Avoiding abelian and additive powers in rich words

Lucas Mol

THOMPSON RIVERS UNIVERSITY

Joint work with Jonathan Andrade

University of Victoria

(And some earlier work with James Currie, Narad Rampersad, and Jeffrey Shallit)

One World Combinatorics on Words Seminar
January 28, 2025

Plan

Introduction

Our constructions

The template method

Three types of powers

Three types of powers

- An (ordinary) square is a nonempty word of the form xx.
  - E.g., `murmur`, `hotshots`

Three types of powers

- An (ordinary) square is a nonempty word of the form xx.
  - E.g., `murmur`, `hotshots`

- An abelian square is a nonempty word of the form xx', where x' is an *anagram* of x.
  - E.g., `reappear`, `intestines`

Three types of powers

- An (ordinary) square is a nonempty word of the form xx.
  - E.g., `murmur`, `hotshots`

- An abelian square is a nonempty word of the form xx', where x' is an *anagram* of x.
  - E.g., `reappear`, `intestines`

- An additive square is a nonempty word of the form xx', where x and x' have the same length and the same sum.
  - E.g., `0312`, `210201`

Three types of powers

- An (ordinary) square is a nonempty word of the form xx.
  - E.g., `murmur`, `hotshots`
- An abelian square is a nonempty word of the form xx', where x' is an *anagram* of x.
  - E.g., `reappear`, `intestines`
- An additive square is a nonempty word of the form xx', where x and x' have the same length and the same sum.
  - E.g., 0312, 210201
- Ordinary, abelian, and additive k-powers are defined analogously for $k \geq 2$.
  - E.g., 011 101 110 011 is an abelian/additive 4-power.

Avoiding abelian powers

Avoiding abelian powers

Theorem (Dekking 1979): Abelian 4-powers are avoidable over $\{0, 1\}$.

Avoiding abelian powers

Theorem (Dekking 1979): Abelian 4-powers are avoidable over $\{0, 1\}$.

Theorem (Dekking 1979): Abelian cubes are avoidable over $\{0, 1, 2\}$.

Avoiding abelian powers

Theorem (Dekking 1979): Abelian 4-powers are avoidable over $\{0, 1\}$.

Theorem (Dekking 1979): Abelian cubes are avoidable over $\{0, 1, 2\}$.

Theorem (Keränen 1992): Abelian squares are avoidable over $\{0, 1, 2, 3\}$.

Avoiding abelian powers

Theorem (Dekking 1979): Abelian 4-powers are avoidable over $\{0, 1\}$.

Theorem (Dekking 1979): Abelian cubes are avoidable over $\{0, 1, 2\}$.

Theorem (Keränen 1992): Abelian squares are avoidable over $\{0, 1, 2, 3\}$.

- All three alphabet sizes are smallest possible.

Avoiding additive powers

Avoiding additive powers

Theorem (Dekking 1979): Additive 4-powers are avoidable over $\{0, 1\}$.

Avoiding additive powers

Theorem (Dekking 1979): Additive 4-powers are avoidable over $\{0, 1\}$.

Theorem (Cassaigne, Currie, Schaeffer, and Shallit 2014): Additive cubes are avoidable over $\{0, 1, 3, 4\}$.

Avoiding additive powers

Theorem (Dekking 1979): Additive 4-powers are avoidable over $\{0, 1\}$.

Theorem (Cassaigne, Currie, Schaeffer, and Shallit 2014): Additive cubes are avoidable over $\{0, 1, 3, 4\}$.

Theorem (Rao 2015): Additive cubes are avoidable over several subsets of $\mathbb{Z}$ of size 3.

Avoiding additive powers

Theorem (Dekking 1979): Additive 4-powers are avoidable over $\{0, 1\}$.

Theorem (Cassaigne, Currie, Schaeffer, and Shallit 2014): Additive cubes are avoidable over $\{0, 1, 3, 4\}$.

Theorem (Rao 2015): Additive cubes are avoidable over several subsets of $\mathbb{Z}$ of size 3.

Open Problem: Are additive squares avoidable over some finite subset of $\mathbb{Z}$?

Rich words

We consider similar problems for rich words.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word 01101 contains the palindromes 0,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`, and `101`,

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`, and `101`, so it is rich.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`, and `101`, so it is rich.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`, and `101`, so it is rich.
  - The word `0120` contains only the palindromes `0`, `1`, and `2`, so it is not rich.

Rich words

We consider similar problems for rich words.

- A palindrome is a finite word that reads the same forwards and backwards.
  - Examples: `01010`, `kayak`, `saippuakivikauppias`

Theorem (Droubay, Justin, and Pirillo 2001): Every word of length n contains at most n distinct nonempty palindromes.

- A word of length n is called rich if it contains n distinct nonempty palindromes.
  - The word `01101` contains the palindromes `0`, `1`, `11`, `0110`, and `101`, so it is rich.
  - The word `0120` contains only the palindromes `0`, `1`, and `2`, so it is not rich.
- An infinite word is called rich if all of its finite factors are rich.

Avoiding abelian and additive powers in rich words

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Theorem (Andrade and Mol 2025+): There is an infinite additive 4-power-free rich word over $\{0, 1, 2\}$.

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Theorem (Andrade and Mol 2025+): There is an infinite additive 4-power-free rich word over $\{0, 1, 2\}$.

- The alphabet sizes are smallest possible, even for abelian powers.

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Theorem (Andrade and Mol 2025+): There is an infinite additive 4-power-free rich word over $\{0, 1, 2\}$.

- The alphabet sizes are smallest possible, even for abelian powers.
- But what about squares and cubes?

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Theorem (Andrade and Mol 2025+): There is an infinite additive 4-power-free rich word over $\{0, 1, 2\}$.

- The alphabet sizes are smallest possible, even for abelian powers.
- But what about squares and cubes?

Theorem (Pelantová and Starosta 2013): Every infinite rich word over a finite alphabet contains an ordinary square.

Avoiding abelian and additive powers in rich words

Theorem (Andrade and Mol 2025+): There is an infinite additive 5-power-free rich word over $\{0, 1\}$.

Theorem (Andrade and Mol 2025+): There is an infinite additive 4-power-free rich word over $\{0, 1, 2\}$.

- The alphabet sizes are smallest possible, even for abelian powers.
- But what about squares and cubes?

Theorem (Pelantová and Starosta 2013): Every infinite rich word over a finite alphabet contains an ordinary square.

Open Problem: Is there an infinite additive (or abelian) cube-free rich word over some finite subset of $\mathbb{Z}$?

Plan

Introduction

Our constructions

The template method

Our constructions

Define $\beta : \{0,1\}^* \to \{0,1\}^*$ and $\gamma : \{0,1,2\}^* \to \{0,1,2\}^*$ by

$$\beta(0) = 00001 \qquad \gamma(0) = 2$$
$$\beta(1) = 01101 \qquad \gamma(1) = 101$$
$$\gamma(2) = 10001$$

Our constructions

Define $\beta : \{0,1\}^* \to \{0,1\}^*$ and $\gamma : \{0,1,2\}^* \to \{0,1,2\}^*$ by

$$\begin{aligned} \beta(0) &= 00001 & \gamma(0) &= 2 \\ \beta(1) &= 01101 & \gamma(1) &= 101 \\ & & \gamma(2) &= 10001 \end{aligned}$$

We show that

- $\mathbf{B} = \beta^\omega(0)$ is rich and additive 5-power-free,

Our constructions

Define $\beta : \{0,1\}^* \to \{0,1\}^*$ and $\gamma : \{0,1,2\}^* \to \{0,1,2\}^*$ by

$$\beta(0) = 00001 \qquad \gamma(0) = 2$$
$$\beta(1) = 01101 \qquad \gamma(1) = 101$$
$$\gamma(2) = 10001$$

We show that

- $\mathbf{B} = \beta^\omega(0)$ is rich and additive 5-power-free, and
- $\mathbf{\Gamma} = \gamma^\omega(1)$ is rich and additive 4-power-free.

Proving richness

We make use of the following characterization.

Theorem (Droubay, Justin, and Pirillo 2001): An infinite word $\mathbf{w}$ is rich if and only if every finite prefix of $\mathbf{w}$ has a unioccurrent palindromic suffix.

Proving richness

We make use of the following characterization.

Theorem (Droubay, Justin, and Pirillo 2001): An infinite word $\mathbf{w}$ is rich if and only if every finite prefix of $\mathbf{w}$ has a unioccurrent palindromic suffix.

For $\mathbf{B}$, we use `Walnut`.

```
morphism b "0->00001 1->01101":
promote B b:
def FactorEq "?msd_5 Ak (k<n)=>(B[i+k]=B[j+k])":
def Occurs "?msd_5 (m<= n) & ( Ek (k+m<=n) & $FactorEq(i,j+k,m))":
def Palindrome "?msd_5 Aj,k ((k<n) & (j+k+1=n)) => (B[i+k]=B[i+j])":
def BisRich  "?msd_5 An Ej $Palindrome(j,n-j) & ~$Occurs(j,0,n-j,n-1)":
```

Proving richness

We make use of the following characterization.

Theorem (Droubay, Justin, and Pirillo 2001): An infinite word **w** is rich if and only if every finite prefix of **w** has a unioccurrent palindromic suffix.

For **B**, we use `Walnut`.

```
morphism b "0->00001 1->01101":
promote B b:
def FactorEq "?msd_5 Ak (k<n)=>(B[i+k]=B[j+k])":
def Occurs "?msd_5 (m<= n) & ( Ek (k+m<=n) & $FactorEq(i,j+k,m))":
def Palindrome "?msd_5 Aj,k ((k<n) & (j+k+1=n)) => (B[i+k]=B[i+j])":
def BisRich  "?msd_5 An Ej $Palindrome(j,n-j) & ~$Occurs(j,0,n-j,n-1)":
```

For **Γ**, we use an inductive proof with several cases.

Proving additive power-freeness

We use a recent variation of the template method.

Proving additive power-freeness

We use a recent variation of the template method.

Theorem (Currie and Rampersad 2012): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains abelian k-powers.

Proving additive power-freeness

We use a recent variation of the template method.

Theorem (Currie and Rampersad 2012): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains abelian k-powers.

Theorem (Rao and Rosenfeld 2018): Less restrictive conditions, can be used for both abelian and additive powers.

Proving additive power-freeness

We use a recent variation of the template method.

Theorem (Currie and Rampersad 2012): There is an algorithm which decides, under certain conditions on h, whether $h^{\omega}(0)$ contains abelian k-powers.

Theorem (Rao and Rosenfeld 2018): Less restrictive conditions, can be used for both abelian and additive powers.

Theorem (Currie, Mol, Rampersad, and Shallit 2024): More restrictive conditions, and works only for additive powers, but (probably) simpler and more efficient.

Proving additive power-freeness

We use a recent variation of the template method.

Theorem (Currie and Rampersad 2012): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains abelian k-powers.

Theorem (Rao and Rosenfeld 2018): Less restrictive conditions, can be used for both abelian and additive powers.

Theorem (Currie, Mol, Rampersad, and Shallit 2024): More restrictive conditions, and works only for additive powers, but (probably) simpler and more efficient.

- Jonathan Andrade implemented this algorithm in general, and then applied it to $\mathbf{B} = \beta^\omega(0)$ and $\mathbf{\Gamma} = \gamma^\omega(1)$.

Plan

Introduction

Our constructions

The template method

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^{\omega}(0)$ contains additive k-powers.

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.
- We can enumerate all short words in $h^\omega(0)$, and check to see if they match any of the templates.

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.
- We can enumerate all short words in $h^\omega(0)$, and check to see if they match any of the templates.

Some questions:

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.
- We can enumerate all short words in $h^\omega(0)$, and check to see if they match any of the templates.

Some questions:

- What are the conditions?

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.
- We can enumerate all short words in $h^\omega(0)$, and check to see if they match any of the templates.

Some questions:

- What are the conditions?
- How do we describe the "seed words" for additive k-powers?

Theorem (Currie, Mol, Rampersad, and Shallit 2024): There is an algorithm which decides, under certain conditions on h, whether $h^\omega(0)$ contains additive k-powers.

The essential ideas:

- Any long additive power in $h^\omega(0)$ must have arisen by applying h repeatedly to some short "seed word".
- These seed words cannot look "too different" from additive powers – there are only finitely many possible *templates* for these seed words.
- We can enumerate all short words in $h^\omega(0)$, and check to see if they match any of the templates.

Some questions:

- What are the conditions?
- How do we describe the "seed words" for additive k-powers?

For simplicity, we'll focus on additive *squares*.

Templates

Templates

- Let $\vec{\sigma}(w)$ denote the vector $\begin{bmatrix} \text{length of } w \\ \text{sum of } w \end{bmatrix}$.

Templates

- Let $\vec{\sigma}(w)$ denote the vector $\begin{bmatrix} \text{length of } w \\ \text{sum of } w \end{bmatrix}$.
- A *template* is a 4-tuple

$$[a_0, a_1, a_2, \vec{d}]$$

letters or ε vector in $\mathbb{Z}^2$

Templates

- Let $\vec{\sigma}(w)$ denote the vector $\begin{bmatrix} \text{length of } w \\ \text{sum of } w \end{bmatrix}$.
- A *template* is a 4-tuple

$$[a_0, a_1, a_2, \vec{d}]$$

letters or ε vector in $\mathbb{Z}^2$

- A word w is an *instance* of this template if

$$w = a_0 w_0 a_1 w_1 a_2 \quad \text{and} \quad \vec{\sigma}(w_1) - \vec{\sigma}(w_0) = \vec{d}.$$

Templates

- Let $\vec{\sigma}(w)$ denote the vector $\begin{bmatrix} \text{length of } w \\ \text{sum of } w \end{bmatrix}$.
- A *template* is a 4-tuple

$$[a_0, a_1, a_2, \vec{d}]$$

letters or ε vector in $\mathbb{Z}^2$

- A word w is an *instance* of this template if

$$w = a_0 w_0 a_1 w_1 a_2 \quad \text{and} \quad \vec{\sigma}(w_1) - \vec{\sigma}(w_0) = \vec{d}.$$

- An instance of $[\varepsilon, \varepsilon, \varepsilon, \vec{0}]$ is an additive square.

Templates

- Let $\vec{\sigma}(w)$ denote the vector $\begin{bmatrix} \text{length of } w \\ \text{sum of } w \end{bmatrix}$.
- A *template* is a 4-tuple

$$[a_0, a_1, a_2, \vec{d}]$$

letters or ε vector in $\mathbb{Z}^2$

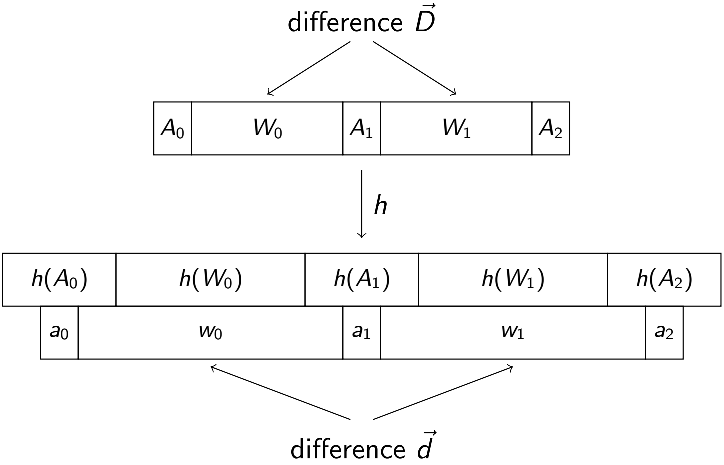
- A word w is an *instance* of this template if

$$w = a_0 w_0 a_1 w_1 a_2 \quad \text{and} \quad \vec{\sigma}(w_1) - \vec{\sigma}(w_0) = \vec{d}.$$

- An instance of $[\varepsilon, \varepsilon, \varepsilon, \vec{0}]$ is an additive square.
- An instance of $[0, 1, 0, [1, 3]^T]$ is "not too far" from an additive square.

Parents

Every long-enough instance of a template must have come from an instance of another template – a *parent*.



The First Two Conditions

- Condition 1: The morphism h is affine, i.e., for all letters x,
  - the length of $h(x)$ is given by $a + bx$ for some $a, b \in \mathbb{Z}$, and
  - the sum of $h(x)$ is given by $c + dx$ for some $c, d \in \mathbb{Z}$.

The First Two Conditions

- Condition 1: The morphism h is affine, i.e., for all letters x,
  - the length of $h(x)$ is given by $a + bx$ for some $a, b \in \mathbb{Z}$, and
  - the sum of $h(x)$ is given by $c + dx$ for some $c, d \in \mathbb{Z}$.
- Record this in the matrix $M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

The First Two Conditions

- Condition 1: The morphism h is affine, i.e., for all letters x,
  - the length of $h(x)$ is given by $a + bx$ for some $a, b \in \mathbb{Z}$, and
  - the sum of $h(x)$ is given by $c + dx$ for some $c, d \in \mathbb{Z}$.
- Record this in the matrix $M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.
- E.g., $\beta(0) = \texttt{00001}$ and $\beta(1) = \texttt{01101}$, so $M_\beta = \begin{bmatrix} 5 & 0 \\ 1 & 2 \end{bmatrix}$.

The First Two Conditions

- Condition 1: The morphism h is affine, i.e., for all letters x,
  - the length of $h(x)$ is given by $a + bx$ for some $a, b \in \mathbb{Z}$, and
  - the sum of $h(x)$ is given by $c + dx$ for some $c, d \in \mathbb{Z}$.
- Record this in the matrix $M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.
- E.g., $\beta(0) = \mathtt{00001}$ and $\beta(1) = \mathtt{01101}$, so $M_\beta = \begin{bmatrix} 5 & 0 \\ 1 & 2 \end{bmatrix}$.
- Then $\vec{\sigma}(h(W)) = M_h \vec{\sigma}(W)$.

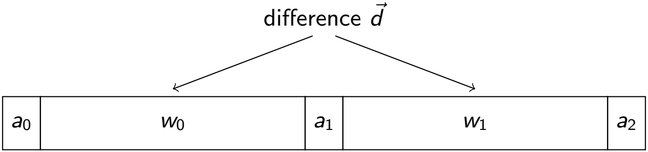
The First Two Conditions

- Condition 1: The morphism h is affine, i.e., for all letters x,
  - the length of $h(x)$ is given by $a + bx$ for some $a, b \in \mathbb{Z}$, and
  - the sum of $h(x)$ is given by $c + dx$ for some $c, d \in \mathbb{Z}$.
- Record this in the matrix $M_h = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.
- E.g., $\beta(0) = 00001$ and $\beta(1) = 01101$, so $M_\beta = \begin{bmatrix} 5 & 0 \\ 1 & 2 \end{bmatrix}$.
- Then $\vec{\sigma}(h(W)) = M_h \vec{\sigma}(W)$.
- Condition 2: M_h is invertible, so that

$$\vec{\sigma}(W) = M_h^{-1} \vec{\sigma}(h(W)).$$

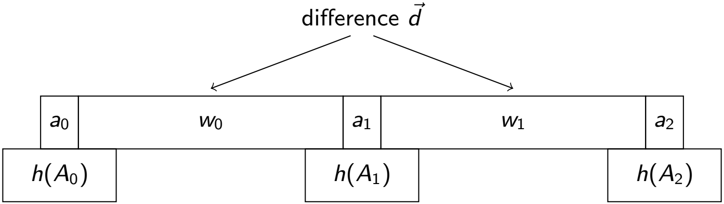
Finding Parents

These first two conditions allow us to find all possible parents of a given template.



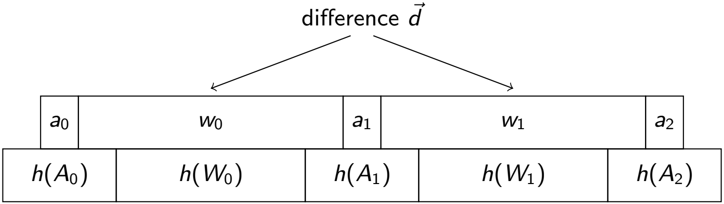
Finding Parents

These first two conditions allow us to find all possible parents of a given template.



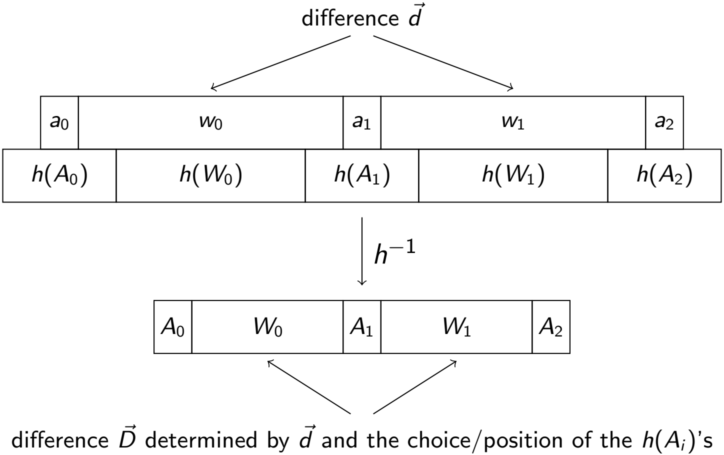
Finding Parents

These first two conditions allow us to find all possible parents of a given template.



Finding Parents

These first two conditions allow us to find all possible parents of a given template.



The Third Condition

- Now that we can compute the parents of a given template t, we want to compute the set of all *ancestors* of t (parents, grandparents, great-grandparents, etc.)

The Third Condition

- Now that we can compute the parents of a given template t, we want to compute the set of all *ancestors* of t (parents, grandparents, great-grandparents, etc.)
- How do we know that this set is finite?

The Third Condition

- Now that we can compute the parents of a given template t, we want to compute the set of all *ancestors* of t (parents, grandparents, great-grandparents, etc.)
- How do we know that this set is finite?
- We need a condition which guarantees that for any ancestor $T = [A_0, A_1, A_2, \vec{D}]$, the difference $\vec{D}$ is not too large.

The Third Condition

- Now that we can compute the parents of a given template t, we want to compute the set of all *ancestors* of t (parents, grandparents, great-grandparents, etc.)
- How do we know that this set is finite?
- We need a condition which guarantees that for any ancestor $T = [A_0, A_1, A_2, \vec{D}]$, the difference $\vec{D}$ is not too large.
- Condition 3: All eigenvalues of M_h are larger than 1 in absolute value.

The Last Condition

- Condition 4: h is strictly growing.

The Last Condition

- Condition 4: h is strictly growing.
- So taking preimages makes words shorter!

The Last Condition

- Condition 4: h is strictly growing.
- So taking preimages makes words shorter!
- Thus, if $h^\omega(0)$ contains an instance of a template t, then $h^\omega(0)$ contains a *short* instance of some ancestor of t.

Description of the Algorithm

Suppose that h satisfies these four conditions.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!
- If $h^{\omega}(0)$ contains an additive square, then it must contain a *short* instance of one of these ancestors.

Description of the Algorithm

Suppose that *h* satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!
- If $h^{\omega}(0)$ contains an additive square, then it must contain a *short* instance of one of these ancestors.
  - These are our potential "seed words".

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!
- If $h^\omega(0)$ contains an additive square, then it must contain a *short* instance of one of these ancestors.
  - These are our potential "seed words".
- We enumerate all short factors of $h^\omega(0)$, and check to see if any of them is an instance of an ancestor of t.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!
- If $h^\omega(0)$ contains an additive square, then it must contain a *short* instance of one of these ancestors.
  - These are our potential "seed words".
- We enumerate all short factors of $h^\omega(0)$, and check to see if any of them is an instance of an ancestor of t.
  - If so, then $h^\omega(0)$ contains an additive square.

Description of the Algorithm

Suppose that h satisfies these four conditions.

- Consider the template $t = [\varepsilon, \varepsilon, \varepsilon, \vec{0}]$.
  - An instance of t is an additive square.
- We enumerate all ancestors of t.
  - This set is finite!
- If $h^\omega(0)$ contains an additive square, then it must contain a *short* instance of one of these ancestors.
  - These are our potential "seed words".
- We enumerate all short factors of $h^\omega(0)$, and check to see if any of them is an instance of an ancestor of t.
  - If so, then $h^\omega(0)$ contains an additive square.
  - If not, then $h^\omega(0)$ is additive square-free!

Outlook

Outlook

Open Problem: Is there an infinite additive square-free word over some finite subset of $\mathbb{Z}$?

Outlook

Open Problem: Is there an infinite additive square-free word over some finite subset of $\mathbb{Z}$?

Open Problem: Is there an infinite additive cube-free rich word over some finite subset of $\mathbb{Z}$?

Outlook

Open Problem: Is there an infinite additive square-free word over some finite subset of $\mathbb{Z}$?

Open Problem: Is there an infinite additive cube-free rich word over some finite subset of $\mathbb{Z}$?

Open Problem: Is there an infinite additive 4-power-free rich word over *every* subset of $\mathbb{Z}$ of size 3?

Thank you!