Schemes for Pattern-Avoiding Words

Lara Pudwell

Rutgers University

Permutation Patterns 2007

Outline

1. Introduction/History
   - Pattern Avoidance in Words
   - Previous Work
2. Prefix Schemes for Words
   - Definitions
   - Examples
   - Success Rate
3. Other Schemes for Words
   - Schemes for Monotone Patterns

Outline

Reduction

- Given a string of letters $p = p_1 ... p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.

Reduction

- Given a string of letters $p = p_1...p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of `2674425` is

Reduction

- Given a string of letters $p = p_1 \ldots p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of `2674425` is `1`● ● ●●`1`●.

Reduction

- Given a string of letters $p = p_1 ... p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of 2674425 is 1••221•.

Reduction

- Given a string of letters $p = p_1 ... p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of `2674425` is `1••2213`.

Reduction

- Given a string of letters $p = p_1 ... p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of `2674425` is `14•2213`.

Reduction

- Given a string of letters $p = p_1 ... p_n$, the reduction of p is the string obtained by replacing the i^{th} smallest letter(s) of p with i.
- For example, the reduction of `2674425` is `1452213`.

Pattern Avoidance in Words

- Given $w \in [k]^n$, and $p = p_1 \dots p_m$, w contains p if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \dots w_{i_m}$ reduces to p.
- Otherwise w avoids p.

Pattern Avoidance in Words

- Given $w \in [k]^n$, and $p = p_1 \dots p_m$, w contains p if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \dots w_{i_m}$ reduces to p.
- Otherwise w avoids p.
- e.g. 1452213 contains 312 (1452213)
  1452213 avoids 212.

Pattern Avoidance in Words

- Given $w \in [k]^n$, and $p = p_1 \ldots p_m$, w contains p if there is $1 \leq i_1 < \cdots < i_m \leq n$ so that $w_{i_1} \ldots w_{i_m}$ reduces to p.
- Otherwise w avoids p.
- e.g. 1452213 contains 312 (1452213)
  1452213 avoids 212.
- Want to count $A_{[a_1,\ldots,a_k]}(\{Q\}) :=$
  $\{w \in [k]^{\sum a_i} \mid w \text{ has } a_i\ i\text{'s}, w \text{ avoids } q \text{ for every } q \in Q\}$

Outline

1. Introduction/History
   - Pattern Avoidance in Words
   - Previous Work
2. Prefix Schemes for Words
   - Definitions
   - Examples
   - Success Rate
3. Other Schemes for Words
   - Schemes for Monotone Patterns

Previous Work for Words

- Results by...
  - Burstein: initial results, generating functions

Previous Work for Words

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

Previous Work 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

- 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

- 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.

Outline

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

Notation

When Q is understood,

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

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

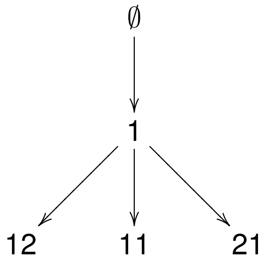
$$A_{[a_1,\ldots,a_k]}\begin{pmatrix} p_1 \ldots p_l \\ i_1 \ldots i_l \end{pmatrix} := \{w \in [k]^{\sum a_i} \mid \text{ } w \text{ has prefix } p_1 \ldots p_l \text{ and } i_1, \ldots i_l \text{ are the first } l \text{ letters of } w\}$$

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:



Refinement

Main consideration:

- for permutations, only permutations appear as prefixes
  e.g. refinements of 1 are 12 and 21

Refinement

Main consideration:

- for permutations, only permutations appear as prefixes
  e.g. refinements of 1 are 12 and 21
- for words, there are many more prefixes
  e.g. refinements of 1 are 12, 21, and 11

Reversibly Deletable

- Given a prefix $p = p_1 \ldots p_t$, 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 $p = p_1 \ldots p_t$, 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 $p = 21\ldots$

21...a...b

Reversibly Deletable

- Given a prefix $p = p_1 \ldots p_t$, 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 $p = 21\ldots$

21...a...b
21...a...b

Reversibly Deletable

- Given a prefix $p = p_1 \ldots p_t$, 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...a...b

21...a...b

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

Reversibly Deletable

- There is always a natural embedding
$$A_{[a_1,\ldots,a_n]}\begin{pmatrix}p_1\ldots p_l\\ i_1\ldots i_l\end{pmatrix}\rightarrow A_{[a_1,\ldots a_j-1,\ldots a_n]}\begin{pmatrix}p_1\ldots\hat{p}_r\ldots p_l\\ i_1\ldots\hat{j}\ldots i_l\end{pmatrix}$$

Reversibly Deletable

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

Reversibly Deletable Example

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

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

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

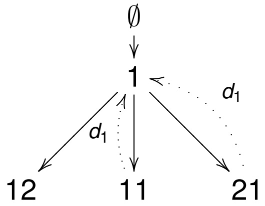
Reversibly Deletable Example

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

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

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

or graphically:



Reversibly Deletable

Main consideration:

- for permutations, reversibly deletable letters can always be removed together

Reversibly Deletable

Main consideration:

- for permutations, reversibly deletable letters can always be removed together
- for words, two letters can be reversibly deletable separately but not together

  e.g. $q = 123$, $p = 11$

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: <a, b, c>

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: <a, b, c>

sorted word: $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$

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: <a, b, c>

sorted word: $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$

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: <a, b, c>

sorted word: $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$

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

e.g. $v =< 0, 0, 1 >$ is a gap vector for $q = 123$, $p = 12$.

Gap Vector Example

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

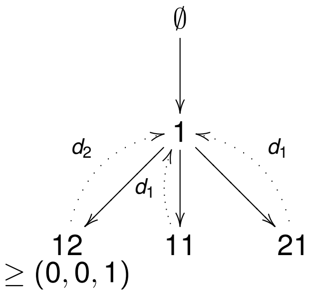
$$A_{[a_1,\ldots,a_k]}\binom{12}{ij} = A_{[a_1,\ldots,a_k]}\binom{12}{ik} = A_{[a_1,\ldots,a_k-1]}\binom{1}{i}$$

Gap Vector Example

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

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

or graphically:



Gap Vectors

Main consideration:

- for permutations, <a,b,c> means $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$

Gap Vectors

Main consideration:

- for permutations, <a,b,c> means $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$
- for words, <a,b+1,c> means $\underbrace{\cdots}_{\geq a} i \underbrace{\cdots}_{\geq b} j \underbrace{\cdots}_{\geq c}$

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.

Implementation

Maple package `mVATTER` has the following functions

- `SchemeF`: input: set of patterns, maximum scheme depth (also faster version with maximum gap weight as input)
  output: scheme

Implementation

Maple package `mVATTER` has the following functions

- `SchemeF`: input: set of patterns, maximum scheme depth (also faster version with maximum gap weight as input)
  output: scheme
- `MiklosA, MiklosTot`: input: scheme, alphabet
  output: number of words obeying input

Implementation

Maple package `mVATTER` has the following functions

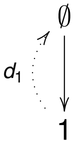
- `SchemeF`: input: set of patterns, maximum scheme depth
  (also faster version with maximum gap weight as input)
  output: scheme
- `MiklosA, MiklosTot`: input: scheme, alphabet
  output: number of words obeying input
- `SipurF`: input: list of pattern lengths, max scheme depth
  output: scheme and statistics for every
  equivalence class of patterns with lengths in list

Outline

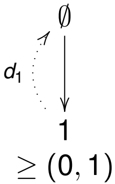
1. Introduction/History
   - Pattern Avoidance in Words
   - Previous Work
2. Prefix Schemes for Words
   - Definitions
   - Examples
   - Success Rate
3. Other Schemes for Words
   - Schemes for Monotone Patterns

The Simplest Examples

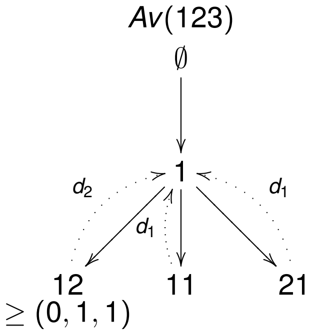
$Av(\emptyset)$

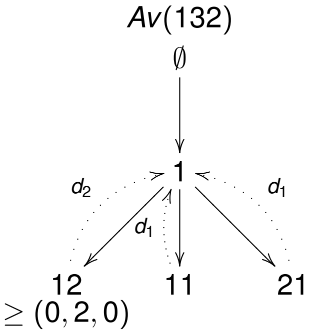


$Av(12)$

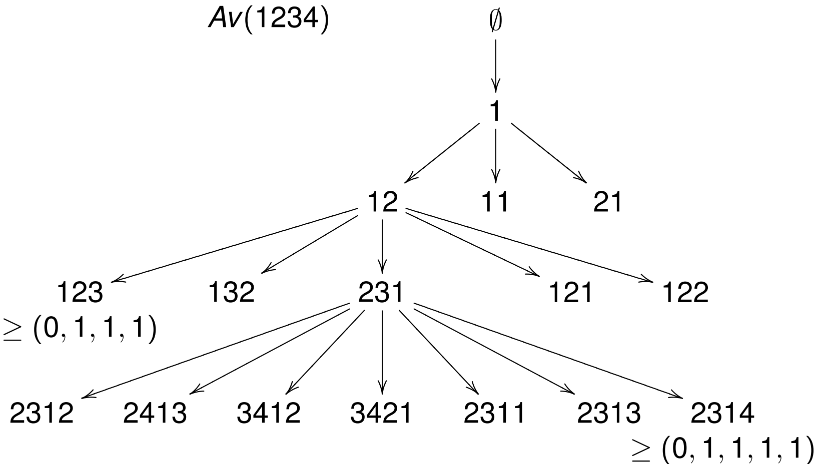


Isomorphic Prefix Schemes

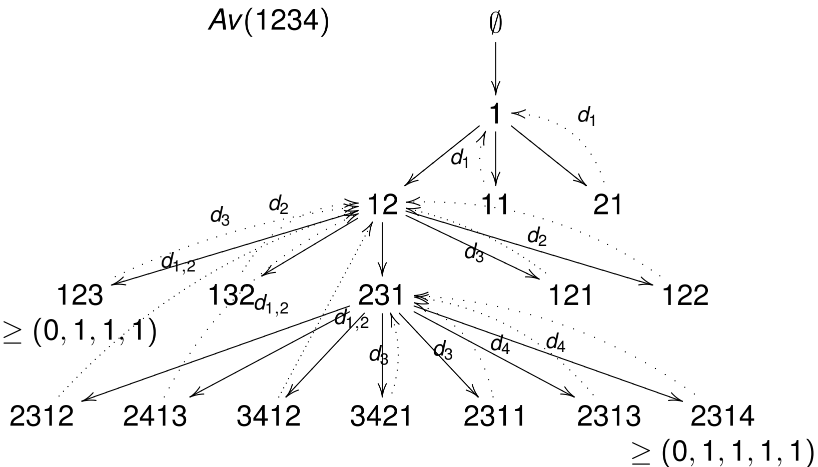




Another Example



Another Example



Outline

1. Introduction/History
   - Pattern Avoidance in Words
   - Previous Work
2. Prefix Schemes for Words
   - Definitions
   - Examples
   - Success Rate
3. Other Schemes for Words
   - Schemes for Monotone Patterns

Statistics

- success rate is bounded above by success rate for permutation schemes

Statistics

- success rate is bounded above by success rate for permutation schemes

Pattern Lengths	Permutations	Words
[2]	1/1 (100%)	1/1 (100%)
[2,3]	1/1 (100%)	1/1 (100%)
[2,4]	1/1 (100%)	1/1 (100%)
[3]	2/2 (100%)	2/2 (100%)
[3,3]	5/5 (100%)	6/6 (100%)
[3,3,3]	5/5 (100%)	6/6 (100%)
[3,3,3,3]	5/5 (100%)	6/6 (100%)
[3,3,3,3,3]	2/2 (100%)	2/2 (100%)

Statistics

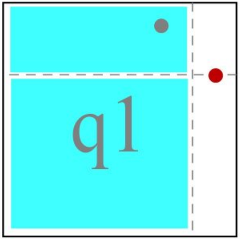
Pattern Lengths	Permutations	Words
[4]	2/7 (28.6%)	2/8 (25%)
[3,4]	17/18 (94.4%)	9/24 (37.5%)
[3,3,4]	23/23 (100%)	27/31 (87.1%)
[3,3,3,4]	16/16 (100%)	20/20 (100%)
[3,3,3,3,4]	6/6 (100%)	6/6 (100%)
[3,3,3,3,3,4]	1/1 (100%)	1/1 (100%)
[4,4]	29/56 (51.8%)	?/84 (in process)
[3,4,4]	92/92 (100%)	38/146 (26%)
[3,3,4,4]	68/68 (100%)	89/103 (86.4%)
[3,3,3,4,4]	23/23 (100%)	29/29 (100%)
[3,3,3,3,4,4]	3/3 (100%)	3/3 (100%)

Avoiding a Pattern With Repeated Letters

- only works to avoid *permutation* patterns
- Let $q = q_1 l q_2 l q_3$, where l is the first repeated letter in q.

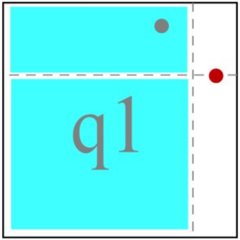
Avoiding a Pattern With Repeated Letters

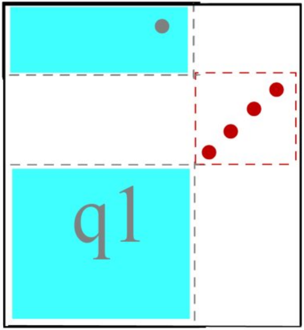
- only works to avoid *permutation* patterns
- Let $q = q_1 l q_2 l q_3$, where l is the first repeated letter in q.



Avoiding a Pattern With Repeated Letters

- only works to avoid *permutation* patterns
- Let $q = q_1 l q_2 l q_3$, where l is the first repeated letter in q.





Another Direction

- Zeilberger's schemes: patterns formed by the *first i* letters of words
  (refinement by adding one letter at a time)
  drawback: only works for permutation-avoiding words

Another Direction

- Zeilberger's schemes: patterns formed by the *first i* letters of words
  (refinement by adding one letter at a time)
  drawback: only works for permutation-avoiding words
- Vatter's schemes: patterns formed by the *smallest i* letters of words
  drawback: $1 \to 11 \to 111 \to \ldots$ is a problem

Another Direction

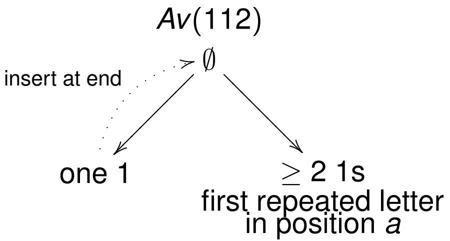
- Zeilberger's schemes: patterns formed by the *first i* letters of words
  (refinement by adding one letter at a time)
  drawback: only works for permutation-avoiding words
- Vatter's schemes: patterns formed by the *smallest i* letters of words
  drawback: $1 \to 11 \to 111 \to \ldots$ is a problem
- Solution: following Vatter, consider the patterns formed by the *smallest* letters of words
  BUT refine by adding *all* copies of a letter simultaneously

Outline

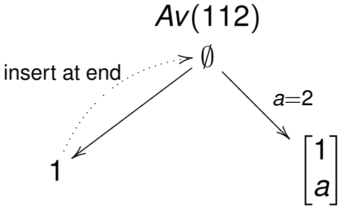
An Example

$$Av(112)$$

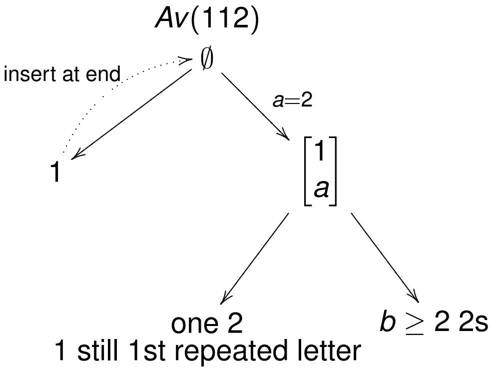
An Example


Av(112)
∅
insert at end
one 1
≥ 2 1s
first repeated letter
in position a

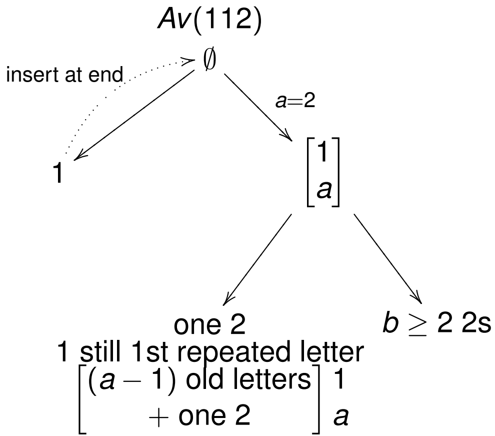
An Example



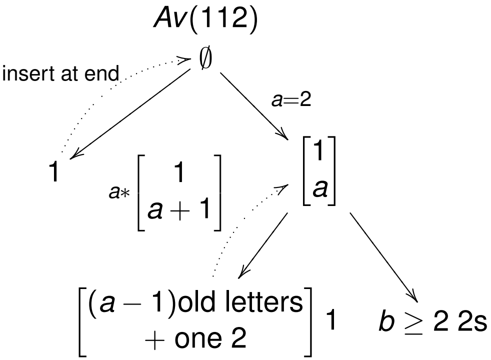
An Example


$Av(112)$
$\emptyset$
insert at end
a=2
1
$\begin{bmatrix} 1 \\ a \end{bmatrix}$
one 2
1 still 1st repeated letter
$b \geq 2$ 2s

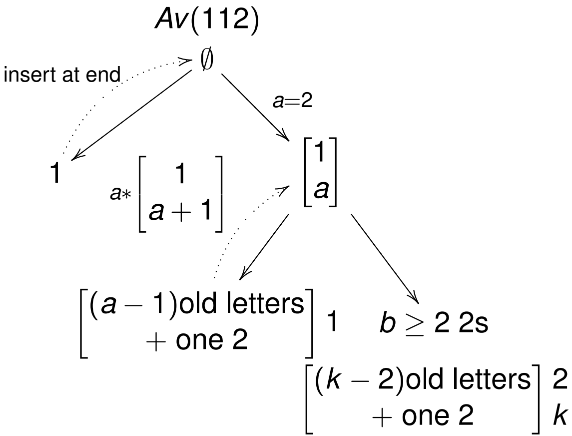
An Example


$Av(112)$
$\emptyset$
insert at end
a=2
1
$\begin{bmatrix} 1 \\ a \end{bmatrix}$
one 2
1 still 1st repeated letter
$\begin{bmatrix} (a-1) \text{ old letters} \\ + \text{ one } 2 \end{bmatrix} \begin{matrix} 1 \\ a \end{matrix}$
$b \geq 2$ 2s

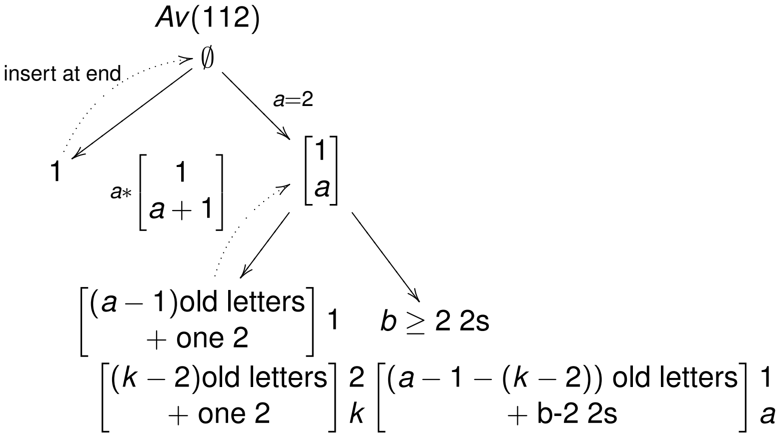
An Example



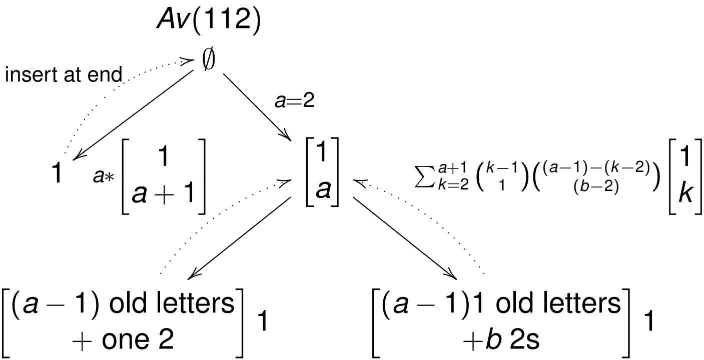
An Example


$Av(112)$
$\emptyset$
insert at end
$a=2$
1
$a* \begin{bmatrix} 1 \\ a+1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ a \end{bmatrix}$
$\begin{bmatrix} (a-1)\text{old letters} \\ +\text{ one } 2 \end{bmatrix} 1$
$b \geq 2$ 2s
$\begin{bmatrix} (k-2)\text{old letters} \\ +\text{ one } 2 \end{bmatrix} \begin{matrix} 2 \\ k \end{matrix}$

An Example


$Av(112)$
$\emptyset$
insert at end
a=2
1
$a* \begin{bmatrix} 1 \\ a+1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ a \end{bmatrix}$
$\begin{bmatrix} (a-1)\text{old letters} \\ +\text{ one } 2 \end{bmatrix} 1$
$b \geq 2$ 2s
$\begin{bmatrix} (k-2)\text{old letters} \\ +\text{ one } 2 \end{bmatrix} \begin{matrix} 2 \\ k \end{matrix} \begin{bmatrix} (a-1-(k-2))\text{ old letters} \\ +\text{ b-2 2s} \end{bmatrix} \begin{matrix} 1 \\ a \end{matrix}$

An Example


$Av(112)$
$\emptyset$
insert at end
$a=2$
$1 \quad a * \begin{bmatrix} 1 \\ a+1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ a \end{bmatrix}$
$\sum_{k=2}^{a+1} \binom{k-1}{1} \binom{(a-1)-(k-2)}{(b-2)} \begin{bmatrix} 1 \\ k \end{bmatrix}$
$\begin{bmatrix} (a-1) \text{ old letters} \\ + \text{ one } 2 \end{bmatrix} 1$
$\begin{bmatrix} (a-1)1 \text{ old letters} \\ +b \text{ } 2\text{s} \end{bmatrix} 1$

An Example

Let

- $A_{[a_1,\ldots,a_k]} := |\{w \in [k]^{\sum a_i} | w \text{ has } a_i\ i\text{s}, w \text{ avoids } 112\}|$

An Example

Let

- $A_{[a_1,\ldots,a_k]} := |\{w \in [k]^{\sum a_i} | w \text{ has } a_i \ i\text{s}, w \text{ avoids } 112\}|$
- $B^{(i)}_{[a_1,\ldots,a_k]} := |\{w \in A_{[a_1,\ldots,a_k]}|$
  w's first repeated letter is in position i}|

An Example

We now have:

$$A_{[a_1,\ldots,a_k]} = \begin{cases} 1 & k = 1 \\ \binom{a_2+\cdots+a_k+1}{1} A_{[a_2,\ldots,a_k]} & k > 1, a_1 = 1 \\ B^{(2)}_{[a_2,\ldots,a_k]} & k > 1, a_1 > 1 \end{cases}$$

An Example

We now have:

$$A_{[a_1,\ldots,a_k]} = \begin{cases} 1 & k = 1 \\ \binom{a_2+\cdots+a_k+1}{1} A_{[a_2,\ldots,a_k]} & k > 1, a_1 = 1 \\ B^{(2)}_{[a_2,\ldots,a_k]} & k > 1, a_1 > 1 \end{cases}$$

$$B^{(i)}_{[a_1,\ldots,a_k]} = \begin{cases} \binom{i-1+a_1}{a_1} & k = 1 \\ i * B^{(i+1)}_{[a_2,\ldots,a_k]} & a_1 = 1 \\ \sum_{k=2}^{i+1} (k-1) \binom{(i-1)-(k-2)+(a_1-2)}{a_1-2} B^{(k)}_{[a_2,\ldots,a_k]} & a_1 > 1 \end{cases}$$

An Example

We now have:

$$A_{[a_1,\dots,a_k]} = \begin{cases} 1 & k = 1 \\ \binom{a_2+\cdots+a_k+1}{1} A_{[a_2,\dots,a_k]} & k > 1, a_1 = 1 \\ B^{(2)}_{[a_2,\dots,a_k]} & k > 1, a_1 > 1 \end{cases}$$

$$B^{(i)}_{[a_1,\dots,a_k]} = \begin{cases} \binom{i-1+a_1}{a_1} & k = 1 \\ i * B^{(i+1)}_{[a_2,\dots,a_k]} & a_1 = 1 \\ \sum_{k=2}^{i+1} (k-1) \binom{(i-1)-(k-2)+(a_1-2)}{a_1-2} B^{(k)}_{[a_2,\dots,a_k]} & a_1 > 1 \end{cases}$$

$$A_{[a_1,\dots,a_k]} = \prod_{i=2}^{k} (a_j + \cdots a_k + 1)$$

.

Non-prefix schemes

- This example can be generalized to find a scheme for words avoiding *any* monotone pattern.

Non-prefix schemes

- This example can be generalized to find a scheme for words avoiding *any* monotone pattern.
- Currently exploring extensions to other types of patterns.

Summary

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

Summary

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

Summary

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

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

Summary

- There are few techniques to count large classes of pattern-avoiding words.
- Extending Zeilberger's and Vatter's schemes gives a good success rate for words avoiding permutations and for words avoiding monotone patterns.
- Future work
  - Find other general techniques for enumerating classes of permutation-avoiding words.
  - Simplify schemes to compute more data more quickly.

Summary

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

- Future work
  - Find other general techniques for enumerating classes of permutation-avoiding words.
  - Simplify schemes to compute more data more quickly.
  - Convert concrete enumeration schemes to closed forms.

References

- M. Albert, R. Aldred, M.D. Atkinson, C. Handley, D. Holton, *Permutations of a multiset avoiding permutations of length 3*, Europ. J. Combin. **22**, 1021-1031 (2001).
- P. Branden, T. Mansour, *Finite automata and pattern avoidance in words*, Joural Combinatorial Theory Series A **110:1**, 127-145 (2005).
- Alexander Burstein, *Enumeration of Words with Forbidden Patterns*, Ph.D. Thesis, University of Pennsylvania, 1998.
- Vince Vatter, *Enumeration Schemes for Restricted Permutations*, Combinatorics, Probability, and Computing, to appear.
- Doron Zeilberger, *Enumeration Schemes, and More Importantly, Their Automatic Generation*, Annals of Combinatorics **2**, 185-195 (1998).
- Doron Zeilberger, *On Vince Vatter's Brilliant Extension of Doron Zeilberger's Enumeration Schemes for Herb Wilf's Classes*, The Personal Journal of Ekhad and Zeilberger, 2006. `http://www.math.rutgers.edu/~ zeilberg/pj.html`.