Combinatorial specifications of permutation classes, via their decomposition trees

Mathilde Bouvel  
(Institut für Mathematik, Universität Zürich)

talk based on joint works with  
F. Bassino, A. Pierrot, C. Pivoteau, D. Rossin

Journées Aléa 2014
Combinatorial specifications and trees
A combinatorial specification describes (most of the time, recursively) a combinatorial class $C$ (= a family of discrete objects) by ways of atoms and admissible constructions, like disjoint union, product, sequence, ... 

Examples:

\[
\mathcal{D} = \varepsilon + uDdD ; \quad \begin{cases}
\mathcal{T} = \mathcal{U} + \mathcal{B} \\
\mathcal{U} = \bullet + \begin{array}{c}
\mathcal{B} \\
\circ + \begin{array}{c}
\mathcal{U} \\
\mathcal{U}
\end{array}
\end{array}
\end{cases} \quad \begin{cases}
\mathcal{A}_1 = \Phi_1(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p) \\
\mathcal{A}_2 = \Phi_2(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p) \\
\ldots \\
\mathcal{A}_p = \Phi_p(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p)
\end{cases}
\]
A **combinatorial specification** describes (most of the time, recursively) a combinatorial class $C$ (= a family of discrete objects) by ways of atoms and admissible constructions, like disjoint union, product, sequence, ... 

**Examples:**

\[
D = \varepsilon + uDdD ; \\
\begin{align*}
T &= U + B \\
U &= \bullet + B \\
B &= \circ + U \cup U
\end{align*}
\]

\[
\begin{align*}
A_1 &= \Phi_1(A_1, A_2, \ldots, A_p) \\
A_2 &= \Phi_2(A_1, A_2, \ldots, A_p) \\
\vdots
\end{align*}
\]

**Systematic transcription** of a specification into:

- System of equations for the generating function $C(z) = \sum c_n z^n$  
  [Flajolet & Sedgewick 09]

- Recursive [Flajolet, Zimmerman & Van Cutsem 94] and Boltzmann random samplers [Duchon, Flajolet, Louchard & Schaeffer 04]
Consider classes of (unlabeled ordered) trees, with nodes from a (finite) set, possibly with some restrictions on the children of a node.

These may be described by a specification using disjoint union, product (and sequence).
Consider classes of (unlabeled ordered) trees, with nodes from a (finite) set, possibly with some restrictions on the children of a node.

These may be described by a specification using disjoint union, product (and sequence).

A specification is like an unambiguous context-free grammar of trees.
Consider classes of (unlabeled ordered) trees, with nodes from a (finite) set, possibly with some restrictions on the children of a node.

These may be described by a specification using disjoint union, product (and sequence).

A specification is like an unambiguous context-free grammar of trees.

“Trees are the prototypical recursive structure” [Flajolet & Sedgewick 09]

They are (one of) the most studied combinatorial objects, and a lot is known about them, both for specific classes of trees, but also for families of classes of trees.
Substitution decomposition and decomposition trees
Substitution decomposition of combinatorial objects

Combinatorial analogue of the decomposition of integers as \textit{products of primes}. Applies to relations, graphs, posets, boolean functions, set systems, \ldots and permutations [Möhring & Radermacher 84]
Substitution decomposition of combinatorial objects

Combinatorial analogue of the decomposition of integers as products of primes. Applies to relations, graphs, posets, boolean functions, set systems, ... and permutations

[Möhring & Radermacher 84]

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 (recursively) decomposed using only “basic objects”
- this decomposition is unique
**Permutations**

**Permutation of size** $n = \text{Bijection from } [1..n] \text{ to itself.}

Set $\mathfrak{S}_n$, and $\mathfrak{S} = \bigcup_n \mathfrak{S}_n$.

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

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

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

**Graphical description, or diagram:**

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\]
**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 with no interval, except the trivial ones: 1, 2, \ldots, n and $\sigma$
Example: 3 1 7 4 6 2 5 is simple
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 with no interval, except the trivial ones: 1, 2, …, $n$ and $\sigma$
Example: 3 1 7 4 6 2 5 is simple

The smallest simple permutations:
12, 21, 2413, 3142, 6 of size 5, …

Remark:
It is convenient to consider 12 and 21 not simple.
**Theorem**: [Albert, Atkinson & Klazar 03]

Every $\sigma (\neq 1)$ is **uniquely** decomposed as

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

**Notations:**

- $\oplus$-indecomposable: that cannot be written as $\oplus[\beta^{(1)}, \beta^{(2)}]$
- $\ominus$-indecomposable: that cannot be written as $\ominus[\beta^{(1)}, \beta^{(2)}]$
**Theorem**: [Albert, Atkinson & Klazar 03]

Every $\sigma \neq 1$ is uniquely decomposed as

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

**Notations:**

- $\oplus$-indecomposable: that cannot be written as $\oplus[\beta^{(1)}, \beta^{(2)}]$ 
- $\ominus$-indecomposable: that cannot be written as $\ominus[\beta^{(1)}, \beta^{(2)}]$ 

**Observation:** Equivalently, we may replace the first two items by

- $12\ldots k[\alpha^{(1)}, \ldots, \alpha^{(k)}] = \oplus[\alpha^{(1)}, \ldots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\oplus$-indecomposable
- $k\ldots 21[\alpha^{(1)}, \ldots, \alpha^{(k)}] = \ominus[\alpha^{(1)}, \ldots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\ominus$-indecomposable
Theorem: [Albert, Atkinson & Klazar 03]

Every \( \sigma (\neq 1) \) is uniquely decomposed as

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

Notations:

- \( \oplus \)-indecomposable: that cannot be written as \( \oplus[\beta^{(1)}, \beta^{(2)}] \)
- \( \ominus \)-indecomposable: that cannot be written as \( \ominus[\beta^{(1)}, \beta^{(2)}] \)

Observation: Equivalently, we may replace the first two items by

- \( 12 \ldots k[\alpha^{(1)}, \ldots, \alpha^{(k)}] = \oplus[\alpha^{(1)}, \ldots, \alpha^{(k)}] \), where the \( \alpha^{(i)} \) are \( \oplus \)-indecomposable
- \( k \ldots 21[\alpha^{(1)}, \ldots, \alpha^{(k)}] = \ominus[\alpha^{(1)}, \ldots, \alpha^{(k)}] \), where the \( \alpha^{(i)} \) are \( \ominus \)-indecomposable

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

Example: Decomposition tree of
\[ \sigma = 10 \, 13 \, 12 \, 11 \, 14 \, 1 \, 18 \, 19 \, 20 \, 21 \, 17 \, 16 \, 15 \, 4 \, 8 \, 3 \, 2 \, 9 \, 5 \, 6 \, 7 \]

\[ \begin{array}{ccccccccc}
& & & & & & & & 3 \, 1 \, 4 \, 2 \\
& & & & & & & & \oplus \ \\
& & & & & & & & \ominus \ \\
& & & & & & & & \oplus \ \\
& & & & & & & & 2 \, 4 \, 1 \, 5 \, 3 \\
\end{array} \]

Notations and properties:

- \( \oplus = 12 \ldots k \), \( \ominus = k \ldots 21 \) = linear nodes.
- \( \pi \) simple of size \( \geq 4 \) = prime node.
- No edge \( \oplus - \oplus \) nor \( \ominus - \ominus \).
- Rooted ordered trees.
- These conditions characterize decomposition trees.

\[ \sigma = 3 \, 1 \, 4 \, 2[\oplus[1, \ominus[1, 1, 1], 1], 1, \ominus[\oplus[1, 1, 1], 1, 1, 1], 1, 1, 1, 1, 2 \, 4 \, 1 \, 5 \, 3[1, 1, \ominus[1, 1], 1, 1, 1, \oplus[1, 1, 1]]] \]
Decomposition tree: witness of this decomposition

Example: Decomposition tree of
\[ \sigma = 10 \ 13 \ 12 \ 11 \ 14 \ 1 \ 18 \ 19 \ 20 \ 21 \ 17 \ 16 \ 15 \ 4 \ 8 \ 3 \ 2 \ 9 \ 5 \ 6 \ 7 \]

\[
\begin{array}{c}
3 \\
\oplus
\end{array}
\quad \begin{array}{c}
142
\end{array}
\quad \begin{array}{c}
\ominus
\end{array}
\quad \begin{array}{c}
24153
\end{array}
\]

Notations and properties:
- \( \oplus = 12 \ldots k, \ominus = k \ldots 21 \) = linear nodes.
- \( \pi \) simple of size \( \geq 4 \) = prime node.
- No edge \( \oplus - \oplus \) nor \( \ominus - \ominus \).
- Rooted ordered trees.
- These conditions characterize decomposition trees.

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

Observation: Adapts to binary case via

\[
T_1 \quad T_2 \quad \cdots \quad T_k \quad \mapsto \quad T_1 \quad \begin{array}{c}
\ominus
\end{array} \quad T_2 \quad \cdots \quad T_k
\]
Decomposition tree: witness of this decomposition

**Example:** Decomposition tree of 
\[ \sigma = 10\ 13\ 12\ 11\ 14\ 1\ 18\ 19\ 20\ 21\ 17\ 16\ 15\ 4\ 8\ 3\ 2\ 9\ 5\ 6\ 7 \]

Notations and properties:
- \( \oplus = 12 \ldots k \), \( \ominus = k \ldots 21 \) = linear nodes.
- \( \pi \) simple of size \( \geq 4 \) = prime node.
- No edge \( \oplus - \oplus \) nor \( \ominus - \ominus \).
- Rooted ordered trees.
- These conditions characterize decomposition trees.

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

**Bijection** between permutations and their decomposition trees.
Decomposition tree: witness of this decomposition

Example: Decomposition tree of
\[ \sigma = \begin{array}{cccccccccccccccccccccccccccccc} 10 & 13 & 12 & 11 & 14 & 1 & 18 & 19 & 20 & 21 & 17 & 16 & 15 & 4 & 8 & 3 & 2 & 9 & 5 & 6 & 7 
\end{array} \]

\[
\begin{array}{c}
3 & 1 & 4 & 2 \\
\oplus & & \oplus & 2 & 4 & 1 & 5 & 3 \\
& & \oplus & & \oplus & & \oplus & & \oplus
\end{array}
\]

Notations and properties:

- \( \oplus = 12 \ldots k, \ominus = k \ldots 21 \)
  - = linear nodes.
- \( \pi \) simple of size \( \geq 4 \)
  - = prime node.
- No edge \( \oplus - \oplus \) nor \( \ominus - \ominus \).
- Rooted ordered trees.
- These conditions characterize decomposition trees.

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

Bijection between permutations and their decomposition trees.

Computation: Linear time algorithm [Uno & Yagiura 00] [Bui Xuan, Habib & Paul 05] [Bergeron, Chauve, Montgolfier & Raffinot 08]
$S$ denotes the set of simple permutations

\[
S = \bullet + G^+ G^- + \sum_{\pi \in S} \pi
\]

\[
G^+ = \bullet + G^- G + \sum_{\pi \in S} \pi
\]

\[
G^- = \bullet + G^+ G + \sum_{\pi \in S} \pi
\]
A tree grammar for permutations

$S$ denotes the set of simple permutations, $S(z)$ their generating function.

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

Allows to relate the (ordinary) generating function for simples with that of all permutations ($F(z) = \sum n!z^n$) [Albert, Atkinson & Klazar 03]:

$$
\begin{align*}
F(z) &= z + 2I(z)F(z) + (S \circ F)(z) \\
I(z) &= z + I(z)F(z) + (S \circ F)(z).
\end{align*}
$$
A tree grammar for permutations

$S$ denotes the set of simple permutations, $S(z)$ their generating function.

\[
\begin{align*}
\mathcal{G} &= \bullet + \mathcal{G}^+ \oplus \mathcal{G}^- \oplus \sum_{\pi \in S} \mathcal{G} \mathcal{G} \ldots \mathcal{G} \\
\mathcal{G}^+ &= \bullet + \mathcal{G}^- \oplus \sum_{\pi \in S} \mathcal{G} \mathcal{G} \ldots \mathcal{G} \\
\mathcal{G}^- &= \bullet + \mathcal{G}^+ \oplus \sum_{\pi \in S} \mathcal{G} \mathcal{G} \ldots \mathcal{G}
\end{align*}
\]

Allows to relate the \textit{(ordinary) generating function} for simples with that of all permutations ($F(z) = \sum n!z^n$) [Albert, Atkinson & Klazar 03]:

\[
\begin{align*}
F(z) &= z + 2I(z)F(z) + (S \circ F)(z) \\
I(z) &= z + I(z)F(z) + (S \circ F)(z).
\end{align*}
\]

Consequences for the \textit{enumeration} of simple permutations:

- Asymptotically $\frac{n!}{e^2}$, but no exact enumeration.
- The generating function is not D-finite.
A tree grammar for permutations

\( S \) denotes the set of simple permutations, \( S(z) \) their generating function.

\[
\begin{align*}
\mathcal{G} &= \bullet + \left( \mathcal{G}^+ + \mathcal{G}^- + \sum_{\pi \in S} \pi \right) \mathcal{G} \mathcal{G} \cdots \mathcal{G} \\
\mathcal{G}^+ &= \bullet + \left( \mathcal{G}^- + \sum_{\pi \in S} \pi \right) \mathcal{G}^+ \mathcal{G} \cdots \mathcal{G} \\
\mathcal{G}^- &= \bullet + \left( \mathcal{G}^+ + \sum_{\pi \in S} \pi \right) \mathcal{G}^- \mathcal{G} \cdots \mathcal{G}
\end{align*}
\]

Can we specialize this tree grammar to subsets of \( \mathcal{G} \), and in particular to permutation classes \( C \)?

Can we do it automatically? even algorithmically?

Yes, when the number of simple permutations in \( C \) is finite.
Permutation patterns
and permutation classes
Pattern relation $\preceq$:
$\pi \in S_k$ is a pattern of $\sigma \in S_n$ if $\exists 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is in the same relative order ($\equiv$) as $\pi$.

Notation: $\pi \preceq \sigma$.

Equivalently:
The normalization of $\sigma_{i_1} \ldots \sigma_{i_k}$ on $[1..k]$ yields $\pi$. 
Pattern relation $\preceq$:

$\pi \in S_k$ is a pattern of $\sigma \in S_n$ if $\exists \ 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is in the same relative order ($\equiv$) as $\pi$.

Notation: $\pi \preceq \sigma$.

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

Example: $2 \ 1 \ 3 \ 4 \preceq 3 \ 1 \ 2 \ 8 \ 5 \ 4 \ 7 \ 9 \ 6$ since $3 \ 1 \ 5 \ 7 \equiv 2 \ 1 \ 3 \ 4$. 
**Pattern relation** $\preceq$:

$\pi \in \mathcal{S}_k$ is a pattern of $\sigma \in \mathcal{S}_n$ if $\exists \ 1 \leq i_1 < \ldots < i_k \leq n$ such that $\sigma_{i_1} \ldots \sigma_{i_k}$ is in the same relative order ($\equiv$) as $\pi$.

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 $3157 \equiv 2134$. 
**Permutation patterns**

**Pattern relation \(\preccurlyeq\):**

\(\pi \in \mathcal{S}_k\) is a pattern of \(\sigma \in \mathcal{S}_n\) if \(\exists 1 \leq i_1 < \ldots < i_k \leq n\) such that \(\sigma_{i_1} \ldots \sigma_{i_k}\) is in the same relative order (\(\equiv\)) as \(\pi\).

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

*Equivalently:*

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

**Example:** \(2 1 3 4 \preccurlyeq 3 1 2 8 5 4 7 9 6\) since \(3 1 5 7 \equiv 2 1 3 4\).
Pattern relation $\preceq$:
\( \pi \in \mathcal{S}_k \) is a pattern of \( \sigma \in \mathcal{S}_n \) if \( \exists 1 \leq i_1 < \ldots < i_k \leq n \) such that \( \sigma_{i_1} \ldots \sigma_{i_k} \) is in the same relative order (\( \equiv \)) as \( \pi \).

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

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

Example: \( 2 \ 1 \ 3 \ 4 \preceq 3 \ 1 \ 2 \ 8 \ 5 \ 4 \ 7 \ 9 \ 6 \)
since \( 3 \ 1 \ 5 \ 7 \equiv 2 \ 1 \ 3 \ 4 \).

Observation: \( \preceq \) is a partial order on \( \mathcal{S} = \bigcup_n \mathcal{S}_n \).
This is the key to defining permutation classes.
A permutation class is a set $C$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in C$, then $\pi \in C$. 
A permutation class is a set $\mathcal{C}$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.

$\ldots$ means $\pi \preceq \sigma$
A permutation class is a set $\mathcal{C}$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.
• A **permutation class** is a set $\mathcal{C}$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.

\[ \begin{array}{c}
1234 \ldots 1423 \ldots 3142 \ldots 4321 \\
123 \ldots 132 \ldots 213 \ldots 312 \ldots 231 \ldots 321 \\
12 \ldots 21 \ldots 1 \\
\end{array} \]

\[ \sigma \quad \text{means} \quad \pi \preceq \sigma \]
A **permutation class** is a set $\mathcal{C}$ of permutations that is downward closed for $\preccurlyeq$, i.e. whenever $\pi \preccurlyeq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$. 

\[ \sigma \preccurlyeq \sigma \]

\[ \pi \preccurlyeq \sigma \]

... means $\pi \preccurlyeq \sigma$
• A **permutation class** is a set $C$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in C$, then $\pi \in C$.

• **Notations:** $Av(\pi) = \text{the set of permutations that avoid the pattern } \pi$
  
  $Av(B) = \bigcap_{\pi \in B} Av(\pi)$

• **Fact:** For every permutation class $C$, $C = Av(B)$ for
  
  $B = \{\sigma \notin C : \forall \pi \preceq \sigma \text{ such that } \pi \neq \sigma, \pi \in C\}$.

  $B$ is an antichain, called the **basis** of $C$. 
Permutation classes

- A permutation class is a set \( C \) of permutations that is downward closed for \( \preceq \), i.e. whenever \( \pi \preceq \sigma \) and \( \sigma \in C \), then \( \pi \in C \).

- Notations: \( \text{Av}(\pi) = \) the set of permutations that avoid the pattern \( \pi \)
  \[ \text{Av}(B) = \bigcap_{\pi \in B} \text{Av}(\pi) \]

- Fact: For every permutation class \( C \), \( C = \text{Av}(B) \) for
  \[ B = \{ \sigma \notin C : \forall \pi \preceq \sigma \text{ such that } \pi \neq \sigma, \pi \in C \} \]
  
  \( B \) is an antichain, called the basis of \( C \).

- Observations:
  - Conversely, every set \( \text{Av}(B) \) is a permutation class.
  - There exist infinite antichains, hence some permutation classes have infinite basis.
Permutation classes

• A permutation class is a set $\mathcal{C}$ of permutations that is downward closed for $\preceq$, i.e. whenever $\pi \preceq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.

• Notations: $Av(\pi) =$ the set of permutations that avoid the pattern $\pi$
  
  $Av(\mathcal{B}) = \bigcap_{\pi \in \mathcal{B}} Av(\pi)$

• Fact: For every permutation class $\mathcal{C}$, $\mathcal{C} = Av(\mathcal{B})$ for
  
  $\mathcal{B} = \{ \sigma \notin \mathcal{C} : \forall \pi \preceq \sigma \text{ such that } \pi \neq \sigma, \pi \in \mathcal{C} \}$.  

  $\mathcal{B}$ is an antichain, called the basis of $\mathcal{C}$.

• Observations:
  
  • Conversely, every set $Av(\mathcal{B})$ is a permutation class.

  • There exist infinite antichains, hence some permutation classes have infinite basis.

  • In this talk, we focus on classes with finite basis.
Main steps of an algorithm to compute a specification

**Data:** $B$ a finite set of permutations

- We are interested in $C = \text{Av}(B)$. 

Mathilde Bouvel (I-Math, UZH)
Main steps of an algorithm to compute a specification

**Data:** $B$ a finite set of permutations
- We are interested in $C = \text{Av}(B)$.

**Step 1:** From $B$ (finite) to the simple permutations in $C$
- Test whether they are in finite number.
- If yes, compute their set $S_C$.

**Step 2:** From $B$ and $S_C$ (both finite) to a specification for $C$
- From decomposition trees, propagate constraints in the subtrees.
Main steps of an algorithm to compute a specification

**Data:** $B$ a finite set of permutations

- We are interested in $C = \text{Av}(B)$.

**Step 1:** From $B$ (finite) to the simple permutations in $C$

- Test whether they are in finite number.
- If yes, compute their set $S_C$.

**Step 2:** From $B$ and $S_C$ (both finite) to a specification for $C$

- From decomposition trees, propagate constraints in the subtrees.

**Result:** A combinatorial specification for $C$. Hence also:

- A polynomial system for the generating function.
- Efficient random samplers of permutations in $C$. 
Main steps of an algorithm to compute a specification

**Data:** $B$ a finite set of permutations

- We are interested in $C = \text{Av}(B)$.

**Step 1:** From $B$ (finite) to the simple permutations in $C$

- Test whether they are in finite number.
- If yes, compute their set $S_C$.

**Step 2:** From $B$ and $S_C$ (both finite) to a specification for $C$

- From decomposition trees, propagate constraints in the subtrees.

**Result:** A combinatorial specification for $C$. Hence also:

- A polynomial system for the generating function.
- Efficient random samplers of permutations in $C$.

**Remark:** Substitution-closed classes are a special (and easier) case.
Def.: A permutation class $\mathcal{C}$ is **substitution-closed** when $\pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}] \in \mathcal{C}$ for all $\pi, \alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)} \in \mathcal{C}$. 
One more definition: substitution-closed classes

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

$\mathcal{S}_\mathcal{C} =$ the set of simple permutations in $\mathcal{C}$.

Observation: $\mathcal{C}$ is substitution-closed iff the decomposition trees of permutations in $\mathcal{C}$ are all decomposition trees built on $\mathcal{S}_\mathcal{C}$ (and $\oplus$ and $\ominus$).
One more definition: substitution-closed classes

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

$\mathcal{S}_\mathcal{C}$ = the set of simple permutations in $\mathcal{C}$.

**Observation:** $\mathcal{C}$ is substitution-closed iff the decomposition trees of permutations in $\mathcal{C}$ are all decomposition trees built on $\mathcal{S}_\mathcal{C}$ (and $\oplus$ and $\ominus$).

**Characterization:** $\mathcal{C} = \text{Av}(B)$ is substitution-closed iff every permutation in $B$ is simple.

**Example:** $\text{Sep} = \text{Av}(2413, 3142)$ is substitution-closed. It corresponds to decomposition trees with no prime nodes ($\mathcal{S}_{\text{Sep}} = \emptyset$).
One more definition: substitution-closed classes

**Def.:** A permutation class $\mathcal{C}$ is **substitution-closed** when

$$\pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}] \in \mathcal{C} \text{ for all } \pi, \alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)} \in \mathcal{C}.$$ 

$S_{\mathcal{C}} =$ the set of simple permutations in $\mathcal{C}$.

**Observation:** $\mathcal{C}$ is substitution-closed iff the decomposition trees of permutations in $\mathcal{C}$ are all decomposition trees built on $S_{\mathcal{C}}$ (and $\oplus$ and $\ominus$).

**Characterization:** $\mathcal{C} = \mathrm{Av}(B)$ is substitution-closed iff every permutation in $B$ is simple.

**Example:** $\text{Sep} = \mathrm{Av}(2413, 3142)$ is substitution-closed. It corresponds to decomposition trees with no prime nodes ($S_{\text{Sep}} = \emptyset$).

**Def.:** The **substitution closure** $\hat{\mathcal{C}}$ of $\mathcal{C}$ is the smallest substitution-closed class containing $\mathcal{C}$.

**Characterization:** $\hat{\mathcal{C}}$ is the substitution-closed class built on $S_{\mathcal{C}}$ ($S_{\mathcal{C}} = S_{\hat{\mathcal{C}}}$).
From the finite basis of $C$

to the simple permutations in $C$
Characterizing when a class contains finitely many simples

Theorem [Brignall, Huczynska & Vatter 08]:
$C = \text{Av}(B)$ contains finitely many simple permutations iff $C$ contains:

1. finitely many parallel alternations
2. and finitely many wedge simple permutations
3. and finitely many proper pin-permutations
Characterizing when a class contains finitely many simples

**Theorem** [Brignall, Huczynska & Vatter 08]:

\( \mathcal{C} = \text{Av}(B) \) contains finitely many simple permutations iff \( \mathcal{C} \) contains:

1. finitely many parallel alternations
2. and finitely many wedge simple permutations
3. and finitely many proper pin-permutations

**Decision procedure** [Brignall, Ruskuc & Vatter 08]:

1. and 2.: tested by pattern matching of patterns of size 3, 4 in \( \beta \in B \).
Theorem [Brignall, Huczynska & Vatter 08]:
\( \mathcal{C} = \text{Av}(B) \) contains finitely many simple permutations iff \( \mathcal{C} \) contains:

1. finitely many parallel alternations
2. and finitely many wedge simple permutations
3. and finitely many proper pin-permutations

Decision procedure [Brignall, Ruškuc & Vatter 08]:

1. and 2.: tested by pattern matching of patterns of size 3, 4 in \( \beta \in B \).
2. is decidable: from automata theory, with a (one-to-many) encoding of proper pin-permutations by strict pin words.

Effectiveness is questionable. Efficiency is not even considered.
Characterizing when a class contains finitely many simples

Theorem [Brignall, Huczynska & Vatter 08]:
\( \mathcal{C} = \text{Av}(\mathcal{B}) \) contains finitely many simple permutations iff \( \mathcal{C} \) contains:

1. finitely many parallel alternations
2. and finitely many wedge simple permutations
3. and finitely many proper pin-permutations

Decision procedure [Brignall, Ruškuc & Vatter 08]:
1. and 2.: tested by pattern matching of patterns of size 3, 4 in \( \beta \in \mathcal{B} \).
2. is decidable: from automata theory, with a (one-to-many) encoding of proper pin-permutations by strict pin words.

Effectiveness is questionable. Efficiency is not even considered.

Goal: Give an efficient algorithm instead.
Testing whether $C = \text{Av}(B)$ contains finitely many simples

**Easy part**: testing whether $C$ contains finitely many parallel alternations and finitely many wedge simple permutations

$\hookrightarrow$ Solved with pattern matching of small patterns in $\beta \in B$

- in $O(n \log n)$ with $n = \sum_{\beta \in B} |\beta|$ from [Albert, Aldred, Atkinson & Holton 01].
Testing whether $\mathcal{C} = \text{Av}(B)$ contains finitely many simples

**Easy part:** testing whether $\mathcal{C}$ contains finitely many parallel alternations and finitely many wedge simple permutations

$\hookrightarrow$ Solved with pattern matching of small patterns in $\beta \in B$

- in $O(n \log n)$ with $n = \sum_{\beta \in B} |\beta|$ from [Albert, Aldred, Atkinson & Holton 01].

**Hard part:** testing whether $\mathcal{C}$ contains finitely many proper pin-permutations

$\hookrightarrow$ Solved using pin words and automata

- in $O(n \cdot 8^p + 2^k \cdot s \cdot 2^s)$ from [Brignall, Ruškuc & Vatter 08]

where $n = \sum_{\beta \in B} |\beta|$, $s \leq p = \max_{\beta \in B} |\beta|$ and $k \leq |B|$. 
Testing whether $\mathcal{C} = \text{Av}(B)$ contains finitely many simples

**Easy part**: testing whether $\mathcal{C}$ contains finitely many parallel alternations and finitely many wedge simple permutations

$\leftrightarrow$ Solved with pattern matching of small patterns in $\beta \in B$

- in $\mathcal{O}(n \log n)$ with $n = \sum_{\beta \in B} |\beta|$ from [Albert, Aldred, Atkinson & Holton 01].

**Hard part**: testing whether $\mathcal{C}$ contains finitely many proper pin-permutations

$\leftrightarrow$ Solved using pin words and automata

- in $\mathcal{O}(n \cdot 8^p + 2^k \cdot s \cdot 2^s)$ from [Brignall, Ruškuc & Vatter 08]

$\leftrightarrow$ Improvement from effective and recursive construction of deterministic and complete automata

- in $\mathcal{O}(n + s^{2k}) = \mathcal{O}(n + 2^{k \cdot 2 \log s})$ [Bassino, Bouvel, Pierrot & Rossin 14+]
- in $\mathcal{O}(n)$ if $\mathcal{C}$ is substitution-closed [Bassino, Bouvel, Pierrot & Rossin 10]

where $n = \sum_{\beta \in B} |\beta|$, $s \leq p = \max_{\beta \in B} |\beta|$ and $k \leq |B|$. 
Computing the set $S_C$ of simple permutations in $C$ . . .

(... assuming that $S_C$ is finite.)

Basic idea: Compute $S_{C,n} = S_C \cap \mathcal{S}_n$, for increasing $n$.
But when to stop?
Computing the set $S_C$ of simple permutations in $C$ …

(…assuming that $S_C$ is finite.)

Basic idea: Compute $S_{C,n} = S_C \cap S_n$, for increasing $n$.
But when to stop?

Theorem: [Albert & Atkinson 05] [Schmerl & Trotter 93]
If there is $n$ such that $C$ contains no simple permutation of size $n$ nor of size $n + 1$, then $C$ contains no simple permutation of size $\geq n$. 
Computing the set $S_C$ of simple permutations in $C$ . . .

(...assuming that $S_C$ is finite.)

Basic idea: Compute $S_{C,n} = S_C \cap \mathfrak{S}_n$, for increasing $n$.

But when to stop?

Theorem: [Albert & Atkinson 05] [Schmerl & Trotter 93]
If there is $n$ such that $C$ contains no simple permutation of size $n$ nor of size $n + 1$, then $C$ contains no simple permutation of size $\geq n$.

Algorithm to compute $S_C$:

- Naive algorithm: $O(\sum_{j=1,\ldots,\ell+2} j!j^{p+1} \cdot |B|)$
- Improved algorithm for substitution-closed classes: $O(N \cdot \ell^4)$
  Using properties of $\preceq$ on simple permutations [Pierrot & Rossin 14+]
- Adaptation to non substitution-closed classes: $O(N \cdot \ell^{p+2} \cdot |B|)$

where $N = |S_C|$, $p = \max_{\beta \in B} |\beta|$, $\ell = \max_{\pi \in S_C} |\pi|$.
From the basis of $\mathcal{C}$ and the simples in $\mathcal{C}$ to a combinatorial specification for $\mathcal{C}$
Theorem:
If $C$ contains a finite number of simple permutations, then it has a finite basis and an algebraic generating function $C(z)$. [Albert, Atkinson 2005]
Theorem:
If $C$ contains a finite number of simple permutations, then it has a finite basis and an algebraic generating function $C(z)$. [Albert, Atkinson 2005]

Constructive proof (the $GF$ part of the theorem):
- Specialize the substitution decomposition theorem to $C$.
- Obtain a (possibly ambiguous) context-free tree grammar for $C$.
- Inclusion-exclusion gives a polynomial system for $C(z)$. 
Theorem:
If $C$ contains a finite number of simple permutations, then it has a finite basis and an algebraic generating function $C(z)$. [Albert, Atkinson 2005]

Constructive proof (the GF part of the theorem):
- Specialize the substitution decomposition theorem to $C$.
- Obtain a (possibly ambiguous) context-free tree grammar for $C$.
- Inclusion-exclusion gives a polynomial system for $C(z)$.

Goals:
- Compute an unambiguous tree grammar (= a specification) for $C$.
- And do it algorithmically.
From a constructive proof to an algorithm

**Theorem:**
If $\mathcal{C}$ contains a finite number of simple permutations, then it has a finite basis and an algebraic generating function $C(z)$. [Albert, Atkinson 2005]

**Constructive proof** (the $GF$ part of the theorem):
- Specialize the substitution decomposition theorem to $\mathcal{C}$.
- Obtain a (possibly ambiguous) context-free tree grammar for $\mathcal{C}$.
- Inclusion-exclusion gives a polynomial system for $C(z)$.

**Goals:**
- Compute an *unambiguous* tree grammar (= a specification) for $\mathcal{C}$.
- And do it algorithmically.

**Remark** (on the *finite basis* part of the theorem): The real restriction is not having a finite basis, but rather containing finitely many simples.
Substitution-closed classes

\[
\begin{aligned}
C & = \bullet + \sum_{\pi \in S_c} C_{\pi} \\
C^+ & = \bullet + \sum_{\pi \in S_c} C_{\pi} \\
C^- & = \bullet + \sum_{\pi \in S_c} C_{\pi}
\end{aligned}
\]

- When \( C \) is substitution-closed, \( S_c \) immediately gives an unambiguous tree grammar for \( C \).
Substitution-closed classes \ldots to all classes

\[
\begin{aligned}
\hat{C} &= \bullet + \hat{C}^+ \hat{C}^+ \\
\hat{C}^+ &= \bullet + \hat{C}^- \hat{C}^- \\
\hat{C}^- &= \bullet + \hat{C}^+ \hat{C}^+
\end{aligned}
\]

\[ + \hat{C}^- \hat{C}^- + \sum_{\pi \in S_C} \hat{C} \hat{C} \cdots \hat{C} \]

\[ + \sum_{\pi \in S_C} \hat{C} \hat{C} \cdots \hat{C} \]

- When \( C \) is substitution-closed, 
  \( S_C \) immediately gives an unambiguous tree grammar for \( C \).

- When \( C = \text{Av}(B) \) is not substitution-closed,
  - It still holds for \( \hat{C} \), with \( S_{\hat{C}} = S_C \).
When $C$ is substitution-closed, $S_C$ immediately gives an unambiguous tree grammar for $C$.

When $C = \text{Av}(B)$ is not substitution-closed,

- It still holds for $\hat{C}$, with $S_{\hat{C}} = S_C$.
- $C = \hat{C}\langle B^* \rangle = \hat{C} \cap \text{Av}(B^*)$, where $B^* =$ the non simples in $B$
Substitution-closed classes . . . to all classes

\[
\begin{aligned}
\hat{C}\langle B^* \rangle &= \bullet + \underbrace{\hat{C}^+ \hat{C}}_{\hat{C}^+} + \underbrace{\hat{C}^- \hat{C}}_{\hat{C}^-} + \sum_{\pi \in S_C} \hat{C} \underbrace{\hat{C} \ldots \hat{C}}_{\hat{C}} \\
\hat{C}^+ &= \bullet + \underbrace{\hat{C}^- \hat{C}}_{\hat{C}^-} + \sum_{\pi \in S_C} \hat{C} \underbrace{\hat{C} \ldots \hat{C}}_{\hat{C}} \\
\hat{C}^- &= \bullet + \underbrace{\hat{C}^+ \hat{C}}_{\hat{C}^+} + \sum_{\pi \in S_C} \hat{C} \underbrace{\hat{C} \ldots \hat{C}}_{\hat{C}}
\end{aligned}
\]

- When \( C \) is substitution-closed, 
  \( S_C \) immediately gives an unambiguous tree grammar for \( C \).
- When \( C = \text{Av}(B) \) is not substitution-closed,
  - It still holds for \( \hat{C} \), with \( S_{\hat{C}} = S_C \).
  - \( C = \hat{C}\langle B^* \rangle = \hat{C} \cap \text{Av}(B^*) \), where \( B^* = \) the non simples in \( B \).
Substitution-closed classes . . . to all classes

\[
\begin{align*}
\hat{C}\langle B^* \rangle &= \bullet + \hat{C} + \hat{C}\langle B^* \rangle + \hat{C}^\prec\hat{C}\langle B^* \rangle + \sum_{\pi \in S_C} \pi \hat{C}\langle B^* \rangle \\
\hat{C}^+ &= \bullet + \hat{C} + \hat{C}^\prec\hat{C} + \sum_{\pi \in S_C} \pi \hat{C}\langle B^* \rangle \\
\hat{C}^- &= \bullet + \hat{C} + \hat{C}^\prec\hat{C} + \sum_{\pi \in S_C} \pi \hat{C}\langle B^* \rangle
\end{align*}
\]

- When \( C \) is substitution-closed, 
  \( S_C \) immediately gives an unambiguous tree grammar for \( C \).

- When \( C = \text{Av}(B) \) is not substitution-closed,
  - It still holds for \( \hat{C} \), with \( S_{\hat{C}} = S_C \).
  - \( C = \hat{C}\langle B^* \rangle = \hat{C} \cap \text{Av}(B^*) \), where \( B^* \) = the non simples in \( B \)
  - The pattern avoidance constraints need to be pushed in the subtrees, adding new equations to the system.
Substitution-closed classes ... to all classes

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

- When \( \mathcal{C} \) is substitution-closed, \( \mathcal{S}_\mathcal{C} \) immediately gives an unambiguous tree grammar for \( \mathcal{C} \).
- When \( \mathcal{C} = \text{Av}(B) \) is not substitution-closed,
  - It still holds for \( \hat{\mathcal{C}} \), with \( \mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C} \).
  - \( \mathcal{C} = \hat{\mathcal{C}} \langle B^* \rangle = \hat{\mathcal{C}} \cap \text{Av}(B^*) \), where \( B^* \) = the non simples in \( B \)
  - The pattern avoidance constraints need to be pushed in the subtrees, adding new equations to the system
Substitution-closed classes . . . to all classes

\[
\begin{aligned}
\hat{C}\langle B? \rangle &= \bullet + \hat{C}^+ \langle B? \rangle + \hat{C}^- \langle B? \rangle + \sum_{\pi \in S_C} \hat{C} \pi \hat{C} \langle B? \rangle \\
\hat{C}^+\langle B? \rangle &= \bullet + \hat{C}^- \langle B? \rangle + \sum_{\pi \in S_C} \hat{C} \pi \hat{C} \langle B? \rangle \\
\hat{C}^-\langle B? \rangle &= \bullet + \hat{C}^+ \langle B? \rangle + \sum_{\pi \in S_C} \hat{C} \pi \hat{C} \langle B? \rangle 
\end{aligned}
\]

- When \( C \) is substitution-closed, 
  \( S_C \) immediately gives an unambiguous tree grammar for \( C \).

- When \( C = Av(B) \) is not substitution-closed,
  - It still holds for \( \hat{C} \), with \( S_{\hat{C}} = S_C \).
  - \( C = \hat{C}\langle B^* \rangle = \hat{C} \cap Av(B^*) \), where \( B^* \) = the non simples in \( B \)
  - The pattern avoidance constraints need to be pushed in the subtrees, adding new equations to the system . . . and recursively so
Example: $\mathcal{C} = \text{Av}(231)$.
We have $S_\mathcal{C} = S_\hat{\mathcal{C}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle$.

$$\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle = \bullet + \hat{\mathcal{C}}^+ \langle 231 \rangle + \hat{\mathcal{C}}^- \langle 231 \rangle$$
Example: $\mathcal{C} = \text{Av}(231)$.
We have $S_{\mathcal{C}} = S_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle$.

$$\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle = \bullet + \hat{\mathcal{C}}^- \circ \hat{\mathcal{C}} \langle 231 \rangle + \hat{\mathcal{C}}^+ \circ \hat{\mathcal{C}} \langle 231 \rangle$$

$$= \bullet$$
Pushing restrictions in the subtrees

Example: $\mathcal{C} = \text{Av}(231)$.
We have $S_{\mathcal{C}} = S_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle$.

\[ \mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle = \bullet + \hat{\mathcal{C}}_+ \langle 231 \rangle + \hat{\mathcal{C}}_- \langle 231 \rangle = \bullet \]

Claim: $\begin{array}{cc}
\pentagon
\end{array} \in \text{Av}(231) \iff \sigma_L \in \text{Av}(12) \text{ and } \sigma_R \in \text{Av}(231)$
Pushing restrictions in the subtrees

Example: \( \mathcal{C} = \text{Av}(231) \).
We have \( S_{\mathcal{C}} = S_{\hat{\mathcal{C}}} = \emptyset \), and \( \mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle \).

\[
\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle = \bullet + \hat{\mathcal{C}}^+ \langle 231 \rangle + \hat{\mathcal{C}}^- \langle 231 \rangle
\]

Claim: \( T_L \rightarrow T_R = \begin{array}{c} \sigma_L \\ \sigma_R \end{array} \in \text{Av}(231) \iff \sigma_L \in \text{Av}(12) \text{ and } \sigma_R \in \text{Av}(231) \)
Pushing restrictions in the subtrees

**Example:** \( \mathcal{C} = \text{Av}(231) \).
We have \( S_\mathcal{C} = S_\hat{\mathcal{C}} = \emptyset \), and \( \mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle \).

\[
\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle = \bullet + \hat{\mathcal{C}}^+ \langle 231 \rangle + \hat{\mathcal{C}}^- \langle 231 \rangle
\]

\[
= \bullet
\]

**Claim:** \( \sigma_L \in \text{Av}(231) \iff \sigma_L \in \text{Av}(12) \) and \( \sigma_R \in \text{Av}(231) \)

Because of an embedding of 231 into 21 = \( \emptyset \):
Pushing restrictions in the subtrees

**Example:** $\mathcal{C} = \text{Av}(231)$. We have $S_\mathcal{C} = S_\widehat{\mathcal{C}} = \emptyset$, and $\mathcal{C} = \widehat{\mathcal{C}} \langle 231 \rangle$.

\[
\mathcal{C} = \widehat{\mathcal{C}} \langle 231 \rangle = \bullet + \widehat{\mathcal{C}}^+ \langle 231 \rangle + \widehat{\mathcal{C}}^- \langle 231 \rangle = \bullet + \widehat{\mathcal{C}}^+ \langle 231 \rangle \widehat{\mathcal{C}} \langle 231 \rangle + \widehat{\mathcal{C}}^- \langle 12 \rangle \widehat{\mathcal{C}} \langle 231 \rangle
\]

**Claim:** $\mathcal{T}_L \mathcal{T}_R = \begin{bmatrix} \sigma_L \\ \sigma_R \end{bmatrix} \in \text{Av}(231) \iff \sigma_L \in \text{Av}(12) \text{ and } \sigma_R \in \text{Av}(231)$

Because of an embedding of 231 into 21 = 0:

\[
= 231 \leftrightarrow 21 = \begin{array}{ccc}
\bullet & & \\
& & \\
& & \bullet
\end{array}
\]
Pushing restrictions in the subtrees

Example: \( C = \text{Av}(231) \).
We have \( S_C = S_{\hat{C}} = \emptyset \), and \( C = \hat{C}\langle 231 \rangle \).

\[
C = \hat{C}\langle 231 \rangle = \bullet + \begin{array}{c}
\hat{C}^+ \\
\hat{C}
\end{array} \langle 231 \rangle + \begin{array}{c}
\hat{C}^- \\
\hat{C}
\end{array} \langle 231 \rangle
\]

\( = \bullet + \begin{array}{c}
\hat{C}^+ \langle 231 \rangle \\
\hat{C} \langle 231 \rangle
\end{array} + \begin{array}{c}
\hat{C}^- \langle 12 \rangle \\
\hat{C} \langle 231 \rangle
\end{array} \)

\( \hat{C}^- \langle 12 \rangle = \ldots \)

Claim: \( \begin{array}{c}
\sigma_L \\
\sigma_R
\end{array} \in \text{Av}(231) \iff \sigma_L \in \text{Av}(12) \text{ and } \sigma_R \in \text{Av}(231) \)

Because of an embedding of \( 231 \) into \( 21 = \emptyset \):

\[
\begin{array}{c}
\bullet \\
\emptyset
\end{array} = 231 \leftrightarrow 21 = \begin{array}{c}
\bullet \\
\bullet
\end{array}
\]

Need of a new equation for \( \hat{C}^- \langle 12 \rangle \ldots \) And keep going
Pattern avoidance constraints in the subtrees come from embeddings of $\beta \in B^*$ into $\pi \in S_C \cup \{12, 21\}$. 
Why the grammar may be ambiguous

Pattern avoidance constraints in the subtrees come from embeddings of $\beta \in B^*$ into $\pi \in \mathcal{S}_C \cup \{12, 21\}$.

Example with $\beta = 3412$ and $\pi = 21$. Three embeddings of $\beta$ into $\pi$:

\[
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array} \rightarrow 
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array} \quad ; \\
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array} \rightarrow 
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array} \quad ; \\
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array} \rightarrow 
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array}
\]
Why the grammar may be ambiguous

Pattern avoidance constraints in the subtrees come from embeddings of \( \beta \in B^* \) into \( \pi \in S_C \cup \{12, 21\} \).

**Example** with \( \beta = 3412 \) and \( \pi = 21 \). Three embeddings of \( \beta \) into \( \pi \):

\[
\begin{align*}
\begin{array}{ccc}
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\downarrow & \downarrow & \downarrow \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array} \ ; \\
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\downarrow & \downarrow & \downarrow \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array} \ ; \\
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\downarrow & \downarrow & \downarrow \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array}
\end{array}
\end{align*}
\]

Hence:

\[
\hat{\mathcal{C}}^- \langle 3412 \rangle = \hat{\mathcal{C}}^- \langle 3412 \rangle \cap \hat{\mathcal{C}}^- \langle 12 \rangle \cap \hat{\mathcal{C}}^- \langle 3412 \rangle \cap \hat{\mathcal{C}}^- \langle 12 \rangle \cap \hat{\mathcal{C}}^- \langle 3412 \rangle \cap \hat{\mathcal{C}}^- \langle 12 \rangle = \hat{\mathcal{C}}^- \langle 12 \rangle \cup \hat{\mathcal{C}}^- \langle 3412 \rangle \cup \hat{\mathcal{C}}^- \langle 3412 \rangle \cup \hat{\mathcal{C}}^- \langle 12 \rangle
\]
Why the grammar may be ambiguous

Pattern avoidance constraints in the subtrees come from embeddings of $\beta \in B^*$ into $\pi \in S_C \cup \{12, 21\}$.

Example with $\beta = 3412$ and $\pi = 21$. Three embeddings of $\beta$ into $\pi$:

Hence:

$$\widehat{\mathcal{C}}^- \langle 3412 \rangle = \widehat{\mathcal{C}}^- \langle 3412 \rangle \cap \widehat{\mathcal{C}}^- \langle 3412 \rangle \cap (\widehat{\mathcal{C}}^- \langle 12 \rangle \cup \widehat{\mathcal{C}}^- \langle 12 \rangle)$$

$$= \widehat{\mathcal{C}}^- \langle 12 \rangle \cap \widehat{\mathcal{C}} \langle 3412 \rangle \cap \widehat{\mathcal{C}}^- \langle 3412 \rangle \cap \widehat{\mathcal{C}} \langle 12 \rangle$$

This is not a disjoint union (consider for instance 21).
Why the grammar may be ambiguous

Pattern avoidance constraints in the subtrees come from embeddings of \( \beta \in B^* \) into \( \pi \in \mathcal{S}_C \cup \{12, 21\} \).

**Example** with \( \beta = 3412 \) and \( \pi = 21 \). Three embeddings of \( \beta \) into \( \pi \):

\[
\begin{array}{ccc}
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array} \\
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array} \\
\begin{array}{ccc}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} & \rightarrow & \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\end{array}
\end{array}
\]

Hence:

\[
\hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 3412 \rangle} = \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 3412 \rangle} \cap \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 3412 \rangle} \cap (\hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 12 \rangle} \cup \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 12 \rangle})
\]

\[
= \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 12 \rangle} \cup \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 3412 \rangle} \cup \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 3412 \rangle} \cup \hat{\mathcal{C}}^\top \hat{\mathcal{C}}^{\langle 12 \rangle}
\]

This is **not** a disjoint union (consider for instance 21).

**Observation:** The new excluded patterns are some \( \alpha \preceq \beta \in B^* \)
Need of introducing pattern *containment* constraints

**Example:** Disambiguation of
\[ \hat{C}^{-}\langle 12 \rangle \cup \hat{C}\langle 3412 \rangle \cup \hat{C}^{-}\langle 3412 \rangle \hat{C}\langle 12 \rangle \].

**Method:**

- \[ A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B} \]
Need of introducing pattern *containment* constraints

**Example:** Disambiguation of

\[ \hat{C} - \langle 12 \rangle \cup \hat{C} \langle 3412 \rangle \cup \hat{C} - \langle 3412 \rangle \cup \hat{C} \langle 12 \rangle. \]

**Method:**

- \( A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B} \)
- By complementation, excluded patterns become **mandatory patterns:** \( C_\gamma \) for \( \gamma \preceq \beta \in B^* \)

\[ \hat{C} - \langle 3412 \rangle = \hat{C} - \langle 12 \rangle \cup \hat{C} \langle 12 \rangle \cup \hat{C}_{12} \langle 3412 \rangle \cup \hat{C} \langle 12 \rangle \cup \hat{C} - \langle 12 \rangle \cup \hat{C}_{12} \langle 3412 \rangle. \]

Notice that the terms \( \hat{C} - \langle 3412 \rangle \) and \( \hat{C}_{3412} \langle 12 \rangle \) and \( \hat{C}_{3412} \langle 12 \rangle \) are empty, and have been deleted.
Need of introducing pattern *containment* constraints

**Example:** Disambiguation of

\[
\begin{array}{c}
\hat{C}_{\langle 12 \rangle} \\
\cup \\
\hat{C}_{\langle 3412 \rangle} \\
\end{array}
\]

**Method:**

- \( A \cup B = A \cap B \cup \overline{A} \cap B \cup A \cap \overline{B} \)
- By complementation, excluded patterns become mandatory patterns: \( C_\gamma \) for \( \gamma \preceq \beta \in B^* \)

\[
\begin{array}{c}
\hat{C}_{\langle 3412 \rangle} = \\
\hat{C}_{\langle 12 \rangle} \hat{C}_{\langle 12 \rangle} \\
\cup \\
\hat{C}_{\langle 3412 \rangle} \hat{C}_{\langle 3412 \rangle} \\
\end{array}
\]

\( \Rightarrow \) Need to propagate avoidance and containment constraints:

\[
\hat{C}^{\varepsilon}_{\gamma_1, \ldots, \gamma_p \langle \alpha_1, \ldots, \alpha_k \rangle} \text{ with } \varepsilon \in \{ , +, - \}
\]

Observation: \( \gamma_i \) and \( \alpha_j \) are all patterns of some \( \beta \in B^* \).
A first specification for $\mathcal{C}$

Find a specification for all
\[
\hat{\mathcal{C}}_{\gamma_1, \ldots, \gamma_p}^{\varepsilon}\langle \alpha_1, \ldots, \alpha_k \rangle
\]

with $\{\gamma_1, \ldots, \gamma_p\} \subseteq \widetilde{B}^*$ and $\{\alpha_1, \ldots, \alpha_k\} \subseteq \widetilde{B}^*$,
where $\widetilde{B}^* = \{\alpha \preceq \beta \mid \beta \in B^*\} = \text{set of patterns of some } \beta \in B^*$.

How to:
For $\alpha \in \widetilde{B}^*$ and $\sigma = \pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}]$, considering embeddings of $\alpha$ in $\pi$, we can decide which patterns $\alpha$ occur in $\sigma$ from the knowledge of which patterns of $\widetilde{B}^*$ occur in $\alpha^{(i)}$, for all $1 \leq i \leq k$. 
A first specification for $\mathcal{C}$

Find a specification for all

$$\hat{C}^\epsilon_{\gamma_1,\ldots,\gamma_p} \langle \alpha_1, \ldots, \alpha_k \rangle$$

with \( \{\gamma_1, \ldots, \gamma_p\} \subseteq \tilde{B}^* \) and \( \{\alpha_1, \ldots, \alpha_k\} \subseteq \tilde{B}^* \),

where \( \tilde{B}^* = \{\alpha \preceq \beta \mid \beta \in B^*\} \) = set of patterns of some \( \beta \in B^* \).

How to:
For \( \alpha \in \tilde{B}^* \) and \( \sigma = \pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}] \),
considering embeddings of \( \alpha \) in \( \pi \),
we can decide which patterns \( \alpha \) occur in \( \sigma \)
from the knowledge of which patterns of \( \tilde{B}^* \) occur in \( \alpha^{(i)} \), for all \( 1 \leq i \leq k \).

Approach reminiscent of the query-complete sets of [Brignall, Huczynska & Vatter 08].
Computing only the necessary restrictions

**Algorithm:** [Bassino, Bouvel, Pierrot, Pivoteau & Rossin, 14+]

- Start from $C = \hat{C}\langle B^* \rangle$, $C^+$ and $C^-$, and propagate the pattern avoidance constraints in the subtrees.
- Disambiguate the equations, introducing pattern containment constraints.
- For each term $\hat{C}\langle \gamma_1, \ldots, \gamma_p \rangle\langle \alpha_1, \ldots, \alpha_k \rangle$ that appears on the RHS, repeat this process, recursively.

**Properties:**

- This algorithm terminates and produces a specification for $C$. 
Computing only the necessary restrictions

Algorithm: [Bassino, Bouvel, Pierrot, Pivoteau & Rossin, 14+]

- Start from $C = \hat{C}(B^*)$, $C^+$ and $C^-$, and propagate the pattern avoidance constraints in the subtrees.
- Disambiguate the equations, introducing pattern containment constraints.
- For each term $\hat{C}_{\gamma_1,\ldots,\gamma_p}(\alpha_1,\ldots,\alpha_k)$ that appears on the RHS, repeat this process, recursively.

Properties:
- This algorithm terminates and produces a specification for $C$.

Questions:
- What is the complexity?
- What is the size of the specification produced?
- It can be exponential in $|B|$. But how big can it be?
Algorithmic chain from $B$ finite to a specification for $C = \text{Av}(B)$. 

where $n = \sum_{\beta \in B} |\beta|$, $p = \max_{\beta \in B} |\beta|$, $k = |B|$, $N = |S_C|$, $\ell = \max_{\pi \in S_C} |\pi|$. 

Remark: It succeeds only when $C$ contains finitely many simples (and this condition is tested algorithmically).
Byproducts of specifications and perspectives
A specification for $C$ gives access to...

- A polynomial system defining $C(z)$ (implicitly)

[Flajolet & Sedgewick 09]

Can it be used to obtain information on the dominant singularity of $C(z)$, or equivalently the growth rate of $C$?
A specification for $\mathcal{C}$ gives access to...

- A polynomial system defining $C(z)$ (implicitly)
  \[\text{[Flajolet & Sedgewick 09]}\]

→ Can it be used to obtain information on the dominant singularity of $C(z)$, or equivalently the growth rate of $C$?

- Random samplers of permutations in $\mathcal{C}$:
  - by the recursive method  \[\text{[Flajolet, Zimmerman & Van Cutsem 94]}\]
  - by the Boltzmann method  \[\text{[Duchon, Flajolet, Louchard & Schaeffer 04]}\]

→ Implementation (in progress) to observe random permutations in permutation classes.

→ Can we describe the “average shape” or average properties of random permutations in permutation classes?
  For some given classes, or for families of classes?
Random permutations in permutation classes

- $C_1 = \text{Av}(2413, 3142) =$ separables.
  Substitution-closed with no simples.
  10000 permutations of size 100 in $C_1$.

- Substitution-closed class $C_2$, with simples 2413, 3142 and 24153.
  10000 permutations of size 500 in $C_2$.

- $C_3 = \text{Av}(2413, 1243, 2341, 531642, 41352)$.
  Not substitution-closed.
  Almost 30000 permutations of size 500 in $C_3$. 