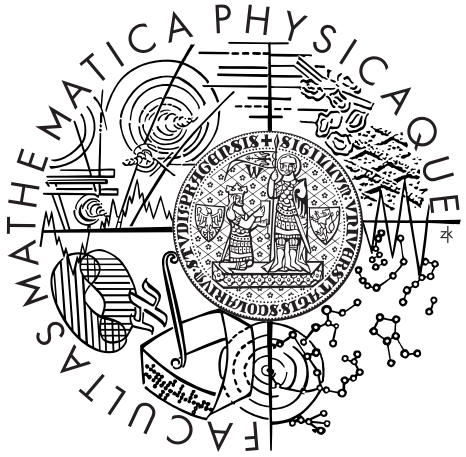
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Matej Juračka

Vztah determinismu a nedeterminismu pro lineární čas

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce:	doc. RNDr. Václav Koubek, DrSc.
Studijní program:	Informatika
Studijní obor:	teoretická informatika

Praha 2010/2011

Ďakujem doc. Václavovi Koubkovi za jeho užitočné rady a pomoc s korekciou textu.

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Vztah determinismu a nedeterminismu pro linární čas

Autor: Matej Juračka

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: doc. RNDr. Václav Koubek, DrSc., Katedra teoretické informatiky a matematické logiky

Abstrakt: Výsledkom práce je rekonštrukcia dôkazu, že triedy jazykov akceptovaných viacpáskovým deterministickým a nedeterministickým strojom v lineárnom čase sú rôzne. Sústredíme sa pritom na úplnosť a zrozumiteľnosť ako samotného dôkazu, tak aj všetkých prípravných tvrdení, ktoré k nemu vedú.

Klíčová slova: vzťah determinizmu a nedeterminizmu, alternujúci Turingov stroj, separácia tried zložitosti.

Title: On determinism and non-determinism for linear time

Author: Matej Juračka

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: doc. RNDr. Václav Koubek, DrSc., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Result of this work is a reconstruction of proof, that non-deterministic linear time is strictly more powerful than deterministic linear time. We focus on completeness and clarity either of proof itself, either of all auxiliary propositions, which lead to this result.

Keywords: relation of determinism and non-determinism, alternating Turing machine, separation of complexity classes.

Obsah

1. Úvod

V článku *On Determinism Versus Non-determinism And Related Problems* [9] podali autori dôkaz tvrdenia, že trieda jazykov akceptovaných deterministickým viacpáskovým Turingovým strojom v lineárnom čase je vlastnou podtriedou jazykov akceptovaných nedeterministickým Turingovým strojom. Tento výsledok je jedným z mála známych, ktoré potvrdzujú predpoklad, že nedeterminizmus môže pridať modelu Turingovho stroja pri obmedzenom čase výpočtovú silu.

Tento dôkaz v texte rekonštruujeme a snažíme sa o jeho zrozumiteľné a prehľadné podanie, držiac sa pritom stratégie prebranej z kapitoly *3. Alternation* knihy *Structural Complexity II* [2]. Na úvod definujeme alternujúci Turingov stroj ako zovšeobecnenie modelu nedeterministického Turingovho stroja, pomocou neho zavedieme niektoré triedy jazykov a preskúmame ich vlastnosti, ktoré v dôkaze využijeme.

Technický aparát, ktorý zavedieme v kapitolách 2 a 3, nám umožní v kapitole 4 sformulovať dôkaz vety o nekonštantnom urýchlení výpočtu deterministického Turingovho stroja s využítím alternácie (Veta 5). Potom už jednoduchou úvahou prídeme k sporu, ktorý vyplynie z predpokladu, že triedy jazykov akceptovaných deterministickým a nedeterministickým Turingovým strojom v lineárnom čase sú totožné. (Veta 6)

V piatej kapitole zhrnieme niektoré výsledky a rozšírenia, ktoré súvisia s dokazovaným tvrdením $DLIN \neq NLIN$.

2. Alternujúci Turingov stroj a jeho vlastnosti

V tejto kapitole uvedieme definície deterministického a nedeterministického Turingovho stroja. Tieto definície sú štandardné a podrobnosti k nim sa dajú nájsť napríklad v práci *Structural Complexity I* [1] . Detailnejšie sa budeme venovať definícii alternujúceho Turingovho stroja a triedam jazykov podľa neho odvodených [2, 3].

2.1 Turingov stroj

Model Turingovho stroja ktorý budeme používať bude bežný viacpáskový Turingov stroj s jednostranne potenciálne nekonečnou páskou. Pred začatím výpočtu bude stroj v začiatočnom stave, na prvej stope bude zapísané vstupné slovo a ostatné pásky (ak existujú) budú prázdne. Počas výpočtu môže hlava na *každej* páske v jednom kroku prečítať symbol nad ktorým sa nachádza a podľa prechodovej funkcie ho zmeniť na iný a posunúť sa na páske o jedno políčko smerom doľava alebo doprava, alebo ostať na pôvodnom mieste.

Definícia 1. *Deterministický Turingov stroj M s k páskami je definovaný ako*

$$M = (Q, \Sigma, \delta, q_0, F)$$

kde

- Q *je konečná množina stavov*
- Σ *je konečná abeceda znakov na páske*
- $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{r, l, 0\}^k$ *je prechodovou funkciou stroja M, ktorá nemusí byť definovaná na celom definičnom obore*
- $q_0 \in Q$ *je začiatočný stav stroja*
- $F \subseteq Q$ *je množinou akceptujúcich stavov*

Výpočet stroja M na vstupe $w \in \Sigma^*$ prebieha deterministicky podľa prechodovej funkcie δ. Na začiatku výpočtu sú všetky hlavy na ľavom okraji pásky, na vstupnej páske je zapísané slovo w, ostatné pásky sú prázdne a stroj je v stave q_0. Pre prázdny symbol na páskach používame symbol B a nazývame ho *blank*. Formálne definujeme začiatočnú konfiguráciu, krok výpočtu a akceptujúci výpočet.

Konfigurácia k-páskového Turingovho stroja pozostáva z aktuálneho stavu stroja, obsahu pások a pozície hláv, formálne je prvkom množiny $Q \times (\Sigma^* \times \mathbb{Z})^k$.

Začiatočná konfigurácia stroja M na vstupe w je

$$\alpha_0 = (q_0, (w, 0), \underbrace{(B, 0), \ldots, (B, 0)}_{k-1})$$

čo znamená, že stroj je v stave q_0, hlavy na páskach sú na začiatku, na prvej páske je zapísané vstupné slovo a ostatné pásky sú prázdne.

Krokom výpočtu nazývame dvojicu konfigurácií $\alpha_1 \vdash \alpha_2$ takú, že konfigurácia α_2 vyplýva z konfigurácie α_1 podľa prechodovej funkcie δ Turingovho stroja.

Výpočtom (alebo cestou výpočtu) Turingovho stroja M na vstupe w rozumieme postupnosť konfigurácií $\alpha_0 \vdash \alpha_1 \vdash \ldots \vdash \alpha_n$, kde α_0 je začiatočnou konfiguráciou pre vstup w.

V prípade, že je prechodová funkcia pre aktuálnu konfiguráciu nedefinovaná, výpočet sa zastaví. V prípade, že stroj sa dostane do akceptujúceho stavu $q \in F$, výpočet sa zastaví a stroj vstup *akceptuje*.

Ku každému DTS M definujeme jazyk akceptovaný strojom M ako množinu

$$L(M) = \{x \in \Sigma^* \mid M \text{ akceptuje } x\}$$

Štandardným spôsobom tiež definujeme nedeterministický Turingov stroj ako stroj, ktorého prechodová funkcia δ je nejednoznačná, t.j. je tvaru

$$\delta : Q \times \Sigma^k \to \mathcal{P}(Q \times \Sigma^k \times \{r, l, 0\}^k)$$

To budeme interpretovať tak, že výpočet stroja sa môže v ľubovoľnom kroku rozdeliť na viac paralelných vetiev výpočtu, pričom stroj vstup akceptuje, ak sa do akceptujúceho stavu dostane v aspoň jednej vetve. Nedeterministické Turingove stroje sú špeciálnym typom alternujúcich Turingových strojov (Σ_1) a preto im pri definícii nebudeme venovať špeciálnu pozornosť.

Definícia 2. *Funkcia $t : \mathbb{N} \to \mathbb{N}$ sa nazýva* časovo konštruovateľnou *práve vtedy, keď existuje deterministický Turingov stroj, ktorý sa na každom vstupe dĺžky n zastaví presne po $t(n)$ krokoch.*

Funkcia $t : \mathbb{N} \to \mathbb{N}$ sa nazýva priestorovo konštruovateľnou *práve vtedy, keď existuje deterministický Turingov stroj, ktorý sa na každom vstupe dĺžky n zastaví, pričom v jeho poslednej konfigurácii sa na páske vyskytuje práve $t(n)$ neprázdnych políčok a počas výpočtu sa nepoužijú žiadne iné.*

Pre ľubovoľnú funkciu $t(n) \leq n + 1$ zavedieme triedy jazykov $DTIME(t(n))$ a $NTIME(t(n))$ nasledovne:

- $L \in DTIME(t(n))$ práve vtedy, keď existuje *deterministický* Turingov stroj M, ktorý sa na každom *akceptovanom* vstupe zastaví za najviac $t(n)$ krokov a $L = L(M)$.
- $L \in NTIME(t(n))$ práve vtedy, keď existuje *nedeterministický* Turingov stroj M, ktorý sa na každom *akceptovanom* vstupe zastaví za najviac $t(n)$ krokov a $L = L(M)$.

Definujeme triedu

$$DLIN = \bigcup_{c>0} DTIME(c.n)$$

a podobne

$$NLIN = \bigcup_{c>0} NTIME(c.n)$$

Keďže nedeterministický Turingov stroj je rozšírením deterministického, triviálne z toho vyplýva, že $DLIN \subseteq NLIN$. Obsahom tejto práce je dôkaz tvrdenia, že opačná inklúzia neplatí a preto $DLIN \neq NLIN$. Predtým, ako naznačíme stratégiu dôkazu tohto tvrdenia, potrebujeme zaviesť nasledovné zovšeobecnenie modelu nedeterministického Turingovho stroja.

2.2 Alternujúci Turingov stroj

Definícia 3. *Alternujúci Turingov stroj (ATS) je pätica* $M = (Q, \Sigma, \delta, q_0, g)$, *kde* Q, Σ, δ *a* q_0 *sú definované rovnakým spôsobom ako pre nedeterministický Turingov stroj a* g *je zobrazenie* $g : Q \to \{\exists, \forall, acc, rej\}$ *určujúce typ každého stavu. Prechodová funkcia* δ *nie je definovaná pre žiadny zo stavov typu* acc, *alebo* rej. *Naopak, pre existenčné* ($\exists$) *a univerzálne* ($\forall$) *stavy je funkcia* δ *definovaná pre každú kombináciu znakov abecedy* Σ *na páskach.*

Konfiguráciu, začiatočnú konfiguráciu a krok výpočtu definujeme rovnako, ako pre deterministické Turingove stroje.

Keďže vo všeobecnosti je prechodová funkcia δ nejednoznačná, má jedna konfigurácia viac možných nasledovníkov. Preto môžeme vytvoriť *strom výpočtu* stroja M pre vstup w tak, že jeho koreň bude tvoriť začiatočná konfigurácia a rekurzívne, ak je α_j vrcholom v strome výpočtu, tak jeho nasledovníkmi budú konfigurácie, do ktorých sa dá dostať z α_j jedným krokom výpočtu. Keďže z existenčných a univerzálnych stavov podľa definície vždy môže výpočet pokračovať, listy tohto stromu budú tvoriť konfigurácie s akceptujúcimi, alebo odmietajúcimi stavmi, v ktorých naopak každý výpočet končí. Rekurzívne môžeme vrchol v strome výpočtu označit ako *akceptujúci*, ak

- je listom stromu a stav danej konfigurácie patrí do množiny akceptujúcich stavov, *alebo*
- je vnútorným vrcholom, jeho príslušný stav je existenčný a *aspoň jeden* z jeho synov je akceptujúci, *alebo*
- je vnútorným vrcholom, jeho príslušný stav je univerzálny a *každý* z jeho synov je akceptujúci.

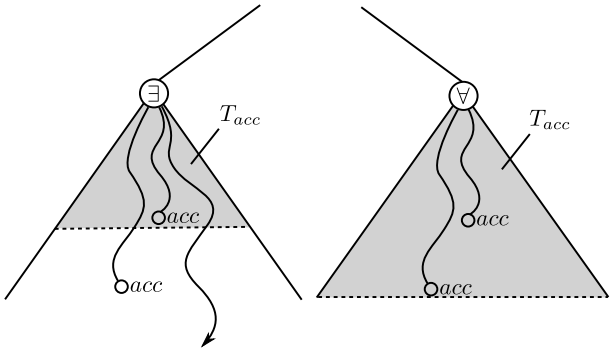
Pre daný alternujúci Turingov stroj M a vstup w vieme zostrojiť potenciálne nekonečný strom výpočtu a ohodnotiť jeho vrcholy ako akceptujúce. Stroj M *akceptuje* vstup w, ak jemu príslušný strom výpočtu má svoj koreň ohodnotený ako akceptujúci.

Aj keď je v niektorých prípadoch strom výpočtu nekonečný, o akceptácii vždy rozhodne jeho konečná časť. Pre dôkaz niektorých tvrdení budeme potrebovať,

aby Turingov stroj ukončil svoj výpočet tak rýchlo, ako je to možné, t.j. v prípade, že sa nachádzame v existenčnom stave, a existuje viac možností, ako rozhodnúť o akceptácii, vyberieme si vždy tú najrýchlejšiu.

Ak je koreň označený ako akceptujúci, zostrojíme jeho *akceptujúci podstrom výpočtu* T_{acc} prerezaním zostrojeného stromu výpočtu nasledovne:

Ak je vrchol existenčný, niektorá cesta z neho je celá akceptujúca. Ak existuje takýchto ciest viac, vyberieme *najkratšiu* možnú a do stromu T_{acc} zaradíme podstrom tohto vrcholu orezaný v hĺbke tejto najkratšej cesty. Ak je vrchol univerzálny, musíme do T_{acc} zobrať celý jeho podstrom, keďže každá jeho cesta akceptáciu ovplyvní. Tento podstrom bude zaručene konečný, pretože každá cesta musí skončiť v akceptujúcom stave.



Obr. 2.1: Akceptujúci podstrom výpočtu pre existenčné a univerzálne vrcholy

Takto vytvoríme pre každý akceptujúci výpočet strom, ktorý je najplytší možný, ale už o akceptácii rozhodne. Pomocou stromu T_{acc} budeme definovať čas výpočtu Turingových strojov a niektoré dôkazy budú závislé na tejto slabej definícii.

Pre daný alternujúci Turingov stroj M definujeme jazyk ním akceptovaný, ako

$$L(M) = \{x \in \Sigma^* \mid M \text{ akceptuje } x\}$$

Vo všeobecnosti alternujúce Turingove stroje akceptujú práve triedu rekurzívne vyčísliteľných jazykov. Zadefinujeme niektoré triedy jazykov akceptované alternujúcimi Turingovými strojmi s nasledovnými obmedzeniami.

Hovoríme, že alternujúci Turingov stroj M je časovo ohraničený funkciou $t : \mathbb{N} \to \mathbb{N}$ (alternatívne M je $t(n)$ časovo ohraničený), ak platí, že pre každé $w \in L(M)$ také, že $|w| = n$, existuje strom výpočtu, ktorého *akceptujúci podstrom* T_{acc} má hĺbku najviac $t(n)$.

Pre každú funkciu $t : \mathbb{N} \to \mathbb{N}$ označme triedu jazykov, ktoré sú akceptované nejakým $t(n)$ časovo ohraničeným k-páskovým alternujúcim Turingovým strojom ako $ATIME_k(t(n))$. V prípade, že počet pások nie je obmedzený, získavame triedu $ATIME(t(n)) = \bigcup_{k=1}^{\infty} ATIME_k(t(n))$.

Ak obmedzíme konštantou $k-1$ počet alternácií medzi existenčnými a univerzálnymi stavmi, ktoré môže stroj počas svojho výpočtu na ľubovoľnom *akceptovanom* vstupe urobiť, hovoríme v prípade, že prvý stav stroja počas výpočtu je existenčný, že stroj je typu Σ_k a v prípade, že prvý stav je univerzálny, že stroj je typu Π_k.

Spojením obmedzenia pre čas aj alternácie dostávame pre každú časovo konštruovateľnú funkciu $t(n)$ a konštantu k triedy jazykov Σ_k-$TIME(t(n))$ a Π_k-$TIME(t(n))$. Jazyk patrí do triedy Σ_k-$TIME(t(n))$, ak existuje alternujúci Turingov stroj typu Σ_k, ktorý patrí do triedy $ATIME(t(n))$. Analogicky definujeme triedu Π_k-$TIME(t(n))$. Stroj typu Σ_1 je podľa tejto definície stroj, ktorý má všetky stavy v priebehu výpočtu existenčné a neurobí žiadnu alternáciu, preto je to klasický nedeterministický Turingov stroj a platí Σ_1-$TIME(t(n)) = NTIME(t(n))$.

Ďalej dokázaná veta 1 hovorí, že pridanie pások do alternujúceho Turingovho stroja nám nepomôže urýchliť výpočet. Dôkaz prebieha skonštruovaním jednopáskového stroja, ktorý akceptuje rovnaký jazyk, ako pôvodný k-páskový [10].

V dôkaze budeme potrebovať využiť to, že alternujúci Turingov stroj dokáže overiť rovnosť dvoch od seba vzdialených reťazcov rýchlo, t.j. v čase $O(n)$ [10, 2]. Toto tvrdenie na úvod dokážeme ako lemmu.

Lemma 1. *Pre ľubovoľnú konečnú abecedu Σ a oddeľovač $\# \notin \Sigma$ je jazyk*

$$L = \{x\#y\#x \mid x, y \in \Sigma^*\}$$

rozpoznávaný nejakým jednopáskovým alternujúcim Turingovým strojom v čase $O(n)$, kde $n = 2|x| + |y| + 2$.

Dôkaz. Lemmu dokážeme konštrukciou alternujúceho Turingovho stroja s jednou páskou. Na vstupe máme slovo w dĺžky $|w| = n$. Na jeho koniec pripíšeme oddelovač $\#$ a overíme nasledujúce podmienky :

1. w je tvaru $x\#y\#z\#$, kde $x, y, z \in \Sigma^*$
2. $x\# = z\#$

Prvá podmienka sa overí deterministicky v čase $O(n)$. Druhú podmienku overíme algoritmom popísaným v zvyšnej časti dôkazu. Keďže $\#$ sa v slovách x, y, ani z nenachádza, $x = z$ platí práve vtedy, keď je $x\#$ prefixom $z\#$. V pokračovaní dôkazu pre prehľadnosť $x\#$ stotožníme s x a tiež $z\#$ s z.

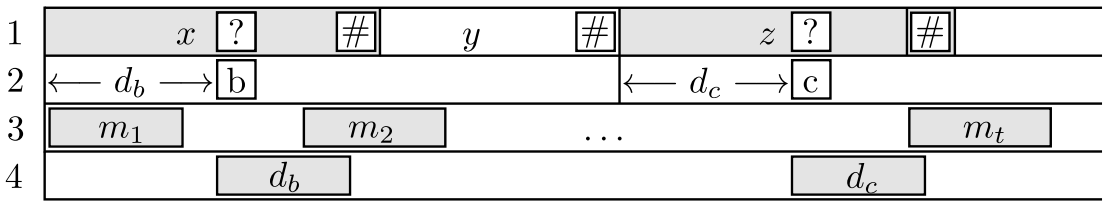
Na overenie toho, či je x prefixom z, využijeme možnosti, ktoré nám dáva alternácia: x je prefixom z práve vtedy, keď pre každú pozíciu písmena v slove x existuje nejaká pozícia v slove z, na ktorej je rovnaký symbol a ktorá je od začiatku slova vzdialená rovnakú dĺžku.

Univerzálne vyberieme niektorý znak v slove x a na stopu 2 pod neho zapíšeme marker b. Potom existenčne vyberieme kandidáta na rovnaký znak pod slovom z, na stopu 2 na toto miesto zapíšeme marker c. To, či sú nad týmito markermi rovnaké znaky, overíme deterministicky v čase $O(n)$. Potrebujeme ukázať, že tieto znaky sú od začiatkov slov vzdialené rovnakú vzdialenosť.

Rýchle overenie rovnosti krátkych slov

> Alternujúcim Turingovým strojom s jednou páskou vieme overiť v čase $O(n)$ rovnosť dvoch krátkych slov d_1 a d_2, $|d_i| \in O(\log n)$ vzdialených od seba najviac n políčok. Na jednej stope existenčne uhádneme slová $n_1, n_2, \ldots n_t$, pričom každé zo slov bude dĺžky najviac $O(\log n)$ a ich vzájomná vzdialenosť bude $O(n/\log n)$. Tieto slová budú od seba oddelené aspoň jedným prázdnym znakom. Univerzálne vyberieme n_i, $i \in \{2, \ldots, t\}$ a overíme, či $n_i = n_{i-1}$. Dĺžka každého z n_i je vo vhodných vetvách výpočtu $O(\log n)$, ich vzájomná vzdialenosť $O(n/\log n)$ a preto stačí testovať deterministicky znak po znaku, v celkovom čase $O(\log n).O(n/\log n) = O(n)$. Ak tento test zlyhá, vstup odmietneme. Potom už stačí iba deterministicky porovnať d_1 s najbližším n_i a rovnako d_2 s jemu najbližším $n_{i'}$. Máme overené, že $n_i = n_{i'}$ a preto tento test rozhodne, či $d_1 = n_i = n_{i'} = d_2$. Tento princíp tiež môžeme využiť na *prenos* krátkych ($O(\log n)$) slov na dlhé ($O(n)$) vzdialenosti – na ľubovoľnom mieste existenčne uhádneme kandidáta na prenášané slovo a pomocou popísaného mechanizmu overíme ich vzájomnú rovnosť.

Na stope 3 vytvoríme *míľniky* m_i – binárne reprezentácie čísiel, ktoré určujú, na ktorom políčku pásky sa míľnik začína. Pomocou týchto míľnikov potom vieme vypočítať pozíciu ľubovoľného políčka v čase $O(n)$. To využijeme na výpočet binárnej reprezentácie vzdialenosti markeru b od začiatku slova x, ktorú označíme ako d_b a analogicky vzdialenosti c od začiatku slova z, ktorú označíme ako d_c. Slová d_b a d_c budú síce ďaleko od seba (až $O(n)$), ale budú iba dĺžky $O(\log n)$.



Obr. 2.2: Vybrané stopy alternujúceho Turingovho stroja pre overovanie rovnosti slov

Popíšeme podrobnejšie konštrukciu míľnikov. Na stope 3 si existenčne vyberieme binárne reprezentácie čísiel $m_1, \ldots, m_t$, kde $t \leq n$, $m_1 = 1$, zápis m_1 bude začínať pod prvým znakom slova x a m_t bude začínať pod posledným znakom slova z. Keďže výber týchto slov prebieha existenčne, v niektorej vetve výpočtu bude platiť, že všetky *slová* m_i budú dĺžky $O(\log n)$ a budú od seba navzájom vzdialené $O(n/\log n)$. V niektorých vetvách sa stane, že uhádnutých slov bude príliš veľa, alebo budú od seba príliš vzdialené. To nám však nebude vadiť, nakoľko tieto slová hádame existenčne a keďže vieme, že existujú, niekedy ich určite uhádneme. Tu je dôležitá definícia pre časové ohraničenie alternujúceho Turingovho stroja - vetvy, ktoré trvajú príliš dlho, ale sú pre rozhodnutie o akceptácii zbytočné, do časového ohraničenia nezahŕňame. Táto vlastnosť nášho modelu TS sa ukáže ako dôležitá aj v niektorých ďalších dôkazoch.

Pre všetky medzery medzi m_i a m_{i-1}, t.j. pre univerzálne vybraté $i \in \{2, \ldots, t\}$ deterministicky overíme, či dĺžky týchto medzier, t.j. vzdialenosti *začiatkov slov*

m_i a m_{i-1} odpovedajú rozdielu čísel, ktoré m_i a m_{i-1} reprezentujú.

Slovo m_{i-1} skopírujeme na pomocnú stopu pod neho. Následne sa s ním budeme hýbať smerom doprava k slovu m_i a pri každom presune k nemu pripočítame jednotku.

Za predpokladu, že m_{i-1} je od m_i vzdialené $O(n/\log n)$ a posun slova dĺžky $O(\log n)$ o jedno políčko spolu s pričítaním jednotky trvá $O(\log n)$ krokov, prebehne celkový presun slova m_{i-1} pod m_i v čase $O(n/\log n).O(\log n) = O(n)$.

Potom už iba deterministicky overíme rovnosť presunutého m_{i-1} a m_i nad ním. Ak sa rovnajú, rozdiel $m_i - m_{i-1}$ je rovný veľkosti vzdialenosti medzi nimi. Ak overovanie zlyhá, odmietneme. V opačnom prípade platí, že každé slovo m_i začína na pozícii, ktorú reprezentuje – m_i budú tvoriť míľniky na páske.

Pomocou týchto míľnikov teraz môžeme rýchlo vypočítať binárne reprezentácie vzdialeností d_b a d_c. Keďže slovo x začína na začiatku pásky, vzdialenosť d_b zistíme ako pozíciu políčka s markerom b na páske. Existenčne uhádneme pod markerom b binárny zápis jeho pozície a overíme, či vzdialenosť markeru od najbližšieho míľnika je rovná rozdielu týchto čísel. Toto overovanie je rovnaké ako pri konštrukcii míľnikov a prebehne rovnako v čase $O(n)$.

Podobným spôsobom môžeme vypočítať aj pozíciu začiatku slova z a odčítať ju od pozície markeru c, čím získame aj vzdialenosť d_c. Jediný problém je, že začiatok slova z a marker c môžu byť od seba vzdialené až $O(n)$, takže jednu z vypočítaných pozícií musíme *preniesť* pod druhú pomocou algoritmu na rýchle prenášanie krátkych slov. Ich vzájomné odčítanie potom prebehne deterministicky v čase $O(\log n)$.

Následne už iba *rýchlym algoritmom* porovnáme slová d_b a d_c. Ak sa rovnajú, potom x je prefixom z a vstup môžeme akceptovať. □

Pomocou míľnikov zavedených v dôkaze predchádzajúcej lemmy dokážeme vypočítať, resp. existenčne uhádnuť a následne overiť binárnu reprezentáciu dĺžky ľubovoľného slova [10].

Lemma 2. *Pre ľubovoľnú konečnú abecedu Σ a oddeľovač $\# \notin \Sigma$ je jazyk*

$$L = \{x\#y \mid y \text{ je binárnou reprezentáciou } |x|\}$$

rozpoznávaný nejakým jednopáskovým alternujúcim Turingovým strojom v čase $O(n)$, kde $n = |x| + |y| + 1$.

Dôkaz. Zostrojíme míľniky pod časťou x vstupného slova, s tým, že posledný míľnik bude začínať pod posledným znakom x. Ten následne porovnáme s y. Ak sa rovnajú, vstup akceptujeme, v opačnom prípade odmietneme. □

Veta 1. *Pre všetky časovo skonštruovateľné funkcie $t(n)$ platí*

$$ATIME(t(n)) = ATIME_1(t(n))$$

Dôkaz. Nech $M = (Q, \Sigma, \delta, q_0, g)$ je k-páskový alternujúci Turingov stroj, akceptujúci jazyk L v čase $t(n)$. Konštrukciou ukážeme, že existuje jednopáskový alternujúci Turingov stroj M_1, ktorý akceptuje L v čase $O(t(n))$.

Stroj M_1 bude mať iba jednu pásku, ktorú rozdelíme na viac stôp. Jedna z nich, nazvime ju ako *hlavnú stopu*, bude simulovať výpočet stroja M, ostatné budú slúžiť na pomocné výpočty. Pre postupnosť konfigurácií v strome

výpočtu M, ktorej začiatok je v koreni stromu, zadefinujeme jej *protokol* ako postupnosť $e = e_1, e_2, \ldots e_t$, kde

$$e_i \in Q \times \Sigma^k \times Q \times \Sigma^k \times \{l, r, 0\}^k$$

rozpísané podrobnejšie

$$e_i = (q_i, a_i^1, \ldots, a_i^k, q_i', b_i^1, \ldots, b_i^k, s_i^1, \ldots, s_i^k)$$

Každé e_i bude reprezentovať jeden krok v strome výpočtu, t.j. prechod (hranu) v strome výpočtu z jedného vrcholu k niektorému jeho nasledovníkovi. Prvý stav q_i v e_i kóduje vstupný stav stroja pred aktuálnym krokom výpočtu, k znakov abecedy $a_i^1 \ldots a_i^k$ kóduje symboly na jednotlivých páskach pred krokom výpočtu v aktuálnej pozícii hláv, nasledovný stav q_i' a ďalších k znakov abecedy $b_i^1 \ldots b_i^k$ kódujú stav stroja a zmenu symbolov na páskach v aktuálnej pozícii hlavy a nakoniec $s_i^1 \ldots s_i^k$ určujú pre každú hlavu zmenu jej pozície - posunie sa vľavo (l), vpravo (r), alebo ostáva na pôvodnom mieste (0).

Každý protokol môžeme priamočiarym spôsobom previesť na slovo v abecede stroja M_1 a pracovať s ním na páske. Stroj M_1 skonštruujeme tak, že pre každý vrchol v akceptujúcom podstrome výpočtu stroja M bude existovať vrchol v strome výpočtu M_1 taký, že protokol (t.j. cesta od koreňa k danej konfigurácii stroja M) je zapísaný na hlavnej stope stroja M_1. Protokol pre koreň stromu M bude prázdna postupnosť a aj hlavná stopa stroja M_1 bude na začiatku simulácie prázdna.

Pri simulácii stroja M pomocou M_1 uhádneme protokol $e = e_1 \ldots e_t$. Už pri hádaní môžeme dodržať konzistentnosť stavov – zachováme v postupnosti platné $q_i = q_{i-1}'$. Začiatočné znaky na páskach $a_i^1 \ldots a_i^k$ uhádneme pre každé i existenčne. Protokol zapísaný na hlavnej páske priamo odpovedá jednému z vrcholov v strome výpočtu stroja M. Ak je tento vrchol existenčný, potrebujeme preskúmať jeho nasledovníkov a zistiť, či *niektorý* z nich je akceptujúci. Podobne v prípade, že vrchol je univerzálny, potrebujeme zistiť, či sú akceptujúci *všetci* jeho nasledovníci. Na to môžeme priamo využiť alternáciu stroja M_1, preto priebeh kroku výpočtu - teda výstupné znaky $b_i^1 \ldots b_i^k$, výstupný stav q_i' a posuny hláv $s_i^1 \ldots s_i^k$, uhádneme existenčne, alebo univerzálne, podľa stavu q_i v M. Toto hádanie zároveň prebieha konzistentne s prechodovou funkciou stroja M.

Po každom vygenerovaní novej časti protokolu e_i overíme, či je potrebné pokračovať. Ak je stav q_i' finálny a odmietajúci, M_1 vstup odmietne. Ak je stav q_i' finálny a akceptujúci, M_1 prejde do fázy overovania korektnosti simulácie popísanej ďalej. Ak stav q_i' finálny nie je, M_1 vygeneruje ďalšiu časť protokolu.

Aby bol protokol korektný a zodpovedal priebehu výpočtu stroja M, potrebujeme overiť, či sú uhádnuté znaky na páskach správne inicializované, a či sa v priebehu výpočtu menia iba vtedy, keď je nad nimi hlava. To znamená, že potrebujeme overiť nasledovnú podmienku:

(P) Podmienka korektnosti protokolu

> Nech $p_i^v = |\{u | s_u^v = r \wedge u < i\}| - |\{u | s_u^v = l \wedge u < i\}|$. Potom p_i^v bude reprezentovať pozíciu hlavy v v i-tom kroku. Pre všetky $i = 1 \ldots t$ a pre všetky pásky $v = 1 \ldots k$ platí :

(P1) Ak $i' < i$ je najväčšie možné také, že $p_i^v = p_{i'}^v$, potom $b_{i'}^v = a_i^v$, t.j. ak je hlava v rovnakej pozícii na páske v i-tom kroku a predtým tam naposledy bola v kroku i', potom sa symbol na páske po kroku i' a pred krokom i nezmenil.

(P2) Ak také i' neexistuje, potom je a_i^1 v prípade, že $1 \leq p_i^1 \leq |input|$, rovné symbolu na p_i^1-tom mieste vstupného slova, alebo B v inom prípade. Táto podmienka zaručí, že na prvej páske je pred prvým prečítaním daného políčka správny znak zo vstupného slova, alebo symbol pre *blank*.

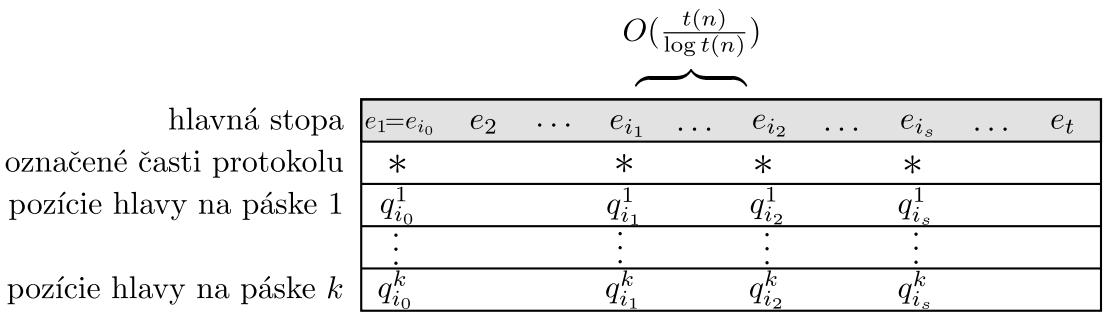
Pre rozhodnutie akceptovania v simulácii potrebujeme iba (dostatočne rýchlo) overiť, či pre vygenerovaný protokol platí podmienka (P). Ak áno, potom priebeh protokolu a jeho korektnosť zaručí, že stroj M daný vstup akceptuje a teda ho môže akceptovať aj M_1. Ak protokol korektný nie je, M_1 vstup odmietne. To pre nás znamená, že *neexistuje* akceptujúci výpočet stroja M na danom vstupe, pretože v opačnom prípade by existoval aj jeho protokol, a ten bol strojom M_1 hádaný existenčne.

Na zakódovanie jedného kroku protokolu e_i potrebujeme $2k$ znakov pre vstupné a výstupné znaky na páskach, 2 znaky pre stavy stroja M a k znakov pre pohyb hláv. Stroj M ukončí výpočet pre vstupné slovo dĺžky n za $t(n)$ krokov, preto protokol tohto výpočtu bude mať dĺžku $(3k+2)t(n) \in O(t(n))$ a bude vygenerovaný v rovnakom čase.

Overenie korektnosti protokolu bude prebiehať tiež v čase $O(t(n))$. Na začiatku overovania si M_1 na ďalšej stope existenčne vyberie a označí niektoré časti protokolu $e_{i_1} \dots e_{i_s}$. Stav e_1 označí tiež a v zvyšku dôkazu ho budeme používať aj ako e_{i_0}.

Pre každý z označených stavov M_1 existenčne uhádne možné polohy hláv - t.j. čísla $r_{i_j}^1 \dots r_{i_j}^k$ a zapíše ich binárne reprezentácie na dalších k pomocných stôp, vždy pod začiatok príslušnej časti protokolu e_{i_j}.

Pre pokračovanie overovania budeme potrebovať, aby označených častí bolo vhodne veľa a boli vhodne ďaleko od seba, podobne ako pri hádaní míľnikov v dôkaze lemmy 1. Zápis protokolu má dĺžku $O(t(n))$, každé $r_{i_j}^v$ označuje potenciálnu pozíciu hlavy v stroja M počas výpočtu, ktorého dĺžka je $t(n)$, binárna reprezentácia $r_{i_j}^v$ má teda dĺžku nanajvýš $O(\log t(n))$. Vhodný počet označených častí protokolu bude $O(\log t(n))$ a vzdialenosti medzi nimi budú $O(t(n)/\log t(n))$. Budeme predpokladať, že takýto výber označenia existuje a dá sa existenčne uhádnuť, teda dostatočne rýchle overenie bude existovať.



Obr. 2.3: Páska stroja M_1 s označenými časťami protokolu

Pre overenie správnosti kandidátov na pozície hláv najprv overíme, či sú hlavy na začiatku výpočtu na ľavom okraji pásky, t.j $r^v_{i_0} = 0$ pre všetky pásky $v = 1 \dots k$. Ďalej si M_1 univerzálne vyberie e_{i_j} $(j = 1 \dots s)$ z označených častí protokolu a pásku $v \in 1 \dots k$. Pre tieto overí, či kandidáti $r^v_{i_j}$ a $r^v_{i_{j-1}}$ sú konzistentní s testovaným protokolom: hlava na páske v v časti i_{j-1} je na pozícii $r^v_{i_{j-1}}$, ďalej sa hýbe podľa protokolu medzi $e_{i_{j-1}}$ a e_{i_j} a skončí v pozícii $r^v_{i_j}$. Zapísané formálne – musíme overiť, či platí

$$r^v_{i_j} = r^v_{i_{j-1}} + |\{u | s^v_u = r \wedge i_{j-1} \leq u < i_j\}| - |\{u | s^v_u = l \wedge i_{j-1} \leq u < i_j\}|$$

Číslo $r^v_{i_{j-1}}$ skopírujeme pod neho na novú stopu. Následne sa s ním budeme hýbať po protokole doprava smerom k číslu $r^v_{i_j}$. Vždy, keď sa nachádzame pod časťou protokolu rozhodujúcou o pohybe danej hlavy, zmeníme toto číslo podľa pohybu – ak sa hlava posúva doprava, pričítame k nemu jednotku, ak doľava, tak jednotku odčítame a ak ostáva na mieste, číslo nezmeníme. Keď sa dostaneme pod číslo $r^v_{i_j}$ iba porovnáme, či sa rovnajú. Ak nie, protokol je nekorektný a M_1 vstup odmietne. Ak je táto podmienka splnená, potom čísla $r^v_{i_j}$ označujú polohy hláv počas výpočtu M, t.j. $r^v_{i_j} = p^v_{i_j}$.

Ďalej potrebujeme overiť, či platí podmienka (P), teda nemennosť a správna inicializácia znakov na páskach v protokole. Univerzálne vyberieme niektoré e_i $(i \geq 1)$ a pásku v. K nim existenčne vyberieme niektoré $e_{i'}$ pred e_i, t.j. $i' < i$, alebo fiktívny stav, ktorý bude znamenať, že vhodné i' neexistuje. Podľa toho otestujeme z podmienky (P) prípad (P1), alebo (P2).

Najprv otestujeme, či i', ktoré sme vybrali, spĺňa predpoklad, že i' je *najväčšie* také, že $p^v_i = p^v_{i'}$, prípadne, že takéto i' neexistuje. M_1 univerzálne vyberie $e_{i''}$ také, že $i' < i'' < i$, v prípade, že i' neexistuje, sa prvá časť nerovnosti vynechá. Následne vypočítame pomocou $r^v_{i_j}$ polohy hláv p^v_i, $p^v_{i'}$ a $p^v_{i''}$ a pomocou lemmy 1 overíme platnosť $p^v_i = p^v_{i'}$ a zároveň $p^v_i \neq p^v_{i''}$. Ak toto overovanie zlyhá, i' sme vybrali nesprávne a stroj M_1 vstup odmietne.

Ak sme si vybrali i', musíme iba deterministicky overiť, či sú v protokole znaky na páskach po kroku i' a pred krokom i rovnaké. Ak je táto podmienka splnená, stroj M_1 svoj vstup akceptuje (a akceptuje ho aj M). V opačnom prípade odmieta.

V prípade, že i' sme nevybrali, najprv vypočítame polohu hlavy p^v_i pomocou najbližšieho $r^v_{i_j}$. Ďalej zapíšeme na ďalšiu stopu binárnu reprezentáciu dĺžky vstupu n podľa lemmy 2 a porovnáme ju s p^v_i. Ak sme na inej ako prvej páske, alebo ak $n < p^v_i$, teda mimo vstupného slova, overíme iba, či $a^v_i = B$.

V opačnom prípade overíme, či na p^v_i-tom mieste vstupného slova je znak a^v_i. Máme k dispozícii binárnu reprezentáciu čísla p^v_i. Potrebujeme zistiť, ktoré miesto vo vstupnom slove jej zodpovedá. To urobíme nasledovným spôsobom. Existenčne uhádneme vo vstupnom slove pozíciu l, ktorá je kandidátom na pozíciu p^v_i. Zvyšnú časť vstupného slova, t.j. znaky $l+1, l+2, \dots, n$ zmažeme. Z podslova vstupu na pozíciách $1, 2, \dots, l$ zistíme podľa lemmy 2 jeho dĺžku v binárnej reprezentácii (t.j. binárnu reprezentáciu čísla l) a porovnáme ju s p^v_i podľa lemmy 1. Ak sa rovnajú, overíme už iba, či je na l-tom, t.j. poslednom znaku vstupného slova (zvyšná časť je zmazaná) znak a^v_i. Podľa toho akceptujeme alebo odmietame. □

Predchádzajúca veta hovorí, že každý m-páskový alternujúci Turingov stroj sa dá bez výrazného spomalenia prerobiť na jednopáskový. Tento výsledok teraz

využijeme na dokázanie vety o časovej hierarchii [11, 2] . Podobná veta platí pre triedy jazykov nedeterministických Turingových strojov [8], ale jej dôkaz nie je analógiou uvedeného. Pre deterministické TS je veta o hierarchii [5, 6] dokázaná iba pre funkcie oddelené aspoň logaritmicky, t.j. $t(n) \in o(T(n)/\log T(n))$.

Veta 2. *Nech $T(n)$ je časovo konštruovateľná funkcia a $t(n)$ taká, že platí $t(n) \in o(T(n))$. Potom trieda jazykov $ATIME(t(n))$ je vlastnou podmnožinou $ATIME(T(n))$, t.j. existuje jazyk L taký, že $L \in ATIME(T(n))$ a zároveň $L \notin ATIME(t(n))$.*

Dôkaz. Dôkaz prebieha využitím diagonalizácie. Všetky jednopáskové alternujúce Turingove stroje očíslujeme ako M_x, $x \in \{0,1\}^*$. Skonštruujeme dvojpáskový ATS M, prijímajúci jazyk $L = L(M)$ v čase $T(n)$.

Na pomocnej páske pre vstup w, $|w| = n$ zaznačíme dĺžku $T(n)$. Funkcia T je časovo konštruovateľná a teda tento krok prebehne v čase $O(T(n))$. Po konštrukcii sa pustí na tejto páske časovač počítajúci kroky, ktorý zabezpečí zastavenie nasledujúcej simulácie v čase $T(n)$.

Ďalej stroj M overí, či je vstup w tvaru $w = x\#y$ pre nejaké $x, y \in \{0,1\}^*$. Ak nie je, vstup w odmietne.

Na prvej páske teraz prebehne simulácia stroja M_x na vstupe w s tým, že sa navzájom vymenia všetky univerzálne stavy s existenčnými a všetky akceptujúce s odmietajúcimi. Teda pre výpočty prebiehajúce v čase $T(n)$ stroj M akceptuje vstup w práve vtedy, keď M_x vstup w odmietne a vice versa.

Ak sa časovač počas prebiehajúcej simulácie dostane na začiatok pásky, t.j. uplynie povolená doba výpočtu, stroj M vstup akceptuje. Tým sa negujú nekonečné (a teda odmietnuté), alebo príliš dlhé výpočty.

Stroj M zjavne pracuje v čase $O(T(n))$, čo je zabezpečené použitím časovača. Dokážeme teraz, že jazyk $L(M)$ nepatrí do triedy $ATIME(t(n))$. Stroj M sme síce zostrojili ako dvojpáskový, ale podľa vety 1 k nemu existuje jednopáskový pracujúci až na konštantné spomalenie v rovnakom čase. Nech $L = L(M)$ teda patrí do $ATIME(t(n))$. To znamená, že existuje jednopáskový alternujúci Turingov stroj M_x, ktorý akceptuje všetky slová z jazyka L v čase $O(t(n))$. Keďže $t(n) \in o(T(n))$, stroj M bude mať pre všetky vstupy tvaru $x\#y$ na simuláciu výpočtu stroja M_x dostatočne veľa času, pričom vyhodnotí výsledok presne opačným spôsobom, ako stroj M_x. To je však spor s tvrdením, že $L(M) = L(M_x)$.

□

Poznámka 1. *Všimnime si, že v skutočnosti sme dokázali silnejšie tvrdenie, že triedy jazykov*

$$\Pi_k\text{-}TIME(T(n)) \setminus \Sigma_k\text{-}TIME(t(n))$$

aj

$$\Sigma_k\text{-}TIME(T(n)) \setminus \Pi_k\text{-}TIME(t(n))$$

sú neprázdne. To vyplýva z toho, že stroj, ktorý vznikol diagonalizáciou patrí (napríklad) do triedy $\Pi_k\text{-}TIME(T(n))$, ale pritom sa správa inak, ako ľubovoľný stroj z triedy $\Sigma_k\text{-}TIME(t(n))$. Pre ďalšie dokazovanie nám však bude stačiť hore uvedené znenie vety.

Ďalej dokážeme vetu o lineárnom zrýchlení, platnú pre alternujúce Turingove stroje. Tá nám umožní stotožniť triedy $ATIME(t(n))$ a $ATIME(ct(n))$ pre

ľubovoľné $c > 0$ a $t(n)$ také, že $n \in o(t(n))$. Dôkaz tejto vety je analogický k dôkazu o lineárnom zrýchlení pre deterministické Turingove stroje, ktorý je dostatočne známy a dá sa nájsť v práci *Structural Complexity I* [1]. Uvedieme preto iba ideu tohto dôkazu a zdôrazníme rozdiel, ktorý prináša alternácia.

Veta 3. *Nech L je jazyk akceptovaný nejakým 1-páskovým alternujúcim Turingovým stojom M v čase $t(n)$ takom, že $n \in o(t(n))$ a nech $c > 0$ je reálne číslo. Potom existuje alternujúci Turingov stroj M' taký, že akceptuje L v čase $c.t(n)$.*

Dôkaz. Nech $m > 0$ je prirodzené čislo. Ukážeme, že m krokov výpočtu ATS M na úseku pásky dĺžky m vieme simulovať *dvomi* krokmi výpočtu upraveného stroja M'.

Strom výpočtu ATS počítajúceho m krokov začína v existenčnom, alebo univerzálnom stave a má hĺbku m. Predpokladajme, že v tejto hĺbke m sa už nachádzajú vrcholy, o ktorých akceptácii je rozhodnuté. Ukážeme, že m krokov výpočtu vieme nahradiť dvomi, za cenu exponenciálneho rozvetvenia tohto stromu.

Induktívne priradíme každému vrcholu stromu výpočtu formulu výrokového počtu nasledovne:

- Ak je vrchol v hĺbke m, (čiže o jeho akceptácii je rozhodnuté mimo uvažovaný strom výpočtu) priradíme mu unikátnu výrokovú premennú $\mathfrak{F} \equiv \alpha_i$.
- Ak je vrchol existenčný a jeho synovia sú ohodnotení formulami $\mathfrak{F}_1, \mathfrak{F}_2, \ldots, \mathfrak{F}_i$, potom mu priradíme formulu $\mathfrak{F} \equiv \mathfrak{F}_1 \vee \mathfrak{F}_2 \vee \ldots \vee \mathfrak{F}_i$.
- Ak je vrchol univerzálny a jeho synovia sú ohodnotení formulami $\mathfrak{F}_1, \mathfrak{F}_2, \ldots, \mathfrak{F}_i$, potom mu priradíme formulu $\mathfrak{F} \equiv \mathfrak{F}_1 \wedge \mathfrak{F}_2 \wedge \ldots \wedge \mathfrak{F}_i$.

Je vidieť, že ak sú listy stromu výpočtu v hĺbke m ohodnotené výrokovými premennými α_i, a ak týmto premenným priradíme pravdivostné hodnoty podľa toho, či ich stavy akceptujú ($True$), alebo odmietajú ($False$), potom akceptácia každého vrcholu stromu výpočtu bude určená pravdivostnou hodnotou k nemu priradenej formule.

Pre formulu $\mathfrak{F}$ ohodnocujúcu koreň stromu výpočtu, existuje formula výrokového počtu $\mathfrak{F}_{NF}$, ktorá je buď v konjunktívnom, alebo disjunktívnom normálnom tvare a pre ktorú platí $\mathfrak{F} \equiv \mathfrak{F}_{NF}$.

Podobným (opačným) spôsobom, ako sme vyrábali formulu k vrcholom stromu, zostrojíme nový strom výpočtu pomocou formule $\mathfrak{F}_{NF}$. Z toho, že táto formula je v jednom z normálnych tvarov vyplýva, že hĺbka nového stromu bude 2, napríklad v prípade $\mathfrak{F} \equiv \mathfrak{F}_{CNF}$ bude prvý vrchol existenčný a rozvetví sa do niekoľkých univerzálnych, ktoré sa ďalej rozvetvia do pôvodných listov. Šírka tohto nového stromu výpočtu sa môže zväčšiť až exponenciálne vzhľadom k počtu listov pôvodného stromu hĺbky m.

V samotnej simulácii postupujeme rovnako, ako v štandardnom dôkaze. Na začiatku prejdeme celú pásku a upravíme jej obsah tak, že skomprimujeme m znakov do jedného, ktorý zapisujeme na pomocnú pásku.

Potom počas simulácie načítame do konečnej pamäte obsah troch už skomprimovaných políčok (t.j. dokopy $3m$ pôvodných znakov), vrátime sa na stredné políčko a v dvoch krokoch bez posúvania hlavy prevedieme simuláciu m krokov

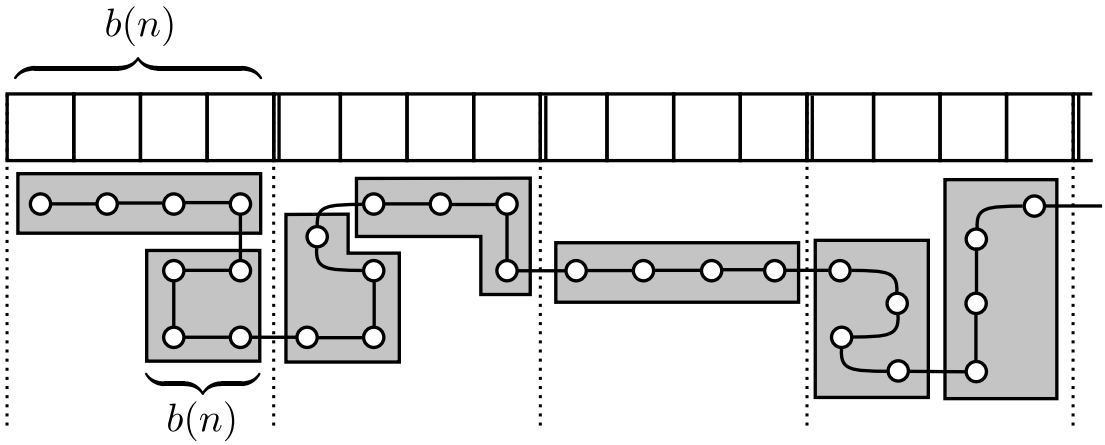
pôvodného stroja na prečítaných dátach. Zo začiatku stredného políčka sa za m krokov hlavou dostaneme maximálne do jedného z krajných, takže následne zapíšeme do dvoch políčok nové znaky po prebehnutej simulácii. Vrátime sa na miesto a simulácia môže pokračovať.

Kompresia pásky trvá čas $n + \lceil \frac{n}{m} \rceil$ a na simuláciu potrebujeme čas $7\frac{t(n)}{m}$. Pre dané $c > 0$ preto vyberieme $m > 14/c$ a zostrojíme stroj M' podľa uvedenej konštrukcie. Ten bude pracovať v čase $n + \lceil \frac{n}{m} \rceil + 7\frac{t(n)}{m}$. To pre $m > 2$ zhora odhadneme ako $2n + 7 + 7\frac{t(n)}{m}$. Z predpokladu $n \in o(t(n))$ dostávame pre dostatočne veľké n horný odhad $\frac{c}{2}t(n) + 7\frac{t(n)}{m}$ a ďalej pre $m > 14/c$ už finálny odhad $\frac{c}{2}t(n) + \frac{c}{2}t(n) = ct(n)$. Konečne veľa vstupov malej dĺžky ošetríme vyhľadaním v tabuľke. □

2.3 Graf výpočtu Turingovho stroja pracujúceho v blokoch

Na dokázanie vety o urýchlení výpočtu deterministického Turingovho stroja pomocou alternujúceho Turingovho stroja s konštantným počtom alternácií (Veta 5), ktorá tvorí jadro dôkazu $DLIN \subsetneq NLIN$ potrebujeme zaviesť technickú modifikáciu deterministických Turingových strojov.

Pre danú funkciu $b(n)$ môžeme rozdeliť pásku deterministického Turingovho stroja na bloky veľkosti $b(n)$. Podobne môžeme rozdeliť do jednotlivých časových etáp dĺžky $b(n)$ aj priebeh výpočtu. Ak výpočet na vstupe dĺžky n trvá $t(n)$ krokov, potom jednotlivých časových fáz výpočtu bude $a(n) = \lceil \frac{t(n)}{b(n)} \rceil$. Rozdelenie pásky aj výpočtu zobrazuje obrázok 2.4.



Obr. 2.4: Výpočet 1-páskového DTS pracujúceho v blokoch dĺžky $b(n) = 4$

Pre i-ty blok na páske budeme používať označenie B_i. Formálne, ak očíslujeme políčka v-tej pásky číslami $0, 1, 2, \ldots$, bude $B_i{}^v = \{j \mid ib(n) \leq j < (i+1)b(n)\}$. Keďže väčšinou budeme hovoriť o bloku jednej, pevnej určenej pásky, horný index nebudeme používať. Po sebe nasledujúcich $b(n)$ krokov výpočtu, počas ktorých hlava na žiadnej páske neprejde z jedného bloku do druhého, budeme nazývať *blokovou*, alebo *časovou fázou*. Všimnime si, že raz hovoríme o bloku na *páske*, t.j. o $b(n)$ susediacich políčkach na páske, a inokedy o blokovej *fáze*, t.j. o $b(n)$ následných krokoch výpočtu.

Definícia 4. *k-páskový deterministický Turingov stroj M* pracuje v blokoch, *ak počas výpočtu v každej časovej fáze dĺžky $b(n)$ každá hlava na páske pracuje iba v jednom, pevne určenom bloku B_i veľkosti $b(n)$.*

Za cenu pridania jednej pásky práca v blokoch nespôsobí významné spomalenie výpočtu. Ak nás počet pások nebude zaujímať, potom ku každému DTS existuje DTS pracujúci v blokoch prijímajúci rovnaký jazyk v rádovo rovnakom čase. To je obsahom nasledujúcej lemmy.

Lemma 3. *Ak je $t(n)$ časovo konštruovateľná, L je jazyk akceptovaný k-páskovým deterministickým Turingovým strojom M v čase $t(n)$, potom existuje $(k+1)$-páskový deterministický Turingov stroj M_b pracujúci v blokoch a akceptujúci L v čase $O(t(n))$.*

Dôkaz. Popíšeme konštrukciu stroja M_b pomocou M. Predpokladajme, že máme časovo konštruovateľnú funkciu $t(n)$ a priestorovo konštruovateľnú funkciu $b(n)$, ktorá pre daný výpočet určuje dĺžku jedného bloku. Túto dĺžku si na začiatku výpočtu skonštruujeme na jednu pomocnú pásku (tú, ktorú máme oproti stroju M navyše) a ďalej na nej budeme počítať časové úseky tejto dĺžky. Predpokladáme, že $b(n)$ vieme skonštruovať v čase $O(t(n))$.

Každú pásku pôvodného stroja M rozdelíme na tri stopy. Stredná stopa bude obsahovať simuláciu pôvodného výpočtu. Na hornej stope bude uložený reverz slova na páske v bloku vľavo a na spodnej stope reverz slova na páske v bloku vpravo od aktuálneho. To nám umožní počas simulácie namiesto opustenia aktuálneho bloku pokračovať v ňom a pritom pracovať s informáciami zo susedných blokov. Nové stopy na páskach ilustruje obrázok 2.5.

	a^R	$\mathrm{b}_4\ \mathrm{b}_3\ \mathrm{b}_2\ \mathrm{b}_1$	c^R
a	$\mathrm{b}_1\ \mathrm{b}_2\ \mathrm{b}_3\ \mathrm{b}_4$	c	d
$\mathrm{b}_4\ \mathrm{b}_3\ \mathrm{b}_2\ \mathrm{b}_1$	c^R	d^R	

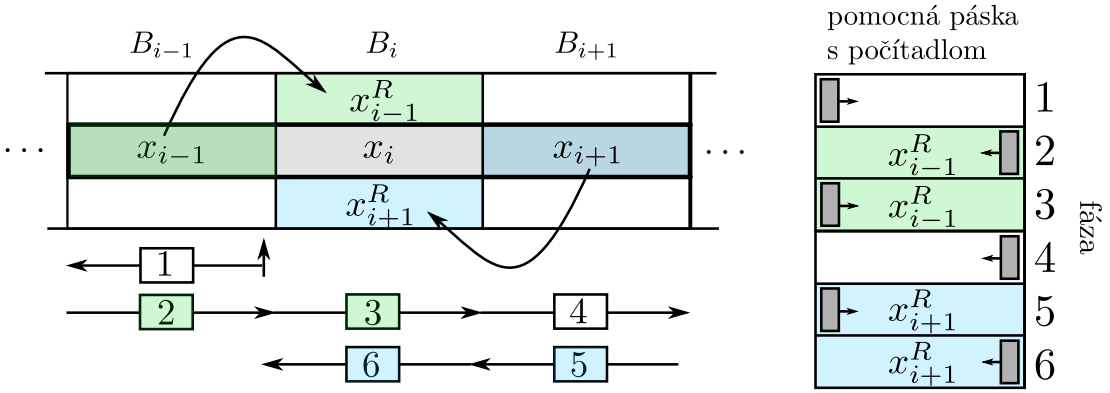
Obr. 2.5: Tri stopy DTS pracujúceho v blokoch

Keď sa niektorá hlava v pôvodnej simulácii rozhodne vstúpiť do nového bloku, musíme naformátovať jeho hornú a spodnú stopu podľa blokov vľavo a vpravo. Na to využijeme pomocnú pásku na ktorej prebieha počítanie blokových fáz. Hlava vchádza do bloku B_i, ktorý ešte nie je naformátovaný.

- Počas jednej blokovej fázy sa hlava presunie na začiatok bloku B_{i-1}.
- V druhej fáze tento blok prejde a skopíruje obsah strednej stopy na pomocnú pásku, kam sa presne zmestí – jeho dĺžka je práve $b(n)$.
- V tretej fáze hlava prechádza blokom B_i a kopíruje pomocnú pásku do hornej stopy, pričom pomocnú pásku čítame z opačnej strany, ako sme ju zapísali, preto sa do hornej stopy dostane reverz slova vľavo.
- V štvrtej fáze prejdeme na koniec bloku B_{i+1}.
- V piatej fáze prechádzame odzadu blok B_{i+1} a zapisujeme obsah jeho strednej stopy na pomocnú pásku s počítadlom. Keďže slovo v tomto bloku čítame odzadu, na konci čítania bude hlava na pomocnej stope na začiatku tohto slova.

- V poslednej šiestej fáze zapíšeme obsah pomocnej pásky na spodnú stopu (zapisujeme odzadu bloku B_i, preto opäť vzniká reverz pôvodného slova), čím sa dostávame na začiatok bloku B_i, ktorý sa takto stal správne naformátovaným.

Tento proces je zobrazený na obrázku 2.6.



Obr. 2.6: Formátovanie blokov pásky pre DTS pracujúci v blokoch.

Na naformátovanie bloku sme využili 6 blokových fáz, potrebný čas je $6b(n)$ krokov.

Ak sa neskôr počas výpočtu hlava pokúsi tento blok opustiť, pričom pracuje ešte len na strednej stope, využije namiesto toho informácie z príslušnej stopy hore, alebo dole a výpočet pokračuje v rovnakom bloku ďalej.

Ak dôjde k tomu, že výpočet už prebieha na niektorej pomocnej stope a chce pokračovať v ďalšom bloku, ktorý predformátovaný ešte nie je, simulácia sa preruší, počká sa, kým na časovači nedôjde k ukončeniu jednej blokovej fázy (najviac $b(n)-1$ krokov), príslušné stopy sa opäť s využitím pomocnej pásky skopírujú na svoje správne miesto do strednej stopy vedľajších blokov (rovnako $6b(n)$ krokov ako na formátovanie, ide o presne opačný proces) a výpočet pokračuje.

Je zrejmé, že stroj M_b pracuje v blokoch a akceptuje rovnaký jazyk ako pôvodný stroj M. Ostáva nám ukázať, že spomalenie stroja M_b bude konštantné. K simulácii $b(n)$ krokov pôvodného výpočtu pribudlo $6b(n)$ krokov na naformátovanie bloku, najviac $b(n)-1$ krokov sa čaká na dokončenie blokovej fázy pri prechode z jedného naformátovaného bloku do druhého a na prekopírovanie hornej a spodnej stopy bloku do vedľajších potrebujeme ďalších $6b(n)$ krokov. Takéto zdržanie je možné na každej páske a preto každých $b(n)$ krokov výpočtu pôvodného stroja je simulovaných najviac $(1+6+1+6)k.b(n) = 14k.b(n)$ krokmi stroja M_b. □

Ku DTS pracujúcemu v blokoch zostrojíme graf jeho výpočtu, ktorý bude zobrazovať opakované návštevy hláv v blokoch na páske. Tento graf bude orientovaným acyklickým grafom s násobnými hranami. Zároveň o týchto grafoch dokážeme, že pre ne platí veta o existencii m-segregátoru, ktorej dôkazu venujeme samostatnú kapitolu.

Definícia 5. Graf výpočtu *deterministického Turingovho stroja M pracujúceho v blokoch na vstupe x dĺžky $|x| = n$ je orientovaný acyklický graf $G = (V, E)$, kde $V = \{1, 2, \ldots, a(n)\}$ a vrcholy i a j, $i < j$ sú spojené hranou práve vtedy,*

keď $j = i + 1$, alebo stroj M hlavou na niektorej páske navštívi blok B v časovej fáze i aj j, ale medzi nimi sa touto hlavou do bloku B nedostane. Hrany tvaru $(i, i + 1)$ budeme nazývať aj líniou výpočtu, keďže sú to práve hrany určujúce časovú následnosť jednotlivých fáz.

Vytvorenie grafu výpočtu z dvojpáskového deterministického Turingovho stroja pracujúceho v blokoch je zobrazené na obrázku 2.7. Jednotlivé políčka na páskach sú tvorené blokmi veľkosti $b(n)$. Každý uzol znázorňujúci polohu hlavy na páske reprezentuje $b(n)$ krokov výpočtu.

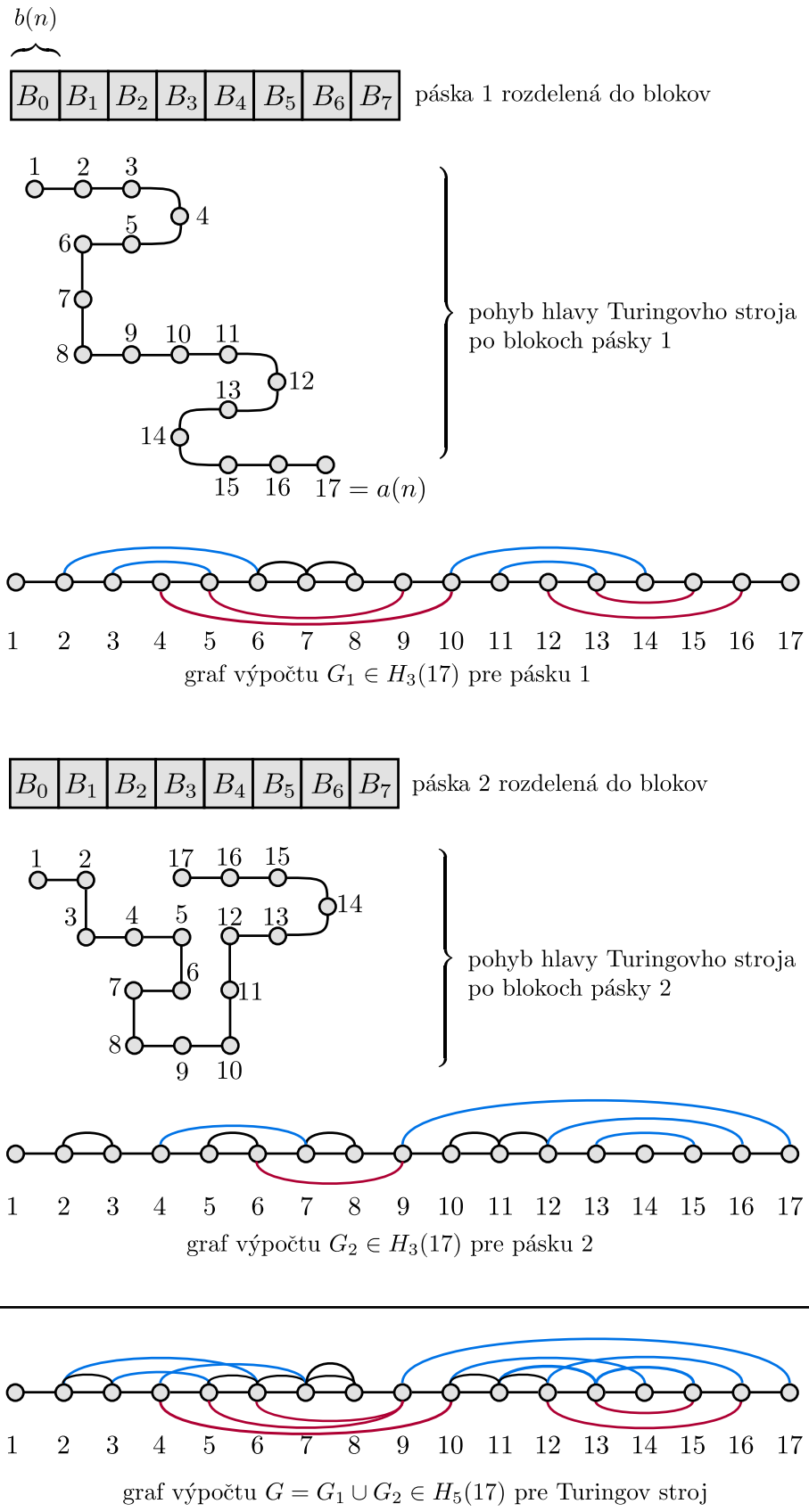
Orientované acyklické grafy s vrcholmi $V = \{1, 2, \ldots, n\}$ kde pre všetky hrany $e = (v_1, v_2)$ platí $v_1 < v_2$ označme ako triedu grafov $H(n)$. Z tejto triedy pomocou nasledujúcej definície vyčleníme podtriedy, do ktorých zaradíme grafy výpočtov podľa počtu pások Turingových strojov im zodpovedajúcim.

Poznámka 2. *Na obrázkoch znázorňujúcich grafy z triedy $H(n)$ nebudeme kresliť šípky označujúce orientáciu hrany, orientácia každej hrany je rovnaká a smeruje od vrcholu s nižším číslom k vyššiemu.*

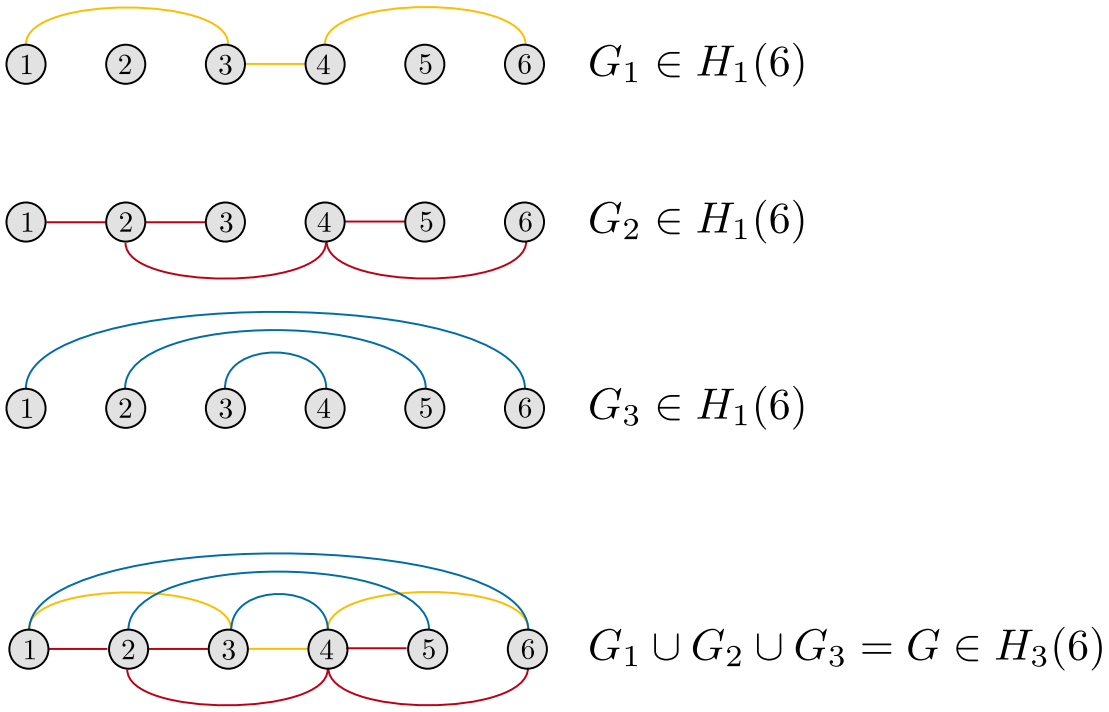
Definícia 6. *Hrany $e_1 = (i_1, j_1)$ a $e_2 = (i_2, j_2)$ grafu G s vrcholmi $V = \{1, 2, \ldots, n\}$ sa* križia*, ak platí $i_1 < i_2 < j_1 < j_2$. Kríženie hrán znázorňuje obrázok 2.8.*

Obr. 2.8: Hrany e_1 a e_2 sa v grafe vľavo križia, v grafe vpravo nie

Definícia 7. *Graf $G = (V, E)$ patrí do triedy $H_1(n)$ práve vtedy, keď $V = \{1, 2, \ldots, n\}$ a žiadne dve jeho hrany sa nekrížia. Graf G patrí do triedy $H_r(n)$ práve vtedy, keď je zjednotením r grafov z triedy $H_1(n)$. Grafy z triedy $H_r(n)$ nazývame aj* push-down *grafmi. Príklady grafov tried $H_r(n)$ zobrazuje obrázok 2.9.*



Obr. 2.7: Dvojpáskový Turingov stroj pracujúci v blokoch a jeho graf výpočtu.



Obr. 2.9: Príklad grafu z triedy $H_3(6)$

Lemma 4. *Graf výpočtu G k-páskového deterministického Turingovho stroja pracujúceho v blokoch patrí do triedy $H_{2k+1}(a(n))$.*

Dôkaz. Graf G má z definície $a(n)$ vrcholov. Pre každú pásku zadefinujeme grafy L_v a R_v s $a(n)$ vrcholmi, $v = 1, \ldots, k$. K nim pridáme ešte jeden graf C. Do týchto zadefinovaných grafov roztriedime hrany grafu výpočtu G.

Hrana $e = (i, j)$ sa do grafu G podľa definície môže dostať jedným z nasledujúcich spôsobov:

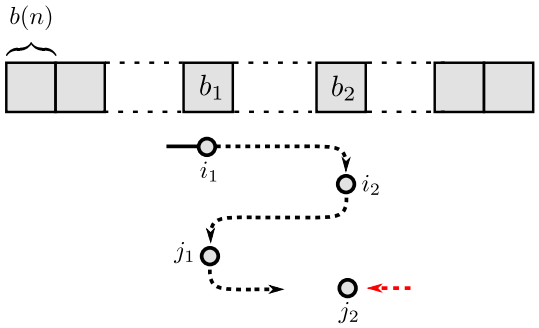
- hrana je jednou z hrán línie výpočtu, t.j. $e = (i, i + 1)$, vtedy ju zaradíme do grafu C.
- hrana vznikla tým, že DTS M sa vrátil hlavou v do niektorého bloku na v-tej páske vo fáze j, pričom naposledy sa v ňom na tejto páske vyskytol vo fáze i. Ak sa na páske do tohto bloku vrátil sprava, zaradíme hranu do grafu R_v, ak zľava, do grafu L_v a v prípade, že v danom bloku bol priamo v predchádzajúcej fáze (a teda sa nevracia zo žiadnej strany), hrana patrí do grafu C.

Pre graf G_1 z obrázku 2.7 budú do grafu L_1 patriť červené hrany, do grafu R_1 modré hrany a zvyšné čierne do grafu C. Analogicky pre graf G_2 a jeho rozdelenie L_2 a R_2. Graf C je pre oba grafy G_1 a G_2 spoločný.

Každý z grafov C, R_v a L_v je z triedy $H_1(a(n))$. Dokážeme to pre graf R_v, dôkaz pre L_v bude úplne analogický a pre graf C je toto tvrdenie triviálne. Keďže $G = C \cup \bigcup_{v=1}^{k} R_v \cup \bigcup_{v=1}^{k} L_v$, platí $G \in H_{1+k+k}(a(n))$.

Nech pre niektoré hrany $e_1 = (i_1, j_1)$ a $e_2 = (i_2, j_2)$ grafu R_v platí $i_1 < i_2 < j_1 < j_2$. Označme blok, ktorý vytvorí hranu e_1 ako b_1, analogicky blok b_2. Potom po fáze i_1 DTS opúšťa blok b_1 smerom doprava, vchádza do bloku b_2 vo fáze i_2 a opúšťa ho smerom doľava, pretože sa doňho nevráti pred opätovným navštívením bloku

b_1. Vo fáze j_1 sa nachádza opäť v bloku b_1, no a do bloku b_2 sa nemôže vrátiť sprava, pretože je od neho naľavo. To je spor s tvrdením, že $e_2 \in R_v$. Situáciu ilustruje obrázok 2.10. □



Obr. 2.10: Ilustrácia k dôkazu Lemmy 4

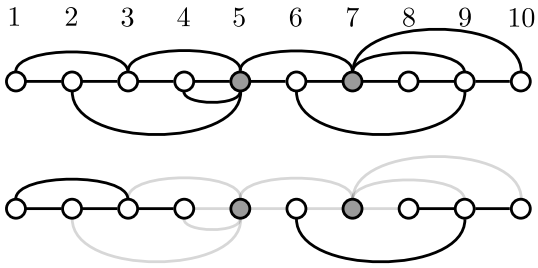
Poznámka 3. *Všimnime si, že grafy z triedy $H_r(n)$ môžu mať vo všeobecnosti (pretože sú to grafy s násobnými hranami) ľubovoľne veľa hrán. Ale každý graf, ktorý vznikne ako graf výpočtu nejakého k-páskového DTS pracujúceho v blokoch a teda patrí do triedy $H_{(2k+1)}(a(n))$, bude mať hrán najviac $(2k+1)a(n)$. Túto vlastnosť budeme ďalej predpokladať všeobecne pre grafy z triedy $H_r(n)$, pretože s inými, ako s grafmi výpočtov, nebudeme pracovať.*

3. Veta o existencii m-segregátoru

Definícia 8. *V grafe $G = (V, E)$ je vrchol v_1* priamym predchodcom *vrcholu v_2, ak $(v_1, v_2) \in E$. Vrchol v_1 je* predchodcom *vrcholu v_2, $v_2 \neq v_1$, ak existuje v grafe G cesta z v_1 do v_2.*

Definícia 9. *Nech $G = (V, E)$ je ľubovoľný orientovaný acyklický graf, m kladné číslo. Množina vrcholov $J \subset V$ sa nazýva* m-segregátor *pre G, ak každý vrchol v $G \setminus J$ má najviac m predchodcov v $G \setminus J$.*

Obrázok 3.1 zobrazuje 3-segregátor veľkosti 2 pre graf $G \in H(10)$



Obr. 3.1: Množina $J = \{5, 7\}$ tvorí v grafe $G \in H(10)$ 3-segregátor veľkosti 2

Definícia 10. *Definujme funkciu $\log^{(k)}(n)$ nasledovne :*

- $\log^{(0)}(n) = n$
- $\log^{(k+1)}(n) = \log_2(\log^{(k)}(n))$

Potom definujeme funkciu iterovaného logaritmu $\log^* : \mathbb{N} \to \mathbb{N}$ *ako*

$$\log^* n = min\{k \in \mathbb{N} \mid \log^{(k)}(n) \leq 1\}$$

Veta dokázaná v tejto kapitole hovorí, že pre grafy z triedy $H_r(n)$, čo sú práve naše grafy výpočtov, existuje m-segregátor s vhodne obmedzenou veľkosťou. Jej formulácia znie nasledovne

Veta 4. *Nech orientovaný acyklický graf $G = (V, E)$ je z triedy $H_r(n)$ a nech $n \geq 16$. Potom G má $\frac{6n}{\log^* n}$-segregátor veľkosti najviac $\frac{15rn}{\log^* n}$.*

Zadefinujeme indukciou pomocnú funkciu $e(k, i)$ nasledovne

- $e(k, 0) = 1$
- $e(k, i + 1) = k^{2+2e(k,i)}$

a definujeme iterovanú exponenciálu ako sprava inverznú funkciu k iterovanému logaritmu

- $tower_2(0) = 1$
- $tower_2(k + 1) = 2^{tower_2(k)}$

Triviálne platí $\log^*(tower_2(k)) = k$. Naopak, $tower_2(\log^*(n) - 1) \leq n$ pre $n \geq 2$.

K dôkazu vety o existencii segregátoru budeme potrebovať nasledovnú lemmu.

Lemma 5. *Pre každé $n \in \mathbb{N}$, $n \geq 16$ a $k = \lceil \frac{\log^* n}{3} \rceil$ existujú $d_0, d_1, \ldots, d_k \in \mathbb{N}$ také, že platí nasledovné*

1. $d_0 = 1$, $d_k \leq 2n$
2. $d_0 \mid d_1 \mid \ldots \mid d_k$
3. $\forall i > 0 \quad \lceil \frac{n}{d_{i+1}} \rceil \leq \log_k \lceil \frac{n}{d_i} \rceil - 1$

Dôkaz. Podmienku 3. môžeme alternatívne upraviť na tvar $k^{\lceil \frac{n}{d_{i+1}} \rceil} \leq \frac{\lceil \frac{n}{d_i} \rceil}{k}$.

Dokážeme indukciou, že pre $i \in \{0, 1, 2, \ldots, k-1\}$ platí

$$e(k, i) \leq tower_2(k + 2i - 1)$$

Z toho potom vyplýva, že

$$e(k, k-1) \leq tower_2(k+2(k-1)-1) = tower_2(3\lceil \frac{\log^* n}{3} \rceil - 3) \leq tower_2(\log^*(n)-1) \leq n$$

- Pre $i = 0$ platí $e(k, 0) = 1 \leq tower_2(k-1)$ triviálne. Všimnime si, že $k \geq 2$, keďže $n \geq 16$.
- Nech podľa indukčného predpokladu $e(k, i) \leq tower_2(k + 2i - 1)$. Potom nasledovnými úpravami dostávame

  $$e(k, i+1) = k^{2+2e(k,i)} \leq (2^k)^{2+2e(k,i)} = 2^{2k+2k.e(k,i)} \leq 2^{2k+2k.tower_2(k+2i-1)}$$

  Využijeme, že pre $k \geq 2$ platí $k \leq tower(k + 2i - 1)$, preto

  $$e(k, i+1) \leq 2^{2tower_2(k+2i-1)+2k.tower_2(k+2i-1)} \leq 2^{[2tower_2(k+2i-1)]^2}$$

  Ďalej pre $k > 4$ platí $[2tower_2(k + 2i - 1)]^2 \leq tower_2(k + 2i)$. Preto

  $$e(k, i+1) \leq 2^{tower_2(k+2i)} = tower_2(k + 2i + 1) = tower_2(k + 2(i + 1) - 1)$$

  Pre $k \in \{2, 3, 4\}$ a $i = 1, \ldots, k-1$ musíme platnosť tvrdenia dokázať priamym dosadením kombinácií hodnôt k a i, pretože použité odhady sú príliš hrubé a funkcia $tower_2$ začína rásť výrazne rýchlo až od čísla 4.

Tým sme dokázali, že $e(k, 0) \leq e(k, 1) \leq \ldots \leq e(k, k-1) \leq n$ pre $k \geq 2$. Pokračujme v dôkaze lemmy. Definujme čísla d_i nasledovne :

- $d_0 = 1$
- $d_i = 2^{\lceil \log_2(\frac{n}{e(k,k-i)}) \rceil}$ pre $1 \leq i \leq k$

Potom je splnená podmienka 1. triviálne, pre podmienku 2. si stačí všimnúť, že d_i je neklesajúca postupnosť mocnín čísla 2, keďže funkcia $e(k, l)$ je rastúca v druhom parametri, ktorý so stúpajúcim i klesá a preto celý zlomok rastie.

Overíme poslednú podmienku z lemmy:
Platí

$$d_{i+1} = 2^{\lceil \log_2(\frac{n}{e(k,k-i-1)}) \rceil} \geq 2^{\log_2(\frac{n}{e(k,k-i-1)})} = \frac{n}{e(k,k-i-1)}$$

z čoho vyplýva, že

$$\lceil \frac{n}{d_{i+1}} \rceil \leq \lceil \frac{n}{\frac{n}{e(k,k-i-1)}} \rceil = \lceil e(k,k-i-1) \rceil = e(k,k-i-1)$$

Ďalšími úpravami dostávame

$$k^{\lceil \frac{n}{d_{i+1}} \rceil} \leq k^{e(k,k-i-1)} \leq k^{2e(k,k-i-1)} = \frac{k^{2+2e(k,k-i-1)}}{k^2} = \frac{e(k,k-i)}{k^2}$$

Potom platí $e(k,k-i) \leq \lceil \frac{2n}{d_i} \rceil$, pretože

$$\lceil \frac{2n}{d_i} \rceil \geq \frac{2n}{2^{\lceil \log_2(\frac{n}{e(k,k-i)}) \rceil}} \geq \frac{2n}{2.2^{\log_2(\frac{n}{e(k,k-i)})}} = \frac{2n}{2\frac{n}{e(k,k-i)}} = e(k,k-1)$$

Pokračovaním ďalej dostávame

$$k^{\lceil \frac{n}{d_{i+1}} \rceil} \leq \frac{e(k,k-i)}{k^2} \leq \frac{\lceil \frac{2n}{d_i} \rceil}{k^2} \leq \frac{\lceil \frac{n}{d_i} \rceil}{k}$$

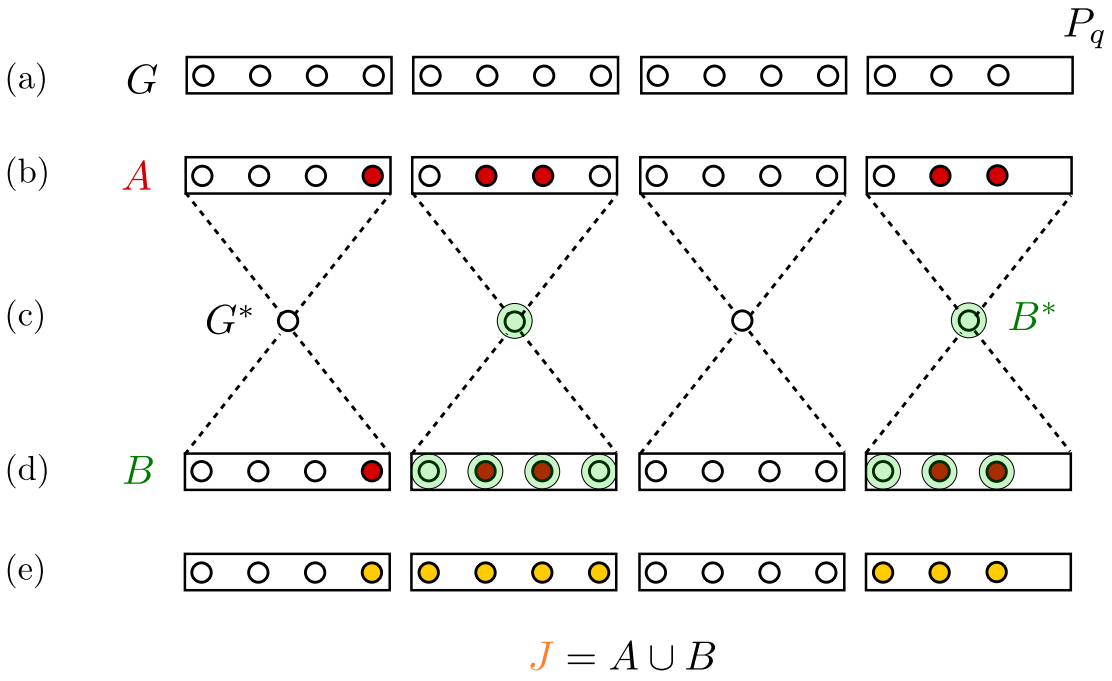
čím je dôkaz lemmy dokončený. Posledná nerovnosť platí v prípade, že $k \geq 2$, čím je určené počiatočné obmedzenie $n \geq 16$. □

Dôkaz samotnej vety o existencii m-segregátoru je technicky pomerne zložitý. Predveďme na úvod stratégiu, pomocou ktorej ho budeme realizovať. Tento rámcový postup je zobrazený na diagrame 3.2.

Rozdelíme vrcholy grafu G na skupiny po sebe idúcich vrcholov, pričom každá zo skupín bude obsahovať rovnaký počet vrcholov (až na poslednú). Možných rozdelení na skupiny vrcholov budeme uvažovať viac, budeme ich nazývať partíciami P_i a z nich jednu vybranú partíciu označíme ako P_q. Na obrázku 3.2 je partícia P_q zobrazená v riadku (a).

Pomocou tejto partície P_q vyberieme množinu A, ktorá bude tvoriť časť hľadaného segregátoru (riadok (b)). Partíciu P_q si preto budeme vyberať opatrne – s ohľadom na to, že pomocou nej budeme vyrábať časť segregátoru, ktorý nesmie byť príliš veľký.

Ku grafu $G \setminus A$ vytvoríme asociovaný graf G^*, jeho vrcholy budú tvorené jednotlivými blokmi partície P_q, z nich vyberieme podmnožinu B^* (riadok (c)), ktorú spätne prenesieme na množinu vrcholov B v grafe G (riadok (d)). Toto $B \subseteq G$ bude tvoriť druhú časť segregátoru $J = A \cup B$ (riadok (e)).



Obr. 3.2: Schéma vytvárania m-segregátoru pre dôkaz vety 4

Jednotlivé kroky, ktoré zabezpečia, že nájdená množina $J = A \cup B$ je vhodným segregátorom s vhodnou veľkosťou, sú popísané v nasledujúcom dôkaze. Pre prehľadnosť zopakujme znenie vety:

Veta 4. *Nech orientovaný acyklický graf $G = (V, E)$ je z triedy $H_r(n)$ a nech $n \geq 16$. Potom G má $\frac{6n}{\log^* n}$-segregátor veľkosti najviac $\frac{15rn}{\log^* n}$.*

Dôkaz. Nech k a $d_0, d_1, \ldots, d_k$ sú prirodzené čísla definované v lemme 5. Pre každé j, $0 \leq j \leq k$ zadefinujeme rozdelenie vrcholov P_j tak, že v každom bloku partície P_j (až na posledný) bude za sebou idúcich d_j vrcholov. Ak napríklad $d_2 = 4$ a $n = 10$, potom $P_2 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \{9, 10\}\}$.

Ku každej hrane $(v_1, v_2) \in E$ asociujeme partíciu P_j s najvyšším možným $j \leq k - 1$ takú, že v_1 a v_2 sa nachádzajú v P_j v rôznych blokoch. Rozdelenie na partície a asociáciu hrán ilustruje obrázok 3.3.

Spomedzi partícií P_j pre $0 \leq j \leq k - 1$ vyberieme takú, ktorá má k sebe asociovaných najviac $\frac{rn}{k}$ hrán. Taká partícia musí existovať, pretože hrán je v grafe z triedy $H_r(n)$ najviac rn (pozri poznámku 3) a počet partícií s ktorými asociujeme hrany je k.

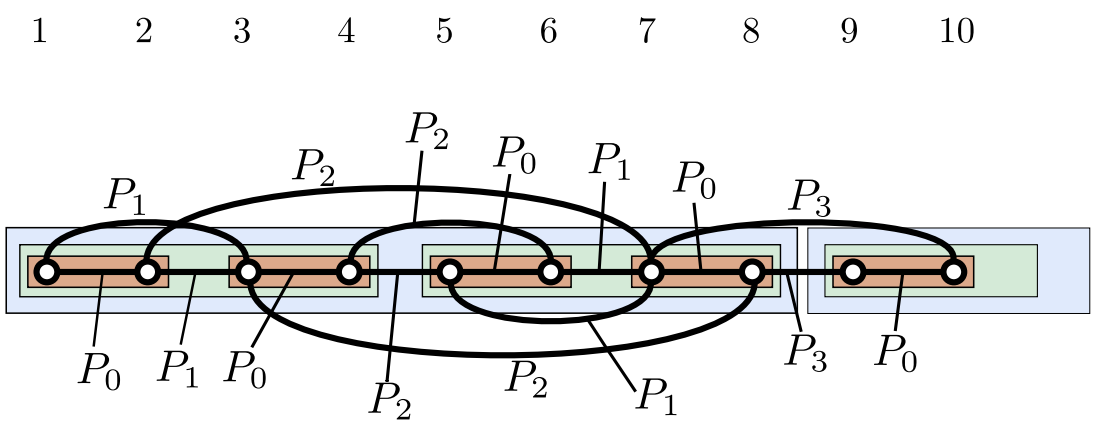
Nech A je množina vrcholov, ktorá obsahuje tie vrcholy v_1, pre ktoré existuje hrana $(v_1, v_2) \in E$ asociovaná s partíciou P_q. Zrejme platí, že $|A| \leq \frac{rn}{k} \leq \frac{3rn}{\log^* n}$.

Z grafu $G \setminus A$ zostrojíme graf $G^* = (V^*, E^*)$. Každý z jeho vrcholov bude reprezentovať jeden z blokov partície P_q a medzi vrcholmi X^* a Y^* bude hrana, ak $G \setminus A$ obsahuje hranu z nejakého vrcholu v bloku X partície P_q do nejakého vrcholu v bloku Y. Počet blokov v partícii P_q, t.j. počet vrcholov grafu G^* je $\lceil \frac{n}{d_q} \rceil$.

Označme ako $D(Y^*)$ počet priamych predchodcov vrcholu Y^* v grafe G^*, t.j.

$$D(Y^*) = |\{X^* \in V^* | (X^*, Y^*) \in E^*\}|$$

Chceli by sme odhadnúť $\sum_{Y^* \in G^*} D(Y^*)$.



$$
\begin{aligned}
d_0 = 1 \quad & P_0 = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}\} \\
d_1 = 2 \quad & P_1 = \{\{1,2\},\{3,4\},\{5,6\},\{7,8\},\{9,10\}\} \\
d_2 = 4 \quad & P_2 = \{\{1,2,3,4\},\{5,6,7,8\},\{9,10\}\} \\
d_3 = 8 \quad & P_3 = \{\{1,2,3,4,5,6,7,8\},\{9,10\}\} \\
d_4 = 16 \quad & P_4 = \{\{1,2,3,4,5,6,7,8,9,10\}\}
\end{aligned}
$$

Obr. 3.3: Rozdelenie grafu $G \in H(10)$ na partície a asociácia hrán

Keďže graf G je z triedy $H_r(n)$, existujú $G_1, G_2, \ldots, G_r \in H_1(n)$ také, že $G = \bigcup_{i=1}^{r} G_i$. Rovnakým spôsobom, ako sme zostrojili graf G^* z grafu G, môžeme podľa už vybranej partície P_q zostrojiť grafy G_i^* pre každý z G_i. Platí $G^* = \bigcup_{i=1}^{r} G_i^*$. Podobne, ako sme definovali počet priamych predchodcov vrcholu Y^* v grafe G^*, môžeme definovať aj počet priamych predchodcov v jednotlivých grafoch G_i^* a označiť ho ako $D_i(Y^*)$. Dokážeme, že platí

$$\sum_{X^* \in V^*} D_i(X^*) \leq 2\lceil \frac{n}{d_q} \rceil$$

Potom už priamo dostávame potrebný odhad

$$\sum_{Y^* \in G^*} D(Y^*) \leq \sum_{i=1}^{r} \sum_{X^* \in V^*} D_i(X^*) \leq 2r\lceil \frac{n}{d_q} \rceil$$

Uvažujme nad vrcholmi grafu G_i^*. Nech X^* je v G_i^* priamym predchodcom Y^* aj Z^* takých, že $Y^* < Z^*$. Potom X^* je zároveň najmenším predchodcom Y^*, pretože ak by existoval vrchol $W^* < X^*$ ako priamy predchodca Y^*, potom by sa hrany (W^*, Y^*) a (X^*, Z^*) krížili. A ak sa krížia hrany v grafe G_i^*, potom sa krížia aj niektoré hrany v grafe G_i, na základe ktorých hrany v G_i^* vznikli. To je však spor s predpokladom, že $G_i \in H_1(n)$. Preto každé dva uzly majú najviac jedného spoločného priameho predchodcu. Ak zoberieme počet priamych predchodcov každého vrcholu zmenšený o jedna (jeden možný spoločný predchodca), tento súčet musí byť menší ako počet vrcholov grafu G_i^*. Ten, ako vieme, je $\lceil \frac{n}{d_q} \rceil$.

$$\sum_{X^* \in V^*} (D_i(X^*) - 1) \leq \lceil \frac{n}{d_q} \rceil$$

a keďže počet vrcholov v V^* je najviac $\lceil \frac{n}{d_q} \rceil$, dostávame hľadaný odhad

$$\sum_{X^* \in V^*} D_i(X^*) \leq 2\lceil \frac{n}{d_q} \rceil$$

Označme B^* množinu vrcholov z G^*, ktoré majú počet priamych predchodcov väčší ako k

$$B^* = \{X^* \in V^* | D(X^*) > k\}$$

a platí

$$|B^*| \leq \frac{\sum_{X^* \in V^*} D(X^*)}{k} \leq \frac{2r}{k} \lceil \frac{n}{d_q} \rceil$$

Množinu vrcholov $B^* \subset V^*$ spätne prenesieme do grafu G, t.j. zadefinujeme množinu $B \subset V$ takú, že $v \in B$ práve vtedy, keď sa v nachádza v takom bloku, ktorého obraz v G^* je v množine B^*.

Odhadneme veľkosť množiny B pomocou odhadu veľkosti B^*. Každý blok z P_q obsahuje najviac d_q vrcholov a platí $\lceil \frac{n}{d_q} \rceil d_q \leq 2n$, preto

$$|B| \leq d_q |B^*| \leq \frac{2r}{k} \lceil \frac{n}{d_q} \rceil d_q \leq \frac{4rn}{k} \leq \frac{12rn}{\log^* n}$$

Segregátor J zvolíme ako $J = A \cup B$. Jeho veľkosť máme odhadnutú ako

$$|J| \leq |A| + |B| \leq \frac{3rn}{\log^* n} + \frac{12rn}{\log^* n} = \frac{15rn}{\log^* n}$$

čo spĺňa požiadavku zo zadania vety.

Každá hrana v G^* vznikla z niektorej hrany $e = (v_1, v_2)$ v grafe $G \setminus A$. Vrcholy tejto hrany e sa musia nachádzať v rôznych blokoch partície P_q, inak by v G^* patrili pod rovnaký vrchol a nevznikla by z nich žiadna hrana. Zároveň sa musia nachádzať v rôznych blokoch partície P_{q+1}. V prípade, že by sa nachádzali v rovnakom, bola by partícia P_q najvyššia taká, že v_1 a v_2 sú v rôznych blokoch, hrana e by sa asociovala k partícii P_q a vrchol v_1 by padol do množiny A, čo by znamenalo, že sa táto hrana v grafe $G \setminus A$ vôbec neobjaví. Preto najdlhšia spojitá cesta v grafe G^* je tvorená najviac $\lceil \frac{n}{d_{q+1}} \rceil$ vrcholmi a jej dĺžka je najviac $\lceil \frac{n}{d_{q+1}} \rceil - 1$.

Pre každý vrchol grafu $G^* \setminus B^*$ podľa jeho definície platí, že má najviac k priamych predchodcov. Keď vezmeme do úvahy, že najdlhšia možná cesta v G^* má dĺžku maximálne $\lceil \frac{n}{d_{q+1}} \rceil - 1$, potom počet všetkých predchodcov vrcholu v grafe $G^* \setminus B^*$ bude menší ako $k^{\lceil \frac{n}{d_{q+1}} \rceil}$, čo je podľa lemmy 5 a vďaka výberu čísiel d_i menšie ako $\frac{\lceil \frac{n}{d_q} \rceil}{k}$

Teraz môžeme odhadnúť počet predchodcov každého vrcholu v $G \setminus J$. Nech vrchol v_x v bloku X partície P_q je predchodcom vrcholu v_y v bloku Y. Potom v grafe $G^* \setminus B^*$ je X^* predchodcom Y^*. Každý blok partície P_q má najviac d_q vrcholov. Počet možných predchodcov vrcholu v grafe $G^* \setminus B^*$ je, ako sme dokázali, menší ako $\frac{\lceil \frac{n}{d_q} \rceil}{k}$. Preto najvyšší počet možných predchodcov vrcholu v grafe $G \setminus J$ je

$$d_q \frac{\lceil \frac{n}{d_q} \rceil}{k} \leq \frac{2n}{k} \leq \frac{6n}{\log^* n}$$

Tým je veta dokázaná.

□

4. $DLIN \neq NLIN$

V tejto kapitole sa dostaneme k dôkazu tvrdenia, že nedeterminizmus pridáva výpočtovú silu pre jazyky akceptované v lineárnom čase. Dôkaz je nekonštruktívny a prebieha sporom. Dokážeme, že pre každý jazyk z triedy $DTIME(t(n))$ existuje alternujúci Turingov stroj, ktorý tento jazyk akceptuje v čase $O(t(n)/\log^* t(n))$, čiže možnosť alternácie dokáže nekonštantne urýchliť deterministický výpočet.

Potom ukážeme, že za predpokladu $DLIN = NLIN$ platí $DTIME(t(n)) = \Sigma_k\text{-}TIME(t(n))$ pre ľubovoľné k a vhodné $t(n)$, teda že ak nedeterminizmus výpočtovú silu pre jazyky v lineárnom čase nepridá, tak ju nepridá ani ľubovoľný konečný počet alternácií a to aj pre časovú zložitosť vyššiu ako lineárnu.

Tieto dva výsledky nám umožnia dôjsť ku sporu, pretože ak sa dá každý výpočet alternujúceho stroja nahradiť deterministickým, a ten sa dá ďalej pomocou alternácie lepšie ako konštantne zrýchliť, potom nemôže platiť dokázaná veta 2 o časovej hierarchii pre alternujúce Turingove stroje.

Dokážme najprv vetu o zrýchlení deterministického výpočtu pomocou alternujúceho Turingovho stroja.

Veta 5. *Pre ľubovoľné časovo konštruovateľné $t(n) \geq n \log^* n$ platí $DTIME(t(n)) \subseteq \Sigma_4\text{-}TIME(t(n)/\log^* t(n))$.*

Dôkaz. Nech L je jazyk patriaci do triedy $DTIME(t(n))$. Potom existuje k-páskový deterministický Turingov stroj M_d akceptujúci každé slovo z jazyka L dĺžky n v čase $t(n)$.

Podľa lemmy 3 k nemu existuje pre dané $a(n) = \lceil t(n)^{\frac{1}{3}} \rceil$ a $b(n) = \lceil t(n)^{\frac{2}{3}} \rceil$ deterministický Turingov stroj M_b s $k+1$ páskami, ktorý pracuje v blokoch dĺžky $b(n)$ a akceptuje rovnako v čase $O(t(n))$.

Zatiaľ platí $L = L(M_d) = L(M_b)$. Zostrojíme alternujúci Turingov stroj, pracujúci v čase $O(t(n)/\log^* t(n))$, ktorý bude simulovať činnosť blokového stroja M_b a urobí štyri prechody medzi univerzálnymi a existenčnými stavmi. V simulácii využijeme vetu 4 o existencii m-segregátoru s veľkosťou $e(n) = \lceil 15(2k+3)a(n)/\log^* a(n) \rceil$, kde $m = m(n) = \lceil 6a(n)/\log^* a(n) \rceil$.

1. *Existenčná fáza.*

   (a) Existenčne uhádneme čísla blokov, v ktorých sa počas výpočtu stroja M_b budú nachádzať hlavy na koncoch časových segmentov. Jednotlivých časových segmentov je $a(n)$, hláv je k a zápis čísla bloku na páske prebehne v čase $\log a(n)$ (blokov na páske je určite najviac toľko, ako časových segmentov). Celý tento krok prebehne v čase $O(a(n) \log a(n))$.

   (b) Deterministicky skonštruujeme graf výpočtu G podľa uhádnutých pozícií v predchádzajúcom kroku. Tento graf má $a(n)$ vrcholov a pre každý vrchol nájdeme v čase $O(a(n))$ všetky hrany, ktoré doňho smerujú. Celkový čas tohto kroku preto bude $O(a(n)^2)$.

   (c) Existenčne uhádneme množinu vrcholov J veľkosti najviac $e(n)$, ktorá bude kandidátom na segregátor v grafe G. Podľa vety o existencii segregátoru vieme, že takýto segregátor existuje a v niektorej vetve

výpočtu ho uhádneme. Vrcholov je $a(n)$, z ktorých najviac $e(n) \in O(a(n))$ označujeme, čas na to potrebný je teda $O(a(n))$

(d) Nech v je posledný vrchol grafu výpočtu. Označme $J_v = J \cup \{v\}$. Pre každú blokovú fázu z J_v existenčne uhádneme obsah každej pásky a stav stroja na začiatku daného bloku a následne hádame ďalšie konfigurácie – stavy, pohyby hláv a nové znaky na páskach – a deterministicky overujeme ich konzistentnosť s prechodovou funkciou. To robíme až do konca bloku, t.j. spolu $b(n) - 1$ krát. Začiatočnú a koncovú konfiguráciu v bloku si pre každý vrchol segregátoru (a posledný vrchol grafu výpočtu) uložíme, vnútorné konfigurácie nebudeme potrebovať. Dĺžka bloku je $b(n)$, veľkosť množiny J_v je $e(n)+1$, celkový čas potrebný na uhádnutie výpočtov a ich overenie bude $O(b(n).e(n)) = O(b(n)a(n)/\log^* a(n)) \subseteq O(t(n)/\log^* t(n))$.

(e) Deterministicky overíme, či koncová konfigurácia v poslednom vrchole v končí akceptujúcim stavom. Ak nie, vstupné slovo odmietneme.

2. *Univerzálna fáza.* Univerzálne vyberieme vrchol $j \in J_v$. Deterministicky vypočítame množinu I_j všetkých predchodcov vrcholu j v grafe $G \setminus (J_v \setminus \{j\})$. Túto množinu budú tvoriť predchodcovia vrcholu j, takže za predpokladu, že vo fáze 1. sme vybrali $m(n)$-segregátor, nesmie veľkosť tejto množiny presiahnuť $m(n)$. To síce v tomto momente nemáme ako overiť, pretože číslo $m(n)$ nestíhame dostatočne rýchlo skonštruovať, vieme však, že takýto segregátor existuje a preto ho v niektorej vetve výpočtu nájdeme. V tomto momente je kľúčová slabá definícia časovej zložitosti pre alternujúce Turingove stroje.

3. *Existenčná fáza.* Pre každý časový segment v množine I_j existenčne uhádneme k nemu odpovedajúci výpočet stroja M_b, rovnakým spôsobom, ako v kroku (1d) prvej existenčnej fázy. Veľkosť $|I_j| = m(n) = \lceil 6a(n)/\log^* a(n) \rceil$, dĺžka bloku je $b(n)$ a preto čas priebehu tejto fázy bude
$O(b(n)m(n)) \subseteq O(a(n)b(n)/\log^* a(n)) \subseteq O(t(n)/\log^* t(n))$.

4. *Univerzálna fáza.* Univerzálne vyberieme vrchol $i \in I_j \cup \{j\}$. Ostáva overiť, či uhádnutá začiatočná konfigurácia časového segmentu i je podľa prechodovej funkcie stroja M_b konzistentná s konečnými konfiguráciami všetkých jeho priamych predchodcov v grafe G. Pre vrchol v grafe výpočtu a jeho priameho predchodcu platí, že hlava sa na páske do bloku ku ktorému tieto vrcholy patria medzi týmito časovými fázami nedostane, a preto sa dané konfigurácie musia zachovať (nemá ich čo zmeniť). V prípade, že niektoré z týchto overovaní zlyhá, vstup odmietneme, v opačnom prípade môžeme akceptovať. Pre pevné i je veľkosť množiny $|I_j| \leq e(n)$ a počet možných predchodcov každého vrcholu je najviac $a(n)$, preto deterministické overenie prebehne v čase
$O(a(n)e(n)) = O(a(n)^2/\log^* a(n)) \subseteq O(t(n)/\log^* t(n))$.

Každá fáza a preto aj celý výpočet prebehne v čase $O(t(n)/\log^* t(n))$.

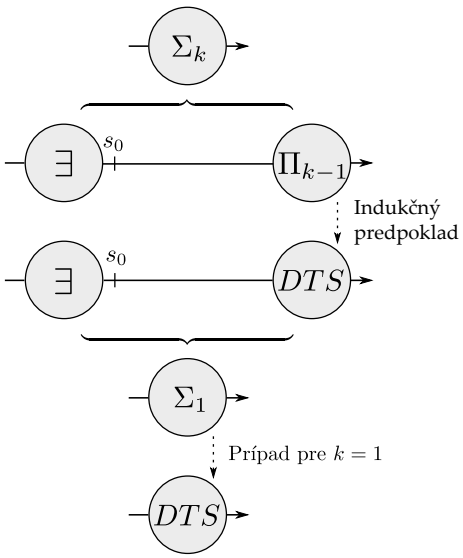
□

Lemma 6. *Ak $DLIN = NLIN$, potom pre ľubovoľné k platí $DLIN = \Sigma_k\text{-}TIME(n)$ (a zároveň $DLIN = \Pi_k\text{-}TIME(n)$).*

Dôkaz. Lemmu dokážeme indukciou vzhľadom na počet alternácií k.

Nech $k = 1$. Potom je jazyk akceptovaný takým alternujúcim Turingovým strojom M, ktorý má všetky stavy rovnakého typu. V prípade, že všetky stavy sú existenčné, je tento Turingov stroj nedeterministický, teda $L(M) \in NLIN$ a podľa predpokladu $NLIN = DLIN$ aj $L(M) \in DLIN$.
Ak sú všetky stavy univerzálne, potom doplnok $\overline{L(M)} \in NLIN$, podľa predpokladu $\overline{L(M)} \in DLIN$ a triviálne aj $L(M) \in DLIN$.

Nech je L jazyk akceptovaný alternujúcim Turingovým strojom M, ktorý urobí počas výpočtu k alternácií. Upravíme stavy M tak, aby v nich bol uložený počet alternácií, ktoré počas výpočtu prebehli. Ďalej stroj upravíme tak, aby svoju prvú alternáciu urobil vždy na ľavom okraji pásky a následne prešiel do stavu s_0. Svoj pôvodný stav si zapamätá na páske. Turingov stroj, ktorý začína v stave s_0 urobí už iba $k-1$ alternácií (a akceptuje jazyk v $\Sigma_{k-1}\text{-}TIME(n)$, alebo $\Pi_{k-1}\text{-}TIME(n)$), preto k nemu podľa indukčného predpokladu existuje deterministický Turingov stroj, pracujúci v lineárnom čase. Jeho nahradením dostávame alternujúci Turingov stroj, ktorý má všetky stavy jedného typu, žiadnu alternáciu už neurobí, pracuje v čase $O(n)$ a akceptuje rovnaký jazyk ako pôvodný stroj M. Nakoniec využijeme už dokázané tvrdenie pre $k = 1$.



Obr. 4.1: Indukcia k dôkazu Lemmy 6.

□

Lemma 7. *Ak $DLIN = NLIN$, potom pre ľubovoľné časovo konštruovateľné $t(n)$ a pre ľubovoľné k platí $\Sigma_k\text{-}TIME(t(n)) = DTIME(t(n))$*

Dôkaz. $DTIME(t(n)) \subseteq \Sigma_k\text{-}TIME(t(n))$ platí triviálne.

Naopak, nech $L \in \Sigma_k\text{-}TIME(t(n))$. Skonštruujeme jazyk $L' = \{\omega\#^{t(|\omega|)} | \omega \in L\}$ kde $\#$ nie je symbolom v abecede jazyka L. Potom $L' \in \Sigma_k\text{-}TIME(n)$, podľa lemmy 6 $L' \in DTIME(n)$ a preto $L \in DTIME(t(n))$ □

Veta 6. $DLIN \subsetneq NLIN$

Dôkaz. $DLIN \subseteq NLIN$ je triviálne platné z definície. Predpokladajme, že $DLIN = NLIN$. Potom z lemmy 7 pre $t(n) = n \log^* n$ a $k = 4$ dostávame platnosť

$$\Sigma_4\text{-}TIME(n \log^* n) = DTIME(n \log^* n) \tag{4.1}$$

a z vety 5 vieme, že deterministický výpočet sa dá pomocou alternujúceho Turingovho stroja urýchliť, t.j.

$$DTIME(n \log^* n) \subseteq \Sigma_4\text{-}TIME(n). \tag{4.2}$$

Spojením obidvoch vzťahov dostávame

$$\Sigma_4\text{-}TIME(n \log^* n) \subseteq \Sigma_4\text{-}TIME(n) \tag{4.3}$$

čo je v spore s vetou 2 o alternujúcej hierarchii, keďže $n \in o(n \log^* n)$. □

5. Záver a rozšírenia

Na záver spomenieme niekoľko výsledkov, ktoré sa našej práce priamo, alebo nepriamo dotýkajú.

5.1 Zrýchlená simulácia deterministického výpočtu pomocou Σ_2-stroja

Balcazar v cvičeniach ku tretej kapitole knihy *Structural Complexity II* [2] uviedol, že v simulácii z vety 5 o zrýchlení deterministického Turingovho stroja pomocou Σ_4-stroja stačia dve alternácie namiesto štyroch a teda použitý stroj bude typu Σ_2.

Toto postulované tvrdenie dokázal S. Gupta v [4] spolu s dôsledkom, že dve alternácie sú ostro silnejšie, ako deterministický čas, t.j.

$$DTIME(t(n)) \subsetneq \Sigma_2\text{-}TIME(t(n))$$

pre každé časovo konštruovateľné $t(n) \geq n$.

Ukážeme, ako sa dá výpočet deterministického Turingovho stroja zrýchliť pomocou dvoch alternácií a naznačíme, ako z toho vyplýva uvedený dôsledok.

Veta 7. *Pre každú časovo konštruovateľnú funkciu $t(n) > n \log^* n$ platí $DTIME(t(n)) \subseteq \Sigma_2\text{-}TIME(t(n)/\log^* t(n))$.*

Dôkaz. Predpoklady k simulácii pomocou Σ_2 stroja, ktorú predvedieme, sú rovnaké, ako v dôkaze pre Σ_4 stroje vo vete 5. Pre každý jazyk $L \in DTIME(t(n))$ existuje deterministický Turingov stroj M_d, akceptujúci jazyk L v čase $t(n)$ a k nemu existuje deterministický Turingov stroj M_b pracujúci v blokoch dĺžky $b(n)$ pre dané $a(n) = \lceil t(n)^{\frac{1}{3}} \rceil$ a $b(n) = \lceil t(n)^{\frac{2}{3}} \rceil$ akceptujúci jazyk L v čase $O(t(n))$. Rovnako využijeme platnosť vety 4 o existencii m-segregátoru veľkosti $e(n) = \lceil 15(2k+3)a(n)/\log^* a(n) \rceil$, kde $m = m(n) = \lceil 6a(n)/\log^* a(n) \rceil$.

Zostrojíme alternujúci Turingov stroj typu Σ_2 akceptujúci jazyk L v čase $t(n)/\log^* t(n)$.

1. *Existenčná fáza.* Prvá fáza ostáva oproti dôkazu pre Σ_4 stroj bez zmeny. Preto iba stručne :

   (a) Uhádneme polohu každej hlavy v jednotlivých blokoch počas výpočtu stroja M_b.

   (b) Zostrojíme graf výpočtu podľa uhádnutých polôh hláv v predchádzajúcom kroku.

   (c) Uhádneme množinu $J \subseteq G$ ako kandidáta na segregátor.

   (d) Pre každý vrchol v segregátore J a posledný vrchol grafu výpočtu v uhádneme začiatočnú konfiguráciu v tomto bloku a odsimulujeme deterministický výpočet stroja M_b. Začiatočné a koncové konfigurácie si zapamätáme na pomocných páskach, označíme ich pre vrchol j ako $Init_J(j)$ a $Final_J(j)$.

(e) Overíme, či koncová konfigurácia v poslednom vrchole grafu výpočtu v končí v akceptujúcom stave. Ak nie, vstup odmietneme.

2. *Univerzálna fáza.*

(a) Univerzálne vyberieme $j \in J_v$. Deterministicky vypočítame množinu I_j všetkých predchodcov vrcholu j v grafe $G \setminus (J \setminus \{j\})$. Overíme, či je posledný pohyb hláv v konfigurácii $Final_J(j)$ konzistentný s uhádnutými polohami hláv v kroku (1a), ak nie je, odmietame.

(b) Postupne budeme podľa časovej následnosti prechádzať vrcholy v I_j a pomocou konfigurácií $Final_J$ simulovať priebeh výpočtu. Vrcholy v I_j majú svojich priamych predchodcov buď v množine I_j, alebo v množine J.

Pre vrcholy z množiny J poznáme konfigurácie $Final_J$, ktoré počas simulácie využijeme - t.j. nebudeme začiatočné konfigurácie hádať, ako v dôkaze pre Σ_4 stroje.

Pre vrcholy z množiy I_j budeme koncové konfigurácie postupne v ich časovej následnosti vyrábať pri simulácii. Potrebujeme preto k už odsimulovaným vrcholom množiny I_j pristupovať dostatočne rýchlo. Na to využijeme poznatok, že graf G je typu $H_{2k+1}(a(n))$ (Lemma 4). Pre každú komponentu $G_i \in H_1(a(n)), i \in \{1, \ldots, 2k+1\}$ použijeme jednu pásku, takže celkovo simuláciu prevedieme na $2k+1$ páskach. Každá z hrán $e = (v_1, v_2)$ v grafe G_i hovorí, že simulovaná hlava sa nachádza vo fázach v_1 a v_2 v rovnakom bloku a zároveň sa žiadne hrany nekrížia. Preto môžeme k odsimulovaným blokom pristupovať tak, ako na zásobníku (LIFO). Nekrížením hrán máme zaručené, že pri ceste späť na páske budeme mať potrebný blok na správnej pozícii.

Na rozdelenie grafu G do podmnožín $G_i \in H_1(a(n))$, vybranie potrebných konfigurácií z $Final_J$ a simuláciu blokových fáz množiny I_j je potrebný čas $O(b(n)m(n)) \subseteq O(t(n)/\log^* t(n))$.

Pre vrcholy $j \in J$ takto paralelne odsimulujeme každú časť I_j. Na záver ešte potrebujeme rovnako, ako v poslednej univerzálnej fáze 4 dôkazu vety 5 pre Σ_4 stroje paralelne overiť konzistenciu začiatočných konfigurácií vrcholov $j \in J$ s koncovými konfiguráciami ich priamych predchodcov v grafe G. Keďže J je segregátorom, jeho veľkosť je $e(n)$, počet priamych predchodcov vrcholu nie je väčší ako $a(n)$ a preto overenie prebieha v čase $O(e(n)a(n)) \subseteq O(t(n)/\log^* t(n))$.

□

Veta 8. *Pre každú časovo konštruovateľnú funkciu $t(n)$, $n\sqrt{\log^* n} \in o(t(n))$ je $DTIME(t(n)) \subsetneq \Sigma_2\text{-}TIME(t(n))$.*

Dôkaz. Predpokladajme, že pre niektoré $t(n)$ zo znenia vety platí

$$DTIME(t(n)) = \Sigma_2\text{-}TIME(t(n))$$

Z vety 7 vyplýva, že

$$DTIME(t(n)) \subseteq DTIME(t(n)\sqrt{\log^* t(n)}) \subseteq DTIME(t(n)\log^* t(n)) \subseteq \Sigma_2\text{-}TIME(t(n))$$

a ak $DTIME(t(n)) = \Sigma_2\text{-}TIME(t(n))$, potom sa všetky tieto uvedené množiny rovnajú.

Navyše $DTIME(t(n)) = \overline{DTIME(t(n))}$ a preto aj

$$\Sigma_2\text{-}TIME(t(n)) = \Pi_2\text{-}TIME(t(n)) = DTIME(t(n)\sqrt{\log^* t(n)}) \qquad (\alpha)$$

Keďže $\frac{t(n)}{\sqrt{\log^* t(n)}} \in o(t(n))$ potom podľa poznámky 1 k dôkazu vety 2 platí

$$\Pi_2\text{-}TIME(t(n)) \ \setminus \ \Sigma_2\text{-}TIME(\frac{t(n)}{\sqrt{\log^* t(n)}}) \neq \emptyset$$

Podľa vety 7

$$DTIME(t(n)\sqrt{\log^* t(n)}) \subseteq \Sigma_2\text{-}TIME(\frac{t(n)}{\sqrt{\log^* t(n)}})$$

a potom aj

$$\Pi_2\text{-}TIME(t(n)) \ \setminus \ DTIME(t(n)\sqrt{\log^* t(n)}) \neq \emptyset$$

a to je spor s tým, že sa tieto triedy rovnajú (α). □

Predpoklad $n\sqrt{\log^* n} \in o(t(n))$ nie je v predchádzajúcej vete nevyhnutný a dá sa ho pomocou malej úpravy zbaviť. V našom dôkaze sme ho použili preto, že delenie výrazom $\sqrt{\log^* t(n)}$ by nás dostávalo pod lineárnu časovú zložitosť, kde v plnom znení neplatí väčšina dokázaných tvrdení, ktoré používame. Idea dôkazu však ostáva rovnaká a jeho plná verzia sa nachádza v práci [4]. Potom tvrdenie platí pre každú časovo konštruovateľnú funkciu $t(n)$.

5.2 Oddelenie času a priestoru

Dôležitý výsledok v kategorizácii tried zložitostí je tvrdenie dokázané v článku [7]. Keďže priestorovú zložitosť pre deterministické Turingove stroje sme ešte nedefinovali, urobíme tak teraz.

Definícia 11. *Pre ľubovoľnú funkciu $t(n) \geq n$ definujeme triedu $DSPACE(t(n))$ nasledovne: Jazyk L patrí do triedy $DSPACE(t(n))$, práve vtedy, keď existuje deterministický Turingov stroj M, ktorý na každom akceptovanom vstupe dĺžky n počas výpočtu využije najviac $t(n)$ políčok na všetkých svojich páskach a platí $L = L(M)$.*

Tvrdenie 1. *Pre každú funkciu $t(n)$ platí*

$$DTIME(t(n)) \subseteq DSPACE(t(n)/\log t(n))$$

Použitím vety o priestorovej hierarchii pre deterministické Turingove stroje ako dôsledok dostávame separáciu $DTIME(t(n)) \subsetneq DSPACE(t(n))$ pre každé $t(n) \geq n$.

Nasledujúce tvrdenie je priamym rozšírením vety $DLIN \neq NLIN$.

V článku [13] autor dokazuje pomocou obmedzenia vetvenia stromov výpočtov tvrdenie

Tvrdenie 2. *Pre každú funkciu $t(n)$ takú, že $t(n) \in o(n \log^* n)$ platí $DTIME(t(n)) \neq NTIME(t(n))$.*

Ako dôsledok výberom $t(n) = n\sqrt{\log^* n}$ dostávame platnosť $DTIME(n\sqrt{\log^* n}) \neq NTIME(n\sqrt{\log^* n})$.

Takto vetu $DLIN \neq NLIN$ posúvame na vyššiu ako lineárnu zložitosť. Keďže však nevieme, či sú oddelené triedy $DLIN$ a $DTIME(n\sqrt{\log^* n})$, nevieme ani, či je dokázané tvrdenie skutočne silnejšie.

Druhým výsledkom textu [13] je dôkaz, že aspoň jedno z nasledovných tvrdení platí:

1. $P \neq L$
2. Pre každé polynomiálne ohraničené $t(n)$ platí $DTIME(t(n)) \neq NTIME(t(n))$.

kde P je trieda jazykov akceptovaných deterministickým Turingovým strojom v polynomiálnom čase a L je trieda jazykov akceptovaných deterministickým Turingovým strojom v logaritmickom priestore. Na zavedenie logaritmického priestoru je nutné upraviť model Turingovho stroja, pridať mu jednu vstupnú pásku na ktorú nie je možné zapisovať a v definícii priestorového ohraničenia brať do úvahy iba obsah pracovných pások.

Literatúra

[1] Jose Luis Balcazar, Josep Diaz, and Joaquim Gabarro. *Structural Complexity I.* 1988.

[2] Jose Luis Balcazar, Josep Diaz, and Joaquim Gabarro. *Structural Complexity II.* 1988.

[3] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[4] Sanjay Gupta. Alternating time versus deterministic time: A separation. In *Structure in Complexity Theory Conference*, pages 266–277, 1993.

[5] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[6] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape turing machines. *J. ACM*, 13:533–546, October 1966.

[7] J.E. Hopcroft, W.J. Paul, and Valiant L.G. On time versus space and related problems. In *Proc. 16th IEEE-FOCS 1975*, pages 57–64, Berkeley, 1975.

[8] Stanislav Žák. A turing machine time hierarchy. *Theor. Comput. Sci.*, pages 327–333, 1983.

[9] W. J. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On determinism versus nondeterminism and related problems. In *IEEE Symposium on Foundations of Computer Science*, 1983.

[10] W.J. Paul, E. Prauss, and R. Reischuk. On alternation. 14:243–255, 1980.

[11] W.J. Paul and R. Reischuk. On alternation II. 14:391–403, 1980.

[12] W.J. Paul and R. Reischuk. On time versus space II. 22(3):312–1981, 1981.

[13] Rahul Santhanam. On separators, segregators and time versus space. In *Proceedings of the 16th Annual Conference on Computational Complexity*, pages 286–, Washington, DC, USA, 2001. IEEE Computer Society.