Searching for combinatorial covers using linear programming

Bjarni Jens Kristinsson

Reykjavik University

December 16, 2019

Background

- Builds upon „*Automatic discovery of structural rules of permutation classes*“ (2019) by Christian Bean, Bjarki Guðmundsson and Henning Ulfarsson
- Struct is only written for permutation classes

Our contribution

- CombCov published as a module for Python
- General framework for all kinds of combinatorial objects
- Automated old results of permutation classes avoiding mesh patterns

A combinatorial object: *words*

A combinatorial object: *words*

Definition

A *word of length n* is a sequence of *characters* $c_1 \cdots c_n$ over an *alphabet* Σ. If $n = 0$ then the word is the *empty word* and we denote it with ϵ.

A combinatorial object: *words*

Definition

A *word of length* n is a sequence of *characters* $c_1 \cdots c_n$ over an *alphabet* Σ. If $n = 0$ then the word is the *empty word* and we denote it with ϵ.

Example

abba is a word of length 4 over the alphabet $\Sigma = \{a, b\}$.

A combinatorial object: *words* (continued)

A combinatorial object: *words* (continued)

Definition

We say that a word $u = u_1 \cdots u_n$ *contains* another word $v = v_1 \cdots v_k$ as a *subword* if there exists an i such that $u_{i+1} \cdots u_{i+k} = v_1 \cdots v_k$. If u does not contain v, we say that u *avoids* v and define $\mathrm{Av}_n(v)$ as the set of all words of length n avoiding v and write $\mathrm{Av}(v) = \bigcup_{n=0}^{\infty} \mathrm{Av}_n(v)$.

A combinatorial object: *words* (continued)

Definition

We say that a word $u = u_1 \cdots u_n$ *contains* another word $v = v_1 \cdots v_k$ as a *subword* if there exists an i such that $u_{i+1} \cdots u_{i+k} = v_1 \cdots v_k$. If u does not contain v, we say that u *avoids* v and define $\mathrm{Av}_n(v)$ as the set of all words of length n avoiding v and write $\mathrm{Av}(v) = \bigcup_{n=0}^{\infty} \mathrm{Av}_n(v)$.

Example

The word *abba* contains the subword *bb* but avoids *aa*.

Avoidance sets of words

Avoidance sets of words

Definition

Let V be a set of words over the alphabet Σ. We define $\mathrm{Av}(V) = \bigcap_{v \in V} \mathrm{Av}(v)$ and call this an *avoidance set* of words.

Avoidance sets of words

Definition

Let V be a set of words over the alphabet Σ. We define $\mathrm{Av}(V) = \bigcap_{v \in V} \mathrm{Av}(v)$ and call this an *avoidance set* of words.

Example

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$ is an avoidance set of words.

Avoidance sets of words

Definition

Let V be a set of words over the alphabet Σ. We define $\mathrm{Av}(V) = \bigcap_{v \in V} \mathrm{Av}(v)$ and call this an *avoidance set* of words.

Example

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$ is an avoidance set of words.

Research questions

Avoidance sets of words

Definition

Let V be a set of words over the alphabet Σ. We define $\mathrm{Av}(V) = \bigcap_{v \in V} \mathrm{Av}(v)$ and call this an *avoidance set* of words.

Example

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$ is an avoidance set of words.

Research questions

- *Can we find a formula for the number of elements of specific length in the avoidance sets?*

Avoidance sets of words

Definition

Let V be a set of words over the alphabet Σ. We define $\mathrm{Av}(V) = \bigcap_{v \in V} \mathrm{Av}(v)$ and call this an *avoidance set* of words.

Example

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$ is an avoidance set of words.

Research questions

- *Can we find a formula for the number of elements of specific length in the avoidance sets?*
- *Can we describe the avoidance sets such as* $\mathrm{Av}(aa)$ *differently, i.e., in "simpler terms"?*

The core idea behind CombCov

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$

  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$

  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$

  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \varnothing$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) \colon |w| \leqslant \ell\}$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) \colon |w| \leqslant \ell\}$
  - $R_i = \{w \in S_i \colon |w| \leqslant \ell\}$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) \colon |w| \leqslant \ell\}$
  - $R_i = \{w \in S_i \colon |w| \leqslant \ell\}$
- Now we solve the finite problem

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) : |w| \leqslant \ell\}$
  - $R_i = \{w \in S_i : |w| \leqslant \ell\}$
- Now we solve the finite problem
  (1) $\bigcup_{i \in I} R_i = R$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) : |w| \leqslant \ell\}$
  - $R_i = \{w \in S_i : |w| \leqslant \ell\}$
- Now we solve the finite problem
  (1) $\bigcup_{i \in I} R_i = R$
  (2) $R_i \cap R_j = \emptyset$ if $i \neq j$

The core idea behind CombCov

- Goal: Find k disjoint subsets S_i that *cover* $\mathrm{Av}(S)$
  (1) $\bigcup_{i \in I} S_i = \mathrm{Av}(S)$
  (2) $S_i \cap S_j = \emptyset$ if $i \neq j$
- We call $\mathrm{Av}(S)$ *the root* and S_i *rules*
- Problem: Computers are unable to compute with infinitely many objects
- Solution: Create *finite* representations and compute using them instead of the infinite counterparts
  - $R = \{w \in \mathrm{Av}(S) \colon |w| \leqslant \ell\}$
  - $R_i = \{w \in S_i \colon |w| \leqslant \ell\}$
- Now we solve the finite problem
  (1) $\bigcup_{i \in I} R_i = R$
  (2) $R_i \cap R_j = \emptyset$ if $i \neq j$
- In the next few slides we go step by step through how the algorithm finds a cover for $\mathrm{Av}(aa)$

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- How do we pick the rules S_i?

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- How do we pick the rules S_i?
- Our solution: Generate rules of the form $u\mathrm{Av}(S')$ where

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- How do we pick the rules S_i?
- Our solution: Generate rules of the form $u\mathrm{Av}(S')$ where
  - u is a word in $\mathrm{Av}(S)$ of length $\leqslant \max\{1, |w| : w \in S\}$ and

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- How do we pick the rules S_i?
- Our solution: Generate rules of the form $u\mathrm{Av}(S')$ where
  - u is a word in $\mathrm{Av}(S)$ of length $\leqslant \max\{1, |w| \colon w \in S\}$ and
  - S' is either the whole alphabet Σ or a set of words each of which is a subword of a word in S

Rules

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- How do we pick the rules S_i?
- Our solution: Generate rules of the form $u\mathrm{Av}(S')$ where
  - u is a word in $\mathrm{Av}(S)$ of length $\leqslant \max\{1, |w| : w \in S\}$ and
  - S' is either the whole alphabet Σ or a set of words each of which is a subword of a word in S
- Next step: Verify *valid* rules and throw away *invalid* rules

Bitstrings

Remember

$\text{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R
    - $B' = 111111$ denotes the whole set R

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R
  - $B' = 111111$ denotes the whole set R
  - $B'' = 011001$ denotes the subset $\{a, b, bb\}$

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R
  - $B' = 111111$ denotes the whole set R
  - $B'' = 011001$ denotes the subset $\{a, b, bb\}$
  - $B''' = 100000$ denotes the subset $\{\epsilon\}$

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R
  - $B' = 111111$ denotes the whole set R
  - $B'' = 011001$ denotes the subset $\{a, b, bb\}$
  - $B''' = 100000$ denotes the subset $\{\epsilon\}$
- The rule $a\mathrm{Av}(a)$ generates $R' = \{a, ab\} \subseteq R$ with corresponding bitstring 010100 so it is valid

Bitstrings

Remember

$\mathrm{Av}(aa) = \{\epsilon, a, b, ab, ba, bb, aba, abb, bab, bba, bbb, \ldots\}$

- Define the *precision* $\ell = 2$
- Then $R = \{\epsilon, a, b, ab, ba, bb\}$
- We use *bitstrings* to denote subsets of R
  - $B' = 111111$ denotes the whole set R
  - $B'' = 011001$ denotes the subset $\{a, b, bb\}$
  - $B''' = 100000$ denotes the subset $\{\epsilon\}$
- The rule $a\mathrm{Av}(a)$ generates $R' = \{a, ab\} \subseteq R$ with corresponding bitstring 010100 so it is valid
- The rule $a\mathrm{Av}(b)$ generates $R'' = \{a, aa\} \nsubseteq R$ so the rule is invalid

Linear programming

Remember

$R = \{\epsilon, a, b, ab, ba, bb\}$

Linear programming

Remember

$R = \{\epsilon, a, b, ab, ba, bb\}$

- In total 16 rules — 15 valid — 9 distinct bitstrings

Linear programming

Remember

$R = \{\epsilon, a, b, ab, ba, bb\}$

- In total 16 rules — 15 valid — 9 distinct bitstrings
- Using *Gurobi* (or *COIN CLP/CBC LP*) to solve

$$
\begin{array}{lrl}
\text{Min} & z = x_1 + \cdots + x_9 & \\
\text{s.t.} & x_6 + x_8 + x_9 & = 1 \\
 & x_1 & = 1 \\
 & x_2 + x_7 & = 1 \\
 & x_3 + x_8 + x_9 & = 1 \\
 & x_4 + x_7 & = 1 \\
 & x_5 + x_9 & = 1 \\
\text{with} & x_i \in \{0, 1\} \text{ for } i = 1, \ldots, 9. &
\end{array}
$$

Cover for $\mathrm{Av}(aa)$

- One solution for the system of equations is $x_1 = x_7 = x_9 = 1$ which represents the cover

$$\epsilon\mathrm{Av}(a, b) \cup a\mathrm{Av}(a) \cup b\mathrm{Av}(aa).$$

Cover for Av(aa)

- One solution for the system of equations is $x_1 = x_7 = x_9 = 1$ which represents the cover

$$\epsilon \mathrm{Av}(a, b) \cup a\mathrm{Av}(a) \cup b\mathrm{Av}(aa).$$

- However, this is an *incorrect* solution!

Cover for Av(aa)

- One solution for the system of equations is $x_1 = x_7 = x_9 = 1$ which represents the cover

$$\epsilon \mathrm{Av}(a, b) \cup a\mathrm{Av}(a) \cup b\mathrm{Av}(aa).$$

- However, this is an *incorrect* solution!
- $abba \in \mathrm{Av}(aa)$ but none of the rules generates this word!

Cover for $\mathrm{Av}(aa)$

- One solution for the system of equations is $x_1 = x_7 = x_9 = 1$ which represents the cover

$$\epsilon\mathrm{Av}(a, b) \cup a\mathrm{Av}(a) \cup b\mathrm{Av}(aa).$$

- However, this is an *incorrect* solution!
- $abba \in \mathrm{Av}(aa)$ but none of the rules generates this word!
- By increasing the precision ℓ to $\geqslant 3$ we get the correct solution:

$$\mathrm{Av}(aa) = \epsilon\mathrm{Av}(a, b) \cup a\mathrm{Av}(a, b) \cup b\mathrm{Av}(aa) \cup ab\mathrm{Av}(aa)$$

Cover for Av(aa)

- One solution for the system of equations is $x_1 = x_7 = x_9 = 1$ which represents the cover

$$\epsilon \mathrm{Av}(a, b) \cup a\mathrm{Av}(a) \cup b\mathrm{Av}(aa).$$

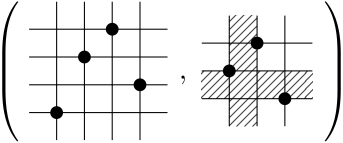
- However, this is an *incorrect* solution!
- $abba \in \mathrm{Av}(aa)$ but none of the rules generates this word!
- By increasing the precision ℓ to $\geqslant 3$ we get the correct solution:

$$\mathrm{Av}(aa) = \epsilon \mathrm{Av}(a, b) \cup a\mathrm{Av}(a, b) \cup b\mathrm{Av}(aa) \cup ab\mathrm{Av}(aa)$$

- It is easy to prove that the *enumeration* of this avoidance set of words is the Fibonacci sequence shifted by two, i.e., $|\mathrm{Av}_n(aa)| = F_{n+2}$

Our area of interest

Av



Permutations

Permutations

Definition

A *permutation of length n* is a bijection from the set of the first n integers, $[\![n]\!] = \{1, \ldots, n\}$ to itself. The set of all permutations of length n is denoted with $\mathfrak{S}_n$ and $\mathfrak{S} = \bigcup_{n=0}^{\infty} \mathfrak{S}_n$ is the set of all permutations. The only permutation of length 0 is called the *empty permutation* and it is denoted by ϵ.

Permutations

Definition

A *permutation of length n* is a bijection from the set of the first n integers, $[\![n]\!] = \{1, \ldots, n\}$ to itself. The set of all permutations of length n is denoted with $\mathfrak{S}_n$ and $\mathfrak{S} = \bigcup_{n=0}^{\infty} \mathfrak{S}_n$ is the set of all permutations. The only permutation of length 0 is called the *empty permutation* and it is denoted by ϵ.

Example

An example of a permutation of length 5 is $\pi = 35142$ with $\pi(1) = 3$, $\pi(2) = 5$, $\pi(3) = 1$, $\pi(4) = 4$ and $\pi(5) = 2$.

Subpermutations

Subpermutations

Definition

A permutation $\pi \in \mathfrak{S}_n$ *contains* a permutation $\sigma \in \mathfrak{S}_k$ as a *subpermutation* if there exists k indices $1 \leqslant i_1 < \cdots < i_k \leqslant n$ such that $\pi(i_1) \cdots \pi(i_k)$ has the same relative ordering as σ, meaning that $\pi(i_j) < \pi(i_l)$ if and only if $\sigma(j) < \sigma(l)$. We call $\pi(i_1) \cdots \pi(i_k)$ an *occurrence* of σ in π.
If π does not contain σ, we say that π *avoids* σ. The set of all permutations that avoid π is denoted with $\mathrm{Av}(\pi)$.

Subpermutations

Definition

A permutation $\pi \in \mathfrak{S}_n$ *contains* a permutation $\sigma \in \mathfrak{S}_k$ as a *subpermutation* if there exists k indices $1 \leqslant i_1 < \cdots < i_k \leqslant n$ such that $\pi(i_1) \cdots \pi(i_k)$ has the same relative ordering as σ, meaning that $\pi(i_j) < \pi(i_l)$ if and only if $\sigma(j) < \sigma(l)$. We call $\pi(i_1) \cdots \pi(i_k)$ an *occurrence* of σ in π.
If π does not contain σ, we say that π *avoids* σ. The set of all permutations that avoid π is denoted with $\mathrm{Av}(\pi)$.

Example

The permutation $\pi = 35142$ contains the permutation $\sigma = 213$ because $\pi(1)\pi(3)\pi(4) = 314$ is order relative to 213. π avoids 123.

Grid representation of permutations

Grid representation of permutations

Definition

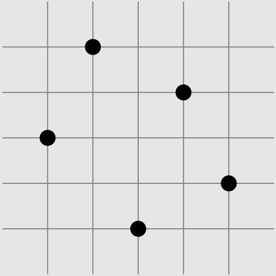
The visual *grid representation* of π, denoted with $\mathrm{Gr}(\pi)$, is the plot of $\{(i, \pi(i)) \mid i \in [\![n]\!]\}$ in a Cartesian coordinate system.

Grid representation of permutations

Definition

The visual *grid representation* of π, denoted with $\mathrm{Gr}(\pi)$, is the plot of $\{(i, \pi(i)) \mid i \in [\![n]\!]\}$ in a Cartesian coordinate system.

Example



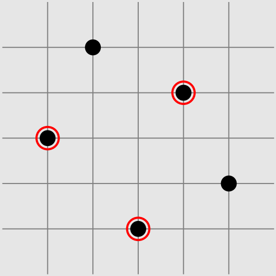
On the left we have plotted $\mathrm{Gr}(\pi)$ with $\pi = 35142$ from previous examples.

Grid representation of permutations

Definition

The visual *grid representation* of π, denoted with $\mathrm{Gr}(\pi)$, is the plot of $\{(i, \pi(i)) \mid i \in [\![n]\!]\}$ in a Cartesian coordinate system.

Example



On the left we have plotted $\mathrm{Gr}(\pi)$ with $\pi = 35142$ from previous examples. Highlighted with red circles is an occurrence of $\sigma = 213$ in π.

Mesh patterns

Mesh patterns

Definition

A *mesh pattern* is a a pair

$$p = (\sigma, R) \text{ with } \sigma \in \mathfrak{S}_k \text{ and } R \subseteq [\![0, k]\!] \times [\![0, k]\!]$$

where $[\![0, k]\!] = \{0, 1, \ldots, k\}$ and R is a set of Cartesian coordinates $\ulcorner i, j \lrcorner$ denoting the lower left corners of the squares in the grid representation of σ which are *shaded*.

Mesh patterns

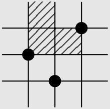
Definition

A *mesh pattern* is a a pair

$$p = (\sigma, R) \text{ with } \sigma \in \mathfrak{S}_k \text{ and } R \subseteq [\![0, k]\!] \times [\![0, k]\!]$$

where $[\![0, k]\!] = \{0, 1, \ldots, k\}$ and R is a set of Cartesian coordinates $\ulcorner i, j \lrcorner$ denoting the lower left corners of the squares in the grid representation of σ which are *shaded*.

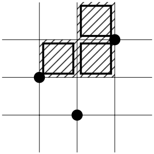
Example

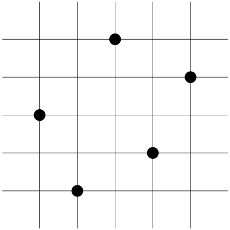


The mesh pattern $p = (\sigma, R)$ where $\sigma = 213$ and $R = \{\ulcorner 1, 2 \lrcorner, \ulcorner 1, 3 \lrcorner, \ulcorner 2, 2 \lrcorner\}$ is shown on the left.

Permutations containing mesh patterns

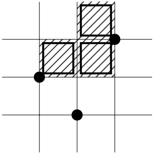
Permutations containing mesh patterns

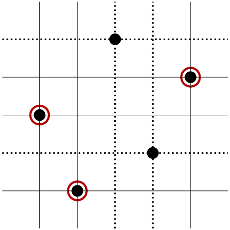




- The mesh pattern $p = (213, \{\boxed{1, 2}, \boxed{2, 2}, \boxed{2, 3}\})$ (left) and the permutation $\pi = 31524$ (middle).

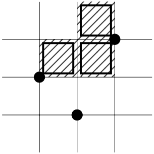
Permutations containing mesh patterns

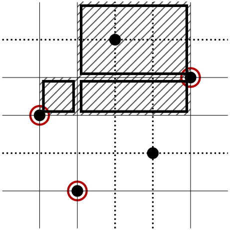




- The mesh pattern $p = (213, \{\boxed{1,2}, \boxed{2,2}, \boxed{2,3}\})$ (left) and the permutation $\pi = 31524$ (middle).
- Even though 314 (highlighted with red circles) is an occurrence of 213 in π

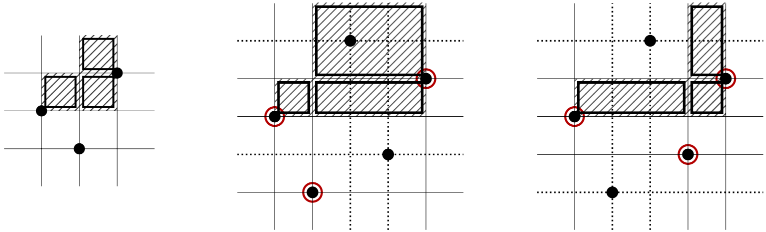
Permutations containing mesh patterns





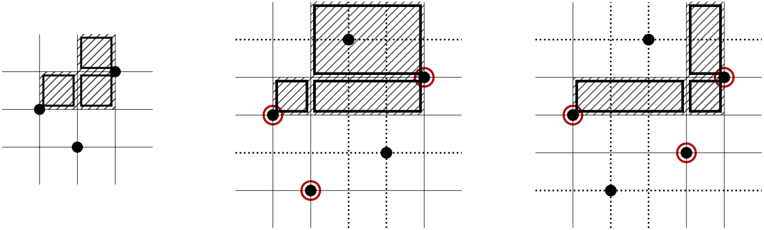
- The mesh pattern $p = (213, \{\boxed{1,2}, \boxed{2,2}, \boxed{2,3}\})$ (left) and the permutation $\pi = 31524$ (middle).
- Even though 314 (highlighted with red circles) is an occurrence of 213 in π this is not an occurrence of p in π.

Permutations containing mesh patterns



- The mesh pattern $p = (213, \{\ulcorner 1,2 \lrcorner, \ulcorner 2,2 \lrcorner, \ulcorner 2,3 \lrcorner\})$ (left) and the permutation $\pi = 31524$ (middle).
- Even though 314 (highlighted with red circles) is an occurrence of 213 in π this is not an occurrence of p in π.
- However, p is contained in π because with the occurrence 324 the shaded areas do not overlap with any point in π.

Permutations containing mesh patterns



- The mesh pattern $p = (213, \{\boxed{1, 2}, \boxed{2, 2}, \boxed{2, 3}\})$ (left) and the permutation $\pi = 31524$ (middle).
- Even though 314 (highlighted with red circles) is an occurrence of 213 in π this is not an occurrence of p in π.
- However, p is contained in π because with the occurrence 324 the shaded areas do not overlap with any point in π.
- The set of all permutations that *avoid* the mesh pattern p is denoted with $\mathrm{Av}(p)$.

Generating functions

Generating functions

Definition

The *generating function (GF)* of the avoidance set $\mathrm{Av}(S)$ is the sum

$$\sum_{n=0}^{+\infty} F_n x^n$$

where $F_n = |\mathrm{Av}_n(S)|$ for all n. The *exponential generating function (EGF)* of $\mathrm{Av}(S)$ is the sum $\sum_{n=0}^{+\infty} \frac{F_n}{n!} x^n$. We say that the avoidance set is *enumerated* by the GF (or EGF) and that the number sequence $(F_n)_{n \geqslant 0}$ is the *enumeration* of $\mathrm{Av}(S)$.

Generating functions

Definition

The *generating function (GF)* of the avoidance set $\mathrm{Av}(S)$ is the sum

$$\sum_{n=0}^{+\infty} F_n x^n$$

where $F_n = |\mathrm{Av}_n(S)|$ for all n. The *exponential generating function (EGF)* of $\mathrm{Av}(S)$ is the sum $\sum_{n=0}^{+\infty} \frac{F_n}{n!} x^n$. We say that the avoidance set is *enumerated* by the GF (or EGF) and that the number sequence $(F_n)_{n \geqslant 0}$ is the *enumeration* of $\mathrm{Av}(S)$.

Example

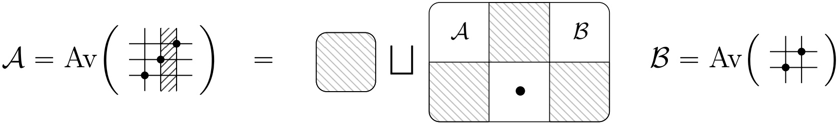
The GF of $\mathrm{Av}(21) = \{\epsilon, 1, 12, \ldots\}$ is $\sum_{n=0}^{+\infty} x^n = \frac{1}{1-x}$ so the enumeration is $(1, 1, 1, \ldots)$. The EGF of $\mathrm{Av}(21)$ is $\sum_{n=0}^{+\infty} \frac{1}{n!} x^n = e^x$.

Interesting results

Now we will look at some interesting results obtained with CombCov.

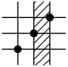
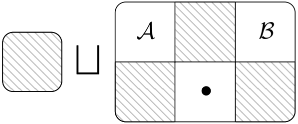
Generalized Pattern Avoidance (2001)

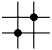
One of the papers that we tried replicating results from was the above one by Anders Claesson where he studies *generalized permutation patterns* that are specific type of mesh patterns. Out of the 6 results in the paper we could replicate 5 of them. Below is one of them.



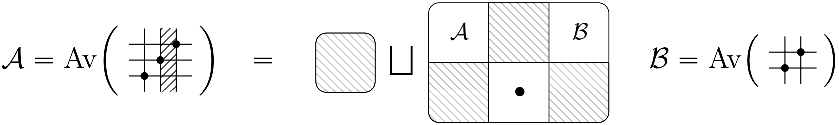
After verifying that the cover is indeed correct it is interesting to derive the *exponential generating function (EGF)* $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$ of the avoidance set and show that $a_n = B_n$, proving that the sequence is enumerated by the Bell numbers.

Deriving the EGF

$\mathcal{A} = \mathrm{Av}\Big($  $\Big)$ $=$ 

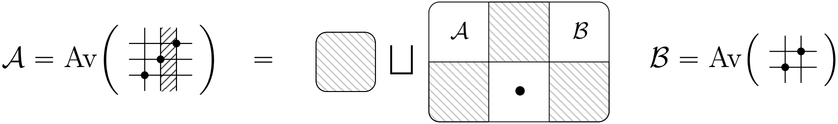
$\mathcal{B} = \mathrm{Av}\Big($  $\Big)$

Deriving the EGF



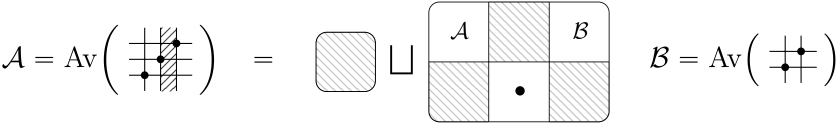
- $\mathcal{A}$ has EGF $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$

Deriving the EGF



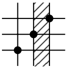
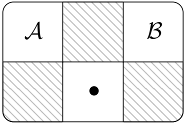
- $\mathcal{A}$ has EGF $F = \sum_{n\geqslant 0} \frac{a_n}{n!} x^n$
- $\mathcal{B}$ has EGF $\sum_{n\geqslant 0} \frac{1}{n!} x^n = e^x$

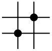
Deriving the EGF



- $\mathcal{A}$ has EGF $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$
- $\mathcal{B}$ has EGF $\sum_{n \geqslant 0} \frac{1}{n!} x^n = e^x$
- This gives us the recurrence relation $F = 1 + \int F e^x \, dx$

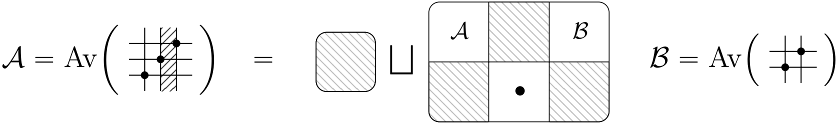
Deriving the EGF

$\mathcal{A} = \mathrm{Av}\Big($  $\Big) =$  $\bigsqcup$ 

$\mathcal{B} = \mathrm{Av}\Big($  $\Big)$

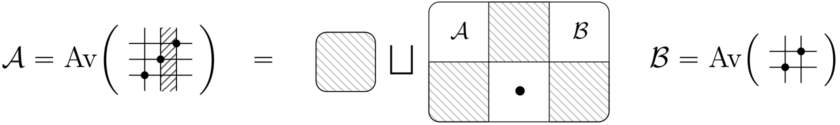
- $\mathcal{A}$ has EGF $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$
- $\mathcal{B}$ has EGF $\sum_{n \geqslant 0} \frac{1}{n!} x^n = e^x$
- This gives us the recurrence relation $F = 1 + \int F e^x dx$
- We solve it and get $F = Ae^{e^x}$

Deriving the EGF



- $\mathcal{A}$ has EGF $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$
- $\mathcal{B}$ has EGF $\sum_{n \geqslant 0} \frac{1}{n!} x^n = e^x$
- This gives us the recurrence relation $F = 1 + \int F e^x dx$
- We solve it and get $F = A e^{e^x}$
- Knowing that there is only one permutation of length zero in $\mathcal{A}$ we put $a_0 = 1$ into the equation at $x = 0$ and get $A = e^{-1}$

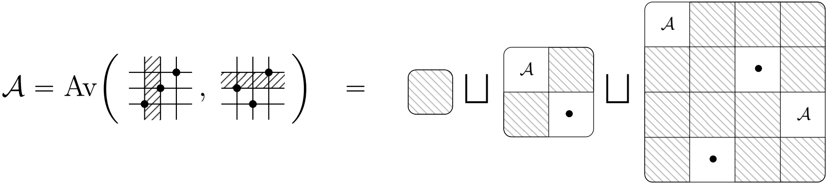
Deriving the EGF



- $\mathcal{A}$ has EGF $F = \sum_{n \geqslant 0} \frac{a_n}{n!} x^n$
- $\mathcal{B}$ has EGF $\sum_{n \geqslant 0} \frac{1}{n!} x^n = e^x$
- This gives us the recurrence relation $F = 1 + \int F e^x dx$
- We solve it and get $F = A e^{e^x}$
- Knowing that there is only one permutation of length zero in $\mathcal{A}$ we put $a_0 = 1$ into the equation at $x = 0$ and get $A = e^{-1}$
- We have now shown that $F = e^{e^x - 1}$ which is indeed the EGF for the Bell numbers.

Enumerations of Permutations Simultaneously Avoiding a Vincular and a Covincular Pattern of Length 3 (2017)

Paper by C. Bean, H. Ulfarsson and A. Claesson. Out of 40 results we could replicate 11, one of them shown below.



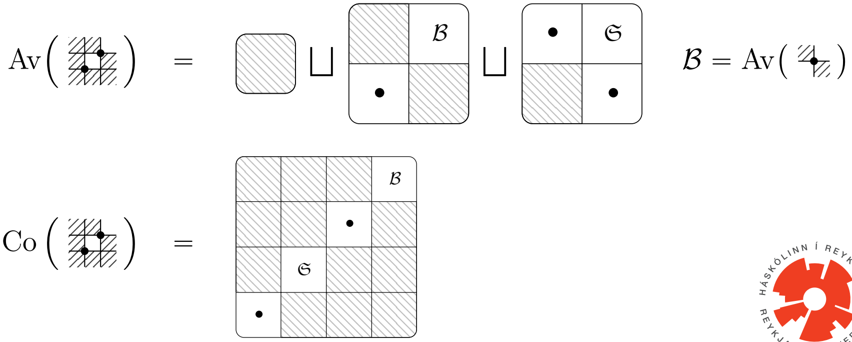
From this cover it is easy to see that the GF satisfies

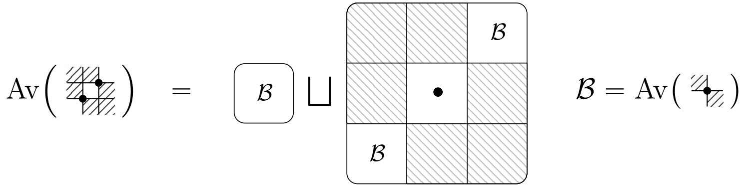
$$F(x) = 1 + xF(x) + x^2F(x)^2$$

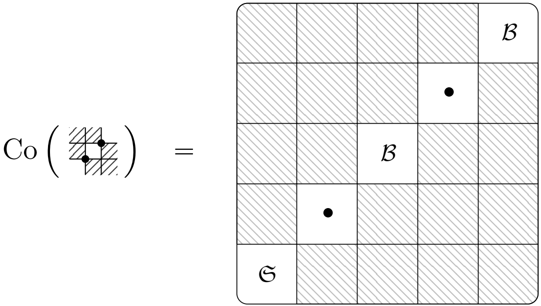
which indeed gives us the *Motzkin* numbers M_n to x^n in $F(x)$.

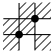
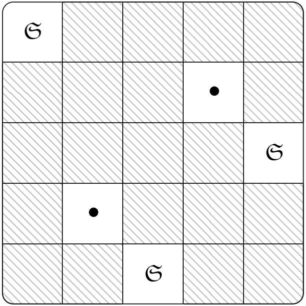
Wilf-Classification of Mesh Patterns of Short Length (2015)

Paper by Í. Hilmarsson, I. Jónsdóttir, S. Sigurðardóttir, L. Viðarsdóttir and H. Ulfarsson. Out of 65 results we managed to replicate 15. Some of them shown here without further comments.




Av
=
$\mathcal{B}$
$\bigsqcup$
$\mathcal{B}$
$\mathcal{B}$
$\mathcal{B}$ = Av


Co
=
$\mathcal{B}$
$\mathcal{B}$
$\mathfrak{S}$

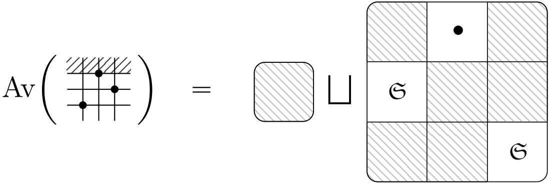
$\mathrm{Co}\Big($  $\Big) =$ 

Wilf classification of bi-vincular permutation patterns (2009)

Preprint by R. Parviainen. Out of 24 results we replicated 7.

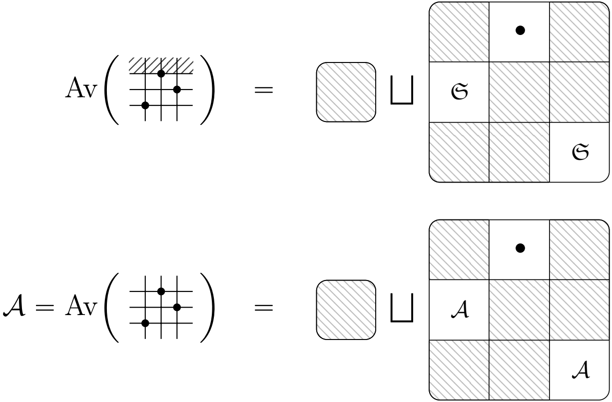
Wilf classification of bi-vincular permutation patterns (2009)

Preprint by R. Parviainen. Out of 24 results we replicated 7. Below is one of them.



Wilf classification of bi-vincular permutation patterns (2009)

Preprint by R. Parviainen. Out of 24 results we replicated 7. Below is one of them. It is interesting to compare it to the well known $\mathrm{Av}(132)$.



Conclusions

- We showed that CombCov is a powerful tool in guiding humans by coming up with conjectures that would otherwise have required substantial effort to discover manually.
- We were pleasantly surprised in how many published results it found covers for.
- CombCov's shortcomings may be remedied with improved ways of coming up with and generate the rules.

Any questions?

"There is no such thing as a dumb question."

— Carl Sagan