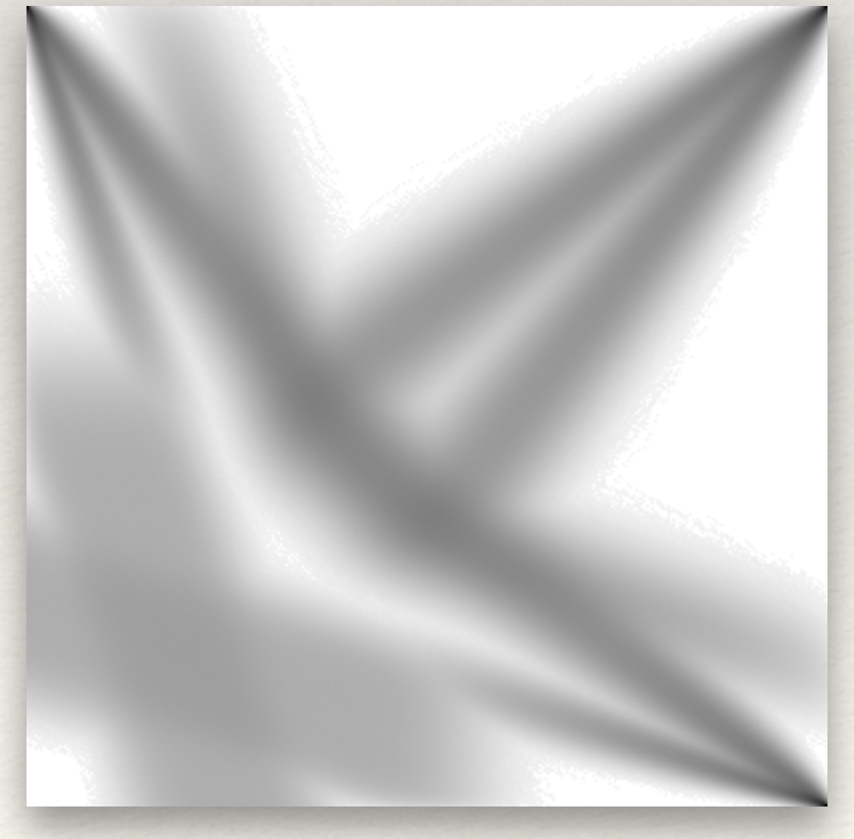
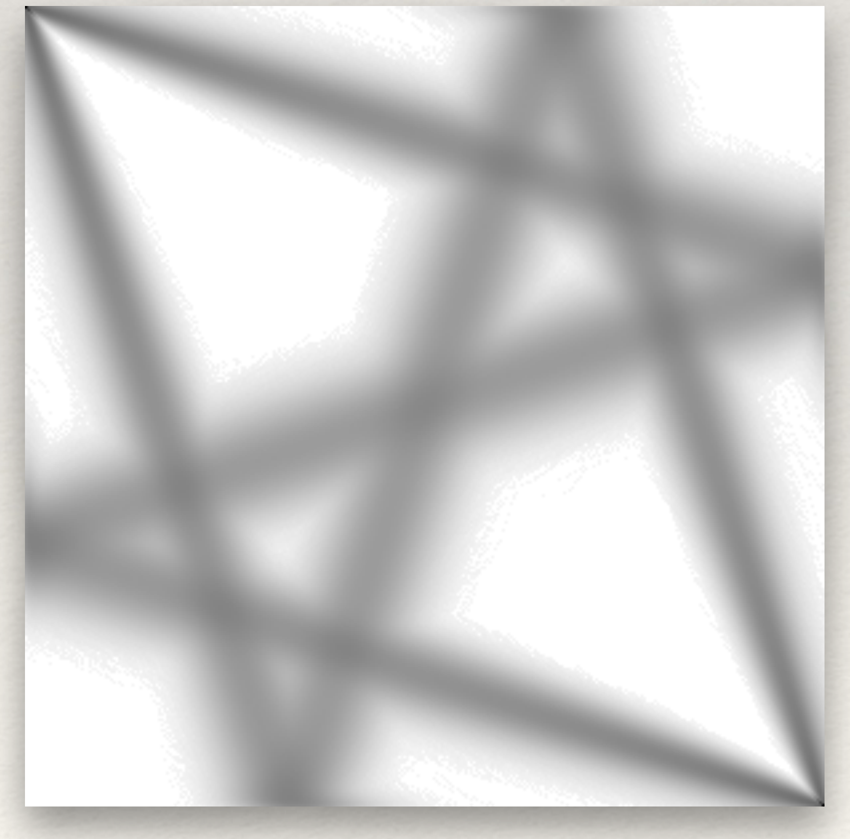
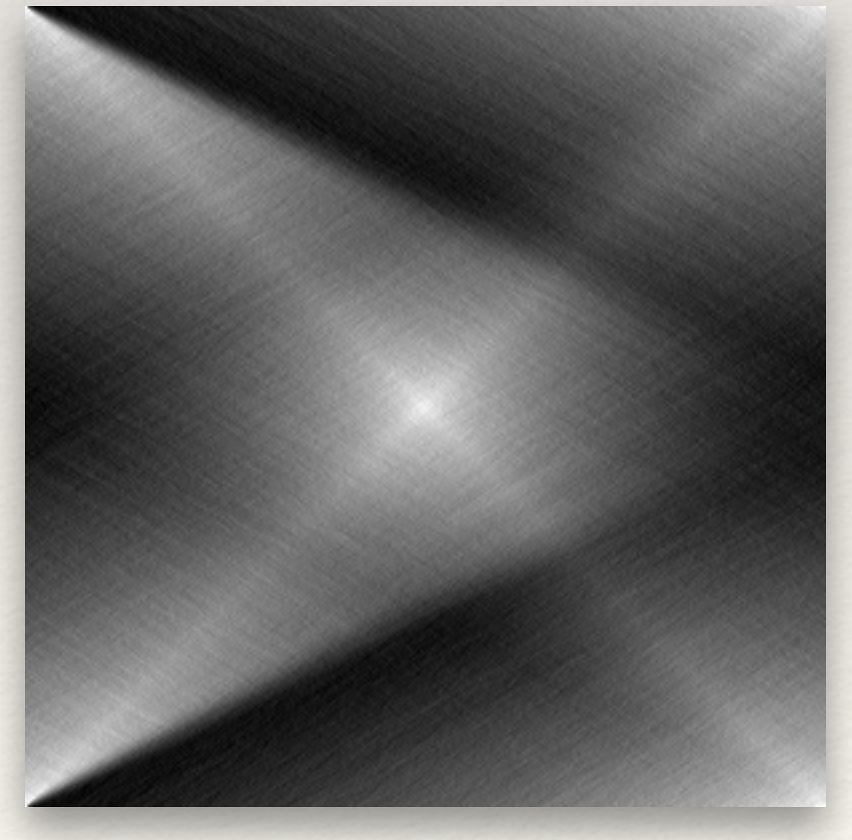
Combinatorial exploration and permutation classes

Christian Bean (he/him)
Keele University

joint work with Michael Albert, Anders Claesson, Émile Nadeau, Jay Pantone, and Henning Ulfarsson







Counting

Counting is the act of determining the number of objects in a set

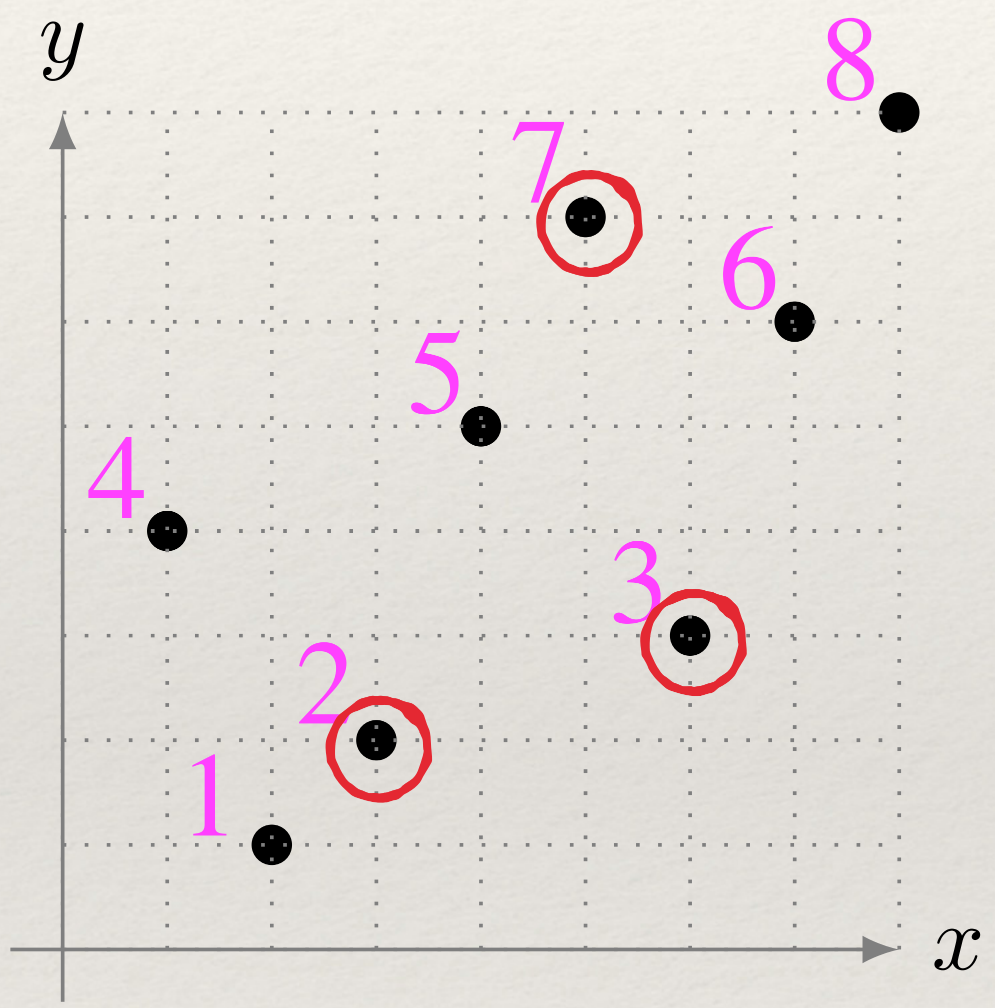
Combinatorics is often described as the mathematics of counting

A *combinatorial set* is a set of objects with a notion of size which has finitely many of each each size

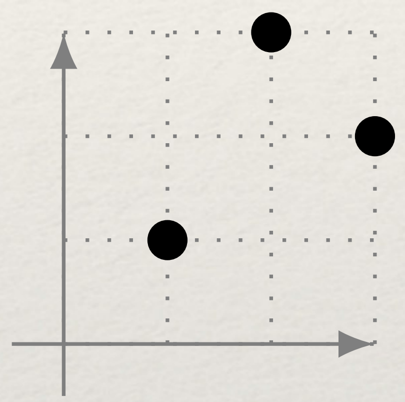
Examples
Words, lattice paths, graphs, set partitions, permutations, etc

This talk will focus on *permutation classes*, i.e., permutations avoiding substructures

The permutation 41257368



This permutation *contains* 132.



For example in the subsequence 273.

The containment order is a partial order on the set of permutations

Permutation classes

A *permutation class* is a set of permutations that is closed downwards

Uniquely defined by minimal permutations not in the set, called the *basis*

We write $Av(B)$ for the permutation class that avoids the permutations in the set B

Our question:

Given a basis B, how many permutations of size n are in $Av(B)$?

Example:

$Av(21)$?

For all $n \geq 0$, there is one permutation of size n in $Av(21)$.

Avoiding size three permutations

Theorem

For every permutation σ of size three, the size of $|\mathrm{Av}_n(\sigma)|$ is $C_n = \dfrac{1}{n+1}\dbinom{2n}{n}$.

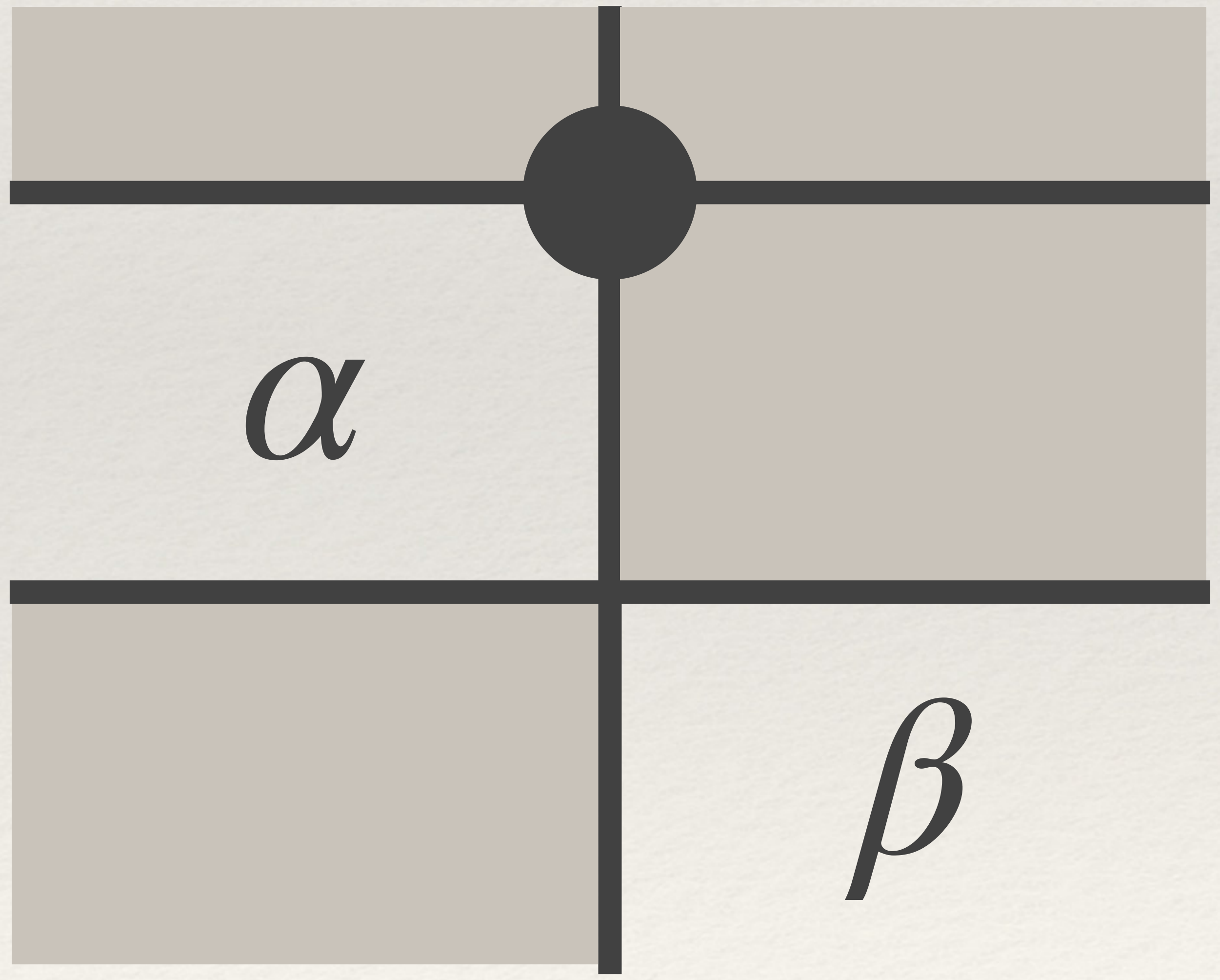
There are two symmetry classes: $\{123, 321\}$ and $\{132, 213, 231, 312\}$

Example

Av(132)?

A permutation in of size $n > 0$ in Av(132) can be written as $\alpha n \beta$ where α and β avoid 132.
All of the entries in α are below the entries in β.

Let $C_n = |\mathrm{Av}_n(132)|$ then $C_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{k=0}^{n-1} C_k C_{n-k-1} & \text{if } n > 0. \end{cases}$

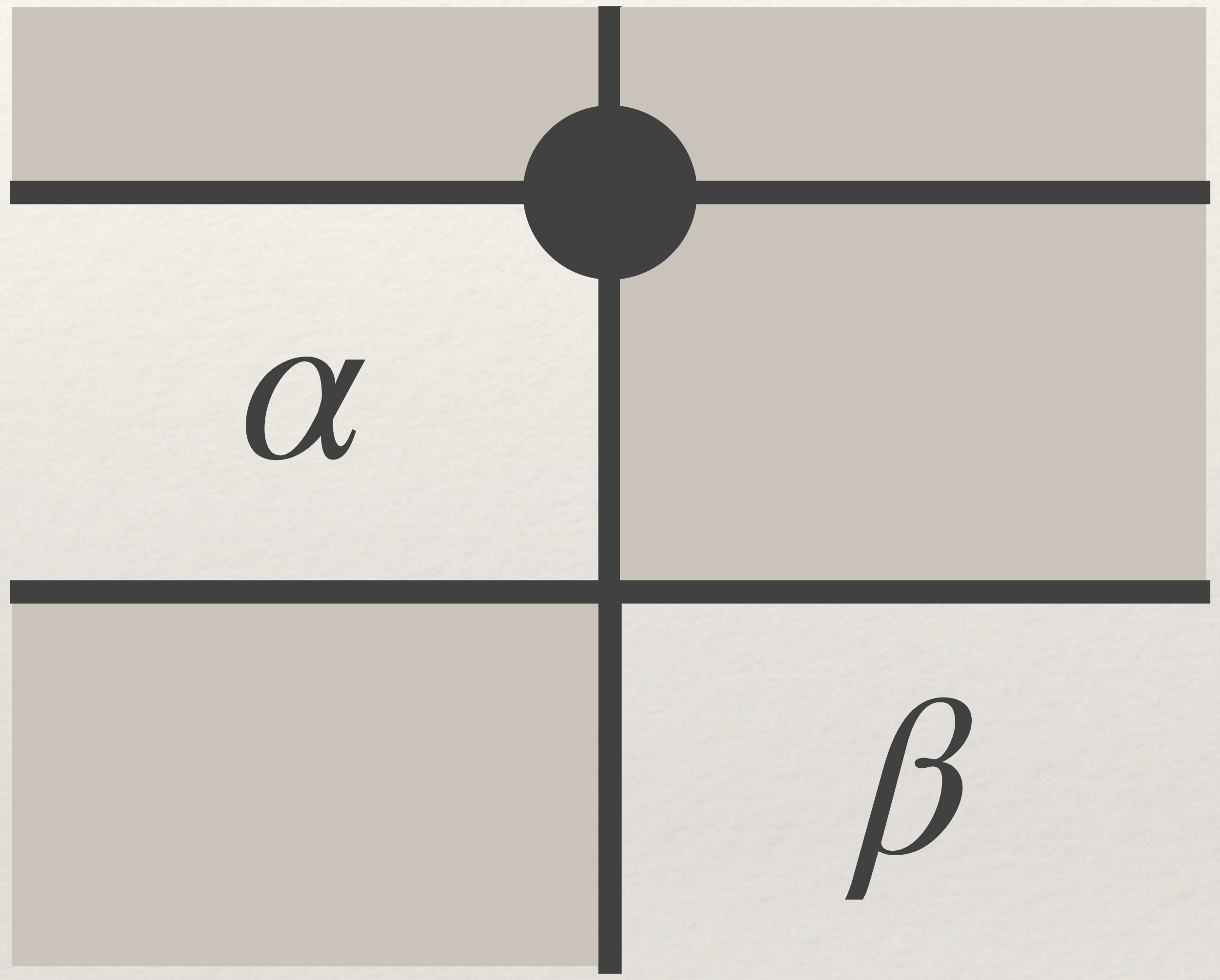


Example

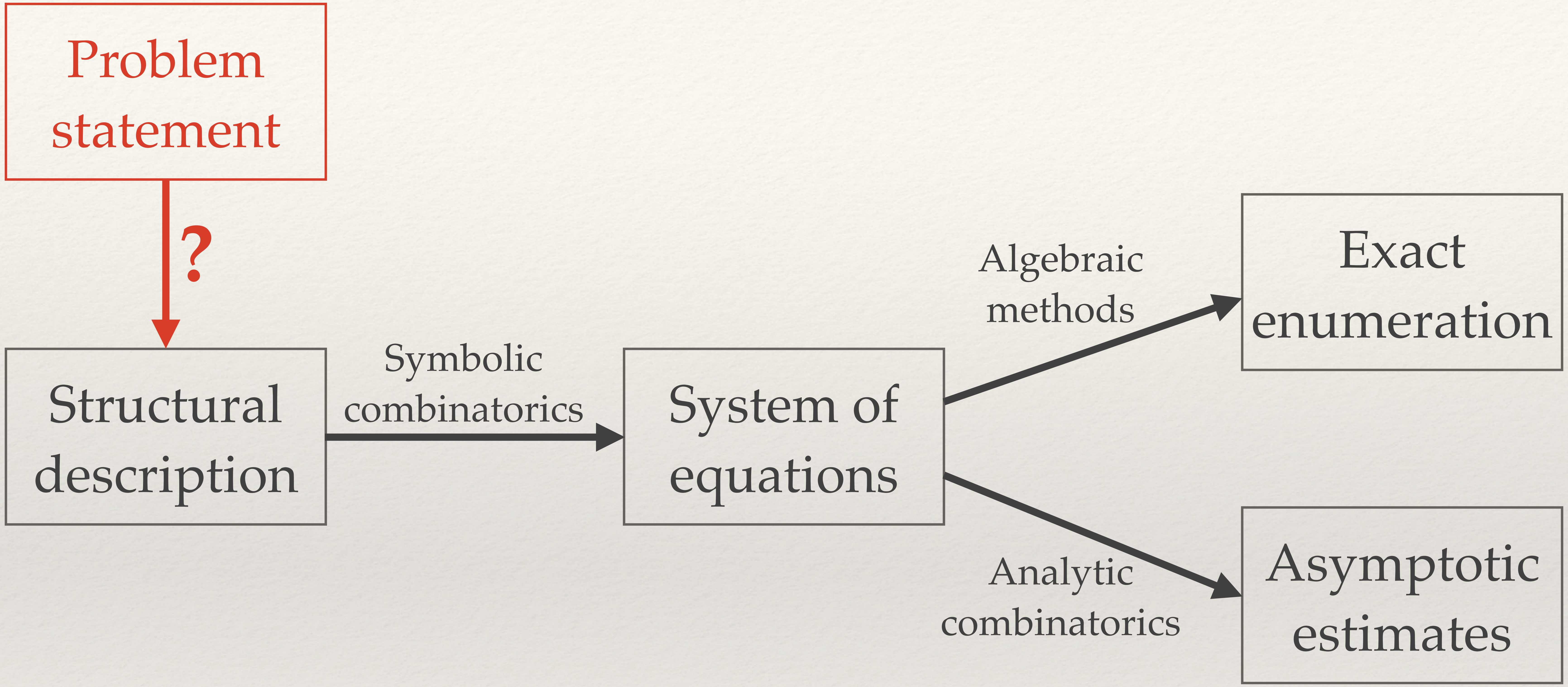
A permutation in of size $n > 0$ in Av(132) can be written as $\alpha n \beta$ where α and β avoid 132.
All of the entries in α are below the entries in β.

Let $F(x) = \sum_{n\geq 0} \left| \mathrm{Av}_n(132) \right| x^n$, then $F(x) = 1 + xF(x)^2$.

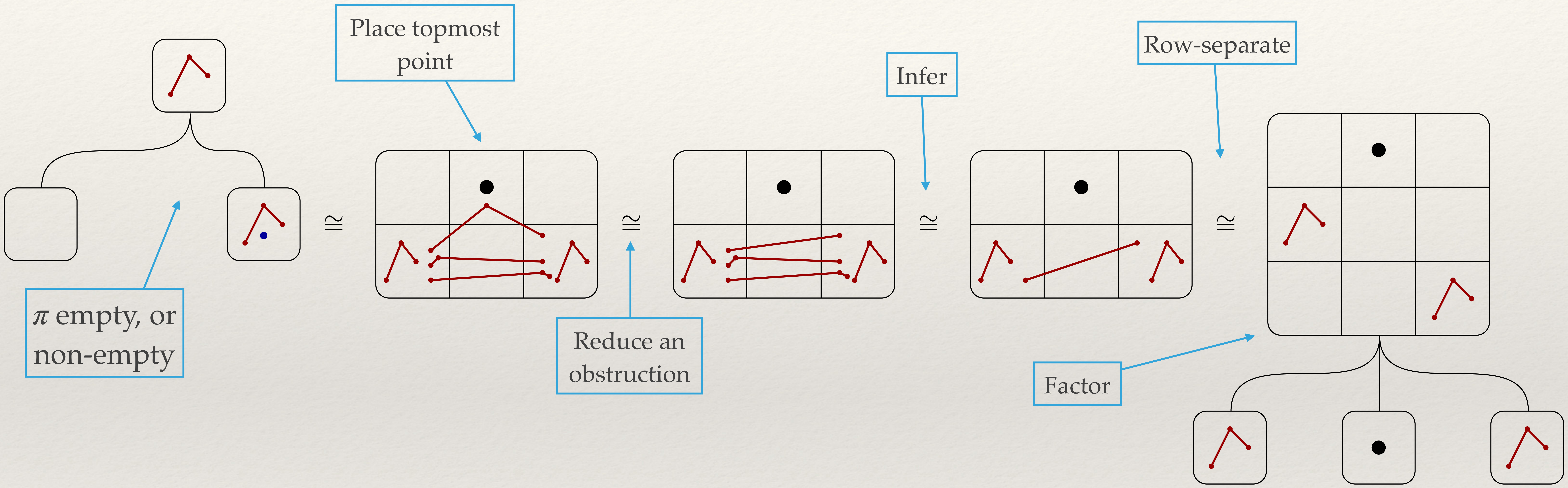
Solving gives $F(x) = \dfrac{1 - \sqrt{1-4x}}{2x}$.



A pipeline for enumerative combinatorics



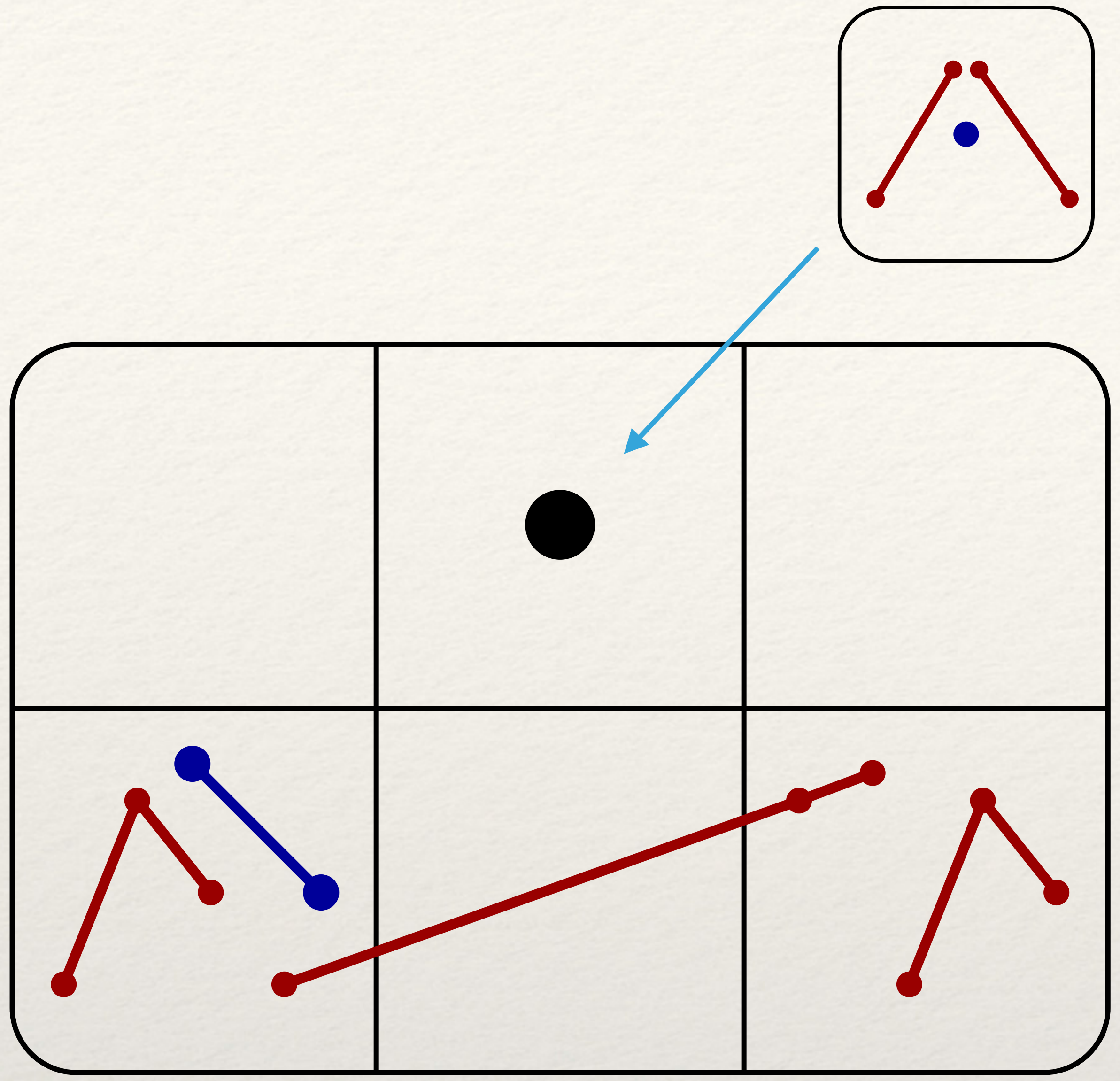
Av(132) by computer

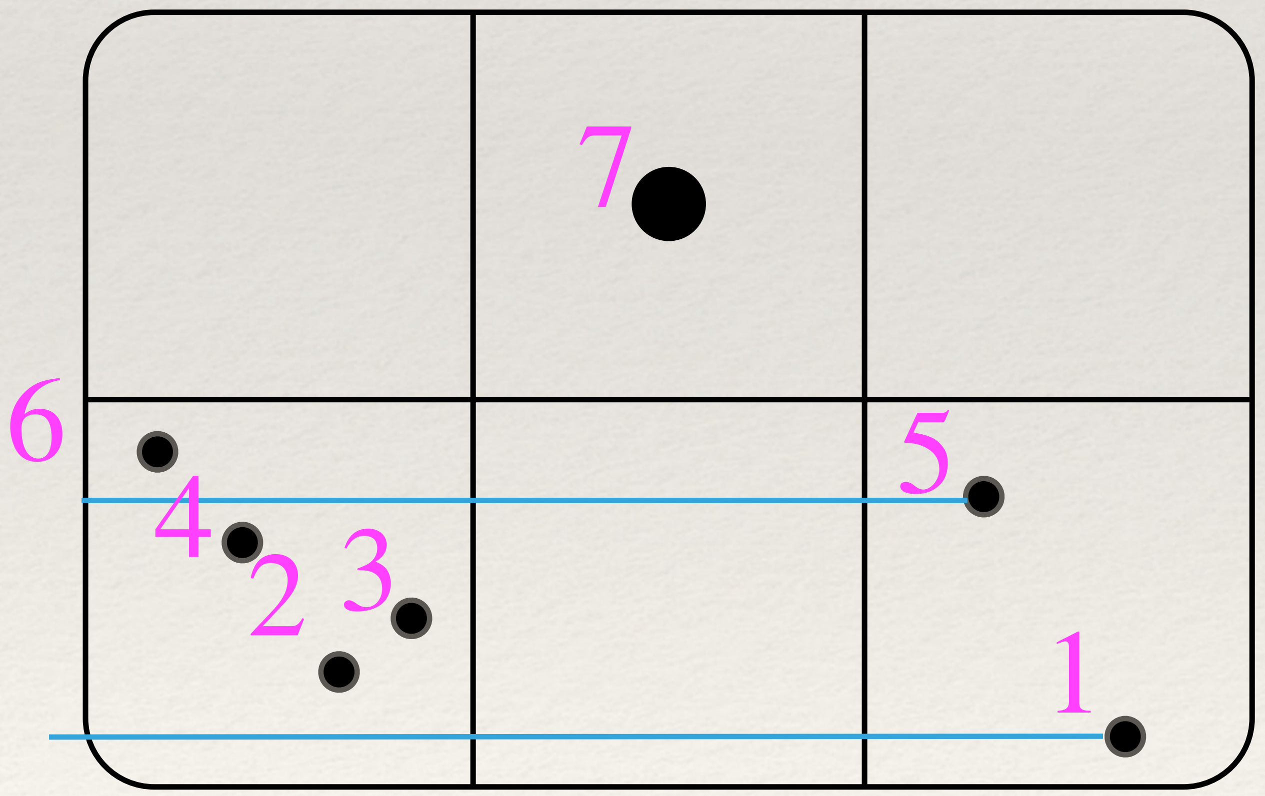


We call this a *proof tree*, or *combinatorial specification*

Each branching is called a *rule*, that comes from applying a *strategy*

Gridded permutations and tilings





A *tiling* is a triple $((n, m), \mathcal{O}, \mathcal{R})$, where (n, m) are the dimensions, $\mathcal{O}$ are the *obstructions*, and $\mathcal{R}$ are the *requirements*

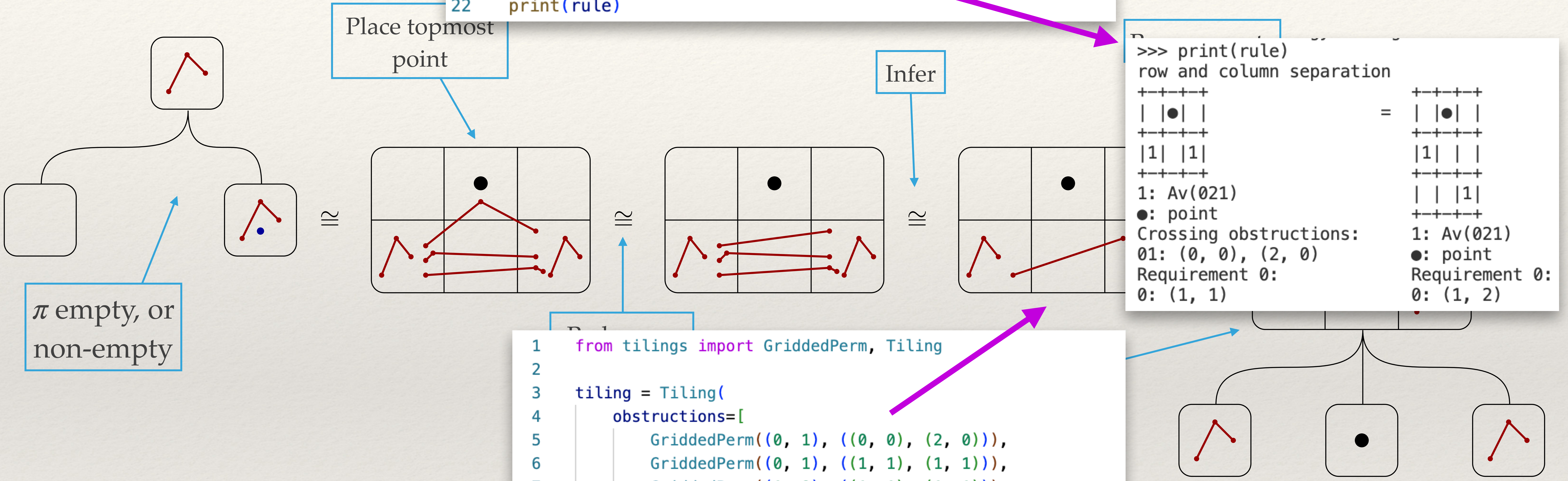
The tiling represents the set of (*gridded*) permutations that can be drawn on the tiling, without containing any obstruction, while containing every requirement

Here we get the permutation 6423751, although, strictly speaking we should also write the coordinate of each point

Av(132) by computer

```python
from tilings.strategies import RowColumnSeparationStrategy

strategy = RowColumnSeparationStrategy()
rule = strategy(tiling)
print(rule)
```




```
>>> print(rule)
row and column separation
+-+-+-+                    +-+-+-+
| |●| |                  = | |●| |
+-+-+-+                    +-+-+-+
|1| |1|                    |1| | |
+-+-+-+                    +-+-+-+
1: Av(021)                 | | |1|
●: point                   +-+-+-+
Crossing obstructions:     1: Av(021)
01: (0, 0), (2, 0)         ●: point
Requirement 0:             Requirement 0:
0: (1, 1)                  0: (1, 2)
```

```python
from tilings import GriddedPerm, Tiling

tiling = Tiling(
    obstructions=[
        GriddedPerm((0, 1), ((0, 0), (2, 0))),
        GriddedPerm((0, 1), ((1, 1), (1, 1))),
        GriddedPerm((1, 0), ((1, 1), (1, 1))),
        GriddedPerm((0, 2, 1), ((0, 0), (0, 0), (0, 0))),
        GriddedPerm((0, 2, 1), ((2, 0), (2, 0), (2, 0))),
    ],
    requirements=[
        [
            GriddedPerm((0,), ((1, 1),)),
        ],
    ],
)
```

Insertion encoding

A language for encoding permutations

$\diamond \mapsto \diamond n \diamond$	represented by **m** (for middle)
$\diamond \mapsto n \diamond$	represented by **l** (for left)
$\diamond \mapsto \diamond n$	represented by **r** (for right)
$\diamond \mapsto n$	represented by **f** (for fill)

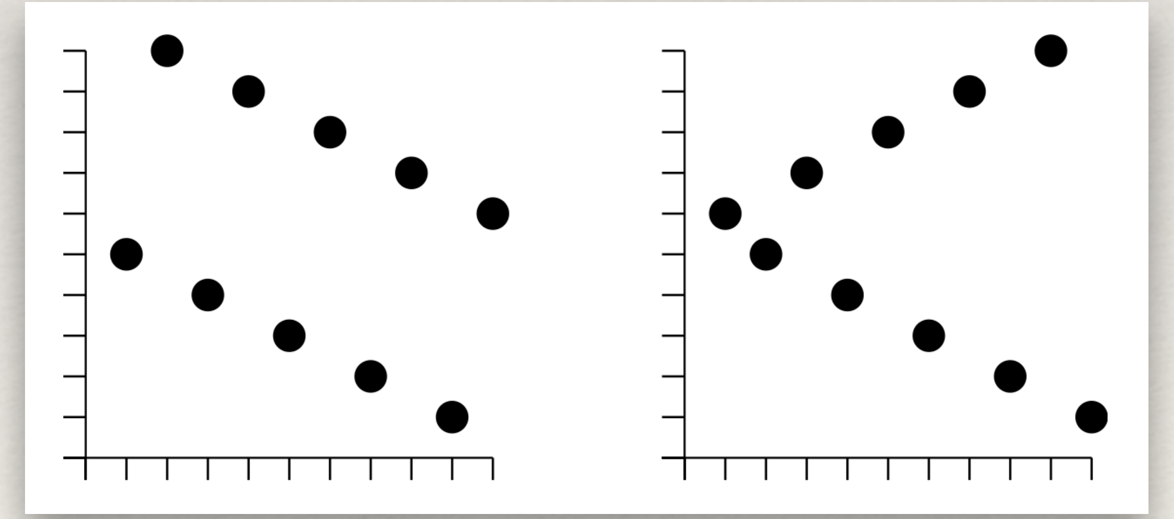
$\diamond$
$\diamond\ 1\ \diamond$
$\diamond\ 2\ \diamond\ 1\ \diamond$
$3\ 2\ \diamond\ 1\ \diamond$
$3\ 2\ \diamond\ 1\ 4\ \diamond$
$3\ 2\ 5\ 1\ 4\ \diamond$
$3\ 2\ 5\ 1\ 4\ 6$

The insertion encoding of 325146 is $\mathbf{m}_1\mathbf{m}_1\mathbf{f}_1\mathbf{l}_2\mathbf{f}_1\mathbf{f}_1$

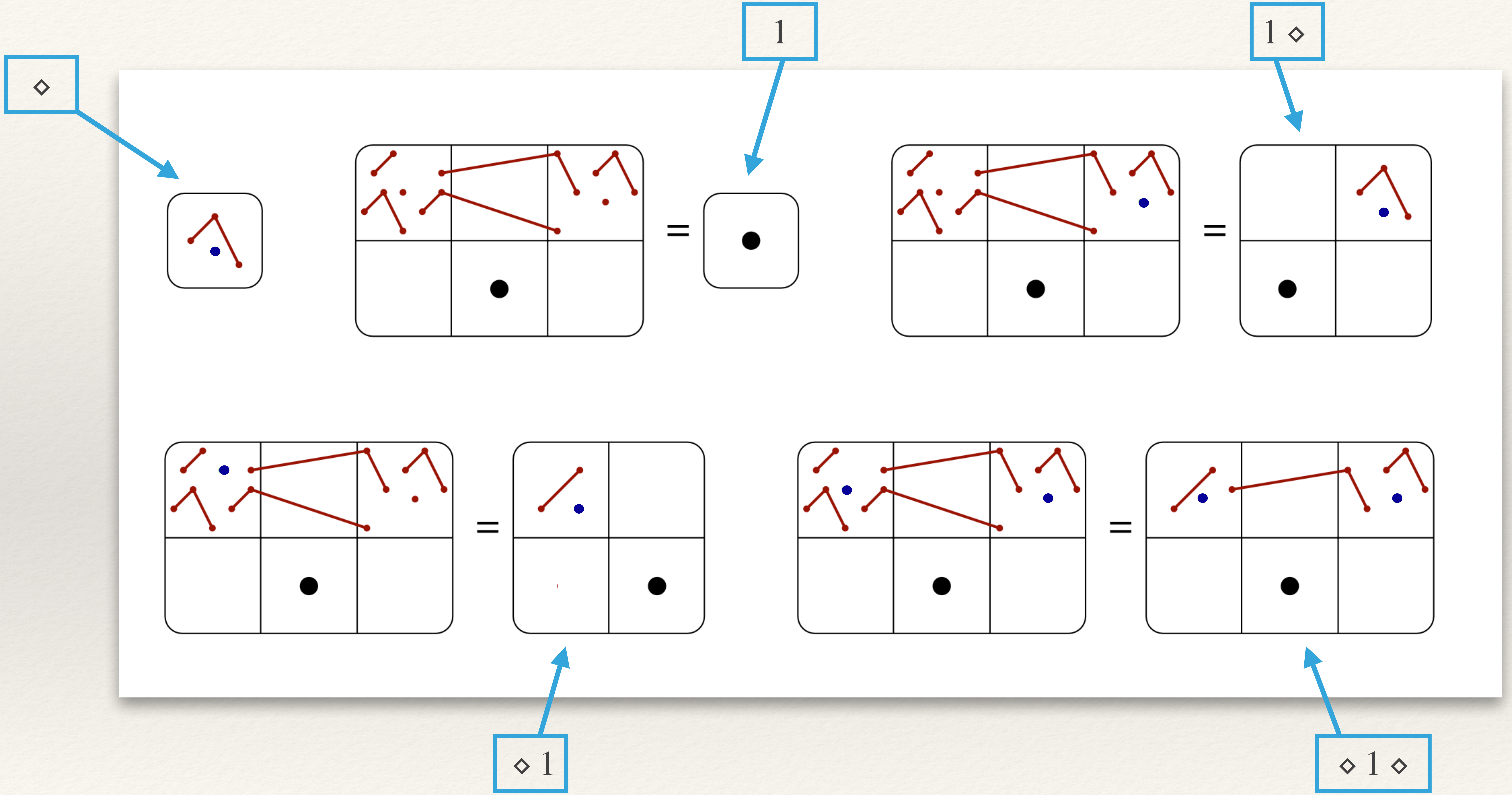
Let $\mathscr{L}(\mathscr{C})$ be the language formed by the insertion encodings of a permutation class $\mathscr{C}$

Theorem [Albert, Linton, and Ruškuc (2005), Vatter (2012)]
For a permutation class Av(B), the following are equivalent

1. The language $\mathscr{L}(\text{Av}(B))$ is regular
2. There are at most k slots in any evolution
3. The set B contains at least one permutation in each of Av(132, 312), Av(213, 231), Av(123, 3142, 3412), and Av(321, 2143, 2413)



Insertion encoding as tilings

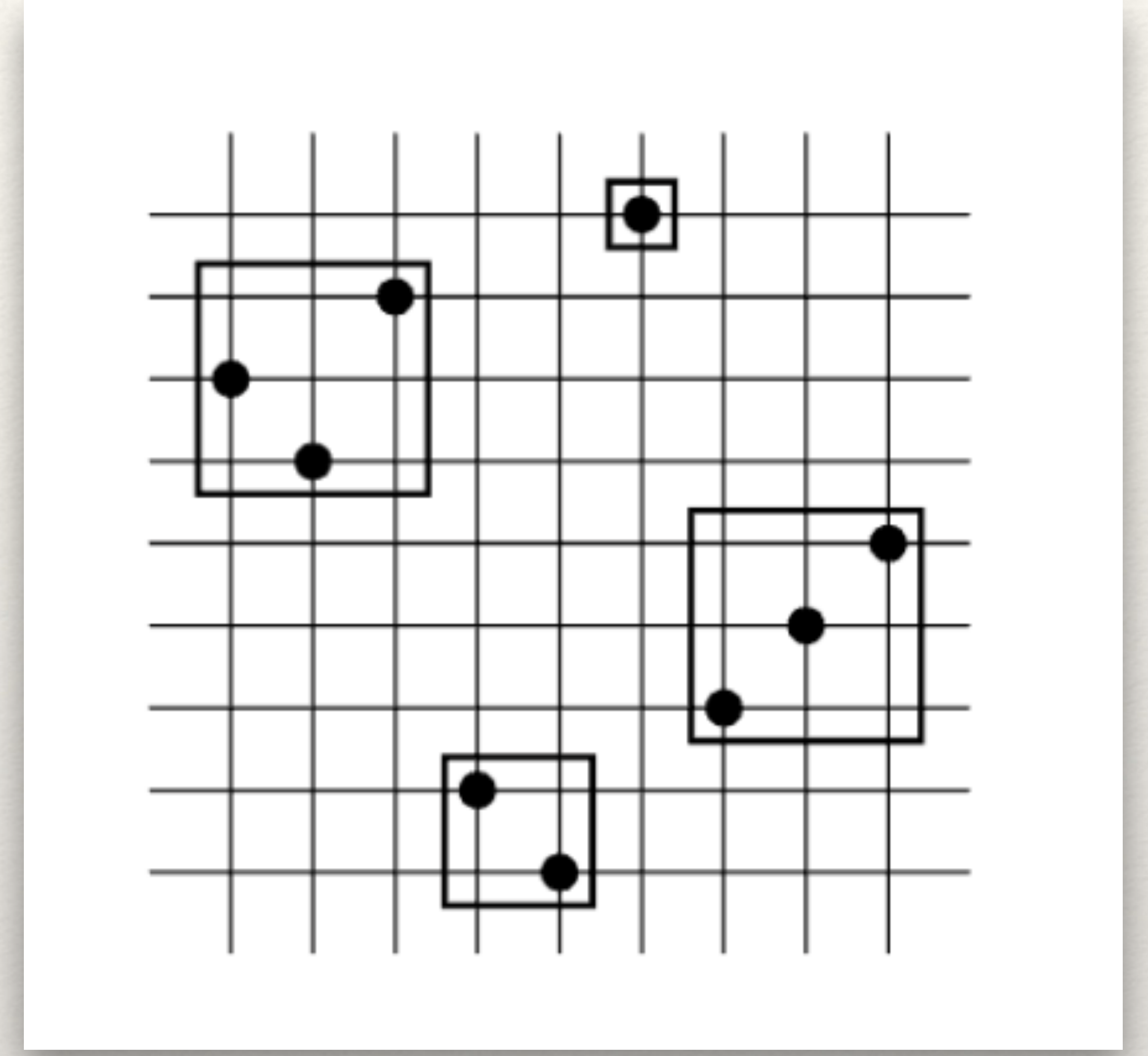


Inflations of simple permutations

An *interval* in a permutation is a set of consecutive indices where the values are consecutive

A permutation of size at least two is *simple* if it has no proper intervals

Theorem [Albert and Atkinson (2005)]
Every permutation of size at least two can be written as an inflation $\pi[\sigma_1, \ldots, \sigma_m]$ where π is a unique simple permutation. If $m \geq 4$, the σ_i are unique.
If $\pi = 12$ (or $\pi = 21$ resp.), and σ_1 is sum-indecomposable (or skew-indecomposable resp.) then it is also unique.

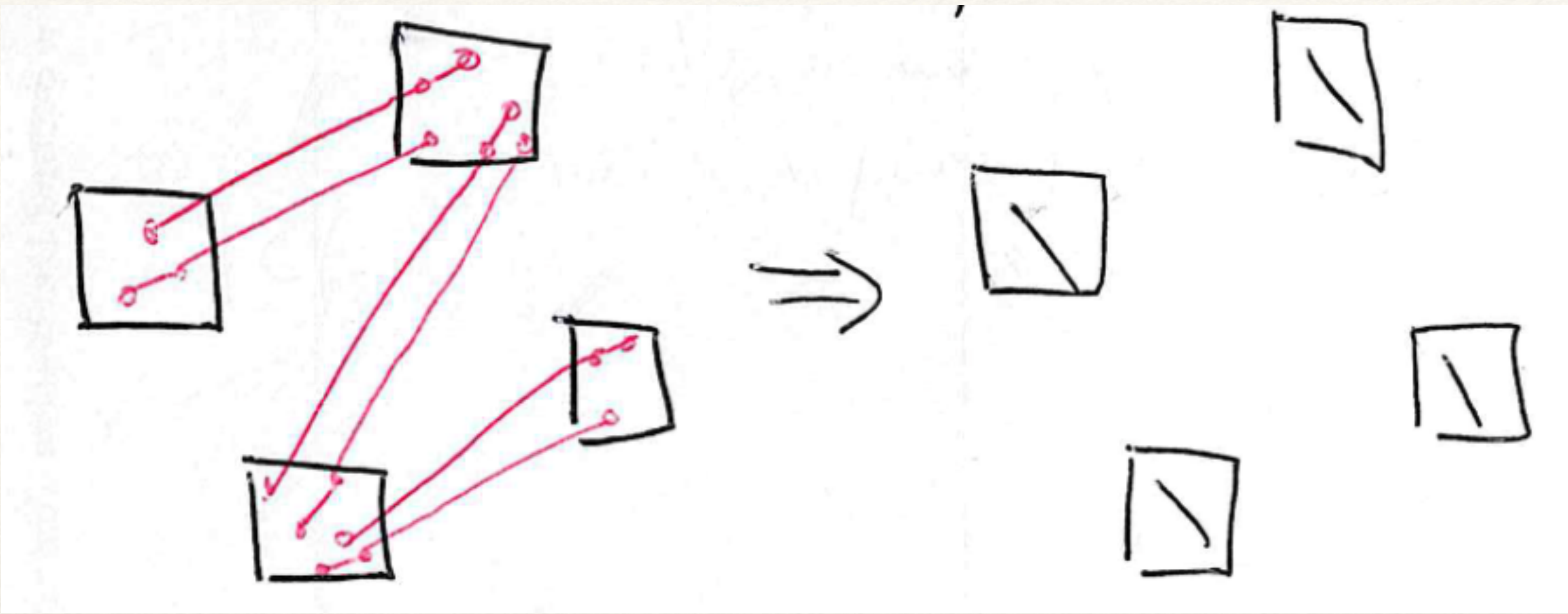


3142[213, 21, 1, 123]

Theorem [Albert and Atkinson (2005)]
A permutation class with finitely many simple permutations is finitely based and has an algebraic generating function.

This gives a recipe to build the inflations of a simple permutation in a permutation class

It can be extended to any permutation class with finitely many simple permutations



$$\left(\frac{x}{1-x}\right)^4$$

Theorem [Brignall, Ruškuc, and Vatter (2008)]
It is decidable if a permutation class has finitely many simple permutations.

Theorem [Bassino, Bouvel, Pierrot, and Rossin (2015)]
For a fixed basis B, there is a polynomial time algorithm that decides if $\mathrm{Av}(B)$ has finitely many simple permutations.

An algorithm for this procedure was given by Bassino, Bouvel, Pierrot, Pivoteau, and Rossin (2017)

Apply strategies to create a *universe* of rules, and then search for a specification within the universe

Algorithm 1 Combinatorial Specification Searcher

Input: A set of combinatorial rules U
Output: The union of all combinatorial specifications contained in U

changed $\leftarrow$ **True**
while *changed* **do**
  changed $\leftarrow$ **False**
  for $\mathcal{A} \xleftarrow{S} (\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}) \in U$ **do**
    if any $\mathcal{B}^{(j)}$ is not on the left-hand side of any rule in U **then**
      $U \leftarrow U \setminus \{\mathcal{A} \xleftarrow{S} (\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)})\}$
      changed $\leftarrow$ **True**
    end if
  end for
end while
$V \leftarrow U$
return V

Theorem 3.1. For any set of combinatorial rules U, the set V returned by Algorithm 1 is equal to the union of all combinatorial specifications that are contained in U.

Definition 4.2. We call an m-ary strategy S a *productive strategy* if the following two conditions hold for all combinatorial sets $\mathcal{A}$ with corresponding decomposition $d_S(\mathcal{A}) = (\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)})$, and for all $i \in \{1, \ldots, m\}$.

1. For all $N \in \mathbb{N}$, if $\mathcal{A}_N$ relies on $\mathcal{B}_j^{(i)}$, then $j \leqslant N$.
2. If $\mathcal{A}_N$ relies on $\mathcal{B}_N^{(i)}$ for some $N \in \mathbb{N}$, then
   (a) $|\mathcal{A}_n| \geqslant |\mathcal{B}_n^{(i)}|$ for all $n \in \mathbb{N}$, and
   (b) $|\mathcal{A}_\ell| > |\mathcal{B}_\ell^{(i)}|$ for some $\ell \in \mathbb{N}$.

Theorem 4.3. Let P be a proof tree, or the equivalent combinatorial specification, composed entirely of rules derived from productive strategies. Then P is productive, i.e., the infinite system of equations derived from its counting functions has a unique solution.

TileScope

We call our implementation of combinatorial exploration with tilings the TileScope algorithm

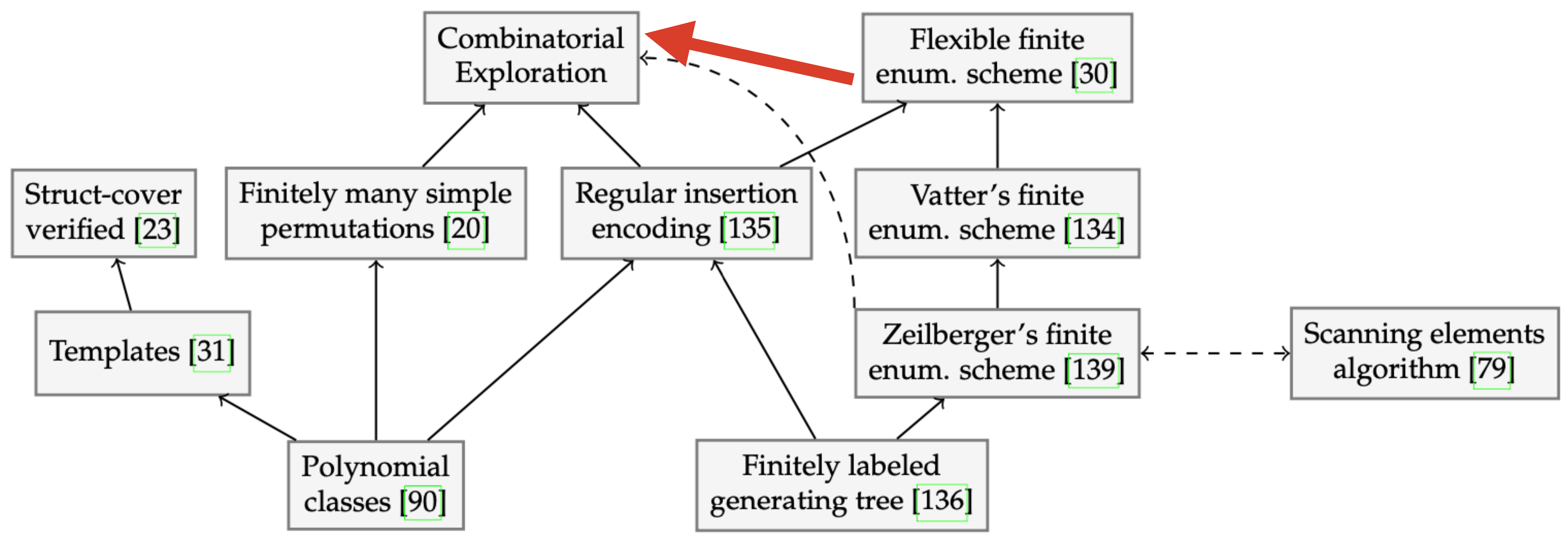


Figure 3: Comparison of algorithmic enumeration methods.

As well as unifying earlier methods, one key advantage of TileScope is its ability to utilise a growing library of strategies in a simultaneous manner to build a greater understanding of the structure of the permutation classes

Combinatorial exploration

Apply strategies to create a *universe* of rules, and then search for a specification within the universe

What is a specification? A set of rules where each class appears once on the left. What is a rule? First, we need strategies. Let X be the set of combinatorial sets.

Def (an m-ary strategy S)

1. decomposition function

$$d_S : X \to X^m \cup \{DNA\}$$

(DNA: does not apply)

$$d_S(A) = (B^{(1)}, \dots, B^{(m)})$$

2. reliance profile function

$$r_S : \mathbb{N} \to \mathbb{N}^m$$

Let $r_S^{(i)}(n)$ be the i^{th} component

$$r_S(n) = (r_S^{(1)}(n), \dots, r_S^{(m)}(n))$$

3. counting functions, $c_{S,(n)}$ with $n \in \mathbb{N}$.

$$c_{S,(n)}(\omega^{(1)}(n), \dots, \omega^{(m)}(n)) = |A_n|$$

with

$$\omega^{(i)}(n) = (|B_0^{(i)}|, \dots, |B^{(i)}_{r_S^{(i)}(n)}|).$$

Examples of strategies

Example (size 0 or not, Z)

1. $d_Z(A) = (B, C)$ — B: size 0, C: not — $A = B \dot{\cup} C$

2. $r_Z(n) = (n, n)$

3. $c_{Z,(n)}((b_0, \ldots, b_n), (c_0, \ldots, c_n)) = b_n + c_n$

Example (factor around max F)

1. $d_F(A) = \begin{cases} (B, C) & \text{if } A = \\ \text{DNA} \end{cases}$

2. $r_F(n) = (n-1, n-1)$

3. $c_{F,(n)}((b_0, \ldots, b_{n-1}), (c_0, \ldots, c_{n-1})) = \sum_{j=0}^{n-1} b_j c_{n-j}$ — $A(x) = xB(x)C(x)$

If $d_S(A) = (B^{(1)}, \ldots, B^{(m)})$

we write $A \overset{S}{\leftarrow} (B^{(1)}, \ldots, B^{(m)})$ — combinatorial rule

Productive specifications

A specification is a set of rules where each combinatorial set appears exactly once on the LHS.

Def: (productive)

A specification with k rules is productive. The counting functions have a unique solution in $(\mathbb{C}^{\mathbb{N}})^k$

Reliance graphs

An infinite directed graph for a specification

For each combinatorial set $\mathcal{B}^{(i)}$, there is infinite set of vertices $\{\mathcal{B}_n^{(i)} : n \in \mathbb{N}\}$.

Directed edge from $\mathcal{B}_{j_1}^{(i_1)}$ to $\mathcal{B}_{j_2}^{(i_2)}$:

- $\mathcal{B}^{(i_2)}$ on RHS of rule with LHS $\mathcal{B}^{(i_1)}$
- $\mathcal{B}_{j_1}^{(i_1)}$ relies on $\mathcal{B}_{j_2}^{(i_2)}$

i.e if $\mathcal{B}^{(i_2)}$ is the k^{th} in $d_s(\mathcal{B}^{(i_1)})$

then $j_2 \leq r_s^{(k)}(j_1)$

Theorem 4.1. Let P be a proof tree (or the corresponding specification) involving combinatorial sets $\mathcal{B}^{(1)}, \mathcal{B}^{(2)}, \ldots, \mathcal{B}^{(N)}$ and whose reliance graph contains no infinite directed walks. Let $\mathcal{S}(P)$ be the system of equations in the indeterminates $\{b_j^{(i)} : j \in \mathbb{N},\ 1 \leqslant i \leqslant N\}$. There exists a unique solution to the system

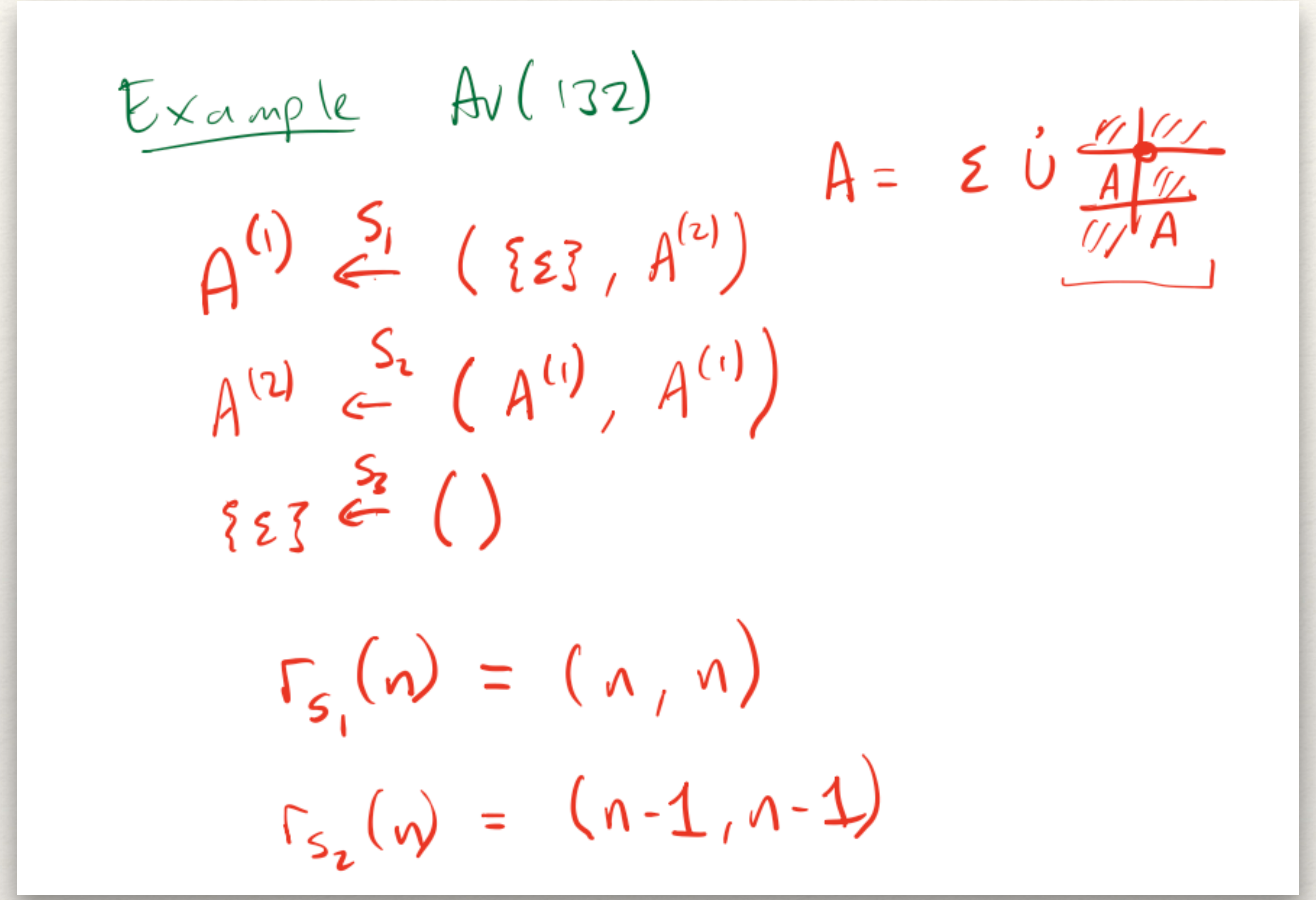
$$\left(\left(\tilde{b}_0^{(1)}, \tilde{b}_1^{(1)}, \ldots\right), \left(\tilde{b}_0^{(2)}, \tilde{b}_1^{(2)}, \ldots\right), \ldots, \left(\tilde{b}_0^{(N)}, \tilde{b}_1^{(N)}, \ldots\right)\right) \in \left(\mathbb{C}^{\mathbb{N}}\right)^N.$$

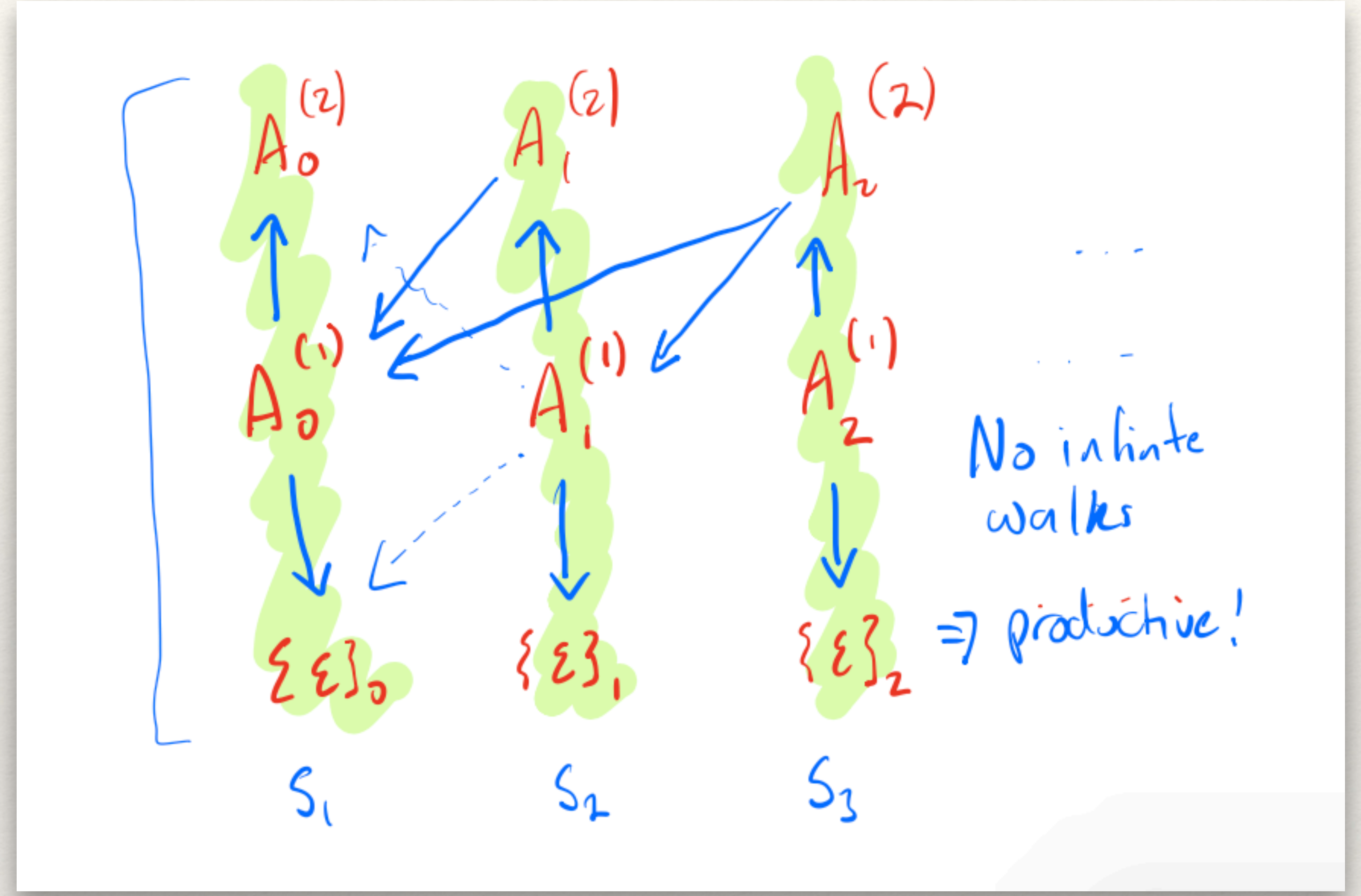
In other words, P is a productive proof tree.

Theorem:

A specification is productive if its reliance graph contains no infinite walks.

Example of reliance graph





Combinatorial exploration

Apply strategies to create a *universe* of rules, and then search for a specification within the universe

Definition 4.2. We call an m-ary strategy S a *productive strategy* if the following two conditions hold for all combinatorial sets $\mathcal{A}$ with corresponding decomposition $d_S(\mathcal{A}) = (\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)})$, and for all $i \in \{1, \ldots, m\}$.

1. For all $N \in \mathbb{N}$, if $\mathcal{A}_N$ relies on $\mathcal{B}_j^{(i)}$, then $j \leqslant N$.
2. If $\mathcal{A}_N$ relies on $\mathcal{B}_N^{(i)}$ for some $N \in \mathbb{N}$, then
   (a) $|\mathcal{A}_n| \geqslant |\mathcal{B}_n^{(i)}|$ for all $n \in \mathbb{N}$, and
   (b) $|\mathcal{A}_\ell| > |\mathcal{B}_\ell^{(i)}|$ for some $\ell \in \mathbb{N}$.

Theorem 4.3. Let P be a proof tree, or the equivalent combinatorial specification, composed entirely of rules derived from productive strategies. Then P is productive, i.e., the infinite system of equations derived from its counting functions has a unique solution.

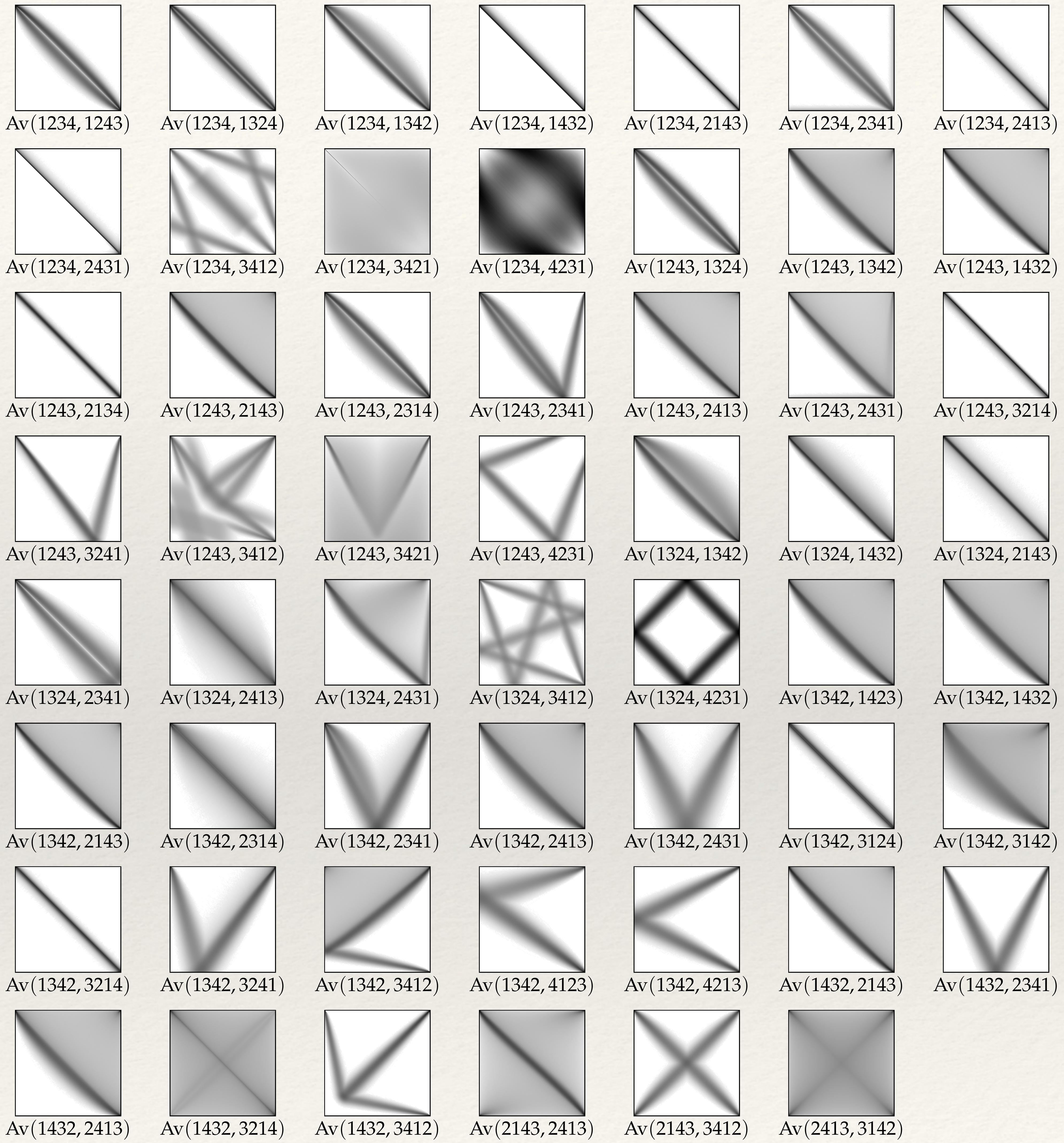
There are productive specification with rules from strategies that are not productive.
A story for another day.

This is a 99 page preprint available on the arXiv: https://arxiv.org/abs/2202.07715 to appear in Memoirs of the AMS.

- automatically and rigorously study the structure of combinatorial sets and derive their counting sequences and generating functions
- enumerate and randomly sample objects
- strengthen the foundations of combinatorial specifications

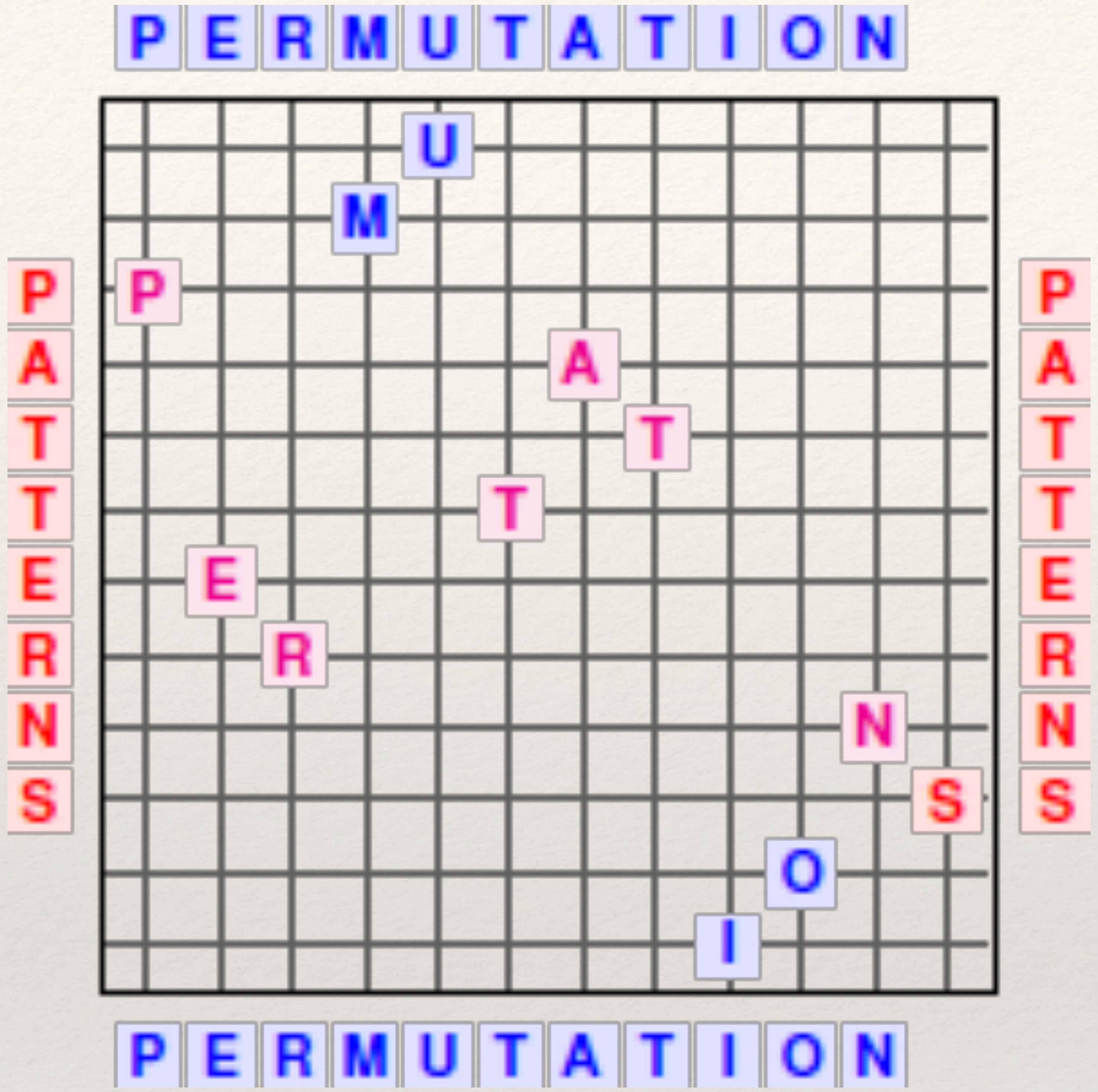
This is a 99 page preprint available on the arXiv: https://arxiv.org/abs/2202.07715 to appear in Memoirs of the AMS.

- apply extensively to permutation pattern problems
    - rederive hundreds of results in literature in a unified manner and prove many new results
    - share the results in a new public database: https://permpal.com



Other directions

- Forests - specifications with non-productive strategies!
- Bijections
- Other objects, e.g, lattice paths, set partitions, alternating sign matrices, polyominoes
- Mesh patterns and grid classes - we’ve all had enough definitions, for another day…



University of St Andrews, Scotland, July 7th-11th 2025

Invited speakers are David Bevan and Natasha Blitvic

Important dates:

Abstract submission deadline: April 11th 2025
Early registration deadline: May 1st 2025
Late registration deadline: June, 2025

Conference dates: July 7th-11th 2025
Pre-conference workshop: July 4th-5th 2025

End credits (start at 4.38)