A general and algorithmic method for computing the generating function of permutation classes and for their random generation

Mathilde Bouvel (LaBRI)
avec Frédérique Bassino (LIPN), Adeline Pierrot (LIAFA), Carine Pivoteau (LIGM), Dominique Rossin (LIX)

GT Combi - 9 décembre 2011
Guideline for the talk

Data:
- $B$ a finite set of permutations (the excluded patterns),
- $\mathcal{C} = Av(B)$ the class of permutations that avoid every pattern of $B$.

Problem:
Describe an algorithm to obtain automatically from $B$ a combinatorial specification for $\mathcal{C}$, and hence:
- some enumerative results on $\mathcal{C}$, in terms of generating function $C(z) = \sum |Av_n(B)|z^n$,
- a random sampler of permutations in $\mathcal{C}$, that is uniform on $Av_n(B)$ for each $n$.

Result:
Such an algorithm ... that works under some hypothesis on $\mathcal{C}$, also tested algorithmically.
Outline

1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes
Outline

1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes
**Permutation**: Bijection from \([1..n]\) to itself. Set \(S_n\).

- **Linear representation**: 
  \[ \sigma = 1 \ 8 \ 3 \ 6 \ 4 \ 2 \ 5 \ 7 \]

- **Two lines representation**: 
  \[ \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 3 & 6 & 4 & 2 & 5 & 7 \end{pmatrix} \]

- **Representation as a product of cycles**: 
  \[ \sigma = (1) \ (2 \ 8 \ 7 \ 5 \ 4 \ 6) \ (3) \]

- **Graphical representation**: 
  ![Graphical representation of a permutation](image-url)
Patterns in permutations

**Pattern (order) relation** $\preceq$:

$\pi \in \mathfrak{S}_k$ is a pattern of $\sigma \in \mathfrak{S}_n$ if $\exists 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is order isomorphic ($\equiv$) to $\pi$.

Notation: $\pi \preceq \sigma$.

*Equivalently:*

The normalization of $\sigma_{i_1} \ldots \sigma_{i_k}$ on $[1..k]$ yields $\pi$.

**Example:** $2134 \preceq 312854796$ since $3157 \equiv 2134$. 

Mathilde Bouvel

Algorithmic methodology for the enumeration and random generation of permutation classes
**Patterns in permutations**

**Pattern (order) relation** \( \preceq \):
\[
\pi \in \mathcal{S}_k \text{ is a pattern of } \sigma \in \mathcal{S}_n \text{ if } \exists 1 \leq i_1 < \ldots < i_k \leq n \text{ such that } 
\sigma_{i_1} \ldots \sigma_{i_k} \text{ is order isomorphic (} \equiv \text{) to } \pi.
\]

Notation: \( \pi \preceq \sigma \).

*Equivalently:*

The normalization of \( \sigma_{i_1} \ldots \sigma_{i_k} \) on \([1..k]\) yields \( \pi \).

**Example:** \( 2134 \preceq 312854796 \)
since \( 3157 \equiv 2134 \).
Patterns in permutations

**Pattern (order) relation** ≼:

π ∈ S_k is a pattern of σ ∈ S_n if ∃ 1 ≤ i_1 < ... < i_k ≤ n such that σ_{i_1}...σ_{i_k} is order isomorphic (≡) to π.

Notation: π ≼ σ.

**Equivalently:**
The normalization of σ_{i_1}...σ_{i_k} on [1..k] yields π.

Example: 2 1 3 4 ≼ 3 1 2 8 5 4 7 9 6 since 3 1 5 7 ≡ 2 1 3 4.
## Permutation classes

**Permutation class**: set of permutations downward-closed for \(<\).

\(\text{Av}(B)\): the class of permutations that avoid every pattern of \(B\).
If \(B\) is an antichain then \(B\) is the basis of \(\text{Av}(B)\).

**Conversely**: Every class \(C\) can be characterized by its basis:

\[C = \text{Av}(B)\text{ for } B = \{\sigma \notin C : \forall \pi < \sigma \text{ such that } \pi \neq \sigma, \pi \in C\}\]

A class has a **unique** basis.
A basis can be either finite or infinite.

**Origin**: [Knuth 73] with stack-sortable permutations = \(\text{Av}(231)\)

**Enumeration**: [Stanley & Wilf 92][Marcus & Tardos 04]: \(|C \cap \mathcal{S}_n| \leq c^n\)
Problematics

- **Combinatorics**: study of classes defined by their basis.
  - Enumeration.
  - Exhaustive generation.

- **Algorithmics**: problematics from text algorithmics.
  - Pattern matching, longest common pattern.
  - Linked with testing the membership of $\sigma$ to a class.

- **Combinatorics (and algorithms)**: study families of classes.
  - The basis of the class is not always given.
  - Obtain general results on permutation classes…
  - …and do it automatically (with algorithms).
1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes
Substitution decomposition: main ideas

Analogous to the decomposition of integers as products of primes.

- [Möhring & Radermacher 84]: general framework.
- Specialization: Modular decomposition of graphs.

Relies on:
- a principle for building objects (permutations, graphs) from smaller objects: the substitution.
- some “basic objects” for this construction: simple permutations, prime graphs.

Required properties:
- every object can be decomposed using only “basic objects”.
- this decomposition is unique.
**Substitution** or inflation: \( \sigma = \pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}] \).

**Example**: Here, \( \pi = 1 \ 3 \ 2 \), and

\[
\begin{align*}
\alpha^{(1)} &= 2 \ 1 = \\
\alpha^{(2)} &= 1 \ 3 \ 2 = \\
\alpha^{(3)} &= 1 =
\end{align*}
\]

Hence \( \sigma = 1 \ 3 \ 2[2 \ 1, 1 \ 3 \ 2, 1] = 2 \ 1 \ 4 \ 6 \ 5 \ 3 \).
Simple permutations

**Interval (or block)** = set of elements of \( \sigma \) whose positions and values form intervals of integers

*Example*: 5 7 4 6 is an interval of 2 5 7 4 6 1 3

**Simple permutation** = permutation that has no interval, except the trivial intervals: 1, 2, \ldots, n and \( \sigma \)

*Example*: 3 1 7 4 6 2 5 is simple.

*The smallest simple*: 1 2, 2 1, 2 4 1 3, 3 1 4 2

---

Mathilde Bouvel

Algorithmic methodology for the enumeration and random generation of permutation classes
Substitution decomposition and decomposition trees

Substitution decomposition of permutations

**Theorem:** Every $\sigma \neq 1$ is **uniquely** decomposed as

- $12[\alpha^{(1)}, \alpha^{(2)}]$, where $\alpha^{(1)}$ is $\oplus$-indecomposable
- $21[\alpha^{(1)}, \alpha^{(2)}]$, where $\alpha^{(1)}$ is $\ominus$-indecomposable
- $\pi[\alpha^{(1)}, \ldots, \alpha^{(k)}]$, where $\pi$ is simple of size $k \geq 4$

**Remarks:**

- $\oplus$-indecomposable: that cannot be written as $12[\alpha^{(1)}, \alpha^{(2)}]$
- Result stated as in [Albert & Atkinson 05]
- Can be rephrased changing the first two items into:
  - $12 \ldots k[\alpha^{(1)}, \ldots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\oplus$-indecomposable
  - $k \ldots 21[\alpha^{(1)}, \ldots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\ominus$-indecomposable

Decomposing recursively inside the $\alpha^{(i)} \Rightarrow$ **decomposition tree**
Decomposition tree: witness of this decomposition

**Example**: Decomposition tree of $\sigma =$

10 13 12 11 14 1 18 19 20 21 17 16 15 4 8 3 2 9 5 6 7

3 1 4 2

3 1 4 2

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

$\oplus$ $\oplus$ $\oplus$

Notations and properties:

- $\oplus = 12 \ldots k$ and $\ominus = k \ldots 21 = \text{linear nodes}.$
- $\pi$ simple of size $\geq 4 = \text{prime node}.$
- No edge $\oplus - \oplus$ nor $\ominus - \ominus.$
- *Ordered* trees.

Expansion of $T_1 T_2 T_3 \ldots$ into and recursively, for the version of the trees of [AA05]

$\sigma = 3 1 4 2[\oplus[1, \ominus[1, 1, 1], 1], 1, \ominus[\oplus[1, 1, 1], 1, 1, 1], 2 4 1 5 3[1, 1, \ominus[1, 1], 1, \oplus[1, 1, 1]]]$

Bijection between permutations and their decomposition trees.

Mathilde Bouvel

Algorithmic methodology for the enumeration and random generation of permutation classes
Computation and examples of application

**Computation:** in **linear** time. [Uno & Yagiura 00] [Bui Xuan, Habib & Paul 05] [Bergeron, Chauve, Montgolfier & Raffinot 08]

**In algorithms:**
- Pattern matching [Bose, Buss & Lubiw 98] [Ibarra 97]
- Algorithms for bio-informatics [Bérard, Bergeron, Chauve & Paul 07] [Bérard, Chateau, Chauve, Paul & Tannier 08]

**In combinatorics:**
- Simple permutations [Albert, Atkinson & Klazar 03]
- Classes closed by substitution product [Atkinson & Stitt 02] [Brignall 07] [Atkinson, Ruškuc & Smith 09]
- Exhibit the structure of classes [Albert & Atkinson 05] [Brignall, Huczynska & Vatter 08a,08b] [Brignall, Ruškuc & Vatter 08]
Outline

1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes
Combinatorial classes and generating functions

Notations:
- \( \mathcal{C} = \bigcup_{n \geq 0} \mathcal{C}_n \) with finite number \( c_n = |\mathcal{C}_n| \) of objects of size \( n \)
- Generating function \( C(z) = \sum c_n z^n \)

Recursive description with constructors \( \Rightarrow \) Equation on the g.f.:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Notation</th>
<th>( C(z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atom</td>
<td>( \mathcal{Z} )</td>
<td>( z )</td>
</tr>
<tr>
<td>Disjoint Union</td>
<td>( \mathcal{A} + \mathcal{B} )</td>
<td>( A(z) + B(z) )</td>
</tr>
<tr>
<td>Cartesian Product</td>
<td>( \mathcal{A} \times \mathcal{B} )</td>
<td>( A(z)B(z) )</td>
</tr>
<tr>
<td>Sequence</td>
<td>( \text{SEQ}(\mathcal{A}) )</td>
<td>( \frac{1}{1 - A(z)} )</td>
</tr>
<tr>
<td>Restricted Seq.</td>
<td>( \text{SEQ}_{\leq k}(\mathcal{A}) )</td>
<td>( A(z)^k )</td>
</tr>
</tbody>
</table>

[Flajolet & Sedgewick 09]
Combinatorial classes and random samplers

**Uniform sampling**: objects of size $n$ have the same probability

Two methods based on the recursive description of objects:

- **Recursive method** [Flajolet, Zimmerman & Van Cutsem 94]:
  
  size $n$ chosen in advance. Requires to know the $c_k$ for $k \leq n$.

- **Boltzmann method** [Duchon, Flajolet, Louchard & Schaeffer 04]:
  
  size $n$ not fixed. Needs the evaluation of $C(z)$ at one point $x$.

<table>
<thead>
<tr>
<th>$Z$</th>
<th>return an atom</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{A} + \mathcal{B}$</td>
<td>call $\Gamma A(x)$ with proba. $\frac{A(x)}{A(x)+B(x)}$, else $\Gamma B(x)$</td>
</tr>
<tr>
<td>$\mathcal{A} \times \mathcal{B}$</td>
<td>call $\Gamma A(x)$ and $\Gamma B(x)$</td>
</tr>
<tr>
<td>$\text{SEQ}(\mathcal{A})$</td>
<td>choose $k$ according to a geometric law of parameter $A(x)$ and call $\Gamma A(x)$ $k$ times</td>
</tr>
<tr>
<td>$\text{SEQ}_{\leq k}(\mathcal{A})$</td>
<td>call the sampler $\Gamma A(x)$ $k$ times</td>
</tr>
</tbody>
</table>
Example: binary trees

\[ \mathcal{B} = \bigcup_{n \geq 1} \mathcal{B}_n \]
where \( \mathcal{B}_n \) denotes the set of binary trees with \( n \) leaves.

Recursive description (also called specification): \( \mathcal{B} = \bullet + \mathcal{B} \mathcal{B} \)

Equation for the g.f.: \( B(z) = z + B(z)^2 \), hence \( B(z) = \frac{1 - \sqrt{1 - 4z}}{2} \).

Boltzmann random sampler \( \Gamma \mathcal{B}(x) \) for \( \mathcal{B} \):

- **Data:** \( x, B(x) \)
- **Result:** a random binary tree
- **Procedure:**
  - Choose \( r \) uniformly at random on \([0, 1]\)
  - If \( \frac{x}{B(x)} < r \) then return \( \bullet \)
  - Else return \( \Gamma \mathcal{B}(x) \mathcal{B}(x) \)
Specifications for permutation classes

For all permutations, with $S$ the set of all simple permutations:

$$
\begin{align*}
S &= \bullet + S^+ + S^- + \sum_{\pi \in S} S \cdot \cdots \cdot S \\
S^+ &= \bullet + S^- + \sum_{\pi \in S} S \cdot \cdots \cdot S \\
S^- &= \bullet + S^+ + \sum_{\pi \in S} S \cdot \cdots \cdot S
\end{align*}
$$

$\Rightarrow$ The generating functions of $S$ and $S$ are related [Albert, Atkinson & Klazar 03].

This can be adapted to (substitution-closed and arbitrary) permutation classes [Albert & Atkinson 05].
The simpler case of substitution-closed classes

A permutation class $C$ is **substitution-closed** when $\pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}] \in C$ for all $\pi, \alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)} \in C$.

Hence, with $S_C = C \cap S$ the set of simple permutations in $C$:

\[
C = \bullet + C^+ \circ C + C^- \circ C + \sum_{\pi \in S_C} C \circ C \cdots C
\]

When $S_C$ is finite, this is a **simple family of trees** in the sense of [Flajolet & Sedgewick 09].

⇒ Enumerative results and random samplers can be obtained by efficient algorithms.
For general permutation classes

For non substitution-closed classes, we have only a strict inclusion:

\[ \mathcal{C} \subsetneq \bullet + \mathcal{C}^+ \mathcal{C} + \mathcal{C}^- \mathcal{C} + \sum_{\pi \in \mathcal{S}_C} \mathcal{C} \pi \mathcal{C} \cdots \mathcal{C} \]

Example: \( 231 = 21[12, 1] \notin \text{Av}(231) \) whereas \( 21, 12, 1 \in \text{Av}(231) \).

The system describing \( \mathcal{C} \) has to be refined with new equations for these constraints. The system can be computed by an algorithm.

⇒ Enumerative results and random samplers can be obtained algorithmically, but this is less efficient.
Outline

1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives
Summary of results

**B: finite basis of excluded patterns**

- **B contains only simple permutations**
  - Av(B) is substitution-closed

- **B contains permutations that are not simple**
  - Av(B) is not substitution-closed

**Is there a finite number of simple permutations in the class C=Av(B)?**

- **NO**
  - Stop

- **YES**

**Computation of the subset Sc of simple permutations in C**

- **O(N, \ell^4)**
  - Direct

**Specification for C**

- Generating function and Boltzmann sampler

**Constraints propagation**

- Exponential (?)
First semi-decision procedure

**Theorem** [Albert & Atkinson 05]: If \( C \) contains a **finite** number of simple permutations, then \( C \) has a **finite basis** and an **algebraic g.f.**.

**Constructive proof:** compute, for each given class,
- the specification for decomposition trees of \( C \)
- a system of equations satisfied by the g.f.
from the **finite** set of simple permutations in \( C \)

**Testing the precondition:**
- Semi-decision procedure
  - Find simples of size 4, 5, 6, ... until \( k \) and \( k + 1 \) for which there are 0 simples [Schmerl & Trotter 93]
- “Very exponential” \( \sim n! \) computation of the simples in \( C \)
Step 1: Is there a finite number of simple permutations in \( C \)? A first decision result

**Theorem** [Brignall, Ruškuc & Vatter 08]: It is **decidable** whether \( C \) given by its **finite basis** contains a finite number of simples.

**Prop:** \( C = \text{Av}(B) \) contains infinitely many simples iff \( C \) contains:

1. either infinitely many parallel permutations
2. or infinitely many simple wedge permutations
3. or infinitely many proper pin-permutations

<table>
<thead>
<tr>
<th>Decision procedure</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. and 2. : pattern matching of patterns of size 3 or 4 in the ( \beta \in B ).</td>
<td>( \mathcal{O}(n \log n) )</td>
</tr>
<tr>
<td>3. : Decidability with automata techniques</td>
<td>2ExpTime</td>
</tr>
</tbody>
</table>

Mathilde Bouvel

**Algorithmic methodology for the enumeration and random generation of permutation classes**
Polynomial algorithms for the finite number of simples

Points similar to [BRV 08]:
- Encoding by pin words on \( \{1, 2, 3, 4, L, R, U, D\} \)
- Construction of automata accepting words of pin-permutations \( \pi \) such that \( \beta \preceq \pi \) for some \( \beta \in B \)

Study of pin-permutations [BBR 09] ⇒ better understanding of the relationship between pin words and patterns in permutations

Points specific to [BBPR 10 & 11]:
- Polynomial construction of a (deterministic, complete) automaton for the language \( \mathcal{L} = \) pin words of proper pin-permutations containing some \( \beta \in B \)
- Is this language co-finite? Polynomial.
  - Yes iff the class contains finitely many simples.

Polynomial w.r.t. \( n = \sum_{\beta \in B} |\beta| \), but \( k = |B| \) is an exponent.
An algorithm from the finite basis to the specification

**Step 2: Finding the set of simple permutations in** \( C \)

**Starting point:** Find simple permutations in \( C \) of size 4, 5, 6, \ldots until \( k \) and \( k + 1 \) for which there are 0 simples

**Problem:** There are \( \sim \frac{n!}{e^2} \) simple permutations of size \( n \)

Reduce the number of simples \( \sigma \) of size \( n \) that are candidate to the membership to \( C \) [Pierrot & Rossin, 11].

**Prop:** The simples of \( C_{n+1} \) can be described as one-point (or special two-points) extensions of the simples of \( C_n \)

\( \Rightarrow \) There are at most \( O(n^2 \cdot |S \cap C_n|) \) candidates of size \( n + 1 \).

Test whether \( \sigma \) contains an occurrence of \( \beta \in B \): in \( O(n^{|eta|}) \).

**Theorem:** Computing the finite set of simple permutations in \( C \) is done in \( O(N \cdot \ell^{p+2} \cdot |B|) \) with \( N = |S \cap C|, \ p = \max\{|\beta| : \beta \in B\} \) and \( \ell = \max\{|\pi| : \pi \in S \cap C\} \).
Refinement for substitution-closed classes

Prop: \( C = Av(B) \) is substitution-closed iff \( B \) contains only simples.

Prop [Pierrot & Rossin, 11]: If \( \beta \preceq \sigma \) for \( \beta \) and \( \sigma \) simples, then there are simples \( \beta = \sigma_1 \preceq \sigma_2 \ldots \preceq \sigma_k = \sigma \) s.t. for all \( i \), \( |\sigma_i| - |\sigma_{i-1}| = 1 \) (or 2 in special cases).

Improvement of the complexity:

- Avoid testing occurrences of \( \beta \in B \) in \( \sigma \) candidate simple of \( C \).
- Instead, test whether for every one point (or special two points) deletion in \( \sigma \) resulting in \( \sigma' \) simple, then \( \sigma' \in C \).

\( \Rightarrow \) It is more efficient for computing \( S \cap C_{n+1} \) from \( S \cap C_n \).

Theorem: Computing the finite set of simple permutations in \( C \) is done in \( O(N \cdot \ell^4) \) for substitution-closed classes.
Step 3: Compute the specification for $\mathcal{C}$

From the set $S_C$ of simple permutations in $\mathcal{C}$, the specification for the substitution closure $\hat{\mathcal{C}}$ of $\mathcal{C}$ is obtained immediately:

$$
\hat{\mathcal{C}} = \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}} + \hat{\mathcal{C}}^- \hat{\mathcal{C}} + \sum_{\pi \in S_C} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}
$$

$$
\hat{\mathcal{C}}^+ = \bullet + \hat{\mathcal{C}}^- \hat{\mathcal{C}} + \sum_{\pi \in S_C} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}
$$

$$
\hat{\mathcal{C}}^- = \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}} + \sum_{\pi \in S_C} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}
$$

- If $\mathcal{C}$ is substitution-closed, $\mathcal{C} = \hat{\mathcal{C}}$ and we are done.
- Otherwise, $\mathcal{C} = \hat{\mathcal{C}} \langle B^* \rangle$ and propagate the constraints from $B^* = \{ \beta \in B : \beta \text{ is not simple} \}$ into the subtrees.
Step 3: Compute the specification for $\mathcal{C}$

From the set $\mathcal{S}_\mathcal{C}$ of simple permutations in $\mathcal{C}$, the specification for the substitution closure $\hat{\mathcal{C}}$ of $\mathcal{C}$ is obtained immediately:

\[
\begin{align*}
\hat{\mathcal{C}}\langle B^* \rangle &= \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}}\langle B^* \rangle + \hat{\mathcal{C}}^- \hat{\mathcal{C}}\langle B^* \rangle + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}} \langle B^* \rangle \\
\hat{\mathcal{C}}^+ &= \bullet + \hat{\mathcal{C}}^- \hat{\mathcal{C}} + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}} \\
\hat{\mathcal{C}}^- &= \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}} + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}
\end{align*}
\]

- If $\mathcal{C}$ is substitution-closed, $\mathcal{C} = \hat{\mathcal{C}}$ and we are done.
- Otherwise, $\mathcal{C} = \hat{\mathcal{C}}\langle B^* \rangle$ and propagate the constraints from $B^* = \{ \beta \in B : \beta \text{ is not simple} \}$ into the subtrees.
Step 3: Compute the specification for $\mathcal{C}$

From the set $\mathcal{S}_\mathcal{C}$ of simple permutations in $\mathcal{C}$, the specification for the substitution closure $\hat{\mathcal{C}}$ of $\mathcal{C}$ is obtained immediately:

$$
\begin{align*}
\hat{\mathcal{C}}\langle B^? \rangle &= \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}}\langle B^? \rangle + \hat{\mathcal{C}}^- \hat{\mathcal{C}}\langle B^? \rangle + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}\langle B^? \rangle \\
\hat{\mathcal{C}}^+\langle B^? \rangle &= \bullet + \hat{\mathcal{C}}^- \hat{\mathcal{C}}\langle B^? \rangle + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}\langle B^? \rangle \\
\hat{\mathcal{C}}^-\langle B^? \rangle &= \bullet + \hat{\mathcal{C}}^+ \hat{\mathcal{C}}\langle B^? \rangle + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \hat{\mathcal{C}} \hat{\mathcal{C}} \cdots \hat{\mathcal{C}}\langle B^? \rangle 
\end{align*}
$$

- If $\mathcal{C}$ is substitution-closed, $\mathcal{C} = \hat{\mathcal{C}}$ and we are done.
- Otherwise, $\mathcal{C} = \hat{\mathcal{C}}\langle B^* \rangle$ and propagate the constraints from $B^* = \{ \beta \in B : \beta \text{ is not simple} \}$ into the subtrees.
Embeddings of $\beta \in B^*$ into $\pi \in S_C$

- **Example:** for $\Theta[C^-, C][231]$, and for the embedding $(23, 1) \mapsto (2, 1)$, we get $C^-\langle 12 \rangle$.
- additional restrictions $\alpha$ in $B^?$ that are blocks of $\beta \in B^*$
- and do it inductively while new constraints $\alpha$ appear
- this terminates since each $\alpha \preceq \beta$ for some $\beta \in B^*$
Constraint propagation 1/2

**Embeddings** of $\beta \in B^*$ into $\pi \in S_C$

- **Example:** for $\ominus[C^-C\langle231\rangle$, and for the embedding $(23,1) \hookrightarrow (2,1)$, we get $C^-\langle12\rangle$.

- additional restrictions $\alpha$ in $B^?$ that are blocks of $\beta \in B^*$

- and do it inductively while new constraints $\alpha$ appear

- this terminates since each $\alpha \preceq \beta$ for some $\beta \in B^*$

**Result:** A system describing $C$, that may be ambiguous

**Example:** For $2413[C,C,C,C]\langle1234\rangle$, the embeddings $(1,234) \hookrightarrow (2,4)$ and $(1,234) \hookrightarrow (1,3)$ produce the terms $2413[C,C\langle123\rangle,C,C]$ and $2413[C,C,C,C\langle123\rangle]$ whose intersection is not empty.
Constraint propagation 2/2

Disambiguation of the system:

- Use formulas of the type $A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B}$
- In complement set, excluded patterns become mandatory patterns: $C_{\gamma}$ for $\gamma \preceq \beta \in B^*$
- Propagate also mandatory restrictions
Disambiguation of the system:

- Use formulas of the type $A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B}$
- In complement set, excluded patterns become mandatory patterns: $C_\gamma$ for $\gamma \preceq \beta \in B^*$
- Propagate also mandatory restrictions

**Result:** An unambiguous system describing $C$, where the left-hand-sides are $C_{\gamma_1, \ldots, \gamma_p}^{\varepsilon}\langle \alpha_1, \ldots, \alpha_k \rangle$ with $\varepsilon \in \{\ -, +, -\}$.

**Termination:** all $\alpha_i$ and $\gamma_j$ are patterns of some $\beta \in B^*$
An algorithm from the finite basis to the specification

Constraint propagation 2/2

Disambiguation of the system:
- Use formulas of the type $A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B}$
- In complement set, excluded patterns become mandatory patterns: $C_{\gamma}$ for $\gamma \preceq \beta \in B^*$
- Propagate also mandatory restrictions

Result: An unambiguous system describing $C$, where the left-hand-sides are $C_{\gamma_1, \ldots, \gamma_p}^{\varepsilon}(\alpha_1, \ldots, \alpha_k)$ with $\varepsilon \in \{\ , +, -\}$.

Termination: all $\alpha_i$ and $\gamma_j$ are patterns of some $\beta \in B^*$

Theorem: The propagation of the constraints to obtain a specification for $C$ is algorithmic, but there is an explosion of the number of equations in the system.
An algorithm from the finite basis to the specification

**Putting things together**

<table>
<thead>
<tr>
<th>B: finite basis of excluded patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>B contains only simple permutations</td>
</tr>
<tr>
<td>Av(B) is substitution-closed</td>
</tr>
</tbody>
</table>

**Is there a finite number of simple permutations in the class C = Av(B)?**

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(n \log n) )</td>
<td>( O(n^{3k}) )</td>
</tr>
</tbody>
</table>

**Computation of the subset Sc of simple permutations in C**

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(N, \ell^4) )</td>
<td>( O(N, \ell^{p+2}</td>
</tr>
</tbody>
</table>

**Specification for C**

<table>
<thead>
<tr>
<th>Constraints propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponential (?)</td>
</tr>
</tbody>
</table>

**Generating function and Boltzmann sampler**
Outline

1. Permutations, patterns and permutation classes
2. Substitution decomposition and decomposition trees
3. Permutations and trees as combinatorial structures
4. An algorithm from the finite basis to the specification
5. Perspectives

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes
What next?

About the algorithm:
- Implementation in progress
- Complexity analysis of step 3 (explosion of the system)
- Dependency of the complexity of Boltzmann random samplers w.r.t. the size of the specification

With the algorithm:
- From the specifications, estimate growth rates of classes
- Are random permutations in $C$ “like” in $\mathcal{G}$?
- Compare statistics on $C$ and $\mathcal{G}$, or on $C_1$ and $C_2$

Related questions:
- How general is our algorithm?
- Classes with infinite set of simples, but finitely described?
- Use specification of a class to decide membership efficiently?
Almost 30 000 permutations of size 500 in $Av(2413, 1243, 2341, 531642, 41352)$
Prop: \( \mathcal{C} = \text{Av}(B) \) is substitution-closed iff \( B \) contains only simple permutations.

For simple \( \beta \), \( \beta \preceq \pi \) translates into a factor relation on pin words. 

\[ \Rightarrow B \text{ gives a set of factors } F \text{ (whose lengths sum to } \mathcal{O}(n) \text{) such that } \]

\( w \) has a factor in \( F \) iff \( \beta \preceq \pi_w \) for some \( \beta \in B \)

[Aho & Corasick 75]:
build in linear time a complete deterministic automaton \( A_F \) recognizing the language of words containing a factor in \( F \)

\( \mathcal{L}(A_F) \) co-finite iff finite number of simples in \( \mathcal{C} \)

\( \ldots \) and testing the co-finiteness of \( \mathcal{L}(A_F) \) is in linear time.

**Theorem:** Testing the finiteness of the number of simple permutations in a substitution-closed class is solved in \( \mathcal{O}(n \log n) \)
Polynomial algorithm for general classes

When $\beta$ is not simple (but is a pin permutation), $\beta \preceq \pi$ translates into a piecewise factor relation on pin words.

**Def:** $f = (f_1, f_2, \ldots, f_k)$ is a piecewise factor of $w$ iff $w = w_0 f_1 w_1 f_2 w_2 \ldots w_{k-1} f_k w_k$.

Piecewise factors $F_\beta$ corresponding to $\beta \in B$ are computed inductively on the decomposition trees of $\beta$.

And similarly for the deterministic automaton $A_\beta$ recognizing the language of words containing a piecewise factor in $F_\beta$.

Construction of $A_\beta$ in $O(|\beta|^3)$.

Then build the product of the $A_\beta$ for $\beta \in B$ (deterministic union).

**Theorem:** Testing the finiteness of the number of simple permutations in a permutation class is solved in $O(n^{3k})$. 

Mathilde Bouvel
Algorithmic methodology for the enumeration and random generation of permutation classes