# Automatic Enumeration of Restricted Permutations

Michael Albert, Ragnar Ardal, *Christian Bean*, Anders Claesson, Jay Pantone and Henning Ulfarsson

June 26th, 2017

# Virtual combinatorist

Our goal is to create a virtual combinatorist. This is a three step problem.

1. have "a brilliant idea", we call these strategies
2. train the idea to the computer
3. let the computer do research

In particular we focus on permutation classes.

# Existing automatic methods

**Enumeration Schemes** - Zeilberger (1998), Vatter (2008)
- break up a permutation class into smaller parts and find recurrence relations.

**Insertion Encoding** - Albert et al. (2005), Vatter (2012)
- the insertion encoding is an encoding of finite permutations
- complete characterization for regular insertion encodable classes
- automatically compute rational generating function for regular insertion encodable classes

**Substitution Decomposition** Albert and Atkinson (2005), Bassino et al. (2012)
- derives a combinatorial specification for a permutation class
- needs the basis and set of simple permutations in the class, and both must be finite

# Struct

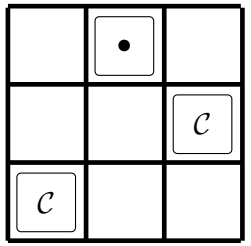
Given a permutation $\pi$ of length $n$, and two subsets $X, Y \subseteq [n]$, then $\pi(X \times Y)$ is the permutation that is order isomorphic to the subword with indices from $X$ and values in $Y$. For example $35216748([3,7] \times [2,6]) = 132$, from the subword 264.

Suppose $M$ is a $t \times u$ matrix whose entries are permutation sets. An *M-gridding* of a permutation $\pi$ of length $n$ is a pair of sequences $1 = c_1 \leq \cdots \leq c_{t+1} = n+1$ and $1 = r_1 \leq \cdots \leq r_{u+1} = n+1$ such that $\pi([c_k, c_k+1) \times [r_l, r_{l+1}))$ is in $M_{k,l}$ for all $k$ in $[t]$ and $\ell$ in $[u]$. The set $Grid(M)$ consists of all permutations with an $M$-gridding.

A tiling $T$ is a matrix as above which saitisfies the property that all permutations can have at most one $T$-gridding.

# Constructing permutation classes

Let $\mathcal{C}$ be the set of permutations avoiding 231. It has the Struct cover

$$\mathcal{C} = \square \sqcup$$ 

,

or simliarly

$$\mathcal{C} = \mathcal{E} + \mathcal{Z} \times \mathcal{C} \times \mathcal{C},$$

and its generating function

$$F = 1 + x \cdot F \cdot F.$$

# Combinatorial specification

A class $\mathcal{C}$ of combinatorial structures is a set of objects with a notion of size. A *constructible* combinatorial class is a set of structure that can be defined from atomic structures, possibly empty structures and assembled by means of admissable constructors.
A combinatorial specification for a combinatorial class $\mathcal{C}_1$ is an equation or system of equations of the form

$$\begin{cases} \mathcal{C}_1 = \mathcal{H}_1(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m), \\ \mathcal{C}_2 = \mathcal{H}_2(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m), \\ \vdots \\ \mathcal{C}_m = \mathcal{H}_m(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m) \end{cases}$$

where each $\mathcal{H}_i$ is a term built from $\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ and the admissable constructors.

# ATRAP

Struct found conjectures, but now we are looking to find proofs. Introducing our algorithm ATRAP: Automatic Theorems Regarding Avoidance of Patterns.

A combinatorial specification for a combinatorial class $\mathcal{C}_1$ is an equation or system of equations of the form

$$\begin{cases} \mathcal{C}_1 = \mathcal{H}_1(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m), \\ \mathcal{C}_2 = \mathcal{H}_2(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m), \\ \vdots \\ \mathcal{C}_m = \mathcal{H}_m(\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m) \end{cases}$$

where each $\mathcal{H}_i$ is a term built from $\mathcal{E}, \mathcal{Z}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ and the admissable constructors.

# Strategies

We have five types of strategies

- batch,
- verification,
- equivalence,
- inferral and
- decomposition.

Enumeration schemes and Insertion Encoding can be described by these types.

# Verification

This is the strategy we use to determine what are atomic structures are. If a tiling is verified by some verification strategy then we say that it is an atomic structure.

Example strategy: A tiling is verified if it is finite and a subset of $\mathcal{C}$. Consider the permutations avoiding 231 again. With this strategy the tiling

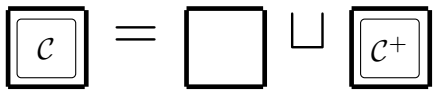
is verified.

# Batch

This is $+$. Given a tiling $T$, a batch strategy is a disjoint set of tilings $T_1, T_2, \ldots, T_k$, such that when each $T_i$ is intersected with $\mathcal{C}$ their disjoint union is equal to the intersection of $T$ and $\mathcal{C}$ i.e.

$$(T \cap \mathcal{C}) = (T_1 \cap \mathcal{C}) \sqcup (T_2 \cap \mathcal{C}) \sqcup \ldots \sqcup (T_k \cap \mathcal{C}).$$

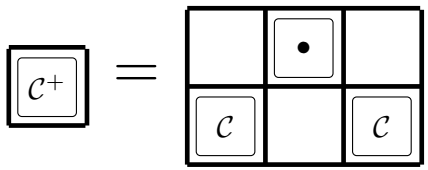
Example strategy: A tiling $T$ with a cell that is a set which is possibly empty is the disjoint union of the tiling where the cell is empty, and where the cell contains a point. Back to the avoiders of 231



# Equivalence

Given a tiling $T$, an equivalent strategy gives a tiling $T'$ such that $T'$ is isomorphic.

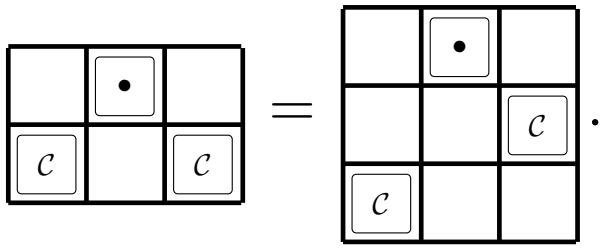
Example strategy: For a tiling with a cell that can not be empty, isolate an extreme point.



# Inferral

Given a tiling $T$, an inferral strategy gives a tiling $T'$ such that $T' \subset T$, but moreover $T \cap \mathcal{C} = T' \cap \mathcal{C}$.

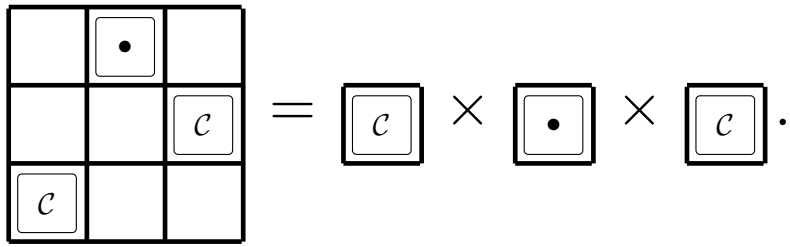
Example strategy: For a tiling $T$ with two cells in the same row, if you can't place a 21 across the cells, then separate them.
For the avoiders of 231 we get



# Recursive

For a tiling $T$ a recursive strategy is a set of tilings $T_1, T_2, \ldots, T_k$ such that there is a way to "glue" $T_1, T_2$ and $T_k$ together to give $T$.

Example strategy: For a tiling $T$ with cells that do not mix, such that no basis pattern is created across the cells, return the $1 \times 1$ tilings corresponding to the cells.
For the avoiders of 231 we get



# Examples

Check out permpal.ru.is

# Successes

We have found a proof tree for all permutation classes where the basis has permutations with length at most four, and at least one permutation of length less than four, including enumerations.

There exists a proof tree for all non-regular-insertion-encodable four-by-four permutation classes. 259 out of 363 have been enumerated from the tree.

We have proof trees for 26 non-insertion-encodable two-by-four permutation classes, with 7 being enumerated. In particular, there are proof trees for $Av(1243, 2341)$ enumerated by Bevan (2015) $Av(1342, 3412)$ enumerated by Miner (2016) and $Av(1234, 1324)$ conjectured to be non-ADE.
There is also a proof tree for $Av(1234)$.

# What's next?

Currently we have the framework for searching with the strategies as above. However, this is a prototype, and the code base will need written from the ground up, so is not yet available. There is also the challenge to come up with more strategies and improve our automated enumeration techniques.

Last week we implemented strategies which allows you to find a proof tree for the simple permutations in the class. It should be possible to inflate the structure and therefore give another proof method for enumerating classes.

Our algorithm is built upon the python library permuta which will be demoed at tomorrow's poster session.

All of our results are on permpal.ru.is. This will also be presented at tomorrow's poster session.