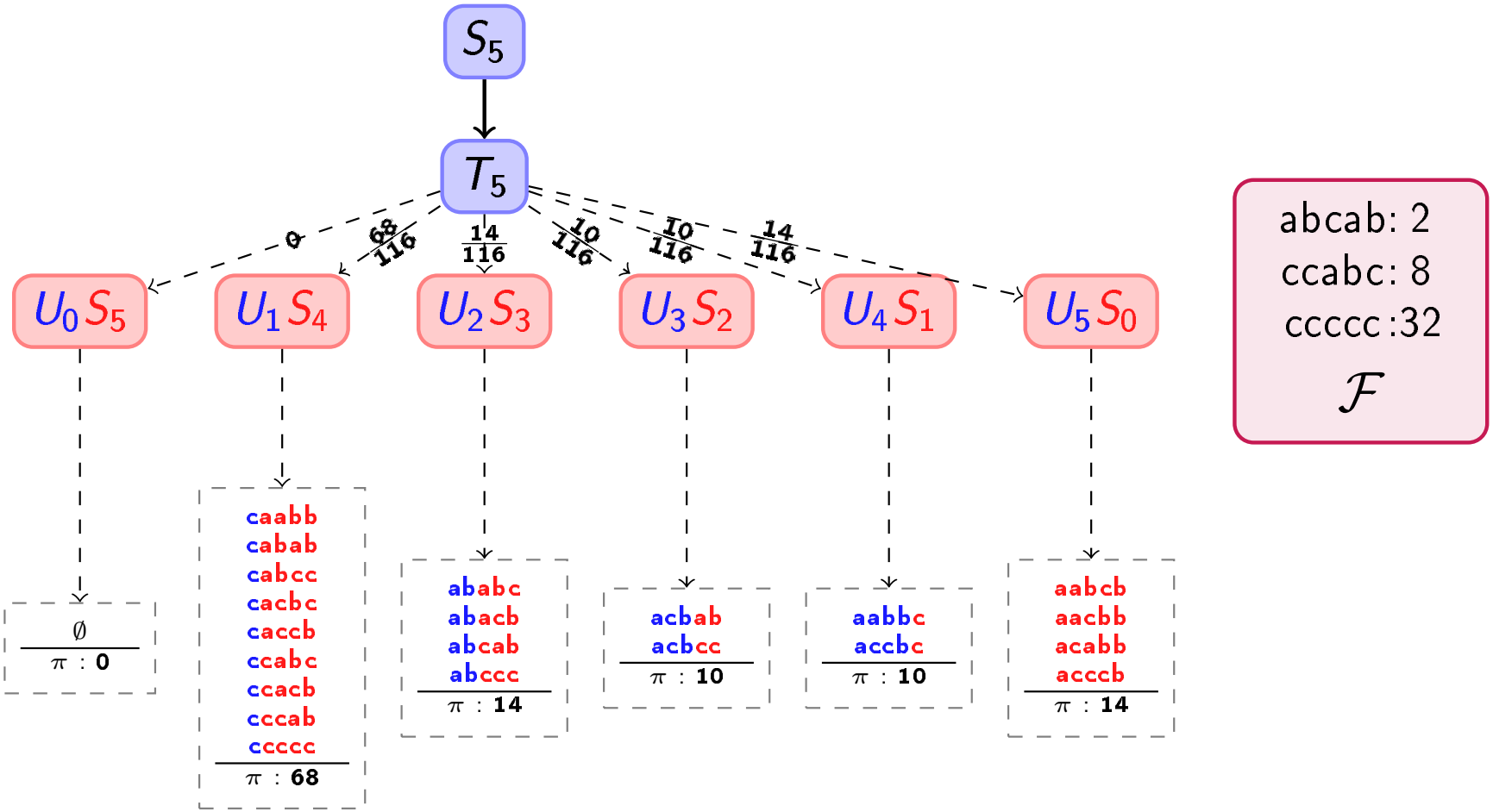
- Capture context by linearizing the generation process. ▸ Example
- Data structure to efficiently subtract contributions from $\mathcal{F}$. ▸ Example

Remark: We can get `PowerSet` by starting from $\mathcal{F} = \emptyset$.

Example (Weighted Motzkin words):

Return

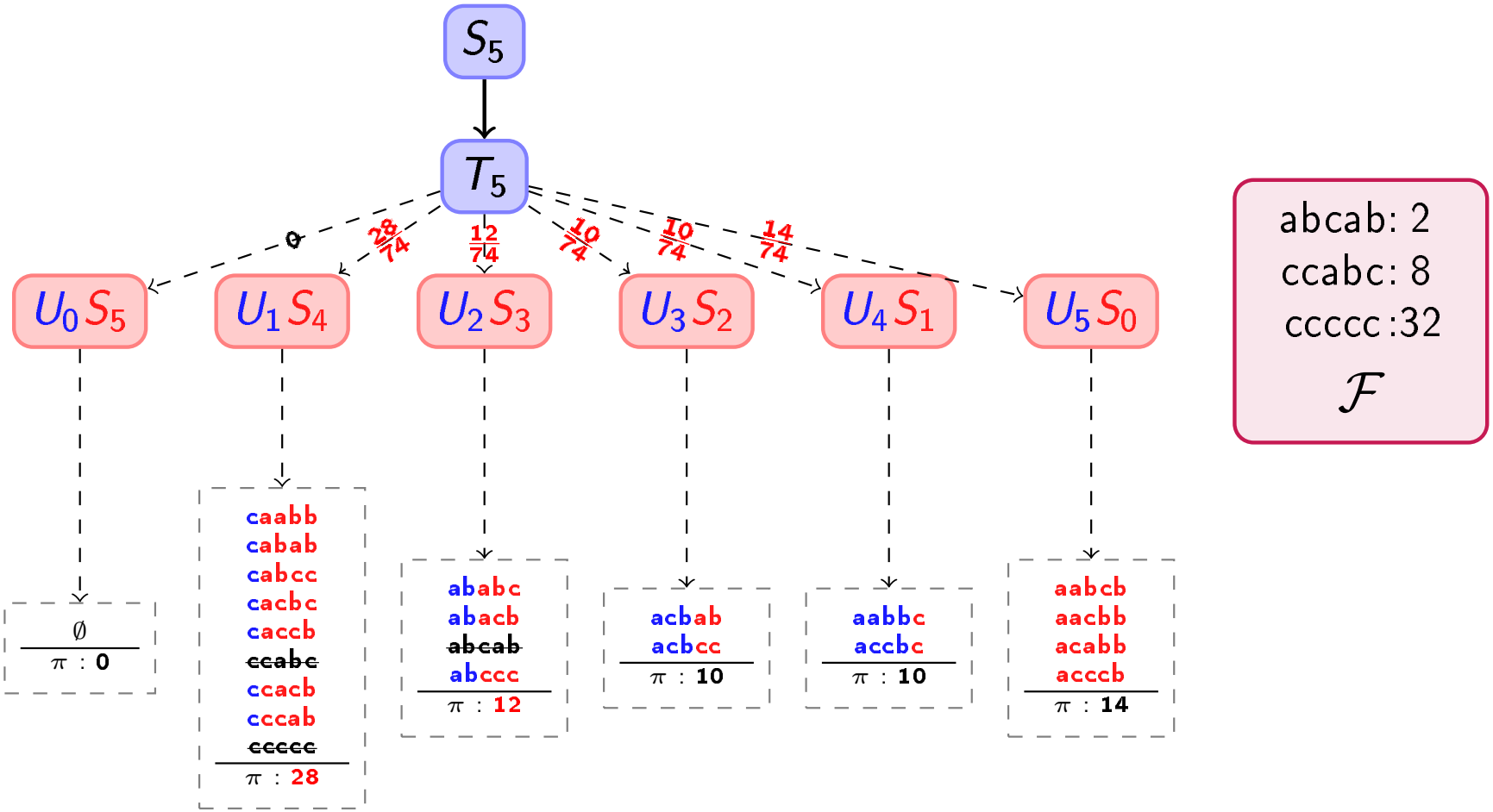
$$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$$



Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Main principle

Let $\mathcal{F} \subset \mathcal{L}(\mathcal{G})$ be the set of **forbidden** words.

Goal: Generate from $\mathcal{L}(\mathcal{G})/\mathcal{F}$ in the weighted distribution. ▸ Example

Problem: We cannot simply modify the s_i's !!!
(Same non-terminals occur in different contexts)

Idea

- Capture context by linearizing the generation process. ▸ Example
- Data structure to efficiently subtract contributions from $\mathcal{F}$. ▸ Example

Remark: We can get `PowerSet` by starting from $\mathcal{F} = \emptyset$.

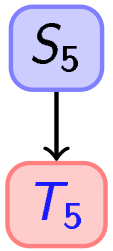
Example (Weighted Motzkin words):

Return

S_5

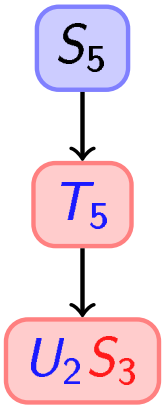
Example (Weighted Motzkin words):

Return



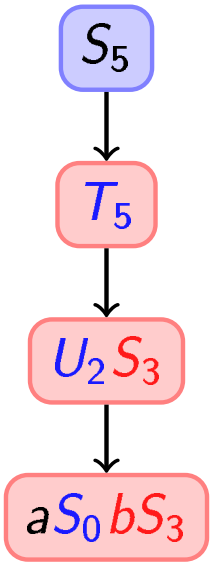
Example (Weighted Motzkin words):

Return



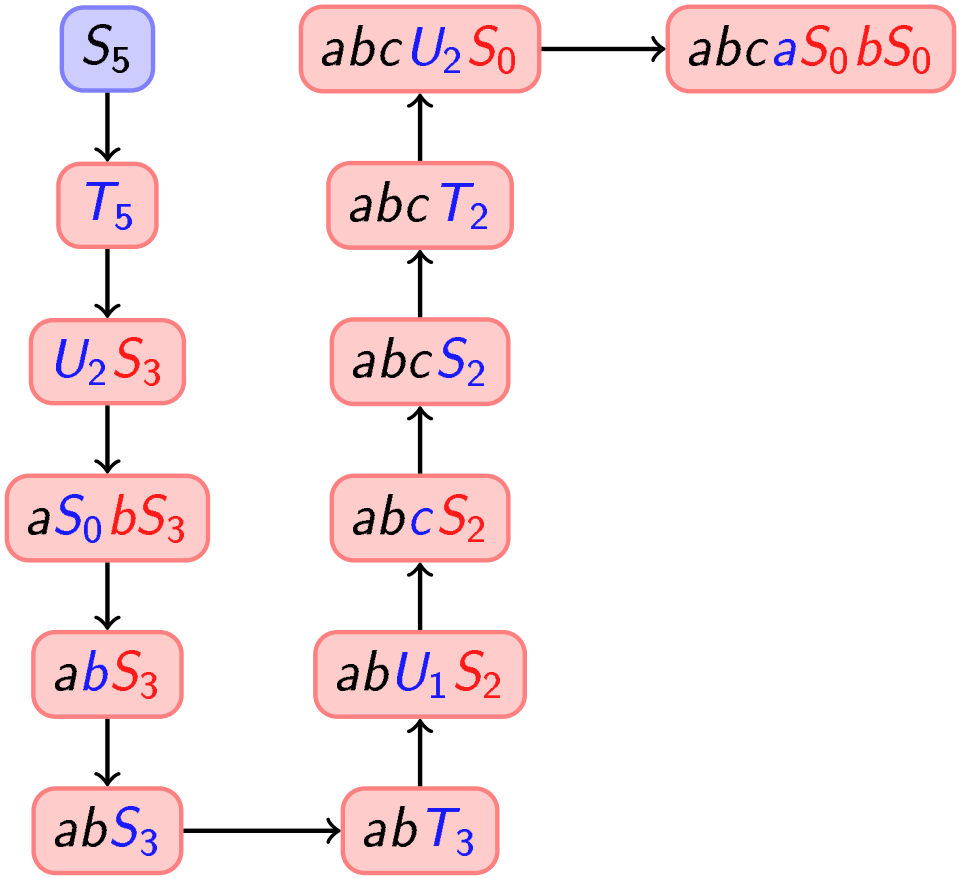
Example (Weighted Motzkin words):

Return



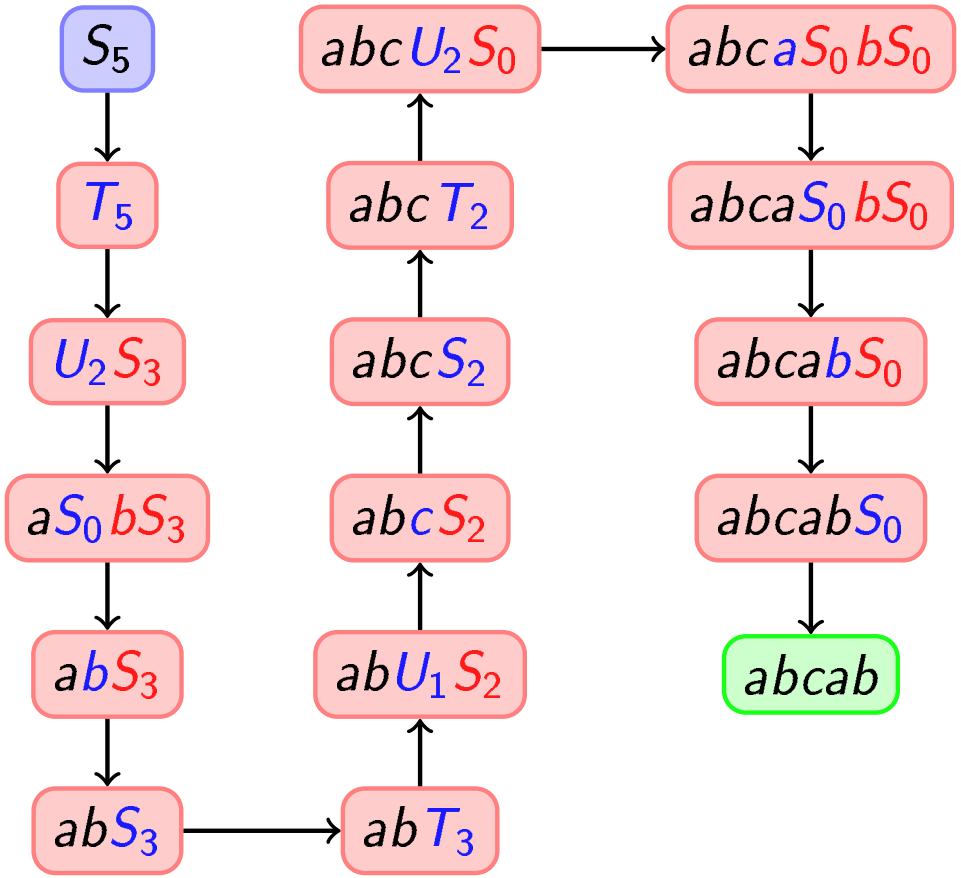
Example (Weighted Motzkin words):

Return



Example (Weighted Motzkin words):

Return



Main principle

Let $\mathcal{F} \subset \mathcal{L}(\mathcal{G})$ be the set of **forbidden** words.

Goal: Generate from $\mathcal{L}(\mathcal{G})/\mathcal{F}$ in the weighted distribution. ▸ Example

Problem: We cannot simply modify the s_i's !!!
(Same non-terminals occur in different contexts)

Idea

- Capture context by linearizing the generation process. ▸ Example
- Data structure to efficiently subtract contributions from $\mathcal{F}$. ▸ Example

Remark: We can get `PowerSet` by starting from $\mathcal{F} = \emptyset$.

Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$

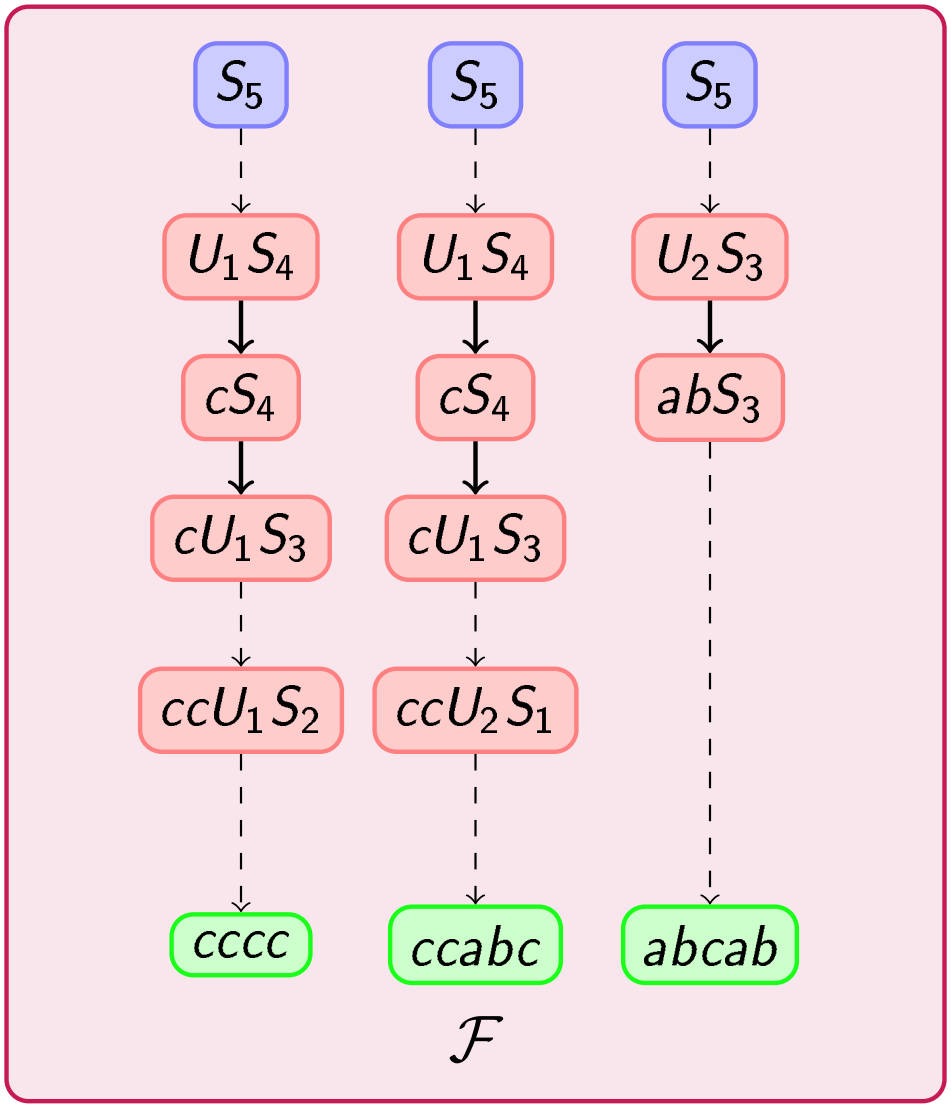
abcab: 2
ccabc: 8
ccccc :32

$\mathcal{F}$

Example (Weighted Motzkin words):

Return

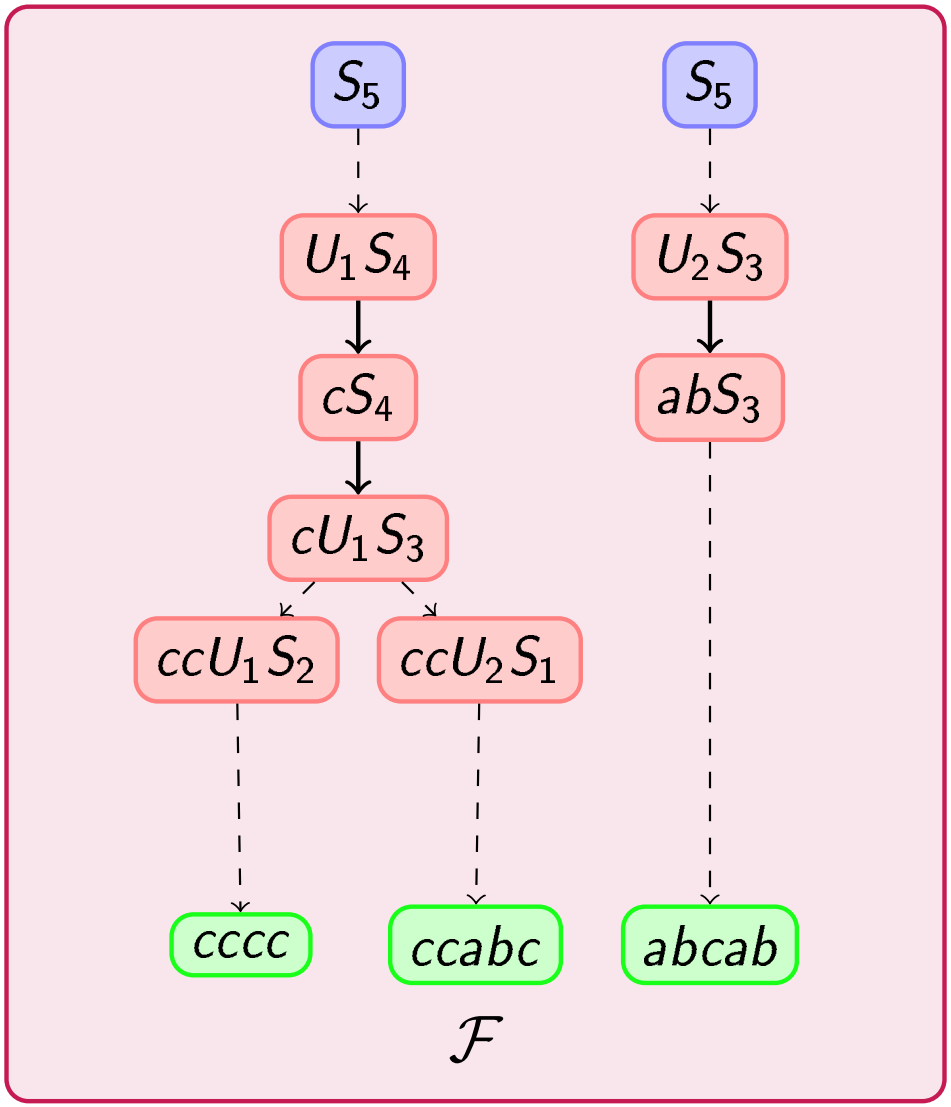
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

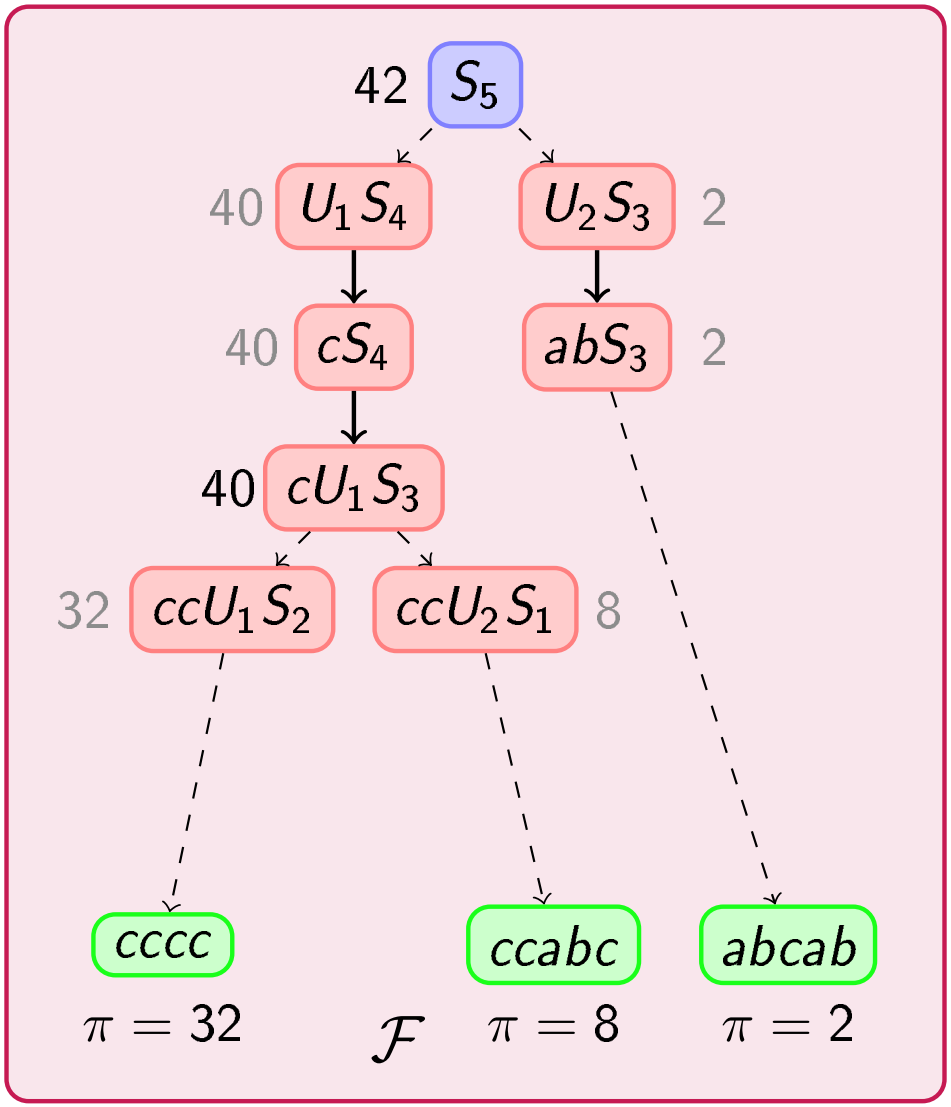
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

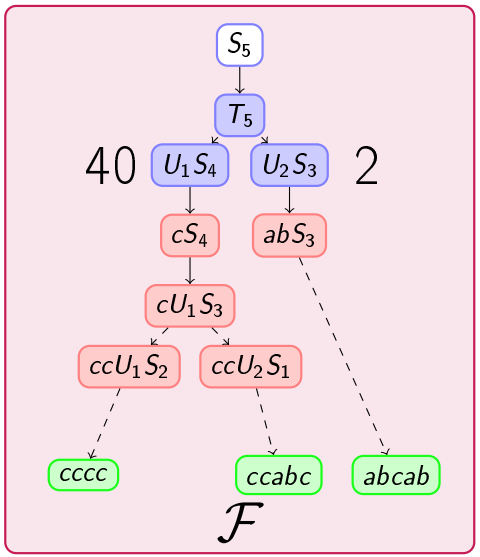
$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$



Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$

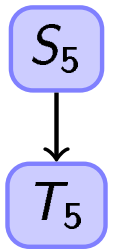


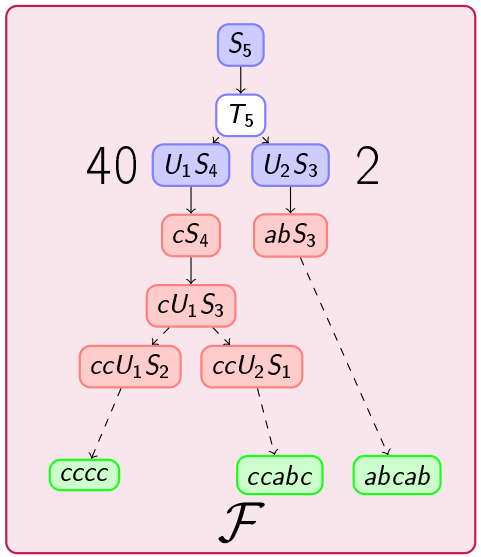
S_5

Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$

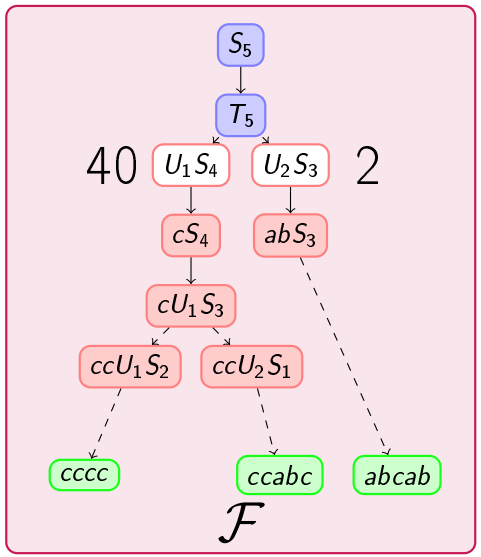


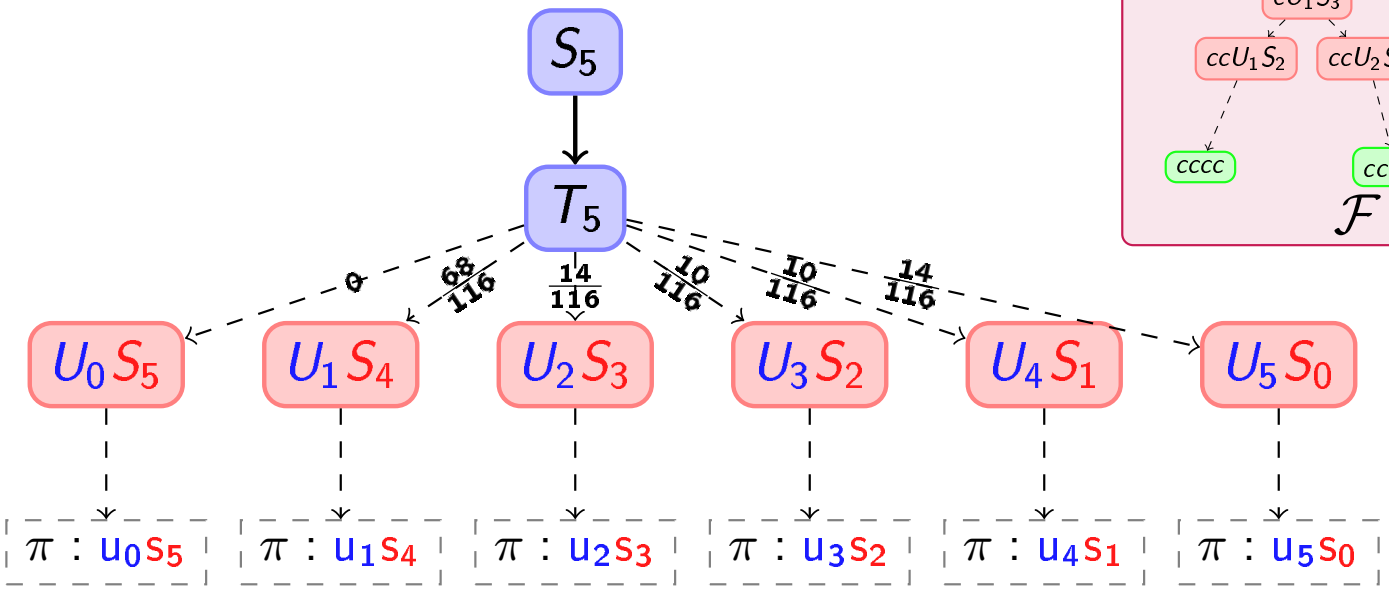


Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$

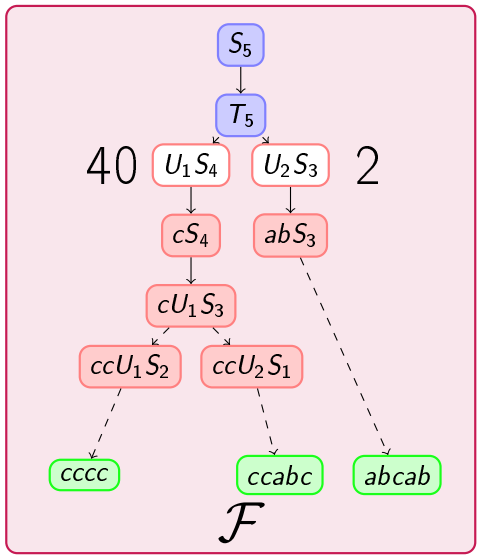


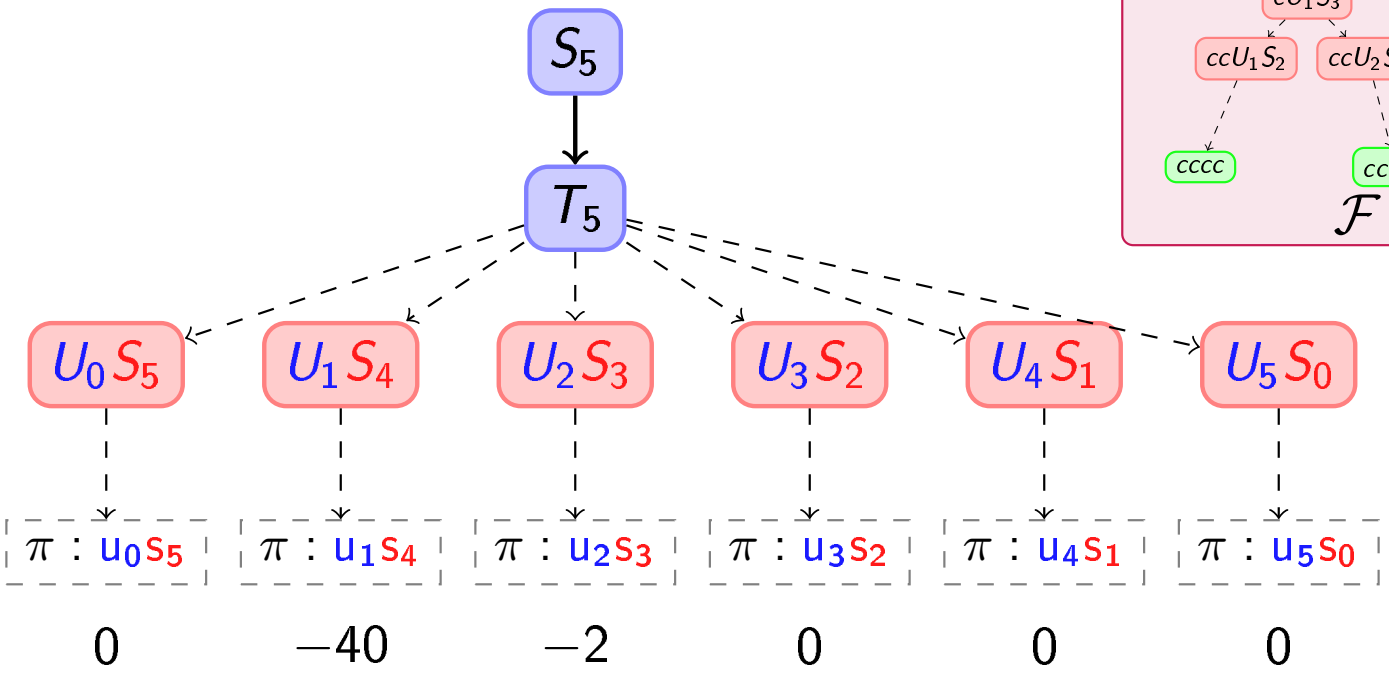


Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$

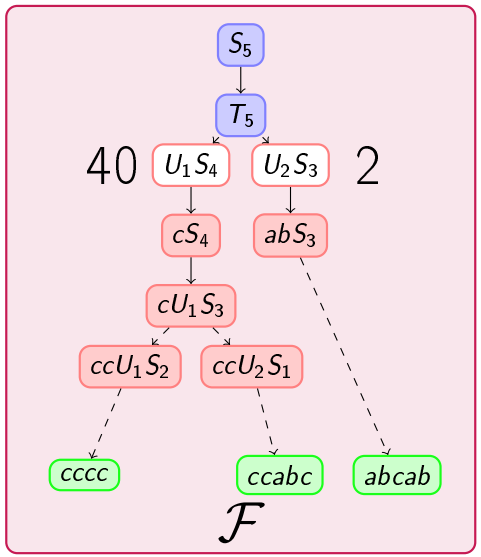


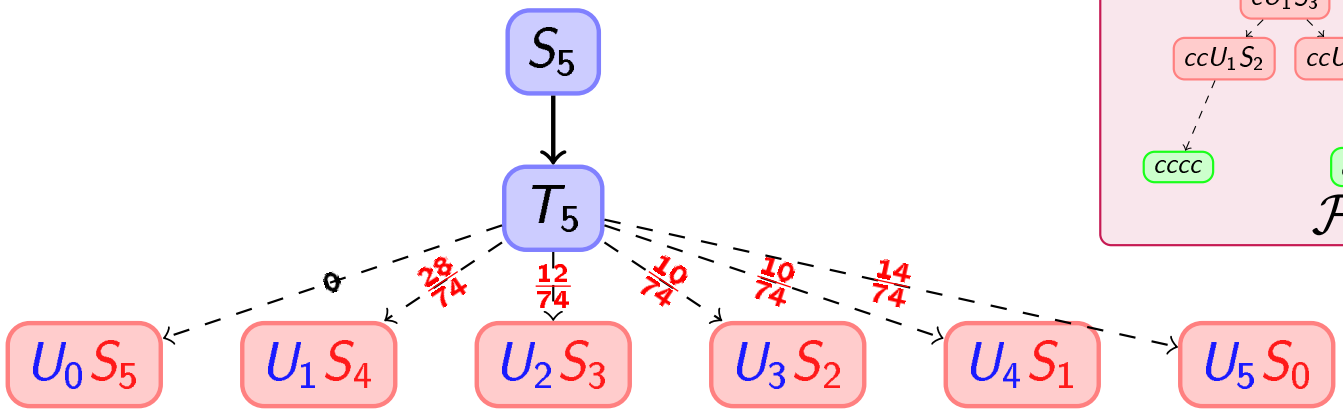


Example (Weighted Motzkin words):

Return

$S \to T \mid \varepsilon \quad T \to U\,S \quad U \to a\,S\,b \mid c \qquad \pi(c) = 2 \quad \pi(a) = \pi(b) = 1$





Algorithm $\mathcal{A}$

- Precompute the weights s_n of words of length n generated from S.
- Turn $\mathcal{F}$ into parse traces, gathered in a weighted prefix tree $\mathcal{T}$.
- During the generation, walk in $\mathcal{T}$ to obtain the contributions $k_\pi(x)$ of words from $\mathcal{L}(x) \cap \mathcal{F}$.
- Iterate atomic derivations $w \longrightarrow w'$ w.p. $\frac{\pi(\mathcal{L}(w')) - k_\pi(w')}{\pi(\mathcal{L}(w)) - k_\pi(w)}$.

Theorem

Algorithm $\mathcal{A}$ draws a word $\omega \in \mathcal{L}(\mathcal{G})_n / \mathcal{F}$ with respect to the weighted (renormalized) distribution.

Proof: Let $w_1, \ldots, w_m$ be the intermediate productions involved in building ω, s.t. $w_1 = \mathcal{S}_{|\omega|}$ and $w_1 \longrightarrow w_2 \longrightarrow \ldots \longrightarrow w_m \longrightarrow \omega$. Then

$$\mathbb{P}(\omega) = \frac{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_1)/\mathcal{F}\right)} \frac{\pi\left(\mathcal{L}(w_3)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)} \frac{\pi\left(\mathcal{L}(w_4)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_3)/\mathcal{F}\right)} \cdots \frac{\pi\left(\omega\right)}{\pi\left(\mathcal{L}(w_m)/\mathcal{F}\right)}.$$

Algorithm $\mathcal{A}$

- Precompute the weights s_n of words of length n generated from S.
- Turn $\mathcal{F}$ into parse traces, gathered in a weighted prefix tree $\mathcal{T}$.
- During the generation, walk in $\mathcal{T}$ to obtain the contributions $k_\pi(x)$ of words from $\mathcal{L}(x) \cap \mathcal{F}$.
- Iterate atomic derivations $w \longrightarrow w'$ w.p. $\frac{\pi(\mathcal{L}(w')) - k_\pi(w')}{\pi(\mathcal{L}(w)) - k_\pi(w)}$.

Theorem

Algorithm $\mathcal{A}$ draws a word $\omega \in \mathcal{L}(\mathcal{G})_n / \mathcal{F}$ with respect to the weighted (renormalized) distribution.

Proof: Let $w_1, \ldots, w_m$ be the intermediate productions involved in building ω, s.t. $w_1 = \mathcal{S}_{|\omega|}$ and $w_1 \longrightarrow w_2 \longrightarrow \ldots \longrightarrow w_m \longrightarrow \omega$. Then

$$\mathbb{P}(\omega) = \frac{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_1)/\mathcal{F}\right)} \frac{\pi\left(\mathcal{L}(w_3)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)} \frac{\pi\left(\mathcal{L}(w_4)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_3)/\mathcal{F}\right)} \cdots \frac{\pi\left(\omega\right)}{\pi\left(\mathcal{L}(w_m)/\mathcal{F}\right)}.$$

Algorithm $\mathcal{A}$

- Precompute the weights s_n of words of length n generated from S.
- Turn $\mathcal{F}$ into parse traces, gathered in a weighted prefix tree $\mathcal{T}$.
- During the generation, walk in $\mathcal{T}$ to obtain the contributions $k_\pi(x)$ of words from $\mathcal{L}(x) \cap \mathcal{F}$.
- Iterate atomic derivations $w \to w'$ w.p. $\frac{\pi(\mathcal{L}(w')) - k_\pi(w')}{\pi(\mathcal{L}(w)) - k_\pi(w)}$.

Theorem

Algorithm $\mathcal{A}$ draws a word $\omega \in \mathcal{L}(\mathcal{G})_n/\mathcal{F}$ with respect to the weighted (renormalized) distribution.

Proof: Let $w_1, \ldots, w_m$ be the intermediate productions involved in building ω, s.t. $w_1 = \mathcal{S}_{|\omega|}$ and $w_1 \to w_2 \to \ldots \to w_m \to \omega$. Then

$$\mathbb{P}(\omega) = \frac{1}{\pi\left(\mathcal{L}(w_1)/\mathcal{F}\right)} \frac{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)}{1} \frac{\pi\left(\mathcal{L}(w_4)/\mathcal{F}\right)}{\pi\left(\mathcal{L}(w_2)/\mathcal{F}\right)} \cdots \frac{\pi\left(\omega\right)}{\pi\left(\mathcal{L}(w_m)/\mathcal{F}\right)}.$$

Algorithm $\mathcal{A}$

- Precompute the weights s_n of words of length n generated from S.
- Turn $\mathcal{F}$ into parse traces, gathered in a weighted prefix tree $\mathcal{T}$.
- During the generation, walk in $\mathcal{T}$ to obtain the contributions $k_\pi(x)$ of words from $\mathcal{L}(x) \cap \mathcal{F}$.
- Iterate atomic derivations $w \to w'$ w.p. $\frac{\pi(\mathcal{L}(w')) - k_\pi(w')}{\pi(\mathcal{L}(w)) - k_\pi(w)}$.

Theorem

Algorithm $\mathcal{A}$ draws a word $\omega \in \mathcal{L}(\mathcal{G})_n/\mathcal{F}$ with respect to the weighted (renormalized) distribution.

Proof: Let $w_1, \ldots, w_m$ be the intermediate productions involved in building ω, s.t. $w_1 = \mathcal{S}_{|\omega|}$ and $w_1 \to w_2 \to \ldots \to w_m \to \omega$. Then

$$\mathbb{P}(\omega) = \frac{1}{\pi\left(\mathcal{L}(w_1)/\mathcal{F}\right)} \frac{1}{1} \frac{\pi\left(\mathcal{L}(w_4)/\mathcal{F}\right)}{1} \cdots \frac{\pi\left(\omega\right)}{\pi\left(\mathcal{L}(w_m)/\mathcal{F}\right)}.$$

Algorithm $\mathcal{A}$

- Precompute the weights s_n of words of length n generated from S.
- Turn $\mathcal{F}$ into parse traces, gathered in a weighted prefix tree $\mathcal{T}$.
- During the generation, walk in $\mathcal{T}$ to obtain the contributions $k_\pi(x)$ of words from $\mathcal{L}(x) \cap \mathcal{F}$.
- Iterate atomic derivations $w \to w'$ w.p. $\frac{\pi(\mathcal{L}(w')) - k_\pi(w')}{\pi(\mathcal{L}(w)) - k_\pi(w)}$.

Theorem

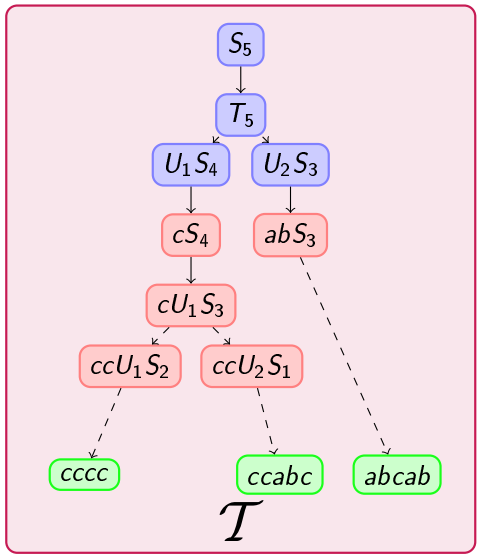
Algorithm $\mathcal{A}$ draws a word $\omega \in \mathcal{L}(\mathcal{G})_n/\mathcal{F}$ with respect to the weighted (renormalized) distribution.

Proof: Let $w_1, \ldots, w_m$ be the intermediate productions involved in building ω, s.t. $w_1 = \mathcal{S}_{|\omega|}$ and $w_1 \to w_2 \to \ldots \to w_m \to \omega$. Then

$$\mathbb{P}(\omega) = \frac{\pi(\omega)}{\pi(\mathcal{L}(w_1)/\mathcal{F})} = \frac{\pi(\omega)}{\pi(\mathcal{L}(\mathcal{S}_{|\omega|})/\mathcal{F})} = \frac{\pi(\omega)}{\pi(\mathcal{L}(\mathcal{G})_n/\mathcal{F})}.$$

Complexity considerations

- Don't store the partial words in internal nodes!
  Only **diff** with parent node will suffice.
  $\Rightarrow \mathcal{O}(1)$ memory for each node.
- Numbers stored in the internal nodes are encoded on $\mathcal{O}(n)$ bits.
  $\Rightarrow$ Keeping them could take up to $\mathcal{O}(n^2.k)$ bits?
  No, there is $\mathcal{O}(k)$ different numbers on the tree.
  (In any tree, the number of nodes of degree>1 is smaller than the number of leaves.)



Starting from $\mathcal{F} := \emptyset$, the overhead is negligible:
$\Rightarrow$ Generating k distinct samples of size n takes $\mathcal{O}(kn \log n)$ arithmetic operations and requires the storage of $\mathcal{O}(kn)$ numbers.

Perspectives

- **Speeding up generation:** Transposition of classic optimizations for the recursive generation (Boustrophedon search, linear recurrences for coeffs, ...) and beyond (Boltzmann sampling, unranking ...)
- **Implementation:** Watching the computer explode ???
  Potential numerical stability issues ...
- **Applications:** Adapt techniques to RNA structure (Folding) and sequence (Design) sampling. Alternative to local search for hard optimization problems ?
- **Open problem:** Precise complexity study of the rejection approach under a weighted model. Upper bounds (Coupon collector) could be obtained in the spirit of [Flajolet *et al.*, 1992].

Thanks for listening !!!

And to Alain Denise for advise and support over the years...


Denise, A., Roques, O., & Termier, M. 2000.
Random generation of words of context-free languages according to the frequencies of letters.
Pages 113–125 of: Mathematics and computer science: Algorithms, trees, combinatorics and probabilities.


Flajolet, Philippe, Gardy, Danièle, & Thimonier, Loßs. 1992.
Birthday paradox, coupon collectors, caching algorithms and self-organizing search.
Discrete appl. math., **39**(3), 207–229.


Wilf, H. S. 1977.
A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects.
Advances in mathematics, **24**, 281–291.


Zimmermann, P. 1995.
Uniform random generation for the powerset construction.
Pages 589–600 of: Proceedings of the 7th conference on formal power series and algebraic combinatorics.