Punkty stałe nietrywialnych morfizmów

Wyznaczanie morfizmu dla danego słowa

Małgorzata Dymek

Informatyka stopień II rok I
Uniwesytet Jagielloński

25.03.2021

Cel pracy

- Czy dane słowo skończone jest punktem stałym jakiegoś morfizmu?
- Jaki to morfizm?

Podstawowe pojęcia

Σ – aflabet, $alph(w)$ – zbiór liter w słowie w,

$|w|_a$ – liczba wystąpień litery a w słowie w, $|w|_L = \sum_{a \in L} |w|_a$ dla $L \subseteq \Sigma$,

słowo w jest **punktem stałym** morfizmu $h : \Sigma^* \to \Sigma^*$ jeśli $h(w) = w$,

słowo w jest **morficznie nieprymitywne** jeśli $\exists a \in w : h(a) \neq a$,

słowo w jest **morficznie prymitywne** jeśli $\forall a \in w : h(a) = a \Leftrightarrow h = I$ dla $alph(w)$

Litery śmiertelne i rozszerzające się

$$\mathcal{M}_h = \{a \in \Sigma \mid \exists j \in \mathbb{N} : h^j(a) = \varepsilon\}$$
$$\mathcal{E}_h = \{b \in \Sigma \mid h(b) = xby,\ xy \in \mathcal{M}_h^*\}$$

w jest punktem stałym $h \ \Rightarrow \ w \in (\mathcal{M}_h \cup \mathcal{E}_h)^*$

$exp(h) = min(\{t \in \mathbb{N} \mid h^t(a) = \varepsilon \ \forall a \in \mathcal{M}_h\})$

$h^{exp(h)} = id \Leftrightarrow h = id$

morfizm jest **stabilny** jeśli $exp(h) = 1$

Faktoryzacja względem morfizmu

Faktoryzacja względem morfizmu

stabilny morfizm h definiuje k-elementową krotkę słów:

$$(w_1, w_2, \ldots, w_k)$$

taką, że $w = w_1 w_2 \ldots w_k$ oraz $\forall i = 1, \ldots, k \ \exists a_i \in \Sigma$ takie, że

$$w_i = h(a_i), \ |w_i|_{a_i} = 1.$$

Faktoryzacja jest **trywialna**, jeśli $\forall i = 1, \ldots, k \ |w_i| = 1$.

Examples

Zdefiniujmy

$$h(a) = \varepsilon, \ h(b) = a, \ h(c) = bcd, \ h(d) = \varepsilon, \ h(e) = de$$

ze punktem stałym $abcdde$ i $exp(h) = 2$.
Faktoryzacja względem morfizmu h^2 wygląda następująco: $(abcd, de)$.

Rozwinięcie przykładu

Examples

$$h(a) = \varepsilon,\ h(b) = a,\ h(c) = bcd,\ h(d) = \varepsilon,\ h(e) = de$$

w	a	b	c			d	d	e	
$h(w)$	ε	a	b	c	d	ε	ε	d	e
$h^2(w)$		ε	a	bcd	ε			ε	de

$F = (abcd, de)$

$w_1 = abcd,\ a_1 = c,\ h^2(c) = h(bcd) = abcd,$

$w_2 = de,\ a_2 = e,\ h^2(e) = h(de) = de,$

$\mathcal{M}_h = \{a, b, d\},\quad \mathcal{E}_h = \{c, e\}$

Prymitywność morficzna słów

Theorem (1)

Słowo $w \in \Sigma^*$ jest nieprymitywne morficzne $\Leftrightarrow$ istnieje stabilny morfizm h zdefiniowany dla $alph(w)$ taki, że $h(w) = w$ i faktoryzacja słowa w względem tego morfizmu zawiera przynajmniej jedno słowo długości większej od 1.

Istnieje faktoryzacja $F = (w_1, w_2, \ldots, w_k)$ słowa w oraz podzbiór $\mathcal{E}_f \subset alph(w)$, takie że:

(1) $|w_i|_{\mathcal{E}_f} = 1 \quad \forall i \in [1, \ldots, k]$,

(2) $alph(w_i) \cap \mathcal{E}_f = alph(w_j) \cap \mathcal{E}_f \;\Rightarrow\; w_i = w_j$.

Jeśli mamy morfizm, to możemy ustalić $\mathcal{E}_f = \mathcal{E}_h$ i faktoryzację taką, że $\forall w_i \in F$ $w_i = h(a_i),\ a_i \in \mathcal{E}_h \Rightarrow w_i = l(a_i)a_i r(a_i)$, $l(a_i), r(a_i) \in \mathcal{M}_h$.

Jeśli mamy faktoryzację, możemy zdefiniować morfizm $h(a) = w_i\ \forall a \in \mathcal{E}_f$, gdzie $\{a\} = alph(w_i) \cap \mathcal{E}_f$ i $h(b) = \varepsilon\ \forall b \notin \mathcal{E}_f$.

Morfizmy i faktoryzacje

Examples

Dla danego słowa *caddbccaddbc* możemy wyznaczyć kilka faktoryzacji według różnych morfizmów:

1. $F_1 = (caddbc, caddbc)$, wyznaczana przez dwa różne morfizmy h_1 i h'_1:
$$h_1(a) = caddbc, \quad \mathcal{M}_{h_1} = \{b, c, d\},$$
$$h'_1(b) = caddbc, \quad \mathcal{M}_{h'_1} = \{a, c, d\}.$$
2. $F_2 = (cad, dbc, cad, dbc)$, wyznaczana przez morfizm h_2:
$$h_2(a) = cad, \quad h_2(b) = dbc, \quad \mathcal{M}_{h_2} = \{c, d\}.$$
3. F_3 – faktoryzacja trywialna.

Sąsiedztwa liter

Definition

Zdefiniujmy $\mathbf{r}_a$ jako takie słowo, że $a\mathbf{r}_a$ stanowi najdłuższy wspólny prefiks wszystkich sufiksów słowa w zaczynających się literą a.
Analogicznie, zdefiniujmy $\mathbf{l}_a$ jako takie słowo, że $\mathbf{l}_a a$ stanowi najdłuższy wspólny sufiks wszystkich prefiksów słowa w kończących się literą a.
Największe wspólne sąsiedztwo wszystkich wystąpień litery a w słowie w definiujemy jako:

$$\mathbf{n}_a = \mathbf{l}_a a \mathbf{r}_a.$$

Co więcej, $a \notin \mathbf{l}_a$, $a \notin \mathbf{r}_a$ oraz jeśli $h(w) = w$ to $h(a)$ jest jakimś składnikiem $\mathbf{n}_a$.

Lemma (2)

$$\forall b \in alph(\mathbf{n}_a) \quad |w|_b \geq |w|_a$$

Sąsiedztwo – przykład

Examples

Dla słowa

$$abcabcaba$$

Mamy:

$\mathbf{n}_a = a$	$\mathbf{n}_b = \mathbf{a}b$	$\mathbf{n}_c = \mathbf{abcab}$
$\mathbf{a}$ $\quad$ $\mathbf{a}$	$\mathbf{ab}$ $\quad$ $\mathbf{b}a$	$\mathbf{abc}$ $\quad$ $\mathbf{cab}a$
$abc\mathbf{a}$ $\quad$ $\mathbf{a}ba$	$abc\mathbf{ab}$ $\quad$ $\mathbf{b}caba$	$abc\mathbf{abc}$ $\quad$ $\mathbf{cab}caba$
	$abcabc\mathbf{ab}$	

Bliźniacze litery

Definition

Litery a i b nazywamy **bliźniaczymi** gdy $b \in alph(\mathbf{n}_a)$ i $a \in alph(\mathbf{n}_b)$, co zapisujemy jako $a||b$.

Lemma (3)

(1) $a||a$,

(2) *jeśli* $a||b$, *to* $|w|_a = |w|_b$,

(3) *jeśli* $|w|_a = |w|_b$ *i* $a \in alph(\mathbf{n}_b)$, *to* $a||b$,

(4) *jeśli* $a||b$, *to* $\mathbf{n}_a = \mathbf{n}_b$.

Pozycja i częstotliwość występowania liter

Zbiór pozycji liter w słowie definiujemy jako

$$\mathcal{C}_w = \{0, 1, \ldots, |w|\},$$

gdzie dla $w = a_1 a_2 \ldots a_k$, a_i to litery, $w[i, j] = a_{i+1} \ldots a_j$.

Zbiór liter występujących najrzadziej w słowie zapisujemy:

$$\mu(w) = \{a | a \in alph(w) \wedge |w|_a \leq |w|_b \forall b \in alph(w)\}.$$

Pozycja względem litery rozszerzającej

Dla faktoryzacji F słowa w i zbioru $\mathcal{E}_F$ możemy zdefiniować zbiory pozycji będących po lewej lub prawej stronie od litery rozszerzającej w danym elemencie, używając morfizmu h takiego, że $\mathcal{E}_h = \mathcal{E}_F$:

$$\mathcal{L} = \mathcal{L}(F, \mathcal{E}_F) = \{i | i \geq |h(w[0, i])|\},$$
$$\mathcal{R} = \mathcal{R}(F, \mathcal{E}_F) = \{i | i \leq |h(w[0, i])|\}.$$

Zbiór $\mathcal{L} \cap \mathcal{R}$ zawiera litery $a_i \dots a_k$ wyznaczające faktoryzację.

Właściwości liter najrzadziej występujących

Lemma (4)

F – faktoryzacja słowa w względem stabilnego morfizmu h,
$\mathcal{E}_F$ – zbiór liter rozszerzajacych,
i, j – pozycje z $\mathcal{C}_w$ takie, że $i < j$, $i \in \mathcal{L}$ i $j \in \mathcal{R}$.

Jeśli litera $c \in \mu(w[i,j])$ to albo $c \in \mathcal{E}_F$, albo istnieje litera $c' \in alph(w[i,j])$ taka, że $c' \in \mathcal{E}_F$, $c||c'$ i $c \in alph(h(c'))$.

Dowód:
Z założeń na i i j wiemy, że $alph(w[i,j]) \cap \mathcal{E}_h \neq \emptyset$, tj. leży między nimi litera rozszerzająca. Ponadto dla każdego $b \in alph(w[i,j]) \cap \mathcal{M}_h$ istnieje $a \in alph(w[i,j]) \cap \mathcal{E}_h$ taka, że $b \in alph(h(a))$.
Jeśli $c \notin \mathcal{E}_h$, to istnieje $c' \in alph(w[i,j]) \cap \mathcal{E}_h$ takie, że $c \in alph(h(c'))$. Wtedy z Lematu 2 wiemy, że $|w|_c \geq |w|_{c'}$. Z założenia $c \in \mu(w[i,j])$ wynika, że $|w|_c = |w|_{c'}$, a więc z Lematu 3 otrzymujemy $c||c'$.

Właściwości liter najrzadziej występujących

Lemma (4)

F – faktoryzacja słowa w względem stabilne morfizmu h,
$\mathcal{E}_F$ – zbiór liter rozszerzajacych,
i, j – pozycje z $\mathcal{C}_w$ takie, że $i < j$, $i \in \mathcal{L}$ i $j \in \mathcal{R}$.

Jeśli litera $c \in \mu(w[i,j])$ to albo $c \in \mathcal{E}_F$, albo istnieje litera $c' \in alph(w[i,j])$ taka, że $c' \in \mathcal{E}_F$, $c||c'$ i $c \in alph(h(c'))$.

Dla słowa $w[i,j]$ rozszerzającego się pod wpływem morfizmu h, jedna z liter występujących najrzadziej jest literą rozszerzającą.

Właściwości liter najrzadziej występujących

Examples

Słowo w takie, że $|w|_a = 1$ i $w \neq a$ jest morficznie nieprymitywne. Możemy zdefiniować bowiem $\mathcal{E}_h = \{a\}$ i $h(a) = w$.
Wiemy, że:

- $0 \in \mathcal{L} \cap \mathcal{R}$, ponieważ $|h(w[0,0])| = |h(\varepsilon)| = 0$.
- $|w| \in \mathcal{L} \cap \mathcal{R}$, ponieważ $|h(w[0,|w|])| = |h(w)| = |w|$.
- $a||b$ dla każdego b takiego, że $|w|_b = 1$.

Z Lematu 4 zatem albo a, albo dowolna inna występująca w w pojedyncza litera jest literą rozszerzającą.

$\mathcal{L}$ i $\mathcal{R}$ są zależne of $\mathcal{E}_F$, co powoduje pewną niejednoznaczność.

Właściwości liter najrzadziej występujących

Lemma (5)

Niech $F = (w_1, w_2 \dots w_k)$ będzie faktoryzacją słowa w. Wtedy możemy wybrać zbiór $\mathcal{E}_F$ jako zestaw reprezentantów z kolekcji zbiorów $\mu(w_1), \mu(w_2), \dots, \mu(w_k)$.
Co więcej, dla każdego $i \in \{1, \dots, k\}$ elementy $\mu(w_i)$ są parami bliźniacze.

Dowód:
Bezpośrednie wynikanie z Lematu 4.

W celu ujednoznacznienia wyboru $\mathcal{E}_F$ w algorytmie zdefiniujemy standardowy zbiór liter rozszerzających jako zbiór takich liter a_i, że należą one do $\mu(w_i)$ i występują na najwcześniejszym indeksie od lewej w słowie w_i (w stosunku do pozostałych z $\mu(w_i)$).

Sprawdzenie istnienia faktoryzacji jest problemem NP

Jeśli mamy dostępny zbiór liter rozszerzających $\mathcal{E}$ danej faktoryzacji, to możemy zapisać słowo w jako

$$w = z_0 a_1 z_1 a_2 z_2 \dots z_{k-1} a_k z_k$$

gdzie $a_1, \dots, a_k \in \mathcal{E}^*$ i $alph(z_0 \dots z_k) \cap \mathcal{E} = \emptyset$. Aby wyznaczyć faktoryzację wystarczy w odpowiedni sposób podzielić słowa $z_0, \dots, z_k$ tak, aby ten podział spełniał kryteria poprawnej faktoryzacji.

Wyznaczenie podziału jest trywialne, ponieważ znamy litery rozszerzające. Głównym problemem zatem jest znalezienie $\mathcal{E}$.

Pozycje w słowie a litery rozszerzające

Lemma (6)

Dla danej faktoryzacji F i zbioru liczb rozszerzających $\mathcal{E}_F$:

$$w[i, i+1] = a \ \wedge \ a \in \mathcal{E}_F \Rightarrow i \in \mathcal{L} \ \wedge \ i+1 \in \mathcal{R}.$$

Dowód:
Wynika bezpośrednio z definicji $\mathcal{L}$ i $\mathcal{R}$, ponieważ a jest literą rozszerzającą.

Pozycje w słowie a litery rozszerzające

Lemma (7)

Jeśli $w[i,j] = \mathbf{n}_a$, to dla każdej faktoryzacji F takiej, że $a \in \mathcal{E}_F$ spełnione będzie $i \in \mathcal{R}$ oraz $j \in \mathcal{L}$.

Dowód:
Załóżmy, że istnieje faktoryzacja F taka, że $i \notin \mathcal{R}$. Z tego wynika, że $|h(w[0,i])| < i$, czyli i nie jest pozycją litery rozszerzającej a.
Istnieje zatem litera b taka, że $\mathbf{l}_a = u_1 b u_2$, $u_1, u_2 \in \Sigma^*$.
Ponieważ $|h(w[0,i])| < i$, to faktoryzacja zawiera słowo tu_1bv, gdzie t jest niepuste, $v \cap \mathcal{E}_F \neq \emptyset$.
To implikuje, że sąsiedztwo $\mathbf{n}_a$ powinno zostać rozszerzone do postaci $t\mathbf{n}_a$, co jest sprzeczne z definicją $\mathbf{n}_a$ jako maksymalnego sąsiedztwa. Zatem założenie jest sprzeczne, $i \in \mathcal{R}$.

Dowód dla $j \in \mathcal{L}$ przeprowadzamy analogicznie.

Pozycje w sąsiedztwach tej samej litery

Lemma (8)

Niech F będzie faktoryzacją słowa w, a literą należącą do $\mathcal{E}_F$ i niech $w[i,j] = w[i',j'] = \mathbf{n}_a$. Jeśli $i + k \in \mathcal{L}$ (analogicznie $i + k \in \mathcal{R}$), z założeniem $i \leq i + k \leq j$, to $i' + k \in \mathcal{L}$ (analogicznie $i' + k \in \mathcal{R}$).

Dowód:
Niech h będzie stabilnym morfizmem wyznaczającym faktoryzację F z $\mathcal{E}_h = \mathcal{E}_F$. Przyjmijmy

$$w[m, m+1] = w[m', m'+1] = a$$

dla pewnych pozycji $i \leq m \leq j$, $i' \leq m \leq j'$. Zauważmy, że $m \neq m'$ oraz

$$m - (i + k) = m' - (i' + k) \tag{1}$$

Ponieważ $a \in \mathcal{E}_h$ mamy

$$|h(w[0, m])| - m = |h(w[0, m'])| - m' \tag{2}$$

Pozycje w sąsiedztwach tej samej litery

Lemma (8)

Niech F będzie faktoryzacją słowa w, a literą należącą do $\mathcal{E}_F$ i niech $w[i,j] = w[i',j'] = \mathbf{n}_a$. Jeśli $i + k \in \mathcal{L}$ (analogicznie $i + k \in \mathcal{R}$), z założeniem $i \leq i + k \leq j$, to $i' + k \in \mathcal{L}$ (analogicznie $i' + k \in \mathcal{R}$).

Dowód:
Wiemy, że $w[i + k, m] = w[i' + k, m']$ dla $i + k \leq m$ i $w[m, i + k] = w[m', i' + k]$ wpp. Z tego wnisokujemy, że

$$|h(w[0, m])| - |h(w[0, i + k])| = |h(w[0, m'])| - |h(w[0, i' + k])| \tag{3}$$

Podstawiając zależności 1 i 2 do 3 otrzymujemy

$$|h(w[0, i + k])| - (i + k) = |h(w[0, i' + k])| - (i' + k)$$

co było do udowodnienia.

Algorytm wyznaczania faktoryzacji dla słowa

Dla danego słowa w i $E \subset alph(w)$ wyznaczamy $L(E) \subset \mathcal{C}_w$ oraz $R(E) \subset \mathcal{C}_w$ jako najmniejsze zbiory spełniające następujące założenia:

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) jeśli $w[i, i+1] \in E$, to $i \in L(E)$ oraz $i+1 \in R(E)$,

(c) jeśli $w[i,j] = \mathbf{n}_a$ dla pewnego $a \in E$, to $i \in R(E)$ oraz $j \in L(E)$,

(d) jeśli $w[i,j] = w[i',j'] = \mathbf{n}_a$ dla pewnego $a \in E$, to
- $i + k \in L(E)$, gdzie $i \leq i + k \leq j$, implikuje $i' + k \in L(E)$; oraz
- $i + k \in R(E)$, gdzie $i \leq i + k \leq j$, implikuje $i' + k \in R(E)$.

W celu skontruowania $L(E)$ i $R(E)$ używamy założeń (a)-(c), a następnie weryfikujemy czy (d) pozostaje spełnione. Jeśli żadne założenie nie wpłynie na poszerzenie zbiorów, konstrukcja została skończona.

Mając zbiory $L, R \subset \mathcal{C}_w$ definiujemy $E(L,R) \subset alph(w)$ jako najmniejszy zbiór spełniający:

(e) jeśli $i < j$, $i \in L$ oraz $j \in R$, to $E(L,R)$ zawiera literę z $\mu(w[i,j])$ występującą najwcześniej z lewej w $w[i,j]$.

Algorytm wyznaczania faktoryzacji dla słowa

```
FixedPoint(w)
1  E ← ∅;
2  repeat
3     L ← L(E); R ← R(E);
4     E ← E(L, R);
5   until L = L(E) and R = R(E);
6  return E, L, R;
```

Algorytm wyznaczania faktoryzacji dla słowa

Mając zbiór E możemy zapisać

$$w = z_0 a_1 z_1 a_2 z_2 \dots z_{k-1} a_k z_k$$

gdzie $k = |w|_E$. Szukamy zatem słów $u_1, v_1, u_2, v_2, \dots, u_{k-1}, v_{k-1}$ takich, że:

- $u_i v_i = z_i$,
- jeśli $a_i = a_j$, to $u_{i-1} a_i v_i = u_{j-1} a_j v_j$ (gdzie $u_0 = z_0$ i $v_k = z_k$).

Podziału możemy dokonać w taki sposób, że u_i zawiera te pozycje z z_i, które należą do zbioru R, a v_i zawiera te pozycje z z_i, które należą do zbioru L.
W szczególności możemy zdefiniować u_i jako maksymalny prefiks z_i taki, że $|z_0 a_1 z_1 a_2 \dots z_{i-1} a_i z_i| \in R$. Taki prefiks zawsze istnieje, bo założenie spełnia słowo puste.

Algorytm wyznaczania faktoryzacji dla słowa

`MorphicFactorization(`w`)`

1 $E, L, R \leftarrow$ `FixedPoint(`w`)`;
2 $k \leftarrow |w|_E$;
3 **if** $E = alph(w)$
4 **then return** `Primitive`;
5 **then return** `Imprimitive`;
6 $(z_0, a_1, z_1, a_2, \ldots, z_{k-1}, a_k, z_k) \leftarrow$ słowa z rozkładu
7 **for** $i = 1, \ldots, k-1$
8 **do** $u_i \leftarrow$ maksymalny prefix z_i taki, że $|z_0 a_1 z_1 a_2 \ldots z_{i-1} a_i z_i| \in R$;
9 $v_i \leftarrow u_i^{-1} z_i$;
10 **return** $(z_0 a_1 u_1, v_1 a_2 u_2, \ldots, v_{k-1} a_k z_k)$;

Algorytm – przykład 1

Examples

$w = caabaaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Pierwsze przejście: $E = \emptyset$,

- z (a): $L(\emptyset) = \{0, 12\}$, $R(\emptyset) = \{0, 12\}$,
- z (e): $E(\{0, 12\}, \{0, 12\}) = \{c\}$ bo $\mu(w[0, 12]) = \{b, c\}$.

$E = \{c\}$, $L = \{0, 12\}$, $R = \{0, 12\}$

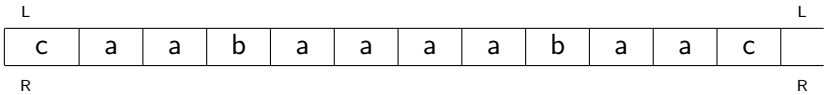
(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.



Algorytm – przykład 1

Examples

$w = caabaaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Drugie przejście:

- z (b): $c \in E \Rightarrow 0,11 \in L,\ 1,12 \in R$,
- z (c): $1,12 \in L,\ 0,11 \in R$,
- z (e): $b \in E$ bo $\mu(w[1,11]) = \{b\}$.

$E = \{b, c\},\ L = \{0, 1, 11, 12\},\ R = \{0, 1, 11, 12\}$

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i+k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.

L	L									L	L
c	a	a	b	a	a	a	a	b	a	a	c	
R	R									R	R

Algorytm – przykład 1

Examples

$w = caabaaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Trzecie przejście:

- z (b): $b \in E \Rightarrow 3, 8 \in L,\ 4, 9 \in R$,
- z (c): $6, 11 \in L,\ 1, 6 \in R$,
- (d) spełnione, (e) nie powoduje rozszerzenia E.

$E = \{b, c\}$, $L = \{0, 1, 3, 6, 8, 11, 12\}$,
$R = \{0, 1, 4, 6, 9, 11, 12\}$

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,
(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,
(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,
(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.

	L	L		L			L		L			L	L
	c	a	a	b	a	a	a	a	b	a	a	c	
	R	R		R			R		R			R	R

Algorytm – przykład 1

Examples

$w = caabaaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

(a) $u_0 = z_0$, $v_k = z_k$,
(b) $u_i v_i = z_i$, u_i pozycje z R, v_i pozycje z L

$E = \{b, c\}$, $L = \{0, 1, 3, 6, 8, 11, 12\}$, $R = \{0, 1, 4, 6, 9, 11, 12\}$

c	a	a	b	a	a	a	a	b	a	a	c
a_1	z_1		a_2	z_2				a_3	z_3		a_4

$u_0 = z_0 = \varepsilon$, $u_1 = \varepsilon, v_1 = aa$, $u_2 = aa, v_2 = aa$, $u_3 = aa, v_3 = \varepsilon$, $v_4 = z_4 = \varepsilon$

np. $u_2 = aa$ bo $|caabaa| = 6 \in R$, a $|caabaaa| = 7 \notin R$
więc

$w_1 = u_0 a_1 u_1 = c$, $w_2 = v_1 a_2 u_2 = aabaa$, $w_3 = v_2 a_3 u_3 = aabaa$, $w_4 = v_3 a_4 z_4 = c$
$F = (c, aabaa, aabaa, c)$, $\mathcal{E} = \{b, c\}$.

$h(a) = \varepsilon$, $h(b) = aabaa$, $h(c) = c$.

Algorytm – przykład 2

Examples

$w = caabaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Pierwsze przejście: $E = \emptyset$,

- z (a): $L(\emptyset) = \{0, 11\}$, $R(\emptyset) = \{0, 11\}$,
- z (e): $E(\{0, 11\}, \{0, 11\}) = \{c\}$ bo $\mu(w[0, 11]) = \{b, c\}$.

$E = \{c\}$, $L = \{0, 11\}$, $R = \{0, 11\}$

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.

L											L
c	a	a	b	a	a	a	b	a	a	c	
R											R

Algorytm – przykład 2

Examples

$w = caabaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Drugie przejście:

- z (b): $c \in E \Rightarrow 0, 10 \in L,\ 1, 11 \in R$,
- z (c): $1, 11 \in L,\ 0, 10 \in R$,
- z (e): $b \in E$ bo $\mu(w[1, 10]) = \{b\}$.

$E = \{b, c\}$, $L = \{0, 1, 10, 11\}$, $R = \{0, 1, 10, 11\}$

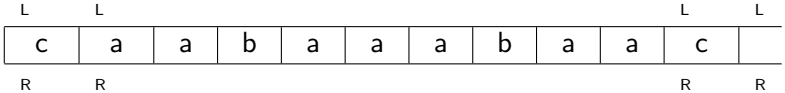
(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.



Algorytm – przykład 2

Examples

$w = caabaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Trzecie przejście:

- z (b): $b \in E \Rightarrow 3, 7 \in L,\ 4, 8 \in R$,
- z (c): $6, 10 \in L,\ 1, 5 \in R$.

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

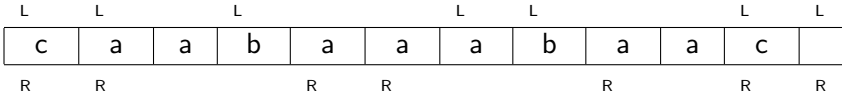
(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.

$E = \{b, c\},\ L = \{0, 1, 3, 6, 7, 10, 11\},\ R = \{0, 1, 4, 5, 8, 10, 11\}$



Algorytm – przykład 2

Examples

$w = caabaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Trzecie przejście cd.:

- z (d): ponieważ $w[1,6] = w[5,10] = \mathbf{n}_b$, to:
  - $1 \in L \cap R \Rightarrow 5 \in L \cap R \Rightarrow 9 \in L \cap R$
  - $10 \in L \cap R \Rightarrow 6 \in L \cap R \Rightarrow 2 \in L \cap R$
- z (e): $a \in E$, bo
  $\mu(w[1,2]) = \mu(w[5,6]) = \mu(w[9,10]) = a$.

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i,j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i,j] = w[i',j'] = \mathbf{n}_a, \forall k : i \leq i+k \leq j$:
- $i+k \in L(E) \Rightarrow i'+k \in L(E)$,
- $i+k \in R(E) \Rightarrow i'+k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L,R)$ zawiera literę z $\mu(w[i,j])$ występującą najwcześniej z lewej.

$E = \{a, b, c\}$, $L = \{0, 1, 2, 3, 5, 6, 7, 9, 10, 11\}$, $R = \{0, 1, 2, 4, 5, 6, 8, 9, 10, 11\}$

	L	L	L	L		L	L	L		L	L	L
	c	a	a	b	a	a	a	b	a	a	c	
	R	R	R		R	R	R		R	R	R	R

Algorytm – przykład 2

Examples

$w = caabaaabaac$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aabaa$, $\mathbf{n_c} = c$

Czwarte przejście:
Mamy $E = \{a, b, c\}$, więc korzystając z (b) otrzymujemy $L = R = \{0, 1, \ldots, 11\}$.

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

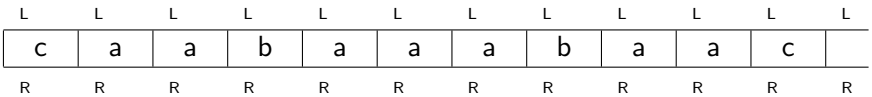
(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i, j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i, j] = w[i', j'] = \mathbf{n}_a, \forall k : i \leq i + k \leq j$:
- $i + k \in L(E) \Rightarrow i' + k \in L(E)$,
- $i + k \in R(E) \Rightarrow i' + k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L, R)$ zawiera literę z $\mu(w[i, j])$ występującą najwcześniej z lewej.

Słowo w jest morficznie prymitywne.



Algorytm – przykład 3

Examples

$w = caabcaab$, $\mathbf{n}_a = a$, $\mathbf{n}_b = aab$, $\mathbf{n_c} = caa$

(a) $u_0 = z_0$, $v_k = z_k$,
(b) $u_i v_i = z_i$, u_i pozycje z R, v_i pozycje z L

$E = \{b, c\}$, $L = \{0, 3, 4, 7, 8\}$, $R = \{0, 1, 4, 5, 8\}$

c	a	a	b	c	a	a	b
a_1	z_1		a_2	a_3	z_3		a_4

$u_0 = z_0 = \varepsilon$, $u_1 = \varepsilon, v_1 = aa$, $u_2 = \varepsilon, v_2 = \varepsilon$, $u_3 = \varepsilon, v_3 = aa$, $v_4 = z_4 = \varepsilon$
$w_1 = u_0 a_1 u_1 = c$, $w_2 = v_1 a_2 u_2 = aab$, $w_3 = v_2 a_3 u_3 = c$, $w_4 = v_3 a_4 z_4 = aab$

$F = (c, aab, c, aab)$, $\mathcal{E} = \{b, c\}$

Istnieje więcej poprawnych faktoryzacji słowa w – np. (ca, ab, ca, ab) lub (caa, b, caa, b).

Poprawność algorytmu

Theorem (9)

Niech w będzie słowem, E, L, R wynikiem działania algorytmu `FixedPoint(w)`*, a F wynikową faktoryzacją algorytmu* `MorphicFactorization(w)`*.*

(1) *F jest faktoryzacją słowa w taką, że $\mathcal{E}_F = E$.*

(2) *Niech F' będzie faktoryzacją w i $\mathcal{E}_{F'}$ jej zbiorem liter rozszerzających. Wtedy $E \subseteq \mathcal{E}_{F'}$, $L \subseteq \mathcal{L}(F', \mathcal{E}_{F'})$ oraz $R \subseteq \mathcal{R}(F', \mathcal{E}_{F'})$.*

W szczególności, w jest morficznie prymitywne wtw. $E = alph(w)$.

Dowód:
(1) Oznaczamy $v_0 = z_0$, $v_k = z_k$. Ze sposobu konstrukcji faktoryzacji wprost wynika, że $|v_{i-1}a_i v_i|_E = 1$ dla każdego $i = 1, \ldots, k$. Wystarczy wykazać, że

$$a_j = a_m \Rightarrow v_{j-1} = v_{m-1} \wedge u_j = u_m$$

Poprawność algorytmu

$$a_j = a_m \Rightarrow v_{j-1} = v_{m-1} \wedge u_j = u_m$$

1. u_j i u_m są prefiksami $\mathbf{r}_a$.

Dowód nie wprost: któreś z nich nie jest. Załóżmy, że u_j nie jest prefiksem $\mathbf{r}_a$, zatem $\mathbf{r}_a$ jest prefiksem u_j.
Oznaczmy $a_j = a_m = a$ i zdefiniujmy $d_i = |z_0 a_1 z_1 a_2 \dots z_{i-1} a_i|$ dla $i \in \{1, 2, \dots, k\}$.
Wtedy z założenia (c): $w[d_j, d_j + |\mathbf{r}_a|] = \mathbf{r}_a \Rightarrow d_j + |\mathbf{r}_a| \in L$.
Z definicji u: $d_j + |u_j| \in R$.
Stąd z Lematu 4 słowo $w[d_j + |\mathbf{r}_a|, d_j + |u_j|]$ powino być czynnikiem rozszerzającym, ale z definicji nie zawiera ono żadnej litery rozszerzającej, co stoi w sprzeczności z (e) i $E = E(L, R)$.

Poprawność algorytmu

$$a_j = a_m \Rightarrow v_{j-1} = v_{m-1} \wedge u_j = u_m$$

2. $a_j = a_m \Rightarrow u_j = u_m$

Skoro u_j i u_m są prefiksami $\mathbf{r}_a$, to są również prefiksami słowa $z_i a_{i+1} z_{i+1} \dots a_k z_k$ dla $i = j$ oraz $i = m$.
Z tego wynika, że u_j i u_m są prefiksami z_j i z_m, ponieważ żadne z u_j, u_m nie zawiera litery rozszerzającej.
Z warunku (d) oraz definicji u_j i u_m wyprowadzamy $u_j = u_m$.

Dowód dla $v_{j-1} = v_{m-1}$ przeprowadzamy analogicznie.

Poprawność algorytmu

(2) $E \subseteq \mathcal{E}_{F'}, L \subseteq \mathcal{L}(F', \mathcal{E}_{F'}), R \subseteq \mathcal{R}(F', \mathcal{E}_{F'})$ dla każdej poprawnej faktoryzacji F'

Algorytm `FixedPoint` dodaje elementy do zbiorów według warunków (a)-(e).

Warunek (a) wynika z definicji $\mathcal{L}$ i $\mathcal{R}$, dla których zawsze zachodzi $0 \in \mathcal{L} \cap \mathcal{R}$ i $|w| \in \mathcal{L} \cap \mathcal{R}$.
Warunki (b), (c), (d) odpowiadają Lematom 6, 7 i 8.
Warunek (e) wynika z Lematu 4.

Wynika z tego, że wszystkie operacje dodania do zbiorów E, L, R są wymuszone przez zachodzące dla $\mathcal{E}$, $\mathcal{L}$, $\mathcal{R}$ zależności.

Dodatkowo, $E = alph(w)$ wyznacza prymitywną faktoryzację, co kończy dowód.

Złożoność algorytmu wyznaczania faktoryzacji

Theorem (10)

Złożoność czasowa algorytmu `MorphicFactorization` *wynosi*

$$O((m + \log n) \cdot n),$$

gdzie $n = |w|$, *a* m *jest mocą zbioru* $alph(w)$.

Złożoność algorytmu wyznaczania faktoryzacji

```
MorphicFactorization(w)
1  E, L, R ← FixedPoint(w);
2  k ← |w|_E;
3  if E = alph(w)
4    then return Primitive;
5    then return Imprimitive;
6  (z_0, a_1, z_1, a_2, ..., z_{k-1}, a_k, z_k) ← słowa z (4)
7  for i = 1, ..., k − 1
8    do u_i ← maksymalny prefix z_i taki, że |z_0 a_1 z_1 a_2 ... z_{i-1} a_i z_i| ∈ R;
9       v_i ← u_i^{-1} z_i;
10 return (z_0 a_1 u_1, v_1 a_2 u_2, ..., v_{k-1} a_k z_k);
```

Złożoność algorytmu wyznaczania faktoryzacji

```
FixedPoint(w)
1  E ← ∅;
2  repeat
3     L ← L(E); R ← R(E);
4     E ← E(L, R);
5   until L = L(E) and R = R(E);
6  return E, L, R;
```

(a) $0, |w| \in L(E)$ oraz $0, |w| \in R(E)$,

(b) $w[i, i+1] \in E \Rightarrow i \in L(E) \wedge i+1 \in R(E)$,

(c) $\exists a \in E : w[i,j] = \mathbf{n}_a \Rightarrow i \in R(E) \wedge j \in L(E)$,

(d) $\exists a \in E : w[i,j] = w[i',j'] = \mathbf{n}_a, \forall k : i \leq i+k \leq j$:

- $i+k \in L(E) \Rightarrow i'+k \in L(E)$,
- $i+k \in R(E) \Rightarrow i'+k \in R(E)$.

(e) $i < j, i \in L, j \in R \Rightarrow E(L,R)$ zawiera literę z $\mu(w[i,j])$ występującą najwcześniej z lewej.

- (a) – sprawdzenie w $\mathbf{O(1)}$.
- (b) – dla każdej nowej litery w E, pozycje przy wszystkich jej wystąpieniach w w – $\mathbf{O(n)}$.
- (c) – dla każdej nowej litery w E, wyznacza indeksy obejmujące jej sąsiedztwa w w – $\mathbf{O(n)}$.
- (d) – wymaga skonstruowania grafu, gdzie pozycje (węzły) łączymy krawędziami wg. założeń (d). W każdej iteracji dodajemy $(|\mathbf{n}_a|+1)(|w|_a - 1)$ krawędzi indukowanych nową literą a w E, a więc – $\mathbf{O(n)}$.
- (e) – dla każdej pozycji z L sprawdzamy, czy po niej występuje litera z E a następnie pozycja z R; pozycji jest n – $\mathbf{O(n)}$.

Złożoność algorytmu wyznaczania faktoryzacji

```
FixedPoint(w)
1  E ← ∅;
2  repeat
3     L ← L(E); R ← R(E);
4     E ← E(L, R);
5   until L = L(E) and R = R(E);
6  return E, L, R;
```

```
MorphicFactorization(w)
1  E, L, R ← FixedPoint(w);
2  k ← |w|_E;
3  if E = alph(w)
4     then return Primitive;
5     then return Imprimitive;
6  (z_0, a_1, z_1, a_2, ..., z_{k-1}, a_k, z_k) ← słowa z (4)
7  for i = 1, ..., k − 1
8     do u_i ← max prefix z_i zawierający się w R;
9        v_i ← u_i^{-1} z_i;
10 return (z_0 a_1 u_1, v_1 a_2 u_2, ..., v_{k-1} a_k z_k);
```

- Złożoność jednego sprawdzenia warunków: $O(1) + 4 \cdot O(n) = \mathbf{O(n)}$.
- Sprawdzenie wykonujemy, gdy dodajemy literę do E, a więc cała pętla: $\mathbf{O(n \cdot m)}$.
- Utrzymywanie grafu z (d) wymagana łączenia fragmentów nowo utworzonymi krawędziami – algorytm *find-union* dla wszystkich iteracji – $\mathbf{O(n \cdot \log n)}$.
- Pozostała część (for) algorytmu `MorphicFactorization` – $O(n)$.
- Ostatecznie:

$$O(n \cdot m) + O(n \cdot \log n) = \mathbf{O((m + \log n) \cdot n)}$$

Koniec

Dziękuję

Bibliografia


Štěpán Holub (2009)
Polynomial-time algorithm for fixed points of nontrivial morphisms
Discrete Mathematics vol. 309, no. 16, 5069–5076.