α-β-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

palm

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

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

Simon's Congruence

Example

$\sim_k$

Simon's Congruence

Example

$\texttt{abaa} \sim_3 \texttt{abaaa}$?

$\sim_k$

Simon's Congruence

Example

$\mathtt{abaa} \sim_3 \mathtt{abaaa}?$

$\mathrm{ScatFact}_3(\mathtt{abaa})$

$=$

$\{\mathtt{aaa}, \mathtt{aba}, \mathtt{baa}\}$

$\sim_k$

Simon's Congruence

Example

$abaa \sim_3 abaaa?$

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

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

$\sim_k$

Simon's Congruence

Example

$\texttt{abaa} \sim_3 \texttt{abaaa}?$

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

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

$\sim_k$

Simon's Congruence

Example

$abaa \sim_3 abaaa?$

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

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

$\sim_k$

$abba \sim_3 baab?$

Simon's Congruence

Example

$abaa \sim_3 abaaa?$

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

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

$\sim_k$

$abba \sim_3 baab?$

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

Simon's Congruence

Example

$abaa \sim_3 abaaa?$

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

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

$\sim_k$

$abba \sim_3 baab?$

$\mathrm{ScatFact}_3(abba) = \{aba, abb, bba\}$

$\mathrm{ScatFact}_3(baab) = \{aab, baa, bab\}$

Simon's Congruence

Example

$abaa \sim_3 abaaa$?

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

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

$\sim_k$

$abba \sim_3 baab$?

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

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

Simon's Congruence

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

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

Simon's Congruence

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

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

- 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

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

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

Arch Factorisation (Hébrard)

"Formalisation"

aabacbcbcabcabc

Arch Factorisation (Hébrard)

"Formalisation"

aabacbcbcabcabc

# of Arches

$\iota(w) = 3$

Arch Factorisation (Hébrard)

"Formalisation"

aabacbcbcabcabc

# of Arches	Rest
$\iota(w) = 3$	$\mathrm{r}(w) = \mathrm{bc}$

Arch Factorisation (Hébrard)

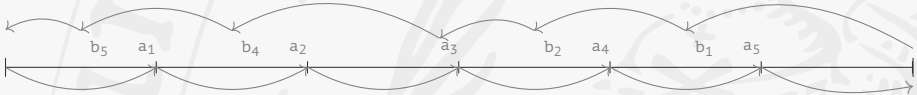
"Formalisation"

aabacbcbcabcabc

# of Arches	Rest	Modus
$\iota(w) = 3$	$\mathrm{r}(w) = \mathrm{bc}$	$\mathrm{m}(\mathrm{aabacbcbcabcabc}) = \mathrm{caa}$

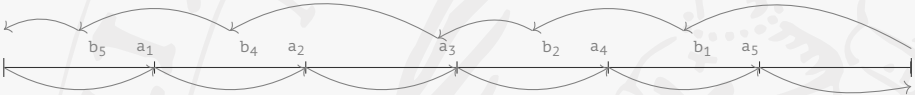
α-β-Factorisation

both directions



α-β-Factorisation

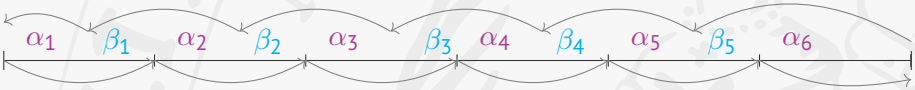
both directions



- arches and reverse arches always overlap

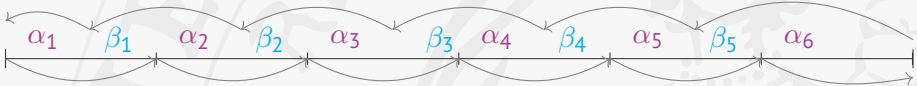
α-β-Factorisation

Refinement



α-β-Factorisation

Refinement

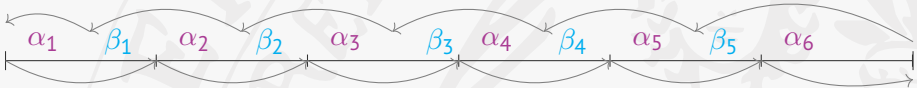


- first and last letter in β_i unique

- α_i to fill up arches with β_{i-1} and β_i
- letters in α_i arbitrarily often and permuted

α-β-Factorisation

Refinement



- first and last letter in β_i unique

- α_i to fill up arches with β_{i-1} and β_i
- letters in α_i arbitrarily often and permuted

Assume $\iota(w) < k$!

α-β-Factorisation

key properties

Lemma.

- $w \sim_k \tilde{w} \Rightarrow \alpha_i \sim_{k-\iota(w)} \tilde{\alpha}_i$

α-β-Factorisation

key properties

Lemma.

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

α-β-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

α-β-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 α bordered factors w.r.t. $\sim_{k-\ell}$ (ℓ β-factors *outside*)

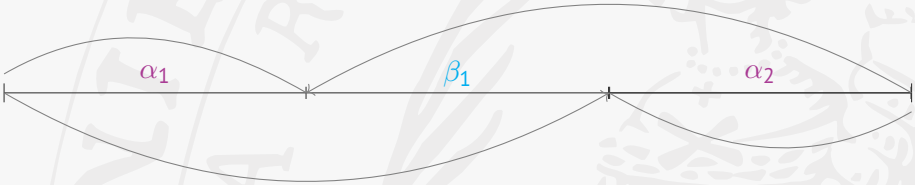
α-β-Decomposition

just triples

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

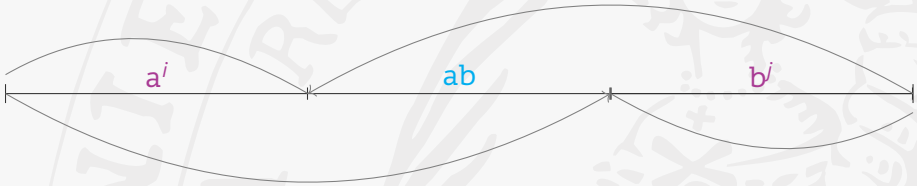
Binary Case

α-β-Factorisation



Binary Case

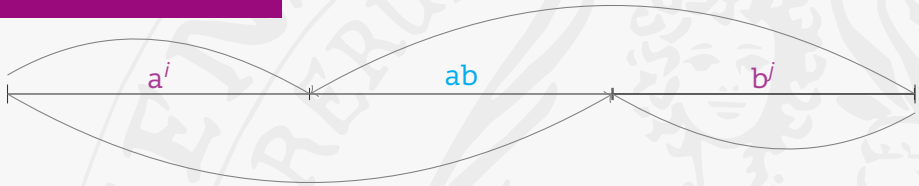
α-β-Factorisation



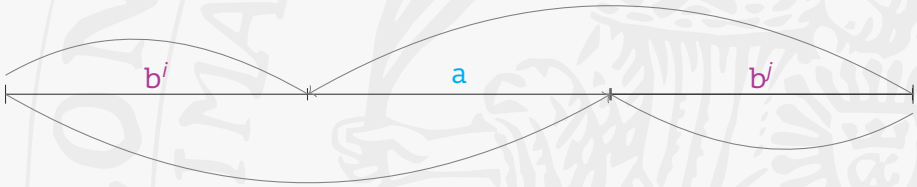
- $i, j \in \mathbb{N}_0$

Binary Case

α-β-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 $|\Sigma^*/\sim_k|$

Idea:

- seperate by $\iota(w)$

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

Idea:

- seperate by $\iota(w)$
- **notice** not all α-β combinations are allowed

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

Idea:

- seperate by $\iota(w)$
- **notice** not all α-β combinations are allowed
- **problem** α_i, $\tilde{\alpha}_i$ only congruent modulo $k - \iota(w) - 1$

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

Idea:

- seperate by $\iota(w)$
- **notice** not all α-β combinations are allowed
- **problem** α_i, $\tilde{\alpha}_i$ only congruent modulo $k - \iota(w) - 1$
- **solution**
  - increase k and $\iota(w)$ together

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

Idea:

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

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

Idea:

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

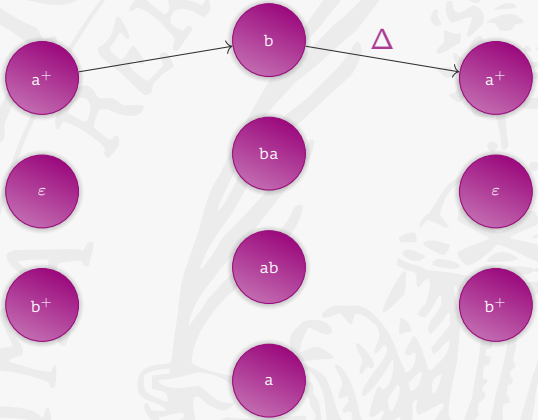
Enumerate $|\Sigma^* / \sim_k|$

Idea:

- seperate by $\iota(w)$
- **notice** not all α-β combinations are allowed
- **problem** α_i, $\tilde{\alpha}_i$ 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 $\Delta = k - \iota(w) \geq 1$

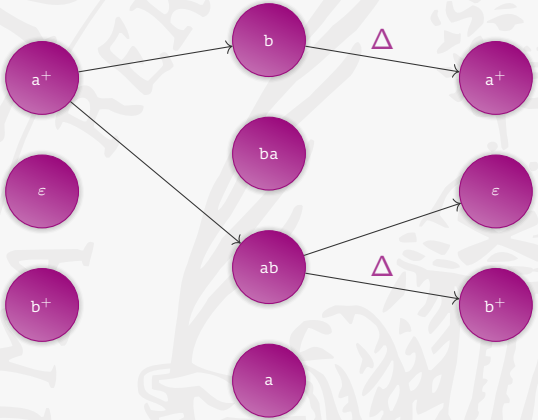
Shape of $\alpha\beta\alpha$ triples

Binary Case



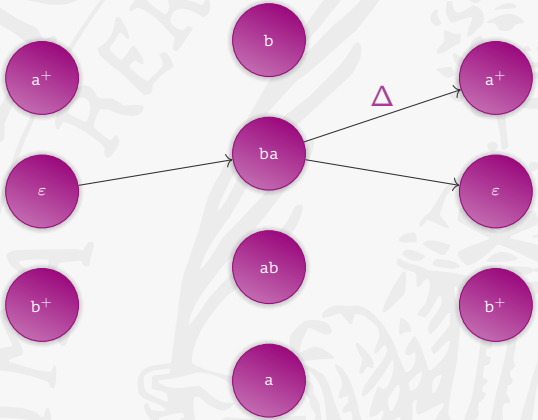
Shape of $\alpha\beta\alpha$ triples

Binary Case



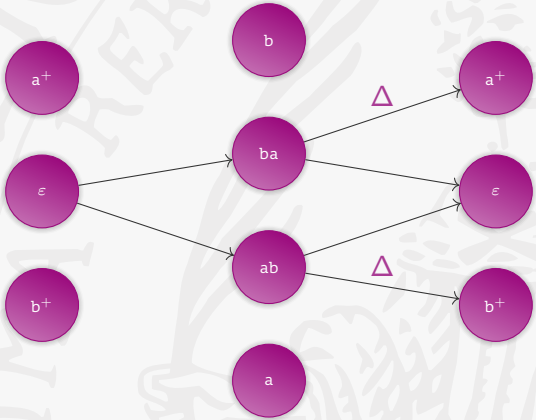
Shape of $\alpha\beta\alpha$ triples

Binary Case



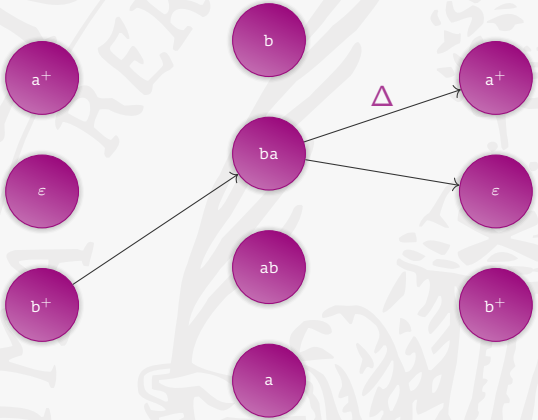
Shape of $\alpha\beta\alpha$ triples

Binary Case



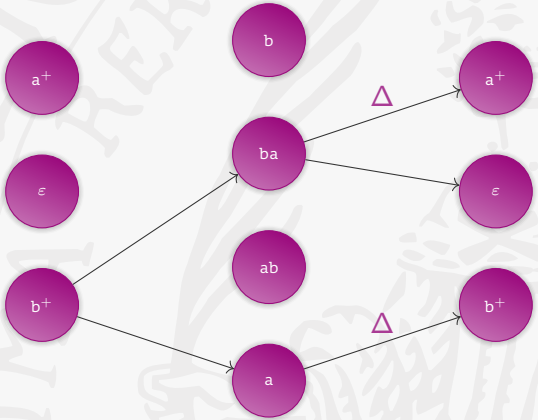
Shape of $\alpha\beta\alpha$ triples

Binary Case



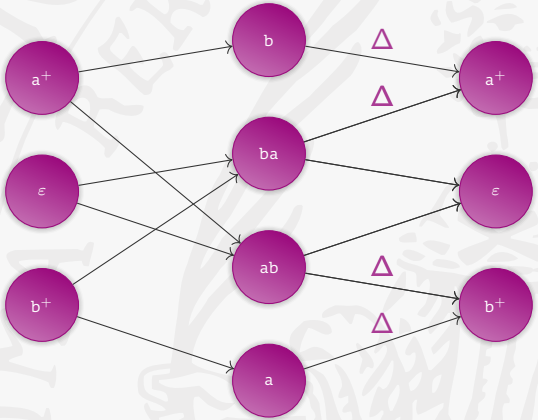
Shape of $\alpha\beta\alpha$ triples

Binary Case



Shape of $\alpha\beta\alpha$ triples

Binary Case



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

Closed Formula

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

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

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

Closed Formula

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

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

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

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

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

Closed Formula

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

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

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

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

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

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

Closed Formula

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

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

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

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

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

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

Ternary Alphabet

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