How to Cleverly Count Pattern-Avoiding Words

Lara Pudwell

Rutgers University

San Diego Joint Math Meetings
January 8, 2008

Outline

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of `2674425` is

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of 2674425 is 1●●●●1●.

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of 2674425 is 1••221•.

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of 2674425 is 1●●2213.

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of 2674425 is 14•2213.

Reduction

- Given a string of letters $q = q_1 \cdots q_n$, the reduction of q is the string obtained by replacing the i^{th} smallest letter(s) of q with i.
- For example, the reduction of `2674425` is `1452213`.

Pattern Avoidance in Words

- Given strings $w = w_1 \cdots w_n$ and $q = q_1 \cdots q_m$, w contains q as a pattern if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \cdots w_{i_m}$ reduces to q.

Pattern Avoidance in Words

- Given strings $w = w_1 \cdots w_n$ and $q = q_1 \cdots q_m$, w contains q as a pattern if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \cdots w_{i_m}$ reduces to q.
- Otherwise w avoids q.

Pattern Avoidance in Words

- Given strings $w = w_1 \cdots w_n$ and $q = q_1 \cdots q_m$, w contains q as a pattern if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \cdots w_{i_m}$ reduces to q.
- Otherwise w avoids q.
- 1452213 contains 312 (1452213)

Pattern Avoidance in Words

- Given strings $w = w_1 \cdots w_n$ and $q = q_1 \cdots q_m$, w contains q as a pattern if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \cdots w_{i_m}$ reduces to q.
- Otherwise w avoids q.
- 1452213 contains 312 (1452213)
  1452213 avoids 212.

Pattern Avoidance in Words

- **Easy Question:** Fix w. What patterns are contained in w?

Pattern Avoidance in Words

- **Easy Question:** Fix w. What patterns are contained in w?
  $w = 14522$ contains 1, 12, 11, 21, 122, 123, 132, 211, 231, 1322, 1342, 2311, and 13422 as patterns.

Pattern Avoidance in Words

- **Easy Question:** Fix w. What patterns are contained in w?
  $w = 14522$ contains 1, 12, 11, 21, 122, 123, 132, 211, 231, 1322, 1342, 2311, and 13422 as patterns.
- **Hard Question:** Fix q.
  Enumerate $A_{[a_1,\dots,a_k],Q} :=$
  $\{w \in [k]^{\sum a_i} \mid w \text{ has } a_i\ i\text{'s}, w \text{ avoids } q \text{ for every } q \in Q\}$

Previous Work for Words

- For words, results by...
  - Burstein: initial results, generating functions

Previous Work for Words

- For words, results by...
  - Burstein: initial results, generating functions
  - Albert, Aldred, Atkinson, Handley, Holton: results for specific 3-letter patterns

Previous Work for Words

- For words, results by...
  - Burstein: initial results, generating functions
  - Albert, Aldred, Atkinson, Handley, Holton: results for specific 3-letter patterns
  - Brändén, Mansour: automata for enumeration, for specific k

Previous Work for Words

- For words, results by...
  - Burstein: initial results, generating functions
  - Albert, Aldred, Atkinson, Handley, Holton: results for specific 3-letter patterns
  - Brändén, Mansour: automata for enumeration, for specific k
- Note: most work is for *specific* patterns; would like a *universal* technique that works well regardless of pattern or alphabet size

Previous Work for Words

- For words, results by...
  - Burstein: initial results, generating functions
  - Albert, Aldred, Atkinson, Handley, Holton: results for specific 3-letter patterns
  - Brändén, Mansour: automata for enumeration, for specific k
- Note: most work is for *specific* patterns; would like a *universal* technique that works well regardless of pattern or alphabet size
- For permutations, one *universal* technique is Zeilberger and Vatter's enumeration schemes.

Refinement

Main Idea:

- Can't always directly find a recurrence to count $A_{[a_1,\ldots,a_k],Q}$
- Instead, divide and conquer according to pattern formed by first i letters
- Look for recurrences between these subsets of $A_{[a_1,\ldots,a_k],Q}$

Notation

When Q is understood,
$A_{[a_1,\ldots,a_k]}(p_1 \cdots p_l) := \{w \in [k]^{\sum a_i} \mid w \text{ has prefix } p_1 \cdots p_l\}$

Notation

When Q is understood,

$$A_{[a_1,\ldots,a_k]}(p_1 \cdots p_l) := \{w \in [k]^{\sum a_i} \mid w \text{ has prefix } p_1 \cdots p_l\}$$

and, for $1 \leq i_1 \leq \cdots \leq i_l \leq k$,

$$A_{[a_1,\ldots,a_k]}\begin{pmatrix} p_1 \cdots p_l \\ i_1 \cdots i_l \end{pmatrix} := \{w \in [k]^{\sum a_i} \mid w \text{ has prefix } p_1 \cdots p_l \text{ and } i_1, \ldots, i_l \text{ are the first } l \text{ letters of } w\}$$

Refinement Example

We have, $A_{[a_1,\ldots,a_k]}() = A_{[a_1,\ldots,a_k]}(1)$

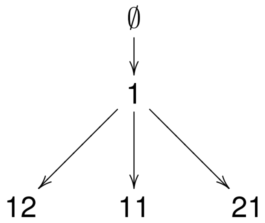
$= A_{[a_1,\ldots,a_k]}(12) \cup A_{[a_1,\ldots,a_k]}(11) \cup A_{[a_1,\ldots,a_k]}(21)$

Refinement Example

We have, $A_{[a_1,\ldots,a_k]}() = A_{[a_1,\ldots,a_k]}(1)$

$= A_{[a_1,\ldots,a_k]}(12) \cup A_{[a_1,\ldots,a_k]}(11) \cup A_{[a_1,\ldots,a_k]}(21)$

or graphically:



Reversibly Deletable

- Given a prefix $p = p_1 \cdots p_t$, position r is reversibly deletable if every possible bad pattern involving p_r implies another bad pattern without p_r.

Reversibly Deletable

- Given a prefix $p = p_1 \cdots p_t$, position r is reversibly deletable if every possible bad pattern involving p_r implies another bad pattern without p_r.
- For example, avoid $q = 123$, and let $p = 21$.

$$21 \cdots a \cdots b$$

Reversibly Deletable

- Given a prefix $p = p_1 \cdots p_t$, position r is reversibly deletable if every possible bad pattern involving p_r implies another bad pattern without p_r.
- For example, avoid $q = 123$, and let $p = 21$.

$$21 \cdots a \cdots b$$
$$21 \cdots a \cdots b$$

Reversibly Deletable

- Given a prefix $p = p_1 \cdots p_t$, position r is reversibly deletable if every possible bad pattern involving p_r implies another bad pattern without p_r.
- For example, avoid $q = 123$, and let $p = 21$.

$$21 \cdots a \cdots b$$

$$21 \cdots a \cdots b$$

$p_1 = 2$ is *reversibly deletable* for $q = 123$, $p = 21$.

Reversibly Deletable

- There is always a natural embedding

$$A_{[a_1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots p_l \\ i_1 \cdots i_l \end{pmatrix} \to A_{[a_1,\ldots,a_j-1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots \hat{p}_r \cdots p_l \\ i_1 \cdots \hat{j} \cdots i_l \end{pmatrix}$$

Reversibly Deletable

- There is always a natural embedding
$$A_{[a_1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots p_l \\ i_1 \cdots i_l \end{pmatrix} \to A_{[a_1,\ldots,a_j-1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots \hat{p}_r \cdots p_l \\ i_1 \cdots \hat{j} \cdots i_l \end{pmatrix}$$
- If p_r is reversibly deletable, and the role of p_r is played by letter j, then
$$\left|A_{[a_1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots p_l \\ i_1 \cdots i_l \end{pmatrix}\right| = \left|A_{[a_1,\ldots,a_j-1,\ldots,a_n]}\begin{pmatrix} p_1 \cdots \hat{p}_r \cdots p_l \\ i_1 \cdots \hat{j} \cdots i_l \end{pmatrix}\right|.$$

Reversibly Deletable Example

For $Q = \{123\}$, we have,

$$\left| A_{[a_1,\dots,a_k]} \binom{21}{ij} \right| = \left| A_{[a_1,\dots,a_j-1,\dots a_k]} \binom{1}{i} \right|$$

$$\left| A_{[a_1,\dots,a_k]} \binom{11}{ij} \right| = \left| A_{[a_1,\dots,a_i-1,\dots a_k]} \binom{1}{j} \right|$$

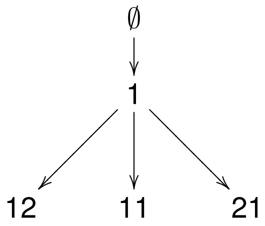
Reversibly Deletable Example

For $Q = \{123\}$, we have,

$$\left| A_{[a_1,\ldots,a_k]} \begin{pmatrix} 21 \\ ij \end{pmatrix} \right| = \left| A_{[a_1,\ldots,a_j-1,\ldots a_k]} \begin{pmatrix} 1 \\ i \end{pmatrix} \right|$$

$$\left| A_{[a_1,\ldots,a_k]} \begin{pmatrix} 11 \\ ij \end{pmatrix} \right| = \left| A_{[a_1,\ldots,a_i-1,\ldots a_k]} \begin{pmatrix} 1 \\ j \end{pmatrix} \right|$$

or graphically:



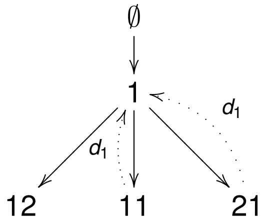
Reversibly Deletable Example

For $Q = \{123\}$, we have,

$$\left| A_{[a_1,\ldots,a_k]} \binom{21}{ij} \right| = \left| A_{[a_1,\ldots,a_j-1,\ldots a_k]} \binom{1}{i} \right|$$

$$\left| A_{[a_1,\ldots,a_k]} \binom{11}{ij} \right| = \left| A_{[a_1,\ldots,a_i-1,\ldots a_k]} \binom{1}{j} \right|$$

or graphically:



Gap Vectors

Consider words that avoid $q = 123$ and begin with prefix $p = 12$

sorted prefix:	1 2
letters involved in prefix:	$i \;\; j$
vector:	$\langle a, b, c \rangle$

Gap Vectors

Consider words that avoid $q = 123$ and begin with prefix $p = 12$

sorted prefix: 1 2
letters involved in prefix: i j
vector: $\langle a, b, c \rangle$

sorted word: $\underbrace{\cdots}_{\geq a}\, i\, \underbrace{\cdots}_{\geq b-1}\, j\, \underbrace{\cdots}_{\geq c}$ ($b = 0$ denotes a repeated letter)

Gap Vectors

Consider words that avoid $q = 123$ and begin with prefix $p = 12$

sorted prefix: 1 2
letters involved in prefix: $i \; j$
vector: $\langle a, b, c \rangle$

sorted word: $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b-1} j \underbrace{\cdots}_{\geq c}$ ($b = 0$ denotes a repeated letter)

v is a gap vector for p if there are no words avoiding q with prefix p and spacing v.

Gap Vectors

Consider words that avoid $q = 123$ and begin with prefix $p = 12$

sorted prefix: 1 2
letters involved in prefix: i j
vector: $\langle a, b, c \rangle$

sorted word: $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b-1} j \underbrace{\cdots}_{\geq c}$ ($b = 0$ denotes a repeated letter)

v is a gap vector for p if there are no words avoiding q with prefix p and spacing v.

e.g. $v = \langle 0, 1, 1 \rangle$ is a gap vector for $q = 123$, $p = 12$.

Gap Vector Example

For $Q = \{123\}$, we have,

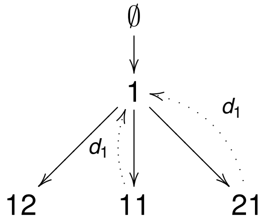
$$\left|A_{[a_1,\dots,a_k]}\begin{pmatrix}12\\ ij\end{pmatrix}\right| = \left|A_{[a_1,\dots,a_k]}\begin{pmatrix}12\\ ik\end{pmatrix}\right| = \left|A_{[a_1,\dots,a_k-1]}\begin{pmatrix}1\\ i\end{pmatrix}\right|$$

Gap Vector Example

For $Q = \{123\}$, we have,

$$\left|A_{[a_1,\ldots,a_k]}\begin{pmatrix}12\\ij\end{pmatrix}\right| = \left|A_{[a_1,\ldots,a_k]}\begin{pmatrix}12\\ik\end{pmatrix}\right| = \left|A_{[a_1,\ldots,a_k-1]}\begin{pmatrix}1\\i\end{pmatrix}\right|$$

or graphically:

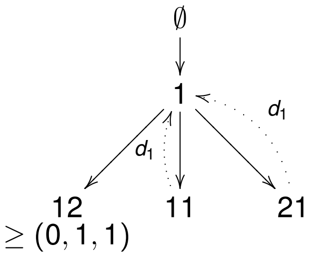


Gap Vector Example

For $Q = \{123\}$, we have,

$$\left|A_{[a_1,\dots,a_k]}\binom{12}{ij}\right| = \left|A_{[a_1,\dots,a_k]}\binom{12}{ik}\right| = \left|A_{[a_1,\dots,a_k-1]}\binom{1}{i}\right|$$

or graphically:

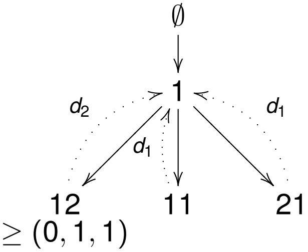


Gap Vector Example

For $Q = \{123\}$, we have,

$$\left|A_{[a_1,\ldots,a_k]}\binom{12}{ij}\right| = \left|A_{[a_1,\ldots,a_k]}\binom{12}{ik}\right| = \left|A_{[a_1,\ldots,a_k-1]}\binom{1}{i}\right|$$

or graphically:



Enumeration Scheme

An *enumeration scheme* is a set of triples $[p_i, R_i, G_i]$ such that for each triple

- p_i is a reduced word of length n
- R_i a subset of $\{1, \ldots, n\}$
- G_i is a set of vectors of length $n + 1$
  and
- either R_i is non-empty or all refinements of p_i are also in the scheme.

Enumeration Scheme

An *enumeration scheme* is a set of triples $[p_i, R_i, G_i]$ such that for each triple

- p_i is a reduced word of length n (prefix)
- R_i a subset of $\{1, \ldots, n\}$ (reversibly deletable positions)
- G_i is a set of vectors of length $n + 1$ (gap vectors) and
- either R_i is non-empty or all refinements of p_i are also in the scheme.

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:
$S = \{[\emptyset, \emptyset, \emptyset]\}$

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:
$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset]\}$

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:

$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset], [12, R_{12}, G_{12}], [11, R_{11}, G_{11}], [21, R_{21}, G_{21}]\}$

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:
$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset], [12, R_{12}, G_{12}], [11, \{1\}, \emptyset], [21, \{1\}, \emptyset]\}$

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:

$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset], [12, R_{12}, \{<0, 1, 1>\}], [11, \{1\}, \emptyset], [21, \{1\}, \emptyset]\}$

Enumeration Scheme Example

For the pattern $q = 123$, we have constructed the following scheme:
$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset], [12, \{2\}, \{< 0, 1, 1 >\}], [11, \{1\}, \emptyset], [21, \{1\}, \emptyset]\}$

Enumeration Scheme Example

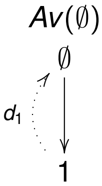
For the pattern $q = 123$, we have constructed the following scheme:
$S = \{[\emptyset, \emptyset, \emptyset], [1, \emptyset, \emptyset], [12, \{2\}, \{< 0, 1, 1 >\}], [11, \{1\}, \emptyset],$
$[21, \{1\}, \emptyset]\}$

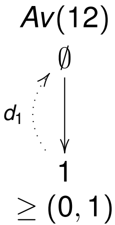
Enumeration Schemes

- Refinements
- Reversibly deletable elements
- Gap vectors

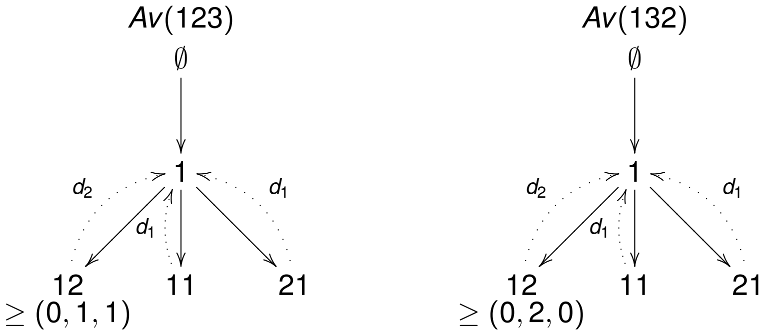
can all be found completely automatically, so we have an algorithm to compute an enumeration schemes for words.

The Simplest Examples

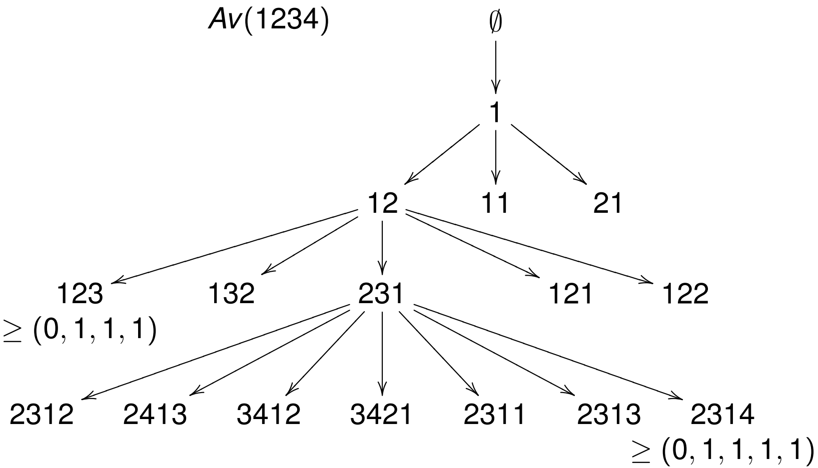




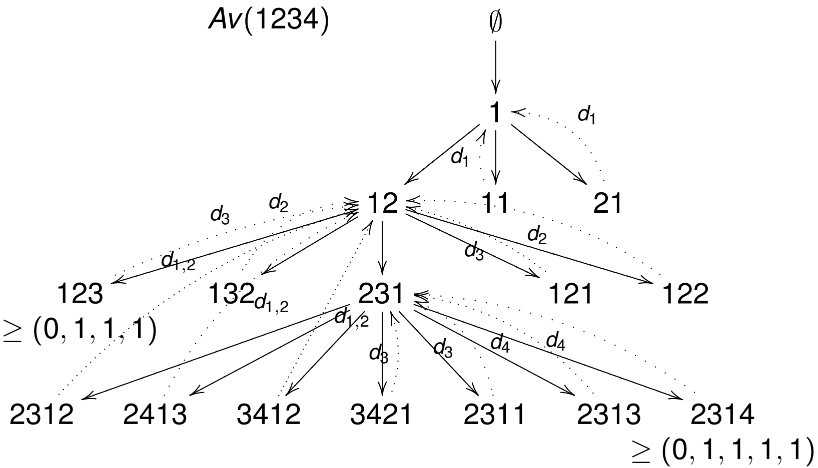
Isomorphic Prefix Schemes



Another Example



Another Example



Wilf Equivalence

Given $w \in [k]^n$, $w = w_1 \cdots w_n$, let

- $w^r = w_n \cdots w_1$ (reverse)
- $w^c = y_1 \cdots y_n$ such that $y_i = k + 1 - w_i$ (complement)

Wilf Equivalence

Given $w \in [k]^n$, $w = w_1 \cdots w_n$, let

- $w^r = w_n \cdots w_1$ (reverse)
- $w^c = y_1 \cdots y_n$ such that $y_i = k + 1 - w_i$ (complement)

Then

- $\left|A_{[a_1,\ldots,a_k],q}\right| = \left|A_{[a_1,\ldots,a_k],q^r}\right|$
- $\left|A_{[a_1,\ldots,a_k],q}\right| = \left|A_{[a_1,\ldots,a_k],q^c}\right|$

Wilf Equivalence

Given $w \in [k]^n$, $w = w_1 \cdots w_n$, let

- $w^r = w_n \cdots w_1$ (reverse)
- $w^c = y_1 \cdots y_n$ such that $y_i = k + 1 - w_i$ (complement)

Then

- $\left|A_{[a_1,\ldots,a_k],q}\right| = \left|A_{[a_1,\ldots,a_k],q^r}\right|$
- $\left|A_{[a_1,\ldots,a_k],q}\right| = \left|A_{[a_1,\ldots,a_k],q^c}\right|$

It is only necessary to find an enumeration scheme for one member of each Wilf equivalence class of patterns.

Statistics

Success rate for words avoiding *permutation* patterns:

Pattern Lengths	**Number of Wilf Classes with an Enumeration Scheme**
[2]	1/1 (100%)
[2,3]	1/1 (100%)
[2,4]	1/1 (100%)
[3]	2/2 (100%)
[3,3]	6/6 (100%)
[3,3,3]	6/6 (100%)
[3,3,3,3]	6/6 (100%)
[3,3,3,3,3]	2/2 (100%)

Statistics

Success rate for words avoiding *permutation* patterns:

Pattern Lengths	Number of Wilf Classes with an Enumeration Scheme
[4]	2/8 (25%)
[3,4]	9/24 (37.5%)
[3,3,4]	27/31 (87.1%)
[3,3,3,4]	20/20 (100%)
[3,3,3,3,4]	6/6 (100%)
[3,3,3,3,3,4]	1/1 (100%)
[4,4]	?/84 (in process)
[3,4,4]	38/146 (26%)
[3,3,4,4]	89/103 (86.4%)
[3,3,3,4,4]	29/29 (100%)
[3,3,3,3,4,4]	3/3 (100%)

Summary

- There are few techniques to count many classes of pattern-avoiding words.

Summary

- There are few techniques to count many classes of pattern-avoiding words.
- Extending Zeilberger's and Vatter's schemes gives a good success rate for words avoiding permutations.

Summary

- There are few techniques to count many classes of pattern-avoiding words.
- Extending Zeilberger's and Vatter's schemes gives a good success rate for words avoiding permutations.

- Future work
  - Find other general techniques for enumerating classes of pattern-avoiding words.

Summary

- There are few techniques to count many classes of pattern-avoiding words.
- Extending Zeilberger's and Vatter's schemes gives a good success rate for words avoiding permutations.

- Future work
    - Find other general techniques for enumerating classes of pattern-avoiding words.
    - Convert enumeration schemes to generating functions or closed forms.

Summary

- There are few techniques to count many classes of pattern-avoiding words.
- Extending Zeilberger's and Vatter's schemes gives a good success rate for words avoiding permutations.
- Future work
  - Find other general techniques for enumerating classes of pattern-avoiding words.
  - Convert enumeration schemes to generating functions or closed forms.
  - Extend enumeration schemes to count other pattern-avoiding objects.