

C | A | U

# $\alpha$ - $\beta$ -Factorisation and the binary case of Simon's Congruence

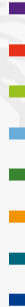
Pamela Fleischmann, Jonas Höfer, Annika Huch, Dirk Nowotka

33. Theorietag  
"Automaten und Formale Sprachen" 2023

# What is a Scattered Factor?

Example

palindrome



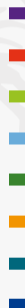
# What is a Scattered Factor?

Example

palindrome



pal



# What is a Scattered Factor?

Example

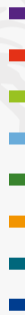
palindrome



palm



and



# What is a Scattered Factor?

Example

palindrome



palm



and



lime

# What is a Scattered Factor?

Example

palindrome



palm



and



lime



dome

# Scattered Factors

## Definition

**Definition.**

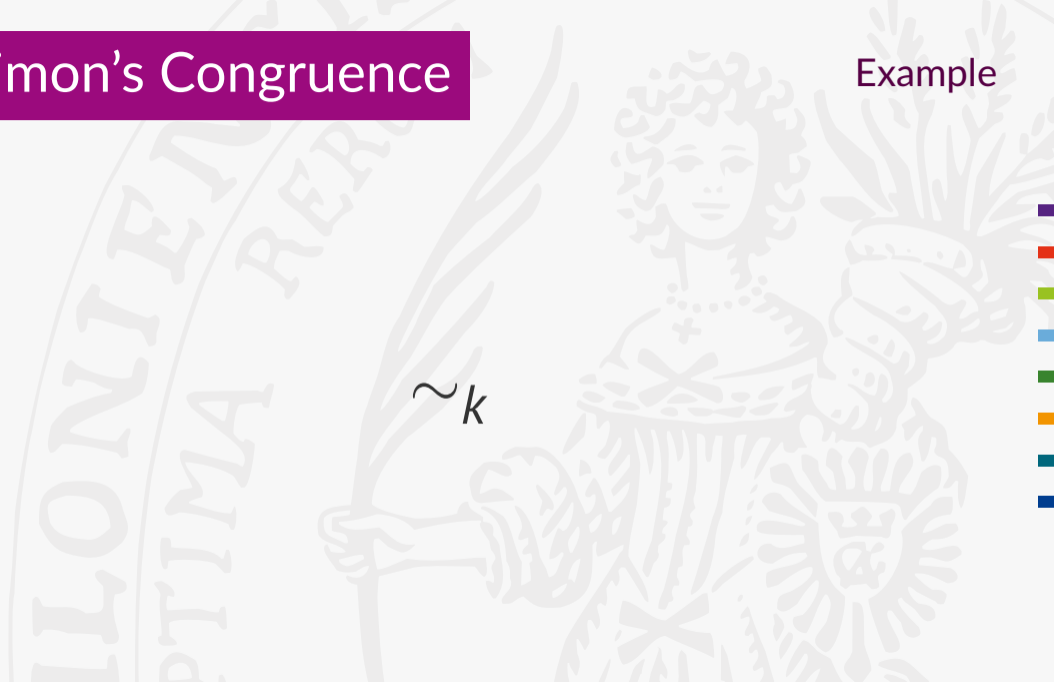
A word  $u = u[1] \cdots u[n] \in \Sigma^*$  **scattered factor** of  $v \in \Sigma^*$  if

$$\exists x_1, \dots, x_{n+1} \in \Sigma^* : v = x_1 u[1] x_2 u[2] \cdots x_n u[n] x_{n+1}$$

# Simon's Congruence

Example

$\sim_k$



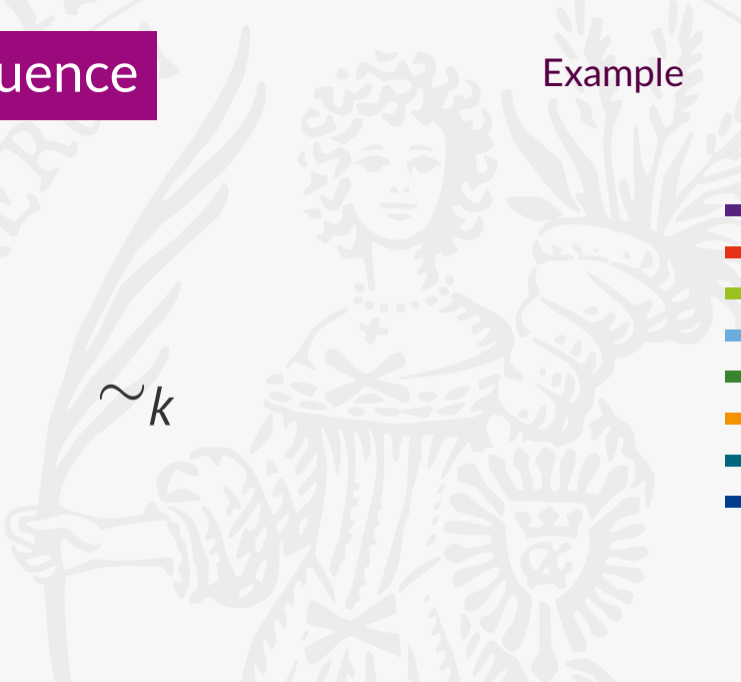


# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$\sim_k$



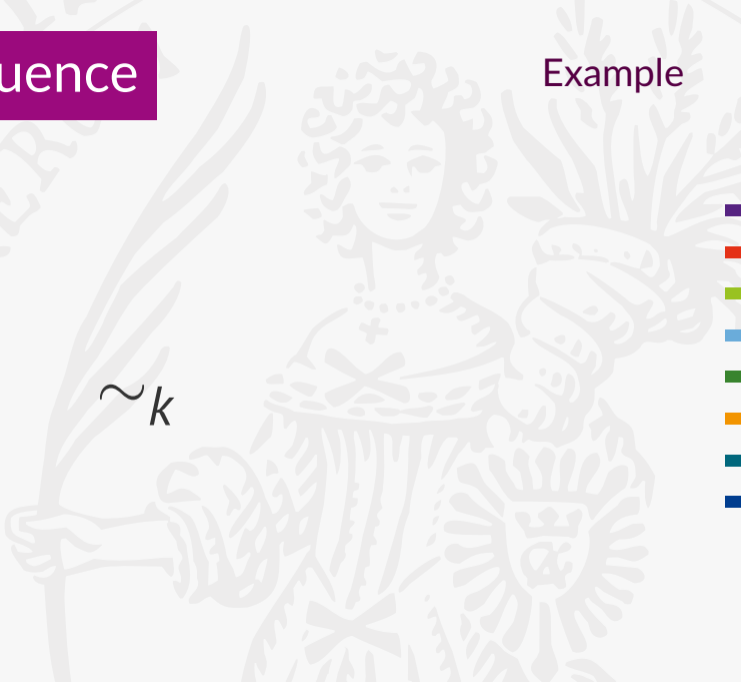
# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$\text{ScatFact}_3(abaa)$   
=  
{aaa, aba, baa}

$\sim_k$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$\text{ScatFact}_3(abaa)$

$=$

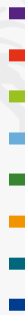
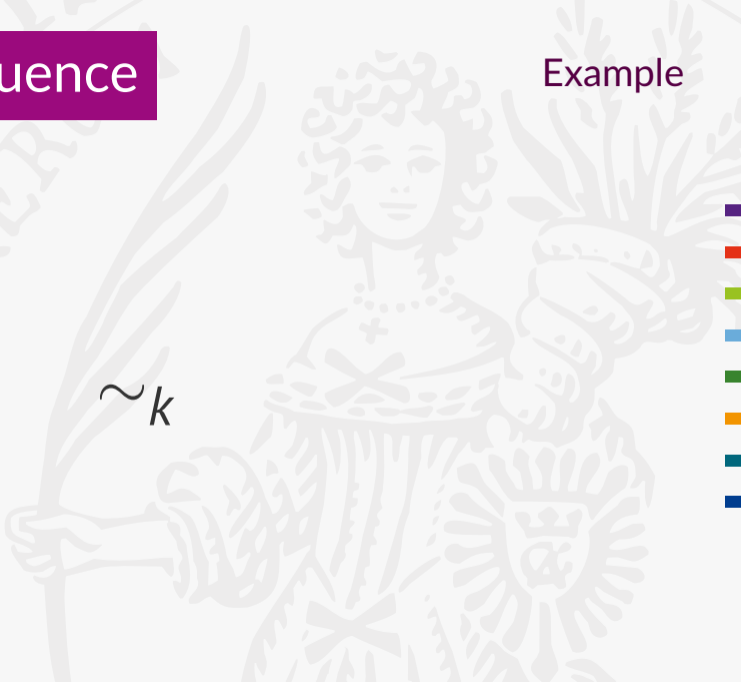
$\{aaa, aba, baa\}$

$\sim_k$

$\text{ScatFact}_3(abaaa)$

$=$

$\{aaa, aba, baa\}$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$\text{ScatFact}_3(abaa)$   
=  
{aaa, aba, baa}

$\text{ScatFact}_3(abaaa)$   
=  
{aaa, aba, baa}

$\sim_k$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

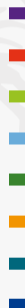
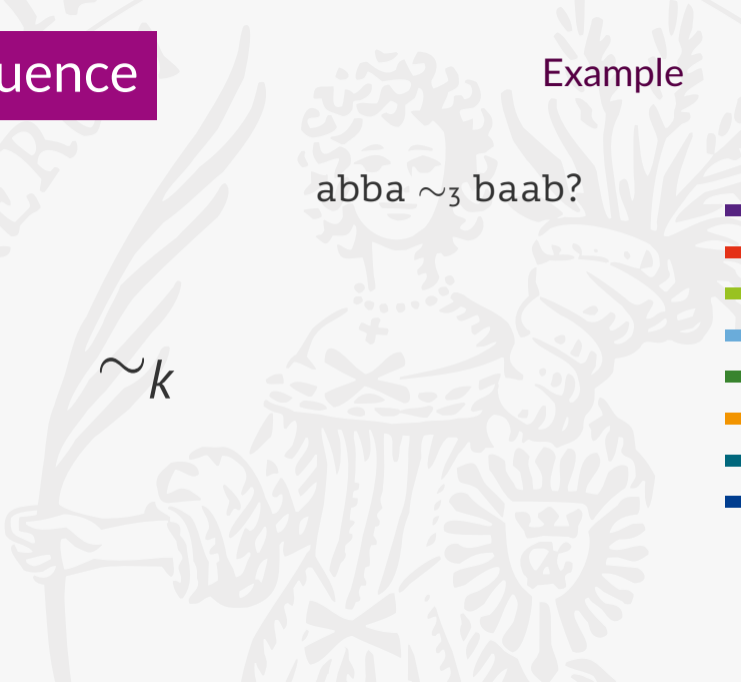
$$\begin{aligned} \text{ScatFact}_3(abaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(abaaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$\sim_k$



$abba \sim_3 baab?$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$$\begin{aligned} \text{ScatFact}_3(abaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(abaaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$\sim_k$

$abba \sim_3 baab?$

$$\begin{aligned} \text{ScatFact}_3(abba) \\ = \\ \{aba, abb, bba\} \end{aligned}$$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$$\begin{aligned} \text{ScatFact}_3(abaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(abaaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$\sim_k$

$abba \sim_3 baab?$

$$\begin{aligned} \text{ScatFact}_3(abba) \\ = \\ \{aba, abb, bba\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(baab) \\ = \\ \{aab, baa, bab\} \end{aligned}$$



# Simon's Congruence

Example

$abaa \sim_3 abaaa?$

$$\begin{aligned} \text{ScatFact}_3(abaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(abaaa) \\ = \\ \{aaa, aba, baa\} \end{aligned}$$



$\sim_k$

$abba \sim_3 baab?$

$$\begin{aligned} \text{ScatFact}_3(abba) \\ = \\ \{aba, abb, bba\} \end{aligned}$$

$$\begin{aligned} \text{ScatFact}_3(baab) \\ = \\ \{aab, baa, bab\} \end{aligned}$$





# Simon's Congruence

## Comparing Words

**Definition.**

The words  $u$  and  $v$  are **Simon congruent modulo**  $k \in \mathbb{N}_0$  if

$$\text{ScatFact}_\ell(u) = \text{ScatFact}_\ell(v) \quad \text{for all } \ell \leq k$$

# Simon's Congruence

## Comparing Words

### Definition.

The words  $u$  and  $v$  are **Simon congruent modulo  $k \in \mathbb{N}_0$**  if

$$\text{ScatFact}_\ell(u) = \text{ScatFact}_\ell(v) \quad \text{for all } \ell \leq k$$

- what is the index  $|\Sigma^* / \sim_k|$  for a fixed  $k \in \mathbb{N}$ ?

# Scattered Factor Universality

## Definition

### Definition.

A word  $w \in \Sigma^*$  is called *k-universal* if

$$\text{ScatFact}_k(w) = \Sigma^k.$$

- $\iota(w)$  largest number such that  $\text{ScatFact}_k(w) = \Sigma^{\iota(w)}$

# Arch Factorisation (Hébrard)

"Formalisation"

aabacbc**bc**abc**bc**abc

The diagram illustrates the Arch Factorisation of the string 'aabacbcabcabc'. The string is written in black text. The first 'bc' pair is highlighted in blue. A thin black line is drawn below the string, starting under the first 'bc' and ending under the second 'bc'. This line is composed of several connected segments, forming a series of arches that connect the first 'bc' to the second 'bc', and then to the third 'bc'. The background features a faint watermark of a university crest with the text 'ALLO NI ET OPTIMA PER' and a vertical bar on the right with colored segments (purple, red, green, blue, orange, dark blue).

# Arch Factorisation (Hébrard)

"Formalisation"

aabacbc**bc**abc**bc**abc

# of Arches

$$\iota(w) = 3$$

# Arch Factorisation (Hébrard)

"Formalisation"

aabacbc**bc**abc**bc**abc

# of Arches

$$\iota(w) = 3$$

Rest

$$r(w) = bc$$

# Arch Factorisation (Hébrard)

"Formalisation"

aabacbc**bc**abc**bc**abc

# of Arches

$$\iota(w) = 3$$

Rest

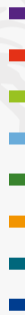
$$r(w) = bc$$

Modus

$$m(\text{aabacbc**bc**abc**bc**abc}) = \text{caa}$$

# $\alpha$ - $\beta$ -Factorisation

both directions





# $\alpha$ - $\beta$ -Factorisation

both directions



- arches and reverse arches **always** overlap

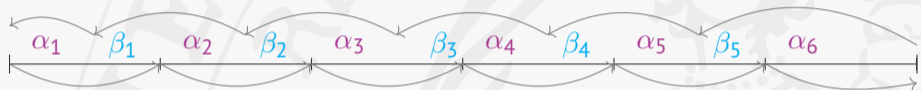
# $\alpha$ - $\beta$ -Factorisation

Refinement



# $\alpha$ - $\beta$ -Factorisation

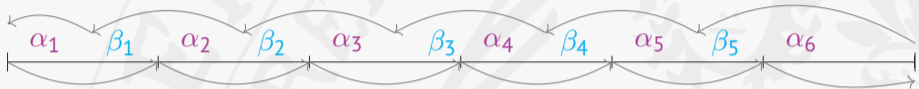
Refinement



- first and last letter in  $\beta_i$  unique
- $\alpha_j$  to fill up arches with  $\beta_{j-1}$  and  $\beta_j$
- letters in  $\alpha_j$  arbitrarily often and permuted

# $\alpha$ - $\beta$ -Factorisation

Refinement



- **first** and **last** letter in  $\beta_i$  **unique**
- $\alpha_j$  to fill up arches with  $\beta_{i-1}$  and  $\beta_i$
- letters in  $\alpha_j$  arbitrarily often and permuted

Assume  $\iota(w) < k!$

# $\alpha$ - $\beta$ -Factorisation

key properties

**Lemma.**

- $W \sim_k \tilde{W} \Rightarrow \alpha_i \sim_{k-l(W)} \tilde{\alpha}_i$

# $\alpha$ - $\beta$ -Factorisation

key properties

**Lemma.**

- $W \sim_k \tilde{W} \Rightarrow \alpha_i \sim_{k-l(w)} \tilde{\alpha}_i$
- $W \sim_k \tilde{W} \Rightarrow \alpha_i \beta_i \cdots \alpha_j \sim_{k-l(w)+(j-i)} \tilde{\alpha}_i \tilde{\beta}_i \cdots \tilde{\alpha}_j$

# $\alpha$ - $\beta$ -Factorisation

key properties

## Lemma.

- $W \sim_k \tilde{W} \Rightarrow \alpha_i \sim_{k-\iota(W)} \tilde{\alpha}_i$
- $W \sim_k \tilde{W} \Rightarrow \alpha_i \beta_i \cdots \alpha_j \sim_{k-\iota(W)+(j-i)} \tilde{\alpha}_i \tilde{\beta}_i \cdots \tilde{\alpha}_j$
- up to  $\sim_k$  factor of form  $\alpha_i \beta_i \cdots \alpha_j$  can be replaced by a  $(k - \iota(W) + (j - i))$ -equivalent one

# $\alpha$ - $\beta$ -Factorisation

key properties

## Lemma.

- $w \sim_k \tilde{w} \Rightarrow \alpha_i \sim_{k-\iota(w)} \tilde{\alpha}_i$
- $w \sim_k \tilde{w} \Rightarrow \alpha_i \beta_i \cdots \alpha_j \sim_{k-\iota(w)+(j-i)} \tilde{\alpha}_i \tilde{\beta}_i \cdots \tilde{\alpha}_j$
- up to  $\sim_k$  factor of form  $\alpha_i \beta_i \cdots \alpha_j$  can be replaced by a  $(k - \iota(w) + (j - i))$ -equivalent one

freely exchange  $\alpha$  bordered factors w.r.t.  $\sim_{k-\ell}$  ( $\ell$   $\beta$ -factors outside)



# $\alpha$ - $\beta$ -Decomposition

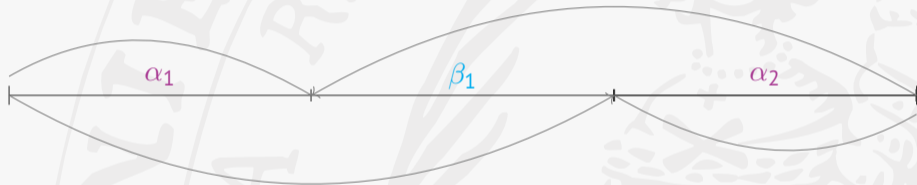
just triples

**Theorem.**

$$w \sim_k \tilde{w} \Leftrightarrow \forall i \in [\ell(w)] : \alpha_i \beta_i \alpha_{i+1} \sim_{k-\ell(w)+1} \tilde{\alpha}_i \tilde{\beta}_i \tilde{\alpha}_{i+1}$$

# Binary Case

$\alpha$ - $\beta$ -Factorisation



# Binary Case

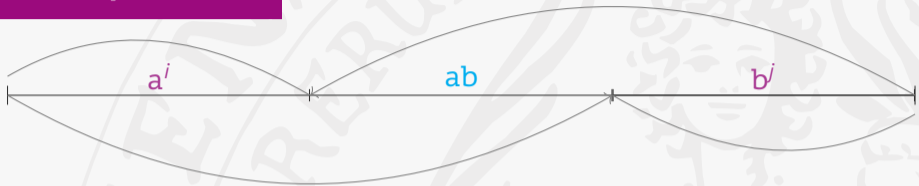
$\alpha$ - $\beta$ -Factorisation



- $i, j \in \mathbb{N}_0$

# Binary Case

$\alpha$ - $\beta$ -Factorisation



- $i, j \in \mathbb{N}_0$



- $i, j \in \mathbb{N}$

# Binary Case

## Characterisation

**Theorem.**

$w \sim_k \tilde{w}$  iff

# Binary Case

## Characterisation

**Theorem.**

$w \sim_k \tilde{w}$  iff

- $\iota(w) = \iota(\tilde{w})$

# Binary Case

## Characterisation

### Theorem.

$w \sim_k \tilde{w}$  iff

- $\iota(w) = \iota(\tilde{w})$
- $\alpha_i \sim_{k-\iota(w)} \tilde{\alpha}_i$  for all  $i \in [\iota(w) + 1]$

# Binary Case

## Characterisation

### Theorem.

$w \sim_k \tilde{w}$  iff

- $\iota(w) = \iota(\tilde{w})$
- $\alpha_i \sim_{k-\iota(w)} \tilde{\alpha}_i$  for all  $i \in [\iota(w) + 1]$
- $\beta_i = \tilde{\beta}_i$  for all  $i \in [\iota(w)]$

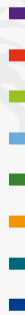


# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$



# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed



# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed
- **problem**  $\alpha_j, \tilde{\alpha}_j$  only congruent modulo  $k - \iota(w) - 1$

# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed
- **problem**  $\alpha_j, \tilde{\alpha}_j$  only congruent modulo  $k - \iota(w) - 1$
- **solution**
  - increase  $k$  and  $\iota(w)$  together

# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed
- **problem**  $\alpha_j, \tilde{\alpha}_j$  only congruent modulo  $k - \iota(w) - 1$
- **solution**
  - increase  $k$  and  $\iota(w)$  together
  - append new  $\beta\alpha$

# Enumerate $|\Sigma^* / \sim_k|$

Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed
- **problem**  $\alpha_j, \tilde{\alpha}_j$  only congruent modulo  $k - \iota(w) - 1$
- **solution**
  - increase  $k$  and  $\iota(w)$  together
  - append new  $\beta\alpha$
  - consider result up to  $\sim_{k+1}$

# Enumerate $|\Sigma^* / \sim_k|$

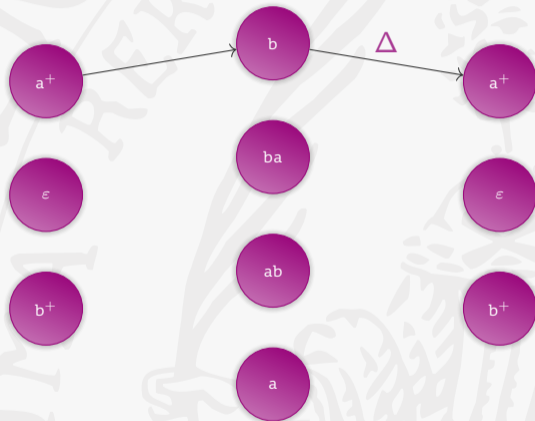
Binary Case

Idea:

- separate by  $\iota(w)$
- **notice** not all  $\alpha\beta$  combinations are allowed
- **problem**  $\alpha_j, \tilde{\alpha}_j$  only congruent modulo  $k - \iota(w) - 1$
- **solution**
  - increase  $k$  and  $\iota(w)$  together
  - append new  $\beta\alpha$
  - consider result up to  $\sim_{k+1}$
  - fix the difference  $\Delta = k - \iota(w) \geq 1$

# Shape of $\alpha\beta\alpha$ triples

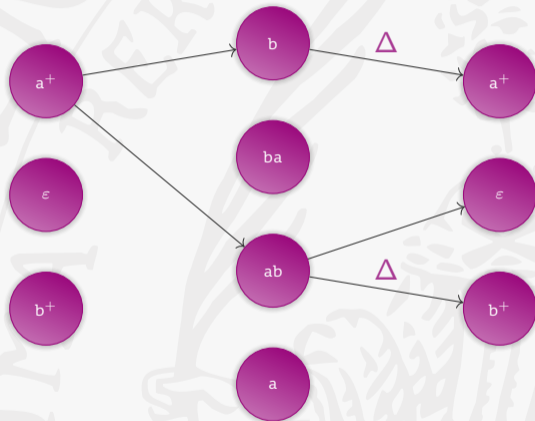
Binary Case





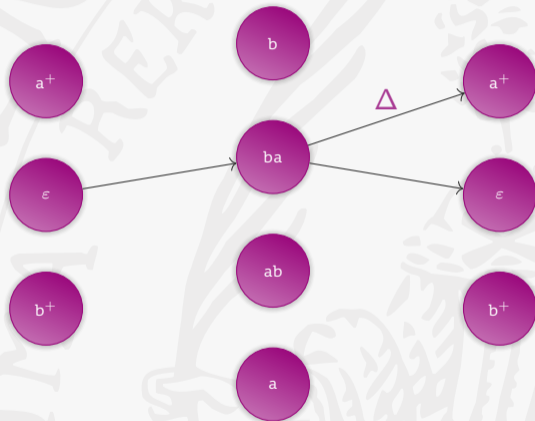
# Shape of $\alpha\beta\alpha$ triples

Binary Case



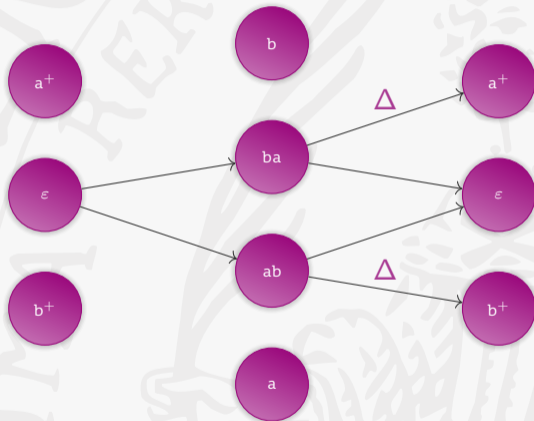
# Shape of $\alpha\beta\alpha$ triples

Binary Case



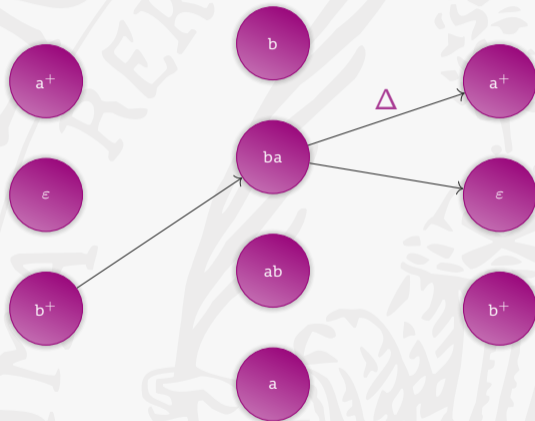
# Shape of $\alpha\beta\alpha$ triples

Binary Case



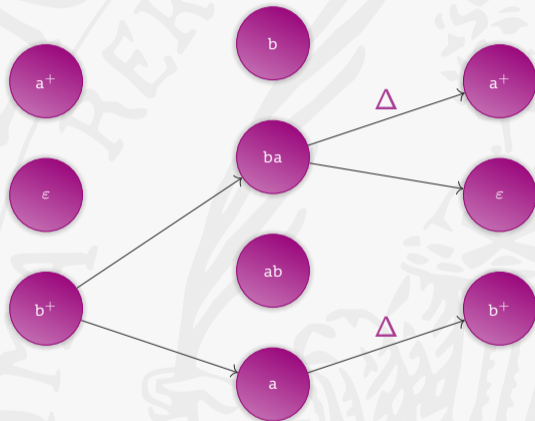
# Shape of $\alpha\beta\alpha$ triples

Binary Case



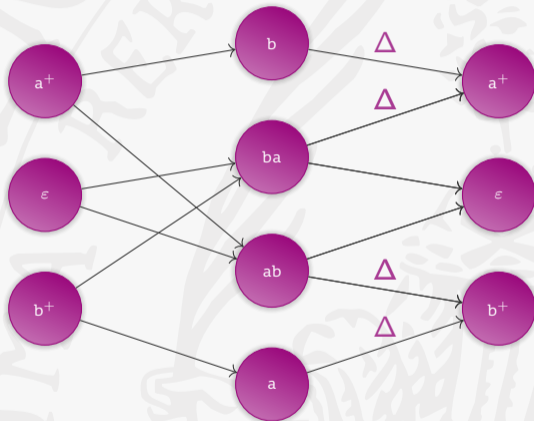
# Shape of $\alpha\beta\alpha$ triples

Binary Case



# Shape of $\alpha\beta\alpha$ triples

Binary Case



# Enumeration of $|\Sigma^* / \sim_k|$

## Closed Formula

- number of classes of words with  $\ell$  arches w.r.t.  $\ell + \Delta$  ( $\Delta = k - \iota(w)$ )

$$c_{\Delta}(\ell) = |\{w \in \Sigma^* \mid \iota(w) = \ell\} / \sim_{\Delta+\ell}| = \left\| \begin{pmatrix} \Delta & \Delta & \Delta \\ 1 & 2 & 1 \\ \Delta & \Delta & \Delta \end{pmatrix}^{\ell} \cdot \begin{pmatrix} \Delta \\ 1 \\ \Delta \end{pmatrix} \right\|_1$$

# Enumeration of $|\Sigma^* / \sim_k|$

## Closed Formula

- number of classes of words with  $\ell$  arches w.r.t.  $\ell + \Delta$  ( $\Delta = k - \iota(w)$ )

$$c_{\Delta}(\ell) = |\{w \in \Sigma^* \mid \iota(w) = \ell\} / \sim_{\Delta+\ell}| = \left\| \begin{pmatrix} \Delta & \Delta & \Delta \\ 1 & 2 & 1 \\ \Delta & \Delta & \Delta \end{pmatrix}^{\ell} \cdot \begin{pmatrix} \Delta \\ 1 \\ \Delta \end{pmatrix} \right\|_1$$

- $c_{\Delta}(-1) = 1, c_{\Delta}(0) = 2\Delta + 1$

$$c_{\Delta}(\ell + 2) = 2(\Delta + 1)c_{\Delta}(\ell + 1) - 2\Delta c_{\Delta}(\ell)$$



# Enumeration of $|\Sigma^* / \sim_k|$

## Closed Formula

- number of classes of words with  $\ell$  arches w.r.t.  $\ell + \Delta$  ( $\Delta = k - \iota(w)$ )

$$c_{\Delta}(\ell) = |\{w \in \Sigma^* \mid \iota(w) = \ell\} / \sim_{\Delta+\ell}| = \left\| \begin{pmatrix} \Delta & \Delta & \Delta \\ 1 & 2 & 1 \\ \Delta & \Delta & \Delta \end{pmatrix}^{\ell} \cdot \begin{pmatrix} \Delta \\ 1 \\ \Delta \end{pmatrix} \right\|_1$$

- $c_{\Delta}(-1) = 1, c_{\Delta}(0) = 2\Delta + 1$

$$c_{\Delta}(\ell + 2) = 2(\Delta + 1)c_{\Delta}(\ell + 1) - 2\Delta c_{\Delta}(\ell)$$

- $|\Sigma^* / \sim_k| = 1 + \sum_{\Delta=1}^k c_{\Delta}(k - \Delta)$

# Enumeration of $|\Sigma^* / \sim_k|$

## Closed Formula

- number of classes of words with  $\ell$  arches w.r.t.  $\ell + \Delta$  ( $\Delta = k - \iota(w)$ )

$$c_{\Delta}(\ell) = |\{w \in \Sigma^* \mid \iota(w) = \ell\} / \sim_{\Delta+\ell}| = \left\| \begin{pmatrix} \Delta & \Delta & \Delta \\ 1 & 2 & 1 \\ \Delta & \Delta & \Delta \end{pmatrix}^{\ell} \cdot \begin{pmatrix} \Delta \\ 1 \\ \Delta \end{pmatrix} \right\|_1$$

- $c_{\Delta}(-1) = 1, c_{\Delta}(0) = 2\Delta + 1$

$$c_{\Delta}(\ell + 2) = 2(\Delta + 1)c_{\Delta}(\ell + 1) - 2\Delta c_{\Delta}(\ell)$$

- $|\Sigma^* / \sim_k| = 1 + \sum_{\Delta=1}^k c_{\Delta}(k - \Delta)$

1, 4, 16, 68, 312, 1560, 8528, 50864, 329248, 2298592, 17203264, 137289920, 1162805376, 10409679744, 98146601216, 971532333824, 10068845515264, ...

# Ternary Alphabet

Beginning

$ alph(\alpha_0) ,  alph(\alpha_1) $	$alph(\alpha_0)$	$alph(\alpha_1)$	$\beta$ RegExp
2,2	$\{a, b\}$ $\{a, b\}$	$\{a, c\}$ $\{a, b\}$	$ba^*c$ $c$
2,1	$\{a, b\}$ $\{a, b\}$	$\{c\}$ $\{a\}$	$(ab^+   ba^+)c$ $ba^*c$
2,0	$\{a, b\}$	$\emptyset$	$(ab^+   b^+a)c$
1,1	$\{a\}$ $\{a\}$	$\{b\}$ $\{a\}$	$ab^+c   ac^+b   ca^+b$ $ba^*c$
1,0	$\{a\}$	$\emptyset$	$ba^*c   ab^+c$
0,0	$\emptyset$	$\emptyset$	$ab^+c$