Matematicko-fyzikální fakulta

Univerzity Karlovy v Praze

Diplomová práce

Vícehlavé konečné automaty s kameny

Michal Koucký

Duben 1998

©Všechna práva vyhrazena

Autor tímto uděluje právo Matematicko-fyzikální fakultě Univerzity Karlovy reprodukovat a distribuovat obsah této práce pro akademické účely v jakékoliv formě a jakékoliv podobě a to jak v celku, tak po částech.

Vedoucí diplomové práce: RNDr. Václav Koubek, DrSc.
Katedra teoretické informatiky

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a použil jsem pouze uvedenou literaturu.

V Praze, 17. dubna 1998

Michal Koucký

Obsah

Kapitola 1

Úvod

V této práci se zabýváme studiem k-hlavých jednosměrných automatů s h kameny. Jedná se o zařízení s konečným počtem vnitřních stavů, s jednou čtecí vstupní páskou zakončenou symbolem \$, s k jednosměrnými čtecími hlavami a h kameny, které mohou být hlavami pokládány podél vstupní pásky a mohou být zvedány. Studujeme zde variantu automatů, při které čtecí hlavy mohou detekovat přítomnost (*koincidenci*) dalších hlav na témže políčku pásky, i variantu, při které tuto přítomnost přímo detekovat nemohou (*bez koincidence hlav*). Třídu jazyků rozpoznávaných k-hlavými jednosměrnými automaty s h kameny označíme pro dané k a h jako $R_{k,h}$. Dále se zde zabýváme deterministickou variantou těchto automatů a třídy jejich jazyků označíme jako $R_{k,h}^D$. Variantě automatů bez koincidence hlav odpovídají třídy jazyků $R_{k,h}^N$ a $R_{k,h}^{ND}$.

Ideu vícehlavých konečných automatů zavedl Piatkowski [11] a následně je zevrubně studoval Rosenberg [14]. Rosenberg mimo jiné ukazoval, že k-hlavý konečný automat má větší výpočetní sílu než $k+1$-hlavý. Jak však později ukázal Floyd [3] důkaz Rosenberga byl neúplný. Skutečný důkaz faktu $R_{k,h} \subsetneq R_{k+1,h}$ nalezl až Yao a Rivest [13]. Yao a Rivest dále ukázali, že $R_{2,0} - \bigcup_{k\in\mathbb{N}} R_{k,0} \neq \emptyset$, tedy že nedeterministické automaty jsou striktně silnější než deterministické.

Vícehlavými dvousměrnými automaty s kameny se pak zabývala práce Ritchieho a Springsteela [12], ve které studovali vztah těchto automatů ke známým třídám jazyků.

Vztah mezi automaty s koincidencí a bez ní, tedy že vícehlavé automaty s koincidencí jsou silnější než automaty bez koincidence, ukázal na jazyce $\left\{0^{n^2} \mid n \in \mathbb{N}\right\}$ Sudborough [15]. Uvedený jazyk lze rozpoznávat tříhlavým konečným automatem s koincidencí, ale žádným vícehlavým konečným automatem bez koincidence.

V této práci se zabýváme vztahy mezi třídami jazyků $R_{k,h}$, $R^D_{k,h}$, $R^N_{k,h}$ a $R^{ND}_{k,h}$ pro různá k a h. Dále se zabýváme uzávěrovými vlastnostmi těchto tříd vůči operacím s jazyky. Mimo to zde ukazujeme Větu O lustrování, která jistým způsobem charakterizuje chování automatů při přijímacích výpočtech.

Ukážeme zde pro $k \geqslant 2$ vztahy $R_{k,h} \subsetneq R_{k+1,h}$ a $R_{k,h} \subsetneq R_{k,h+1}$ a také vztah $R_{1,h} \subsetneq R_{2,h}$. Tyto vztahy platí zvlášť také pro třídy $R^D_{k,h}$, $R^N_{k,h}$ a $R^{ND}_{k,h}$ a dokládají, že s počtem hlav i kamenů ostře roste rozpoznávací síla vícehlavých konečných automatů s kameny. Pro $k = 1$ pak platí, že $R^{ND}_{1,0} = \bigcup_{h\in\mathbb{N}} R_{1,h}$, tedy že jednohlavé konečné automaty s kameny rozpoznávají právě třídu regulárních jazyků.

Dále zde ukážeme vztah $R^N_{2,0} - \bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R^D_{k,h} \neq \emptyset$, který dokládá, že nedeterministické automaty jsou striktně silnější než deterministické. Toto tvrzení plyne mimo jiné z výše uvedeného tvrzení Yao a Rivesta, my zde však pro jeho důkaz použijeme jazyk palindromů ($\{x\#x^R |\ x \in \{0,1\}^*\}$.

Dále zde studujeme vztah mezi automaty s koincidencí a bez koincidence a ukazujeme, že pro $k \geqslant 4$ je $R^{N(D)}_{k,h} \subsetneq R^{(D)}_{k,h}$, neboli že automaty s koincidencí jsou pro daný počet hlav a kamenů striktně silnější než automaty bez koincidence. Pro $k = 2$ a $h \geqslant 1$ je zde naopak ukázáno, že $R^{N(D)}_{2,h} = R^{(D)}_{2,h}$. Mimo to zde ukážeme, že $R^{(D)}_{k,h} \subsetneq R^{N(D)}_{k,h+1}$, což znamená, že schopnost přímo testovat koincidenci hlav lze nahradit jedním kamenem navíc.

Dále zde ukazujeme, že pro $h' \leqslant h$ je $R_{k,h} \subseteq R_{k+\lfloor k/2\rfloor(h-h'),h'}$, neboli že chybějící kameny lze nahradit hlavami. Opačné tvrzení však neplatí, tedy chybějící hlavy nelze nahradit žádným počtem kamenů, to jest $R_{k+1,h} - \bigcup_{h'\in\mathbb{N}} R_{k,h'} \neq \emptyset$. Z uvedených vztahů vyplývá, že $\bigcup_{k\geqslant 1} R_{k,0} = \bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R_{k,h} = \bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R^N_{k,h}$, a že tedy jazyky rozpoznávané vícehlavými konečnými automaty s kameny jsou právě jazyky rozpoznávané vícehlavými konečnými automaty s koincidencí.

Z uzávěrových vlastností tříd $R_{k,h}$ se zde zabýváme uzavřeností těchto tříd na následující operace nad jazyky: průnik s regulární množinou, doplněk, průnik dvou jazyků, sjednocení, reverzi, konkatenaci a neomezenou iteraci jazyků.

1.1 Značení a konvence

V následujícím textu písmeno k obvykle označuje počet hlav automatu a písmeno h obvykle označuje počet kamenů automatu. Písmeno c označuje obvykle konfiguraci automatu, případně označuje nějakou konstantu. Písmena p, q obvykle označují stavy automatu, písmeno s někdy označuje počet stavů automatu. Písmena a, b označují obvykle znaky vstupní abecedy automatu, písmena u, v, w, x, y obvykle označují slova nad vstupní abecedou. Stejně tak znaky α, β označují obvykle slova nad vstupní abecedou. Písmeno r obvykle označuje instrukci automatu. Kterékoliv malé písmeno latinské abecedy pak může být příležitostně použito pro označení celého nebo přirozeného čísla. Písmena L, X, W, K se používají pro označení množin slov – jazyků. Písmeno V se používá pro označení výpočtů. Písmeno H se používá pro označení množiny pozic kamenů.

Množina přirozených čísel se označuje symbolem $\mathbb{N}$.

Symbol ϵ označuje prázdné slovo.

Pro reálné číslo o označuje symbol $\lfloor o \rfloor$ nejbližší nižší celé číslo k číslu o a symbol $\lceil o \rceil$ nejbližší vyšší celé číslo k číslu o. Symbol $|o|$ označuje absolutní hodnotu čísla o.

Pokud n a $k \leqslant n$ jsou přirozená čísla pak symbol $n!$ označuje faktoriál čísla n, to jest $n! = 1.2.3 \dots (n-1)\, n$ pokud $n > 0$ a $n! = 1$ pokud $n = 0$. Symbol $\binom{n}{k}$ pak označuje kombinační číslo n nad k, to jest $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Pokud α je slovo nad abecedou Σ, pak $|\alpha|$ označuje délku tohoto slova. Pro prázdné slovo ϵ platí: $|\epsilon| = 0$. Pokud také β je slovo nad abecedou Σ, pak $\alpha\beta$ označuje spojení (konkatenaci) těchto slov.

Pro slova $\alpha_{l_1}, \alpha_{l_1+1}, \dots, \alpha_{l_2}$ nad abecedou Σ a $m, n \in \{l_1, \dots, l_2\}$ označuje symbol $(\alpha_i)_{i=n}^m$ následující spojení (konkatenaci) slov α_i:

a) pokud $n \leqslant m$, pak $(\alpha_i)_{i=n}^m = \alpha_n \alpha_{n+1} \cdots \alpha_{m-1} \alpha_m$

b) pokud $n > m$, pak $(\alpha_i)_{i=n}^m = \epsilon$

Nehrozí-li nedorozumění, lze použít i zkrácený zápis $(\alpha_i)_n^m$, což je ekvivalentní zápisu $(\alpha_i)_{i=n}^m$.

Příklad: $x_1 = abc, x_2 = bc, x_3 = d$ pak $(x_i)_1^3 = abcbcd$ a $(x_i)_3^1 = \epsilon$.

Pro slovo $\alpha = a_1 a_2 \cdots a_n$ nad abecedou A, kde $a_i \in A$, označíme symbolem α^R otočení (reverzi) slova α, to jest $\alpha^R = a_n a_{n-1} \cdots a_1$.

Pokud A je abeceda a i přirozené číslo, pak symbol A^i označuje množinu všech slov délky i nad abecedou A. Dále A^+ označuje množinu všech neprázdných konečných slov nad abecedou A, tedy $A^+ = \bigcup_{j=1}^{\infty} A^j$, a symbol A^* označuje množinu všech konečných slov nad abecedou A včetně prázdného slova, tedy $A^* = A^+ \cup \{\epsilon\}$. Symbol $A^{<i}$ označuje množinu slov nad abecedou A jejichž délka je menší než i, tedy $A^{<i} = \bigcup_{0 \leqslant j < i} A^j$.

Pokud L je jazyk nad abecedou A, pak symbol $\overline{L}$ označuje doplněk tohoto jazyka, to jest $\overline{L} = A^* - L$, a symbol L^R označuje reverzi jazyka L, to jest $L^R = \left\{x^R \mid x \in L\right\}$.

Jsou-li L_1 a L_2 jazyky nad abecedou A, pak symbol L_1L_2 označuje spojení (konkatenaci) těchto jazyků, tedy $L_1L_2 = \{xy \mid x \in L_1 \;\&\; y \in L_2\}$. Pokud L je jazyk nad abecedou A, pak symbol L^0 označuje jazyk $\{\epsilon\}$, a pro každé přirozené číslo i se rekurentně definuje symbol $L^{i+1} = L^iL$. Zřejmě $L^1 = L$. Dále pak symbol L^* označuje neomezenou iteraci jazyka L, tedy $L^* = \bigcup_{i=0}^{\infty} L^i$.

Je-li H množina, pak symbol $\mathcal{P}(H)$ označuje potenční množinu množiny H, neboli množinu všech jejích podmnožin, tedy $\mathcal{P}(H) = \{H' \mid H' \subseteq H\}$.

Při výpočtech se používají následující odhady. Pokud a, b jsou celá čísla větší než dvě, pak $(a + b) \leqslant a.b$. Pokud a je celé číslo větší než tři, pak $a! > a^2$.

Pokud množina H má i prvků, pak počet jejích podmnožin velikosti j ($j \leqslant i$) lze odhadnout shora číslem $\binom{i}{j} \leqslant i^j$ a počet jejích podmožin menších než j lze shora odhadnout číslem $(i+1)^j$.

Lexikografické uspořádání na n-ticích celých čísel je definováno takto:

$$\langle a_1, \ldots, a_n\rangle < \left\langle a'_1, \ldots, a'_n\right\rangle \iff \exists i \in \{1, \ldots, n\},\ a_i < a'_i \;\&\; \forall 1 \leqslant j < i,\ a_j = a'_j.$$

Dále pak $\langle a_1, \ldots, a_n\rangle \leqslant \langle a'_1, \ldots, a'_n\rangle \iff (\langle a_1, \ldots, a_n\rangle < \langle a'_1, \ldots, a'_n\rangle$ nebo $\langle a_1, \ldots, a_n\rangle = \langle a'_1, \ldots, a'_n\rangle)$.

Kapitola 2

Základní definice a vlastnosti

V této kapitole budou zavedeny některé základní pojmy a budou zde ukázány některé základní vlastnosti vícehlavých konečných automatů s kameny.

2.1 Vícehlavé konečné automaty s kameny

V tomto oddíle bude zaveden pojem vícehlavého konečného automatu s kameny a další s tím související pojmy.

k-hlavým konečným automatem s h-kameny (zkráceně (k, h)-FA) budeme rozumět rozpoznávací zařízení, jež nabývá konečného počtu stavů, s jednou vstupní páskou, nad kterou se nezávisle, jedním směrem pohybuje k čtecích hlav. Dále má toto zařízení k dispozici celkem h kamenů, jež mohou jednotlivé hlavy pokládat vedle políček vstupní pásky a případně je naopak též zvedat. Výpočet takovéhoto zařízení probíhá v krocích a v každém kroku se provedou tyto akce: na základě toho, v jakém stavu se zařízení nachází, se vybere jedna hlava, a zjistí se, jaký symbol je zapsaný na políčku pod touto hlavou a zda u tohoto políčka leží kámen, případně zda se nad tímto políčkem nachází ještě i jiná hlava. Na základě těchto informací a stavu zařízení se buď položí, nebo zvedne kámen, případně se pohne s danou hlavou doprava na další políčko a přejde se do některého, ne nutně jiného, stavu.

Na začátku výpočtu bude na pásce zapsáno vstupní slovo zakončené na pravé straně speciálním symbolem \$, všechny hlavy budou umístěné nad nejlevějším políčkem vstupní pásky, podél pásky nebude ležet žádný kámen a automat bude ve speciálním *počátečním* stavu. Výpočet automatu skončí tehdy, když všechny hlavy dorazí na políčko se speciálním znakem \$. Pokud se automat v okamžiku ukončení výpočtu bude nacházet v jednom z tzv. *koncových* stavů řekneme, že slovo *přijal*, jinak řekneme, že slovo *nepřijal*.

Hlavy nad speciálním znakem \$ mají zakázáno pokládat kameny a pohybovat se směrem doprava.

Následuje formální definice vícehlavého konečného automatu s kameny:

Definice 2.1.1 *Nechť $k \geqslant 1$ a h jsou přirozená čísla. Pak* ***(nedeterministický) k-hlavý konečný automat s h kameny*** *je devítice $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ kde:*

1. *k je* ***počet hlav****.*

2. *h je* ***počet kamenů****.*

3. *S je konečná* ***množina stavů*** *automatu.*

4. *Σ je konečná* ***vstupní abeceda****.*

5. *$\$ \notin \Sigma$ je symbol označující na pásce pravý konec vstupního slova.*

6. *$\delta : S \times (\Sigma \cup \{\$\}) \times \{E, L\} \times \{K, N\} \longrightarrow \mathcal{P}(S \times \{0, 1\} \times \{E, L\})$ je* ***přechodová funkce****.*[1][2]

7. *$\varphi : S \longrightarrow \{1, \ldots, k\}$ je funkce pro výběr hlavy.*

8. *$p_0 \in S$ je* ***počáteční stav****.*

9. *$F \subseteq S$ je* ***množina koncových stavů****.*

Přechodová funkce δ splňuje: $\forall p, q \in S, \forall e \in \{E, L\}, \forall f \in \{K, N\},\ \langle q, \lambda, b\rangle \in \delta(\langle p, \$, e, f\rangle) \Longrightarrow (\lambda = 0 \quad \& \quad b = E)$.

Poznámka 1 *V dalším textu se písmena k a h používají převážně pro označení počtu hlav a kamenů. Nebude-li proto řečeno jinak, předpokládá se, že k a h jsou přirozená čísla, přičemž $k \geqslant 1$.*

[1] *E ... označuje, že na daném políčku neleží kámen*
L ... označuje, že na daném políčku leží kámen

[2] *K ... označuje, že se nad daným políčkem pásky nachází více hlav*
N ... označuje, že se nad daným políčkem pásky nachází pouze jedna hlava

Celkový stav (k,h)-FA po jednotlivých krocích výpočtu nad vstupním slovem x, tj. rozmístění jednotlivých hlav a kamenů nad slovem x zároveň s vnitřním stavem automatu, budeme označovat jako *konfiguraci*.

Definice 2.1.2 *Nechť $M = (k,h,S,\Sigma,\delta,\varphi,\$,p_0,F)$ je (k,h)-FA. Nechť $x \in \Sigma^*$ je slovo nad vstupní abecedou a nechť $n = |x|$ je jeho délka. Pak* ***konfigurace automatu*** *M* ***nad slovem*** *x je $(k+3)$-jice $c = \langle x\$, p, i_1, \ldots, i_k, H\rangle$, kde:*

1. *$p \in S$ je stav, ve kterém se automat M nachází.*
2. *i_j jsou pozice jednotlivých hlav nad slovem x. Platí pro ně: $\forall j \in \{1,\ldots,k\};\ i_j \in \{1,\ldots,n+1\}$.*
3. *H je množina pozic kamenů nad slovem x. Platí pro ni: $H \subseteq \{1,\ldots,n\} \quad \& \quad |H| \leqslant h$.*

Definice 2.1.3 *Konfiguraci c automatu M nad slovem $x \in \Sigma^*$ označíme jako* ***počáteční****, pokud $c = \langle x\$, p_0, 1, \ldots, 1, \emptyset\rangle$. Konfiguraci c automatu M označíme jako* ***koncovou****, pokud $c = \langle x\$, q, n, \ldots, n, H\rangle$ pro nějaké slovo $x \in \Sigma^*$, $n = |x|+1$, $H \subseteq \{1,\ldots,n-1\}$ a $q \in S$. Pokud je navíc $q \in F$, pak takovou koncovou konfiguraci označíme jako* ***přijímající****.*

Poznámka 2 *Množinu všech konfigurací (k,h)-FA M nad slovem $x \in \Sigma^*$ budeme označovat symbolem C^x_M a množinu všech konfigurací automatu M symbolem $C^*_M = \bigcup_{x\in\Sigma^*} C^x_M$. Pokud je z kontextu zřejmé, o který automat se jedná, budeme dolní index udávající automat vynechávat.*

Definice 2.1.4 *Nechť $c = \langle x\$, p, i_1, \ldots, i_k, H\rangle$ je konfigurace (k,h)-FA nad slovem x. Pak* ***typ konfigurace*** *c je $(k+2)$-jice $\widetilde{c} = \langle p, i_1, \ldots, i_k, H\rangle$.*
Pro danou konfiguraci c budeme její typ označovat symbolem $\widetilde{c}$.

Pro daný (k,h)-FA M si nyní definujeme pomocnou funkci $\widetilde{a} : C^*_M \longrightarrow S \times (\Sigma \cup \{\$\}) \times \{E,L\} \times \{K,N\}$. Nechť $c = \langle x\$, p, i_1, \ldots, i_k, H\rangle$ je konfigurace, kde $x = a_1a_2\cdots a_n \in \Sigma^n$, $a_{n+1} = \$$, a nechť $s = \varphi(p)$, pak

$$\widetilde{a}(c) = \begin{cases} \langle p, a_{i_s}, L, K\rangle & \text{když } i_s \in H \quad \& \quad \exists t \in \{1,\ldots,k\} \setminus \{s\},\ i_s = i_t \\ \langle p, a_{i_s}, E, K\rangle & \text{když } i_s \notin H \quad \& \quad \exists t \in \{1,\ldots,k\} \setminus \{s\},\ i_s = i_t \\ \langle p, a_{i_s}, L, N\rangle & \text{když } i_s \in H \quad \& \quad \forall t \in \{1,\ldots,k\} \setminus \{s\},\ i_s \neq i_t \\ \langle p, a_{i_s}, E, N\rangle & \text{když } i_s \notin H \quad \& \quad \forall t \in \{1,\ldots,k\} \setminus \{s\},\ i_s \neq i_t \end{cases}$$

Pomocí této funkce nyní nadefinujeme relaci přechodu mezi konfirguracemi.

Definice 2.1.5 *Nechť M je (k,h)-FA nad abecedou Σ.* ***Relaci přechodu*** $\vdash_M$ ***mezi konfiguracemi*** *definujeme takto:*

$$\forall c_1, c_2 \in C_M^*, \quad c_1 = \langle w_1\$, p, i_1, \ldots, i_k, H_1 \rangle \quad \& \quad c_2 = \langle w_2\$, q, j_1, \ldots, j_k, H_2 \rangle$$

$$\begin{array}{lcl} c_1 \vdash_M c_2 & \overset{def}{\Longleftrightarrow} & w_1 = w_2 \quad \& \quad \langle q, \lambda, b \rangle \in \delta\left(\widetilde{a}\left(c_1\right)\right) \quad \& \\ & & j_{\varphi(p)} = i_{\varphi(p)} + \lambda \quad \& \quad \forall l \neq \varphi\left(p\right), \ j_l = i_l \quad \& \\ & & \left(b = E \Longrightarrow H_2 = H_1 \setminus \left\{i_{\varphi(p)}\right\}\right) \quad \& \quad \left(b = L \Longrightarrow H_2 = H_1 \cup \left\{i_{\varphi(p)}\right\}\right) \end{array}$$

Tranzitivní uzávěr relace přechodu $\vdash_M$ budeme značit $\vdash_M^*$.

Nyní již přirozeným způsobem definujeme *výpočet* vícehlavého konečného automatu s kameny jako posloupnost konfigurací takovou, že jednotlivé po sobě jdoucí konfigurace splňují relaci přechodu. Dále též řekneme, že automat slovo x *přijímá*, pokud existuje výpočet takový, že jeho první konfigurací je počáteční konfigurace a poslední konfigurací je přijímající konfigurace. Pro některá slova nemusí existovat žádný výpočet daného automatu z výchozí konfigurace do konfigurace koncové. Buď by se automat v průběhu výpočtu nad takovými slovy *zacyklil*, tedy některou z konfigurací by opakoval neomezeně mnohokrát, aniž by pohnul hlavami, nebo by automat z poslední konfigurace c výpočtu nemohl přejít do žádné další a to buď z toho důvodu, že by $\delta\left(\widetilde{a}\left(c\right)\right)$ nebyla definovaná, nebo byla rovna $\emptyset$, nebo z toho důvodu, že by musel položit kámen, avšak všechny jeho kameny by již byly položeny (počet kamenů je omezený).

Dva výpočty vícehlavého automatu s kameny budeme považovat za *ekvivalentní*, pokud jejich první a poslední konfigurace budou shodné.

Definice 2.1.6 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k,h)-FA.* ***Výpočtem*** *automatu M nad slovem $x \in \Sigma^*$ nazveme každou posloupnost konfigurací $c_1, \ldots, c_l$ nad slovem x takovou, že $\forall i \in \{1, \ldots, l-1\}$, $c_i \vdash_M c_{i+1}$.*

Definice 2.1.7 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k,h)-FA. Automat M* ***přijímá*** *slovo $x \in \Sigma^*$ právě tehdy, když existuje výpočet $c_1, \ldots, c_l$ nad slovem x takový, že c_1 je výchozí konfigurace a c_l je přijímající konfigurace. Takovýto výpočet budeme označovat jako* ***přijímající****. Pokud výpočet požadovaných vlastností neexistuje, automat M slovo x* ***nepřijímá****.*

Definice 2.1.8 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k, h)-FA a nechť jazyk $L \subseteq \Sigma^*$. Řekneme, že* ***automat*** *M* ***přijímá (rozpoznává) jazyk*** *L, právě tehdy, když platí:*

$$\forall x \in \Sigma^*,\ (x \in L \Longleftrightarrow M \text{ přijímá } x)\,.$$

Výše definovaný (k, h)-FA budeme též označovat jako *nedeterministický (k, h)-FA*, neboť z každé konfigurace může obecně následně přejít do několika různých konfigurací. Naproti tomu, pokud pro kažou konfiguraci existuje právě jedna následující konfigurace (tj. relace přechodu $\vdash_M$ je zobrazení do jednoprvkových množin), budeme takový automat označovat jako *deterministický (k, h)-FA* (zkráceně (k, h)-DFA).

Definovaný automat má také tu vlastnost, že dokáže detekovat, zda čtecí hlava je nad nějakým políčkem zároveň s dalšími hlavami. Budeme jej proto také označovat jako *(k, h)-FA s koincidencí*. Pokud přechodová funkce daného (k, h)-FA nezávisí na tom, zda jsou, nebo nejsou nad políčkem pásky s danou hlavou i další hlavy, budeme automat označovat jako *(k, h)-FA bez koincidence*.

Tyto pojmy lze formálně definovat následujícím způsobem:

Definice 2.1.9 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k, h)-FA. Řekneme, že automat M je* ***deterministický****, právě tehdy, když $\forall c \in C_M^*, |\delta(\widetilde{a}(c))| = 1$.*

Definice 2.1.10 *Řekneme, že (k, h)-FA automat M je* ***bez koincidence****, právě tehdy, když $\forall p \in S, \forall a \in \Sigma \cup \{\$\}, \forall b \in \{E, L\}, \delta(p, a, b, K) = \delta(p, a, b, N)$.*

Zřejmě (k, h)-FA automaty bez koincidence tvoří vlastní třídu automatů a bylo by možné je ekvivalentně definovat tak, že by jejich přechodová funkce δ byla funkcí z množiny $S \times (\Sigma \cup \{\$\}) \times \{E, L\}$ do množiny $\mathcal{P}(S \times \{0, 1\} \times \{E, L\})$.

Definice 2.1.11 *Nechť $k \geqslant 1$ a h jsou přirozená čísla.* ***Třídu jazyků*** *$R_{k,h}$ (resp. $R_{k,h}^D$) pak definujeme takto: jazyk L patří do třídy $R_{k,h}$ (resp. $R_{k,h}^D$) právě tehdy, když existuje automat (k, h)-FA (resp. (k, h)-DFA) nad příslušnou abecedou, který rozpoznává jazyk L.*
Dále definujme ***třídu jazyků*** *$R_{k,h}^N$ (resp. $R_{k,h}^{ND}$) jako třídu těch jazyků, které lze rozpoznávat (k, h)-FA automaty bez koincidence (resp. deterministickými (k, h)-FA automaty bez koincidence.)*

Na závěr si ještě zavedmě jeden pomocný pojem týkající se konfigurací (k, h)-FA automatů.

Definice 2.1.12 *Nechť $w = x_1y_1 \cdots x_ny_n$ je slovo nad abecedou Σ, kde $x_i, y_i \in \Sigma^*$, pro všechna $i \in \{1, \dots, n\}$. Nechť $c = \langle w\$, p, i_1, \dots, i_k, H\rangle$ je konfigurace (k, h)-FA nad slovem w. Pak* ***relativním rozložením hlav a kamenů, včetně stavu, vzhledem k podslovům*** *$x_1, x_2, \dots, x_n$ nazveme $(k + 3)$-jice $\langle p, i'_1, \dots, i'_k, H', o\rangle$. takovou, že:*

1. *$\forall 1 \leqslant j \leqslant k,\ i'_j = \begin{cases} a + \sum_{1\leqslant l<b} |x_l| & \text{pokud } i_j = a + \sum_{1\leqslant l<b} (|x_l| + |y_l|)\ \&\ 0 < a \leqslant |x_b|\,. \\ -1 & \text{jinak.} \end{cases}$*
2. *$i' \in H' \iff \exists i \in H,\ i' = a + \sum_{1\leqslant l<b} |x_l|\ \&\ i = a + \sum_{1\leqslant l<b} (|x_l| + |y_l|)\ \&\ 0 < a \leqslant |x_b|$.*
3. *$o = |H| - |H'|$.*

Mimo tohoto pojmu budeme taktéž používat pojem *relativní rozložení hlav, včetně stavu, vzhledem k podslovům* a *relativní rozložení kamenů vzhledem k podslovům*, což budou restrikce na příslušné složky relativního rozložení hlav a kamenů, včetně stavu, vzhledem k daným podslovům.

Tyto pojmy slouží především pro zpřesnění a formalizaci intuitivní představy o stejném rozložení kamenů a hlav a stejném stavu v některé z konfigurací dvou výpočtů nad částečně shodnými slovy. Pokud například máme přijímající výpočty pro slova xyz a $x'yz'$, které mají konfiguraci, ve které jsou všechny hlavy nad podslovem y a zároveň relativní rozložení hlav a kamenů, včetně stavu, vzhledem k podslovu y jsou v obou konfiguracích stejné, pak, jak snadno nahlédneme, lze sestrojit přijímající výpočet i pro slovo xyz' a $x'yz$. Jedinou informací, kterou totiž automat má k dispozici o podslovu, které již prošel, je jeho stav a rozložení hlav a kamenů. (Všimněte si, že ve zmíněných konfiguracích nemůže být žádný kámen v podslově z, resp. z', a kameny, které jsou v podslově x, resp. x', jsou v dalším výpočtu nepřístupné, tedy jejich poloha je nezajímavá. Jejich počet však je důležitý, neboť spolu s počtem kamenů položených v y jednoznačně určuje počet kamenů, které jsou stále ještě k dispozici.)

2.2 Modifikace (k, h)-FA a jejich základní vlastnosti

V tomto oddíle se zmíníme o některých možných modifikacích vícehlavých automatů s kameny a zařízeních s nimi ekvivalentními.

Vícehlavý konečný automat s kameny by bylo možné též definovat tak, že by se v každém kroku automatu všechny hlavy podívaly, jaký čtou symbol, zda u nich leží kameny a zda se nad daným políčkem nacházejí samy, a na základě toho by automat poté přešel do dalšího stavu, pohnul by případně hlavami a zvedl či položil kameny. Takto definovaný automat by byl zřejmě výpočetní silou ekvivalentní námi definovanému, neboť náš automat může vždy postupně prozkoumat políčka pod svými hlavami, jejich obsah (konečné množství různých kombinací obsahů políček) si zapamatovat ve vnitřním stavu, na základě toho se rozhodnout, co udělat, a po řadě to hlavami vykonat. Při této definici by však nastával jistý problém s řešením konfliktů, kdy by se jedna hlava pokoušela kámen zvednout, zatímco jiná by jej chtěla naopak položit.

Sudborough [15] ukázal, že automaty bez koincidence nemohou rozpoznávat některé jazyky, které mohou rozpoznávat automaty s koincidencí (např. jazyk 0^{n^2}). U vícehlavých konečných automatů s kameny bez koincidence lze však pomocí kamenů testovat, zda je hlava nad políčkem vstupního slova sama nebo ne.

Tento test lze provést tak, že automat nejprve postupně otestuje, které hlavy vidí nějaké kameny, poté zkoumaná hlava kámen položí, resp. zvedne, pokud nad ním právě stála, a pak automat opět otestuje, co vidí ostatní hlavy. Hlavy, u kterých došlo ke změně, jsou zřejmě umístěny nad stejným políčkem pásky, jako je zkoumaná hlava. Poté již automat pouze uvede kameny do původního stavu. V závislosti na výsledku testu je pak automat v jednom ze dvou stavů.

Snadno nahlédneme, že pomocí této metody můžeme ke každému automatu s koincidencí sestrojit automat bez koincidence rozpoznávající stejný jazyk. Takový automat však během výpočtu bude pro test koincidence hlav obecně potřebovat mít k dispozici jeden volný kámen, tedy buď nějaký nepoložený kámen, nebo aby alespoň pod některou z hlav byl vždy před začátkem testu položen kámen (pak si ho může automat "vypůjčit" a provést test). Ve Větě O koincidenci (Věta 3.4.10) se ukazuje, že takový volný kámen nelze obecně zajistit a pro daný počet hlav a kamenů tak jsou automaty s koincidencí silnější, než automaty bez koincidence. Pokud však automatům bez koincidence povolíme jeden další kámen, získají tím schopnost koincidenci testovat a budou schopny rozpoznávat všechny jazyky jako automaty s koincidencí bez tohoto kamene navíc. Tuto skutečnost vyjadřuje následující tvrzení.

Věta 2.2.1 *Nechť $k \geqslant 1, h$ jsou přirozená čísla. Pak třída $R_{k,h} \subseteq R^{N}_{k,h+1}$ a $R^{D}_{k,h} \subseteq R^{ND}_{k,h+1}$.*

Z Věty O hierarchii (Věta 3.2.4) navíc vyplývá, že ve výše uvedeném tvrzení platí dokonce ostré inkluze.

Z definice vícehlavých konečných automatů s kameny snadno nahlédneme, že automat je schopen neustále si pamatovat pořadí svých hlav na pásce, neboť na základě aktuální znalosti tohoto rozmístění může určit rozmístění hlav v příštím kroku díky znalosti toho, zda je daná hlava nad políčkem sama, nebo s dalšími hlavami.[3] Ke každému vícehlavému automatu s kameny lze tedy sestrojit automat se stejným počtem hlav a kamenů, který rozpoznává tentýž jazyk a který si během výpočtu pamatuje aktuální pořadí svých hlav nad vstupním slovem.

Takový automat si bude ve svém stavu pamatovat kromě stavu původního automatu i pořadí hlav na pásce, nebo přesněji rozdělení hlav do skupin nad stejnými políčky a pořadí těchto skupin. Při každém posunu hlavy na další políčko pak automat vyřadí hlavu z původní skupiny (pomocný stav) a poté zjištěním, zda je hlava nad novým políčkem sama, otestuje, zda hlava nevstoupila do následující skupiny hlav. V případě, že ano, tak hlavu zařadí do této skupiny, jinak vytvoří novou jednoprvkovou skupinu. Prázdné skupiny zapomíná. Je zřejmé, že možných rozdělení hlav do skupin a možných vzájemných pořadí těchto skupin je konečně mnoho, a proto takový automat lze sestrojit.

Ke každému automatu, který si pamatuje pořadí hlav na pásce, pak můžeme dále sestrojit automat, jenž bude rozpoznávat ten samý jazyk a bude mít tentýž počet hlav i kamenů a který bude mít v průběhu výpočtu na pásce rozmístěny hlavy stále ve stejném pořadí. Toho se dosáhne jednoduše tak, že nový automat bude svými hlavami simulovat činnost hlav původního automatu a v okamžiku, kdy by se při výpočtu hlavy v původním automatu měly míjet, prohodí se význam těchto míjejících se hlav. Takový automat si tedy bude pamatovat přiřazení, která hlava v daný okamžik výpočtu simuluje kterou hlavu původního automatu. Při míjení si pak hlavy nejprve vymění svůj význam a poté se pohne ta, která má být dále vpředu. Možných vzájemných přiřazení hlav je zřejmě konečně mnoho (faktoriál v počtu hlav) a proto je taková konstrukce možná.

Varianta automatu s pevným pořadím hlav na pásce bude užitečná při simulaci chybějících

[3] Na začátku výpočtu je pořadí hlav vždy stejné.

kamenů pomocí hlav.

Pro konstrukci automatu s detekcí pořadí hlav nebo pevným pořadím hlav je potřeba použít automat s koincidencí hlav. Pokud bychom se chtěli omezit pouze na automaty bez koincidence, pak lze tuto konstrukci provést jen tehdy, pokud je k dispozici volný kámen pro testování koincidence hlav. Znalost pořadí hlav na pásce je zřejmě ekvivalentní možnosti v průběhu výpočtu testovat, zda je hlava na políčku sama.

Pokud si automat eviduje pořadí hlav, může si zároveň s tím podobným způsobem evidovat i pořadí kamenů. Pokud totiž známe celkové vzájemné pořadí hlav a kamenů, pak při položení dalšího kamene můžeme jednoznačně určit jeho pořadí mezi hlavami a ostatními kameny. Jednotlivé kameny pak tedy lze od sebe rozlišovat a každá hlava vždy ví, na který kámen narazila. Kromě toho, že je lze takto od sebe rozlišovat, mohlo by se dokonce i pomyslně pokládat více kamenů na jedno políčko. Výpočetní síla vícehlavých automatů s kameny se tedy nezmění, pokud jim přidáme možnost pokládat na jedno políčko více kamenů, či jim kameny označíme.

Je vhodné uvést zde též ten triviální fakt, že jednohlavé konečné automaty s h kameny mají pro všechna přirozená čísla h stejnou výpočetní sílu jako konečný automat, neboli rozpoznávají právě třídu regulárních jazyků.

Věta 2.2.2 *Platí, že* $R_{1,0}^{ND} = \bigcup_{h\in\mathbb{N}} R_{1,h}$*, tedy že* $\bigcup_{h\in\mathbb{N}} R_{1,h}$ *je třída všech regulárních jazyků.*

Důležitou modifikací vícehlavých konečných automatů s kameny jsou automaty, které při výpočtu nad libovolným slovem vždy dojdou do koncové konfigurace, tj. nikdy se nezacyklí ani se jim nestane, že by nemohly pokračovat ve výpočtu z důvodů nedostatku kamenů, nebo z toho důvodu, že by množina možných přechodů byla prázdná. Ke každému vícehlavému automatu s kameny lze takovýto automat jednoduše nalézt.

Hlavní myšlenka konstrukce takovéhoto automatu spočívá v tom, že původní automat doplníme o dvě počitadla. První počitadlo bude udávat, kolik kroků se již žádná z hlav nepohla. Jakmile se hlava pohne, počitadlo vynulujeme. Pokud počitadlo vzroste nad jistou mez, automat se již zacyklil a tudíž jej necháme přejít do speciálního stavu, který dopraví všechny hlavy na konec vstupního slova a slovo odmítne. Druhé počitadlo použijeme k evidenci počtu položených kamenů. Pokud by automat měl položit další kámen a přitom již všechny povolené kameny vyčerpal, přejde opět do speciálního stavu, který dopraví všechny hlavy na konec slova a slovo odmítne. Tím bude automat v případě zacyklení či nedostatku kamenů pro pokračování převeden do koncové konfigurace. Zbývá ještě přechodovou funkci rozšířit tak, aby z každé konfigurace byl možný alespoň jeden přechod do další konfigurace. To se udělá jednoduše tak, že všude, kde přechody chybí, se doplní přechod opět do speciálního stavu, který dopraví hlavy na konec slova a slovo odmítne.

Horním odhadem pro počet kroků, po který mohou zůstat všechny hlavy na místě, aniž by se automat zacyklil, je například číslo $s2^k$, kde s je počet stavů automatu a k je počet jeho hlav. Odpovídá to hornímu odhadu na počet možných rozmístění několika kamenů pod k hlavami za všech možných stavů. Po takovémto počtu konfigurací bez pohnutí hlavy by se nutně následující konfigurace, nebo již některá z předchozích, musela opakovat. Číslo $s2^k + 1$ lze tedy vzít jako horní mez přidaného počitadla kroků. Horní mez pro počitadlo položených kamenů je zřejmě h, kde h je počet kamenů.

Takto upravený automat zřejmě přijímá stejnou množinu slov jako automat původní. Každé slovo, které lze přijmout původním automatem lze přijmout i modifikovaným (pokud lze slovo přijmout, pak ho lze přijmout výpočtem v němž se neopakuje žádná konfigurace) a žádné slovo nelze přijmout navíc.

Pokud bychom vícehlavému automatu s kameny povolili pokládat neomezeně mnoho kamenů, tedy $h = \infty$, pak bychom získali zařízení označované jako *writing finite automaton*.

Kapitola 3

Hierarchie jazyků

V této kapitole je studována výpočetní síla (k,h)-FA automatů pro různá k a h a dále je zde srovnání deterministických a nedeterministických variant (k,h)-FA a taktéž variant s koincidencí a bez koincidence.

Jsou zde ukázány vztahy $R_{k,h} \subsetneq R_{k+1,h}$ a $R_{k,h} \subsetneq R_{k,h+1}$ pro $k \geqslant 2$ a také $R_{1,h} \subsetneq R_{2,h}$. Tyto vztahy jsou zde ukázány pro deterministické i nedeterministické varianty automatů a taktéž pro varianty automatů s koncidencí i bez ní. Uvedené vztahy dokládají, že s počtem hlav a kamenů ostře roste výpočetní síla vícehlavých automatů s kameny. Vztah $R_{k,0} \subsetneq R_{k+1,0}$ byl prvně dokázán Yao a Rivestem [13] a technika jejich důkazu je použita pro důkaz Věty O lustrování (Věta 3.1.1).

Je zde též ukázán vztah mezi automaty s koincidencí a bez ní, tedy pro $k \geqslant 4$ je $R^{N}_{k,h} \subsetneq R_{k,h}$ a $R^{ND}_{k,h} \subsetneq R^{D}_{k,h}$. To dokládá, že pro daný počet hlav a kamenů jsou automaty s koincidencí ostře silnější, než automaty bez koincidence. Pro $h \geqslant 1$ je zde naopak ukázáno, že $R^{N}_{2,h} = R_{2,h}$ a $R^{ND}_{2,h} = R^{D}_{2,h}$. Pro vícehlavé automaty bez kamenů byl vztah $R^{N}_{k,0} \subsetneq R_{k,0}$ a $R^{ND}_{k,0} \subsetneq R^{D}_{k,0}$, kde $k \geqslant 3$, dokázán již dříve (Harrison a Ibarra [6], Sudborough [15].)

Dále je zde ukázáno, že pro žádné h' neplatí inkluze $R_{k+1,h} \subseteq R_{k,h'}$, což znamená, že hlavu nelze nahradit žádným počtem kamenů. Je zde však ukázáno tvrzení opačné, tedy že pro $h' \leqslant h$ platí $R_{k,h} \subseteq R_{k+(h-h')\lfloor k/2 \rfloor,h'}$, neboli že kámen lze nahradit jistým počtem hlav. Důsledkem toho je tvrzení $\bigcup_{k \geqslant 1} R_{k,0} = \bigcup_{\substack{k \geqslant 1 \\ h \in \mathbb{N}}} R_{k,h}$, neboli že jazyky rozpoznávané vícehlavými konečnými automaty s kameny odpovídají jazykům rozpoznávaným vícehlavými konečnými automaty s koincidencí. Tato tvrzení platí i pro deterministické varianty automatu, tedy pro třídy $R^D_{k,h}$.

Mimo to je zde ukázáno tvrzení $R^N_{2,0} - \left(\bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R^D_{k,h}\right) \neq \emptyset$ z něhož vyplývá, že nedeterministické vícehlavé konečné automaty s kameny jsou ostře silnější než jejich deterministické varianty. Pro vícehlavé automaty bez kamenů bylo toto tvrzení dokázáno Yao a Rivestem [13].

3.1 Lustrování

Slova některých jazyků rozpoznávaných (k,h)-FA automaty lze významově rozčlenit na několik částí (podslov), přičemž některé z částí mohou být na sobě funkčně závislé. Mohou to být například identické kopie (jazyk $\{x\#x|\ x \in \{0,1\}^*\}$), nebo to mohou být zrcadlové obrazy (jazyk palindromů[1]), nebo se může jednat například o restrikci či slití několika jiných částí. Obecně jsou přípustná slova pro jednotlivé části spolu svázána nějakou relací.

Pokud má (k,h)-FA automat takový jazyk rozpoznávat, musí během přijímacích výpočtů nad takovými slovy ověřit, že části přijímaných slov požadované relace splňují. Toto ověření musí ve velkém množství případů proběhnout tak, že během výpočtu se některá z hlav automatu bude nalézat v jedné z částí a současně s tím bude další hlava v druhé z ověřovaných částí (tomu budeme říkat *lustrování* daných částí). Platnost vztahu jednotlivých částí tedy nelze pro velké množství slov ověřovat tak, že by automat prozkoumal nejprve jednu část, cosi si případně zapamatoval, a poté prozkoumal druhou z částí. Ani opakováním tohoto postupu nelze dosáhnout úplného ověření, neboť hlavy automatu se pohybují pouze jedním směrem, a automat by tedy mohl provést pouze omezený počet dílčích ověření, což může být v mnohých případech nedostatečné[2]. Automat je totiž schopen si v daný okamžik zapamatovat pomocí stavu, pozic hlav a pozic kamenů pouze informaci, která je polynomiálně velká vzhledem k délce vstupu. Pokud je možných hodnot jednotlivých částí více než polynomiálně, například tedy exponenciálně, pak nejsme zřejmě schopni opakovaným přenosem polynomiální informace přenést veškerou potřebnou informaci.

Ověření patřičných vztahů mezi jednotlivými částmi je tedy bez lustrování obvykle možné

[1] Jazyk palindromů, jak je ukázáno v Lemmatu 3.3.1, nelze rozpoznávat (k,h)-FA automaty. Příklad je zde uveden pouze pro ilustraci.

[2] Pro některé jazyky to však dostačující být může. Například jazyk $\{x\# \oplus x; x \in \{0,1\}^*\}$, kde $\oplus$ je funkce parity slova, lze rozpoznávat bez lustrování jednohlavým automatem .

pouze pro relativně malou množinu slov z přijímaného jazyka. Pro ostatní slova z přijímaného jazyka je tedy nutné provést lustrování jednotlivých částí.

V případě, že části slova na sobě nejsou závislé po dvou, ale jsou svázány více ární relací, platí toto tvrzení v té podobě, že pro většinu výpočtů nelze rozdělit na sobě závislé části slov na dvě skupiny tak, aby lustrování částí probíhalo pouze v rámci částí stejné skupiny a neprobíhalo by žádné lustrování mezi částmi z různých skupin (*„graf lustrování částí musí být souvislý“*).

Následující tvrzení tak uvádí horní odhad na počty slov, pro které není nutné provádět lustrování.

Definice 3.1.1 *Nechť $uxvyw \in \Sigma^*$. Řekněme, že (k,h)-FA automat při (během) výpočtu V nad slovem $uxvyw$ **lustruje podslova** x **a** y právě tehdy, když ve výpočtu V existuje konfigurace, ve které se jedna z hlav automatu nachází nad podslovem x a zároveň se další hlava nachází nad podslovem y slova $uxvyw$.*

Věta 3.1.1 (O lustrování) *Nechť je dán jazyk $L \subseteq \Sigma^*$ a (k,h)-FA M přijímající tento jazyk. Nechť je dána $(t+1)$-tice slov $a = \langle a_0, \ldots, a_t \rangle \in (\Sigma^*)^{t+1}$. Dále nechť je dána neprázdná množina indexů $\emptyset \neq I \subset \{1, \ldots, t\}$ a množina $U \subseteq (\Sigma^+)^t$ splňující následující tři podmínky:*

1. ***Podmínka neprovedené lustrace**. $\forall x = \langle x_1, \ldots, x_t \rangle \in U$ slovo $a_0 x_1 a_1 \cdots x_t a_t$ patří do jazyka L a existuje pro něj přijímající výpočet automatu M takový, že se při něm neprovádí lustrování podslov x_i a x_j pro žádné $i \in I$ a $j \notin I$.*

2. ***Podmínka reprezentantů**. $\forall x, y \in U$, $(\forall i \in I, x_i = y_i) \iff (\forall i \notin I, x_i = y_i)$.*

3. ***Podmínka diagonalizace**. $\forall x, y \neq x \in U$, slovo $a_0 z_1 a_1 \cdots z_t a_t$, kde $z_i = x_i$ pro $i \in I$ a $z_j = y_j$ pro $j \notin I$, nepatří do jazyka L.*

Pak $|U| < \widetilde{p}\left(\sum_{i=0}^{t} |a_i|\right)$, kde $\widetilde{p}$ je následující polynom v l:

$$\widetilde{p}\,(l) = \left(s(t+h+1)^{k+h+2}\right)^{2tk+1} (l+1)^{(k+h)(2tk+1)} + 1,$$

přičemž s je počet stavů automatu M.

Příklad 3.1.1 *Jazyk $L_1 = \{x\#y |\ x, y \in \{0,1\}^+ \ \&\ (x \in \{0\}^* \iff y \notin \{0\}^*)\}$ je regulární, tj. rozpoznávatelný jednohlavým konečným automatem, a tedy rozpoznávatelný bez lustrování* *podslov x a y. Ačkoliv je dvojic $\langle x, y\rangle$, pro které se nemusí provádět lustrování, neomezeně mnoho, toto tvrzení přesto platí, neboť pokud si zvolíme $a = \langle \epsilon, \#, \epsilon\rangle$, $I = \{1\}$ a nějaký automat M přijímající tento jazyk, pak každá množinu U splňující všechny tři podmínky tvrzení má velikost menší než tři. Tento fakt vyplývá z podmínky diagonalizace, jejímž následkem může množina U obsahovat pouze dva prvky $\langle 0^n, y\rangle$ a $\langle x, 0^m\rangle$, pro m, n přirozená čísla a $x, y \in \{0,1\}^+ - \{0\}^*$.*

Příklad 3.1.2 *Jazyk $L_2 = \{x\#x|\ x \in \{0,1\}^*\}$ je rozpoznávatelný $(2,0)$-FA automatem. Sporem ukážeme, že není rozpoznatelný žádným $(1,h)$-FA automatem. Pro spor předpokládejme, že je, a zvolme si jeden konkrétní $(1,h)$-FA automat M, který ho přijímá. Dále si zvolme $a = \langle \epsilon, \#, \epsilon\rangle$ a $I = \{1\}$. Jelikož jednohlavý automat M nemůže provádět žádné lustrování, množina $U = \{\langle x, x\rangle \mid x \in \{0,1\}^*\}$, splňuje všechny tři podmínky tvrzení. Avšak množina U je nekonečná, čímž dostáváme spor se závěrem tvrzení a $(1,h)$-FA automat rozpoznávající jazyk L_2 nemůže existovat.*

Příklad 3.1.3 *Jazyk $L_3 = \left\{x\#S(x) \mid x \in \{0,1\}^+\right\}$, kde $\forall x = b_1 b_2 \cdots b_{|x|} \in \Sigma^+$, $S(x) = b_2 b_4 b_6 \cdots b_{2\left\lfloor \frac{|x|}{2} \right\rfloor}$. Tento jazyk podobně jako jazyk L_2 je rozpoznávatelný dvouhlavým konečným automatem, avšak žádným jednohlavým. Důkaz se provede stejně jako pro jazyk L_2 s tím, že se zvolí $U = \{\langle b_1 0 b_2 0 \cdots b_m 0,\ b_1 b_2 \cdots b_m\rangle \mid \forall m \geqslant 1, \forall i \in \{1, \ldots, m\}, b_i \in \Sigma\}$*

Důkaz: Důkaz Věty O lustrování provedeme sporem. Vezměme si jazyk L a k němu (k,h)-FA M, který jej rozpoznává. Dále pak $(t+1)$-tici $a = \langle a_0, \ldots, a_t\rangle \in (\Sigma^*)^{t+1}$, neprázdnou množinu indexů $I \subset \{1, \ldots, t\}$ a množinu $U \subseteq (\Sigma^+)^t$ nesplňující závěr tvrzení, tedy $|U| \geqslant \widetilde{p}\left(\sum_{i=0}^{t} |a_i|\right)$. Označme si jako l součet délek slov a_i, tj. $l = \sum_{i=0}^{t} |a_i|$.

Označme si jako L_U množinu všech slov daných množinou U, tedy $L_U = \{a_0 x_1 a_1 \cdots x_t a_t|\ \langle x_1, \ldots, x_t\rangle \in U\}$. Pro každé slovo u z množiny L_U zafixujme jeden konkrétní přijímající výpočet V_u tohoto slova automatem M takový, že dosvědčuje splnění *Podmínky neprovedené lustrace* z předpokladů tvrzení.

Konfigurace výpočtu V_u pro slova $u = a_0 x_1 a_1 \cdots x_t a_t$ z L_U lze rozdělit na tři typy:

1. konfigurace, ve které se některá z hlav automatu M nalézá nad podslovem x_i pro $i \in I$.

2. konfigurace, ve které se některá z hlav automatu M nalézá nad podslovem x_i pro $i \notin I$.

3. konfigurace, ve které žádná z hlav automatu M není nad podslovem x_i pro $i \in \{1, \ldots, t\}$.

Vzhledem k podmínce neprovedené lustrace není možné, aby se ve výpočtu V_u vyskytla konfigurace, ve které by nějaká hlava automatu byla nad podslovem x_i, kde $i \in I$, a zároveň by další hlava byla nad podslovem x_j, kde $j \notin I$. Tedy typ konfigurací výpočtu V_u je určen jednoznačně.

Výpočet V_u (pro $u \in L_U$) rozdělme na *bloky* - maximální souvislé podúseky výpočtu V_u takové, že konfigurace v rámci jednoho bloku jsou stejného typu.

Všimněme si rozmístění hlav v první konfiguraci každého bloku. V každé takové konfiguraci jsou všechny hlavy automatu, až případně na jednu, nad podslovy a_i vstupního slova.[3] Tento fakt vyplývá z toho, že pokud došlo ke změně typu konfigurace, pak buď nějaká hlava (tedy právě jedna) vstoupila do některého podslova x_i (první a druhý typ konfigurace), nebo naopak poslední zbylá hlava z některého podslova x_i vystoupila (třetí typ konfigurace). V případě první možnosti je jedna z hlav nad prvním znakem některého podslova x_i a všechny ostatní hlavy jsou nad podslovy a_i. V případě druhé možnosti jsou zřejmě všechny hlavy nad podslovy a_i, případně nad symbolem \$.

Z uvedeného pravidla existuje jedna výjimka. Tou je první konfigurace výpočtu, pokud $a_0 = \epsilon$. Pak v první konfiguraci prvního bloku výpočtu V_u pro každé slovo z L_U jsou všechny hlavy na prvním symbolu podslova x_1. (Víme, že podslovo x_1 je ve všech $u \in L_U$ různé od prázdného slova.)

Pro počet bloků m výpočtu V_u platí $m \leqslant 2tk + 1$, neboť každá hlava může způsobit změnu typu konfigurace maximálně $2t$-krát a hlav je k.

Pro konfiguraci $c = \langle w\$, p, i_1, \ldots, i_k, H\rangle$ automatu M nad slovem $w = a_0 x_1 a_1 \cdots x_t a_t$ definujme jako *charakter konfigurace*[4] $(k+4)$-ci $\widehat{c} = \left\langle p, i'_1, \ldots, i'_k, H', h_I, h_{\overline{I}} \right\rangle$, kde

$$i_j = \begin{cases} o + \sum_{0 \leqslant m < r} |a_m| & \text{pokud se } j\text{-tá hlava nachází na } o\text{-té pozici podslova } a_r. \\ 1 + \sum_{m=0}^{t} |a_m| & \text{pokud se } j\text{-tá hlava nachází nad koncovým symbolem \$.} \\ -r & \text{pokud se } j\text{-tá hlava nachází nad podslovem } x_r. \end{cases}$$

[3] Případně některé hlavy mohou být nad koncovým symbolem \$, který můžeme pokládat za součást a_t.

[4] Charakter konfigurace je obdobný pojmu relativního rozložení hlav a kamenů (Definice 2.1.12). Liší se však tím, že rozlišuje polohu hlav, které jsou mimo podslova a_i, a také kameny umístěné mimo tato podslova.

$$
\begin{aligned}
H' &= \Big\{ j' = o + \sum\nolimits_{0\leqslant m<r} |a_m| \mid \exists\, j \in H,\ j\text{-tá pozice ve slově } w \text{ přísluší } o\text{-tému symbolu} \\
&\qquad \text{podslova } a_r \Big\}\,. \\
h_I &= \left|\{ j \in H| \ \ j\text{-tá pozice ve slově } w \text{ přísluší některému ze slov } x_i \text{ pro } i \in I\}\right|. \\
h_{\overline{I}} &= \left|\{ j \in H| \ \ j\text{-tá pozice ve slově } w \text{ přísluší některému ze slov } x_i \text{ pro } i \notin I\}\right|.
\end{aligned}
$$

Zřejmě charakter první konfigurace každého bloku jednoznačně určuje rozmístění hlav a kamenů v podslovech a_i v dané konfiguraci.

Vzorem výpočtu $\widehat{V_u}$ pak budeme rozumět posloupnost charakterů prvních konfigurací ze všech bloků výpočtu V_u v pořadí bloků tohoto výpočtu:

$$
V_u = \underbrace{c_{j_1}\cdots c_{j_2-1}}_{\text{blok 1}}\underbrace{c_{j_2}\cdots c_{j_3-1}}_{\text{blok 2}}\cdots\underbrace{c_{j_m}\cdots c_{j_{m+1}-1}}_{\text{blok m}}
$$

$$
\widehat{V_u} = \widehat{c_{j_1}}\widehat{c_{j_2}}\cdots\widehat{c_{j_m}}.
$$

Pro počet d_l možných charakterů konfigurací platí:

$$
\begin{aligned}
d_l &\leqslant \underbrace{s}_{\#\text{ stavů}} \cdot \underbrace{\left(\sum_{i=0}^{i=t}|a_i| + t + 1\right)^k}_{\#\text{ možností rozmístění hlav}} \cdot \underbrace{\left(\sum_{i=0}^{i=t}|a_i| + 1\right)^h (h+1)^2}_{\#\text{ možností rozmnístění kamenů}} \\
&\leqslant s\,(l+t+1)^{k+h}\,(h+1)^2\,.
\end{aligned}
\tag{3.1}
$$

Z počtu bloků výpočtu V_u a počtu možných charakterů konfigurací získáme odhad pro počet o_l různých vzorů výpočtu:

$$
\begin{aligned}
o_l &\leqslant \left(s\,(l+t+1)^{k+h}\,(h+1)^2\right)^{2tk+1} \\
&\leqslant \left(s(t+1)^{(k+h)}\,(h+1)^2\right)^{2tk+1}\,(l+1)^{(k+h)(2tk+1)}
\end{aligned}
\tag{3.2}
$$

kde s je počet stavů automatu M.

Jelikož $|U| \geqslant \widetilde{p}\,(l) > o_l$, existují dvě různé t-tice x_1 a x_2 v množině U takové, že slova u_1 a u_2, která k nim přísluší, mají přijímající výpočty V_{u_1} a V_{u_2} stejného vzoru.

Vezměme si nyní slovo $w_0 = a_0 z_1 a_1 \cdots z_t a_t$, kde $z_i = (x_1)_i$ pro $i \in I$ a $z_i = (x_2)_i$ pro $i \notin I$.

Toto slovo vzhledem k podmínce diagonalizace nepatří do jazyka L. Ukážeme však, že toto slovo automat M přijímá, čímž získáme požadovaný spor.

Automat M přijímá slovo w_0 výpočtem, který lze opět rozdělit na bloky a který má stejný vzor jako výpočty V_{u_1} a V_{u_2}. Automat se při přijímání slova w_0 bude chovat následujícím způsobem. Pokud bude některá z hlav automatu M v některém z podslov z_i pro $i \notin I$, pak se automat bude chovat stejně jako v odpovídajícím bloku přijímajícího výpočtu V_{u_2}. V ostatních případech se bude chovat jako při přijímajícím výpočtu V_{u_1}. Ukážeme, že takovýto výpočet bude korektní.

Nechť $e_1 < e_2 < \cdots < e_{|I|}$ jsou čísla z množiny I a $f_1 < f_2 < \cdots < f_{t-|I|}$ jsou čísla z množiny $\{1, \ldots, t\} \setminus I$. Indukcí lze snadno nahlédnout, že v první konfiguraci každého bloku takovéhoto výpočtu je ve slově w_0 relativní rozložení hlav a kamenů, včetně stavu, vzhledem k podslovům $a_0, \ldots, a_t, z_{e_1}, \ldots, z_{e_{|I|}}$, stejné jako relativní rozložení hlav a kamenů, včetně stavu, vzhledem k podslovům $a_j, (x_1)_{e_1}, \ldots, (x_1)_{e_{|I|}}$ ve slově u_1 v první konfiguraci příslušného bloku výpočtu V_{u_1}. Stejné tvrzení platí pro rozložení hlav a kamenů, včetně stavu, relativně vzhledem k $a_0, \ldots, a_t, z_{f_1}, \ldots, z_{f_{t-|I|}}$ ve slově w_0 a ve slově u_2.

V každém bloku výpočtu nad slovem w_0 se tedy hlavy pohybují v podslovech, která jsou identická s podslovy slova u_1, resp. u_2, a ve kterých je na počátku bloku výpočtu stejné relativní rozložení hlav i kamenů jako je v těchto podslovech při výpočtu V_{u_1}, resp. V_{u_2}. Výpočet proto může probíhat stejně jako nad slovem u_1, resp. u_2, dokud některá z hlav nevstoupí do cizího podslova, tj. nezpůsobí změnu konfigurace. Hlava, která tuto změnu způsobí, je ve všech třech výpočtech ta samá a vstoupí do toho samého podslova.

Zbývá tedy nahlédnout, že relativní rozložení hlav a kamenů, včetně stavu, v první konfiguraci každého bloku splňuje uvedené požadavky. V první konfiguraci prvního bloku toto jistě platí, neboť je to počáteční konfigurace a tudíž všechny hlavy jsou nad prvním symbolem toho samého podslova a žádný kámen není položen. Dále ukážeme, že pokud tvrzení platí pro první konfiguraci i-tého bloku, pak platí i pro první konfiguraci $(i+1)$-ho bloku.

Pokud se jedná o blok s konfiguracemi třetího typu (hlavy jsou pouze v podslovech a_j), tak je tvrzení zřejmé, neboť kameny v podslovech z_i, resp. $(x_1)_i$ a $(x_2)_i$, se při žádném z výpočtů V_{u_1} a V_{u_2} v tomto bloku nepohnou a na konci bloku je při těchto výpočtech relativní rozložení kamenů a hlav vzhledem k a_j stejné, tedy tvrzení je splněné.

Pokud se jedná o blok s konfiguracemi prvního typu (hlavy jsou v podslovech a_j a z_i, $i \in I$), pak výpočet probíhá stejně jako ve výpočtu V_{u_1}. Na konci tohoto bloku tedy jistě bude relativní rozložení hlav a kamenů, včetně stavu, ve slově w_0 vzhledem k $a_0, \ldots, a_t, z_{e_1}, \ldots, z_{e_{|I|}}$ odpovídat rozložení ve slově u_1 při výpočtu V_{u_1}. S kameny v podslovech z_i, $i \notin I$, se v průběhu tohoto bloku výpočtu nehýbe a kameny v podslovech a_j jsou rozmístěny stejně jako při výpočtu nad slovem u_2 (shodnost charakterů konfigurací). Z tohoto faktu a z indukčního předpokladu vyplývá, že relativní rozložení kamenů i hlav, včetně stavu, vzhledem k $a_0, \ldots, a_t, z_{f_1}, \ldots, z_{f_{t-|I|}}$ je stejné jako při výpočtu nad slovem u_2, což jsme chtěli ukázat.

Obdobně nahlédneme platnost tohoto tvrzení pro blok s konfiguracemi druhého typu.

Zbývá si ještě povšimnout, že v průběhu výpočtu nemůže dojít k problémům způsobeným nedostatkem kamenů. Počet nepřístupných kamenů je totiž v daném bloku všech výpočtů vždy stejný, neboť jejich počet je dán tím, kolik jich je uloženo v nepřístupných podslovech. To je vždy stejné vzhledem ke shodnosti charakteru konfigurace.

Tím je tvrzení dokázáno. □

Speciálním případem Věty O lustrování je následující důsledek.

Důsledek 3.1.2 *Nechť je dán jazyk $L \subseteq \Sigma^*$ a (k,h)-FA M přijímající tento jazyk. Nechť jsou dána slova $a_0, a_1, a_2 \in \Sigma^*$. Dále nechť je dána prostá funkce $f : \Sigma^* \longrightarrow \Sigma^*$ taková, že $\forall x, y \in \Sigma^*$, $a_0xa_1ya_2 \in L \Longrightarrow y = f(x)$. A nechť je dána množina slov $U = \{x$, existuje přijímající výpočet automatu M pro slovo $a_0xa_1f(x)a_2$ takový, že se při něm neprovádí lustrování podslova x s podslovem $f(x)\}$. Pak $|U| < \widetilde{p}_{k,h,s}(|a_0a_1a_2|)$, kde $\widetilde{p}_{k,h,s}$ je následující polynom:*

$$\widetilde{p}_{k,h,s}(l) = \left(s(h+3)^{k+h+2}\right)^{4k+1} (l+1)^{(k+h)(4k+1)} + 1,$$

přičemž s je počet stavů automatu M.

Důkaz: Tvrzení plyne z Věty O lustrování. Položí-li se $I = \{1\}$ a $t = 2$. Pak pokud f je prostá funkce splňující předpoklady dokazovaného tvrzení, splňuje množina U požadavky Věty O lustrování a její velikost je tedy podle této věty omezena uvedeným polynomem. □

Prostudováním důkazu Věty O lustrování zjistíme, že odhad velikosti množiny U,

$$|U| < \left(s(t+h+1)^{k+h+2}\right)^{2tk+1} (l+1)^{(k+h)(2tk+1)} + 1,$$

lze ještě mírně zlepšit.

Především rozmístění hlav a kamenů je v počáteční konfiguraci každého výpočtu pro všechna slova stejná. Tedy nerovnost 3.2 pro počet vzorů výpočtů lze zlepšit na:

$$o_l \leqslant (\text{počet různých charakterů konfigurací})^{2tk}.$$

Dále pak si všimněme, že v první konfiguraci každého bloku musí být alespoň jedna z hlav nad prvním znakem některého ze slov a_i nebo x_i. Nerovnost 3.1 lze tedy pro počet charakterů prvních konfigurací zlepšit na:

$$\begin{aligned} d_l &\leqslant \underbrace{s}_{\#\text{ stavů}} \cdot \underbrace{\left(\sum_{i=0}^{i=t} |a_i| + t + 1\right)^{k-1}}_{\#\text{ možností rozmístění } k-1 \text{ hlav}} \cdot \underbrace{2tk}_{\text{pozice a výběr zbylé hlavy}} \\ &\quad \underbrace{\left(\sum_{i=0}^{i=t} |a_i| + 1\right)^{h} (h+1)^2}_{\#\text{ možností rozmnístění kamenů}} \\ &\leqslant 2stk\,(l+t+1)^{k+h-1}\,(h+1)^2. \end{aligned}$$

V důsledku uvedených nerovností musí pro velikost U platit následující silnější odhad:

$$|U| < \left(2stk(t+h+1)^{k+h+1}\right)^{2tk} (l+1)^{(k+h-1)2tk} + 1.$$

Odhad na velikost množiny U z Věty O lustrování však již nelze příliš zlepšit principiálně. Pro jistý jazyk ukážeme příklady množin U_l, kde l je součet délek slov a_i, splňující předpoklady Věty O lustrování takové, že

$$|U_l| > c_1\,(l+1)^{c_2(k+h)(2tk+1)},$$

kde c_1 je konstanta závislá na volbě k, h a t a $c_2 > 0$ je reálná konstanta nezávislá na volbě k, h, t a l .

Vezměme si $t \geqslant 4$ sudé číslo a definujme jazyk $L_t = \left\{(a_i\#x\#)_{i=1}^{t}\, a_{t+1} \mid\ x \in \{0,1\}^* \ \&\ \forall i \in \{1,\dots,t+1\}, a_i \in \{0\}^*\right\}$. Tento jazyk lze zřejmě deterministicky rozpoznávat $(2,0)$-FA automatem tak, že se postupně ověří shodnost všech kopií slova x. (Kopií slova x rozumíme každé druhé podslovo vstupního slovo nad abecedou $\{0,1\}$ oddělené znakem $\#$. Tyto kopie budeme též číslovat $1,\dots,t$ a značit jako $x_1,\dots,x_t$.) Zřejmě pro ověření shodnosti všech kopií slova x stačí ověřit, že $x_i = x_{i+1}$ pro $i = 1,\dots,t-1$.

Vezměme si $k \geqslant 9$ a $h \geqslant 2$ přirozená čísla taková, že $k = 3i+6$ pro i přirozené číslo, a vezměme si $t \geqslant 4$ sudé číslo. Sestrojíme (k,h)-FA automat M rozpoznávající jazyk L_t takový, že pro některá slova z L_t se při některých přijímajících výpočtech nebude provádět lustrování žádné liché kopie x se žádnou sudou kopií x. (Liché kopie x jsou $x_1, x_3, \dots, x_{t-1}$ a sudé kopie x jsou $x_2, x_4, \dots, x_t$.)

Automat M bude pracovat následovně. V prvním kroce se nedeterministicky rozhodne, zda použije způsob výpočtu jako výše uvedený $(2,0)$-FA automat rozpoznávající L_t, nebo zda se vstupní slovo pokusí přijmout bez lustrování sudých kopií x s lichými.

Přijímající výpočet bez lustrování bude probíhat pouze pro slova $(a_i\#x\#)_{i=1}^{t}\, a_{t+1}$ taková, že pro $i = 2,\dots,t+1$ je $|a_1| = \frac{k-9}{3}\,(t-1)\,|a_i|$, $|x| = \frac{k-6}{3}\,(t-1)\,|a_2|$ a počet znaků 1 ve slově x je relativně malý. (Přesný počet znaků 1 ve slově x a jejich rozložení určíme později.) Pokud vstupní slovo nebude takovéhoto typu, automat jej odmítne.

Výpočet tedy proběhne tak, že automat nejprve prověří splnění uvedených podmínek na délku podslov, poté porovnáním prověří, že všechny sudé kopie x jsou identické a všechny liché kopie x jsou identické a nakonec bez přímého porovnání a bez lustrování ověří, že sudé kopie x jsou shodné s lichými.

Ověření splnění podmínek na délku podslov se provede takto. Hlava k se umístí na začátek slova a_1 (ve skutečnosti už tam je) a hlava $k-1$ se umístí na začátek slova a_2. Poté se simultáním pohybem těchto hlav, při které se hlava $k-1$ pohybuje $\frac{k-9}{3}\,(t-1)$ rychleji, ověří vztah pro délku slov a_1 a a_2. Pak se prověří shodnost délek slov $a_2,\dots,a_{t+1}$ a to tak, že se postupně porovná délka slov a_i a a_{i+1} pro $i = 2,\dots,t$. Hlava k se vždy přesune na začátek slova a_i a hlava $k-1$ na začátek slova a_{i+1} a simultáním pohybem těchto hlav se ověří rovnost jejich délky.

Nyní se obdobným způsobem pomocí hlav $k-2$ a $k-3$ ověří, že délka první kopie x je $\frac{k-6}{3}\,(t-1)\,|a_2|$ a že délka druhé kopie x je $\frac{k-6}{3}\,(t-1)\,|a_3|$.

Následně se pomocí hlav $k-4$ a $k-5$ prověří shodnost všech lichých kopií x a shodnost všech sudých kopií x. To se provede postupným porovnáním kopií x_i a x_{i+2} pro $i = 1,\dots,t-2$.

Pokud kterýkoliv z těchto testů selže, je vstupní slovo odmítuto. (Pokud slovo patří do L_t, pak je jistě přijato druhým, standardním způsobem výpočtu.)

Automatu tedy zbývá ověřit, že se sudé kopie x shodují s lichými. Tuto shodu bude automat prověřovat po úsecích slova x délky d, kde $d = |a_2|$. Rozmístění znaků 1 v úseku slova x se "překopíruje" do rozmístěné kamenů v některém ze slov a_i a poté se tato kopie vytvořená z kamenů porovná s odpovídajícím úsekem v jiné kopii x. Kromě kamenů se pro vyznačení polohy znaků 1 použijí i zatím nevyužité hlavy ve slově a_1. Jelikož všechny sudé kopie x jsou stejné a všechny liché kopie x jsou stejné, je nepodstatné, ve kterých kopiích porovnáváme které úseky.

Porovnání úseků bude organizováno v cyklech. V každém cyklu budou tři hlavy $3i-2$, $3i-1$ a $3i$ procházet kopie slova x a porovnávat v nich $(t-1)$ po sobě jdoucích úseků délky d.

V i-tém cyklu se tak porovnají úseky $(t-1)(i-1), \ldots, (t-1)i-1$. Úsek $(t-1)(i-1)$ z kopie x_1 se porovná s tímto úsekem z x_2, úsek $(t-1)(i-1)+1$ z kopie x_2 se porovná s tímto úsekem z x_3 a tak dále. Tedy $((t-1)(i-1)+j-1)$-tý úsek z kopie x_j se porovná s tímto úsekem v kopii x_{j+1}.

Porovnání jednoho úseku, řekněme úseku $((t-1)(i-1)+j-1)$, proběhne tak, že se hlava $3i-2$ nastaví na začátek tohoto úseku v kopii x_j a hlava $3i-1$ se umístí na začátek podslova a_{j+1}. Poté se obě hlavy začnou simultáně pohybovat a vždy, když hlava $3i-2$ narazí v kopii slova x na znak 1, tak hlava $3i-1$ položí ve slově a_{j+1} kámen. Zároveň s těmito hlavami se pohybují hlavy $3i+2, \ldots, k-6$. (Před začátkem porovnávání jednoho úseku se vždy hlavy $3i+1, \ldots, k-6$ umístí na stejnou výchozí pozici ve slově a_1, na pozici hlavy $3i+1$.) Jakmile dojdou automatu kameny,[5] začnou se pohybující se hlavy ve slově a_1 postupně zastavovat a svojí polohou tak vyznačovat polohy dalších znaků 1 ve zkoumaném úseku slova x. Jakmile hlava $3i-1$ ve slově a_{j+1} dorazí na znak #, bude překopírován obsah úseku $((t-1)(i-1)+j-1)$ slova x do polohy kamenů v podslově a_{j+1} a hlav v podslově a_1.

Pak se provede nastavení hlavy $3i-2$ na začátek úseku $((t-1)(i-1)+j-1)$ v kopii x_{j+1}. Hlavy $3i-2$ a $3i$ se nastaví na začátek slova a_{j+1} a hlava $3i-1$ se nastaví na začátek kopie x_{j+1}. Poté se simultáně pohybuje hlavami $3i-2$ a $3i-1$, přičemž hlava $3i-1$ v kopii x_{j+1} se pohybuje $((t-1)(i-1)+j-1)$-krát rychleji než hlava $3i-2$ v podslově a_{j+1}. Když hlava $3i-2$ dorazí

[5]Automat sleduje počet položených kamenů.

na konec slova a_{j+1}, hlava $3i-1$ se zastaví na místě, neboť je na začátku požadovaného úseku, a hlava $3i-2$ dorazí na její pozici.

Nyní proběhne porovnání překopírovaného úseku s tímto úsekem v kopii slova x_{j+1}. Hlava $3i-2$ (v kopii x_{j+1}), hlava $3i$ (v podslově a_{j+1}) a hlava $3i+1$ (ve slově a_1) se začnou simultáně pohybovat. Vždy, když hlava $3i$ narazí na kámen nebo hlava $3i+1$ na jinou hlavu, musí být pod hlavou $3i-2$ ve slově x_{j+1} znak 1, jinak tam musí být znak 0. Pokud se narazí na rozdíl, je slovo odmítnuto. Hlava $3i$ v podslově a_{j+1} kameny sbírá.

Jakmile hlava $3i$ dorazí na konec slova a_{j+1}, je porovnání daného úseku hotovo. Hlava $3i-2$ se nachází na začátku dalšího úseku kopie x_{j+1}, který se má porovnávat. Hlavy $3i+2,\ldots,k-6$ se přesunou na pozici hlavy $3i+1$ a hlava $3i-1$ se nastaví na začátek podslova a_{j+2}.

Pokud je $j \leqslant t-1$ může začít porovnávání dalšího úseku $((t-1)(i-1)+j)$, jinak bylo provedeno všech $(t-1)$ porovnání hlavami $3i-2$, $3i-1$ a $3i$.

Pokud je provedeno všech $(t-1)$ porovnání, přesunou se hlavy $3i-2$ a $3i$ na začátek podslova a_{t+1}. Hlava $3i+1$ se nastaví na začátek kopie x_1 a simultáním pohybem s hlavou $3i-2$, při kterém se hlava $3i+1$ pohybuje $((t-1)\,i)$-krát rychleji, se hlava $3i-2$ umístí na začátek úseku, který se má následně porovnávat. Tím začne další cyklus pro další hlavy.[6] (Na začátku prvního cyklu se hlava 1 umístí na první znak slova x_1, což je první znak úseku, který se má porovnávat.)

Během porovnávání jednoho úseku se hlavy ve slově a_1 posunou o d pozic. Jelikož porovnávaných úseků je $\frac{k-6}{3}\,(t-1)$ a při posledním cyklu již v a_1 žádné hlavy nejsou, je délka slova a_1 dostatečně velká.

Celkem se postupně porovná $\frac{k-6}{3}\,(t-1)$ úseků slova x délky d, tedy celé x. Pokud během porovnávání není zjištěn rozdíl mezi lichými a sudými kopiemi x, je slovo přijato, jinak je odmítnuto. Zřejmě jsou tedy tímto způsobem přijímána pouze slova z jazyka L_t a automat M tak přijímá právě tento jazyk.

Aby byl dostatek kamenů a hlav pro zaznamenání poloh znaků 1 ve slově x, musí pro každé $i \in \left\{1,\ldots,\frac{k-6}{3}\right\}$ úseky $(t-1)(i-1),\ldots,(t-1)\,i-1$ obsahovat nejvýše $k+h-6-3i-1$

[6] Hlavy $3i-2$, $3i-1$ a $3i$ jsou v dalším výpočtu již nevyužité. Pokud by se změnila definice jazyka L_t tak, že by se za slovo a_{t+1} přidalo ještě jedno slovo a_{t+2}, pak by v něm mohly uvedené hlavy pracovat stejně, jako pracují hlavy ve slově a_1, tedy mohly by zde sloužit pro zaznamenávání znaků 1. Tím by se nepatrně zlepšily níže uvedené odhady.

znaků 1.

Pro dané d lze tedy tímto způsobem přijímat všechna slova, kde délky podslov a_i a délka x splňují uvedené podmínky a kde jednotlivé úseky x obsahují nejvýše uvedený počet znaků 1.

Různých slov x, která jsou patřičné délky a mají právě uvedený maximální možný počet znaků 1, je alespoň:

$$\prod_{i=1}^{\frac{k-6}{3}} \binom{d}{k+h-7-3i}^{(t-1)}.$$

Pokud $d \geqslant k+h$, pak pro $i = 1, \ldots, \frac{k-6}{3}$ platí:

$$\binom{d}{k+h-7-3i} > \frac{d^{k+h-7-3i}}{(k+h)^{k+h}}.$$

Dále platí:

$$\begin{aligned}
\sum_{i=1}^{\frac{k-6}{3}} k+h-7-3i &= \frac{k-6}{3}(k+h-7) - 3\sum_{i=1}^{\frac{k-6}{3}} i \\
&= \frac{k-6}{3}(k+h-7) - \frac{3}{2}\frac{k-6}{3}\frac{k-3}{3} \\
&= \frac{k-6}{3}\left(k+h-7-\frac{k}{2}+\frac{3}{2}\right) \\
&= \frac{1}{6}(k-6)(k+2h-11)
\end{aligned}$$

a tedy:

$$\prod_{i=1}^{\frac{k-6}{3}} \binom{d}{k+h-7-3i}^{(t-1)} > \frac{d^{(t-1)(k-6)(k+2h-11)/6}}{(k+h)^{\frac{k-6}{3}(k+h)(t-1)}}$$

Pokud si tedy vezmeme libovolné $d \geqslant k+h$, položíme $a_1 = 0^{d\frac{k-9}{3}(t-1)}$, $a_i = 0^d$ pro $i > 1$, $l = d\left(\frac{k-9}{3}(t-1)+t\right)$ a $I = \{2, 4, \ldots, t\}$, pak množina $U_l = \Big\{\langle x, \ldots, x\rangle \in (\{0,1\}^*)^t \mid |x| = d\frac{k-6}{3}(t-1)$ a pro výskyt znaku 1 ve slově x platí omezení uvedená výše$\}$ bude splňovat všechny tři podmínky Věty O lustrování a bude pro ni platit:

$$|U_l| > c_1 l^{(t-1)(k-6)(k+2h-11)/6} \geqslant c_1 (l+1)^{c_2(k+h)(2tk+1)},$$

kde c_1 je konstanta závislá na automatu M, resp. na počtu jeho hlav a kamenů, a $c_2 > 0$ je reálná konstanta.

Tím jsme ukázali, že uvedený odhad ve Větě O lustrování nelze principiálně příliš zlepšit.

S tím souvisí i následující věta, kterou zde uvedeme bez důkazu pouze pro ilustraci. Její správnost vyplývá z pozorování, že Turingův stroj je ekvivalentní zařízení s konečným počtem vnitřních stavů a se 3 neomezenými počitadly (Demuth, Kryl, Kučera [1]). Pomocí tří počitadel se dá simulovat obsah pásky Turingova stroje během výpočtu, přičemž během simulace pro úsek pásky délky l nad abecedou A dosahuje každé z počitadel nejvýše hodnoty $2^{\lceil \log_2 |A| \rceil l}$. Potřebná počitadla se dají nahradit vzdáleností mezi hlavami automatu a pokud tedy má konečný automat dostatek prostoru, může simulovat činnost Turingova stroje.

Věta 3.1.3 *Nechť $L \subseteq \{0,1\}^*$ je jazyk rozpoznávaný jednopáskovým Turingovým strojem. Nechť $t(x)$ je čas a $s(x)$ je prostor potřebný pro rozhodnutí o slovu x. Pak existuje k a h takové, že jazyk $L' = \left\{x\#0^e \mid x \in L \;\&\; e \geqslant t(x)\, 2^{s(x)}\right\}$ lze rozpoznávat (k,h)-FA automatem.*

Uveďme si ještě příklad jednoho zajímavého jazyka.

Příklad 3.1.4 *Nechť a, b, c jsou přirozená čísla taková, že $1 \leqslant a \leqslant b \leqslant c$. Definujme jazyk:*

$$\begin{aligned} K_{a,b,c} \quad = \quad & \left\{ (w_i\#)_{i=1}^{c} \left(w'_{c-i+1}\#\right)_{i=1}^{c} \mid \forall i \in \{1,\dots,c\},\ w_i, w'_i \in \{0,1\}^* \ \& \right. \\ & \left. \forall i \in \{a,\dots,b\},\ w_i = w'_i \right\}. \end{aligned}$$

Ukážeme, že takový jazyk lze rozpoznávat (k,h)-FA automatem právě tehdy, když $\binom{k}{2} > b - a$.

Lemma 3.1.4 *Nechť $a \leqslant b \leqslant c$ jsou přirozená čísla. Nechť $k \geqslant 1$ a h jsou přirozená čísla. Pak existuje (k,h)-FA automat rozpoznávající jazyk $K_{a,b,c}$ právě tehdy, když $b - a < \binom{k}{2}$.*

Důkaz: *1)* Nejprve ukážeme implikaci doleva. Nechť a, b, c, k, h jsou přirozená čísla splňující předpoklady tvrzení. Nechť $b - a < \binom{k}{2}$. Ukážeme, že existuje (k,h)-FA automat rozpoznávající jazyk $K_{a,b,c}$.

Tento automat bude pracovat následujícím způsobem. Pro zjištění, zda slovo patří do jazyka $K_{a,b,c}$ postačuje ověřit, že dané slovo obsahuje patřičný počet znaků # a že $w_i = w'_i$ pro všechna $a \leqslant i \leqslant b$. Začátky slov w_i a w'_i se poznají právě podle počtu znaků #, které je předcházejí.

Porovnání dvojic slov w_i a w'_i pro všechna $a \leqslant i \leqslant b$ se bude provádět v následujících cyklech. V prvním cyklu se jednotlivé hlavy $2, \dots, k$ umístí na začátky slov w_a až w_{a+k-2} a hlava 1 se umístí na začátek slova w'_{a+k-2}. (Při té příležitosti se vlastně ověří patřičný počet slov w_i a w'_i pro $i > b$.) Nyní automat pohybuje hlavou 1 doprava a zároveň pomocí hlav $2, \dots, k$ postupně porovnává slova w'_{a+k-2} až w'_a se slovy w_{a+k-2} až w_a. Pokud narazí na rozdíl nebo dojdou podslova w'_i, automat vstupní slovo odmítne. Pokud $b - a < k - 2$, pak se provede pouze patřičný počet porovnání slov w_i a w'_i.

V dalším cyklu se hlavy $3, \dots, k$ rozestaví nad slova w_{a+k-1} až w_{a+2k-4} a hlava 3 se umístí nad slovo w'_{a+2k-4}, načež se provede porovnání slov w_{a+k-1} až w_{a+2k-4} se slovy w'_{a+k-1} až w'_{a+2k-4} . Tento postup se neustále opakuje.

V m-tém cyklu se tedy pomocí m-té hlavy procházejí slova $w'_{a-1+\sum_{1 \leqslant j \leqslant m} k-j}$ až $w'_{a+\sum_{1 \leqslant j < m} k-j}$ a pomocí hlav $m+1, \dots, k$ se tato slova porovnávají se svými protějšky.

V m-tém cyklu se tak porovná $k - m$ dvojic slov w_i a w'_i a postup se opakuje tak dlouho, dokud nejsou porovnána všechna slova w_i a w'_i pro $i = a, \dots, b$.

Dohromady je možné tímto způsobem porovnat nejvýše $(k-1) + (k-2) + \cdots + 1$ dvojic slov, což je právě $\binom{k}{2}$. Pokud je tedy $b - a < \binom{k}{2}$, pak je porovnání všech dvojic slov w_i a w'_i pro $i = a, \dots, b$ možné.

Po provedení všech porovnání ještě automat ověří, že za slovem w'_a následuje patřičný počet dalších slov w'_i, to jest, že počet znaků # je za tímto slovem právě a. Poté, pokud slovo splnilo všechny požadavky, bude automatem přijato, jinak bude odmítnuto.

Zřejmě takovýto automat bude deterministický (k, h)-FA automat bez koincidence.

2) Nyní ukážeme opačnou implikaci. Ukážeme ji sporem.

Vezměme si přirozená čísla $1 \leqslant a \leqslant b \leqslant c$ a $k \geqslant 1, h$. Předpokládejme, že $b - a \geqslant \binom{k}{2}$ a že existuje (k, h)-FA automat M přijímající jazyk $K_{a,b,c}$. Z toho odvodíme spor.

Podívejme se na rozmístění podslov w_i a w'_i pro dané i. Snadno nahlédneme, že pokud některé dvě hlavy spolu porovnávají (nebo obecně lustrují) tato podslova pro nějaké i, pak již spolu nemohou lustrovat jinou dvojici slov w_j a w'_j, kde $j \neq i$. Obě hlavy se totiž pohybují pouze doprava a tak pro hlavu ležící nad slovem w'_i jsou dostupná již pouze slova w'_j, kde $j < i$. Pro žádnou z těchto hlav však již slova w_j, kde $j < i$, dostupná nejsou.

Pokud tedy spolu nějaké dvě hlavy lustrují dvojici slov w_i a w'_i pro dané i, nemohou spolu lustrovat již jinou dvojici. Dohromady tedy mohou spolu všechny hlavy lustrovat pouze $\binom{k}{2}$ dvojic w_i s w'_i. Jelikož $b - a \geqslant \binom{k}{2}$ a dvojic slov w_i a w'_i takových, že má platit $w_i = w'_i$, je $b - a + 1$, nemůže se při přijímacím výpočtu pro žádné slovo z $K_{a,b,c}$ provést lustrování všech dvojic slov $(w_a, w'_a), (w_{a+1}, w'_{a+1}), \ldots, (w_b, w'_b)$.

Vezměme si dostatečně velké n a množinu slov $L = \Big\{\#^{a-1} (w_i\#)_a^b \#^{2(c-b)} (w_{a+b-i}\#)_a^b \#^{a-1} |$ $\forall i \in \{a, \ldots, b\}, \ w_i \in \{0,1\}^n\Big\}$. Zřejmě $L \subseteq K_{a,b,c}$ a $|L| = 2^{n(b-a+1)}$. Zafixujme si pro každé slovo u z L jeden jeho přijímající výpočet V_u automatem M. Podle výše uvedeného pozorování se neprovádí v žádném z výpočtů V_u, pro $u \in L$, lustrování všech dvojic slov w_i, kde $i \in \{a, \ldots, b\}$.

Definujme si nyní pro každé $j \in \{a, \ldots, b\}$ a pro každá slova $w_a, \ldots, w_{j-1}, w_{j+1}, \ldots, w_b \in \{0,1\}^n$ množinu

$$\begin{aligned} U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b} &= \{w_j | \ w_j \in \{0,1\}^n \ \& \ \text{pro } u = \#^{a-1} (w_i\#)_a^b \#^{2(c-b)} (w_{a+b-i}\#)_a^b \#^{a-1} \\ &\quad \text{se při výpočtu } V_u \text{ neprovádí lustrování dvojice slov } w_j\}. \end{aligned}$$

Je zřejmé, že každé slovo z L přispěje do alespoň jedné z množin $U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b}$ a že žádná dvě různá slova z L nemohou přispět do téže množiny $U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b}$ stejným slovem w_j. Z uvedeného vyplývá, že součet velikostí všech množin $U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b}$ musí být alespoň $|L| = 2^{(b-a+1)n}$.

Množiny $U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b}$ splňují předpoklady Důsledku 3.1.2 (funkcí f je identita, $a_0 = \#^{a-1} (w_i\#)_a^{j-1}$, $a_1 = \# (w_i\#)_{j+1}^b \#^{2(c-b)} (w_{a+b-i}\#)_a^{a+b-j-1}$, $a_2 = (w_{a+b-i}\#)_{a+b-j+1}^b \#^{a-1}$) a pro jejich velikost tedy platí odhad $\left|U_{j,w_a,\ldots,w_{j-1},w_{j+1},\ldots,w_b}\right| < \widetilde{p}_{k,h,s} (2 (b-a) n + 2c)$. Těchto množin je $(b - a + 1) 2^{(b-a)n}$, tedy součet velikostí všech těchto množin je menší než:

$$\widetilde{p}_{k,h,s} (2 (b-a) n + 2c) (b - a + 1) 2^{(b-a)n}.$$

Vzhledem k volbě n jako dostatečně velkého je však tento horní odhad menší než předchozí dolní odhad na tento součet, což vede ke sporu. (Dolní odhad je pro dostatečně velké n nejméně $2^{n/2}$-krát větší než horní odhad.)

Tím je nalezen požadovaný spor, a automat M tedy nemůže přijímat jazyk $K_{a,b,c}$.

Tím je lemma dokázáno. □

Z tohoto lemmatu mimo jiné vyplývá, že s počtem hlav automatu roste jeho rozpoznávací síla. Tento fakt je pak shrnut ve Větě O hierarchii (Věta 3.2.4).

3.2 Hierarchie jazyků

V tomto oddíle dokážeme pro $k \geqslant 2$ a h přirozená čísla vztahy $R_{k,h} \subsetneq R_{k+1,h}$, $R_{k,h} \subsetneq R_{k,h+1}$ a $R_{1,h} \subsetneq R_{2,h}$. Jelikož neostrá inkluze těchto tříd je zřejmá, budeme dokazovat ostrou inkluzi. Důkaz ostré inkluze provedeme tak, že sestrojíme jazyk, jenž patří do třídy $R_{k,h}$, avšak nepatří do třídy $R_{k-1,h}$ ani do třídy $R_{k,h-1}$. Jelikož nalezený jazyk patří dokonce do třídy $R^{ND}_{k,h}$, budou uvedené ostré inkluze platit i pro třídy $R^{D}_{k,h}$, $R^{N}_{k,h}$ a $R^{ND}_{k,h}$.

Dále zde ukážeme, že pro žádné h' neplatí inkluze $R_{k+1,h} \subseteq R_{k,h'}$, což znamená, že hlavu nelze nahradit žádným počtem kamenů. Ukážeme však tvrzení opačné, tedy že pro $h' \leqslant h$ je $R_{k,h} \subseteq R_{k+(h-h')\lfloor k/2 \rfloor,h'}$, neboli že kámen lze nahradit jistým počtem hlav. Důsledkem toho bude tvrzení $\bigcup_{k\geqslant 1} R_{k,0} = \bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R_{k,h}$, neboli že jazyky rozpoznávané vícehlavými konečnými automaty s kameny odpovídají jazykům rozpoznávaným vícehlavými konečnými automaty s koincidencí. Vzhledem k Větě 2.2.1 platí i $\bigcup_{k\geqslant 1} R_{k,0} = \bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R^{N}_{k,h}$. Uvedené inkluze a rovnosti budou platit i pro třídy $R^{D}_{k,h}$.

Definice 3.2.1 *Nechť $k \geqslant 2$ a h jsou přirozená čísla. Definujeme jazyk $L^{k,h} \subseteq \{0,1,\#\}^*$ takto:*

$$\begin{aligned} L^{k,h} = \ & \Big\{ (x_j\#)_1^{h+1}\, w_1\#w_0\#\, (x_jz_j\#)_1^h\, (w_i\#)_2^b\, (w_{b-i}\#)_0^b\, (z_j\#)_1^{h+1} \mid \\ & \forall 1 \leqslant j \leqslant h+1, x_j, z_j \in \{0,1\}^* \ \ \& \ \ b = \tbinom{k}{2} \ \ \& \ \ \forall 0 \leqslant i \leqslant b, w_i \in \{0,1\}^* \Big\} \end{aligned}$$

a pro přirozené číslo $n \geqslant 1$ jazyk $L_n^{k,h} \subseteq L^{k,h}$ takto:[7]

$$\begin{aligned} L_n^{k,h} = \ & \Big\{ (x_j\#)_1^{h+1}\, w_1\#w_0\#\, (x_jz_j\#)_1^h\, (w_i\#)_2^b\, (w_{b-i}\#)_0^b\, (z_j\#)_1^{h+1} \mid \ \forall 1 \leqslant j \leqslant h+1, x_j, z_j \in \{0\}^* \\ & \& \ \ \forall 1 \leqslant j \leqslant h, |x_jz_j| = 2n^{k+1} \ \& \ |x_j|, |z_j| \neq 0 \ \ \& \ \ \textstyle\sum_{i=1}^{h+1} |x_i| = \sum_{i=1}^{h+1} |z_i| = 2hn^{k+1} \\ & \& \ \ b = \tbinom{k}{2} \ \ \& \ \ \forall 0 \leqslant i \leqslant b, w_i \in \{0,1\}^* \ \& \ |w_i| = n \Big\}. \end{aligned}$$

[7] *Jazyk $L_n^{k,h}$ je konečný, a tedy regulární. Této vlastnosti se však dále nepoužívá.*

Délku slov z jazyka $L_n^{k,h}$ označme $d_{n,k,h}$:

$$d_{n,k,h} = \underbrace{6hn^{k+1}}_{\text{podslova } x_j, z_j \text{ a } x_j z_j} + \underbrace{2\left(\binom{k}{2}+1\right)n}_{\text{podslova } w_i} + \underbrace{3h + 2\binom{k}{2} + 4}_{\text{počet znaků } \#}.$$

Příklad 3.2.1 *Pro $h = 0$ speciálně dostáváme:*

$$\begin{aligned} L^{k,0} &= \Big\{ x_1\#w_1\#w_0\#\,(w_i\#)_2^b\,(w_{b-i}\#)_0^b\,z_1\#| \\ &\qquad x_1, z_1 \in \{0,1\}^* \;\&\; b = \binom{k}{2} \;\&\; \forall 0 \leqslant i \leqslant b, w_i \in \{0,1\}^* \Big\} \end{aligned}$$

a

$$L_n^{k,0} = \left\{ \#w_1\#w_0\#\,(w_i\#)_2^b\,(w_{b-i}\#)_0^b\,\#|\, b = \binom{k}{2} \quad \& \quad \forall 0 \leqslant i \leqslant b, w_i \in \{0,1\}^n \right\}.$$

Značení 3.2.2 *Každé slovo x z jazyka $L^{k,h}$ se přirozeně skládá z několika podslov, která jsou v definici jazyka $L^{k,h}$ označená jako x_j, z_j, w_i pro příslušná i a j. Každé slovo w_i se ve slově x vyskytuje dvakrát; tyto výskyty budeme označovat jako* ***levou*** *a* ***pravou kopii slova*** *w_i. Levou a pravou kopii slova w_i pak budeme též dohromady označovat jako* ***protilehlé kopie slova*** *w_i. Pro dané slovo x z jazyka $L^{k,h}$ budeme podslova $(x_j\#)_1^{h+1}$, $(x_j z_j\#)_1^h$ a $(z_j\#)_1^{h+1}$ nazývat* ***oblastmi slova*** *x. Ukončovací symbol \$ budeme považovat za součást oblasti $(z_j\#)_1^{h+1}$. Pro $h = 0$ je oblast $(x_j z_j\#)_1^h$ prázdné slovo ϵ.*

V následujícím textu ukážeme, že pokud nějaký (k,h)-FA automat přijímá jazyk $L^{p,q}$, pak pro jisté dostatečně velké n a jistou volbu slov w_i musí každý přijímací výpočet tohoto automatu pro každé slovo z $L_n^{p,q}$ sestavené z těchto w_i provádět lustrování všech dvojic kopií $w_i\#$ (Lemma 3.2.1). Vzhledem ke struktuře slov z jazyka $L_n^{p,q}$ to bude znamenat, že pokud $p = k$, pak v jistý okamžik výpočtu nad takovým slovem bude muset být každá hlava v některém z podslov $w_i\#$ (při přechodu poslední hlavy z levé kopie $w_1\#$ do levé kopie $w_0\#$).

V tuto chvíli tedy již nebude žádná hlava v první oblasti $(x_j\#)_1^{q+1}$, avšak žádná hlava ještě nebude ani v poslední oblasti $(z_j\#)_1^{q+1}$. Jelikož celková délka podslov $w_i\#$ je relativně malá, ve srovnání s počtem možných $(q+1)$-tic slov x_j bude i relativně málo možných rozmístění hlav v tento okamžik. Nutně tedy pro některé různé $(q+1)$-tice slov x_j budou hlavy rozmístěny na těch samých pozicích. Dokonce lze nalézt takové $(q+1)$-tice slov x_j, že se bude shodovat i vnitřní stav automatu.

V takovém případě, jelikož už není dostupná informace o velikostech jednotlivých slov x_j, bude muset být informace o velikostech slov z_j, resp. x_j, pro ověření správnosti oblasti $(z_j\#)_1^{q+1}$ uložena v rozmístění kamenů. Pokud však kamenů bude méně než q, pak nebudou schopny v daný okamžik pojmout vyžadovaný objem informace a pro některé dvě různé $(q+1)$-tice slov x_j, resp. jim jednoznačně odpovídajících slov z_j, bude muset automat během výpočtu projít stejným stavem a stejným rozmístěním hlav i kamenů. To ovšem bude nutně znamenat, že pokud vezmeme slovo, jež má oblast $(x_j\#)_1^{q+1}$ vytvořenou z první $(q+1)$-tice a oblast $(z_j\#)_1^{q+1}$ vytvořenou z druhé, bude toto slovo muset být taktéž přijato, ač nepatří do jazyka $L^{p,q}$. Z toho vyplyne, že musí být $h \geqslant q$ (Věta 3.2.2).

Definice 3.2.3 *Nechť $k \geqslant 2, h, n$ jsou přirozená čísla. Definujme množiny $W_n^{k,h}$ a $X_n^{k,h}$ takto:*[8]

$$W_n^{k,h} = \left\{ \left\langle w_0, \dots, w_{\binom{k}{2}} \right\rangle \mid \forall 0 \leqslant i \leqslant \binom{k}{2}, w_i \in \{0,1\}^n \right\}$$

a

$$X_n^{k,h} = \left\{ \langle x_1, \dots, x_h \rangle \mid, \ \forall 1 \leqslant i \leqslant h, x_i \in \{0\}^* \ \& \, 0 < |x_i| < 2n^{k+1} \right\}.$$

Pro tyto množiny definujme přirozeným způsobem funkci $\varpi : W_n^{k,h} \times X_n^{k,h} \longrightarrow L_n^{k,h}$:

$$\begin{aligned} \forall w &= \left\langle w_0, \dots, w_{\binom{k}{2}} \right\rangle \in W_n^{k,h}, \forall x = \langle x_1, \dots, x_h \rangle \in X_n^{k,h}, \\ &\varpi(w,x) = (x_j\#)_1^{h+1} \, w_1\#w_0 \, (x_j z_j\#)_1^h \, (w_i\#)_2^b \, (w_{b-i}\#)_0^b \, (z_j\#)_1^{h+1}, \end{aligned} \tag{3.3}$$

kde slova x_{h+1} a z_j pro $j = 1, \dots, h+1$ jsou jednoznačně určena požadavky na slova z jazyka $L_n^{k,h}$. (Snadno lze nahlédnout, že funkce ϖ je bijekcí.)

Lemma 3.2.1 *Nechť $k, p \geqslant 2$ a h, q jsou libovolná přirozená čísla. Nechť M je (k,h)-FA takový, že přijímá jazyk $L^{p,q}$. Pak existuje n_0 takové, že $\forall n \geqslant n_0, \exists W \in W_n^{p,q}, \forall X \in X_n^{p,q}$ při každém přijímajícím výpočtu automatu M pro slovo $\varpi(W,X)$ automat lustruje levé kopie $w_i\#$ s jejich odpovídajícími pravými kopiemi $w_i\#$ pro všechna přípustná i.*

[8] *Množina $W_n^{k,h}$ nezávisí na velikosti h. Index h je zde pouze pro přehlednost zápisu.*

Toto lemma tedy říká, že lze zvolit taková w_i, že automat musí kopie $w_i\#$ lustrovat ve všech slovech, která patří do jazyka $L_n^{p,q}$ a jsou z těchto w_i utvořena. Takováto w_i lze nalézt od určité meze pro libovolně velká n, speciálně jich tedy lze nalézt libovolně mnoho.

DŮKAZ: Důkaz Lemmatu 3.2.1 provedeme sporem. Vezměme si k, h, p, q a (k,h)-FA M přijímající právě jazyk $L^{p,q}$, pro která toto lemma neplatí. Pro libovolné n_0 tedy existuje $n \geqslant n_0$ takové, že $\forall W \in W_n^{p,q}, \exists X \in X_n^{p,q}$ takové, že nějaký přijímající výpočet pro slovo $\varpi(W,X)$ neprovádí lustrování protilehlých kopií $w_i\#$ pro jisté i. Zvolme si tedy *dostatečně* velké n_0 a příslušné $n \geqslant n_0$. Dále pro každé $W \in W_n^{p,q}$ vyberme jedno $X_W \in X_n^{p,q}$ a jeden přijímající výpočet V_W pro slovo $\varpi(W, X_W)$ takový, že se při něm pro jisté i neprovádí lustrování protilehlých $w_i\#$. Označme si pro každé $X \in X_n^{p,q}$ množinu $H_X = \{\varpi(W,X) \mid W \in W_n^{p,q} \ \& \ X = X_W\}$. Dále si označme množinu $\mathbb{W} = \{\varpi(W, X_W) \mid W \in W_n^{p,q}\}$.

Snadno nahlédneme, že:

$$\begin{aligned} |\mathbb{W}| &= |W_n^{p,q}| = 2^{n\left(\binom{p}{2}+1\right)} \\ |X_n^{p,q}| &\leqslant \left(2n^{p+1}\right)^q \end{aligned}$$

Z toho vyplývá, že existuje $X_0 \in X_n^{p,q}$ takové, že $|H_{X_0}| \geqslant 2^{n\left(\binom{p}{2}+1\right)} / \left(2n^{p+1}\right)^q$. Pro zbytek důkazu si zafixujme jedno takové X_0. Pro každé slovo w z H_{X_0} tedy existuje přijímající výpočet V_w takový, že existuje i, pro které se neprovádí lustrování levé a pravé kopie podslova $w_i\#$.

Vezměme si libovolné $j \in \left\{0, \ldots, \binom{p}{2}\right\}$ a libovolná slova $w_0, \ldots, w_{j-1}, w_{j+1}, \ldots, w_{\binom{p}{2}} \in \Sigma^n$. Ta určují množinu $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}} = \{w_j\# \mid w_j \in \Sigma^n \ \&$ při výpočtu V_w pro slovo $w = \varpi\left(\left\langle w_0, \ldots, w_{\binom{p}{2}}\right\rangle, X_0\right)$ se neprovádí lustrování levé a pravé kopie podslova $w_j\#\}$.

Každá taková množina $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}$ splňuje předpoklady Důsledku 3.1.2 (funkcí f je identita, a_0 je část slova $\varpi\left(\left\langle w_0, \ldots, w_{\binom{p}{2}}\right\rangle, X_0\right)$ od začátku až po levou kopii slova w_j, a_1 je část mezi levou a pravou kopií tohoto slova a a_2 je část tohoto slova od levé kopie po konec) a proto pro ni platí:

$$\left|U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}\right| < \widetilde{p}_{k,h,s}\left(d'\right),$$

kde $\widetilde{p}_{k,h,s}$ je polynom z Důsledku 3.1.2 a d' je délka slova z $L_n^{k,h}$ po vypuštění podslov w_j, což lze shora odhadnout délkou slov $d_{n,k,h}$. ($d_{n,k,h}$ je polynomiální v n.)

Množin $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}$ je pro různá j a různá slova $w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}$ celkem $\left(\binom{p}{2}+1\right)2^{n\binom{p}{2}}$. Celkem tedy nejvýše $\widetilde{p}_{k,h,s}\left(d_{n,k,h}\right)\left(\binom{p}{2}+1\right)2^{n\binom{p}{2}}$ slov z H_{X_0} mohlo přispět do těchto množin. (Každé slovo w z H_{X_0} může přispět do více množin $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}$ pro různá j, avšak žádná dvě různá slova nemohou přispět do téže množiny $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}$ stejným slovem $w_j\#$.) Velikost množiny H_{X_0} je však větší než $2^{n\left(\binom{p}{2}+1\right)}/\left(2n^{p+1}\right)^q$, což vzhledem k volbě n jako dostatečně velkého znamená, že některá slova z H_{X_0} nepřispějí do žádné z množin $U_{j,w_0,\ldots,w_{j-1},w_{j+1},\ldots,w_{\binom{p}{2}}}$. To je však ve sporu s tou vlastností slov z H_{X_0}, že pro ně existuje přijímající výpočet, který neprovádí lustrování kopií nějakého podslova $w_j\#$.

Tím je lemma dokázáno. □

Na základě předchozího lemmatu nyní dokážeme následující větu o hierarchii automatů.

Věta 3.2.2 *Nechť $k \geqslant 2, h$ jsou přirozená čísla. Pak platí:*

1. *Pokud $k' \geqslant 1, h' \in \mathbb{N}$ a existuje (k',h')-FA M přijímající právě jazyk $L^{k,h}$, pak $k' \geqslant k$.*
2. *Pokud $h' \in \mathbb{N}$ a existuje (k,h')-FA M přijímající právě jazyk $L^{k,h}$, pak $h' \geqslant h$.*

Důkaz: *1.* Z Lemmatu 3.2.1 víme, že pro každý automat M přijímající jazyk $L^{k,h}$ a pro vhodné n existuje $W_0 \in W_n^{k,h}$ takové, že se pro všechna z něj utvořená slova patřící do jazyka $L_n^{k,h}$ provádí při přijímacím výpočtu lustrování všech protilehlých podslov $w_i\#$. Ze způsobu výpočtu vícehlavých konečných automatů (hlavy automatu se pohybují pouze doprava) a rozmístění protilehlých $w_i\#$ je zřejmé, že pokud některé dvě hlavy lustrují protilehlá podslova $w_i\#$ pro nějaké konkrétní i, pak již spolu nemohou lustrovat žádná další protilehlá podslova; výjimku zde tvoří podslova $w_1\#$ a $w_0\#$, která mohou být se svými protějšky lustrována týmž párem hlav, přičemž však samozřejmě tento pár hlav již nemůže lustrovat žádná jiná protilehlá podslova $w_i\#$, pro $i \neq 0,1$. Jelikož každá dvojice hlav může lustrovat pouze jednu dvojici protilehlých podslov $w_i\#$ (úsek $w_1\#w_0\#$ považujme na chvíli za jediné podslovo) a dvojic protilehlých podslov, jež vyžadují lustrování, je $\binom{k}{2}$, potřebujeme k lustrování minimálně k hlav nezávisle na počtu kamenů. Pokud tedy automat M přijímá jazyk $L^{k,h}$, pak má alespoň k hlav.[9]

[9]Snadno se nahlédne, že pokud má automat právě k hlav a provádí se lustrování všech kopií $w_i\#$, pak lustrování protilehlých kopií $w_1\#$ i $w_0\#$ musí provést tentýž pár hlav. Tento pár hlav tedy vystupuje v lustrování k protilehlých kopií $w_i\#$, přičemž všechny ostatní hlavy budou vystupovat v $k-1$ lustrováních.

2. Důkaz provedeme sporem. Vezměme si přirozené číslo $h' < h$ a (k, h')-FA M přijímající právě jazyk $L^{k,h}$. Z Lemmatu 3.2.1 víme, že od jisté dolní meze pro libovolně velké n existuje $W \in W_n^{k,h}$ takové, že se pro všechna z něj utvořená slova patřící do jazyka $L_n^{k,h}$ provádí při přijímacím výpočtu lustrování všech protilehlých podslov $w_i\#$. Vezměme si tedy dostatečně velké n, k němu W_0 uvedených vlastností a množinu slov $L = \left\{\varpi\left(W_0, X\right) \mid X \in X_n^{k,h}\right\}$. Pro každé slovo v z množiny L zafixujme jeden konkrétní přijímající výpočet V_v.

Z definice přijímání víme, že se všechny hlavy během výpočtu dostanou na symbol \$ označující pravý konec vstupního slova a tedy během výpočtu projdou i levou kopií podslov $w_1\#$ a $w_0\#$. Rozdělme tedy pro každé slovo v výpočet $V_v = c_1, \ldots, c_l$ na dvě části $V_v^L = c_1, \ldots, c_i$ a $V_v^R = c_{i+1}, \ldots, c_l$ takové, že ve všech konfiguracích $c_1, \ldots, c_i$ se nějaká hlava automatu M nachází ve slově v vlevo od levé kopie $w_0\#$ a naopak v žádné z konfigurací $c_{i+1}, \ldots, c_l$ není žádná hlava automatu M vlevo od levé kopie $w_0\#$.

Vezměme si nyní libovolné slovo v z množiny L. Snadno nahlédneme, že v první konfiguraci části V_v^R se nějaká hlava automatu M nachází nad prvním znakem levé kopie $w_0\#$. Jelikož se provádí lustrování všech protilehlých podslov $w_i\#$, muselo již proběhnout lustrování kopií podslova $w_1\#$.

Na tomto lustrování se musela podílet hlava h_1, která levou kopii $w_1\#$ opouští jako poslední. Kdyby toto lustrování zajišťovala nad levou kopií jiná hlava h_2 než ta poslední, znamenalo by to, že některá hlava h_3 by již musela být nad pravou kopií podslova $w_1\#$. To by ovšem znamenalo, že hlava h_1 nebude ve spolupráci s hlavou h_3 lustrovat žádné $w_i\#$ pro $i > 1$ a nemohla by tudíž provést všech předepsaných $k - 1$ lustrování. (Hlava h_2 by musela podle Poznámky pod čarou 9 provést též lustrování $w_0\#$.)

Víme tedy, že v době opuštění levé kopie $w_1\#$ poslední hlavou musela již nějaká hlava h_0 automatu M dosáhnout pravé kopie podslova $w_1\#$. Podle Poznámky pod čarou 9 musí být tato hlava buď nad pravou kopií podslova $w_1\#$, nebo nad pravou kopií podslova $w_0\#$, protože musí zajistit též lustrování protilehlých podslov $w_0\#$ a do levé kopie $w_0\#$ právě vstoupila poslední hlava, která toto lustrování bude zajišťovat nad levou kopií $w_0\#$. Mimo lustrování podslov $w_1\#$ a $w_0\#$ musí však hlava h_0 vystupovat v dalších $k-2$ lustrováních. Jelikož vpravo od této hlavy již neleží žádná další podslova $w_i\#$ muselo již těchto $k-2$ lustrování proběhnout. Z toho okamžitě plyne, že zbylých $k - 2$ hlav již muselo vstoupit do podslov $w_i\#$ pro $i > 1$ a tedy jsou již vpravo od oblasti $(x_j z_j\#)_1^h$. Zároveň však těchto $k-2$ hlav ještě nemůže být v oblasti $(z_j\#)_1^{h+1}$, neboť hlava nad levou kopií $w_0\#$ musí s jejich pomocí vykonat svých zbylých $k-2$ lustrování. Z uvedeného je zřejmé, že v první konfiguraci posloupnosti V_v^R nemůže být žádná hlava automatu M nad některou z oblastí $(x_j\#)_1^{h+1}$, $(x_j z_j\#)_1^h$ a $(z_j\#)_1^{h+1}$, tedy všechny jsou nad podslovy $w_i\#$.

Definujme si nyní množinu $\mathcal{T} = \{\widetilde{c} |\ \widetilde{c}$ je typ první konfigurace posloupnosti V_v^R pro nějaké $v \in L\}$. Jelikož ve všech posloupnostech V_v^R jsou v prvních konfiguracích všechny hlavy nad oblastmi $w_i\#$, dostáváme pro mohutnost množiny $\mathcal{T}$ následující odhad:

$$
\begin{aligned}
|\mathcal{T}| &\leqslant s.\ \underbrace{\left(2\binom{k}{2}(n+1)\right)^{k-1}}_{\text{pozice hlav}}\ .\ \underbrace{(d_{n,k,h}+1)^{h'}}_{\text{pozice kamenů}} \\
&\leqslant s\left(2\binom{k}{2}(n+1)\right)^{k-1}\left(6hn^{k+1}+2\left(\binom{k}{2}+1\right)n+3h+2\binom{k}{2}+5\right)^{h'} \\
&\leqslant an^{k+h'(k+1)}+b
\end{aligned}
$$

kde s je počet stavů automatu M, $d_{n,k,h}$ je délka slov z jazyka $L_n^{k,h}$ a $a, b \in \mathbb{N}$ jsou konstanty nezávislé na volbě n.

Z definice množiny $X_n^{k,h}$ snadno nahlédneme následující dolní odhad pro její velikost:

$$\left|X_n^{k,h}\right| \geqslant \left(n^{k+1}\right)^h = n^{h(k+1)}.$$

Jelikož $h' \leqslant h-1$, platí od jistého m_0 pro všechna $m > m_0$ nerovnost $m^{h(k+1)} > am^{k+h'(k+1)}+b$. Díky počáteční volbě n jako dostatečně velkého je tedy $\left|X_n^{k,h}\right| > |\mathcal{T}|$. Z toho vyplývá, že existují dvě různá $X_1, X_2 \in X_n^{k,h}$ taková, že pro slova $v_1 = \varpi(W_0, X_1)$ a $v_2 = \varpi(W_0, X_2)$ je typ prvních konfigurací posloupností $V_{v_1}^R$ a $V_{v_2}^R$ totožný.

$$
\begin{aligned}
v_1 &= (x_{1,j}\#)_1^{h+1}\, w_1\#w_0\,(x_{1,j}z_{1,j}\#)_1^h\,(w_i\#)_2^b\,(w_{b-i}\#)_0^b\,(z_{1,j}\#)_1^{h+1} \\
v_2 &= (x_{2,j}\#)_1^{h+1}\, w_1\#w_0\,(x_{2,j}z_{2,j}\#)_1^h\,(w_i\#)_2^b\,(w_{b-i}\#)_0^b\,(z_{2,j}\#)_1^{h+1}
\end{aligned}
$$

Ze slova v_1 utvořme slovo v_3 tak, že ve slově v_1 nahradíme oblast $(z_j\#)_1^{h+1}$ oblastí $(z_j\#)_1^{h+1}$ ze slova v_2.

$$
\begin{aligned}
v_3 &= (x_{1,j}\#)_1^{h+1}\, w_1\#w_0\, (x_{1,j}z_{1,j}\#)_1^h\, (w_i\#)_2^b\, (w_{b-i}\#)_0^b\, (z_{2,j}\#)_1^{h+1} \\
&= (x_{1,j}\#)_1^{h+1}\, w_1\#w_0\, (x_{2,j}z_{2,j}\#)_1^h\, (w_i\#)_2^b\, (w_{b-i}\#)_0^b\, (z_{2,j}\#)_1^{h+1}
\end{aligned}
$$

Všimněte si, že z definice jazyka $L_n^{k,h}$ vyplývá, že slovo $(x_{1,j}z_{1,j}\#)_1^h$ je identické s $(x_{2,j}z_{2,j}\#)_1^h$.

Vzhledem k tomu, že první konfigurace posloupností $V_{v_1}^R$ a $V_{v_2}^R$ mají stejný typ, že se žádná hlava během části výpočtu $V_{v_1}^L$ nedostane do oblasti $(z_j\#)_1^{h+1}$ slova v_3 a že v části výpočtu $V_{v_2}^R$ již není žádná hlava v oblasti $(x_j\#)_1^{h+1}$, spojením posloupností $V_{v_1}^L$ a $V_{v_2}^R$ získáme korektní přijímající výpočet pro slovo v_3. Slovo v_3 však nepatří do jazyka $L^{k,h}$ a tudíž automat M jej nepřijímá. Tím jsme získali požadovaný spor. □

Lemma 3.2.3 *Nechť $k \geqslant 2, h \in \mathbb{N}$. Pak existuje (k,h)-DFA automat M bez koincidence přijímající jazyk $L^{k,h}$.*

Důkaz: Automat M bude pracovat následovně. Postupně bude ověřovat rovnost jednotlivých kopií slov x_j, z_j a w_i. Pokud během porovnávání narazí na rozdíl, přesune všechny hlavy na konec vstupního slova a slovo odmítne. Jinak postupně prověří všechny dvojice slov a slovo přijme.

Prověřování bude provádět tak, že nejprve pomocí dvou vybraných hlav (hlavy h_0 a h_1) zkontroluje kopie slov x_j a zároveň pomocí kamenů označí začátky slov z_j v oblasti $(x_jz_j\#)_1^h$. Poté provede porovnání kopií slov w_0 až w_{k-1}, pak porovnání slov z_j a nakonec dokončí porovnávání dvojic w_k až $w_{\binom{k}{2}}$.

Porovnání dvojic slov x_j provede tak, že hlavu h_0 odpočítáním příslušného počtu znaků $\#$ přesune na první znak oblasti $(x_jz_j\#)_1^h$, tj. na první znak pravé kopie slova x_1. Poté pomocí hlavy h_1 a h_0 porovnává kopie slov x_j a vždy po dokončení porovnání (hlava h_1 dorazila na oddělovač x_j a x_{j+1}) položí kámen na začátek levé kopie slova z_j. Kamenů je k dispozici právě h.

Nyní hlavu h_1 umístí na začátek levé kopie w_1, hlavu h_0 umístí na začátek pravé kopii slova w_{k-1} (odpočítá příslušný počet znaků $\#$) a zbylých $k-2$ hlav rozmístí na začátky levých kopií slov w_2 až w_{k-1}. Nyní pomocí hlavy h_0 (kontroluje pravé kopie) a ostatních hlav (kontrolují vždy levou kopii) provede kontrolu slov w_{k-1} až w_0. (Levé kopie slov w_1 a w_0 kontroluje hlava h_1.)

Pomocí hlavy h_1 a h_0 se nyní provede porovnání slov z_j. Hlavou h_1 vždy najede na nejbližší kámen (začátek dalšího z_j) a provádí kontrolu proti hlavě h_0, která postupuje v oblasti $(z_j\#)_1^{h+1}$.

Tím se hlava h_1 dostane za oblast $(x_j z_j \#)_1^h$ a spolu se zbývajícími $k-2$ hlavami provede prověření zbylých $\binom{k}{2}+1-k=\binom{k-1}{2}$ dvojic slov w_i (dvojice w_k až $w_{\binom{k}{2}}$) a to obdobných způsobem, jako se to provedlo pro dvojice slov w_0 až w_{k-1}. Pokud jsou kopie shodné, automat dojede všemi hlavami na ukončující znak \$ a slovo přijme.

Takto automat ověřuje, zda vstupní slovo splňuje všechny podmínky na jazyk $L^{k,h}$, tedy přijímá právě tento jazyk. $\square$

Důsledkem těchto tvrzení je Věta O hierarchii.

Věta 3.2.4 (O hierarchii) *Pro všechna $k \geqslant 2, h \in \mathbb{N}$ platí, že $R_{k,h+1}^{ND} - R_{k,h} \neq \emptyset$ a $R_{k,h}^{ND} - \bigcup_{h' \in \mathbb{N}} R_{k-1,h'} \neq \emptyset$, a tedy:*

$$R_{k-1,h}^{?} \subsetneq R_{k,h}^{?}, \quad R_{k,h}^{?} \subsetneq R_{k,h+1}^{?}, \quad R_{k,h}^{?} - \bigcup_{h' \in \mathbb{N}} R_{k-1,h'}^{?} \neq \emptyset,$$

kde $^?$ v exponentu třídy $R_{k,h}$ lze vždy v celém výrazu nahradit jedním ze symbolů $\left\{^{\epsilon}, ^{D}, ^{N}, ^{ND}\right\}$.

Následující věty ukazují, že kameny lze nahradit jistým počtem hlav.

Věta 3.2.5 *Nechť $k \geqslant 2, h > h'$ jsou přirozená čísla. Pak $R_{k,h} \subsetneq R_{k+\lfloor \frac{k}{2} \rfloor (h-h'),h'}$.*

Důkaz: Máme ukázat, že pro daná čísla k, h, h' lze každý jazyk ze třídy $R_{k,h}$ rozpoznávat $\left(k+\left\lfloor \frac{k}{2} \right\rfloor (h-h'), h'\right)$-FA automatem, a dále, že ve třídě $R_{k+\lfloor \frac{k}{2} \rfloor (h-h'),h'}$ existuje jazyk, který nelze rozpoznávat žádným (k,h)-FA automatem.

Jazykem, ktrý nelze rozpoznávat žádným (k,h)-FA automatem, ale který lze rozpoznávat (k',h')-FA automatem, kde $k' = k + \left\lfloor \frac{k}{2} \right\rfloor (h-h')$, je podle Lemmatu 3.1.4 jazyk $K_{1,\binom{k'}{2},\binom{k'}{2}}$ z Příkladu 3.1.4.

Ukažme tedy neostrou inkluzi uvedených tříd.

Vezměme si nějaká čísla $k \geqslant 2$, $h > h'$ a (k,h)-FA automat M. Položme $k' = k + \left\lfloor \frac{k}{2} \right\rfloor (h-h')$. Chceme sestrojit (k',h')-FA automat M' rozpoznávající stejný jazyk jako automat M.

Tento automat M' bude pracovat tak, že bude simulovat činnost automatu M, přičemž místo chybějících kamenů bude pro označení místa nad vstupním slovem používat hlavy, které má navíc. Tedy k vybraných hlav (tzv. *původní hlavy*) automatu M' bude simulovat činnost hlav automatu M. Zbylé hlavy (tzv. *pomocné hlavy*) budou simulovat kameny. Automat M' si bude evidovat počet položených skutečných kamenů.

Když se některá původní hlava rozhodne položit kámen a všechny skutečné kameny jsou již položeny, pak se jedna z pomocných hlav přesune na pozici této původní hlavy, kam se má kámen položit, a tato pomocná hlava bude od této chvíle označovat místo, kde kámen leží. Původní hlavy se nad políčkem s takovouto hlavou chovají, jako by tam skutečně kámen ležel.

Každá pomocná hlava si tedy pamatuje, zda právě označuje místo, kde leží kámen, nebo zda je právě nevyužitá. Pokud některá původní hlava na takovém políčku chce kámen zvednout, pak si pomocná hlava označí, že je nevyužitá.

Aby šla tato konstrukce provést, je potřeba použít automat s koincidencí. Automat M' musí totiž sledovat vzájemné pořadí všech svých hlav nad vstupním slovem, aby mohl rozlišit které dvě hlavy spolu stojí nad jedním políčkem. (Z druhé kapitoly víme, že to je možné sledovat.)

Pro takovouto simulaci kamenů pomocí hlav potřebujeme, aby za každou původní hlavou,[10] bylo během výpočtu k dispozici $h - h'$ pomocných hlav. Pokud tedy každé původní hlavě dáme k dispozici $h - h'$ pomocných hlav, které bude moci "pokládat" pouze ona, pak v průběhu výpočtu bude za každou původní hlavou neustále k dispozici potřebný počet pomocných hlav. Takovýto automat by potřeboval $k\,(h - h')$ pomocných hlav.

My chceme sestrojit automat, který používá pouze $\left\lfloor \frac{k}{2} \right\rfloor (h - h')$ pomocných hlav. Toho dosáhneme tak, že výchozí automat M nejprve upravíme.

Upravíme ho tak, aby pořadí jeho hlav bylo nad vstupním slovem během výpočtu stále stejné.[11] Další úpravou, kterou provedeme, bude to, že automat bude kámen pokládat pouze tehdy, pokud hlava, která ho pokládá, zároveň dané políčko opustí. Pokud by chtěla hlava kámen položit, avšak zůstávala by na místě, pak si tato hlava pouze označí, že pod ní leží kámen, avšak žádný kámen skutečně nepoloží. Ostatní hlavy se na takovém políčku budou

[10] Výraz "*za hlavou*" označuje pozice, které jsou blíže k počátku vstupního slova. Výraz "*před hlavou*" naopak označuje pozice, které jsou blíže ke konci vstupního slova, to jest blíže ke koncovému symbolu \$.

[11] Úprava automatu tak, aby měl stále stejné pořadí hlav nad vstupním slovem, je podle úvah ve druhé kapitole možná.

chovat, jako kdyby tam kámen skutečně ležel.[12] (Pokud nějaká hlava bude chtít takový kámen zvednout, označení pokládající hlavy se zruší.) Pokud bude takovýto kámen stále "položen" a hlava, která jej položila, bude mět opustit dané políčko, pak kámen skutečně položí a pohne se.

Takovýto automat bude mít zřejmě tu vlastnost, že pokud nějaká hlava kámen skutečně položí, pak ho jistě zvedne nějaká jiná hlava.

Poslední úpravu uděláme tu, že poslední hlava bude sbírat ležící kameny tak, aby za ní žádný nezůstal ležet. Z ostatních vlastností tohoto automatu tedy plyne, že takováto hlava nebude žádné kameny pokládat.

Z tohoto automatu již snadno sestrojíme automat, který bude potřebovat pouze $\left\lfloor \frac{k}{2} \right\rfloor (h - h')$ pomocných hlav a bude rozpoznávat stejný jazyk jako automat M.

Nechť jsou během výpočtu hlavy tohoto automatu uspořádány nad vstupním slovem neustále v pořadí $1, 2, \dots, k$, přičemž hlava 1 je nejvíce vpředu. Pak každé dvojici hlav $2i - 1$ a $2i$, kde $i = 1, \dots, \left\lfloor \frac{k}{2} \right\rfloor$, dáme k dispozici $h - h'$ pomocných hlav. Každá dvojice hlav bude tedy mít k dispozici svých $h - h'$ hlav, které může "pokládat" místo kamenů. (Zvedat takové "kameny" však může kterákoliv hlava podle požadavků simulovaného automatu nazávisle na tom, zda jí tyto pomocné hlavy patří nebo ne.)

Pokud je k liché, nebude mít hlava k k dispozici žádné pomocné hlavy, neboť tato hlava žádné kameny nepokládá.

Tak získáme výsledný automat, který bude simulovat činnost původního automatu stejným způsobem jako výše popsaná, méně efektivní varianta.

Ukážeme, že každá z původních hlav bude mít vždy, když bude potřebovat položit kámen simulovaný hlavou, k dispozici jednu ze svých pomocných hlav, která bude umístěna až za její pozicí. (Toto bude platit tehdy, pokud původní simulovaný automat M má v daný okamžik výpočtu dostatek kamenů.)

Pro první hlavu z každého páru to jistě platí, neboť pomocné hlavy mohou být ve vstupním slově nejdále na její pozici (nikdo jiný kromě hlav tohoto páru je "položit" nesmí a sami se bezdůvodně nepohybují.)

Pro druhou hlavu to samozřejmě platí také. Kdykoliv má při simulaci položit další kámen

[12] Musíme opět evidovat pořadí hlav nad vstupním slovem.

a všechny skutečné jsou zrovna položeny, pak musí některá z jejích $h - h'$ pomocných hlav být za její pozicí. Kdyby nebyla, pak by všechny pomocné hlavy musely být před ní, což by znamenalo, že je první hlava z páru položila. Jelikož však nad vstupním slovem není mezi hlavami v páru žádná jiná původní hlava, která by kameny mohla zvednout, je v daný okamžik celkově položeno h kamenů a tedy ani simulovaný automat M by nemohl kámen položit. Pokud tedy kámen může položit simulovaný automat M, pak jej může položit i náš simulující automat M'.

Tedy za každou hlavou je v průběhu výpočtu vždy dostatečný počet hlav pro simulaci kamenů automatu M.

Vzhledem k simulaci činnosti automatu M je z konstrukce zřejmé, že pro dané slovo existuje přijímající výpočet automatu M právě tehdy, když pro něj existuje přijímající výpočet automatu M'. Zkonstruovaný automat tedy přijímá stejný jazyk jako automat M.

Tím je tvrzení dokázáno. $\square$

Jelikož se v důkazu nevyužívalo žádných nedeterministických vlastností automatu a nalezený automat je tedy stejného typu jako automat původní, platí i následující tvrzení:

Věta 3.2.6 *Nechť $k \geqslant 2, h > h'$ jsou přirozená čísla. Pak $R^D_{k,h} \subsetneq R^D_{k+\lfloor \frac{k}{2} \rfloor (h-h'),h'}$.*

Dá se předpokládat, že tento odhad je značně nepřesný. Je otevřená otázka, jaký minimální počet hlav je na nahrazení kamenů skutečně potřeba. Dá se však obecně předpokádat (například z Věty O lustrování), že na nahrazení i kamenů je potřeba alespoň i hlav.

Z uvedených tvrzení plyne následující důsledek.

Důsledek 3.2.7 *Platí: $\bigcup_{k \geqslant 1} R_{k,0} = \bigcup_{\substack{k \geqslant 1 \\ h \in \mathbb{N}}} R_{k,h}$ a $\bigcup_{k \geqslant 1} R^D_{k,0} = \bigcup_{\substack{k \geqslant 1 \\ h \in \mathbb{N}}} R^D_{k,h}$.*

Tedy třída jazyků rozpoznávaných vícehlavými konečnými automaty s koincidencí je stejná jako třída jazyků rozpoznávaných vícehlavými konečnými automaty s kameny. Kameny tedy nepřinášejí principiální zvětšení síly vícehlavých konečných automatů.

Vzhledem k Větě 2.2.1 platí i následující důsledek.

Důsledek 3.2.8 *Platí: $\bigcup_{k \geqslant 1} R_{k,0} = \bigcup_{\substack{k \geqslant 1 \\ h \in \mathbb{N}}} R^N_{k,h}$ a $\bigcup_{k \geqslant 1} R^D_{k,0} = \bigcup_{\substack{k \geqslant 1 \\ h \in \mathbb{N}}} R^{ND}_{k,h}$.*

3.3 Deterministické a nedeterministické (k,h)-FA automaty

V tomto oddíle ukážeme fakt, že nedeterministické automaty s kameny jsou striktně silnější než deterministické automaty s kameny. Pro vícehlavé automaty toto tvrzení ukázal Yao a Rivest [13] s využitím speciálního jazyka. Z jejich tvrzení a Důsledků 3.2.7 a 3.2.8 plyne i námi dokazované tvrzení. My však jeho důkaz provedeme s využitím jazyka palindromů.

Nejprve ukážeme, že jazyk palindromů, tj. slov $x\#x^R$, nelze přijímat žádným vícehlavým konečným automatem s kameny. Dále ukážeme, že doplněk tohoto jazyka lze přijímat nedeterministickým $(2,0)$-FA. Důsledkem toho získáme tvrzení, že $R^N_{2,0} - \left(\bigcup_{\substack{k\geqslant 1\\ h\in\mathbb{N}}} R^D_{k,h}\right) \neq \emptyset$, neboť jazyky rozpoznávané deterministickými (k,h)-FA jsou uzavřeny na doplněk (viz Věta 4.1.3 v další kapitole).

Lemma 3.3.1 *Nechť* $L = \left\{x\#x^R \mid x \in \{0,1\}^*\right\} \subseteq \{0,1,\#\}^*$. *Pak pro žádná přirozená čísla* k, h *neexistuje* (k,h)*-FA přijímající jazyk* L.

Důkaz: Důkaz provedeme sporem. Nechť pro nějaká přirozená čísla k a h existuje (k,h)-FA přijímající jazyk L. Vezměme si jeden z těchto automatů a označme jej jako M. Zvolme si dále dostatečně velké přirozené číslo n. Položme $b = \binom{k}{2}$.

Zkoumejme nyní slova $(x_i)_{i=0}^{b} \# \left(x^R_{b-i}\right)_{i=0}^{b}$, kde $x_0,\ldots,x_b \in \Sigma^n$. Tato slova zřejmě patří do jazyka L. Při přijímajících výpočtech pro tato slova však automat M nemůže lustrovat všechny dvojice podslov x_i a x_i^R, neboť těchto dvojic je $b+1$ a každé dvě hlavy automatu, vhledem k rozmístění dvojic podslov, spolu mohou lustrovat nejvýše jednu dvojici podslov x_i a x_i^R, tedy dohromady mohou lustrovat nejvýše b dvojic.

Definujme si nyní pro každé $j \in \{0,\ldots,b\}$ a pro každá slova $x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b$ množinu $U_{j,x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b} = \{x_j \mid x_j \in \Sigma^n$ & existuje přijímající výpočet automatu M pro slovo $(x_i)_{i=0}^{b} \# \left(x^R_{b-i}\right)_{i=0}^{b}$, při kterém se neprovádí lustrování dvojice podslov x_j a x_j^R $\}$. Snadno nahlédneme, že každé ze slov $(x_i)_{i=0}^{b} \# \left(x^R_{b-i}\right)_{i=0}^{b}$ musí přispět do minimálně jedné z množin $U_{j,x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b}$ a žádná dvě různá slova nemohou přispět do téže množiny $U_{j,x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b}$ stejným slovem x_j. Z uvedeného vyplývá, že součet velikostí všech množin $U_{j,x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b}$ musí být alespoň $2^{\left(\binom{k}{2}+1\right)n}$.

Množiny $U_{j,x_0,\ldots,x_{j-1},x_{j+1},\ldots,x_b}$ splňují předpoklady Důsledku 3.1.2 (funkcí f je funkce reverze slov, $a_0 = (x_i)_{i=0}^{j-1}$, $a_1 = (x_i)_{i=j+1}^{b} \# \left(x^R_{b-i}\right)_{i=0}^{b-j-1}$, $a_2 = \left(x^R_{b-i}\right)_{i=b-j+1}^{b}$) a pro jejich velikost tedy platí odhad $\left|U_{j,x_0,\dots,x_{j-1},x_{j+1},\dots,x_b}\right| < \widetilde{p}_{k,h,s}\left(2\binom{k}{2}n+1\right)$. Těchto množin je $\left(\binom{k}{2}+1\right)2^{\binom{k}{2}n}$, tedy součet velikostí všech těchto množin je menší než $\widetilde{p}_{k,h,s}\left(2\binom{k}{2}n+1\right)\left(\binom{k}{2}+1\right)2^{\binom{k}{2}n}$. Vzhledem k volbě n jako dostatečně velkého je však tento horní odhad menší než předchozí dolní odhad na tento součet, což vede ke sporu.

Tím je nalezen požadovaný spor, a automat M nemůže přijímat jazyk L.

Jazyk L nelze tedy přijímat pro žádná k a h žádným (k,h)-FA automatem. $\square$

Lemma 3.3.2 *Jazyk* $L = \left\{x\#y|\ x,y \in \{0,1\}^* \quad \& \quad x \neq y^R\right\}$ *lze rozpoznávat nedeterministickým* $(2,0)$*-FA automatem bez koincidence.*

DŮKAZ: Hledaný automat bude nad vstupním slovem pracovat následujícím způsobem. Bude předpokládat, že vstupní slovo je tvaru $x\#y$, tedy že obsahuje právě jeden znak #. Pokud by se v průběhu výpočtu ukázalo, že tomu tak není, bude vstupní slovo odmítnuto.

Automat nejprve umístí náhodně jednu z hlav nad libovolný znak slova x, přičemž vykoná alespoň jeden krok doprava a zastaví se nejpozději na symbolu # (je-li $x = \epsilon$, pak stačí ověřit, že $y \neq \epsilon$), přičemž druhou hlavu ponechá zatím na počátku.

Pokud první z hlav není nad symbolem #, pak bude postupovat následujícím způsobem. Automat bude simultáně oběmi hlavami pohybovat doprava, dokud první z nich nedorazí na symbol #. Nyní si automat zapamatuje znak slova x pod druhou, levější, hlavou. Poté bude opět simultáně pohybovat hlavami, dokud druhá z nich nedorazí na znak #. Jelikož vzdálenost obou hlav byla po celou tuto dobu stále stejná, nachazí se první hlava nad znakem slova y, který je od symbolu # vzdálen stejně, jako zapamatovaný znak slova x přečtený druhou hlavou. Pokud se tyto znaky liší, je slovo přijato, jinak je odmítnuto. V průběhu výpočtu však první z hlav mohla dorazit na symbol ukončení \$ dříve, než druhá hlava dorazila na znak #. V takovém případě vstupní slovo patří do jazyka L, neboť slova x a y se liší délkou, nejsou tedy svými zrcadlovými obrazy, a automat toto slovo přijme.

Pokud při prvním náhodném výběru dorazí první hlava až na znak #, pak automat bude zjišťovat, zda je slovo x delší než slovo y. Pokud ano, bude vstupní slovo přijato, jinak bude odmítnuto. Ověření se provede simultáním posunem hlav doprava, dokud druhá z hlav taktéž nedorazí na znak #. Nyní stačí ověřit, zda první hlava dorazila ve stejný okamžik na znak ukončení vstupního slova \$.

Popsaný automat přijímá právě ta slova, jež jsou tvaru $x\#y$ a kde buď jsou slova x a y různé délky, nebo se jejich zrcadlový obraz liší od druhého slova v alespoň jednom znaku. Přijímá tedy právě jazyk L. Tento automat je zřejmě bez koincidence. $\square$

Lemma 3.3.3 *Jazyk* $L = \left\{x\#y|\ x, y \in \{0,1\}^* \quad \& \quad x \neq y^R\right\}$ *nelze přijímat žádným deterministickým* (k,h)*-FA automatem pro žádná* k *ani* h*.*

DŮKAZ: Jelikož třídy jazyků $R^D_{k,h}$ jsou uzavřeny na doplněk a průnik s regulární množinou (viz další kapitola), kdyby jazyk L patřil do třídy $R^D_{k,h}$ pro nějaká konkrétní k a h, patřil by do této třídy i jazyk $\left\{x\#x^R|\ x \in \{0,1\}^*\right\} = \overline{L} \cap \{x\#y|\ x, y \in \{0,1\}^*\}$. Z přechozích lemmat však víme, že tento jazyk nelze rozpoznávat žádným vícehlavým konečným automatem s kameny. $\square$

Důsledek 3.3.4 *Existují jazyky, které lze rozpoznávat nedeterministickým dvouhlavým konečným automatem bez kamenů a bez koincidence, avšak žádným deterministickým vícehlavým konečným automatem s kameny. Tedy* $R^N_{2,0} - \left(\bigcup_{\substack{k \geqslant 1\\ h \in \mathbb{N}}} R^D_{k,h}\right) \neq \emptyset$*.*

Důsledek 3.3.5 *Pro každé* $k \geqslant 2, h$ *přirozené číslo* $R^D_{k,h} \subsetneq R_{k,h}$ *a* $R^{ND}_{k,h} \subsetneq R^N_{k,h}$*.*

3.4 Automaty s koincidencí a bez koincidence

V tomto oddíle ukážeme vztah mezi automaty s koincidencí a bez koincidence. Již ve druhé kapitole jsme uvedli, že koincidenci hlav lze nahradit testem pomocí kamene, pokud máme takový volný kámen k dispozici (Věta 2.2.1 – $R_{k,h} \subseteq R^N_{k,h+1}$ a $R^D_{k,h} \subseteq R^{ND}_{k,h+1}$.) Zůstala nezodpovězená otázka, zda takový volný, speciálně vyhrazený kámen je pro simulaci koincidence hlav pomocí kamenů bezpodmínečně nutný. Pro vícehlavé automaty bez kamenů a $k \geqslant 3$ se obecně ví, že $R^N_{k,0} \subsetneq R_{k,0}$ (Harrison a Ibarra [6], Sudborough [15].)

V tomto oddíle ukážeme, že pro počet hlav větší než 4 je jeden volný vyhrazený kámen nutný a že tedy pro $k \geqslant 4$ je $R^N_{k,h} \subsetneq R_{k,h}$ a $R^{ND}_{k,h} \subsetneq R^D_{k,h}$. Naopak se ukazuje, že pro dvouhlavé automaty takový vyhrazený kámen není potřeba, a tedy $R_{2,h} = R^N_{2,h}$ a $R^D_{2,h} = R^{ND}_{2,h}$, pokud $h \geqslant 1$.

Pro automaty se třemi hlavami a několika kameny a pro automaty se dvěma hlavami bez kamenů nám však není nic známo a tato otázka zůstává i nadále nezodpovězená.

Je zřejmé, že $R^N_{k,h} \subseteq R_{k,h}$ a $R^{ND}_{k,h} \subseteq R^D_{k,h}$, a tedy zbývá pro každé $k \geqslant 4$ a každé h ukázat, že existuje jazyk, který lze rozpoznávat (k,h)-FA automatem s koincidencí, ale nejde rozpoznávat žádným (k,h)-FA automatem bez koincidence.

Jazyk, který pro to použijeme, se bude podobat jazyku $L^{k,h}$ z Definice 3.2.1, jenž jsme použili pro důkaz Věty O hierarchii. Ve shodě s dřívějším značením jej proto budeme taktéž označovat $L^{k,h}$ (Definice 3.4.1). Nyní popíšeme struktura slov tohoto jazyka.

Slova z jazyka $L^{k,h}$ budou opět obsahovat oblasti $(x_j\#)_1^h, (x_j z_j\#)_1^h$ a $(z_j\#)_1^h$ uspořádané takto:

$$(x_j\#)_1^h \qquad \cdots \quad \cdots \qquad (x_j z_j\#)_1^h \qquad \cdots \quad \cdots \quad \cdots \qquad (z_j\#)_1^h\,.$$

Při přijímacím výpočtu opět každý automat donutíme, aby v určitý okamžik měl všechny hlavy rozmístěny v relativně malých podslovech mimo tyto oblasti. To bude znamenat, že si bude muset v oblasti $(x_j z_j\#)_1^h$ pomocí kamenů vyznačit začátky slov z_j. (Celkový průběh výpočtu budeme opět řídit přítomností dvojic slov, které bude nutno lustrovat. Budou to slova $u_{i,j}$, $v_{i,j}$ a $w_{i,j}$.)

K tomu, abychom hlavy v určitou chvíli výpočtu donutili být rozmístěné v malém úseku vstupního slova, opět použijeme podslova rozmístěná následujícím způsobem:

$$\cdots \quad u_{1,1}\#u_{1,2}\#u_{1,3}\# \qquad \cdots \quad \cdots \qquad \alpha_2 u_{1,3}\#u_{1,2}\#u_{1,1}\# \qquad \cdots \quad \cdots,$$

kde α_2 bude slovo z jazyka $K_{1,\binom{k}{2}-3,\binom{k}{2}-3}$ z Příkladu 3.1.4, tedy:

$$\alpha_2 = (w_{2,i}\#)_1^{b_2}\,(w_{2,b_2-i+1}\#)_1^{b_2}\,,$$

kde $b_2 = \binom{k}{2} - 3$. Takto rozmístěná podslova si vynucují přítomnost všech hlav v tu samou chvíli.

Před oblast α_2 umístíme ještě podslovo z jazyka $K_{1,\binom{k-1}{2},\binom{k-1}{2}}$:

$$\cdots \quad u_{1,1}\#u_{1,2}\#u_{1,3}\# \qquad \cdots \quad \cdots \qquad \alpha_1\alpha_2 u_{1,3}\#u_{1,2}\#u_{1,1}\# \qquad \cdots \quad \cdots,$$

čímž si vynutíme to, že při lustrování slov $u_{1,1}$, $u_{1,2}$, $u_{1,3}$ bude levé kopie těcho slov lustrovat jedna a táž hlava. K lustrování slov v oblasti α_1 je totiž potřeba současná přítomnost $k-1$

hlav.

Dále se ve slově z $L^{k,h}$ budou mnohokrát opakovat šestice slov $u_{j,1}\#u_{j,2}\#u_{j,3}\#v_{j,1}\#v_{j,2}\#v_{j,3}\#$ a k nim obrácené $u_{j,3}\#u_{j,2}\#u_{j,1}\#v_{j,3}\#v_{j,2}\#v_{j,1}\#$:

$$\cdots \quad u_{j,1}\#u_{j,2}\#u_{j,3}\#v_{j,1}\#v_{j,2}\#v_{j,3}\# \quad \cdots \alpha_1\alpha_2 u_{1,3} \quad \cdots u_{j,3}\#u_{j,2}\#u_{j,1}\#v_{j,3}\#v_{j,2}\#v_{j,1}\# \ \cdots .$$

Jelikož v α_2 bude vázáno $k-4$ hlav čekajících na příchod poslední hlavy, bude muset lustrování kopií slov $u_{i,j}$ a $v_{i,j}$ probíhat tak, že levé kopie bude lustrovat poslední hlava, která též lustrovala levé kopie slov $u_{1,1}$, $u_{1,2}$, $u_{1,3}$, a pravé kopie budou lustrovat tytéž tři hlavy, co lustrovaly pravé kopie slov $u_{1,1}$, $u_{1,2}$, $u_{1,3}$.

Šestice slov $u_{j,1}\#u_{j,2}\#u_{j,3}\#v_{j,1}\#v_{j,2}\#v_{j,3}\#$ budeme označovat jako β_j a jejich protějšky jako $\overline{\beta}_j$.

Mezi po sobě jdoucími β_j bude vždy jeden znak $a_j \in \{0,1\}$ a mezi po sobě jdoucími $\overline{\beta}_j$ bude slovo $\overline{a}_j \in \{0\}^*$:

$$\cdots \quad \cdots \qquad \beta_j a_j \beta_{j+1} \quad \cdots \quad \alpha_1\alpha_2 u_{1,3} \quad \cdots \quad \overline{\beta}_j \overline{a}_j \overline{\beta}_{j+1} \quad \cdots .$$

Slova a_j a $\overline{a}_j$ budou svázána relací $\left(a_j = 1 \iff \overline{a}_j \in \left\{0^{n^2} \mid n \in \mathbb{N}\right\}\right)$.

O jazyce $\left\{0^{n^2} \mid n \in \mathbb{N}\right\}$ je známo, že ho nelze rozpoznávat žádným $(k,0)$-FA automatem bez koincidence, ale lze ho rozpoznávat $(3,0)$-FA automatem s koincidencí.

Tím, že dvojic slov a_j, $\overline{a}_j$ bude velké množství, donutíme automat rozhodovat o příslušnosti slova $\overline{a}_j$ do uvedeného jazyka v době, kdy jedna z hlav bude ve slově $\beta_j a_j \beta_{j+1}$, tři budou ve slově $\overline{\beta}_j \overline{a}_j \overline{\beta}_{j+1}$ a zbylé budou v oblasti α_2, tedy právě v době, kdy kameny budou položeny mimo dosah hlav. Automat bez koincidence však takové rozhodnutí nemůže dělat správně pro všechna slova z $\{0\}^*$, čímž ukážeme, že jazyk $L^{k,h}$ nelze přijímat (k,h)-FA automatem bez koincidence.

Pro vlastní důkaz budeme opět používat podmožiny jazyka $L^{k,h}$, jazyky $L_n^{k,h}$ (Definice 3.4.3), které budou mít délku podslov $w_{i,j}$, $u_{i,j}$ a $v_{i,j}$ v oblastech α_1, α_2 a ve všech β_j rovnu n. Slova $\overline{a}_j$ budou z množiny $\{0\}^{<n^{10}}$. Slova x_j a z_j budeme brát z množiny $\{0\}^{\leqslant n^{\lfloor \log_2 n \rfloor}}$ a počet oblastí β_j a tedy i počet slov a_j a $\overline{a}_j$ bude $\lfloor \log_2 n \rfloor^2$.

Následují formální definice jazyků $L^{k,h}$ a $L_n^{k,h}$.

Připomeňte si definici jazyků $K_{1,\binom{k-1}{2},\binom{k-1}{2}}$ a $K_{1,\binom{k}{2}-3,\binom{k}{2}-3}$ z Příkladu 3.1.4.

Definice 3.4.1 *Nechť $k \geqslant 4$ a $h \geqslant 1$ jsou přirozená čísla. Definujeme jazyk $L^{k,h} \subseteq \{0,1,\#,¢\}^*$ následovně. Do jazyka $L^{k,h}$ patří právě každé slovo*

$$\gamma = (x_j\#)_1^h \left(\beta_j a_j \#\right)_0^{l-1} \beta_l ¢ \, (x_j z_j \#)_1^h \, \alpha_1 \alpha_2 \left(\overline{\beta}_j \overline{a}_j \#\right)_1^{l-1} \overline{\beta}_l ¢ \, (z_j\#)_1^h$$

splňující následující podmínky:

1. $l \in \mathbb{N}$.
2. $\forall 1 \leqslant j \leqslant h$ *jsou slova* $x_j, z_j \in \{0,1\}^*$.
3. $\forall 0 \leqslant j \leqslant l$ *je* $\beta_j = u_{j,1}\#u_{j,2}\#u_{j,3}\#v_{j,1}\#v_{j,2}\#v_{j,3}\#$, *kde* $u_{j,1}$, $u_{j,2}$, $u_{j,3}$, $v_{j,1}$, $v_{j,2}$, $v_{j,3} \in \{0,1\}^*$, *a* $\overline{\beta}_j = u_{j,3}\#u_{j,2}\#u_{j,1}\#v_{j,3}\#v_{j,2}\#v_{j,1}\#$.
4. $\forall 0 \leqslant j < l$ *je jednoznakové slovo* $a_j \in \{0,1\}$ *a slovo* $\overline{a}_j \in \{0\}^*$. *Slova* a_j *a* $\overline{a}_j$ *splňují relaci:*
$$a_j = 1 \iff \overline{a}_j \in \left\{0^{m^2} \mid m \in \mathbb{N}\right\}.$$
5. α_1 *je slovo z jazyka* $K_{1,\binom{k-1}{2},\binom{k-1}{2}}$ *a* α_2 *je slovo z jazyka* $K_{1,\binom{k}{2}-3,\binom{k}{2}-3}$.

Značení 3.4.2 *Podslovo α_1 slova z jazyka $L^{k,h}$ budeme přirozeně dělit na slova $w_{1,1}, \ldots, w_{1,b_1}$ z množiny $\{0,1\}^*$, kde $b_1 = \binom{k-1}{2}$. Tedy $\alpha_1 = (w_{1,i}\#)_1^{b_1} (w_{1,b_1-i+1}\#)_1^{b_1}$. Stejně tak budeme podslovo α_2 dělit na slova $w_{2,1}, \ldots, w_{2,b_2}$, kde $b_2 = \binom{k}{2} - 3$, tedy $\alpha_2 = (w_{2,i}\#)_1^{b_2} (w_{2,b_2-i+1}\#)_1^{b_2}$.*

Definice 3.4.3 *Nechť $k \geqslant 4$ a $h \geqslant 1$ jsou přirozená čísla. Pak pro přirozené číslo $n \geqslant 2$ definujme jazyk $L_n^{k,h} \subseteq L^{k,h}$ následovně. Do jazyka $L_n^{k,h}$ patří každé slovo*

$$\gamma = (x_j\#)_1^h \left(\beta_j a_j \#\right)_0^{l-1} \beta_l ¢ \, (x_j z_j \#)_1^h \, \alpha_1 \alpha_2 \left(\overline{\beta}_j \overline{a}_j \#\right)_1^{l-1} \overline{\beta}_l ¢ \, (z_j\#)_1^h$$

z jazyka $L^{k,h}$, které splňuje následující podmínky:

1. $l = \lfloor \log_2 n \rfloor^2$.
2. $\forall 1 \leqslant j \leqslant h$ *jsou* $x_j, z_j \in \{0\}^*$, $x_j \neq \epsilon$ *a platí* $|x_j z_j| = n^{\lfloor \log_2 n \rfloor}$.
3. $\forall 0 \leqslant j \leqslant l$ *je* $|u_{j,1}| = |u_{j,2}| = |u_{j,3}| = |v_{j,1}| = |v_{j,2}| = |v_{j,3}| = n$.

4. $\forall 0 \leqslant j < l$ *je* $|\overline{a}_j| < n^{10}$.

5. $\forall 1 \leqslant i \leqslant \binom{k-1}{2}$ *je* $|w_{1,i}| = n$. $\forall 1 \leqslant i \leqslant \binom{k}{2} - 3$ *je* $|w_{2,i}| = n$.

Definice 3.4.4 *Maximální délku slov z jazyka* $L_n^{k,h}$ *označme*[13]*:*

$$\begin{aligned} d_{n,k,h} &= (2n+2)\left(\underbrace{6\lfloor\log_2 n\rfloor^2 + 6}_{\beta_j} + \underbrace{\binom{k-1}{2}}_{w_{1,i}} + \underbrace{\binom{k}{2} - 3}_{w_{2,i}}\right) \\ &\quad + \underbrace{3h + 2hn^{\lfloor\log_2 n\rfloor}}_{x_j,z_j} + \underbrace{\left(n^{10}+2\right)\lfloor\log_2 n\rfloor^2}_{a_j,\overline{a}_j} + \underbrace{2}_{\cent}. \end{aligned}$$

Značení 3.4.5 *Každé slovo* x *z jazyka* $L^{k,h}$ *se přirozeně skládá z několika podslov, která jsou v definici jazyka* $L^{k,h}$ *označená jako* $x_j, z_j, u_{i,j}, v_{i,j}, w_{i,j}, \overline{a}_j$ *pro příslušná* i *a* j*. Každé podslovo* $u_{i,j}, v_{i,j}$ *a* $w_{i,j}$ *se ve slově* x *vyskytuje dvakrát; tyto výskyty budeme označovat jako* ***levou*** *a* ***pravou kopii*** *příslušného podslova* $u_{i,j}, v_{i,j}$ *či* $w_{i,j}$*.*
Pro dané slovo x *z jazyka* $L^{k,h}$ *budeme podslova* $(x_j\#)_1^h$, $(x_j z_j\#)_1^h$ *a* $(z_j\#)_1^h$ *nazývat* ***oblastmi slova*** x*. Dále taktéž podslova* $\alpha_1, \alpha_2, \beta_j$ *a* $\overline{\beta}_j$ *slova* x *budeme nazývat* ***oblastmi tohoto slova.***

Nyní si pro jazyk $L^{k,h}$ dokážeme obdobu Lemmatu 3.2.1.

Definice 3.4.6 *Nechť* $k \geqslant 4, h \geqslant 1, n \geqslant 2$ *jsou přirozená čísla. Definujme množiny* $W_n^{k,h}$ *a* $X_n^{k,h}$ *takto:*[14]

$$\begin{aligned} W_n^{k,h} = \ & \Big\{\Big\langle w_{1,1}, \ldots, w_{1,\binom{k-1}{2}}, w_{2,1}, \ldots, w_{2,\binom{k}{2}-3}, u_{0,1}, \ldots, u_{l,3}, v_{0,1}, \ldots, v_{l,3}\Big\rangle \mid \\ & \forall 0 \leqslant i \leqslant \tbinom{k-1}{2}, w_{1,i} \in \{0,1\}^n \ \& \ \forall 0 \leqslant i \leqslant \tbinom{k}{2} - 3, w_{2,i} \in \{0,1\}^n \\ & l = \lfloor\log_2 n\rfloor^2,\ \forall 0 \leqslant j \leqslant l, \forall 1 \leqslant i \leqslant 3,\ u_{j,i}, v_{j,i} \in \{0,1\}^n\Big\} \end{aligned}$$

a

$$\begin{aligned} X_n^{k,h} = \ & \Big\{\langle x_1, \ldots, x_h, \overline{a}_0, \ldots, \overline{a}_{l-1}\rangle \mid \forall 1 \leqslant i \leqslant h, \epsilon \neq x_i \in \{0\}^{\leqslant n^{\lfloor\log_2 n\rfloor}} \\ & l = \lfloor\log_2 n\rfloor^2,\ \forall 0 \leqslant i < l,\ \overline{a}_i \in \{0\}^{<n^{10}}\Big\}. \end{aligned}$$

[13] *Všimněte si, že pro pevné* k *a* h *existuje konstanta* c *taková, že od jistého* n_0 *pro všechna* $n \geqslant n_0$ *je* $d_{n,k,h} \leqslant cn^{\lfloor\log_2 n\rfloor}$*. Nejrychleji rostoucím termem je délka* z_j, x_j*.*

[14] *Množina* $W_n^{k,h}$ *nezávisí na velikosti* h*. Index* h *je zde pouze pro přehlednost zápisu.*

Pro tyto množiny definujme přirozeným způsobem bijekci $\varpi : W_n^{k,h} \times X_n^{k,h} \longrightarrow L_n^{k,h}$. *(*$a_i$ *jsou jednoznačně určená vztahem* $\left(a_i = 1 \iff \exists m \in \mathbb{N},\ \overline{a}_i = 0^{m^2}\right)$ *a* z_i *jsou jednoznačně určená vztahem* $|x_i z_i| = n^{\lfloor \log_2 n \rfloor}$.*)*
Délku vektoru W *z* $W_n^{k,h}$ *budeme značit* $g_{n,k} = \left(6\left\lfloor \log_2 n\right\rfloor^2 + 6 + \binom{k-1}{2} + \binom{k}{2} - 3\right)$.

Lemma 3.4.1 *Nechť* $k \geqslant 2$ *a* $h \geqslant 1$ *jsou libovolná přirozená čísla. Nechť* M *je* (k,h)*-FA takový, že přijímá jazyk* $L^{k,h}$. *Pak existuje* n_0 *takové, že* $\forall n \geqslant n_0, \exists W \in W_n^{k,h}, \forall X \in X_n^{k,h}$ *při každém přijímajícím výpočtu automatu* M *pro slovo* $\varpi(W,X)$ *automat lustruje levé kopie podslov* $w_{j,i}, u_{j,i}, v_{j,i}$ *s jejich odpovídajícími pravými kopiemi pro všechna* i *a* j *z příslušných mezí.*

Toto lemma je obdobou Lemmatu 3.2.1 a opět říká, že lze zvolit taková slova $w_{j,i}, u_{j,i}, v_{j,i}$, že automat musí lustrovat všechny jejich odpovídající si levé a pravé kopie ve všech slovech, která patří do jazyka $L_n^{p,q}$ a jsou z těchto slov utvořena. Takováto $w_{j,i}, u_{j,i}, v_{j,i}$ lze nalézt od určité meze pro libovolně velká n.
Důkaz: Důkaz tohoto lemmatu je obdobný důkazu Lemmatu 3.2.1. Bude proveden sporem. Vezměme si k, h a (k,h)-FA M přijímající právě jazyk $L^{k,h}$, pro která toto lemma neplatí. Pro libovolné n_0 tedy existuje $n \geqslant n_0$ takové, že $\forall W \in W_n^{k,h}, \exists X \in X_n^{k,h}$ takové, že nějaký přijímající výpočet pro slovo $\varpi(W,X)$ neprovádí lustrování protilehlých kopií některého ze slov $w_{j,i}, u_{j,i}$, nebo $v_{j,i}$ pro jisté i a j. Zvolme si tedy *dostatečně* velké n_0 a příslušné $n \geqslant n_0$. Dále pro každé $W \in W_n^{k,h}$ vyberme jedno $X_W \in X_n^{k,h}$ a jeden přijímající výpočet V_W pro slovo $\varpi(W, X_W)$ takový, že se při něm neprovádí lustrování všech protilehlých kopií slov $w_{j,i}, u_{j,i}, v_{j,i}$. Označme si pro každé $X \in X_n^{k,h}$ množinu $H_X = \left\{\varpi(W,X) \mid W \in W_n^{k,h} \,\&\, X = X_W\right\}$. Dále si označme množinu $\mathbb{W} = \left\{\varpi(W, X_W) \mid W \in W_n^{k,h}\right\}$.

Snadno nahlédneme, že pro n dostatečně velké[15]:

$$\begin{aligned}
\left|W_n^{k,h}\right| &= |\mathbb{W}| = 2^{n g_{n,k}} \\
\left|X_n^{k,h}\right| &\leqslant \left(n^{10}\right)^{\lfloor \log_2 n\rfloor^2} . \left(n^{\lfloor \log_2 n\rfloor}\right)^h \\
&\leqslant 2^{11\lfloor \log_2 n\rfloor^3} . 2^{2h\lfloor \log_2 n\rfloor^2} \leqslant 2^{13h\lfloor \log_2 n\rfloor^3}.
\end{aligned}$$

[15]V odhadech je třeba počítat se zaokrouhlovací chybou $n \doteq 2^{\lfloor \log_2 n\rfloor}$.

Z toho vyplývá, že existuje $X_0 \in X_n^{k,h}$ takové, že $|H_{X_0}| \geqslant 2^{ng_{n,k}}/2^{13h\lfloor \log_2 n \rfloor^3}$. Pro zbytek důkazu si zafixujme jedno takové X_0. Pro každé slovo w z H_{X_0} tedy existuje přijímající výpočet V_w, při kterém se neprovádí lustrování protilehlých kopií všech podslov $w_{j,i}, u_{j,i}$ a $v_{j,i}$.

Pro $W \in W_n^{k,h}$, $W = \langle y_1, \ldots, y_{g_{n,k}} \rangle$ a $i \in \{1, \ldots, g_{n,k}\}$ si označme $(g_{n,k}-1)$-tici vzniklou vynecháním i-té složky jako $\overline{W_i} = \langle y_1, \ldots, y_{i-1}, y_{i+1} \ldots, y_{g_{n,k}} \rangle$. Dále si pak pro každé $x \in \Sigma^n$, každé $W = \langle y_1, \ldots, y_{g_{n,k}} \rangle \in W_n^{k,h}$ a každé $i \in \{1, \ldots, g_{n,k}\}$ označme jako $\overline{W_i}(x) = \langle y_1, \ldots, y_{i-1}, x, y_{i+1} \ldots, y_{g_{n,k}} \rangle$ vektor vzniklý nahrazením i-té složky W slovem x.

Pro každé $1 \leqslant i \leqslant g_{n,k}$ a každé $W \in W_n^{k,h}$ si vezměme množinu $U_{i,\overline{W_i}} = \{x \in \Sigma^n |$ při výpočtu V_w pro slovo $w = \varpi\left(\overline{W_i}(x), X_0\right)$ se neprovádí lustrování levé a pravé kopie podslova $w_{j,m}, u_{j,m}$, resp. $v_{j,m}$ odpovídajícího i-té složce vektoru $W\}$. Z definice H_X vyplývá, že každé slovo z H_{X_0} musí přispět do některé z množin $U_{i,\overline{W_i}}$. Některá slova mohou přispět do více množin $U_{i,\overline{W_i}}$ pro různá i, avšak žádná dvě různá slova z H_{X_0} nemohou přispět do téže množinu $U_{i,\overline{W_i}}$ stejným slovem x (slovo z H_{X_0} je jednoznačně určeno čtveřicí $\langle i, \overline{W_i}, x, X_0 \rangle$). Tedy součet velikostí všech množin $U_{i,\overline{W_i}}$ musí být neostře větší než velikost množiny H_{X_0}. Ukážeme však, že tomu tak není, čímž získáme požadovaný spor.

Jelikož každá množina $U_{i,\overline{W_i}}$ splňuje předpoklady Důsledku 3.1.2 (funkcí f je identita, slovy a_0, a_1, a_2 jsou po řadě počáteční, prostřední a koncový úsek slova $w = \varpi\left(\overline{W_i}(x), X\right)$ oddělené levou resp. pravou kopií nelustrovaného podslova $w_{j,m}, u_{j,m}$, resp. $v_{j,m}$), platí pro ni horní odhad:

$$\left|U_{i,\overline{W_i}}\right| < \widetilde{p}_{k,h,s}(d_{n,k,h}),$$

kde $\widetilde{p}_{k,h,s}$ je polynom z Důsledku 3.1.2.

Množin $U_{i,\overline{W_i}}$ je pro různé hodnoty i a různé hodnoty $\overline{W_i}$ celkem $g_{n,k}.2^{n\left(g_{n,k}-1\right)}$. Pro součet jejich velikostí tedy pro n dostatečně velké dostáváme:

$$\begin{aligned} \sum_{i,\overline{W_i}} \left|U_{i,\overline{W_i}}\right| &\leqslant \widetilde{p}_{k,h,s}(d_{n,k,h}).g_{n,k}.2^{n\left(g_{n,k}-1\right)} \\ &\leqslant c_1 2^{c_2 \lfloor \log_2 n \rfloor^2}.\lfloor \log_2 n \rfloor^2.2^{n\left(g_{n,k}-1\right)} \\ &< c_1 2^{ng_{n,k}-n+c_3 \lfloor \log_2 n \rfloor^2} = g(n), \end{aligned}$$

kde c_1, c_2, c_3 jsou konstanty nezávislé na volbě n.

Velikost H_{X_0} je větší než $2^{ng_{n,k}-13h\lfloor \log_2 n\rfloor^3} = f(n)$.

Porovnáním exponentů ve funkcích $g(n)$ a $f(n)$ snadno nahlédneme, že $\lim_{m\to\infty} g(m)/f(m) = 0$, a tedy vzhledem k volbě n jako dostatečně velkého je $g(n)$ menší než $f(n)$, neboli $\sum \left|U_{i,\overline{W_i}}\right| < |H_{X_0}|$. To je však spor s výše uvedenými vlastnostmi těchto veličin.

Tím je lemma dokázáno. $\square$

Nyní zformulujeme jedno pomocné lemma, jehož důkaz ponecháme na později. Toto lemma říká, že pokud si vezmeme slovo $a_0xa_1ya_2z_10^ez_2a_3$ a nad ním výpočet V (k,h)-FA automatu bez koincidence, během kterého tři hlavy projdou ze slova z_1 do slova z_2 a ostatní hlavy přitom budou v podslovech x a y, přičemž všechny kameny budou položeny, a to v podslovech a_0, a_1, a a_2, pak, pokud je slovo 0^e dostatečně dlouhé ve srovnání se slovy x, y, z_1 a z_2, tak existuje i výpočet nad slovem $a'_0xa'_1ya'_2z_10^{e+c}z_2a'_3$, v jehož výchozí konfiguraci bude relativní rozmístění hlav, včetně stavu, vzhledem k x, y a z_1 stejné jako ve výchozí konfiguraci výpočtu V a stejně tak relativní rozmístění hlav, včetně stavu, v poslední konfiguraci vzhledem k x, y a z_2 bude stejné jako v poslední konfiguraci V. Přitom během tohoto výpočtu budou všechny kameny položeny, a to ve slovech a'_0, a'_1 a a'_2.

Uvedené tvrzení bude platit pro c konstantu $(6s^3)!$ a $-(6s^3)!$, kde s je počet stavů zkoumaného automatu.

Lemma 3.4.2 *Nechť $d \geqslant 1$, $k \geqslant 4$ a h jsou libovolná přirozená čísla. Nechť M je (k,h)-FA automat bez koincidence s počtem stavů s. Nechť $h_1, h_2, \ldots, h_k$ jsou hlavy automatu M. Pak existuje n_0 takové, že pro $\forall n \geqslant n_0$ a pro $\forall a_0, a_1, a_2, a_3, x, y, z_1, z_2, a'_0, a'_1, a'_2, a'_3 \in \Sigma^*$ taková, že $|x|, |y|, |z_1|, |z_2| \leqslant dn$, platí:*

Pokud c^0, c'^0 jsou konfigurace nad slovem $a_0xa_1ya_2z_10^ez_2a_3$, kde $e \geqslant n^5$, takové, že hlavy h_1, h_2, h_3 jsou v konfiguraci c^0 v podslově z_1 a v konfiguraci c'^0 v podslově z_2, hlava h_k je v obou konfiguracích v podslově x a zbylé hlavy jsou v obou konfiguracích v podslově y a všechny kameny jsou položeny, a to v podslovech a_0, a_1, a_2, a pokud existuje výpočet automatu M z konfigurace c^0 do konfigurace c'^0,

pak též existují výpočty automatu M nad slovy $a'_0xa'_1ya'_2z_10^{e+(6s^3)!}z_2a'_3$ a $a'_0xa'_1ya'_2z_10^{e-(6s^3)!}z_2a'_3$ takové, že relativní rozmístění hlav, včetně stavu, vůči podslovům x, y, z_1 bude v první konfiguraci těchto výpočtů stejné jako v konfigurace c^0 a relativní rozmístění hlav, včetně stavu, vůči podslovům x, y, z_2 bude v poslední konfiguraci těchto výpočtů stejné jako v konfigurace c'^0, přičemž *všechny kameny budou během těchto výpočtů položeny v podslovech a'_0, a'_1, a'_2.*

Snadno nahlédneme, že uvedené tvrzení lze iterovat, a tedy že jeho závěr platí i pro libovolné slovo $a'_0 x a'_1 y a'_2 z_1 0^{e+i(6s^3)!} z_2 a'_3$, kde i je přirozené číslo.[16] Jedná se tedy o jistý druh Pumping lemmatu.

Důkaz tohoto lemmatu je technicky náročný a proto jej uvedeme až za Větou 3.4.3 a jejím důkazem. Myšlenka tohoto důkazu je však relativně prostá. Vychází z pozorování, že pokud se nějaké hlavy automatu pohybují příliš dlouho nad stejným symbolem, aniž by se nějaké další hlavy pohnuly, pak se nutně musí opakovat posloupnost stavů, kterými automat prošel. Jelikož automat vidí stále stejné symboly pod všemi hlavami, lze takovýto cyklus konfigurací vypustit, čímž získáme výpočet pro kratší slovo, a nebo naopak přidat, čímž získáme výpočet pro delší slovo.

Toto lemma neplatí pro automaty s koincidencí, neboť takové automaty činí svá rozhodnutí nejenom na základě symbolu, který vidí na pásce (a který by mohl být i stále stejný), ale také na základě toho, zda na stejném políčku vidí či nevidí další hlavy. Tento údaj se může v průběhu i dlouhého výpočtu libovolně měnit (viz též Věta 3.1.3 o Turingových strojích.)

Věta 3.4.3 *Nechť $k \geqslant 5, h \geqslant 1$ jsou přirozená čísla. Pak žádný (k,h)-FA automat bez koincidence nemůže rozpoznávat jazyk $L^{k,h}$.*

Důkaz: Důkaz provedeme sporem. Vezmeme si pro nějaká čísla $k \geqslant 5$ a $h \geqslant 1$ (k,h)-FA automat M bez koincidence, který rozpoznává jazyk $L^{k,h}$ a ukážeme, že to vede ke sporu. Důkaz se bude skládat z několika částí. V první části důkazu prostudujeme celkové chování automatu M při přijímacích výpočtech. V další části důkazu pak ukážeme, že v jazyce $L^{k,h}$ musí existovat slova, při jejichž přijímajících výpočtech projde automat stejnou význačnou konfigurací. V poslední části pak z těchto slov sestrojíme slovo, které nebude patřit do jazyka $L^{k,h}$, avšak automat jej přijme.

Nechť s je počet stavů automatu M. Zvolme si *dostatečně* velké n, mocninu dvojky,[17] takové, aby prošlo zbytkem důkazu. Označme $l = \lfloor \log_2 n \rfloor^2$.

[16] Platí do jisté míry i pro i záporná, tedy pokud $e + i\,(6s^3)! \geqslant n^5$.

[17] Ve výpočtech se využívá vztahu $n = 2^{\lfloor \log_2 n \rfloor}$, který pro obecné n platí až na zaokrouhlovací chybu, pro mocniny dvou platí přesně.

Z Lemmatu 3.4.1 víme, že pro dostatečně velká n lze nalézt taková slova $w_{i,j}, u_{i,j}, v_{i,j}$, že se pro všechna $X \in X_n^{k,h}$ při přijímajících výpočtech slov $\varpi\left(\langle \ldots, w_{i,j}, \ldots, u_{i,j}, \ldots, v_{i,j}, \ldots \rangle, X\right)$ provádí lustrování protilehlých kopií všech slov $w_{i,j}, u_{i,j}, v_{i,j}$ pro příslušná i a j. Pro zbytek důkazu si tedy zafixujme jednu takovou volbu slov $w_{i,j}, u_{i,j}, v_{i,j}$, označme si ji jako $W_0 = \langle \ldots, w_{i,j}, \ldots, u_{i,j}, \ldots, v_{i,j}, \ldots \rangle$, a definujme dále množinu $L = \left\{\varpi\left(W_0, X\right) \mid X \in X_n^{k,h}\right\}$. Pro každé slovo $w \in L$ si zafixujme jeden přijímající výpočet V_w.

Dále se již budeme zabývat pouze slovy z množiny L a nechť má proto laskavý čtenář na paměti, že se v přijímajících výpočtech provádí lustrování protilehlých kopií všech slov $w_{i,j}, u_{i,j}, v_{i,j}$ pro příslušná i a j.

I. V této části důkazu prostudujeme průběh přijímajícího výpočtu pro slova z L. Ukážeme, že vždy, když poslední hlava opouští levou kopii slova $u_{i,3}$, je $k-4$ hlav v oblasti α_2 a zbylé tři hlavy jsou v oblasti $\overline{\beta}_i$.

Vezměme si některé slovo z množiny L. Vzhledem k rozložení slov v oblasti α_2 a k rozložení kopií slov $u_{0,1}, u_{0,2}, u_{0,3}$ snadno nahlédneme, že každou dvojici protilehlých kopií slov $w_{2,j}$ a $u_{0,j}$ musí lustrovat jiná dvojice hlav (standardní argument). Celkem je $\binom{k}{2}$ dvojic protilehlých kopií slov $w_{2,j}$ a $u_{0,j}$, tedy každá hlava musí vystupovat v $k-1$ lustrováních těchto slov, a sice každá hlava s každou musí lustrovat některou z dvojic kopií slov $w_{2,j}$ a $u_{0,j}$.

Nejprve se podívejme na rozložení hlav v okamžiku, kdy poslední (nejlevější) hlava, říkejme ji h_k, opouští levou kopii slova $u_{0,1}$. Jelikož již muselo proběhnout lustrování kopií slova $u_{0,1}$, musela již hlava, označme ji h_1, která toto lustrování prováděla v pravé kopii slova $u_{0,1}$, provést všech svých $k-1$ lustrování kopií některých ze slov $w_{2,j}$ a $u_{0,j}$. Tedy nejméně $k-1-3$ hlav muselo již vstoupit do oblasti α_2 (3 lustrování mohla hlava h_1 provést na kopiích slov $u_{0,j}$.) Hlava h_k musí provést taktéž $k-1$ lustrování, z čehož tři mohou být lustrování slov $u_{0,1}, u_{0,2}, u_{0,3}$ a další musí být kopie některých ze slov $w_{2,j}$ v oblasti α_2. Tedy zároveň ještě alespoň $k-4$ hlav musí být v oblasti α_2, nebo vlevo od ní.

Podívejme se nyní na oblast α_1. V té je potřeba provést $\binom{k-1}{2}$ lustrování a opět každá dvojice hlav může provést pouze jedno. Pokud by tři hlavy byly stále ještě vlevo od oblasti α_1, pak by zatím proběhlo maximálně $\binom{k-3}{2}$ těchto lustrování, tedy zbývalo by provést $\binom{k-1}{2} - \binom{k-3}{2} = 2k-5$ lustrování. Zároveň však víme, že $k-3$ hlav je již za oblastí α_1, tedy zbylé tři hlavy by musely provést všech $2k-5$ lustrování. Ty ale mohou provést maximálně 3 lustrování, tedy

$2k-5 \leqslant 3$, což je možné jenom pokud $k \leqslant 4$. Avšak $k \geqslant 5$, tedy možnost, že by vlevo od oblasti α_1 byly tři hlavy, je vyloučená.

Podívejme se na možnost, že jsou vlevo od oblasti α_1 hlavy dvě. Pak mohlo již proběhnout $\binom{k-2}{2}$ lustrování v α_1, tedy zbývá pouze $\binom{k-1}{2} - \binom{k-2}{2} = k-2$ lustrování. Dvě hlavy však mohou spolu vykonat pouze jedno lustrování, tedy $k-2 \leqslant 1$, a z toho získáme, že $k \leqslant 3$. To je ovšem opět ve sporu s tím, že $k \geqslant 5$.

Vlevo od oblasti α_1 tedy může být pouze jediná hlava a tou je hlava h_k.

Nyní se podívejme, která hlava mohla zajistit lustrování levé kopie slova $u_{0,2}$. Pokud by to byla jiná hlava než h_k, pak by to znamenalo, že v jistou dobu byly dvě hlavy vlevo od oblasti α_1. Zároveň však již některá hlava musela být nad pravou kopií slova $u_{0,2}$ a tedy alespoň dalších $k-1-3$ hlav již vstoupilo do oblasti α_2. Z úvah o α_1 však vyplývá, že toto není možné. Tedy lustrování levé kopie $u_{0,2}$ musí provést také hlava h_k. Stejnou úvahou zjistíme, že levou kopii $u_{0,3}$ musí taktéž lustrovat hlava h_k. A tuto úvahu lze pak dále použít pro všechna další $u_{i,j}$ a $v_{i,j}$. Levé kopie všech slov $u_{i,j}$ a $v_{i,j}$ musí tedy lustrovat tatáž hlava h_k.

Vraťme se zpět ke slovům $u_{0,1}, u_{0,2}, u_{0,3}$. Hlava h_k zajišťuje lustrování levých kopií všech slov $u_{0,1}, u_{0,2}, u_{0,3}$. Po celou dobu těchto lustrování musí na tuto hlavu čekat v oblasti α_2 alespoň $k-4$ hlavy. Vzhledem k rozložení pravých kopií slov $u_{0,1}, u_{0,2}, u_{0,3}$ musí každou pravou kopii těchto slov lustrovat jiná hlava. Tedy tři hlavy, jiné než čekajících $k-4$ hlav v oblasti α_2, musí vstoupit do pravých kopií slov $u_{0,1}, u_{0,2}, u_{0,3}$.

V okamžiku, kdy hlava h_k opouští levou kopii slova $u_{0,3}$, čeká na ni v oblasti α_2 $k-4$ hlav. Jelikož proběhlo lustrování kopií slov $u_{0,1}, u_{0,2}, u_{0,3}$, musely již tři další hlavy vstoupit do oblasti $\overline{\beta}_0$. Hlava h_k dále musí provést lustrování levých kopií slov $v_{0,1}, v_{0,2}, v_{0,3}$ a potřebuje k tomu další tři hlavy. Hlavy v oblasti α_2 to být nemohou, tedy hlavy, které s ní lustrovaly kopie slov $u_{0,1}, u_{0,2}, u_{0,3}$, s ní musí v průběhu dalšího výpočtu lustrovat i kopie slov $v_{0,1}, v_{0,2}, v_{0,3}$. Tedy ještě nemohly opustit oblast $\overline{\beta}_0$.

V průběhu následujícího výpočtu musí $k-4$ hlav setrvat v oblasti α_2 dokud tam nedorazí poslední hlava h_k. Než tam tato hlava dorazí, musí provést ve spolupráci se zbylými třemi hlavami lustrování všech $u_{i,j}, v_{i,j}$. Nyní již snadno nahlédneme, že vždy, když hlava h_k opustí levou kopii slova $u_{i,3}$, budou zbylé tři hlavy v oblasti $\overline{\beta}_i$, neboť právě dokončily lustrování pravých kopií slov $u_{i,1}, u_{i,2}, u_{i,3}$ a musí provést lustrování pravých kopií slov $v_{i,1}, v_{i,2}, v_{i,3}$.

První konfiguraci přijímajícího výpočtu pro slovo z L, ve které jsou všechny hlavy za levou kopii slova $u_{0,3}$, budeme označovat jako *vstupní konfiguraci* a první konfiguraci, ve které jsou všechny hlavy za levou kopií slova $u_{l,3}$ budeme označovat jako *konfiguraci výstupní.* Mezi těmito dvěma konfiguracemi probíhá výpočet, při kterém čtyři hlavy lustrují slova z oblastí β_i a při kterém $k-4$ hlav setrvává v oblasti α_2. V dalším textu důkazu budeme podrobněji zkoumat relativní rozložení hlav a kamenů, včetně stavu, ve vstupní konfiguraci vzhledem k podslovu $(\beta_i a_i\#)_0^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1\alpha_2\overline{\beta}_0$ a relativní rozložení hlav ve výstupní konfiguraci vzhledem k podslovům β_l, α_2 a $\overline{\beta}_l$.

II. Vezměme si nyní množinu $\mathbb{X} = \left\{X \mid X \in X_n^{k,h} \cap \left((\{0\}^*)^h \times \left(\left\{0^{n^8}, 0^{n^8+(6s^3)!}\right\}\right)^l\right)\right\}$ a jazyk $L' = \{\varpi(W_0, X) \mid X \in \mathbb{X}\}$.[18] V této části důkazu budeme zkoumat vlastnosti vstupní a výstupní konfigurace přijímacích výpočtů slov z L'.

Při vstupní konfiguraci jsou všechny hlavy rozmístěny v oblastech β_0, α_2 a $\overline{\beta}_0$, přičemž právě jedna z nich je v oblasti β_0, tři jsou v oblasti $\overline{\beta}_0$ a ostatní jsou v oblasti α_2. Délka každé z těchto oblastí je jistě nejvýše $2\binom{k}{2}(n+1)$ a tedy počet různých rozmístění hlav ve vstupní konfiguraci, pro danou volbu toho, která z hlav je v oblasti β_0 a které jsou v oblasti $\overline{\beta}_0$, je nejvýše $\left(2\binom{k}{2}(n+1)\right)^k$. Možností výběru, která z hlav bude v oblasti β_0 a které z hlav budou v oblasti $\overline{\beta}_0$, je $k\binom{k-1}{3}$, což je méně než k^k. Možných různých relativních rozmístění hlav vzhledem k podslovům β_0, α_2 a $\overline{\beta}_0$ je tedy ve vstupní konfiguraci nejvýše $k^k\left(2\binom{k}{2}(n+1)\right)^k = \left(2k\binom{k}{2}(n+1)\right)^k$. Jelikož délka oblasti $(\beta_i a_i\#)_1^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1$ je pro všechna slova z L' stejná, je ve vstupní konfiguraci možných relativních rozmístění hlav vzhledem k podslovu $(\beta_i a_i\#)_0^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1\alpha_2\overline{\beta}_0$ nejvýše $\left(2k\binom{k}{2}(n+1)\right)^k$.

Podobně pro výstupní konfiguraci, kdy jsou všechny hlavy rozmístěny v podslovech β_l, α_2 a $\overline{\beta}_l$, je možných relativních rozmístění hlav vůči podslovům β_l, α_2 a $\overline{\beta}_l$ nejvýše $\left(2k\binom{k}{2}(n+1)\right)^k$.

A. Nyní zkoumejme slova, při jejichž vstupní konfiguraci nejsou položeny všechny kameny nebo některý z kamenů leží v oblasti $(\beta_i a_i\#)_0^{l-1}\,\beta_l$, α_2, nebo $\overline{\beta}_0$.[19] Ukážeme, že slov s takovouto vstupní konfigurací je málo.

Nechť jsou dána libovolná $\overline{a}_0, \dots, \overline{a}_{l-1} \in \left\{0^{n^8}, 0^{n^8+(6s^3)!}\right\}$. Ta jednoznačně určují slova

[18] Všimněte si, že jelikož n je dostatečně velké, tedy mimo jiné větší než $(6s^3)!$, je n^8 mocninou přirozeného čísla, ale $n^8 + (6s^3)!$ není. Přitom 0^{n^8} i $0^{n^8+(6s^3)!}$ patří do množiny $\{0\}^{<n^{10}}$.

[19] Vpravo od oblasti $\overline{\beta}_0$ jistě žádný kámen neleží, neboť všechny hlavy jsou ve vstupní konfiguraci nejdále v této oblasti.

$a_0, \ldots, a_{l-1}$. Ve vstupní konfiguraci je počet možných rozmístění kamenů relativně k podslovu $(\beta_i a_i \#)_0^{l-1} \beta_l \not{c} (x_j z_j \#)_1^h \alpha_1 \alpha_2 \overline{\beta}_0$ takových, že některý z kamenů není položen nebo je položen v oblasti $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2 či $\overline{\beta}_0$, nejvýše:

$$(d_{n,k,h}+1)^{h-1}\left((6n+8)\lfloor\log_2 n\rfloor + 2\left(\binom{k}{2}+3\right)(n+1)+1\right),$$

neboť kámen, který leží v oblasti $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2 nebo $\overline{\beta}_0$, může být na:

$$\left((6n+8)\lfloor\log_2 n\rfloor + 2\left(\binom{k}{2}+3\right)(n+1)\right)$$

pozicích, případně nemusí ležet vůbec. Ostatní kameny mohou ležet na $d_{n,k,h}$ pozicích, nebo také nemusí ležet vůbec.

Celkem je tedy ve vstupní konfiguraci možných nejvýše:

$$\underbrace{s}_{\text{počet stavů}} \cdot \underbrace{\left(2k\binom{k}{2}(n+1)\right)^k}_{\text{možné pozice hlav}} \cdot \underbrace{(d_{n,k,h}+1)^{h-1}\left((6n+8)\lfloor\log_2 n\rfloor + 2\left(\binom{k}{2}+3\right)(n+1)+1\right)}_{\text{případné pozice kamenů}}$$

relativních rozložení hlav a kamenů, včetně stavu, vzhledem k $(\beta_i a_i \#)_0^{l-1} \beta_l \not{c} (x_j z_j \#)_1^h \alpha_1 \alpha_2 \overline{\beta}_0$ takových, že při nich alespoň jeden kámen není položen, nebo leží v oblasti $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2 či $\overline{\beta}_0$.

Jelikož n bylo zvoleno dostatečně velké, lze tento výraz shora odhadnout výrazem:

$$c_4 n^k 2^{\lfloor\log_2 n\rfloor^2 (h-1)} n \lfloor\log_2 n\rfloor,$$

kde c_4 je konstanta nezávislá na volbě n (Podle Poznámky pod čarou 13 na straně 52 je pro jistou konstantu c od jistého n_0 $d_{n,k,h} \leqslant cn^{\lfloor\log_2 n\rfloor}$).

Pokud by některá dvě různá slova z L' pro danou volbu $\overline{a}_0, \ldots, \overline{a}_{l-1}$ lišící se pouze v podslovech x_j, z_j měla stejné rozložení hlav a kamenů, včetně stavu, ve vstupní konfiguraci relativně k podslovu $(\beta_i a_i \#)_0^{l-1} \beta_l (x_j z_j \#)_1^h \alpha_1 \alpha_2 \overline{\beta}_0$, pak bychom nahrazením oblasti $(z_j \#)_1^h$ v jednom ze slov touto oblastí ze slova druhého získali slovo, jež nepatří do jazyka $L^{k,h}$, avšak jež je, jak snadno nahlédneme, přijímáno automatem M. Automat M by se totiž při přijímacím výpočtu choval až po vstupní konfiguraci jako při výpočtu pro první z těchto slov a poté by se choval jako při výpočtu pro druhé z těchto slov.[20] Rozmístění hlav a přístupných kamenů, včetně stavu, vzhledem k podslovu $(\beta_i a_i \#)_0^{l-1} \beta_l ¢ (x_j z_j \#)_1^h \alpha_1 \alpha_2 \overline{\beta}_0$ je v obou výpočtech při vstupní konfiguraci stejné, tedy výpočty by na sebe takto korektně navazovaly. To by vedlo ke sporu s tím, že automat M rozpoznává jazyk $L^{k,h}$. Každá dvě slova z L' pro danou volbu $\overline{a}_0, \ldots, \overline{a}_{l-1}$ musí tedy mít při vstupní konfiguraci různé rozmístění hlav a kamenů, včetně stavu, relativně k podslovu $(\beta_i a_i \#)_0^{l-1} \beta_l ¢ (x_j z_j \#)_1^h \alpha_1 \alpha_2 \overline{\beta}_0$.

Počet různých h-tic slov x_j je $n^{\lfloor \log_2 n \rfloor h} = 2^{h \lfloor \log_2 n \rfloor^2}$, tedy počet slov v L' s danou volbou slov $\overline{a}_0, \ldots, \overline{a}_{l-1}$ je právě $2^{h \lfloor \log_2 n \rfloor^2}$. Pro jednu volbu $\overline{a}_0, \ldots, \overline{a}_{l-1}$ musí tedy alespoň

$$2^{h \lfloor \log_2 n \rfloor^2} - c_4 n^k 2^{\lfloor \log_2 n \rfloor^2 (h-1)} n \lfloor \log_2 n \rfloor$$

slov z L' mít vstupní konfiguraci, při které jsou položeny všechny kameny a zároveň žadný z kamenů není položen v oblasti $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2, ani $\overline{\beta}_0$.[21] Jelikož n je dostatečně velké, je:

$$c_4 n^k 2^{\lfloor \log_2 n \rfloor^2 (h-1)} n \lfloor \log_2 n \rfloor < 2^{h \lfloor \log_2 n \rfloor^2 - 1}$$

a tedy pro uvedený počet slov platí dolní odhad:

$$2^{h \lfloor \log_2 n \rfloor^2} - c_4 n^k 2^{\lfloor \log_2 n \rfloor^2 (h-1)} n \lfloor \log_2 n \rfloor > 2^{h \lfloor \log_2 n \rfloor^2 - 1} \tag{3.4}$$

Jelikož tento odhad platí pro každou volbu $\overline{a}_0, \ldots, \overline{a}_{l-1}$, má množina $\mathbb{X}' = \{X \in \mathbb{X} |$ při přijímacím výpočtu pro slovo $\varpi(W_0, X)$ jsou ve vstupní konfiguraci všechny kameny položeny a jsou mimo oblasti $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2 a $\overline{\beta}_0$ $\}$ velikost větší než $2^{h \lfloor \log_2 n \rfloor^2 - 1} 2^{\lfloor \log_2 n \rfloor^2} = 2^{(h+1) \lfloor \log_2 n \rfloor^2 - 1}$.
B. Nyní se tedy zabývejme slovy, která mají při přijímajícím výpočtu při vstupní konfiguraci všechny kameny položeny a sice položeny mimo oblast $(\beta_i a_i \#)_0^{l-1} \beta_l$, α_2 a $\overline{\beta}_0$. Tato slova tvoří množinu $L'' = \{\varpi(W_0, X) \mid X \in \mathbb{X}'\}$ velikosti větší než $2^{(h+1) \lfloor \log_2 n \rfloor^2 - 1}$.

Zkoumejme relativní rozmístění hlav a kamenů při vstupní a výstupní konfiguraci pro slova z L''. Ve vstupní konfiguraci budeme studovat rozmístění kamenů a hlav, včetně stavu, relativně

[20] Všimněte si, že oblast $(x_j z_j \#)_1^h$ je ve všech slovech jazyka $L_n^{k,h}$ identická.

[21] Pokud by s takovou konfigurací bylo těchto slov méně, pak by některá dvě musela mít stejnou vstupní konfiguraci což by vedlo k výše uvedenému sporu.

k $(\beta_i a_i\#)_0^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1\alpha_2\overline{\beta}_0$, ve výstupní konfiguraci relativně k β_l, α_2 a $\overline{\beta}_l$. Kameny ležící v oblasti $(x_j\#)_1^h$ při vstupní konfiguraci jsou nedosažitelné v průběhu celého zbytku výpočtu a jejich poloha je tudíž nezajímavá, zajímavý může být pouze údaj o jejich počtu (ten je implicitně obsažen v následujícím odhadu na počet různých rozložení hlav a kamenů).

Ve vstupní konfiguraci je počet možných rozmístění hlav a kamenů, včetně stavu, relativně k $(\beta_i a_i\#)_0^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1\alpha_2\overline{\beta}_0$ menší než $s\left(2k\binom{k}{2}(n+1)\right)^k (d_{n,k,h})^h$.

Počet možných rozmístění hlav, včetně stavu, relativně k podslovům β_l, α_2 a $\overline{\beta}_l$ při výstupní konfiguraci je menší než $s\left(2k\binom{k}{2}(n+1)\right)^k$. V průběhu výpočtu mezi vstupní a výstupní konfigurací nemohlo dojít ke změně polohy kamenů, neboť všechny kameny jsou mimo dosah hlav.

Celkem je tedy méně než $s^2\left(2k\binom{k}{2}(n+1)\right)^{2k}(d_{n,k,h})^h \leqslant c_5 n^{2k} 2^{h\lfloor \log_2 n\rfloor^2}$ možných kombinací relativních rozmístění hlav a kamenů, včetně stavu, při vstupní a výstupní konfiguraci přijímajících výpočtů (pro n dostatečně velké a konstantu c_5 nezávislou na jeho volbě).

Jelikož množina L'' obsahuje více než $2^{(h+1)\lfloor \log_2 n\rfloor^2-1}$ slov, mají některá dvě různá slova z L'' stejné relativní rozložení kamenů a hlav při vstupní a výstupní konfiguraci jejich přijímacích výpočtů. Nechť to jsou slova $w_1 = \varpi(W_0, X_1)$ a $w_2 = \varpi(W_0, X_2)$, kde $X_1, X_2 \in \mathbb{X}'$. Rozeberme dvě možnosti, kde se slova w_1 a w_2 mohou lišit.

a. Slovo w_1 se od slova w_2 neliší v podslovech $\overline{a}_0, \ldots, \overline{a}_{l-1}$, a tedy ani v $a_0, \ldots, a_{l-1}$. Musí se tedy lišit v podslovech x_j, z_j a všude jinde být stejná. Jelikož relativní rozmístění hlav a kamenů, včetně stavu, je ve vstupní konfiguraci pro obě slova stejné a v této konfiguraci již všechny hlavy opustily oblast $(x_j\#)_1^h$ a žádná přitom nevstoupila do oblasti $(z_j\#)_1^h$, snadno nahlédneme, že slovo vytvořené nahrazením oblasti $(z_j\#)_1^h$ ve slově w_1 touto oblastí ze slova w_2 bude přijímáno automatem M. Takové slovo však zřejmě nepatří do jazyka $L^{k,h}$, tedy bychom tím dostali spor a proto se slova w_1 a w_2 musí lišit i v podslovech $\overline{a}_0, \ldots, \overline{a}_{l-1}$.

b. Slovo w_1 se od slova w_2 liší v podslovech $\overline{a}_0, \ldots, \overline{a}_{l-1}$ (a tedy i v $a_0, \ldots, a_{l-1}$) a případně se liší i v podslovech x_j, z_j.

Nechť:

$$
\begin{aligned}
w_1 &= (x_j\#)_1^h\,(\beta_i a_i\#)_0^{l-1}\,\beta_l ¢\,(x_j z_j\#)_1^h\,\alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}_i\#\right)_0^{l-1}\overline{\beta}_l\,(z_j\#)_1^h \\
w_2 &= \left(x'_j\#\right)_1^h\left(\beta_i a'_i\#\right)_0^{l-1}\beta_l ¢\left(x'_j z'_j\#\right)_1^h\alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}'_i\#\right)_0^{l-1}\overline{\beta}_l\left(z'_j\#\right)_1^h.
\end{aligned}
$$

Vezměme si slovo w_3, které získáme nahrazením oblasti $\left(\overline{\beta}_i\overline{a}_i\#\right)_0^{l-1}\overline{\beta}_l¢\left(z_j\#\right)_1^h$ ve slově w_1 touto oblastí ze slova w_2:

$$w_3 = \left(x_j\#\right)_1^h \left(\beta_i a_i\#\right)_0^{l-1}\beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}'_i\#\right)_0^{l-1}\overline{\beta}_l\left(z'_j\#\right)_1^h.$$

Takovéto slovo nepatří do jazyka $L^{k,h}$ ($a_i = 1 \Longleftrightarrow \overline{a}'_i = 0^{m^2}$, což není splněno pro všechna i, neboť a_i jsou ze slova w_1, $\overline{a}'_i$ jsou ze slova w_2 a pro nějaké i je $\overline{a}_i \neq \overline{a}'_i$.) Ukážeme však, že automat M slovo w_3 přijímá. Tím získáme požadovaný spor a věta bude dokázána.

Automat M přijímá slovo w_3 následujícím výpočtem. Až do vstupní konfigurace se bude chovat jako pro slovo w_1, poté až do výstupní konfigurace použije modifikovaný kus výpočtu pro slovo w_1 a závěr výpočtu dokončí podle výpočtu pro slovo w_2.

$$\begin{array}{lllll}
\text{První část výpočtu } (=w_1): & \left(x_j\#\right)_1^h \left(\beta_i a_i\#\right)_0^{l-1}\beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\overline{\beta}_0 & \cdots \\
\text{Druhá část výpočtu } (\approx w_1): & \cdots\quad \left(\beta_i a_i\#\right)_0^{l-1}\beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}'_i\#\right)_0^{l-1}\overline{\beta}_l & \cdots \\
\text{Třetí část výpočtu } (=w_2): & \cdots\quad \beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}'_i\#\right)_0^{l-1}\overline{\beta}_l\left(z'_j\#\right)_1^h
\end{array}$$

Snadno nahlédneme, že první úsek takovéhoto výpočtu je korektní, neboť hlavy se pohybují v oblasti $\left(x_j\#\right)_1^h \left(\beta_i a_i\#\right)_0^{l-1}\beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\overline{\beta}_0$, která je identická s touto oblastí ve slově w_1. Pokud prostřední část výpočtu nepohne s kameny a skončí se stejným relativním rozložením hlav, včetně stavu, v podslovech β_l, α_2, $\overline{\beta}_l$, jako je relativní rozložení hlav ve výstupní konfiguraci pro slovo w_2, pak jelikož část $\beta_l¢\left(x_j z_j\#\right)_1^h \alpha_1\alpha_2\left(\overline{\beta}_i\overline{a}'_i\#\right)_0^{l-1}\overline{\beta}_l\left(z'_j\#\right)_1^h$ slova w_3 se shoduje s koncem slova w_2, bude závěr výpočtu taktéž korektní. Zbývá nám ukázat, jak zmodifikujeme prostřední část výpočtu pro slovo w_1.

Část výpočtu V_{w_1} nad slovem w_1 mezi vstupní a výstupní konfigurací rozdělme na l souvislých úseků, *bloků*, podle polohy nejlevější hlavy h_k. První konfigurací i-tého bloku ($i \in \{1,\ldots,l\}$) bude konfigurace, ve které hlava h_k vstoupila do levé kopie slova $v_{i-1,0}$. Prostřední část výpočtu nad slovem w_3 získáme postupnou modifikací jednotlivých bloků výpočtu V_{w_1}.

Z rozboru v první části důkazu víme, že v první konfiguraci i-tého bloku výpočtu se jedna z hlav nachází nad podslovem $u_{i-1,3}$, další tři hlavy jsou v oblasti $\overline{\beta}_{i-1}$ a zbylé hlavy jsou v oblasti α_2. V průběhu výpočtu i-tého bloku pak hlava z podslova $u_{i-1,3}$ projde podslovem $v_{i-1,0}\#v_{i-1,1}\#v_{i-1,2}\#a_{i-1}\#u_{i,0}\#u_{i,1}\#u_{i,2}\#$, hlavy z oblasti $\overline{\beta}_{i-1}$ projdou podslovem

$\overline{\beta}_{i-1}\overline{a}_i\#\overline{\beta}_i$ z oblasti $\overline{\beta}_{i-1}$ do oblasti $\overline{\beta}_i$ a ostatní hlavy se pohybují pouze v rámci podslova α_2.

i-tý blok původního výpočtu nad slovem w_1 (pro $i = 1, \ldots, l$) nahradíme blokem výpočtu nad slovem w_3, který bude mít v první konfiguraci stejné rozložení hlav, včetně stavu, relativně k β_{i-1}, α_2 a $\overline{\beta}_{i-1}$ jako měl i-tý blok výpočtu V_{w_1} a také v konfiguraci následující po poslední konfiguraci tohoto bloku (tj. v první konfigurace následujícího bloku) bude mít stejné rozložení hlav, včetně stavu, relativně k β_i, α_2 a $\overline{\beta}_i$ jako původní blok. Pokud toto provedeme pro všechny bloky, skončí takovýto výpočet nad w_3 s relativním rozložením hlav, včetně stavu, jaké má výstupní konfigurace výpočtu pro w_2. Takovýto výpočet nad slovem w_3 jsme přesně požadovali.

Vytvořme tedy modifikovaný výpočet pro i-tý blok.

Pokud $\overline{a}_{i-1} = \overline{a}'_{i-1}$, pak můžeme použít původní výpočet tohoto bloku pro slovo w_1, neboť oblasti slova w_3, kterými se hlavy pohybují, jsou v obou případech totožné.

Pokud $\overline{a}_{i-1} \neq \overline{a}'_{i-1}$, pak jedno z nich má délku druhé mocniny a druhé ne ($\overline{a}_{i-1}, \overline{a}'_{i-1} \in \left\{0^{n^8}, 0^{n^8+(6s^3)!}\right\}$). Nechť podslovo $\overline{a}_{i-1}$ ve slově w_1 je 0^d a ve slově w_2 nechť je $0^{d'}$. Zřejmě $|d - d'| = \left(6s^3\right)!$ a buď d, nebo d' je druhou mocninou přirozeného čísla.[22] Dále také $d > n^5$. V průběhu i-tého bloku výpočtu nad slovem w_1 se každá z hlav v oblasti α_2 posune nejvýše o $\left(2\binom{k}{2} - 6\right)(n+1)$ pozic, hlava z oblasti β_{i-1} se posune nejvýše o $6n + 8$ pozic a zbylé tři hlavy projdou slovem $\overline{\beta}_{i-1}0^d\overline{\beta}_i$ z oblasti $\overline{\beta}_{i-1}$ do oblasti $\overline{\beta}_i$. Takovýto výpočet splňuje předpoklady Lemmatu 3.4.2 a proto existuje výpočet, při kterém hlavy z oblasti $\overline{\beta}_{i-1}$ projdou slovem $\overline{\beta}_{i-1}0^{d'=d\pm\left(6s^3\right)!}\overline{\beta}_i$ a při kterém rozložení hlav, včetně stavu, relativně k β_{i-1}, α_2 a $\overline{\beta}_{i-1}$ na začátku a relativně k β_i, α_2 a $\overline{\beta}_i$ na konci bude stejné. Ovšem právě takovýto výpočet jsme pro i-tý blok výpočtu nad slovem w_3 hledali a podle zmíněného lemmatu jsme jej také nalezli.

Postupnou náhradou po blocích tak získáme požadovanou prostřední část výpočtu pro slovo w_3, která zachová relativní rozložení hlav, včetně stavu, při výstupní konfiguraci.

Tím jsme sestrojili přijímající výpočet automatu M pro slovo w_3, které však nepatří do jazyka $L^{k,h}$, což je ve sporu s výchozím předpokladem o automatu M. Tím je věta dokázána.□

Věta 3.4.4 *Nechť $k = 4, h \geqslant 1$ jsou přirozená čísla. Pak žádný (k, h)-FA automat bez koincidence nemůže rozpoznávat jazyk $L^{k,h}$.*

---

[22] n je dostatečně velké, tedy $n > \left(6s^3\right)!$, a proto $n^8 + \left(6s^3\right)!$ nemůže být druhou mocninou přirozeného čísla.

Důkaz: Důkaz tohoto tvrzení lze získat modifikací předcházejícího důkazu Věty 3.4.3. Hlavní myšlenka i postup důkazu by byl stejný. Upozorníme zde proto pouze na nutné změny oproti důkazu Věty 3.4.3.

V důkazu tohoto tvrzení bychom se opět omezili na slova, při jejichž přijímacích výpočtech se provádí lustrování všech párů kopií slov $w_{i,j}, u_{i,j}, v_{i,j}$. Při rozboru celkového chování přijímacích výpočtů těchto slov jsme v předcházejícím důkazu zjistili, že v daném slově při lustrování slov $u_{i,j}, v_{i,j}$ lustruje levé kopie těchto slov pro všechna i a j ta samá hlava a pravé kopie těchto slov lustrují stále tytéž tři hlavy. Dokud neproběhne lustrování všech párů kopií slov $u_{i,j}, v_{i,j}$ jsou zbylé hlavy v oblasti α_2. (My se zabýváme automatem se čtyřmi hlavami, tudíž zde žádné zbylé hlavy nejsou.) Ve slovu z jazyka $L^{4,h}$ se oblasti α_1 a α_2 skládají ze tří párů slov, neboť $\binom{k-1}{2} = \binom{k}{2} - 3 = 3$.

Rozdíl ve výpočtu pro slova z $L^{4,h}$ a slova z $L^{>4,h}$ je nyní v tom, že výpočet pro slovo z $L^{4,h}$ by mohl probíhat i tak, že by při lustrování slov $u_{i,j}, v_{i,j}$ lustrovaly levé kopie těchto slov stále tytéž tři hlavy a pravé kopie by lustrovala stále jedna a ta samá hlava. (Oproti druhé možnosti jakoby symetrická situace výpočtu.) Jiný způsob výpočtu (dvě hlavy lustrují levé a dvě hlavy lustrují pravé kopie) není možný, neboť v oblasti α_1 je potřeba lustrovat tři páry kopií slov $w_{1,j}$ a to by při takovém uspořádání výpočtu nebylo možné. Hlavy lustrující pravé kopie slov $u_{i,j}, v_{i,j}$ by mohly v oblasti α_1 lustrovat nejvýše jeden pár slov $w_{1,j}$ a poté hlavy lustrující levé kopie slov $u_{i,j}, v_{i,j}$ by mohly lustrovat další jeden pár slov $w_{1,j}$. Páry jsou však tři, tedy bychom nemohli lustrovat všechny, a proto tento způsob výpočtu není přípustný.

V případě čtyr hlav může tedy výpočet pro slovo z $L^{4,h}$, při kterém se provádí všechna lustrování, probíhat buď tak, že při lustrování slov $u_{i,j}, v_{i,j}$ lustruje levé kopie těchto slov stále tatáž hlava a pravé kopie ostatní tři hlavy, nebo tak, že při lustrování slov $u_{i,j}, v_{i,j}$ lustrují levé kopie těchto slov tři hlavy a pravé kopie jedna a táž hlava. Jelikož jiná možnost není, jedním ze způsobů výpočtu je přijímána alespoň polovina zkoumaných slov.

Pokud se polovina zkoumaných slov chová prvním způsobem, pak lze přímo následovat důkaz Věty 3.4.3 s tím rozdílem, že pracujeme s množinami poloviční velikosti. To však nemá zásadní vliv na úvahy prováděné v tomto důkazu. Jediné místo, které je potřeba podstatněji změnit, je Nerovnost 3.4 a sice na:

$$2^{h\lfloor\log_2 n\rfloor^2-1} - c_4 n^k 2^{\lfloor\log_2 n\rfloor^2(h-1)} n\lfloor\log_2 n\rfloor > 2^{h\lfloor\log_2 n\rfloor^2-2}.$$

Pro množiny L', L'' pak platí poloviční dolní odhad na jejich velikost.

Pokud by se více než polovina slov chovala při výpočtu druhým způsobem, pak bychom opět následovali důkaz Věty 3.4.3 s následujícími odchylkami.

Konfiguraci vstupní a výstupní bychom definovali jako konfiguraci, při které první z hlav vstoupí do pravé kopie slova $v_{0,1}$, resp. $v_{l,1}$, což odpovídá symetrické situaci ve výpočtu. Ostatní tři hlavy budou v těchto konfiguracích zřejmě v oblasti β_0, resp. β_l. Dále by byl důkaz veden stejně s tím, že bychom opět pracovali pouze s polovinou slov a bylo by tedy potřeba provést výše zmíněné úpravy.

V závěrečné části důkazu bychom pak ještě museli použít modifikovanou verzi Lemmatu 3.4.2, které by odpovídalo zkoumané situaci, to jest situaci, při které jedna z hlav prochází ze slova z_1 skrze slovo 0^e do slova z_2 a ostatní hlavy jsou v podslovech x, y omezené velikosti. Takovéto lemma by se dokazovalo stějně jako lemma původní, pouze by situace byla o něco jednodušší v tom, že slovem 0^e prochází pouze jedna hlava. (Srovnej důkaz Lemmatu 3.4.2.)

Tím by byl důkaz tvrzení 3.4.4 hotov. □

Zbývá dokázat Lemma 3.4.2. Nejprve si zavedeme několik pomocných pojmů. Význam všech těchto pojmů je intuitivně zřejmý. Zavedeme zde pojem *instrukce*, což bude čtveřice popisující rozdíl dvou po sobě jdoucích konfigurací ve výpočtu (k,h)-FA automatu. Dále zavedeme pojem *kvazivýpočet*, což bude posloupnost konfigurací, které vzniknou, pokud na nějakou výchozí konfiguraci postupně provádíme změny popsané posloupností instrukcí. Jak snadno nahlédneme, takovýto kvazivýpočet bude výpočtem, pokud po sobě jdoucí konfigurace budou splňovat relaci přechodu. Pokud vzniklý kvazivýpočet bude výpočtem, označíme takovou posloupnost instrukcí pro danou výchozí konfiguraci jako *korektní*.

Definice 3.4.7 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k,h)-FA automat. Pak každou čtveřici $\langle q, t, \lambda, b\rangle \in S \times \{1, \dots, k\} \times \{0, 1\} \times \{E, L\}$ nazveme* ***instrukcí automatu*** *M, přičemž stav q budeme označovat jako* ***cílový stav této instrukce****. Pokud bude automat M zřejmý z kontextu, budeme se na tyto čtveřice odvolávat pouze jako na* ***instrukce****.*

Význam jednotlivých složek instrukce $\langle q, t, \lambda, b\rangle$ je následující. q udává cílový stav instrukce, t udává, se kterou hlavou se pracuje, λ udává, o kolik pozic se hlava t posune, a b udává, zda se na původní pozici položí kámen. O každé instrukci $\langle q, t, \lambda, b\rangle$ tak řekneme, že *pracuje s t-tou hlavou*. O instrukci $\langle q, t, \lambda, b\rangle$ řekneme, že *pohybuje s t-tou hlavou*, právě tehdy, když $\lambda = 1$.

Definice 3.4.8 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k, h)-FA automat. Nechť $r = \langle q, t, \lambda, b\rangle$ je instrukce automatu M a $c = \langle x\$, p, i_1, \ldots, i_k, H\rangle$ je konfigurace nad slovem x. Řekneme, že konfigurace $c' = \langle x\$, q', i'_1, \ldots, i'_k, H'\rangle$* ***vznikla aplikací instrukce*** *r* ***na konfiguraci*** *c, právě tehdy, když splňuje následující tři podmínky:*

1. $q' = q$.

2. $i'_t = i_t + \lambda \ \&\ \forall j \in \{1, \ldots, k\} \setminus \{t\}\,,\ i'_j = i_j$.

3. $(H' = H \cup \{i_t\} \iff b = L)\ \&\ (H' = H \setminus \{i_t\} \iff b = E)\,.$

Pokud c' vznikla aplikací instrukce r na konfiguraci c, pak budeme používat značení: $c' = [c, r\rangle$.

Definice 3.4.9 *Nechť $M = (k, h, S, \Sigma, \delta, \varphi, \$, p_0, F)$ je (k, h)-FA automat. Nechť c je konfigurace automatu M nad slovem $x \in \Sigma^*$ a $r_1, \ldots, r_l$ je posloupnost instrukcí automatu M. Řekneme, že posloupnost konfigurací $c_1, \ldots, c_{l+1}$ je* ***kvazivýpočet*** *V* ***vzniklý aplikací posloupnosti instrukcí*** *$r_1, \ldots, r_l$* ***na konfiguraci*** *c, právě tehdy, když $c_1 = c$ a $\forall i \in \{1, \ldots, l\}\,,\ c_{i+1} = [c_i, r_i\rangle$.*
Budeme používat značení $V = [c, r_1, \ldots, r_l\rangle$.

O kvazivýpočtu vzniklém aplikací instrukcí $r_1, \ldots, r_l$ na konfiguraci c budeme též říkat, že je *určen konfigurací c a posloupností instrukcí $r_1, \ldots, r_l$*.

Ne pro každou výchozí konfiguraci a posloupnost instrukcí musí kvazivýpočet existovat. Pokud by počet instrukcí, které pohybují některou z hlav, byl větší, než o kolik se může hlava z výchozí konfigurace vzhledem k délce vstupního slova pohnout, pak zřejmě kvazivýpočet existovat nemůže. Automat by totiž musel přejít do konfigurace, ve které by byla hlava již za ukončujícím symbolem \$. Taková konfigurace však dle Definice 2.1.2 nemůže existovat. Stejně tak nemusí kvazivýpočet existovat z toho důvodu, že by automat musel položit více kamenů, než má povoleno.

Snadno však nahlédneme, že jednou z postačujících podmínek pro existenci kvazivýpočtu je to, aby počet instrukcí pohybujících jednotlivými hlavami byl menší, než je vzdálenost těchto hlav od ukončujícího symbolu \$ ve výchozí konfiguraci, a aby žádná z instrukcí nepokládala kámen.

Všimněte si, že ne každý kvazivýpočet musí být výpočtem. Aby byl kvazivýpočet výpočtem, musely by jednotlivé konfigurace splňovat relaci přechodu pro daný automat.

Definice 3.4.10 *Nechť c_1 a c_2 jsou konfigurace (k,h)-FA automatu M. Řekneme, že instrukce r je **instrukcí přechodu z konfigurace** c_1 **do konfigurace** c_2 právě tehdy, když $c_2 = [c_1, r\rangle$.*

Definice 3.4.11 *Nechť $V = c_1, \ldots, c_{l+1}$ je výpočet (k,h)-FA automatu. Pak **průběh výpočtu** V je posloupnost instrukcí $r_1, \ldots, r_l$ takových, že pro všechna $i \in \{1, \ldots, l\}$ je r_i instrukcí přechodu z konfigurace c_i do c_{i+1}.*

Snadno nahlédneme, že každý výpočet je jednoznačně určen svojí první konfigurací a svým průběhem a naopak také průběh každého výpočtu je jednoznačně určen.

Definice 3.4.12 *Nechť r je instrukce (k,h)-FA automatu M a c je jeho konfigurace. Řekneme, že **instrukce** r **je z konfigurace** c **korektní**, právě tehdy, když existuje konfigurace $c' = [c, r\rangle$ taková, že $c \vdash_M c'$.*
*Nechť je dále $r_1, \ldots, r_l$ posloupnost instrukcí automatu M. Řekneme, že **posloupnost instrukcí** $r_1, \ldots, r_l$ **je z konfigurace** c **korektní**, právě tehdy, když existuje kvazivýpočet vzniklý aplikací posloupnosti instrukcí $r_1, \ldots, r_l$ na konfiguraci c a tento kvazivýpočet je výpočtem automatu M.*
*O kvazivýpočtu řekneme, že **je korektní**, právě tehdy, když je to výpočet.*

Lemma 3.4.5 *Nechť $r_1, \ldots, r_l$ je posloupnost instrukcí (k,h)-FA automatu M a c_1 je konfigurace automatu M. Nechť existuje kvazivýpočet $V = c_1, \ldots, c_{l+1}$ vzniklý aplikací posloupnosti instrukcí $r_1, \ldots, r_l$ na konfiguraci c. Pak platí: Posloupnost $r_1, \ldots, r_l$ je korektní z konfigurace c_1 (tedy V je výpočet) právě tehdy, když pro všechna $i \in \{1, \ldots, l\}$ je instrukce r_i korektní z konfigurace c_i.*

DŮKAZ: Důkaz plyne z definice. $\square$

Uvedené definice korektnosti formalizují intuitivní představu korektnosti posloupnosti instrukcí. Taková posloupnost je pro danou výchozí konfiguraci korektní, jestliže automat se může touto posloupností instrukcí řídit, neboli může "*vykonávat*" jednotlivé instrukce a bude přecházet do konfigurací, do kterých se může z hlediska své přechodové funkce dostat. Pro danou konfiguraci by tedy nekorektní instrukce byla taková, která by automatu přikazovala přejít do konfigurace, jež by nesplňovala relaci přechodu s výchozí konfigurací, nebo ve které by hlava byla za posledním ukončujícím symbolem \$, nebo ve které by bylo položeno více kamenů, než je přípustné.

Všimněte si, že pro rozhodnutí o tom, zda je instrukce nepokládající kámen z dané konfigurace korektní, je podstatný pouze stav automatu v této konfiguraci, dále symbol pod hlavou,[23] se kterou instrukce pracuje, zda je na tomto symbolu položen kámen a zda se nad tímto symbolem nachází ještě i další hlavy. Jelikož v následujícím textu se budeme zabývat pouze automaty bez koincidence a výpočty, při kterých všechny kameny leží mimo dosah hlav a hlavy žádné kameny nepokládají, bude pro rozhodování o korektnosti dané instrukce podstatný pouze stav automatu a symbol pod hlavou, se kterou tato instrukce pracuje. Tato pozorování shrnuje následující tvrzení.

Lemma 3.4.6 *Nechť je dán (k,h)-FA automat M bez koincidence, $M = (k,h,S,\Sigma,\delta,\varphi,\$,p_0,F)$. Nechť je dána instrukce $r = \langle q,t,\lambda,E\rangle$ a konfigurace c_1 a c_2 takové, že $c_1 = \langle x,p,i_1,\ldots,i_k,H\rangle$ a $c_2 = [c_1,r\rangle$, kde $x = x_1\cdots x_n \in \Sigma^* \times \{\$\}$ a $i_t \notin H$. Nechť je dána konfigurace c_3 a c_4 taková, že $c_3 = \langle y,p',i'_1,\ldots,i'_k,H'\rangle$ a $c_4 = [c_3,r\rangle$, kde $y = y_1\cdots y_m \in \Sigma^* \times \{\$\}$ a $i'_t \notin H'$. Pak platí: pokud $c_1 \vdash_M c_2$, $p = p'$ a $x_{i_t} = y_{i'_t}$, pak $c_3 \vdash_M c_4$.*

DŮKAZ: Důkaz tohoto tvrzení plyne z definic. Pokud $c_1 \vdash_M c_2$ a $c_2 = [c_1,r\rangle$, pak $\varphi(p) = t\ \&\ \langle q,\lambda,E\rangle \in \delta(\langle p,x_{i_t},E,K\rangle)$ (což je rovno $\delta(\langle p,x_{i_t},E,N\rangle)$). Z $p = p'$ a $x_{i_t} = y_{i'_t}$, pak nutně plyne $c_3 \vdash_M c_4 = [c_3,r\rangle$. $\square$

Nyní si zavedeme pojem *cyklu* instrukcí. To je taková podposloupnost posloupnosti instrukcí, že cílový stav instrukce cyklus bezprostředně předcházející je stejný jako cílový stav poslední instrukce cyklu.

[23] Ukončující symbol \$ se nemůže vyskytovat uvnitř žádného slova a přechodová funkce (k,h)-FA automatu musí být dle definice taková, že s hlavou nad symbolem \$ se nesmí pohnout.

Definice 3.4.13 *Nechť $r_1, \dots, r_l$ je posloupnost instrukcí (k,h)-FA automatu. Podposloupnost $r_i, \dots, r_{i+j}$ nazveme* ***cyklem****, pokud instrukce r_{i-1} a r_{i+j} mají stejný cílový stav a žádná z instrukcí $r_i, r_{i+1}, \dots, r_{i+j-1}$ nemá cílový stav totožný s instrukcí r_{i+j}. Stav instrukce r_{i+j} budeme označovat jako* ***cílový stav cyklu****.*
Řekneme, že ***cyklus*** *$r_i, \dots, r_{i+j}$* ***pohybuje hlavou*** *h_t* ***o*** *a* ***pozic****, pokud je mezi instrukcemi cyklu $r_i, \dots, r_{i+j}$ právě a instrukcí, které pohybují hlavou h_t.*

Definice 3.4.14 *Nechť $r_1, \dots, r_l$ je posloupnost instrukcí (k,h)-FA automatu a nechť $r_i, \dots, r_j$ je cyklus v této posloupnosti, kde $i > 1$. Pak* ***posloupnost vzniklá vypuštěním cyklu*** *$r_i, \dots, r_j$* ***z posloupnosti*** *$r_1, \dots, r_l$ je posloupnost instrukcí $r_1, \dots, r_{i-1}, r_{j+1}, \dots, r_l$ (resp. $r_1, \dots, r_{i-1}$ pokud $j = l$) a* ***posloupnost vzniklá zopakováním cyklu*** *$r_i, \dots, r_j$* ***v posloupnosti*** *$r_1, \dots, r_l$ je posloupnost $r_1, \dots, r_{i-1}, r_i, \dots, r_j, r_i, \dots, r_j, r_{j+1}, \dots, r_l$ (resp. $r_1, \dots, r_{i-1}, r_i, \dots, r_j, r_i, \dots, r_j$ pokud $j = l$).*

Všimněte si, že $r_1, \dots, r_{i-1}, r_i, \dots, r_j, r_i, \dots, r_j, r_{j+1}, \dots, r_l = r_1, \dots, r_j, r_i, \dots, r_l$. Dále si také všimněte, že zopakováním cyklu vznikne posloupnost, u které se nedá rozlišit, zda byla "kopie" cyklu vložena před původní cyklus, nebo naopak za původní cyklus. Pro lepší představu však uvažujme, že vkládání se děje za původní cyklus.

Definice 3.4.15 *Nechť $r_1, \dots, r_l$ je posloupnost instrukcí (k,h)-FA automatu M. Nechť automat M má s stavů a q je jeden z jeho stavů. Řekneme, že* ***úsek instrukcí*** *$r_m, \dots, r_n$* ***se skládá z cyklů příslušných stavu*** *q, pokud cílový stav instrukce r_m je q a existují přirozená čísla $m = i_1 < i_2 < \cdots < i_{o+1} = n$ taková, že pro všechna $j \in \{1, \dots, o\}$ je podposloupnost $r_{i_j+1}, \dots r_{i_{j+1}}$ cyklem a $i_{j+1} - i_j \leqslant s$.*

Definice 3.4.16 *Nechť $r_1, \dots, r_l$ je posloupnost instrukcí (k,h)-FA automatu M, který má s stavů. Podposloupnost $r_m, \dots, r_n$ nazveme* ***normalizovanou****, pokud buď $n - m \leqslant s$, nebo existují přirozená čísla $m = n_0 \leqslant m_1 < n_1 < m_2 < \cdots < n_v \leqslant m_{v+1} = n$ a různé stavy $q_1, \dots, q_v$ automatu M takové, že $m_{v+1} - n_v \leqslant s$ a pro všechna $i \in \{1, \dots, v\}$ platí, že $m_i - n_{i-1} \leqslant s$ a úsek instrukcí $r_{m_i}, \dots, r_{n_i}$ se skládá z cyklů příslušných stavu q_i. Každý úsek instrukcí $r_{m_i}, \dots, r_{n_i}$ budeme označovat jako* ***cyklický podblok pro stav*** *q_i.*
Výpočet V nazveme ***normalizovaný*** *pokud je normalizovaný průběh tohoto výpočtu.*

Instrukce normalizovaného průběhu výpočtu, které nepatří do žádného cyklického podbloku, budeme označovat jako *odpad*. Dále také označíme jako odpad první instrukci každého cyklického podbloku. Podmínka $m_i - n_{i-1} \leqslant s$ v předchozí definici spolu s tím, že $v \leqslant s$ (musíme nalézt v různých stavů) říká, že odpadu je při normalizovaném průběhu výpočtu relativně málo, nejvýše s^2 instrukcí.

Následující lemma je nutné dokazovat pouze pro nedeterministické automaty. Pro deterministické automaty toto tvrzení platí triviálně (výpočet deterministického automatu je vždy normalizovaný).

Lemma 3.4.7 *Nechť $k \geqslant 3$, h jsou přirozená čísla. Nechť M je (k,h)-FA automat bez koincidence s s stavy. Nechť $h_1, h_2, \ldots, h_k$ jsou hlavy automatu M. Nechť $x0^ay \in \Sigma^*$. Nechť V je výpočet nad slovem $x0^ay$ takový, že každá z hlav h_1, h_2, h_3 je ve všech konfiguracích buď v podslově 0^a, nebo je v podslově x případně y, všechny ostatní hlavy jsou v podslovech x nebo y, žádná z hlav, která je v podslově x nebo y, se v průběhu výpočtu nepohybuje a všechny kameny jsou položeny, přičemž všechny leží mimo pozice navštívené hlavami v průběhu tohoto výpočtu. Pak existuje normalizovaný výpočet V' ekvivalentní s výpočtem V.*

Důkaz: Důkaz tohoto lemmatu provedeme indukcí podle délky výpočtu V, kde délkou výpočtu rozumíme počet konfigurací výpočtu V.

Pokud je délka výpočtu V nejvýše $s+1$, pak průběh takového výpočtu je normalizovaný, přičemž všechny instrukce lze prohlásit za odpad.

Předpokládejme tedy, že lemma platí pro všechny výpočty uvedených vlastností kratší než $n+1$. Ukážeme, že pak platí i pro výpočty délky $n+1$.

Vezměme si výpočet $V_{n+1} = c_1, \ldots, c_{n+1}$ a jeho průběh výpočtu $r_1, \ldots, r_n$. Podle indukčního předpokladu existuje k výpočtu $V_n = c_1, \ldots, c_n$ ekvivalentní normalizovaný výpočet $V'_n = c'_1, \ldots, c'_n$ s průběhem $r'_1, \ldots, r'_{n-1}$. Snadno nahlédneme, že výpočet $c_1, \ldots, c_{n+1}$ je ekvivalentní s výpočtem $c'_1, \ldots, c'_n, c_{n+1}$, jehož průběh je $r'_1, \ldots, r'_{n-1}, r_n$.

Nechť poslední cyklický podblok průběhu výpočtu $r'_1, \ldots, r'_{n-1}$ končí instrukcí r'_{n-1-d}. Pokud d je menší než $s-1$ pak přidáním instrukce r_n k tomuto průběhu výpočtu nám odpad za posledním cyklickým blokem nepřeroste povolenou mez a tudíž výpočet $r'_1, \ldots, r'_{n-1}, r_n$ bude normalizovaný a není dále co dokazovat.

Pokud je $d = s$ (větší než s být nemůže), pak přidáním instrukce r_n by přerostl poslední úsek odpadu povolenou mez. Za posledním cyklickým blokem by tedy bylo $s + 1$ instrukcí. Jelikož počet stavů automatu je s, musí mít dvě z instrukcí $r'_{n-d}, \ldots, r'_{n-1}, r_n$ stejný cílový stav a tedy existuje zde cyklus.

Označme si instrukci r_n symbolem r'_n. V $r'_{n-d}, \ldots, r'_n$ existuje cyklus $r'_i, \ldots, r'_{i+j}$. Pokud pro cílový stav tohoto cyklu neexistuje v normalizovaném průběhu výpočtu $r'_1, \ldots, r'_{n-1}$ cyklický podblok, pak úsek instrukcí $r'_{i-1}, \ldots, r'_{i+j}$ prohlásíme za nový cyklický podblok posloupnosti $r'_1, \ldots, r'_n$. Tím bude tento průběh výpočtu, jak snadno nahlédneme, znormalizován a výpočet $c'_1, \ldots, c'_n, c_{n+1}$ je hledaný výpočet.

Pokud pro cílový stav cyklu $r'_i, \ldots, r'_{i+j}$ existuje cyklický podblok začínající instrukcí r'_m v posloupnosti $r'_1, \ldots, r'_{n-1}$, pak přesunem instrukcí $r'_i, \ldots, r'_{i+j}$ na začátek tohoto podbloku získáme posloupnost[24] $r'_1, \ldots, r'_m, r'_i, \ldots, r'_{i+j}, r'_{m+1}, \ldots, r'_{i-1}, r'_{i+j+1}, \ldots, r'_n$, která je, jak opět snadno nahlédneme, normalizovaná. Podstatné je také to, že takováto posloupnost bude pro výchozí konfiguraci c_1 výpočtu V korektní a vzniklý výpočet bude ekvivalentní s výpočtem V.

To, že je ekvivalentní s výpočtem V, vyplývá z toho, že celkový posun všech hlav zůstává při prohození instrukcí zachován, z toho, že koncový stav zůstává totožný, a z toho, že kameny byly po celou dobu mimo dosah hlav, a tedy žádný z nich nebyl pokládán ani zvedán.

To, že je takový kvazivýpočet korektní, tedy že vůbec existuje a je to výpočet, vyplývá z toho, že celkový posun hlav zůstává zachován, všechny hlavy, které se v průběhu výpočtu pohybují, se pohybují nad stále stejným symbolem 0, automat je bez koincidence, kameny jsou mimo dosah hlav, a tedy uvedené prohození instrukcí neovlivňuje to, co hlavy automatu vidí při vykonávání jednotlivých instrukcí. Stav automatu v jednotlivých konfiguracích, ze kterých se instrukce provádějí, je po prohození pro všechny instrukce stejný jako původně, neboť je dán cílovým stavem předcházející instrukce. Ke změně by mohlo dojít pouze u instrukce r'_{i+j+1} následující za vyjmutým cyklem, dále u první instrukce r'_i vyjmutého cyklu a instrukce r'_{m+1}. Instrukce předcházející tyto instrukce však mají stejný cílový stav, neboť $r'_i, \ldots, r'_{i+j}$ je cyklus a r_m je první instrukcí cyklického podbloku pro cílový stav instrukce r'_{i+j}. Posloupnost instrukcí je tedy z hlediska přechodové funkce automatu korektní a daný kvazivýpočet je výpočtem.

Ve všech případech jsme tedy nalezli ekvivalentní normalizovaný výpočet a tím je lemma

[24]Pokud $n = i + j$, bude uvedená posloupnost $r'_1, \ldots, r'_m, r'_i, \ldots, r'_{i+j}, r'_{m+1}, \ldots, r'_{i-1}$.

dokázáno. $\square$

Všimněte si, že normalizovaný výpočet nalezený v důkazu předchozího lemmatu má stejnou délku jako původní výpočet.

DŮKAZ (Lemma 3.4.2): Důkaz Lemmatu 3.4.2 je rozdělen do čtyř částí. V první části vybereme výpočet V_0 z konfigurace c^0 do konfigurace c'^0 nad slovem $a_0xa_1ya_2z_10^ez_2a_3$ tak, aby jeho průběh šel rozdělit na cykly s co nejmenším množstvím instrukcí mimo tyto cykly (tyto instrukce budeme označovat jako *odpad*[25]). V další části ukážeme, že množství těchto instrukcí odpadu je pro zvolený výpočet V_0 relativně malé. Ve třetí části důkazu ukážeme, že pokud nalezneme v průběhu výpočtu V_0 cykly takové, že zopakováním některých a vypuštěním jiných se počet instrukcí pohybujících hlavami h_1, h_2, h_3 změní o f, pak bude existovat výpočet nad slovem $a'_0xa'_1ya'_2z_10^{e+f}z_2a'_3$ z konfigurace, která má relativní rozložení hlav v podslovech x, y, z_1 stejné jako konfigurace c^0 do konfigurace, která má relativní rozložení hlav v podslovech x, y, z_2 stejné jako konfigurace c'^0. V závěrečné části důkazu pak ukážeme, že takovéto cykly můžeme pro $f = \pm\left(6s^3\right)!$ skutečně nalézt, a že tedy existuje hledaný výpočet pro slova $a'_0xa'_1ya'_2z_10^{e\pm\left(6s^3\right)!}z_2a'_3$.

Vezměme si libovolný (k, h)-FA automat M s $s \geqslant 1$ stavy a nenulovou konstantu d. Položme $n_0 = 3ds\left(\left(6s^3\right)!\right)(k+3)$. Vezměme si libovolné $n \geqslant n_0$, libovolná slova $a_0, a_1, a_2, a_3, x, y, z_1, z_2, a'_0, a'_1, a'_2, a'_3 \in \Sigma^*$ taková, že $|x|, |y|, |z_1|, |z_2| \leqslant dn$, libovolné $e \geqslant n^5$, a libovolné konfigurace c^0 a c'^0 splňující předpoklady tvrzení a takové, že existuje výpočet V z konfigurace c^0 do konfigurace c'^0. Ukážeme, že pak existuje požadovaný výpočet V^- nad slovem $a'_0xa'_1ya'_2z_10^{e-\left(6s^3\right)!}z_2a'_3$ z konfigurace c^- do konfigurace c'^- a výpočet V^+ nad slovem $a'_0xa'_1ya'_2z_10^{e+\left(6s^3\right)!}z_2a'_3$ z konfigurace c^+ do konfigurace c'^+, kde konfigurace c^- a c^+ mají stejné relativní rozložení hlav, včetně stavu, v podslovech x, y, z_1 jako konfigurace c^0 a konfigurace c'^- a c'^+ mají stejné relativní rozložení hlav, včetně stavu, v podslovech x, y, z_2 jako konfigurace c'^0, přičemž všechny kameny jsou během výpočtů V^- a V^+ v podslovech a'_0, a'_1, a'_2, a'_3.

I. Vezměme si libovolný výpočet $V = c_1, \ldots, c_{l+1}$ nad slovem $a_0xa_1ya_2z_10^ez_2a_3$ z konfigurace $c_1 = c^0$ do konfigurace $c_{l+1} = c'^0$.

Rozdělme tento výpočet na maximální souvislé úseky tak, že se v rámci každého z úseků žádná z hlav, jež je právě mimo podslovo 0^e, nepohne a ani žádná z hlav do podslova 0^e

[25]Termín *odpad* byl již dříve použit pro označení instrukcí mimo cykly v normalizovaném výpočtu. Výpočet V_0 však nebude normalizován jako celek, ale bude normalizován po částech. Mezi instrukce odpadu budeme tedy navíc řadit i instrukce, které oddělují jednotlivé normalizované části výpočtu V_0.

nevstoupí, ani z něj nevystoupí. V rámci jednoho úseku se tedy mohou pohybovat pouze hlavy, které jsou zrovna v podslově 0^e, a to pouze v rámci tohoto podslova. (Pokud se tedy pohne některá z hlav mimo podslovo 0^e, pak výchozí konfigurace tohoto pohybu ukončí jeden úsek výpočtu a cílová konfigurace začne druhý.) Nechť těchto úseků je o a první konfigurací i-tého úseku je konfigurace c_{l_i} ($l_1 = 1$ a dodefinujme $l_{o+1} = l + 2$).

Z předpokladu lemmatu vyplývá, že $o \leqslant (k+3)\,dn$. (Každá z hlav $h_4, \ldots, h_k$ se posune nejvýše o dn pozic a každá z hlav h_1, h_2, h_3 se posune ve slovech z_1 a z_2 nejvýše o $2dn$ pozic.)

Vezměme si průběh výpočtu $r_1, \ldots, r_l$ příslušný k V. Pak i-tým *blokem* tohoto průběhu výpočtu nazveme posloupnost instrukcí $r_{l_i}, r_{l_i+1}, \ldots, r_{l_{i+1}-2}$. (Pokud $l_i + 1 = l_{i+1}$, pak je daný blok prázdný.) Z výše provedeného rozdělení výpočtu V_0 na úseky snadno nahlédneme, že pokud některá z instrukcí nějakého bloku pohne ve výpočtu V hlavou, pak je to jednou z hlav h_1, h_2, h_3 a je to v rámci podslova 0^e ze symbolu 0 opět na symbol 0. Instrukce, které pohybují hlavou mimo podslovo 0^e nejsou součástí žádného bloku.

Rozdělme jednotlivé bloky na nepřekrývající se cykly délky nejvýše s. Ne všechny instrukce bloku musí být součástí nějakého cyklu. Instrukce, které nejsou součástí žádného cyklu budeme, ve shodě s dřívějším značením, označovat jako *odpad*. Mezi instrukce odpadu zařadíme i instrukce, které nejsou součástí žádného bloku, tedy instrukce r_{l_i-1} pro všechna $i \in \{1, \ldots, o\}$.

Jelikož v rámci těchto cyklů se mohou pohybovat pouze hlavy h_1, h_2, h_3, lze každý cyklus charakterizovat trojicí čísel $\langle a, b, c\rangle \in \{0, \ldots, s\}^3$, *typem cyklu*, udávající, o kolik pozic se v průběhu daného cyklu pohne hlava h_1, h_2 a h_3 (počet instrukcí pohybujících danými hlavami.) Různých typů cyklů je tedy $(s+1)^3$. Označme si symbolem $\#\,\langle a, b, c\rangle$ počet cyklů typu $\langle a, b, c\rangle$ v průběhu výpočtu $r_1, \ldots, r_l$ při daném rozdělení. (Pro cyklus $r_i, \ldots, r_{i+j}$ budeme jeho typ značit $\langle a, b, c\rangle_{r_i,\ldots,r_{i+j}}$.)

Pro dané rozdělení na cykly lze výpočet V charakterizovat $(s+1)^3$-ticí přirozených čísel udávajících počet jednotlivých typů cyklů a dále ještě přirozeným číslem udávajícím počet instrukcí odpadu. $(s+1)^3$-tice přirozených čísel lze lexikograficky uspořádat. Ke každému výpočtu V tedy zafixujme takové rozdělení na cykly, při kterém je nejmenší možné množství odpadu a v případě shodnosti množství odpadu pro více rozdělení vezměme některé z nich s nejmenší $(s+1)^3$-ticí udávající počty cyklů.

Nyní ze všech výpočtů nad slovem $a_0xa_1ya_2z_10^ez_2a_3$ z konfigurace c^0 do konfigurace c'^0

vezměme ten výpočet V_0, jehož rozdělení na cykly má nejmenší množství odpadu a jehož $(s+1)^3$-tice čísel udávající počty cyklů je v lexikografickém uspořádání co nejmenší. Jedno takové minimální rozdělení výpočtu V_0 na cykly si pro zbytek důkazu zafixujme.

Ukážeme, že množství odpadu pro výpočet V_0 je menší než $\left(s^2+1\right)(k+3)\,dn$ instrukcí.

II. Když si libovolný výpočet V nad slovem $a_0xa_1ya_2z_10^ez_2a_3$ z konfigurace c^0 do konfigurace c'^0 rozdělíme na o úseků jako výše, pak lze každý z těchto úseků, který obsahuje alespoň dvě konfigurace, nahradit podle Lemmatu 3.4.7 ekvivalentním normalizovaným výpočtem. Po nahrazení všech úseků získáme výpočet V', který je ekvivalentní s původním výpočtem V.

Průběh každého normalizovaného výpočtu lze rozdělit na nepřekrývající se cykly délky nejvýše s a to tak, že instrukcí odpadu je nejvýše s^2. Z toho vyplývá, že jednotlivé bloky průběhu výpočtu V' lze rozdělit na cykly tak, že v každém bloku zůstane nejvýše s^2 instrukcí odpadu. Kromě instrukcí patřících do bloků jsou v průběhu výpočtu V' ještě instrukce, které jednotlivé bloky oddělují (to jsou ty instrukce, které pohybují hlavami mimo podslovo 0^e). Takových instrukcí je právě $o-1$.

Z uvedeného vyplývá, že průběh výpočtu V' lze rozdělit na nepřekrývající se cykly délky nejvýše s tak, že instrukcí odpadu je nejvýše os^2+o-1, a tedy podle výše uvedeného horního odhadu pro o jich je méně než $\left(s^2+1\right)(k+3)\,dn$. Výpočet V_0 jsme vybírali tak, aby jeho průběh šel rozdělit na cykly s co nejmenším množstvím odpadu, a tedy množství odpadu pro zafixované rozdělení je jistě menší než $\left(s^2+1\right)(k+3)\,dn$ instrukcí. Přitom ve všech cyklech tohoto rozdělení platí, že pokud se nějaká hlava pohne, pak je to jedna z hlav h_1, h_2, h_3 nad podslovem 0^e ze symbolu 0 na symbol 0.

Nechť je výpočet $V_0 = c_1, \ldots, c_{l+1}$ a jeho průběh nechť je $r_1, \ldots, r_l$. Nechť $\#\langle a,b,c\rangle$ je počet cyklů typu $\langle a,b,c\rangle$ v průběhu výpočtu $r_1, \ldots, r_l$ při zafixovaném rozdělení.

III. Vezměme si dvě disjunktní množiny nepřekrývajících se cyklů z průběhu výpočtu $r_1, \ldots, r_l$ délky nejvýše s. Nechť to jsou množiny O a P. Nechť pro celé číslo f platí:

$$\sum_{y\in P} \langle a,b,c\rangle_y - \sum_{y\in O} \langle a,b,c\rangle_y = \langle f,f,f\rangle\,. \tag{3.5}$$

Nechť c^f je konfigurace nad slovem $a'_0xa'_1ya'_2z_10^{e+f}z_2a'_3$ taková, že relativní rozložení hlav, včetně stavu, vzhledem k podslovům x, y, z_1 je stejné jako relativní rozložení hlav, včetně stavu, vzhledem k x, y, z_1 v konfiguraci c^0 nad slovem $a_0xa_1ya_2z_10^ez_2a_3$. Nechť jsou v konfiguraci c^f všechny kameny položeny, a to v podslovech a'_0, a'_1, a'_2, a'_3. Nechť posloupnost instrukcí $r^f_1, \ldots, r^f_{l_f}$ je posloupnost instrukcí vzniklá z průběhu výpočtu $r_1, \ldots, r_l$ vypuštěním cyklů z množiny O a zopakováním cyklů z množiny P. Pak posloupnost $r^f_1, \ldots, r^f_{l_f}$ je z konfigurace c^f nad slovem $a'_0xa'_1ya'_2z_10^{e+f}z_2a'_3$ korektní a poslední konfigurace výpočtu $V^f = c^f_1, \ldots, c^f_{l_f+1}$ vzniklého aplikací posloupnosti instrukcí $r^f_1, \ldots, r^f_{l_f}$ na konfiguraci c^f má stejné relativní rozložení hlav, včetně stavu, vzhledem k podslovům x, y, z_2 jako konfigurace $c^{0\prime}$ nad slovem $a_0xa_1ya_2z_10^ez_2a_3$ vzhledem k podslovům x, y, z_2. Tento fakt si nyní ukážeme.

Definujme si nejprve funkci $\tau : \{1, \ldots, l_f\} \longrightarrow \{1, \ldots, l\}$ tak, že $\tau(i) = j$ právě tehdy, když instrukce r^f_i vznikla z instrukce r_j při vytváření posloupnosti $r^f_1, \ldots, r^f_{l_f}$ z průběhu výpočtu $r_1, \ldots, r_l$, tj. buď je to přímo tato instrukce, nebo vznikla jako kopie při opakování cyklů. Pro všechna $i \in \{1, \ldots, l\}$ tedy platí, že $r^f_i = r_{\tau(i)}$. Z následujích obrázků si lze udělat představu o hodnotě funkce τ při opakování cyklu $r_{\tau(i)}, \ldots, r_{\tau(i)+j}$ délky $j+1$:

$$
\begin{array}{ccccccc}
r_{\tau(i-1)} & r_{\tau(i)=\tau(i+j+1)} & \cdots & r_{\tau(i)+j=\tau(i+j)=\tau(i+2j+1)} & & & r_{\tau(i)+j+1=\tau(i+2j+2)} \\
\uparrow & \uparrow & & \uparrow & & \nwarrow & \uparrow \\
r^f_{i-1} & r^f_i & \cdots & r^f_{i+j} & r^f_{i+j+1} & \cdots \quad r^f_{i+2j+1} & r^f_{i+2j+2}
\end{array}
$$

a vypuštění cyklu délky $j+1$:

$$
\begin{array}{ccccccc}
\cdots & r_{\tau(i-1)} & r_{\tau(i-1)+1} & \cdots & r_{\tau(i-1)+j+1} & r_{\tau(i)=\tau(i-1)+j+2} & \cdots \\
& \uparrow & & & & \uparrow & \\
\cdots & r^f_{i-1} & & & & r^f_i & \cdots \;.
\end{array}
$$

Všimněte si, že jelikož posloupnost $r^f_1, \ldots, r^f_{l_f}$ vznikla z posloupnosti $r_1, \ldots, r_l$ opakováním a vypouštěním cyklů a jelikož první instrukce žádné posloupnosti instrukcí nemůže být součástí cyklu, musí být $r^f_1 = r_1$ a $\tau(1) = 1$.

Nejprve ukážeme korektnost posloupnosti instrukcí $r^f_1, \ldots, r^f_{l_f}$ z konfigurace c^f. Musíme ukázat, že existuje kvazivýpočet $c^f_1, \ldots, c^f_{l_f+1}$ vzniklý aplikací posloupnosti $r^f_1, \ldots, r^f_{l_f}$ na konfiguraci c^f a dále to, že tento kvazivýpočet je výpočtem.

Postačující podmínkou pro existenci tohoto kvazivýpočtu je to, aby počet instrukcí pohybujících každou z hlav nebyl příliš velký. Vzhledem k tomu, že ve vypouštených a opakovaných cyklech nemohou být žádné instrukce pohybující hlavami $h_4, \ldots, h_k$, tak počet instrukcí, které pohybují těmito hlavami, je v posloupnosti $r_1^f, \ldots, r_{l_f}^f$ stejný jako v posloupnosti $r_1, \ldots, r_l$. Jelikož v konfiguraci c^f jsou pozice těchto hlav v podslovech x a y stejné jako v konfiguraci c^0 a při výpočtu V_0 neopustí žádná z hlav $h_4, \ldots, h_k$ své podslovo x resp. y, jsou hlavy $h_4, \ldots, h_k$ dostatečně daleko od koncového symbolu \$, aby se mohly vykonat všechny instrukce jimi pohybující. Počet instrukcí, které pohybují hlavami h_1, h_2, h_3, se vlivem vypouštění a opakování cyklů z O a P změnil právě o f. V konfiguraci c^f jsou pozice těchto hlav v podslově z_1 stejné jako v konfiguraci c^0, přičemž celkový posun těchto hlav ve výpočtu V_0 (a tedy počet instrukcí jimi pohybujících) je takový, že se všechny hlavy dostanou právě do podslova z_2. Zkoumaný kvazivýpočet však probíhá nad slovem $a'_0 x a'_1 y a'_2 z_1 0^{e+f} z_2 a'_3$, ve kterém je délka podslova 0^{e+f}, jímž musí hlavy h_1, h_2, h_3 projít, právě o f jiná než ve slově $a_0 x a_1 y a_2 z_1 0^e z_2 a_3$. Díky tomu hlavy h_1, h_2, h_3 jsou od ukončujícího symbolu \$ v konfiguraci c^f dostatečně daleko, aby se mohly vykonat všechny instrukce jimi pohybující. Tedy kvazivýpočet $c_1^f, \ldots, c_{l_f+1}^f$ vzniklý aplikací posloupnosti $r_1^f, \ldots, r_{l_f}^f$ na konfiguraci c^f existuje.

Nyní ukážeme, že kvazivýpočet $c_1^f, \ldots, c_{l_f+1}^f$ je výpočtem. Podle Lemmatu 3.4.5 stačí pro každou instrukci r_i^f, $i \geqslant 1$, ukázat, že je tato instrukce korektní z konfigurace c_i^f.

Korektnost instrukce r_1^f z konfigurace c_1^f vyplývá z toho, že relativní rozložení hlav, včetně stavu, vzhledem k x, y, z_1 je v konfiguraci $c_1^f = c^f$ stejné jako v konfiguraci c^0, a z toho, že instrukce $r_1^f = r_1$. Zbývá tedy ukázat korektnost instrukcí r_i^f pro $i \geqslant 2$.

Instrukci r_i^f odpovídá instrukce $r_{\tau(i)}$ z původního průběhu výpočtu. K tomu, abychom ukázali, že instrukce r_i^f je korektní z konfigurace c_i^f, nám podle Lemmatu 3.4.6 postačuje ukázat, že stav automatu v konfiguraci c_i^f je stejný jako v konfiguraci $c_{\tau(i)}$ a že symbol pod hlavou, se kterou instrukce $r_i^f = r_{\tau(i)}$ pracuje, je v obou konfiguracích stejný. (V je výpočet a tedy $c_{\tau(i)} \vdash_M c_{\tau(i)+1}$.)

Ukažme nejprve, že stavy automatu se v konfiguracích c_i^f a $c_{\tau(i)}$ rovnají. Stav automatu v konfiguraci c_i^f, pro $i \geqslant 2$, je roven cílovému stavu instrukce r_{i-1}^f a obdobně stav automatu v konfiguraci c_j, pro $j \geqslant 2$, je roven cílovému stavu instrukce r_{j-1}. Stačí tedy ukázat, že cílový stav instrukce r_{i-1}^f, která předchází instrukci r_i^f, je roven cílovému stavu instrukce $r_{\tau(i)-1}$, která předchází instrukci $r_{\tau(i)}$.

Rozlišme dva případy. Pokud $\tau(i-1) = \tau(i) - 1$, pak $r^f_{i-1} = r_{\tau(i-1)} = r_{\tau(i)-1}$ a není, co dokazovat. Tento případ odpovídá situaci, kdy instrukce r^f_{i-1} a r^f_i pocházejí ze dvou po sobě jdoucích instrukcí $r_{\tau(i)-1}$ a $r_{\tau(i)}$, tedy buď zůstaly původní, nebo vznikly opakováním cyklu jako kopie původních instrukcí cyklu.

Podívejme se na případ, kdy $\tau(i-1) \neq \tau(i) - 1$. Tento případ odpovídá tomu, že před instrukcí r^f_i byl vyjmut cyklus nebo je instrukce r^f_i první instrukcí opakováním vloženého cyklu. Pokud je r^f_i první instrukcí vloženého cyklu, pak je před ní instrukce r^f_{i-1} totožná s poslední instrukcí vloženého cyklu (viz. opakování cyklu). Tato poslední instrukce má stejný cílový stav, jako instrukce $r_{\tau(i)-1}$, která cyklus předcházela v $r_1, \ldots, r_l$ (viz. definice cyklu a obrázek výše). Tedy cílový stav instrukce r^f_{i-1} je totožný s cílovým stavem instrukce $r_{\tau(i)-1}$, což jsme chtěli ukázat. Pokud byl před instrukcí r^f_i vypuštěn cyklus, pak opět cílový stav instrukce, která tento cyklus bezprostředně předcházela musí být totožný, s cílovým stavem poslední instrukce cyklu. Pokud bylo vypuštěno více bezprostředně po sobě jdoucích cyklů, pak všechny cykly musely být pro stejný cílový stav a opět instrukce, která je v posloupnosti $r_1, \ldots, r_l$ předcházela musí mít stejný cílový stav jako poslední instrukce cyklu, tedy cílový stav instrukce $r_{\tau(i)-1}$ je stejný jako cílový stav instrukce r^f_i, což jsme chtěli ukázat.

Víme tedy, že stav automatu v konfiguraci c^f_i je stejný jako v konfiguraci $c_{\tau(i)}$. Zbývá ukázat pro kažou instrukci r_i, že symbol pod hlavou, se kterou instrukce r_i pracuje, je v konfiguraci c^f_i stejný jako v konfiguraci $c_{\tau(i)}$. Abychom toto ukázali stačí pro instrukce pracující s hlavami $h_4, \ldots, h_k$ ukázat, že relativní poloha těchto hlav vzhledem k podslovům x, y je v konfiguraci c^f_i stejná jako v konfiguraci $c_{\tau(i)}$. U instrukcí pracujících s hlavami h_1, h_2, h_3 bude situace jen o málo složitější.

Vezměme si jednu z hlav $h_4, \ldots, h_k$, hlavu h_t. Pro každé i takové, že instrukce r^f_i pracuje s hlavou h_t, ukážeme, že symbol pod hlavou h_t je v konfiguraci c^f_i stejný jako symbol pod touto hlavou v konfiguraci $c_{\tau(i)}$. Bez újmy na obecnosti můžeme předpokládat, že se hlava h_t nachází v průběhu výpočtu V_0 v podslově x (pokud by se nacházela v podslově y byl by důkaz stejný). V konfiguraci $c^f_1 = c^f$ je hlava h_t nad v pořadí stejným symbolem ve slově x jako v konfiguraci c^0 (stejné relativní rozložení hlav v těchto konfiguracích.) Posloupnost $r^f_1, \ldots, r^f_{l_f}$ vznikla z posloupnosti $r_1, \ldots, r_l$ vypuštěním některých cyklů a zopakováním některých jiných cyklů. V žádném z cyklů nemůže být instrukce pohybující hlavou h_t (takový byl požadavek při dělení na cykly.) Každá instrukce z $r_1, \dots, r_l$, která pohybuje hlavou h_t, musí tedy být i v posloupnosti $r_1^f, \dots, r_{l_f}^f$, neboť nemohla být vypuštěna s žádným cyklem. Zároveň ani žádná taková instrukce nemůže být v posloupnosti $r_1^f, \dots, r_{l_f}^f$ navíc oproti posloupnosti $r_1, \dots, r_l$. Vzájemné pořadí těchto instrukcí, které pohybují hlavou h_t, musí být v $r_1^f, \dots, r_{l_f}^f$ stejné jako v $r_1, \dots, r_l$, neboť vypouštěním ani opakováním cyklů nedochází ke změně pořadí těchto instrukcí. Z toho plyne, že pokud instrukce r_i^f pohybuje hlavou h_t, pak v konfiguraci c_i^f je hlava h_t umístěna nad v pořadí stejným symbolem slova x jako v konfiguraci $c_{\tau(i)}$, neboť instrukci r_i^f i $r_{\tau(i)}$ předchází v posloupnostech $r_1^f, \dots, r_{l_f}^f$ a $r_1, \dots, r_l$ stejný počet instrukcí pohybujících hlavou h_t. Tedy symbol pod hlavou h_t je v konfiguraci c_i^f i $c_{\tau(i)}$ stejný.

Pokud instrukce r_i^f pracuje s hlavou h_t, avšak nepohybuje s ní, pak může být buď původní, nebo mohla vzniknout při opakování cyklů. Každou takovou instrukci r_i^f však v posloupnosti $r_1^f, \dots, r_{l_f}^f$ předchází stejný počet instrukcí pohybujících hlavou h_t jako v posloupnosti $r_1, \dots, r_l$ instrukci $r_{\tau(i)}$. Zopakováním cyklu totiž instrukce r_i^f nemohla změnit své pořadí vzhledem k instrukcím pohybujícím hlavou h_t. Z toho opět vyplývá, že v konfiguraci c_i^f je hlava h_t nad stejným symbolem jako v konfiguraci $c_{\tau(i)}$. Tedy pro každou instrukci r_i^f pracující s hlavou h_t je symbol pod touto hlavou v konfiguraci c_i^f stejný jako v konfiguraci $c_{\tau(i)}$. To jsme chtěli ukázat.

Zbývá ukázat shodnost symbolu pro hlavy h_1, h_2, h_3. Vezměme si jednu z nich, hlavu h_t, a podívejme se na instrukce, které s ní pracují. V první konfiguraci c_1^f je hlava h_t umístěna nad v pořadí stejným symbolem slova z_1 jako v konfiguraci c^0. Instrukce, které hlavou h_t při výpočtu V_0 pohybují ve slově z_1, musely zůstat v posloupnosti $r_1^f, \dots, r_{l_f}^f$ zachovány, neboť nemohou být součástí žádného cyklu.

Nechť r_m je instrukce, při které se ve výpočtu V_0 pohne hlava h_t z posledního symbolu slova z_1 na první symbol slova 0^e, neboli v konfiguraci c_m je hlava h_t na posledním symbolu podslova z_1 a v konfiguraci c_{m+1} je na prvním symbolu podslova 0^e. Tato instrukce nemůže být součástí žádného cyklu, a tedy existuje i_m takové, že $\tau(i_m) = m$. Všechny instrukce r_j, $j \leqslant m$, které pohybují hlavou h_t, jsou v posloupnosti $r_1^f, \dots, r_{l_f}^f$ zachovány a každá je tam právě jednou, neboť se nemohly zopakovat s cyklem. Snadno nahlédneme, že jejich pořadí v posloupnosti $r_1^f, \dots, r_{l_f}^f$ zůstalo taktéž zachováno. Stejnou úvahou jako pro hlavy $h_4, \dots, h_k$ tak nahlédneme, že pro každou instrukci r_i^f, která pracuje s hlavou h_t a $i \leqslant i_m$, je v konfiguraci c_i^f hlava h_t nad v pořadí stejným symbolem slova z_1 jako v konfiguraci $c_{\tau(i)}$. V konfiguraci $c_{i_m+1}^f$ je tak hlava

h_t nad prvním symbolem podslova 0^{e+f}.

Nechť r_n je instrukce, která při výpočtu V_0 pohne hlavou h_t z posledního symbolu podslova 0^e na první symbol podslova z_2, neboli v konfiguraci c_n je hlava h_t na posledním symbolu podslova 0^e a v konfiguraci c_{n+1} je na prvním symbolu podslova z_2. Zřejmě instrukce r_n zůstala v posloupnosti $r_1^f, \ldots, r_{l_f}^f$ opět zachována, a tedy existuje i_n takové, že $\tau(i_n) = n$. Snadno nahlédneme, že mezi instrukcemi $r_{m+1}, \ldots, r_{n-1}$ je instrukcí, které pohybují hlavou h_t, právě $e - 1$. Tyto instrukce mohou být součástí cyklů, a tudíž ne všechny musí být zachovány v posloupnosti $r_1^f, \ldots, r_{l_f}^f$, a naopak některé z nich mohou být zopakovány. Rozdíl počtu instrukcí, které pohybují hlavou h_t a byly zopakovány s cykly, a počtu instrukcí, které pohybují s hlavou h_t a byly vypuštěny s cykly, je podle Rovnice 3.5 právě f. Tedy počet instrukcí pohybujících hlavout h_t v posloupnosti $r_{i_m+1}^f, \ldots, r_{i_n-1}^f$ je $e - 1 + f$. Z toho vyplývá, že v konfiguraci $c_{i_n}^f$ je hlava h_t nad posledním symbolem podslova 0^{e+f}. Ve všech konfiguracích $c_{m+1}, \ldots, c_{n-1}$ je hlava h_t nad symbolem 0 (je v podslovu 0^e) a stejně tak ve všech konfiguracích $c_{i_m+1}^f, \ldots, c_{i_n-1}^f$ je hlava h_t nad symbolem 0 (je v podslovu 0^{e+f}). Pro každou instrukci r_i^f posloupnosti $r_{i_m+1}^f, \ldots, r_{i_n-1}^f$ pracujícící s hlavou h_t zřejmě platí: $m < \tau(i) < n$. Tedy pro každou takovou instrukce r_i^f je symbol pod hlavou h_t v konfiguraci c_i^f stejný jako v konfiguraci $c_{\tau(i)}$.

Zbývá ukázat, že to platí i pro instrukce z $r_{i_n}^f, \ldots, r_{l_f}^f$ pracující s hlavou h_t. Jelikož instrukce $r_{i_n}^f$ pohybuje hlavou h_t, je v konfiguraci $c_{i_n}^f$ stejně jako v konfiguraci c_n hlava h_t nad prvním symbolem slova z_2. Z toho vyplývá, že instrukce r_i^f, $i \geqslant i_n$, pohybující hlavou h_t pocházejí z instrukcí $r_n, \ldots, r_l$, a zůstaly tedy stejně jako instrukce pohybující touto hlavou ve slově z_1 zachovány v posloupnosti $r_{i_n}^f, \ldots, r_{l_f}^f$ a to ve stejném pořadí. Úvahou obdobnou jako pro instrukce $r_1^f, \ldots, r_{i_m}^f$ pak zjistíme, že pro každou instrukci r_i^f z posloupnosti $r_{i_n}^f, \ldots, r_{l_f}^f$ pracující s hlavou h_t je v konfiguraci c_i^f hlava h_t nad stejným symbolem jako v konfiguraci $c_{\tau(i)}$.

Tím jsme pro kažou instrukci r_i ukázali, že symbol pod hlavou, se kterou instrukce r_i pracuje, je v konfiguraci c_i^f stejný jako v konfiguraci $c_{\tau(i)}$. Jelikož jsme taktéž ukázali, že stav automatu v konfiguraci c_i^f je stejný jako v konfiguraci $c_{\tau(i)}$, tak podle Lemmatu 3.4.6 víme, že konfigurace c_i^f a c_{i+1}^f pro $i \in \{1, \ldots, l_f\}$ splňují relaci přechodu a tedy posloupnost $c_1^f, \ldots, c_{l_f+1}^f$ je výpočtem. To jsme chtěli ukázat.

Dále máme ukázat, že relativní rozložení hlav v konfiguraci $c_{l_f+1}^f$ vzhledem k podslovům

x, y, z_2 je stejné jako v konfiguraci $c'^0 = c_{l+1}$. To však vyplývá přímo z výše provedného rozboru.

Pokud tedy máme ukázat, že pro slova $a'_0 x a'_1 y a'_2 z_1 0^{e+(6s^3)!} z_2 a'_3$ a $a'_0 x a'_1 y a'_2 z_1 0^{e-(6s^3)!} z_2 a'_3$ existuje výpočet z konfigurace c^+ resp. c^- vlastností požadovaných v lemmatu, stačí ukázat, že můžeme nalézt dvě disjunktní množiny cyklů O a P takové, že

$$\sum_{y \in P} \langle a, b, c \rangle_y - \sum_{y \in O} \langle a, b, c \rangle_y = \left\langle (6s^3)!, (6s^3)!, (6s^3)! \right\rangle$$

resp.

$$\sum_{y \in P} \langle a, b, c \rangle_y - \sum_{y \in O} \langle a, b, c \rangle_y = \left\langle -(6s^3)!, -(6s^3)!, -(6s^3)! \right\rangle.$$

Předtím, než se vydáme tyto množiny hledat, si všimněme ještě jedné vlastnosti výpočtu V_0. V důkazu o korektnosti posloupnosti $r_1^f, \ldots, r_{l_f}^f$ z konfigurace c^f se nijak nevyužívalo vlastností slov $a_0, a_1, a_2, a_3, a'_0, a'_1, a'_2, a'_3$. Tedy posloupnost $r_1^f, \ldots, r_{l_f}^f$ by byla korektní i z konfigurace nad slovem $a_0 x a_1 y a_2 z_1 0^{e+f} z_2 a_3$ se stejným relativním rozložením hlav, včetně stavu, vzhledem k x, y, z_1 jako v konfiguraci c^0. Pokud bychom tedy pro výše zafixované dělení výpočtu V_0 na cykly našli dvě disjunktní množiny cyklů O a P takové, že $O \cup P \neq \emptyset$ a $\sum_{y \in P} \langle a, b, c \rangle_y - \sum_{y \in O} \langle a, b, c \rangle_y = \langle 0, 0, 0 \rangle$, pak by posloupnost instrukcí vzniklá z posloupnosti $r_1, \ldots, r_l$ vypuštěním cyklů z O a zopakovaním cyklů z P byla korektní z konfigurace c^0 nad slovem $a_0 x a_1 y a_2 z_1 0^e z_2 a_3$. Stějně tak by byla korektní posloupnost instrukcí vzniklá vypuštěním cyklů z P a zopakovaním cyklů z O.

Při vypouštení ani opakování cyklů se nemění množství instrukcí odpadu (pro vzniklý výpočet bychom ho mohli případně zmenšit při jiném přerozdělení na cykly). Výpočet V_0 je minimální s ohledem na množství odpadu a mezi výpočty se stejným množstvím odpadu má charakteristiku podle počtu cyklů jednotlivých typů v lexikografickém uspořádání nejmenší. Pokud by typy cyklů z množiny P byly různé od typů cyklů z množiny O, pak by vzniklé výpočty musely mít tuto charakteristiku různou od této charakteristiky výpočtu V_0. Podstatné je však to, že jeden z těchto výpočtů by ji měl větší a druhý menší než výpočet V_0 (uspořádání je lexikografické). To by byl spor s výběrem výpočtu V_0.

V posloupnosti $r_1, \ldots, r_l$ tedy nelze pro zafixované rozdělení na cykly nalézt dvě disjunktní množiny cyklů O a P takové, že by $O \cup P \neq \emptyset$, typy cyklů z množiny O by byly různé od typů cyklů z množiny P a $\sum_{y \in P} \langle a, b, c \rangle_y - \sum_{y \in O} \langle a, b, c \rangle_y = \langle 0, 0, 0 \rangle$.

Mimo jiné z toho vyplývá, že žádný z cyklů zafixovaného rozdělení nemůže být typu $\langle 0,0,0\rangle$. Dále také není možné, aby existovaly tři různé typy $\langle a_0,b_0,c_0\rangle, \langle a_1,b_1,c_1\rangle, \langle a_2,b_2,c_2\rangle$ takové, že pro nějaká celá čísla i_1,i_2,i_3 taková, že $\forall 1\leqslant j\leqslant 3, |i_j|\leqslant \#\langle a_j,b_j,c_j\rangle$ & $|i_1|+|i_2|+|i_3|>0$, by bylo $\sum_{1\leqslant j\leqslant 3} i_j\langle a_2,b_2,c_2\rangle = \langle 0,0,0\rangle$. Zařazením $|i_j|$ cyklů typu $\langle a_j,b_j,c_j\rangle$ do množiny O, pokud $i_j<0$, resp. jejich zařazením do množiny P, pokud $i_j\geqslant 0$, pro $j=1,2,3$, bychom získali množiny O a P, které by byly ve sporu s výběrem výpočtu V_0. Pro takové O a P je totiž $\sum_{y\in P}\langle a,b,c\rangle_y - \sum_{y\in O}\langle a,b,c\rangle_y = \sum_{1\leqslant j\leqslant 3} i_j\langle a_2,b_2,c_2\rangle = \langle 0,0,0\rangle$.[26]

IV. V této části důkazu ukážeme, že v posloupnosti $r_1,\ldots,r_l$ lze pro zafixované rozdělení na cykly vždy nalézt disjunktní množiny cyklů O a P takové, že $\sum_{y\in P}\langle a,b,c\rangle_y - \sum_{y\in O}\langle a,b,c\rangle_y = \left\langle (6s^3)!, (6s^3)!, (6s^3)!\right\rangle$ resp. $\left\langle -(6s^3)!, -(6s^3)!, -(6s^3)!\right\rangle$. Tím bude důkaz Lemmatu 3.4.2 hotov.

Víme, že počet instrukcí odpadu v průběhu výpočtu V_0 je menší než $\left(s^2+1\right)(k+3)\,dn$.

Hlava h_1 se v průběhu výpočtu pohne alespoň o n^5 pozic a jelikož v každém cyklu se může pohnout maximálně o s pozic, musí existovat typ cyklu s první složkou nenulovou, který se v průběhu výpočtu $r_1,\ldots,r_l$ vyskytuje alespoň $\left(n^5-\left(s^2+1\right)(k+3)\,dn\right)/\left(s\,(s+1)^3\right)$ krát. Nechť je to typ $\langle a_0,b_0,c_0\rangle$.

1. Rozeberme případ $a_0=b_0=c_0$.

V případě $a_0=b_0=c_0$ se v průběhu výpočtu $r_1,\ldots,r_l$ vyskytují cykly, ve kterých jsou hlavy h_1,h_2,h_3 nad podslovem 0^e a ve kterých se tyto hlavy pohnou o stejný počet pozic.

Pro jejich počet $\#\langle a_0,b_0,c_0\rangle$ po dosazení za n získáme:

$$\begin{aligned}
\#\langle a_0,b_0,c_0\rangle &\geqslant \frac{n^5-\left(s^2+1\right)(k+3)\,dn}{s\,(s+1)^3}\\
&\geqslant \frac{3ds\left(6s^3\right)!\,(k+3)\,n^4-\left(s^2+1\right)(k+3)\,dn}{s\,(s+1)^3}\\
&\geqslant n^4\frac{3ds\left(6s^3\right)!\,(k+3)-\left(s^2+1\right)(k+3)\,dn^{-3}}{s\,(s+1)^3}\\
&\geqslant n^4\frac{2ds\left(6s^3\right)!\,(k+3)}{s\,(s+1)^3}
\end{aligned}$$

[26] Z uvedeného lze nahlédnout, že nejvýše tři druhy cyklů se v zafixovaném rozdělení mohou vyskytovat ve větším množství. Další by na nich musely být lineárně závislé a tudíž by bylo možné obdobně je nakombinovat tak, aby celkový pohyb hlav byl nulový. Z řešení soustav lineárních rovnic pomocí Cramerova pravidla se dá nahlédnout, že by k tomu stačilo $6s^3$ cyklů od každého druhu.

$$
\begin{aligned}
&\geqslant n^4 &(3.6)\\
&\geqslant \left(6s^3\right)!.
\end{aligned}
$$

Pokud tedy do množiny O vybereme $\frac{(6s^3)!}{a_0}$ cyklů typu $\langle a_0, b_0, c_0\rangle$ a množinu P ponecháme prázdnou, pak

$$
\begin{aligned}
\sum_{y\in P}\langle a,b,c\rangle_y - \sum_{y\in O}\langle a,b,c\rangle_y &= -\sum_{y\in O}\langle a,b,c\rangle_y\\
&= -\frac{\left(6s^3\right)!}{a_0}\langle a_0,a_0,a_0\rangle\\
&= \left\langle -\left(6s^3\right)!, -\left(6s^3\right)!, -\left(6s^3\right)!\right\rangle.
\end{aligned}
$$

Tím jsme nalezli množiny O a P požadovaných vlastností.

2. Pokud není $a_0 = b_0 = c_0$, je situace o něco složitější.

Jelikož hlavy h_1, h_2, h_3 jsou na začátku výpočtu V_0 ve slově z_1 a na konci ve slově z_2, kde $|z_1|, |z_2| \leqslant dn$, mohou se počty instrukcí, které pohybují jednotlivými hlavami v posloupnosti $r_1, \ldots, r_l$, lišit nejvýše o $2dn$. Rozdíl počtu instrukcí pohybujících hlavou h_1 a počtu instrukcí pohybujících hlavou h_2 označme q_1. Snadno nahlédneme, že pro něj platí:

$$
\begin{aligned}
&-\left(s^2+1\right)(k+3)\,dn + \sum_{\substack{\langle a,b,c\rangle\\ \in\{0,\ldots,s\}^3}} (a-b)\,.\#\langle a,b,c\rangle \leqslant q_1\\
&\leqslant \left(s^2+1\right)(k+3)\,dn + \sum_{\substack{\langle a,b,c\rangle\\ \in\{0,\ldots,s\}^3}} (a-b)\,.\#\langle a,b,c\rangle\,.
\end{aligned}
$$

Pro každé $\langle a,b,c\rangle \in \{0,\ldots,s\}^3$ totiž výraz $(a-b)\,.\#\langle a,b,c\rangle$ udává rozdíl počtu instrukcí pohybujících hlavou h_1 a počtu instrukcí pohybujících hlavou h_2, které jsou v cyklech typu $\langle a,b,c\rangle$. Instrukce odpadu, kterých je méně než $\left(s^2+1\right)(k+3)\,dn$, pak případně tento rozdíl ještě zvýší, či sníží.

Zároveň víme, že $-2dn \leqslant q_1 \leqslant 2dn$. Z uvedených nerovností vyplývá:

$$
-\left(s^2+1\right)(k+3)\,dn - 2dn \leqslant \sum_{\substack{\langle a,b,c\rangle\\ \in\{0,\ldots,s\}^3}} (a-b)\,.\#\langle a,b,c\rangle \leqslant \left(s^2+1\right)(k+3)\,dn + 2dn.
$$

Pro rozdíl pohybu hlav h_1 a h_3 odvodíme obdobný vztah:

$$-\left(s^2+1\right)(k+3)\,dn-2dn \leqslant \sum_{\substack{\langle a,b,c\rangle \\ \in\{0,\ldots,s\}^3}} (a-c)\,.\#\,\langle a,b,c\rangle \leqslant \left(s^2+1\right)(k+3)\,dn+2dn.$$

Nyní ukážeme, že musí existovat $\langle a_1,b_1,c_1\rangle$ různý od $\langle a_0,b_0,c_0\rangle$ takový, že $\langle a_1,b_1,c_1\rangle > n^2$.

Jelikož není $a_0=b_0=c_0$, je (a_0-b_0) nebo (a_0-c_0) různé od nuly. Bez újmy na obecnosti můžeme předpokládat, že $(a_0-b_0)\neq 0$. (Případ $(a_0-c_0)\neq 0$ by byl symetrický s tímto.)

Tedy:

$$\left|(a_0-b_0)\,.\#\,\langle a_0,b_0,c_0\rangle + \sum_{\substack{\langle a,b,c\rangle \\ \neq\langle a_0,b_0,c_0\rangle}} (a-b)\,.\#\,\langle a,b,c\rangle\right| \leqslant 2dn+\left(s^2+1\right)(k+3)\,dn.$$

Pokud by pro každý typ cyklu $\langle a,b,c\rangle$ různý od $\langle a_0,b_0,c_0\rangle$ byl počet cyklů tohoto typu nejvýše n^2, tedy $\#\,\langle a,b,c\rangle \leqslant n^2$, pak by platilo:[27]

$$\begin{aligned}
\frac{n^5-\left(s^2+1\right)(k+3)\,dn}{s\,(s+1)^3} &\leqslant \#\,\langle a_0,b_0,c_0\rangle \\
&\leqslant \left|(a_0-b_0)\,.\#\,\langle a_0,b_0,c_0\rangle\right| \\
&\leqslant 2dn+\left(s^2+1\right)(k+3)\,dn+\left|\sum_{\substack{\langle a,b,c\rangle \\ \neq\langle a_0,b_0,c_0\rangle}} (a-b)\,.\#\,\langle a,b,c\rangle\right| \\
&\leqslant 2dn+\left(s^2+1\right)(k+3)\,dn+(s+1)^3\,sn^2
\end{aligned}$$

a podle Nerovnosti 3.6 a $n\geqslant n_0$:

$$\begin{aligned}
n^4 &\leqslant \frac{n^5-\left(s^2+1\right)(k+3)\,dn}{s\,(s+1)^3} \\
&\leqslant 2dn+\left(s^2+1\right)(k+3)\,dn+s\,(s+1)^3\,n^2 \\
&\leqslant n^2\left(2d+\left(s^2+1\right)(k+3)\,d+s\,(s+1)^3\right) \\
&\leqslant n^3.
\end{aligned}$$

[27] $\frac{n^5-\left(s^2+1\right)(k+3)dn}{s(s+1)^3} > 0.$

To by však byl spor, neboť $n > 1$.

Existuje tedy $\langle a_1, b_1, c_1 \rangle \neq \langle a_0, b_0, c_0 \rangle$ takový, že $\# \langle a_1, b_1, c_1 \rangle > n^2$. Rozlišme dva případy.

A. Existuje na $\langle a_0, b_0, c_0 \rangle$ lineárně závislý $\langle a_1, b_1, c_1 \rangle$ takový, že $\# \langle a_1, b_1, c_1 \rangle > n^2$. Pak ovšem

$$a_1 \langle a_0, b_0, c_0 \rangle - a_0 \langle a_1, b_1, c_1 \rangle = \langle 0, 0, 0 \rangle$$

a jelikož $a_0, a_1 \leqslant s < n^2$, dostali bychom spor se závěrečnou poznámkou Části III důkazu. Tento případ tedy nenastává.

B. Neexistuje na $\langle a_0, b_0, c_0 \rangle$ lineárně závislý $\langle a_1, b_1, c_1 \rangle$ takový, že $\# \langle a_1, b_1, c_1 \rangle > n^2$, a tedy existuje $\langle a_1, b_1, c_1 \rangle$ lineárně nezávislý na $\langle a_0, b_0, c_0 \rangle$ takový, že $\# \langle a_1, b_1, c_1 \rangle > n^2$.

Rozlišme dva případy podle hodnosti matice:

$$B = \begin{pmatrix} a_0 & a_1 & 1 \\ b_0 & b_1 & 1 \\ c_0 & c_1 & 1 \end{pmatrix}.$$

1) Matice B je singulární. Pak má lineární soustava rovnic:

$$\begin{aligned} ia_0 \quad &+ja_1 &= f \\ ib_0 \quad &+jb_1 &= f \\ ic_0 \quad &+jc_1 &= f, \end{aligned}$$

kde f je celé číslo, řešení.

Jelikož $\langle a_0, b_0, c_0 \rangle$ a $\langle a_1, b_1, c_1 \rangle$ jsou lineárně nezávislé, je:

$$\text{hodnost} \begin{pmatrix} a_0 & b_0 & c_0 \\ a_1 & b_1 & c_1 \end{pmatrix} = \text{hodnost} \begin{pmatrix} a_0 & a_1 \\ b_0 & b_1 \\ c_0 & c_1 \end{pmatrix} = 2.$$

Z toho vyplývá, že alespoň jedna dvojice vektorů $\langle a_0, a_1 \rangle$ a $\langle b_0, b_1 \rangle$ nebo $\langle a_0, a_1 \rangle$ a $\langle c_0, c_1 \rangle$ nebo $\langle b_0, b_1 \rangle$ a $\langle c_0, c_1 \rangle$ je lineárně nezávislá. Bez újmy na obecnosti předpokládejme, že $\langle a_0, a_1 \rangle$ a $\langle b_0, b_1 \rangle$ jsou lineárně nezávislé. (V ostatních případech by se postupovalo obdobně.)

Tedy $a_0 b_1 - a_1 b_0 \neq 0$ a podle Cramerova pravidla dostáváme:

$$i = \frac{f b_1 - f a_1}{a_0 b_1 - a_1 b_0} \qquad\qquad j = \frac{f a_0 - f b_0}{a_0 b_1 - a_1 b_0}.$$

Položme $f = \left(6s^3\right)!$. Pak i, j jsou zřejmě celá čísla v absolutní hodnotě nejvýše rovná číslu $2s\left(6s^3\right)! < n^2$.

Tedy zařazením $|i|$ cyklů typu $\langle a_0, b_0, c_0\rangle$ do množiny O, pokud $i < 0$, resp. jejich zařazením do množiny P, pokud $i \geqslant 0$, a dále pak zařazením $|j|$ cyklů typu $\langle a_1, b_1, c_1\rangle$ do množiny O, pokud $j < 0$, resp. jejich zařazením do množiny P, pokud $j \geqslant 0$, získáme množiny cyklů O a P takové, že $\sum_{y\in P} \langle a, b, c\rangle_y - \sum_{y\in O} \langle a, b, c\rangle_y = \left\langle \left(6s^3\right)!, \left(6s^3\right)!, \left(6s^3\right)! \right\rangle$.

Takovéto množiny jsme chtěli nalézt.

2) Matice B je regulární. Pak existuje ještě typ cyklu $\langle a_2, b_2, c_2\rangle$ různý od $\langle a_0, b_0, c_0\rangle$ a $\langle a_1, b_1, c_1\rangle$ takový, že $\# \langle a_2, b_2, c_2\rangle > n^2$. Ukážeme to sporem.

Předpokládejme, že neexistuje.

Jelikož platí:

$$\begin{aligned}
&\left|(a_0 - b_0) .\# \langle a_0, b_0, c_0\rangle + (a_1 - b_1) .\# \langle a_1, b_1, c_1\rangle + \sum_{\substack{\langle a,b,c\rangle \neq \\ \langle a_0,b_0,c_0\rangle, \langle a_1,b_1,c_1\rangle}} (a - b) .\# \langle a, b, c\rangle\right| \\
\leqslant\ & 2dn + \left(s^2 + 1\right)(k + 3)\, dn,
\end{aligned}$$

dostáváme:

$$\begin{aligned}
&\left|(a_0 - b_0) .\# \langle a_0, b_0, c_0\rangle + (a_1 - b_1) .\# \langle a_1, b_1, c_1\rangle\right| \\
\leqslant\ & 2dn + \left(s^2 + 1\right)(k + 3)\, dn + \left| \sum_{\substack{\langle a,b,c\rangle \neq \\ \langle a_0,b_0,c_0\rangle, \langle a_1,b_1,c_1\rangle}} (a - b) .\# \langle a, b, c\rangle \right| \\
\leqslant\ & 2dn + \left(s^2 + 1\right)(k + 3)\, dn + (s + 1)^3 sn^2 \\
\leqslant\ & n^2 \left(2d + \left(s^2 + 1\right)(k + 3)\, d + (s + 1)^3 s\right) \\
\leqslant\ & n^3.
\end{aligned}$$

Stejnou nerovnost dostáváme pro $(a_0 - c_0)$ a $(a_1 - c_1)$ a platí tedy:

$$
\begin{aligned}
\left|\#\langle a_0, b_0, c_0\rangle (a_0 - b_0) + \#\langle a_1, b_1, c_1\rangle (a_1 - b_1)\right| &\leqslant n^3 && (3.7)\\
\left|\#\langle a_0, b_0, c_0\rangle (a_0 - c_0) + \#\langle a_1, b_1, c_1\rangle (a_1 - c_1)\right| &\leqslant n^3. && (3.8)
\end{aligned}
$$

Nechť u a v jsou takové, že splňují následující rovnosti:

$$
\begin{aligned}
\#\langle a_0, b_0, c_0\rangle (a_0 - b_0) + \#\langle a_1, b_1, c_1\rangle (a_1 - b_1) &= u\\
\#\langle a_0, b_0, c_0\rangle (a_0 - c_0) + \#\langle a_1, b_1, c_1\rangle (a_1 - c_1) &= v.
\end{aligned}
$$

Toto je soustava rovnic pro neznámé $\#\langle a_0, b_0, c_0\rangle$ a $\#\langle a_1, b_1, c_1\rangle$.

Jelikož matice B je regulární, je regulární i matice:

$$
B' = \begin{pmatrix} a_0 & a_1 & 1 \\ a_0 - b_0 & a_1 - b_1 & 0 \\ a_0 - c_0 & a_1 - c_1 & 0 \end{pmatrix}.
$$

Z toho vyplývá, že podmatice:

$$
\begin{pmatrix} a_0 - b_0 & a_1 - b_1 \\ a_0 - c_0 & a_1 - c_1 \end{pmatrix}
$$

je také regulární a tedy $(a_0 - b_0)(a_1 - c_1) - (a_1 - b_1)(a_0 - c_0) \neq 0$. Řešením výše uvedené soustavy pomocí Cramerova pravidla dostaneme:

$$
\#\langle a_0, b_0, c_0\rangle = \frac{u(a_1 - c_1) - v(a_1 - b_1)}{(a_0 - b_0)(a_1 - c_1) - (a_1 - b_1)(a_0 - c_0)}.
$$

Podle Nerovnice 3.6 je:

$$
n^4 \leqslant \#\langle a_0, b_0, c_0\rangle,
$$

podle Nerovnic 3.7 a 3.8 je $|u|, |v| \leqslant n^3$. Jelikož $a_0, a_1, b_0, b_1, c_0, c_1$ jsou celá čísla, je

$$
\left|(a_0 - b_0)(a_1 - c_1) - (a_1 - b_1)(a_0 - c_0)\right| \geqslant 1.
$$

Dostáváme:

$$\begin{aligned}
n^4 &\leqslant \# \langle a_0, b_0, c_0 \rangle \\
&= \frac{u\,(a_1 - c_1) - v\,(a_1 - b_1)}{(a_0 - b_0)\,(a_1 - c_1) - (a_1 - b_1)\,(a_0 - c_0)} \\
&\leqslant \left| \frac{u\,(a_1 - c_1) - v\,(a_1 - b_1)}{(a_0 - b_0)\,(a_1 - c_1) - (a_1 - b_1)\,(a_0 - c_0)} \right| \\
&\leqslant \frac{|u\,(a_1 - c_1) - v\,(a_1 - b_1)|}{|(a_0 - b_0)\,(a_1 - c_1) - (a_1 - b_1)\,(a_0 - c_0)|} \\
&\leqslant |u\,(a_1 - c_1)| + |v\,(a_1 - b_1)| \\
&\leqslant 4sn^3.
\end{aligned}$$

Jelikož však $n \geqslant n_0$, je $n^4 > 4sn^3$. Tím jsme získali spor a musí tedy existovat typ cyklu $\langle a_2, b_2, c_2 \rangle$ různý od $\langle a_0, b_0, c_0 \rangle$ i $\langle a_1, b_1, c_1 \rangle$ takový, že $\# \langle a_2, b_2, c_2 \rangle > n^2$, což jsme měli ukázat.

Rozeberme dva případy:

a) $\langle a_0, b_0, c_0 \rangle$, $\langle a_1, b_1, c_1 \rangle$ a $\langle a_2, b_2, c_2 \rangle$ jsou lineárně nezávislé.

Řešme soustavu lineárních rovnic:

$$\begin{array}{llll}
ia_0 & +ja_1 & +ka_2 & = f \\
ib_0 & +jb_1 & +kb_2 & = f \\
ic_0 & +jc_1 & +kc_2 & = f,
\end{array}$$

kde f je celé číslo. Tato soustava je podle předpokladů regulární. Označme matice:

$$A = \begin{pmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{pmatrix} \qquad A_i = \begin{pmatrix} f & a_1 & a_2 \\ f & b_1 & b_2 \\ f & c_1 & c_2 \end{pmatrix}$$
$$A_j = \begin{pmatrix} a_0 & f & a_2 \\ b_0 & f & b_2 \\ c_0 & f & c_2 \end{pmatrix} \qquad A_k = \begin{pmatrix} a_0 & a_1 & f \\ b_0 & b_1 & f \\ c_0 & c_1 & f \end{pmatrix}.$$

Podle Cramerova pravidla je $i = \det A_i / \det A$, $j = \det A_j / \det A$ a $k = \det A_k / \det A$.

Jelikož všechny prvky matice A jsou čísla z množiny $\{0, \ldots, s\}$, je zřejmě $|\det A| \leqslant 6s^3$. Dále také snadno nahlédneme, že determinanty matic A_i, A_j a A_k jsou násobkem čísla f.

Položme $f = \left(6s^3\right)!$. Pak i, j, k jsou zřejmě celá čísla v absolutní hodnotě nejvýše rovná číslu $6s^2\left(6s^3\right)! < n^2$.

Tedy zařazením $|i|$ cyklů typu $\langle a_0, b_0, c_0\rangle$ do množiny O, pokud $i < 0$, resp. jejich zařazením do množiny P, pokud $i \geqslant 0$, dále pak zařazením $|j|$ cyklů typu $\langle a_1, b_1, c_1\rangle$ do množiny O, pokud $j < 0$, resp. jejich zařazením do množiny P, pokud $j \geqslant 0$, a zařazením $|k|$ cyklů typu $\langle a_2, b_2, c_2\rangle$ do množiny O, pokud $k < 0$, resp. jejich zařazením do množiny P, pokud $k \geqslant 0$, získáme množiny cyklů O a P takové, že $\sum_{y\in P} \langle a, b, c\rangle_y - \sum_{y\in O} \langle a, b, c\rangle_y = \left\langle \left(6s^3\right)!, \left(6s^3\right)!, \left(6s^3\right)!\right\rangle$. Takovéto množiny jsme chtěli nalézt.

b) $\langle a_0, b_0, c_0\rangle$, $\langle a_1, b_1, c_1\rangle$ a $\langle a_2, b_2, c_2\rangle$ jsou lineárně závislé. Podle předpokladů jsou však $\langle a_0, b_0, c_0\rangle$ a $\langle a_1, b_1, c_1\rangle$ lineárně nezávislé.

Hledejme řešení soustavy:

$$\begin{aligned} ia_0 &\ +ja_1 &= ka_2 \\ ib_0 &\ +jb_1 &= kb_2 \\ ic_0 &\ +jc_1 &= kc_2, \end{aligned}$$

kde k je celé číslo.

Z lineární nezávislosti $\langle a_0, b_0, c_0\rangle$ a $\langle a_1, b_1, c_1\rangle$ vyplývá, že alespoň jedna dvojice vektorů $\langle a_0, a_1\rangle$ a $\langle b_0, b_1\rangle$ nebo $\langle a_0, a_1\rangle$ a $\langle c_0, c_1\rangle$ nebo $\langle b_0, b_1\rangle$ a $\langle c_0, c_1\rangle$ je lineárně nezávislá. Bez újmy na obecnosti předpokládejme, že $\langle a_0, a_1\rangle$ a $\langle b_0, b_1\rangle$ jsou lineárně nezávislé. (V ostatních případech by se postupovalo obdobně.)

Tedy $a_0b_1 - a_1b_0 \neq 0$ a platí:

$$i = \frac{ka_2b_1 - ka_1b_2}{a_0b_1 - a_1b_0} \qquad\qquad j = \frac{ka_0b_2 - ka_2b_0}{a_0b_1 - a_1b_0}.$$

Pokud položíme $k = a_0b_1 - a_1b_0$, pak $i = a_2b_1 - a_1b_2$ a $j = a_0b_2 - a_2b_0$ jsou celá čísla. Zřejmě $|i|, |j|, |k| \leqslant 2s^2 < n^2$. Dále platí:

$$\begin{aligned} ia_0 &\ +ja_1 &\ -ka_2 &= 0 \\ ib_0 &\ +jb_1 &\ -kb_2 &= 0 \\ ic_0 &\ +jc_1 &\ -kc_2 &= 0. \end{aligned}$$

To je ovšem spor se závěrečnou poznámkou Části III důkazu. Případ, kdy by $\langle a_0, b_0, c_0\rangle$,

$\langle a_1, b_1, c_1\rangle$ a $\langle a_2, b_2, c_2\rangle$ byly lineárně závislé, tedy nemůže nastat.

Tím je důkaz Lemmatu 3.4.2 hotov. $\square$

Lemma 3.4.8 *Nechť $k \geqslant 4, h \geqslant 1$ jsou přirozená čísla. Pak existuje (k,h)-DFA automat M přijímající jazyk $L^{k,h}$.*

DŮKAZ: Automat M bude pracovat následovně. Postupně ověří rovnost jednotlivých kopií slov x_i, z_i, $u_{i,j}$, $v_{i,j}$ a $w_{i,j}$ a dále též pro všechna i ověří splnění podmínky $a_i = 1 \iff \exists m \in \mathbb{N},\ \overline{a}_i = 0^{m^2}$.

Pokud během porovnávání narazí na rozdíl mezi kopiemi slov nebo nesplnění podmínky pro a_i, přesune všechny hlavy na konec vstupního slova a slovo odmítne. Jinak postupně prověří všechny dvojice kopií slov a splnění podmínky na a_i a slovo přijme .

Prověřování bude provádět tak, že nejprve pomocí dvou vybraných hlav (hlavy h_1 a h_2) zkontroluje kopie slov x_i a zároveň pomocí kamenů označí začátky slov z_i v oblasti $(x_i z_i \#)_1^h$. Poté pomocí hlav $h_1, \ldots, h_{k-1}$ provede porovnání kopií slov z oblasti α_1 a slov $w_{2,1}$ až $w_{2,3k-9}$. Následně pomocí hlav h_1, h_2, h_3 a h_k provede porovnání kopií slov $u_{i,j}$ a $v_{i,j}$ a ověření podmínky pro a_i, pak pomocí hlavy h_k a h_1 porovná kopie slov z_i a nakonec dokončí porovnávání zbylých dvojic slov $w_{2,3k-8}$ až $w_{2,\binom{k}{2}-3}$ v oblasti α_2 pomocí hlav $h_4, \ldots, h_k$.

Porovnání dvojic slov x_i se provede tak, že se hlava h_1 přesune na první znak oblasti $(x_i z_i \#)_1^h$, jejíž začátek se pozná podle toho, že jí bezprostředně předchází symbol ¢. Hlava h_1 tedy bude na prvním znaku pravé kopie slova x_1. Poté se pomocí hlavy h_1 a h_2 porovnávají kopie slov x_i a vždy po dokončení porovnání kopií jednoho slova x_i (hlava h_2 dorazí na znak # oddělující levé kopie slov x_i a x_{i+1}) položí hlava h_1 kámen na začátek levé kopie slova z_i. Kamenů je k dispozici právě h, čili tato fáze výpočtu může bez problémů proběhnout.

Nyní se $k-1$ hlav $h_1, \ldots, h_{k-1}$ přesune na začátek oblasti α_1, kde provedou porovnání slov $w_{1,j}$ pro všechna j. Začátek této oblasti poznají hlavy $h_2, \ldots, h_{k-1}$ tak, že od začátku vstupního slova dojedou na nejbližší znak ¢ a dále pak odpočítají příslušný počet znaků #. Vlastní porovnání slov $w_{1,j}$ se provede standardním způsobem (viz Lemma 3.1.4.)

Pak se hlavy $h_1, \ldots, h_{k-1}$ přesunou do oblasti α_2. Tam se hlavy $h_2, \ldots, h_{k-1}$ rozmístí na začátky levých kopií slov $w_{2,1}$ až $w_{2,k-2}$ (odpočítá se příslušný počet znaků #), a pomocí hlavy h_1 procházející pravé kopie slov $w_{2,k-2}$ až $w_{2,1}$ se provede porovnání těchto slov. Poté se hlavy $h_3, \dots, h_{k-1}$ umístí na začátky levých kopií slov $w_{2,k-1}$ až $w_{2,2k-4}$ a pomocí hlavy h_2 procházející pravé kopie těchto slov se provede porovnání kopií slov $w_{2,k-1}$ až $w_{2,2k-5}$. Pak se hlavy $h_4, \dots, h_{k-1}$ umístí na začátky levých kopií slov $w_{2,2k-4}$ až $w_{2,3k-9}$ a pomocí hlavy h_3 procházející pravé kopie těchto slov se provede porovnání kopií slov $w_{2,2k-4}$ až $w_{2,3k-9}$.

Nyní se pomocí hlavy h_k a hlav h_1, h_2, h_3 budou porovnávat slova $u_{i,j}$ a $v_{i,j}$ a bude se ověřovat platnost vztahu pro a_i. Porovnání slov $u_{i,1}, u_{i,2}, u_{i,3}$ pro dané i se vždy provede tak, že se hlavy h_1, h_2, h_3 rozestaví do pravých kopií slov $u_{i,1}, u_{i,2}, u_{i,3}$ a pomocí hlavy h_k procházející levé kopie slov $u_{i,1}, u_{i,2}, u_{i,3}$ se porovnají kopie těchto slov. Následně se stejným způsobem porovnají kopie slov $v_{i,1}, v_{i,2}, v_{i,3}$. Pak hlavy h_1, h_2, h_3 zjistí, zda $\overline{a}_i$ je, nebo není slovem délky druhé mocniny přirozeného čísla a pomocí hlavy h_k se ověří, že zjištěná skutečnost pro slovo $\overline{a}_i$ odpovídá znaku a_i. Po tomto ověření se začne provádět lustrování slov $u_{i,1}, u_{i,2}, u_{i,3}$ pro další i. Tento proces bude ukončen poté, co hlavy h_1, h_2, h_3 a h_k dorazí na znak $\cent$ signalizující, že zde již nejsou další slova $u_{i,j}$.

Zjištění, zda slovo $\overline{a}_i$, resp. jeho délka, je, nebo není druhou mocninou přirozeného čísla, se provede následujícím způsobem. Hlava h_1 a h_2 se postaví na první znak tohoto slova a hlava h_3 se postaví na znak bezprostředně předcházející toto slovo (znak $\#$). (Pokud je $\overline{a}_i = \epsilon$, tj. následují dva znaky $\#$ bezprostředně za sebou, pak slovo $\overline{a}_i$ je rovno druhé mocnině a dále není pro toto slovo potřeba nic zkoumat.) Nyní se bude s hlavami neustále opakovat následující postup, dokud hlava h_2 nedorazí na znak $\#$ ukončující slovo $\overline{a}_i$. Během postupu bude na začátku m-tého opakování vždy platit, že hlavy h_1 a h_2 jsou ve slově $\overline{a}_i$ na pozici m^2 a hlava h_3 je $2m+1$ pozic za hlavou h_1 a h_2. Postup jejich pohybu pak bude tento: pohybuj simultáně hlavou h_2 a h_3 dokud hlava h_3 nedorazí na stejou pozici jako je hlava h_1. Pak hlavu h_2 posuň o další dvě pozice. Poté hlavu h_1 přesuň na stejnou pozici, jako je nyní hlava h_2. Tím je jeden cyklus ukončen.

Zřejmě rozmístění hlav na začátku dalšího opakování bude splňovat uvedené podmínky, neboť vzdálenost hlav h_2 a h_3 se zvětšila o dvě. To, že hlavy h_1 a h_2 budou po dokončení m-tého cyklu na pozici $(m+1)^2$, vyplývá z toho, že $(m+1)^2 = m^2 + 2m + 1$. Zda je slovo $\overline{a}_i$ mocninou dvou, nebo není se pozná podle toho, zda hlava h_2 při simultáním pohybu s hlavou h_3 prvním krokem vstoupí na znak $\#$. Pokud ano, je to mocnina dvou. Pokud na něj vstoupí až v dalších krocích simultáního pohybu nebo posunu o dvě, není slovo $\overline{a}_i$ mocninou dvou. Jakmile na tento znak hlava h_2 vstoupí, ukončí se testování slova $\overline{a}_i$ a hlavy h_1, h_2, h_3 se přesunou na začátek následujícího slova $u_{i+1,j}$.

Po prověření všech slov $u_{i,j}, v_{i,j}$ a a_i se hlava h_k dostane na první znak $\cent$ a hlavy h_1, h_2, h_3 se dostanou na druhý znak $\cent$. Nyní hlava h_k spolu s hlavou h_1 bude porovnávat kopie slov z_i. Hlavu h_k vždy přesuneme na nejbližší kámen označující začátek dalšího slova z_i a hlavu h_1 postupně posouváme po jednotlivých slovech z_i. Jejich simultáním pohybem ověřujeme shodnost těchto slov.

Nakonec hlavou h_1 dojedeme do oblasti α_2 (odpočítá se příslušný počet znaku $\#$) a ve spolupráci s hlavami $h_4, \ldots, h_{k-1}$ se provede porovnání kopií zbylých slov $w_{2,3k-8}$ až $w_{2,\binom{k}{2}-3}$.

Tím bude ověřování hotové a všechny hlavy se přesunou na koncový symbol \$ a vstupní slovo se přijme. Pokud by se během porovnávání narazilo na rozdíl oproti definici jazyka $L^{k,h}$, hlavy by se přesunuly na koncový symbol \$ a slovo by se odmítlo.

Takto tedy automat rozpoznává jazyk $L^{k,h}$. $\square$

Lemma 3.4.9 *Nechť $h \geqslant 1$ je přirozené číslo. Pak $R^N_{2,h} = R_{2,h}$.*

DŮKAZ: Vezměme si přirozené číslo $h \geqslant 1$ a $(2,h)$-FA automat M s koincidencí. Sestrojíme $(2,h)$-FA automat M' bez koincidence rozpoznávající stejný jazyk jako automat M.

Upravme nejprve automat M tak, aby nad vstupním slovem nezanechával za poslední hlavou ležet žádné kameny (M je s koincidencí, tedy tuto konstrukci lze podle druhé kapitoly provést.)

Snadno nahlédneme, že pokud je během výpočtu nějaký kámen položený, pak leží v úseku vstupního slova mezi hlavami upraveného automatu M. Žádný kámen nemůže zůstat ležet za v pořadí druhou hlavou automatu a stejně tak nemůže ležet žádný kámen před první hlavou. Mimo to, každý kámen leží na jiném políčku.

Pokud jsou tedy v daný okamžik výpočtu položeny alespoň dva kameny, pak nutně hlavy musí být na různých políčkách. Pokud je položen kámen jeden a obě hlavy kámen vidí, pak jsou nad stejným políčkem. Pokud alespoň jedna z hlav položený kámen nevidí, jsou na různých políčkách. (Kámen může být mezi hlavami nebo pod jednou z nich.)

Pokud není položen žádný kámen, pak hlavy mohou být, nebo nemusí nad stejným políčkem. Jelikož však $h \geqslant 1$, tak lze pomocí jednoho z kamenů otestovat, zda hlavy jsou nad stejným políčkem. První hlava kámen položí, druhá se podívá, zda kámen vidí, a poté první hlava opět kámen zvedne. Pokud druhá hlava kámen na svém políčku viděla, pak jsou obě hlavy na stejném políčku, jinak jsou na různých políčkách.

Automat M' tedy snadno získáme tak, že upravenému automatu M přidáme počitadlo, které bude během výpočtu udávat počet aktuálně položených kamenů. Automat M' pak bude simulovat činnost upraveného automatu M, přičemž pro rozhodování, zda hlavy jsou na stejném políčku, bude používat výše uvedené testy, které nevyžadují přímý test koincidence hlav.

Zřejmě tedy automat M' bude bez koincidence a bude rozpoznávat stejný jazyk jako automat M.

Tím je lemma dokázáno. □

Věta 3.4.10 (O koincidenci) *Nechť h je přirozené číslo. Pak pro třídy jazyků $R_{k,h}$ platí tyto vztahy:*

$$\begin{aligned} & R^N_{1,h} = R_{1,h} = R^{ND}_{1,h} = R^D_{1,h}, && (3.9)\\ h \geqslant 1 \implies & R^N_{2,h} = R^N_{2,h}, \quad R^{ND}_{2,h} = R^{ND}_{2,h}, && (3.10)\\ & R^N_{3,0} \subsetneq R_{3,0}, \quad R^{ND}_{3,0} \subsetneq R^D_{3,0}, \quad R^D_{3,0} - R^N_{3,0} \neq \emptyset, && (3.11)\\ k \geqslant 4 \implies & R^N_{k,h} \subsetneq R_{k,h}, \quad R^{ND}_{k,h} \subsetneq R^D_{k,h}, \quad R^D_{k,h} - R^N_{k,h} \neq \emptyset. && (3.12) \end{aligned}$$

Důkaz: Rovnosti (3.9) jsou triviální. Rovnosti (3.10) plynou z předchozího Lemmatu 3.4.9.

Inkluze (3.11) jsou obecně známé a pro jejich důkaz se používá jazyk $\left\{0^{n^2} \mid n \in \mathbb{N}\right\}$, který lze rozpoznávat deterministickým tříhlavým konečným automat s konincidencí, ale žádným vícehlavým konečným automatem bez kamenů a bez koincidence (blíže Harrison a Ibarra [6] a Sudborough [15].)

Zbylé inkluze a nerovnost (3.12) plynou z Lemmatu 3.4.8 a dále z Vět 3.4.3 a 3.4.4, přičemž pro $h = 0$ jsou opět obecně známy. □

Zůstává otevřená otázka, zda $R^N_{2,0} \subsetneq R_{2,0}$ a zda $R^N_{3,h} \subsetneq R_{3,h}$ pro $h \geqslant 1$. Dá se však předpokládat, že obě tyto ostré inkluze skutečně platí. Jedním z jazyků, který by mohl posloužit k důkazů první z inkluzí je jazyk $L = \{xy \mid x, y \in \{0,1\}^+$ & počet jedniček ve slově x je roven $|x|/2$ a počet jedniček ve slově y je roven $|y|/2\}$ navržený V. Koubkem. Tento jazyk by pak dále bylo nejspíše možné použít i pro důkaz druhé z inkluzí, kdy bychom upravili Definici 3.4.1

jazyka $L^{k,h}$ pro $k = 3$ a použili pro $\overline{a}_i$ slova z jazyka L.[28]

[28] Uvedený jazyk L však lze rozpoznávat čtyřhlavým nedeterministickým automatem bez koincidence. První dvě hlavy porovnají počet jedniček a nul v celém slově a další dvě hlavy poté simultáním pohybem nedeterministicky určí začátek podslova y a ověří pro něj splnění podmínky na počet jedniček.

Kapitola 4

Uzávěrové vlastnosti tříd $R_{k,h}$

V této kapitole ukážeme vlastnosti tříd $R_{k,h}$, $R^D_{k,h}$, $R^N_{k,h}$ a $R^{ND}_{k,h}$ vzhledem k následujícím operacím nad jazyky: průnik s regulární množinou, doplněk, průnik dvou jazyků, sjednocení, reverzi, konkatenaci a neomezenou iteraci jazyků.

4.1 Uzávěrové vlastnosti

V tomto oddíle ukážeme vlastnosti tříd $R_{k,h}$, $R^D_{k,h}$, $R^N_{k,h}$ a $R^{ND}_{k,h}$ vzhledem k výše uvedeným operacím s jazyky.

Pro $k = 1$ a h přirozené číslo jsou třídy $R_{1,h}$, $R^D_{1,h}$, $R^N_{1,h}$ a $R^{ND}_{1,h}$ třídami regulárních jazyků, tedy všechny tyto třídy jsou uzavřeny na uvedené operace. Pro $k \geqslant 2$ a h přirozená čísla ukážeme, že jednotlivé třídy jsou uzavřeny na následující operace s jazyky.

	$R_{k,h}$	$R^D_{k,h}$	$R^N_{k,h}$	$R^{ND}_{k,h}$
průnik s regulární množinou	$+$	$+$	$+$	$+$
doplněk jazyka	$-$	$+$	$-$	$+$
průnik dvou jazyků	$-$	$-$	$-$	$-$
sjednocení dvou jazyků	$+$	$-$	$+$	$-$
reverze jazyka	$+$	$-$	?	$-$
konkatenace dvou jazyků	$+$	$-$	?	$-$
neomezená iterace	$+$	$-$	?	$-$

Podle Věty 2.2.2 je $R^{ND}_{1,0} = \bigcup_{h \in \mathbb{N}} R_{1,h}$ a tedy platí $R_{1,0} = R_{1,h} = R^{D}_{1,h} = R^{N}_{1,h} = R^{ND}_{1,h}$. Z obecně známých uzávěrových vlastností třídy regulárních jazyků plyne následující tvrzení.

Věta 4.1.1 *Třídy jazyků $R_{1,h}$, $R^{D}_{1,h}$, $R^{N}_{1,h}$ a $R^{ND}_{1,h}$ jsou uzavřeny na doplněk, průnik, sjednocení, reverzi, konkatenaci a neomezenou iteraci jazyků.*

Zabývejme se tedy dále dalšími třídami jazyků $R_{k,h}$.

Nechť $k \geqslant 2, h$ jsou přirozená čísla. Vezměme si libovolný (k,h)-FA automat M_L a libovolný konečný automat M_R (tj. jednohlavý konečný automat bez kamenů). Jazyk rozpoznávaný automatem M_L označme L a jazyk rozpoznávaný automatem M_R označme R. Pak jazyk $L \cap R$ lze rozpoznávat (k,h)-FA automatem. Automat rozpoznávající tento jazyk bude pracovat jako automat M_L, pouze bude navíc při výpočtu pomocí jedné z hlav sledovat stav (simulovat činnost) automatu M_R. Z tohoto vyplývá následující tvrzení:

Věta 4.1.2 *Třídy jazyků $R_{k,h}$, $R^{D}_{k,h}$, $R^{N}_{k,h}$ a $R^{ND}_{k,h}$ jsou uzavřeny na průnik s regulární množinou.*

Jak již bylo uvedeno v druhé kapitole, každý (k,h)-FA automat lze upravit tak, že při výpočtu nad libovolným slovem vždy dojde do koncové konfigurace. U takto upraveného deterministického (k,h)-FA automatu pak pro každé slovo existuje právě jeden výpočet z počáteční konfigurace do koncové konfigurace. Pokud tedy u takového automatu množinu koncových stavů nahradíme jejím doplňkem, získáme tak automat, který přijímá doplněk jazyka přijímaného původním automatem. Dostáváme tak následující tvrzení:

Věta 4.1.3 *Třídy jazyků $R^{D}_{k,h}$ a $R^{ND}_{k,h}$ jsou uzavřeny na doplněk jazyků.*

Podle Lemmatu 3.3.1 nelze jazyk $L_1 = \left\{x\#x^R \mid x \in \{0,1\}^*\right\}$ rozpoznávat žádným (k,h)-FA automatem pro žádné k ani h. Jazyk $L_2 = \left\{x\#y \mid x,y \in \{0,1\}^* \quad \& \quad x \neq y^R\right\}$ lze podle Lemmatu 3.3.2 rozpoznávat nedeterministickým $(2,0)$-FA automatem bez koincidence. Zřejmě $L_1 = \overline{L_2} \cap \{x\#y \mid x,y \in \{0,1\}^*\}$. Pokud by pro některé $k \geqslant 2$ a h přirozené číslo byla některá ze tříd $R_{k,h}$, $R^{N}_{k,h}$ uzavřená na doplněk, pak by podle Věty 4.1.2 jazyk L_1 bylo možné rozpoznávat (k,h)-FA automatem, což možné není. Tedy platí následující tvrzení:

Věta 4.1.4 *Třídy $R_{k,h}$ a $R^{N}_{k,h}$ pro $k \geqslant 2$ nejsou uzavřené na doplněk jazyků.*

Podle Lemmatu 3.1.4 lze jazyky $K_{1,\binom{k}{2},\binom{k}{2}+1}$ a $K_{2,\binom{k}{2}+1,\binom{k}{2}+1}$ z Příkladu 3.1.4 rozpoznávat deterministickými $(k,0)$-FA automaty bez koincidence. Průnik těchto jazyků je jazyk $K_{1,\binom{k}{2}+1,\binom{k}{2}+1}$, který podle uvedeného lemmatu nelze rozpoznávat žádným (k,h)-FA automatem. Tedy platí:

Věta 4.1.5 *Třídy jazyků $R_{k,h}$, $R^D_{k,h}$, $R^N_{k,h}$ a $R^{ND}_{k,h}$ pro $k \geqslant 2$ nejsou uzavřeny na průnik jazyků.*

Jelikož třídy $R^D_{k,h}$ a $R^{ND}_{k,h}$ pro $k \geqslant 2$ jsou uzavřeny na doplněk jazyků a nejsou uzavřeny na průnik jazyků, nemohou být uzavřeny ani na sjednocení jazyků. Platí totiž $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$. Tedy platí následující tvrzení:

Věta 4.1.6 *Třídy jazyků $R^D_{k,h}$ a $R^{ND}_{k,h}$ pro $k \geqslant 2$ nejsou uzavřeny na sjednocení jazyků.*

Pokud máme dva jazyky L_1 a L_2 rozpoznávatelné (k,h)-FA automaty pro nějaké k a h, pak lze snadno sestrojit nedeterministický (k,h)-FA automat, jenž rozpoznává jazyk $L_1 \cup L_2$. Takovýto automat bude obsahovat kopii automatu pro jazyk L_1 a kopii automatu pro jazyk L_2. V počátečním stavu se při výpočtu nad každým slovem nedeterministicky rozhodne, který z těchto automatů použije pro zbytek výpočtu, tedy buď použije automat pro jazyk L_1, nebo automat pro jazyk L_2. Slovo přijme, pokud ho přijme spuštěný automat. Zřejmě takovýto automat bude přijímat právě slova z $L_1 \cup L_2$. Tedy platí následující tvrzení:

Věta 4.1.7 *Třídy jazyků $R_{k,h}$ a $R^N_{k,h}$ jsou uzavřeny na sjednocení jazyků.*

Pokud L_1 a L_2 jsou jazyky rozpoznávatelné (k,h)-FA automaty pro nějaké k a h, pak lze sestrojit nedeterministický (k,h)-FA automat, jenž rozpoznává konkatenaci jazyků L_1L_2. Podle úvah ve druhé kapitole existují (k,h)-FA automaty M_1 a M_2 rozpoznávající jazyky L_1 a L_2 takové, že si evidují pořadí hlav nad vstupním slovem a za poslední hlavou nezůstává položen žádný kámen ("poslední hlava kameny sbírá").

Automat pro jazyk L_1L_2 bude pracovat tak, že nejprve nechá běžet výpočet podle automatu M_1. V určitou chvíli při posunu první hlavy na další políčko se automat nedeterministicky rozhodne, že tato hlava vkročila za konec slova, jež by mělo být z jazyka L_1. Od toho okamžiku se výpočet chová, jako by tato první hlava byla nad znakem \$. Výpočet probíhá dále podle automatu M_1 s tím, že kdykoliv některá z hlav vstoupí na stejné políčko jako je uvedená první hlava, začne se i tato hlava chovat, jako by pod ní byl znak \$. Takto probíhá výpočet, dokud nejsou všechny hlavy nad stejným políčkem jako první hlava myslící si, že stojí nad znakem \$. Pokud v okamžiku, kdy na toto políčko vstoupí poslední hlava je automat v jednom z koncových stavů, bude výpočet pokračovat dál, neboť zvolený počáteční úsek vstupního slova patří do jazyka L_1. Jinak bude vstupní slovo rovnou odmítnuto.

Pokud bude výpočet pokračovat dále, tedy automat byl v koncovém stavu, tak bude pokračovat podle automatu M_2. Pokud tento automat zbytek slova přijme, pak náš automat přijme celé vstupní slovo. Jinak ho odmítne. Všechny hlavy jsou totiž nad prvním znakem zbytku slova, o kterém chce automat ověřit, že patří do jazyka L_2. Mimo to, jelikož automat M_1 nezanechává nad vstupním slovem žádné kameny, jsou automatu M_2 k dispozici všechny kameny a výpočet může opravdu probíhat podle automatu M_2.

Zřejmě takovýto automat přijímá ta slova, jež vznikla spojením slova z L_1 a slova z L_2. Dostáváme tak následující tvrzení:

Věta 4.1.8 *Třída jazyků $R_{k,h}$ je uzavřená na konkatenaci jazyků.*

Obdobnou konstrukcí lze ke každému automatu přijímajícímu jazyk L z nějaké třídy $R_{k,h}$ sestrojit automat, jenž přijímá jazyk L^* a má stejný počet hlav a kamenů. Rozdíl v konstrukci bude ten, že se v druhé fázi výpočtu místo automatu M_2 spustí opět ten samý automat. Tedy:

Věta 4.1.9 *Třída jazyků $R_{k,h}$ je uzavřená na neomezenou iteraci jazyků.*

Výše uvedená konstrukce v obecném případě vyžaduje automat s koincidencí hlav. Zjištění, zda hlava dorazila na stejné místo jako první z hlav, by se dalo nahradit nedeterministickým odhadem. Hlavy by se místo tohoto zjišťování jednoduše kdykoliv rozhodly, že právě vstoupily nad symbol \$. Poté, co by se takto rozhodly všechny, by pomocí jednoho z kamenů ověřily, že stojí všechny nad stejným políčkem.

Problém je zde však ten, že u takovéhoto automatu nelze obecně zaručit, že v první části slova nezůstaly ležet nějaké kameny. Tedy pro další část výpočtu by nám některé kameny mohly chybět. Jelikož nevíme, která z hlav je poslední, nelze takové kameny v průběhu výpočtu jednoduše sbírat. Zůstává zde tedy otevřená otázka, zda třídy $R^N_{k,h}$ jsou uzavřeny na konkatenaci a neomezenou iteraci jazyků.

Věta 4.1.10 *Třída jazyků $R_{k,h}$ je uzavřená na reverzi jazyků.*

DŮKAZ: Uvedeme zde opět pouze ideu konstrukce příslušného automatu. Vezměme si pro nějaké k a h (k,h)-FA automat M přijímající jazyk L. Chceme sestrojit automat M^R přijímající jazyk L^R.

Bez újmy na obecnosti můžeme podle úvah ve druhé kapitole předpokládat, že automat M nezanechává na vstupním slově za poslední hlavou ležet žádné kameny (tedy v koncové konfiguraci není položen žádný kámen.)

Automat M^R bude pracovat tak, že se nad vstupním slovem x^R bude odzadu pokoušet uhodnout přijímající výpočet V automatu M pro slovo x. Vyjde tedy jakoby z koncové konfigurace nad slovem x a bude se snažit dostat do počáteční konfigurace nad slovem x. Určité konfigurace výpočtu automatu M^R, mezi kterými bude vždy několik pomocných konfigurací, budou tedy postupně odzadu odpovídat konfiguracím výpočtu V. Odpovídat jim budou v tom smyslu, že rozmístění hlav a kamenů nad slovem x^R bude odpovídat rozmístění hlav a kamenů nad slovem x a stav automatu M^R bude odpovídat některému ze stavů automatu M.

Během výpočtu se tedy automat M^R bude chovat následujícím způsobem. Automat M^R je v nějaké konfiguraci odpovídající konfiguraci automatu M a snaží se uhodnout možnou předcházející konfiguraci automatu M. To udělá takto:

Nedeterministicky uhodne předchozí stav automatu M (tento stav implikuje hlavu, se kterou by se mělo pracovat), dále uhodne symbol pod příslušnou hlavou, zda byl pod touto hlavou kámen a zda s ní byly na stejném políčku další hlavy. Pak uhodne, zda se s hlavou hýbalo a zda tato hlava položila, či zvedla kámen.

Následně ověří, zda uhodnutý stav, symbol, kámen a přítomnost ostatních hlav povolují přechod s uhodnutým pohybem hlavy, položením, či zvednutím kamene a stavem, ve kterém se automat M má nacházet (tento stav odpovídá aktuálnímu stavu automatu M^R, který sleduje stavy automatu M). Pokud uhodnuté skutečnosti nejsou přechodovou funkcí povoleny, vstupní slovo se odmítne. Jinak se pokračuje dále.

Pokud automat hádal, že se mělo hlavou pohnout, pak se s ní pohne na další políčko slova x^R (tj. jakoby na předchozí symbol ve slově x), přičemž kámen na výchozím políčku se nechá beze změny (leží/neleží).

Na políčku, kde se nyní hlava nachází, se ověří, zda tam je kámen, pokud jsme hádali, že ho automat položil, či naopak, že tam kámen není, pokud jsme hádali, že ho zvednul. V případě nesrovnalostí se slovo odmítne.[1]

Pak se ověří, zda souhlasí uhodnutý symbol se symbolem pod hlavou.

Dále pak, pokud automat hádal, že kámen na políčku pod hlavou byl, tak ho tam položí, jinak ho zvedne. Nakonec změní stav simulovaného automatu M na uhodnutý stav.

Tím se automat dostane do konfigurace, jež odpovídá možné předchozí konfiguraci při výpočtu automatu M.

Pokud tedy za výchozí konfiguraci automatu M^R nedeterministicky zvolíme jednu z možných přijímajících konfigurací automatu M a automat M^R se po konečném počtu kroků dostane do konfigurace odpovídající počáteční konfiguraci automatu M, automat M^R slovo přijme. Jinak ho odmítne. (Zřejmě tak bude přijímat právě jazyk L^R.)

Je zde potřeba si všimnout ještě několika drobností. Automat M^R pracuje nad slovem $x^R\$$, kdežto automat M pracuje nad slovem $x\$$.

To, zda konfigurace automatu M^R odpovídá počáteční konfiguraci automatu M, bude automat M^R testovat nedeterministicky. Pokud se dostane do konfigurace, ve které budou všechny hlavy nad stejným políčkem, kameny budou zvednuty a odpovídající stav automatu M by byl počátečním stavem automatu M, automat M^R se rozhodne, zda bude pokračovat dále ve výpočtu, nebo zda ověří, že toto odpovídá počáteční konfiguraci automatu M. K tomu stačí jediné, pohnout jednou z hlav na další políčko a ověřit, že se tam nalézá symbol \$. Pokud uhodl správně, že konfigurace je výchozí, slovo přijme, neboť se z koncové konfigurace automatu M dostal do přijímající a tedy vstupní slovo patří do jazyka L^R.

Další rozdíl je v tom, že v koncové konfiguraci automatu M jsou hlavy umístěny nad symbolem \$, avšak vstupní slovo automatu M^R začíná přímo posledním symbolem slova x. Toto se však vyřeší tak, že automat M^R si na začátek vstupního slova pomyslně znak \$ přidá, neboli u každé hlavy si bude pamatovat, zda se již ze symbolu \$ jakoby pohla. Pokud ne, tak bude pracovat, jako by pod danou hlavou byl stále symbol \$. Jakmile by se pak hlava měla pohnout, tak fakticky zůstane na místě, jelikož se již nalézá nad posledním symbolem slova x, pouze si automat poznamená, že se už pohla a od této chvíle se již tato hlava bude chovat

[1] Ve výpočtu V musela ležící kámen nějaká hlava zvednout a tato hlava jej tedy při zpětném výpočtu již položila

normálně.[2]

Zbývá dořešit to, že automat M^R nesmí nad vstupním slovem zanechávat za poslední hlavou žádné kameny. Takto zanechané kameny by totiž odpovídaly situaci, kdy by při výpočtu automatu M nad vstupním slovem x byly již na začátku některé kameny rozmístěny. Automat M^R tedy při výpočtu sleduje, která z jeho hlav je poslední, a pokud by za sebou zanechala nějaký kámen, tak výpočet ukončí a slovo odmítne. Takovýto výpočet totiž jistě nemůže dospět do počáteční konfigurace.

Tím je konstrukce automatu M^R úplná. $\square$

Stejně jako při konstrukci automatu pro konkatenaci jazyků a neomezenou iteraci není zřejmé, zda tuto konstrukci lze provést v rámci automatů bez koincidence. Opět je zde totiž potřeba zajistit sběr kamenů za poslední hlavou. Tedy otázka, zda třídy $R^N_{k,h}$ jsou uzavřené na reverzi jazyka, zůstává otevřená.

Pro úplnost zde na závěr uvedeme důkaz, že třídy $R^D_{k,h}$ a $R^{ND}_{k,h}$ nejsou uzavřeny na konkatenaci jazyků, neomezenou iteraci a reverzi. Tento důkaz pro vícehlavé konečné automaty pochází od Rosenberga [14]. Jak ukázal Floyd [3], je v uvedeném článku v důkazu příslušnosti jistých jazyků do třídy $R_{k,0} - R_{k-1,0}$, kterých se pro důkaz neuzavřenosti třídy $R^D_{k,0}$ používá, chyba. Pokud však místo těchto jazyků použijeme jazyky $K_{1,\binom{k}{2},\binom{k}{2}+1}$ a $K_{2,\binom{k}{2}+1,\binom{k}{2}+1}$ z Příkladu 3.1.4, lze původní důkaz použít beze změny.

Věta 4.1.11 *Třídy $R^D_{k,h}$ a $R^{ND}_{k,h}$ pro $k \geqslant 2$ nejsou uzavřené na konkatenaci, reverzi, a neomezenou iteraci jazyků.*

DŮKAZ: Vezměme si $k \geqslant 2$ a h přirozené číslo. Označme si jazyk $L_1 = K_{1,\binom{k}{2},\binom{k}{2}+1}$ a $L_2 = K_{2,\binom{k}{2}+1,\binom{k}{2}+1}$ z Příkladu 3.1.4. Podle vlastností těchto jazyků (Lemma 3.1.4) jazyk $L_1 \cap L_2$ nepatří do třídy $R_{k,h}$, a tedy ani do tříd $R^D_{k,h}$ a $R^{ND}_{k,h}$, a podle Lemmatu 4.1.3 tak nepatří do tříd $R^D_{k,h}$, $R^{ND}_{k,h}$ ani jazyk $\overline{L_1} \cup \overline{L_2}$. Vezměme si symbol ¢, který nepatří do abecedy, nad kterou jsou tyto jazyky. Snadno nahlédneme, že jazyk $L_3 = \{¢\}\,\overline{L_1}\,\{¢\} \cup \overline{L_2}$ patří do třídy $R^D_{k,h}$ i $R^{ND}_{k,h}$ (automat se podle prvního znaku deterministicky rozhodne, na příslušnost ke kterému z jazyků $\overline{L_1}$, $\overline{L_2}$ se má slovo zkoumat.)

[2] Situace je zjednodušená tím, že na znak \$ se nesmí pokládat kameny.

Zřejmě jazyk $\{¢, ¢¢\}$ patří do třídy $R^D_{k,h}$ i $R^{ND}_{k,h}$. Ukážeme sporem, že jazyk $\{¢, ¢¢\}\,L_3$ nepatří do třídy $R^D_{k,h}$ ani do třídy $R^{ND}_{k,h}$. (Stačí ukázat, že nepatří do třídy $R^D_{k,h}$)

Pokud by jazyk $\{¢, ¢¢\}\,L_3$ patřil do třídy $R^D_{k,h}$, pak by tam podle Lemmatu 4.1.2 patřil i jazyk $L_4 = \{¢, ¢¢\}\,L_3 \cap \{¢¢\}\,(\{0,1\}^* \cup \{0,1\}^*\,\{¢\})$, což je průnik jazyka z $R^D_{k,h}$ a regulární množiny. Avšak $L_4 = \{¢¢\}\left(\overline{L_1}\,\{¢\} \cup \overline{L_2}\right)$ a pokud by tedy L_4 patřil do třídy $R^D_{k,h}$, pak by zřejmě do této třídy patřil i jazyk $\overline{L_1}\,\{¢\} \cup \overline{L_2}$ a tedy i, jak snadno nahlédneme, jazyk $\overline{L_1} \cup \overline{L_2}$. To by byl spor.

Tedy jazyk $\{¢, ¢¢\}\,L_3$ nepatří do třídy $R^D_{k,h}$ ani $R^{ND}_{k,h}$ a tyto třídy tedy nejsou uzavřeny na konkatenaci jazyků.

Jelikož $L_1 = L_1^R$ a $L_2 = L_2^R$ je i $\overline{L_1} = \overline{L_1}^R$ a $\overline{L_2} = \overline{L_2}^R$, a tedy $L_3 = L_3^R$. Snadno nahlédneme, že $L_3\,\{¢, ¢¢\}$ patří do třídy $R^D_{k,h}$ i $R^{ND}_{k,h}$. Jazyk $\{¢, ¢¢\}\,L_3 = (L_3\,\{¢, ¢¢\})^R$ tam však nepatří. Tedy třídy $R^D_{k,h}$ a $R^{ND}_{k,h}$ nejsou uzavřeny na reverzi jazyků.

Zbývá ukázat, že tyto třídy nejsou uzavřeny na neomezenou iteraci. Vezměme si jazyk $L_5 = \left(\{¢\}\,\overline{L_1} \cup \overline{L_2}\right)\{¢\} \cup \{¢\}$. Tento jazyk zřejmě patří do třídy $R^D_{k,h}$ i $R^{ND}_{k,h}$. Sporem ukážeme, že jazyk L_5^* nepatří do těchto tříd.

Pokud by L_5 patřil do $R^D_{k,h}$, pak tam by podle Lemmatu 4.1.2 patřil i jazyk $L_6 = L_5^* \cap \{¢\}\,\{0,1\}^*\,\{¢\}$. Jelikož však $L_6 = \{¢\}\left(\overline{L_1} \cup \overline{L_2}\right)\{¢\}$, tak by do třídy $R^D_{k,h}$ patřil i jazyk $\left(\overline{L_1} \cup \overline{L_2}\right)$. To by však byl spor.

Třídy $R^D_{k,h}$ a $R^{ND}_{k,h}$ tedy nejsou uzavřeny ani na neomezenou iteraci. $\square$

Kapitola 5

Závěr

5.1 Otevřené problémy

V oblasti hierarchie vícehlavých konečných automatů s kameny a jejich uzávěrových vlastností zbývají následující otevřené otázky.

1. Nahrazování kamenů hlavami.

Není znám minimální počet hlav nutný pro nahrazení d chybějících kamenů. Ve Větě 3.2.5 ukazujeme, že pro $h \geqslant h'$ platí $R_{k,h} \subseteq R_{k+\left\lfloor \frac{k}{2} \right\rfloor (h-h'),h'}$, neboli že d chybějících kamenů lze nahradit $\left\lfloor \frac{k}{2} \right\rfloor d$ hlavami. Je otázka, jak vypadají minimální funkce $f, f^D : \mathbb{N}^3 \to \mathbb{N}$ takové, že pro $h \geqslant h'$ je $R_{k,h} \subseteq R_{k+f(k,h,h'),h'}$ a $R^D_{k,h} \subseteq R^D_{k+f^D(k,h,h'),h'}$.

Z Věty O lustrování se dá předpokládat, že $f^{(D)}(k,h,h') \geqslant h - h'$, přičemž z Věty 3.2.5 víme, že $\left\lfloor \frac{k}{2} \right\rfloor (h-h') \geqslant f^{(D)}(k,h,h')$.

2. $R^N_{3,h} \subsetneq R_{3,h}$ a $R^N_{2,0} \subsetneq R_{2,0}$?

Ve Větě O koincidenci jsme vyřešili vztah mezi třídami $R_{k,h}$ a $R^N_{k,h}$ pro $k \geqslant 4$. Pro $k = 2$ a $h \geqslant 1$ víme, že $R^N_{2,h} = R_{2,h}$, a také víme, že $R^N_{3,0} \subsetneq R_{3,0}$. Neznáme však vztah mezi třídami $R_{3,h}$ a $R^N_{3,h}$ pro $h \geqslant 1$ a třídami $R_{2,0}$ a $R^N_{2,0}$.

Dá se však předpokládat, že mezi těmito třídami platí ostré inkluze (neostrá mezi nimi platí triviálně). Jedním z jazyků, který by mohl posloužit k důkazů ostrosti druhé z inkluzí je jazyk $L = \left\{xy \middle| \, x,y \in \{0,1\}^+ \right.$ & počet jedniček ve slově x je roven $|x|/2$ a počet jedniček ve slově y je roven $|y|/2\}$ (V. Koubek). Tento jazyk by pak dále bylo nejspíše možné použít i pro důkaz ostrosti první z inkluzí, kdy bychom modifikovali důkaz Věty 3.4.3 a využili v něm pro $k = 3$ upravený jazyk $L^{k,h}$ z Definice 3.4.1, ve kterém bychom pro podslova $\overline{a}_i$ používali slova ze zmíněného jazyka L.

3. Třída $R^N_{k,h}$ je uzavřená na reverzi jazyků, konkatenaci jazyků a neomezenou iteraci jazyků?

Není známo zda třída $R^N_{k,h}$ je uzavřená na uvedené operace. Důkaz uzavřenosti třídy $R_{k,h}$ na tyto operace využíval koincidence hlav pro sbírání kamenů z pásky za poslední hlavou. Není však zřejmé, zda je toto sbírání kamenů nutné dělat skutečně vždy, nebo zda je obecně možné se bez něj obejít.

5.2 Poděkování

Především bych zde rád poděkoval mému vedoucímu diplomové práce, doktoru Václavu Koubkovi, za mnohé podnětné nápady a připomínky a za pečlivé čtení všech mých rukopisů.

Rád bych zde též poděkoval všem, kteří mě během mého studia, a to nejen na vysoké škole, podporovali, tedy zejména mým rodičům a sourozencům, a také všem mým přátelům a kamarádům.

Literatura

[1] DEMUTH, O., KRYL, R., KUČERA, A., Teorie algoritmů, SPN Praha (1984), 93-95.

[2] FEYNMAN, R.P., LEIGHTON, R., Surely you're joking, Mr. Feynman!, W. W. Norton and Company, (1985), (český překlad – Mladá fronta, 1989).

[3] FLOYD, R.W., Review 14, 352 (of paper by Rosenberg [14]), *Computing Rev.* 9/5 (1968), 280.

[4] GINSBURG, S., AFL with the semilinear property, *J. of Comput. and System Sci.* 5 (1971), 365-396.

[5] GORALČÍK, P., GORALČÍKOVÁ, A., KOUBEK, V., Alternation with a Pebble, *Inform. Proc. Letters* 38 (1991), 7-13.

[6] HARRISON, M. A., IBARRA, O. H., Multi-tape and multi-head pushdown automata, *Inform. and Control* 13 (1968), 433-470.

[7] HOPCROFT, J. E., ULLMAN, J. D., Formálne jazyky a automaty, Alfa Bratislava (1978).

[8] JAGLOM, A. M., JAGLOM, I. M., Pravděpodobnost a informace, Academia Praha (1964).

[9] JIANG, T., LI, M., k one-way heads cannot do String-matching, *Proc. of 25 ACM Symposium on Theory of Comp.* (1993), 62-70.

[10] LI, M., VITÁNYI, P., An introduction to Kolmogorov complexity and its applications, Springer-Verlag New York (1997).

[11] PIATKOWSKI, T. F., N-head finite state machines, *Ph.D. Th.*, U. of Michigan, Ann Arbor, Mich. (1963).

[12] RITCHIE, R. W., SPRINGSTEEL, F. N., Language recognition by marking automata, *Inform. and Control* 20 (1972), 313-330.

[13] RIVEST, R. L., YAO, A. C., $k+1$ heads are better than k, *Journal of ACM* 25/2 (1978), 337-340.

[14] ROSENBERG, A. L., On multi-head finite automata, *IBM J. Res. and Develop. 10* (1966), 388-394.

[15] SUDBOROUGH, I. H., Bounded reversal multihead finite automata languages, *Inform. and Control* 25 (1974), 317-328.

[16] SUDBOROUGH, I. H., One-way multihead writing finite automata, *Inform. and Control* 30 (1976), 1-20.