Random Sampling of Ordered Trees according to the Number of Occurrences of a Pattern

Gwendal Collet, Julien David, Alice Jacquot

GASCom, June 2nd 2016
Definitions

\( S \)-Trees: \( T = (\text{root}, (T_1, \ldots, T_k)) \) where \( k \in S \supseteq 0, S \) finite

ex: Binary trees \( (S = 0, 2) \), Motzkin trees \( (S = \{0, 1, 2\}) \), Plane trees

Prefix

[Diagram of S-Trees]
Definitions

**S-Trees:** $T = (\text{root}, (T_1, \ldots, T_k))$ where $k \in S \supseteq 0$, $S$ finite

ex: Binary trees ($S = 0, 2$), Motzkin trees ($S = \{0, 1, 2\}$), Plane trees

![Diagram of S-Trees](image-url)
Definitions

$S$-Trees: $T = (\text{root}, (T_1, \ldots, T_k))$ where $k \in S \supseteq 0$, $S$ finite

ex: Binary trees ($S = 0, 2$), Motzkin trees ($S = \{0, 1, 2\}$), Plane trees

Pattern = Prefix of suffix
Definitions

$S$-Trees: $T = (\text{root}, (T_1, \ldots, T_k))$ where $k \in S \supseteq 0$, $S$ finite
ex: Binary trees ($S = 0, 2$), Motzkin trees ($S = \{0, 1, 2\}$), Plane trees

Not a pattern!
Definitions

$S$-Trees: $T = (\text{root}, (T_1, \ldots, T_k))$ where $k \in S \supseteq 0$, $S$ finite

ex: Binary trees ($S = 0, 2$), Motzkin trees ($S = \{0, 1, 2\}$), Plane trees

2 occurrences
Definitions

\( S\)-Trees: \( T = (\text{root}, (T_1, \ldots, T_k)) \) where \( k \in S \supseteq 0, S \text{ finite} \)

ex: Binary trees \((S = 0, 2)\), Motzkin trees \((S = \{0, 1, 2\})\), Plane trees

2 overlapping occurrences
+ 1 occurrence
Definitions

$S$-Trees: $T = (\text{root}, (T_1, \ldots, T_k))$ where $k \in S \supseteq 0$, $S$ finite

ex: Binary trees ($S = 0, 2$), Motzkin trees ($S = \{0, 1, 2\}$), Plane trees

Problem: Given $S$ and a $S$-tree $P$, how to sample randomly a $S$-tree with $n$ node and exactly $k$ occurrences of $P$?

[Chyzak, Drmota, Klausner, Kok’2008] Expected number of occurrences is Gaussian in unordered trees

[Flouri, Melichar, Janousek’2009] Linear algorithm to count the number of occurrences
Idea of the algorithm

Given a $S$-tree $P$:

Precalculus: algorithm to generate a tree language specification
- recognizing any $S$-tree
- marking each occurrence of $P$
⇒ adapt Aho-Corasic algorithm on words to tree structures

Random sampler:
- translate the specification into a system of algebraic equations on generating series
- build a bivariate Boltzmann sampler based on these equations
Idea of the algorithm

Given a $S$-tree $P$:

Precalculus: algorithm to generate a tree language specification
  → recognizing any $S$-tree
  → marking each occurrence of $P$
  ⇒ adapt Aho-Corasic algorithm on words to tree structures

Random sampler:
  → translate the specification into a system of algebraic equations
     on generating series
  → build a bivariate Boltzmann sampler based on these equations

Remark on Boltzmann samplers:
  quasi-automatically built on generating series (+ singularity extraction)
  uniform among elements of same size
  linear in approximated size
  quadratic in exact size by reject
Idea of the algorithm

Read the tree from top to bottom

At a given height: does a node belong to an occurrence of $P$?
  → depends on nodes above, at a bounded distance ($h(P)$)
  → depends on neighbors, at a bounded distance ($\max(\text{arity})^h(P)$)
  → depends on nodes below (to check later)
⇒ Only need to check a subtree of bounded size

Strong dependencies between nodes at same height
⇒ Need to consider simultaneously tuples of nodes
Idea of the algorithm

Read the tree from top to bottom

At a given height: does a node belong to an occurrence of $P$?
→ depends on nodes above, at a bounded distance ($h(P)$)
→ depends on neighbors, at a bounded distance ($\max(\text{arity})^h(P)$)
→ depends on nodes below (to check later)
⇒ Only need to check a subtree of bounded size

Strong dependencies between nodes at same height
⇒ Need to consider simultaneously tuples of nodes

Build a grammar where:
→ Non.terminals correspond to tuples of nodes associated to a subtree which is candidate to contain an occurrence
→ Rules describe what happens when this subtree grows
Generalized tree grammar

Let $G = (N, A, S, R)$ be a grammar if

- $N$ = set of non-terminals
- $A$ = axiom (starting non-terminal)
- $S$ = terminals (here arities)
- $R$ = set of rules $r$ such that:

$$r = (n, (s_1, \ldots, s_{|n|}), \lambda, (n_1, \ldots, n_{|\lambda|}))$$

$n \in N, n_j \in N, s_i \in S$

$|n|$ number of nodes in $n$

$\lambda$ partition of $\{1, 2, \ldots, \sum_{i=1}^{k} s_i\}$

$|\lambda|$ number of parts in $\lambda, |\lambda_j| = |n_j|$

ex:

![Diagram of a tree structure](image)

$r = (n, (2, 1, 0, 3), \{13|245|6\}, (n_1, n_2, n_3))$
Generalized tree grammar

Let $G = (N, A, S, R)$ be a grammar if:
- $N$ = set of non-terminals
- $A$ = axiom (starting non-terminal)
- $S$ = terminals (here arities)
- $R$ = set of rules $r$ such that:

Let $G = (N, A, S, R)$ be a grammar if

\[ N = \text{set of non-terminals} \]
\[ A = \text{axiom (starting non-terminal)} \]
\[ S = \text{terminals (here arities)} \]
\[ R = \text{set of rules } r \text{ such that:} \]

\[ r = (n, (s_1, \ldots, s_{|n|}), \lambda, (n_1, \ldots, n_{|\lambda|})) \]

$n \in N, n_j \in N, s_i \in S$

$|n|$ number of nodes in $n$

$\lambda$ partition of $\{1, 2, \ldots, \sum_{i=1}^{k} s_i\}$

$|\lambda|$ number of parts in $\lambda, |\lambda_j| = |n_j|$

ex: $n$

\[
\begin{align*}
1 & \square \\
2 & \square \\
3 & \square \\
4 & \square \\
5 & \square \\
6 & \bullet
\end{align*}
\]

$n_1$, $n_2$, $n_3$

\[
\begin{align*}
r = (n, (2, 1, 0, 3), \{13|245|6\}, (n_1, n_2, n_3))
\end{align*}
\]

$S = \{0, 1, 2, 3\}$

A given pattern:

The grammar we obtain: * marks a rule that produces an occurrence of the pattern.
Dealing with overlappings

\[ P \]

Double comb
Dealing with overlappings

Double comb
Dealing with overlappings

$P$

Double comb
Dealing with overlappings

Double comb

Node belonging to two different prefixes of $P$
Dealing with overlappings

Double comb

Disjoint nodes belonging to two overlapping prefixes
Dealing with overlappings

Double comb

New non-terminal!
Dealing with overlappings

⇒ If two prefixes share at least one leaf, all their leaves must be taken in the same part of $\lambda$

→ Might create new non-terminals by superposing prefixes of $P$

→ Possible exponential explosion of the number of non-terminals in pathological cases (like double comb)
Dealing with overlappings

$P$

Double comb

⇒ If two prefixes share at least one leaf, all their leaves must be taken in the same part of $\lambda$

→ Might create new non-terminals by superposing prefixes of $P$

→ Possible exponential explosion of the number of non-terminals in pathological cases (like double comb)

Remark: Costly precalculus in some cases (in the size of $P$)

but in practice, the pattern is small compared to the generated trees

Boltzmann sampler still linear, but at a cost in memory space, due to the size of the generated grammar
Backbone of the algorithm

Input: a $S$-tree $P$
Output: a grammar $G = (N, U, A, S, R)$

\[
N \leftarrow \{A\}, U \leftarrow \emptyset, R \leftarrow \emptyset
\]

For each non terminal $n \in N$ do
  For each $(s_1, \ldots, s_{|n|}) \in S^{n}$ do
    Compute new tree $T$
    Compute new prefixes of $P$ in $T$
    Compute partition $\lambda$ of independant nodes
    Compute subtree associated to each part
    If new subtree $T' \notin N$
      Then add $T'$ to $N$
    If $\text{height}(\text{new subtree } T') = \text{height}(P)$
      Then add $T'$ to $U$
    Add rule $(n, (s_1, \ldots, s_{|n|}), \lambda, (n_1, \ldots, n_{|\lambda|}))$ to $R$

Return $(N, U, A, S, R)$
Experimental results

Size of the generated grammar for 100 random patterns

- **Binary tree with 20 nodes:**
  - Number of rules: \( \leq 50 \), \( \leq 500 \), \( \leq 5000 \), \( \leq 50000 \), \( \leq 400000 \)
  - Proportion:
    - \( \leq 50 \): 24
    - \( \leq 500 \): 49
    - \( \leq 5000 \): 16
    - \( \leq 50000 \): 2
    - \( \leq 400000 \): 9

- **Motzkin tree with 20 nodes (30% unary):**
  - Number of rules: \( \leq 200 \), \( \leq 500 \), \( \leq 3000 \)
  - Proportion:
    - \( \leq 200 \): 74
    - \( \leq 500 \): 13
    - \( \leq 3000 \): 13

- **Motzkin tree with 50 nodes (40% unary):**
  - Number of rules: \( \leq 500 \), \( \leq 1000 \), \( \leq 2000 \), \( \leq 5000 \), \( \leq 50000 \), \( \leq 300000 \)
  - Proportion:
    - \( \leq 500 \): 25
    - \( \leq 1000 \): 21
    - \( \leq 2000 \): 19
    - \( \leq 5000 \): 20
    - \( \leq 50000 \): 12
    - \( \leq 300000 \): 3

- **Motzkin tree with 50 nodes (50% unary):**
  - Number of rules: \( \leq 500 \), \( \leq 1000 \), \( \leq 2000 \), \( \leq 5000 \), \( \leq 50000 \), \( \leq 300000 \)
  - Proportion:
    - \( \leq 500 \): 35
    - \( \leq 1000 \): 28
    - \( \leq 2000 \): 15
    - \( \leq 5000 \): 12
    - \( \leq 50000 \): 9
    - \( \leq 300000 \): 1
Thank you!