Combinatorial specifications of permutation classes, via their decomposition trees

Mathilde Bouvel

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

Discrete Math Seminar, Uni. Zurich, May 2014.

Combinatorial specifications and trees

Combinatorial specifications and their byproducts

[Flajolet & Sedgewick 09]

A combinatorial specification describes (most of the time, recursively) a combinatorial class $\mathcal{C}$ (= a family of discrete objects) by ways of atoms and admissible constructions, like disjoint union, product, sequence, ...

Examples:

$$\mathcal{D} = \varepsilon + u\mathcal{D}d\mathcal{D}\,;$$

$$\begin{cases} \mathcal{T} = \mathcal{U} + \mathcal{B} \\ \mathcal{U} = \bullet + \begin{array}{c}\bullet\\ | \\ \mathcal{B}\end{array} \\ \mathcal{B} = \circ + \begin{array}{c}\circ\\ / \quad \backslash \\ \mathcal{U} \quad \mathcal{U}\end{array} \end{cases};$$

$$\begin{cases} \mathcal{A}_1 = \Phi_1(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \\ \mathcal{A}_2 = \Phi_2(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \\ \dots \\ \mathcal{A}_p = \Phi_p(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \end{cases}$$

Combinatorial specifications and their byproducts

[Flajolet & Sedgewick 09]

A combinatorial specification describes (most of the time, recursively) a combinatorial class $\mathcal{C}$ (= a family of discrete objects) by ways of atoms and admissible constructions, like disjoint union, product, sequence, ...

Examples:

$$\mathcal{D} = \varepsilon + u\mathcal{D}d\mathcal{D}\,; \qquad \begin{cases} \mathcal{T} = \mathcal{U} + \mathcal{B} \\ \mathcal{U} = \bullet + \begin{array}{c}\bullet\\ | \\ \mathcal{B}\end{array} \\ \mathcal{B} = \circ + \begin{array}{c}\circ\\ / \quad \backslash \\ \mathcal{U} \quad \mathcal{U}\end{array} \end{cases} ; \qquad \begin{cases} \mathcal{A}_1 = \Phi_1(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \\ \mathcal{A}_2 = \Phi_2(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \\ \dots \\ \mathcal{A}_p = \Phi_p(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_p) \end{cases}$$

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]

Combinatorial specifications of trees

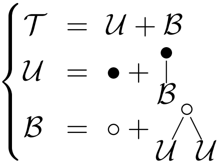
Consider classes of (unlabeled ordered) trees, with nodes from a (finite) set, possibly with some restrictions on the children of a node.

$$\begin{cases} \mathcal{T} &= \mathcal{U} + \mathcal{B} \\ \mathcal{U} &= \bullet + \overset{\bullet}{\underset{\mathcal{B}}{|}} \\ \mathcal{B} &= \circ + \overset{\circ}{\overset{/ \;\; \backslash}{\mathcal{U} \;\; \mathcal{U}}} \end{cases}$$

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

Combinatorial specifications 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.

Combinatorial specifications of trees

Consider classes of (unlabeled ordered) trees, with nodes from a (finite) set, possibly with some restrictions on the children of a node.

$$\begin{cases} \mathcal{T} = \mathcal{U} + \mathcal{B} \\ \mathcal{U} = \bullet + \overset{\bullet}{\underset{\mathcal{B}}{|}} \\ \mathcal{B} = \circ + \overset{\circ}{\diagup\diagdown}_{\mathcal{U}\ \mathcal{U}} \end{cases}$$

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 products of primes. Applies to relations, graphs, posets, boolean functions, set systems, ... 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

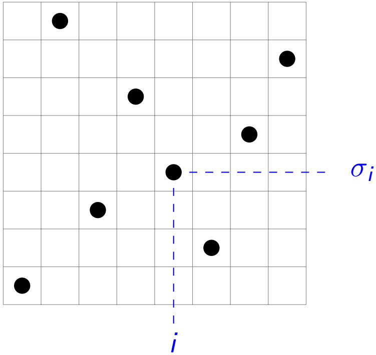
Permutation of size n = Bijection from $[1..n]$ to itself.
Set $\mathfrak{S}_n$, and $\mathfrak{S} = \cup_n \mathfrak{S}_n$.

- 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:
$\sigma = (1)\ (2\ 8\ 7\ 5\ 4\ 6)\ (3)$

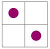
- Graphical description, or diagram:

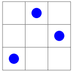


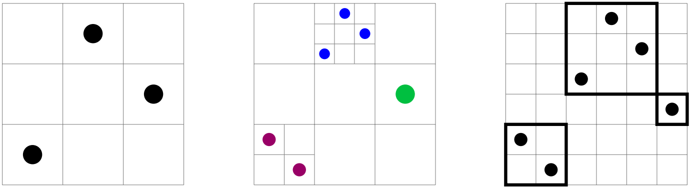
Substitution for permutations

Substitution or inflation : $\sigma = \pi[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}]$.

Example: Here, $\pi = 1\,3\,2$, and $\begin{cases} \alpha^{(1)} = 2\,1 = \\ \alpha^{(2)} = 1\,3\,2 = \\ \alpha^{(3)} = 1 = \end{cases}$.







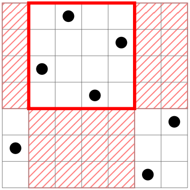
Hence $\sigma = 1\,3\,2[2\,1, 1\,3\,2, 1] = 2\,1\,4\,6\,5\,3$.

Simple permutations

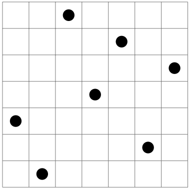
Interval (or block) = set of elements of σ whose positions **and** values form intervals of integers
Example: 5 7 4 6 is an interval of 2 5 7 4 6 1 3

Simple permutation = permutation with no interval, except the trivial ones: $1, 2, \ldots, n$ and σ
Example: 3 1 7 4 6 2 5 is simple

Not simple:



Simple:

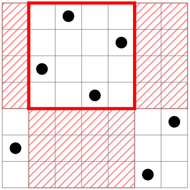


Interval (or block) = set of elements of σ whose positions **and** values form intervals of integers
Example: 5 7 4 6 is an interval of 2 5 7 4 6 1 3

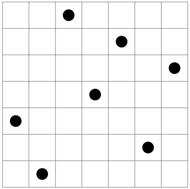
Simple permutation = permutation with no interval, except the trivial ones: $1, 2, \ldots, n$ and σ
Example: 3 1 7 4 6 2 5 is simple

The smallest simple permutations:
$12, 21, \quad 2413, 3142$, 6 of size 5, ...
Remark:
It is convenient to consider 12 and 21 **not** simple.

Not simple:



Simple:



Substitution decomposition theorem(s) for permutations

Theorem: [Albert, Atkinson & Klazar 03]
Every σ ($\neq 1$) 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
- $\pi[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where π 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 $\ominus[\beta^{(1)}, \beta^{(2)}]$

Substitution decomposition theorem(s) for permutations

Theorem: [Albert, Atkinson & Klazar 03]
Every σ ($\neq 1$) 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
- $\pi[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where π 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

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

Substitution decomposition theorem(s) for permutations

Theorem: [Albert, Atkinson & Klazar 03]
Every σ ($\neq 1$) 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
- $\pi[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where π 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

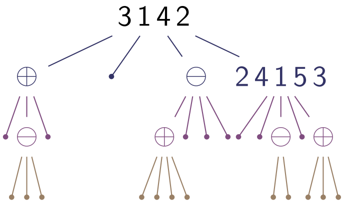
- $12 \dots k[\alpha^{(1)}, \dots, \alpha^{(k)}] = \oplus[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\oplus$-indecomposable
- $k \dots 21[\alpha^{(1)}, \dots, \alpha^{(k)}] = \ominus[\alpha^{(1)}, \dots, \alpha^{(k)}]$, where the $\alpha^{(i)}$ are $\ominus$-indecomposable

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



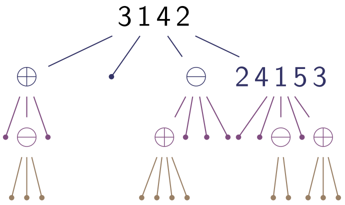
Notations and properties:

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

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

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

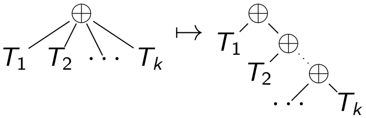


Notations and properties:

- $\oplus = 12\dots k$, $\ominus = k \dots 21$ = linear nodes.
- π simple of size ≥ 4 = prime node.
- No edge $\oplus - \oplus$ nor $\ominus - \ominus$.
- Rooted ordered trees.
- These conditions characterize decomposition trees.

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

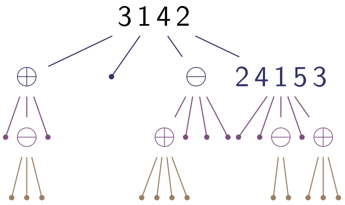
Observation: Adapts to binary case *via*



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



$\sigma = 3142[\oplus[1, \ominus[1,1,1],1],1,\ominus[\oplus[1,1,1,1],1,1,1],24153[1,1,\ominus[1,1],1,\oplus[1,1,1]]]$

Notations and properties:

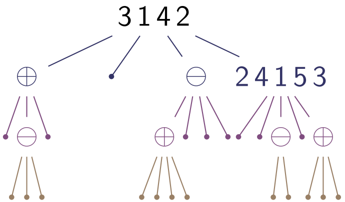
- $\oplus = 12\dots k$, $\ominus = k\dots 21$ = linear nodes.
- π simple of size ≥ 4 = prime node.
- No edge $\oplus - \oplus$ nor $\ominus - \ominus$.
- Rooted ordered trees.
- These conditions characterize decomposition trees.

Bijection between permutations and their decomposition trees.

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:

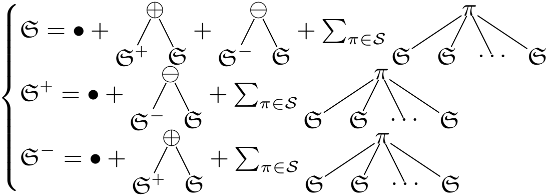
- $\oplus = 12\ldots k$, $\ominus = k\ldots 21$ $=$ linear nodes.
- π simple of size ≥ 4 $=$ prime node.
- No edge $\oplus - \oplus$ nor $\ominus - \ominus$.
- Rooted ordered trees.
- These conditions characterize decomposition trees.

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

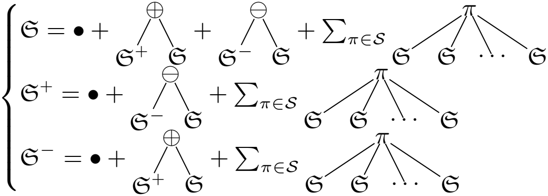
Bijection between permutations and their decomposition trees.

Computation: Linear time algorithm [Uno & Yagiura 00] [Bui Xuan, Habib & Paul 05] [Bergeron, Chauve, Montgolfier & Raffinot 08]

$\mathcal{S}$ denotes the set of simple permutations



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

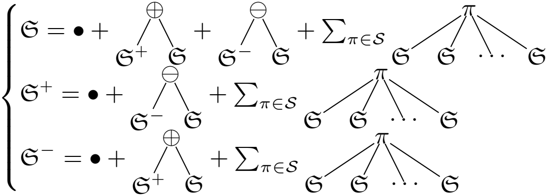


Allows to relate the (ordinary) generating function for simples with that of all permutations ($F(z) = \sum n! z^n$) [Albert, Atkinson & Klazar 03]:

$$\begin{cases} F(z) = z + 2I(z)F(z) + (S \circ F)(z) \\ I(z) = z + I(z)F(z) + (S \circ F)(z). \end{cases}$$

A tree grammar for permutations

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



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

$$\begin{cases} F(z) = z + 2I(z)F(z) + (S \circ F)(z) \\ I(z) = z + I(z)F(z) + (S \circ F)(z). \end{cases}$$

Consequences for the 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

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

$$\begin{cases} \mathfrak{S} = \bullet + \oplus(\mathfrak{S}^+, \mathfrak{S}) + \ominus(\mathfrak{S}^-, \mathfrak{S}) + \sum_{\pi \in \mathcal{S}} \pi(\mathfrak{S}, \mathfrak{S}, \ldots, \mathfrak{S}) \\ \mathfrak{S}^+ = \bullet + \ominus(\mathfrak{S}^-, \mathfrak{S}) + \sum_{\pi \in \mathcal{S}} \pi(\mathfrak{S}, \mathfrak{S}, \ldots, \mathfrak{S}) \\ \mathfrak{S}^- = \bullet + \oplus(\mathfrak{S}^+, \mathfrak{S}) + \sum_{\pi \in \mathcal{S}} \pi(\mathfrak{S}, \mathfrak{S}, \ldots, \mathfrak{S}) \end{cases}$$

Can we specialize this tree grammar to subsets of $\mathfrak{S}$, and in particular to permutation classes $\mathcal{C}$?

Can we do it automatically? even algorithmically?

Yes, when the number of simple permutations in $\mathcal{C}$ is finite.

Permutation patterns
and permutation classes

Permutation patterns

Pattern relation $\preccurlyeq$:
$\pi \in \mathfrak{S}_k$ 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 in the same relative order ($\equiv$) as π.

Notation: $\pi \preccurlyeq \sigma$.

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

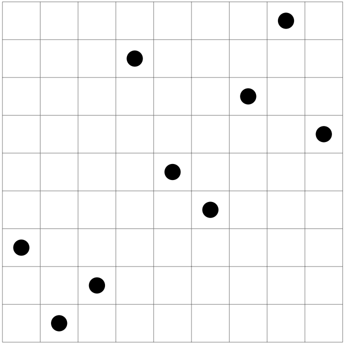
Permutation patterns

Pattern relation $\preccurlyeq$:
$\pi \in \mathfrak{S}_k$ 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 in the same relative order ($\equiv$) as π.

Notation: $\pi \preccurlyeq \sigma$.

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

Example: $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$ since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.



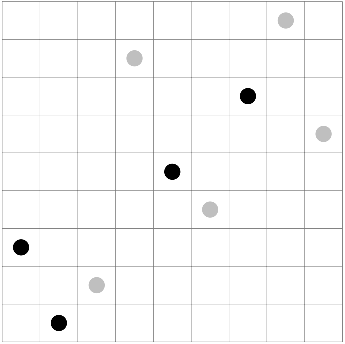
Permutation patterns

Pattern relation $\preccurlyeq$:
$\pi \in \mathfrak{S}_k$ 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 in the same relative order ($\equiv$) as π.

Notation: $\pi \preccurlyeq \sigma$.

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

Example: $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$
since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.



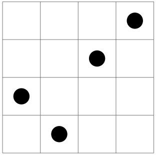
Pattern relation $\preccurlyeq$:

$\pi \in \mathfrak{S}_k$ 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 in the same relative order ($\equiv$) as π.

Notation: $\pi \preccurlyeq \sigma$.

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

Example: $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$ since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.

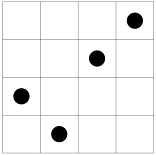


Permutation patterns

Pattern relation $\preccurlyeq$:
$\pi \in \mathfrak{S}_k$ 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 in the same relative order ($\equiv$) as π.

Notation: $\pi \preccurlyeq \sigma$.

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



Example: $2\,1\,3\,4 \preccurlyeq \mathbf{3}\,\mathbf{1}\,2\,8\,\mathbf{5}\,4\,\mathbf{7}\,9\,6$
since $3\,1\,5\,7 \equiv 2\,1\,3\,4$.

Observation: $\preccurlyeq$ is a partial order on $\mathfrak{S} = \bigcup_n \mathfrak{S}_n$.

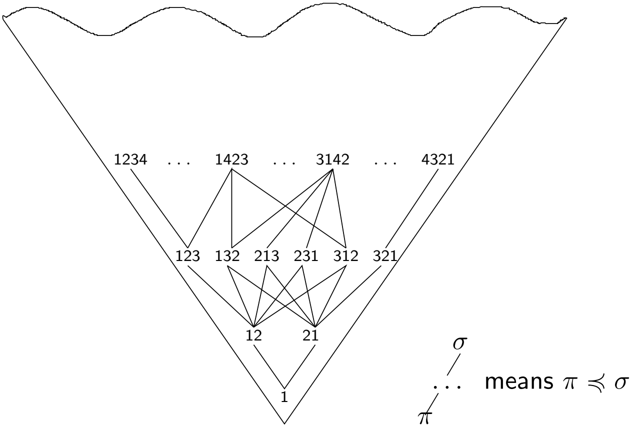
This is the key to defining permutation classes.

Permutation classes

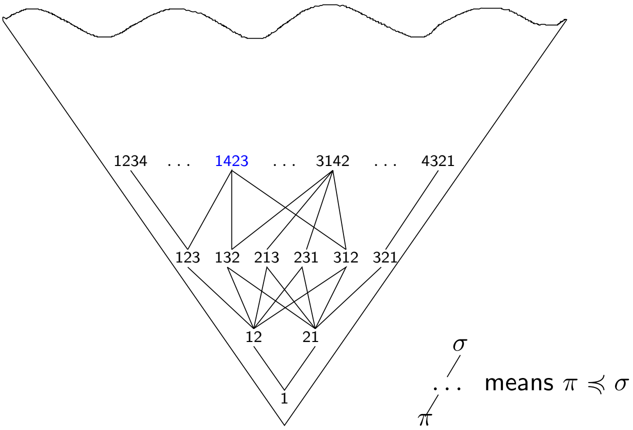
• A **permutation class** is a set $\mathcal{C}$ of permutations that is downward closed for $\preccurlyeq$, i.e. whenever $\pi \preccurlyeq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.

Permutation classes

• 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 $\pi \in \mathcal{C}$.

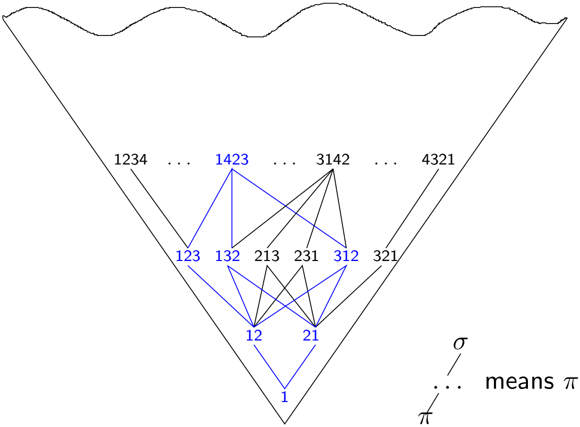


• A **permutation class** is a set $\mathcal{C}$ of permutations that is downward closed for $\preccurlyeq$, i.e. whenever $\pi \preccurlyeq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.



Permutation classes

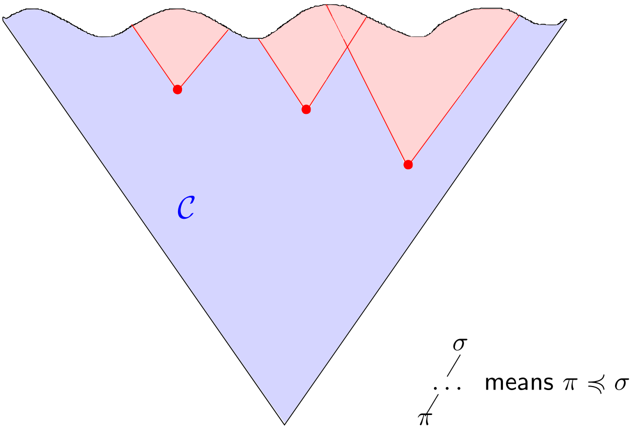
• A **permutation class** is a set $\mathcal{C}$ of permutations that is downward closed for $\preccurlyeq$, i.e. whenever $\pi \preccurlyeq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.



means $\pi \preccurlyeq \sigma$

Permutation classes

• 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 $\pi \in \mathcal{C}$.



Permutation classes

• 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 $\pi \in \mathcal{C}$.

• Notations: $Av(\pi)$ = the set of permutations that avoid the pattern π

$$Av(B) = \bigcap_{\pi \in B} Av(\pi)$$

• Fact: For every permutation class $\mathcal{C}$, $\mathcal{C} = Av(B)$ for

$$B = \{\sigma \notin \mathcal{C} : \forall \pi \preccurlyeq \sigma \text{ such that } \pi \neq \sigma, \pi \in \mathcal{C}\}.$$

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

Permutation classes

• 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 $\pi \in \mathcal{C}$.

• Notations: $Av(\pi)$ = the set of permutations that avoid the pattern π

$$Av(B) = \bigcap_{\pi \in B} Av(\pi)$$

• Fact: For every permutation class $\mathcal{C}$, $\mathcal{C} = Av(B)$ for

$$B = \{\sigma \notin \mathcal{C} : \forall \pi \preccurlyeq \sigma \text{ such that } \pi \neq \sigma, \pi \in \mathcal{C}\}.$$

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

• Observations:

- Conversely, every set $Av(B)$ 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 $\preccurlyeq$, i.e. whenever $\pi \preccurlyeq \sigma$ and $\sigma \in \mathcal{C}$, then $\pi \in \mathcal{C}$.

• Notations: $Av(\pi)$ = the set of permutations that avoid the pattern π

$$Av(B) = \bigcap_{\pi \in B} Av(\pi)$$

• Fact: For every permutation class $\mathcal{C}$, $\mathcal{C} = Av(B)$ for

$$B = \{\sigma \notin \mathcal{C} : \forall \pi \preccurlyeq \sigma \text{ such that } \pi \neq \sigma, \pi \in \mathcal{C}\}.$$

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

• Observations:

- Conversely, every set $Av(B)$ 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 $\mathcal{C} = \mathsf{Av}(B)$.

Main steps of an algorithm to compute a specification

Data: B a finite set of permutations

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

Step 1: From B (finite) to the simple permutations in $\mathcal{C}$

- Test whether they are in finite number.
- If yes, compute their set $\mathcal{S}_{\mathcal{C}}$.

Step 2: From B and $\mathcal{S}_{\mathcal{C}}$ (both finite) to a specification for $\mathcal{C}$

- 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 $\mathcal{C} = \mathrm{Av}(B)$.

Step 1: From B (finite) to the simple permutations in $\mathcal{C}$

- Test whether they are in finite number.
- If yes, compute their set $\mathcal{S}_{\mathcal{C}}$.

Step 2: From B and $\mathcal{S}_{\mathcal{C}}$ (both finite) to a specification for $\mathcal{C}$

- From decomposition trees, propagate constraints in the subtrees.

Result: A combinatorial specification for $\mathcal{C}$. Hence also:

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

Main steps of an algorithm to compute a specification

Data: B a finite set of permutations

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

Step 1: From B (finite) to the simple permutations in $\mathcal{C}$

- Test whether they are in finite number.
- If yes, compute their set $\mathcal{S}_{\mathcal{C}}$.

Step 2: From B and $\mathcal{S}_{\mathcal{C}}$ (both finite) to a specification for $\mathcal{C}$

- From decomposition trees, propagate constraints in the subtrees.

Result: A combinatorial specification for $\mathcal{C}$. Hence also:

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

Remark: Substitution-closed classes are a special (and easier) case.

One more definition: substitution-closed classes

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

One more definition: substitution-closed classes

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

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

One more definition: substitution-closed classes

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

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

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

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 $\mathcal{S}_{\mathcal{C}}$ ($\mathcal{S}_{\mathcal{C}} = \mathcal{S}_{\hat{\mathcal{C}}}$).

From the finite basis of $\mathcal{C}$ to the simple permutations in $\mathcal{C}$

Characterizing when a class contains finitely many simples

Theorem [Brignall, Huczynska & Vatter 08]:
$\mathcal{C} = \mathsf{Av}(B)$ 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

Characterizing when a class contains finitely many simples

Theorem [Brignall, Huczynska & Vatter 08]:
$\mathcal{C} = \mathsf{Av}(B)$ 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

Testing 1., 2. and 3. decides whether $\mathcal{C}$ contains finitely many simples.

1.,2. are tested by pattern matching of patterns of size 3, 4 in $\beta \in B$
[Brignall, Ruškuc & Vatter 08]

$\hookrightarrow$ Efficient algorithm of [Albert, Aldred, Atkinson & Holton 01]

3. is tested using automata theory

$\hookrightarrow$ First (and inefficient) decision procedure in [BRV08]

$\hookrightarrow$ Algorithm of tractable complexity (more efficient for substitution-closed classes) [Bassino, Bouvel, Pierrot & Rossin 10,14+]

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

(... assuming that $\mathcal{S}_{\mathcal{C}}$ is finite.)

Basic idea: Compute $\mathcal{S}_{\mathcal{C},n} = \mathcal{S}_{\mathcal{C}} \cap \mathfrak{S}_n$, for increasing n.
But when to stop?

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

(...assuming that $\mathcal{S}_\mathcal{C}$ is finite.)

Basic idea: Compute $\mathcal{S}_{\mathcal{C},n} = \mathcal{S}_\mathcal{C} \cap \mathfrak{S}_n$, for increasing n.
But when to stop?

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

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

(... assuming that $\mathcal{S}_\mathcal{C}$ is finite.)

Basic idea: Compute $\mathcal{S}_{\mathcal{C},n} = \mathcal{S}_\mathcal{C} \cap \mathfrak{S}_n$, for increasing n.
But when to stop?

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

Algorithm to compute $\mathcal{S}_\mathcal{C}$:

- Naive algorithm from [Albert & Atkinson 05]
- Improved algorithm for substitution-closed classes
  Using properties of $\preccurlyeq$ on simple permutations [Pierrot & Rossin 14+]
- Adaptation to non substitution-closed classes [PR14+]

From the basis of $\mathcal{C}$ and the simples in $\mathcal{C}$ to a combinatorial specification for $\mathcal{C}$

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]

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 $C(z)$.

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.

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:

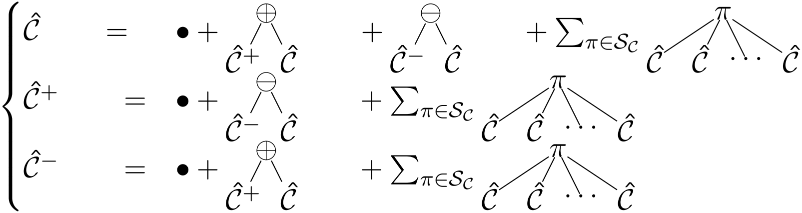
- Compute an unambiguous tree grammar (= a specification) for $\mathcal{C}$.
- And do it algorithmically.

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

$$\begin{cases} \mathcal{C} = \bullet + \oplus[\mathcal{C}^+, \mathcal{C}] + \ominus[\mathcal{C}^-, \mathcal{C}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\mathcal{C}, \mathcal{C}, \ldots, \mathcal{C}] \\ \mathcal{C}^+ = \bullet + \ominus[\mathcal{C}^-, \mathcal{C}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\mathcal{C}, \mathcal{C}, \ldots, \mathcal{C}] \\ \mathcal{C}^- = \bullet + \oplus[\mathcal{C}^+, \mathcal{C}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\mathcal{C}, \mathcal{C}, \ldots, \mathcal{C}] \end{cases}$$

- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_\mathcal{C}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.



- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_{\mathcal{C}}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.

- When $\mathcal{C} = \mathsf{Av}(B)$ is not substitution-closed,
  - It still holds for $\hat{\mathcal{C}}$, with $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_{\mathcal{C}}$.

Substitution-closed classes . . . to all classes

$$\begin{cases} \hat{\mathcal{C}} = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}] + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}] \\ \hat{\mathcal{C}}^+ = \bullet + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}] \\ \hat{\mathcal{C}}^- = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}] + \sum_{\pi \in \mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}] \end{cases}$$

- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_\mathcal{C}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.

- When $\mathcal{C} = \mathrm{Av}(B)$ is not substitution-closed,
  - It still holds for $\hat{\mathcal{C}}$, with $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$.
  - $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star \rangle = \hat{\mathcal{C}} \cap \mathrm{Av}(B^\star)$, where $B^\star =$ the non simples in B

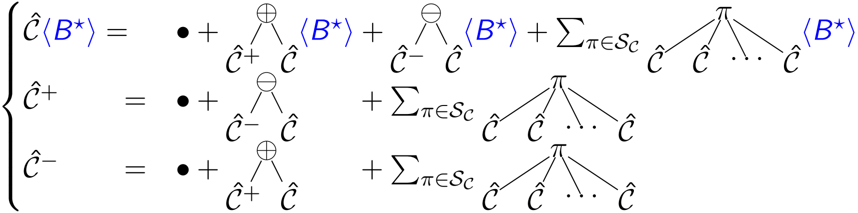
Substitution-closed classes ... to all classes

$$\begin{cases} \hat{\mathcal{C}}\langle B^\star\rangle = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}]\langle B^\star\rangle + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}]\langle B^\star\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}]\langle B^\star\rangle \\ \hat{\mathcal{C}}^+ = \bullet + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}] + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}] \\ \hat{\mathcal{C}}^- = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}] + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}] \end{cases}$$

- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_\mathcal{C}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.

- When $\mathcal{C} = \mathsf{Av}(B)$ is not substitution-closed,
  - It still holds for $\hat{\mathcal{C}}$, with $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$.
  - $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star\rangle = \hat{\mathcal{C}} \cap \mathsf{Av}(B^\star)$, where $B^\star =$ the non simples in B



- 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 $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$.
  - $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star\rangle = \hat{\mathcal{C}} \cap \text{Av}(B^\star)$, where $B^\star =$ 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

$$\begin{cases} \hat{\mathcal{C}}\langle B^?\rangle = \bullet + \oplus[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}]\langle B^?\rangle + \ominus[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^?\rangle \\ \hat{\mathcal{C}}^+\langle B^?\rangle = \bullet + \ominus[\hat{\mathcal{C}}^-,\hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^?\rangle \\ \hat{\mathcal{C}}^-\langle B^?\rangle = \bullet + \oplus[\hat{\mathcal{C}}^+,\hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}},\hat{\mathcal{C}},\dots,\hat{\mathcal{C}}]\langle B^?\rangle \end{cases}$$

- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_\mathcal{C}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.

- When $\mathcal{C} = \mathrm{Av}(B)$ is not substitution-closed,
  - It still holds for $\hat{\mathcal{C}}$, with $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$.
  - $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star\rangle = \hat{\mathcal{C}} \cap \mathrm{Av}(B^\star)$, where $B^\star =$ 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

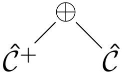
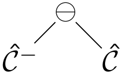
$$\begin{cases} \hat{\mathcal{C}}\langle B^?\rangle = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}]\langle B^?\rangle + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}]\langle B^?\rangle \\ \hat{\mathcal{C}}^+\langle B^?\rangle = \bullet + \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}]\langle B^?\rangle \\ \hat{\mathcal{C}}^-\langle B^?\rangle = \bullet + \oplus[\hat{\mathcal{C}}^+, \hat{\mathcal{C}}]\langle B^?\rangle + \sum_{\pi\in\mathcal{S}_\mathcal{C}} \pi[\hat{\mathcal{C}}, \hat{\mathcal{C}}, \dots, \hat{\mathcal{C}}]\langle B^?\rangle \end{cases}$$

- When $\mathcal{C}$ is substitution-closed,

$\mathcal{S}_\mathcal{C}$ immediately gives an unambiguous tree grammar for $\mathcal{C}$.

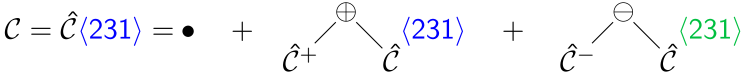
- When $\mathcal{C} = \mathsf{Av}(B)$ is not substitution-closed,
  - It still holds for $\hat{\mathcal{C}}$, with $\mathcal{S}_{\hat{\mathcal{C}}} = \mathcal{S}_\mathcal{C}$.
  - $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star\rangle = \hat{\mathcal{C}} \cap \mathsf{Av}(B^\star)$, where $B^\star =$ 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: $\mathcal{C} = \mathsf{Av}(231)$.
We have $\mathcal{S}_{\mathcal{C}} = \mathcal{S}_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231\rangle$.

$\mathcal{C} = \hat{\mathcal{C}}\langle 231\rangle = \bullet \quad +$  $\langle 231\rangle \quad +$  $\langle 231\rangle$

Pushing restrictions in the subtrees

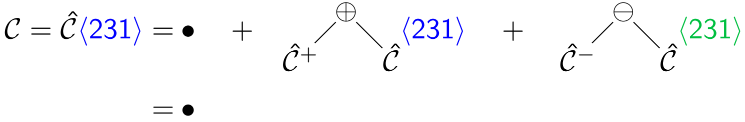
Example: $\mathcal{C} = \text{Av}(231)$.
We have $\mathcal{S}_{\mathcal{C}} = \mathcal{S}_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle$.

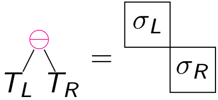


$$= \bullet$$

Pushing restrictions in the subtrees

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

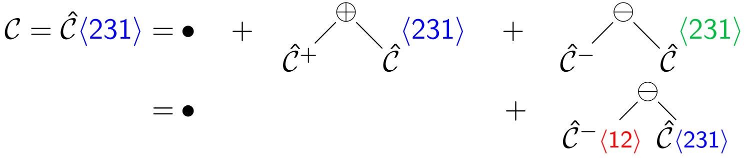


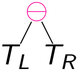
Claim: 

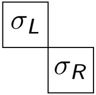
$\in \text{Av}(231) \Leftrightarrow \sigma_L \in \text{Av}(12)$ and $\sigma_R \in \text{Av}(231)$

Pushing restrictions in the subtrees

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



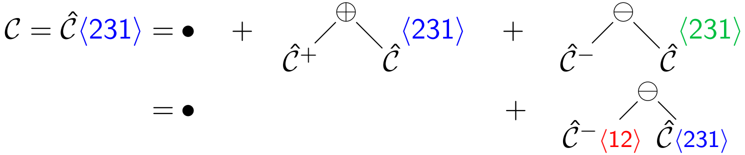
Claim: 

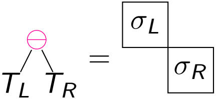
$=$ 

$\in \mathrm{Av}(231) \Leftrightarrow \sigma_L \in \mathrm{Av}(12)$ and $\sigma_R \in \mathrm{Av}(231)$

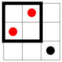
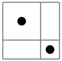
Pushing restrictions in the subtrees

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



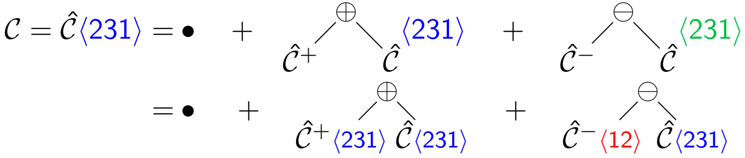
Claim: 

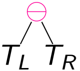
$\in \mathrm{Av}(231) \Leftrightarrow \sigma_L \in \mathrm{Av}(12)$ and $\sigma_R \in \mathrm{Av}(231)$

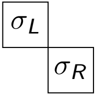
Because of an embedding of 231 into $21 = \ominus$:  $= 231 \hookrightarrow 21 =$ 

Pushing restrictions in the subtrees

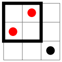
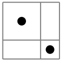
Example: $\mathcal{C} = \text{Av}(231)$.
We have $\mathcal{S}_{\mathcal{C}} = \mathcal{S}_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231 \rangle$.



Claim: 

= 

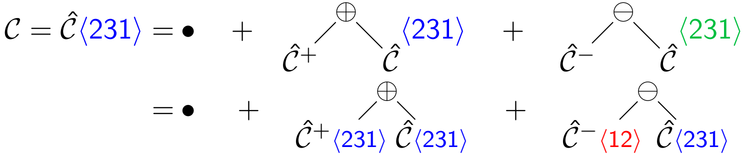
$\in \text{Av}(231) \Leftrightarrow \sigma_L \in \text{Av}(12)$ and $\sigma_R \in \text{Av}(231)$

Because of an embedding of 231 into $21 = \ominus$:  $= 231 \hookrightarrow 21 =$ 

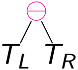
Pushing restrictions in the subtrees

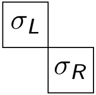
Example: $\mathcal{C} = \mathrm{Av}(231)$.

We have $\mathcal{S}_{\mathcal{C}} = \mathcal{S}_{\hat{\mathcal{C}}} = \emptyset$, and $\mathcal{C} = \hat{\mathcal{C}}\langle 231\rangle$.

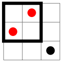
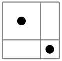


$\hat{\mathcal{C}}^{-}\langle 12\rangle = \dots$

Claim: 

$=$ 

$\in \mathrm{Av}(231) \Leftrightarrow \sigma_L \in \mathrm{Av}(12)$ and $\sigma_R \in \mathrm{Av}(231)$

Because of an embedding of 231 into $21 = \ominus$:  $= 231 \hookrightarrow 21 =$ 

Need of a new equation for $\hat{\mathcal{C}}^{-}\langle 12\rangle$ … And keep going

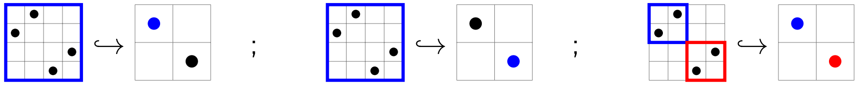
Why the grammar may be ambiguous

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

Why the grammar may be ambiguous

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

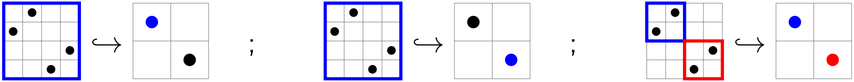
Example with $\beta = 3412$ and $\pi = 21$. Three embeddings of β into π:



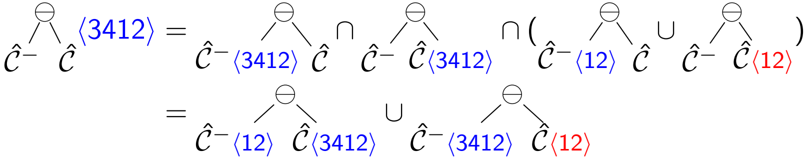
Why the grammar may be ambiguous

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

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



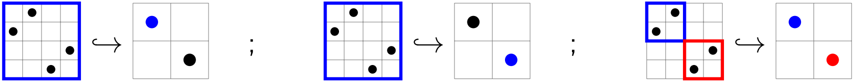
Hence:



Why the grammar may be ambiguous

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

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



Hence: $\ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}]\langle 3412\rangle = \ominus[\hat{\mathcal{C}}^-\langle 3412\rangle, \hat{\mathcal{C}}] \cap \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}\langle 3412\rangle] \cap \left(\ominus[\hat{\mathcal{C}}^-\langle 12\rangle, \hat{\mathcal{C}}] \cup \ominus[\hat{\mathcal{C}}^-, \hat{\mathcal{C}}\langle 12\rangle] \right)$

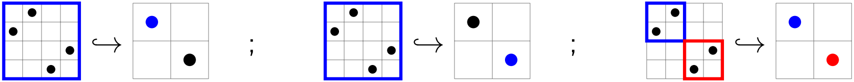
$= \ominus[\hat{\mathcal{C}}^-\langle 12\rangle, \hat{\mathcal{C}}\langle 3412\rangle] \cup \ominus[\hat{\mathcal{C}}^-\langle 3412\rangle, \hat{\mathcal{C}}\langle 12\rangle]$

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 $\beta \in B^\star$ into $\pi \in \mathcal{S}_\mathcal{C} \cup \{12, 21\}$.

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



Hence:

$$\ominus(\hat{\mathcal{C}}^-, \hat{\mathcal{C}})\langle 3412\rangle = \ominus(\hat{\mathcal{C}}^-\langle 3412\rangle, \hat{\mathcal{C}}) \cap \ominus(\hat{\mathcal{C}}^-, \hat{\mathcal{C}}\langle 3412\rangle) \cap \big(\ominus(\hat{\mathcal{C}}^-\langle 12\rangle, \hat{\mathcal{C}}) \cup \ominus(\hat{\mathcal{C}}^-, \hat{\mathcal{C}}\langle 12\rangle)\big)$$

$$= \ominus(\hat{\mathcal{C}}^-\langle 12\rangle, \hat{\mathcal{C}}\langle 3412\rangle) \cup \ominus(\hat{\mathcal{C}}^-\langle 3412\rangle, \hat{\mathcal{C}}\langle 12\rangle)$$

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

Observation: The new excluded patterns are some $\alpha \preccurlyeq \beta \in B^\star$

Need of introducing pattern *containment* constraints

Example: Disambiguation of $\ominus\left(\hat{\mathcal{C}}^{-}\langle 12 \rangle, \hat{\mathcal{C}}\langle 3412 \rangle\right) \cup \ominus\left(\hat{\mathcal{C}}^{-}\langle 3412 \rangle, \hat{\mathcal{C}}\langle 12 \rangle\right)$.

Method:

- $A \cup B = A \cap B \uplus \overline{A} \cap B \uplus A \cap \overline{B}$

Need of introducing pattern *containment* constraints

Example: Disambiguation of $\ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle,\ \hat{\mathcal{C}}\langle 3412\rangle] \cup \ominus[\hat{\mathcal{C}}^{-}\langle 3412\rangle,\ \hat{\mathcal{C}}\langle 12\rangle]$.

Method:

- $A \cup B = A \cap B \ \uplus\ \overline{A} \cap B \ \uplus\ A \cap \overline{B}$
- By complementation, excluded patterns become mandatory patterns: $\mathcal{C}_\gamma$ for $\gamma \preccurlyeq \beta \in B^\star$

$$\ominus[\hat{\mathcal{C}}^{-},\ \hat{\mathcal{C}}]\langle 3412\rangle = \ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle,\ \hat{\mathcal{C}}\langle 12\rangle] \uplus \ominus[\hat{\mathcal{C}}^{-}_{12}\langle 3412\rangle,\ \hat{\mathcal{C}}\langle 12\rangle] \uplus \ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle,\ \hat{\mathcal{C}}_{12}\langle 3412\rangle].$$

Notice that the terms $\ominus[\hat{\mathcal{C}}^{-}\langle 3412\rangle,\ \hat{\mathcal{C}}_{3412}\langle 12\rangle]$ and $\ominus[\hat{\mathcal{C}}^{-}_{3412}\langle 12\rangle,\ \hat{\mathcal{C}}\langle 3412\rangle]$ are empty, and have been deleted.

Need of introducing pattern *containment* constraints

Example: Disambiguation of $\ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle, \hat{\mathcal{C}}\langle 3412\rangle] \cup \ominus[\hat{\mathcal{C}}^{-}\langle 3412\rangle, \hat{\mathcal{C}}\langle 12\rangle]$.

Method:

- $A \cup B = A \cap B \uplus \overline{A} \cap B \uplus A \cap \overline{B}$
- By complementation, excluded patterns become mandatory patterns: $\mathcal{C}_{\gamma}$ for $\gamma \preccurlyeq \beta \in B^{\star}$

$$\ominus[\hat{\mathcal{C}}^{-}, \hat{\mathcal{C}}]\langle 3412\rangle = \ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle, \hat{\mathcal{C}}\langle 12\rangle] \uplus \ominus[\hat{\mathcal{C}}^{-}_{12}\langle 3412\rangle, \hat{\mathcal{C}}\langle 12\rangle] \uplus \ominus[\hat{\mathcal{C}}^{-}\langle 12\rangle, \hat{\mathcal{C}}_{12}\langle 3412\rangle].$$

$\Rightarrow$ Need to propagate avoidance and containment constraints:

$$\hat{\mathcal{C}}^{\varepsilon}_{\gamma_1,\dots,\gamma_p}\langle \alpha_1, \dots, \alpha_k\rangle \text{ with } \varepsilon \in \{\ \ , +, -\}$$

Observation: γ_i and α_j are all patterns of some $\beta \in B^{\star}$.

A first specification for $\mathcal{C}$

Find a specification for all

$$\hat{\mathcal{C}}^{\varepsilon}_{\gamma_1,\dots,\gamma_p}\langle \alpha_1, \dots, \alpha_k \rangle$$

with $\{\gamma_1, \dots, \gamma_p\} \subseteq \widetilde{B^\star}$ and $\{\alpha_1, \dots, \alpha_k\} \subseteq \widetilde{B^\star}$,
where $\widetilde{B^\star} = \{\alpha \preccurlyeq \beta \mid \beta \in B^\star\}$ = set of patterns of some $\beta \in B^\star$.

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

A first specification for $\mathcal{C}$

Find a specification for all

$$\hat{\mathcal{C}}^{\varepsilon}_{\gamma_1,\ldots,\gamma_p}\langle \alpha_1,\ldots,\alpha_k\rangle$$

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

How to:
For $\alpha \in \widetilde{B^\star}$ and $\sigma = \pi[\alpha^{(1)},\alpha^{(2)},\ldots,\alpha^{(k)}]$,
considering embeddings of α in π,
we can decide which patterns α occur in σ
from the knowledge of which patterns of $\widetilde{B^\star}$ 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 $\mathcal{C} = \hat{\mathcal{C}}\langle B^\star \rangle$, $\mathcal{C}^+$ and $\mathcal{C}^-$, and propagate the pattern avoidance constraints in the subtrees.
- Disambiguate the equations, introducing pattern containment constraints.
- For each term $\hat{\mathcal{C}}^{\varepsilon}_{\gamma_1,\ldots,\gamma_p}\langle \alpha_1, \ldots, \alpha_k \rangle$ that appears on the RHS, repeat this process, recursively.

Properties:

- This algorithm terminates and produces a specification for $\mathcal{C}$.

Computing only the necessary restrictions

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

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

Properties:

- This algorithm terminates and produces a specification for $\mathcal{C}$.

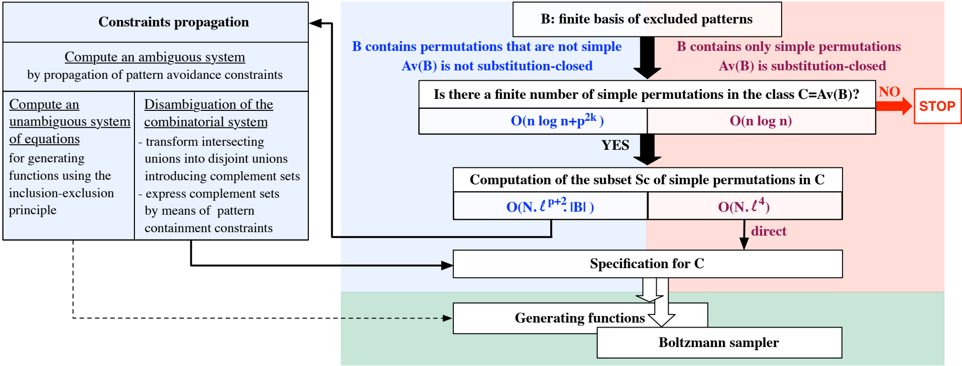
Questions:

- What is the complexity?
- What is the size of the specification produced?

$\hookrightarrow$ It can be exponential in $\sum_{\beta \in B} |\beta|$. But how big can it be?

Summary: From the basis to the specification

Algorithmic chain from B finite to a specification for $\mathcal{C} = \text{Av}(B)$.



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

Remark: It succeeds only when $\mathcal{C}$ contains finitely many simples (and this condition is tested algorithmically).

Byproducts of specifications and perspectives

A specification for $\mathcal{C}$ gives access to...

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

[Flajolet & Sedgewick 09]

$\hookrightarrow$ Deduce the first terms of $C(z)$, as many as desired.

$\hookrightarrow$ What can be said about the growth rate of $\mathcal{C}$?

A specification for $\mathcal{C}$ gives access to...

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

  [Flajolet & Sedgewick 09]

↪ Deduce the first terms of $C(z)$, as many as desired.

↪ What can be said about the growth rate of $\mathcal{C}$?

- Random samplers of permutations in $\mathcal{C}$:

▶ by the recursive method [Flajolet, Zimmerman & Van Cutsem 94]

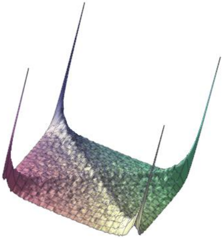
▶ by the Boltzmann method [Duchon, Flajolet, Louchard & Schaeffer 04]

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

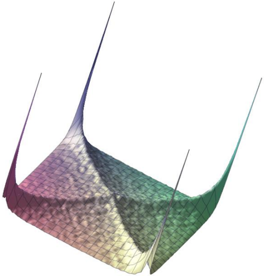
↪ Can we describe the "average shape" or average properties of random permutations in permutation classes?
For some given classes, or for families of classes?

Random permutations in permutation classes

• $\mathcal{C}_1 = \text{Av}(2413, 3142) =$ separables.
Substitution-closed with no simples.
10000 permutations of size 100 in $\mathcal{C}_1$.



• Substitution-closed class $\mathcal{C}_2$,
with simples 2413, 3142 and 24153.
10000 permutations of size 500 in $\mathcal{C}_2$.



• $\mathcal{C}_3 = \text{Av}(2413, 1243, 2341, 531642, 41352)$.
Not substitution-closed.
Almost 30000 permutations of size 500 in $\mathcal{C}_3$.