Sveučilište u Zagrebu

Prirodoslovno-matematički fakultet

Mia Filić

Teorem o vremenskoj hijerarhiji

Zagreb, 2017

1 Teorem o vremenskoj hijerarhiji

Teorem 1.1 (Teorem o vremenskoj hijerarhiji)**.** *Za svaku vremenski konstruktibilnu funkciju* $f : \mathbb{N} \to \mathbb{N}$ *postoji jezik* $\boldsymbol{L} \in DTIME(f(n))$*, ali* $\boldsymbol{L}$ *nije odlučiv u vremenu* $o(\frac{f(n)}{log_2(f(n))})$

Zašto proučavati *Teorem o vremenskoj hijerarhiji* i koja je njegova važnost. Upravo spomenuti teorem je dokaz jednoga od najvažnijih rezultata računarske teorije složenosti: *"Bez obzira na složenost problema, uvijek će postojati teži."* Općenito, glavnim mjerama složenosti smatramo: (1) vremensku i (2) prostornu složenost. *Teorem o vremenskoj hijerarhiji* daje dokaz gornjeg rezultata u terminima vremenske, a *Teorem o prostornoj hijerarhiji* u terminima prostorne složenosti [Sip06a].

Inačica teorema o vremenskoj hijerarhiji je više: $DTIME$ ili $NTIME$ o Turingovovom stroju s jednom ili više traka. Sve inačice sadrže istu ideju, ali se razlikuju u detaljima. U ovom radu, bavimo se isključivo $DTIME$ inačicom o Turingovom stroj s jednom trakom koju je moguće pronaći u [Sip06a]. Iskaze nekih drugih inačica teorema je moguće pronaći na kraju u prilogu A.

Sljedeće stranice donose postepeni dokaz odabrane inačice teorema. Kako bi se on što bolje razumio, savjetuje se čitatelja da utvrdi razumijevanje sljedećih pojmova:

1. Vremenski konstruktibilna funkcija,
2. Mala o notacija i velika O notacija,
3. Turingov stroj (TS), vrste i vremenska složenost (DTIME).

čije je definicije moguće pronaći u knjizi [Sip06a].

Organizacija je sljedeća: (1) definicija i analiza vremenske složenosti Turingovih strojeva koji se koriste u dokazu teorema, (2) dokaz teorema gdje se upućuje na nejasnoću koju uvodi, (3) razjašnjavanje spomenute nejasnoće djelomičnom promjenom dokaza teorema.

Napomena 1.1. *Za Turingove strojeve koje definiramo u nastavku podrazumjevamo sljedeće: (1) abeceda ulaza* $\Sigma = \{0, 1\}$*, (2) radna abeceda* $\Gamma = \{0, 1, \sqcup\}$ *gdje* $\sqcup$ *označava prazan znak. Registar stroja koji sadrži* $\sqcup$ *smatramo da je prazan. Skup* Q *čiji su elementi stanja stroja i funkcija prijelaza definiraju se opisom rada stroja. Definiramo Turingove strojeve (TS) - prepoznačivače koji imaju dva završna stanja: (1) stanje prihvaćanja* q_{da}*, (2) stanje odbijanja* q_{ne} *i TS - pretvarače. TS pretvarači su Turingovi strojevi čiji izlaz nije samo završno stanje,* q_{DA} *ili* q_{NE} *nego riječ izlazne abecede stroja. Svrha TS pretvarača je izračunavanje funkcija. Početno stanje je uvijek samo jedno, u oznaci:* q_0*.*
Na početku rada Turingovog stroja pretpostavlja se (1) da je glava za čitanje i pisanje postavljena na početak ulazne riječi, (2) na ulaznoj traci / tragu se nalazi samo ulazna riječ i ostale trake / tragovi su prazni (ukoliko nije naglašeno drugačije), (3) stroj se nalazi u početnom stanju.

U skladu s odabranom inačicom *Teorema o vremenskoj hijerarhiji*, prilikom dokaza teorema nije preporučljivo korištenje višetračnog Turingovog stroja. Višetračni Turingovi strojevi se mogu reducirati na jednotračne prema *Teoremu o redukciji višetračnog* [Vuk16]. Redukcija povećava vremensku složenost za kvadrat. Budući da je bit teorema pokazati da povećanje dopustive vremenske složenosti Turingovih strojeva već za logaritamski faktor odlučuje veću klasu jezika, njezino povećanje za kvadrat nije zanemarivo.[1] Dakle, ne želimo koristiti višetračne TS-ove. Alternativu predstavljaju višetragovni TS-ovi.

Definicija 1.1 (Turingov stroj s više tragova)**.** *Neka je* $k \in \mathbb{N}$*.* ***k-tragovni Turingov stroj*** *je uređena (šest+k)-torka* $(Q, \Sigma, \Gamma_1, \Gamma_2, \ldots, \Gamma_k, \delta, q_0, q_{DA}, q_{NE})$*, gdje je redom:*

- Q *konačan skup koji nazivamo* ***stanja****,*

[1] Povećanje složenosti opisane funkcijom $f(n)$ za kvadrat rezultira složenošću koja se opisuje funkcijom $f(n)^2$.

- Σ *je konačan skup, čije elemente nazivamo ulazni simboli. Pretpostavljamo da ne sadrži "prazan simbol" kojeg označavamo sa* $\sqcup$.

- $(\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$ *uređena k-torka radnih abeceda stroja i nazivamo ju* ***abeceda Turingovog stroja***. Γ_i *je radna abeceda* $i-$*tog traga stroja. Smatramo kako svaki* Γ_i *sadrži svoj istaknuti simbol* $\sqcup$ *koji nazivamo "prazan simbol".*

- $\delta : Q \times \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_k \to Q \times \Gamma_1 \times \Gamma_2 \times \ldots \times \Gamma_k \times \{L, D, S\}$ *proizvoljna funkcija koju nazivamo* ***funkcija prijelaza***.

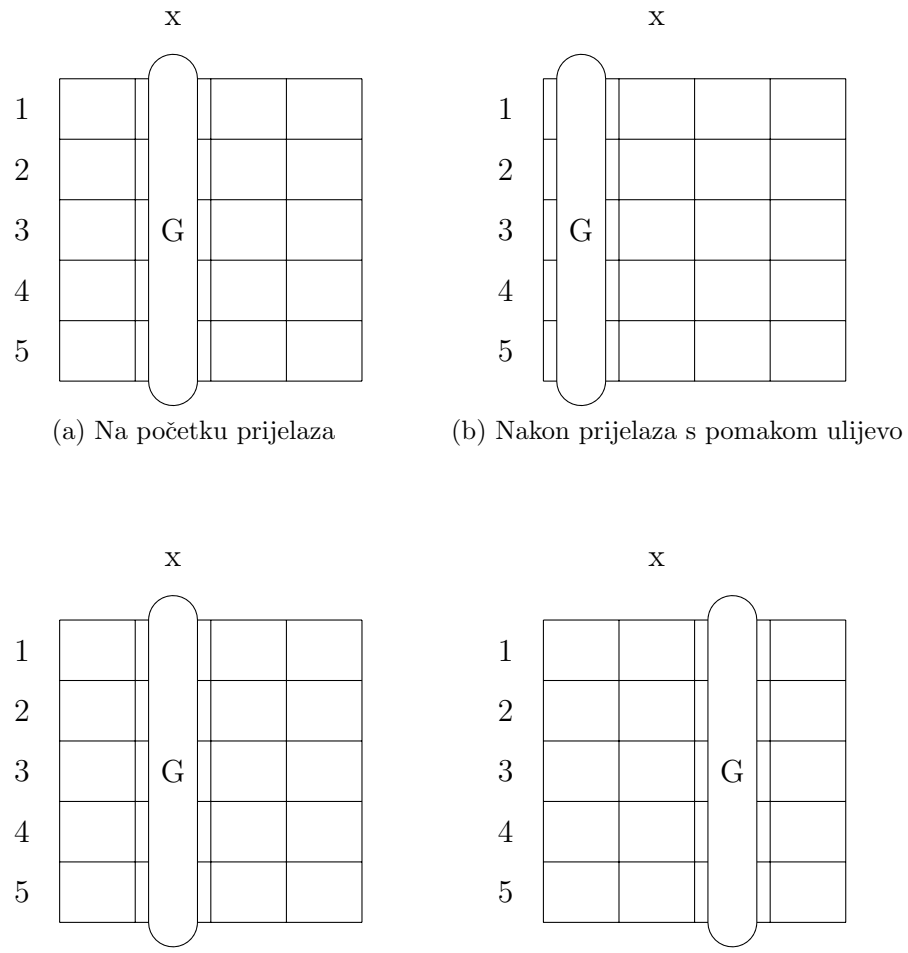
- q_0 *je element iz* Q *i naziva se* ***početno stanje***,

- q_{DA} *je element iz* Q *i naziva se* ***stanje prihvaćanja***,

- q_{NE} *je element iz* Q *i naziva se* ***stanje odbijanja***.

Pretpostavljamo kako $(\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$ *sadrži jedan istaknuti simbol* $\sqcup$ *koji nazivamo "prazan znak". Poistovjećujemo* $\alpha \in \Sigma$ *s* $(\alpha, \sqcup, \ldots, \sqcup)$ *i smatramo* $\Sigma \subset \Gamma$ *i pretpostavljamo kako se ulazna riječ, uvijek nalazi na 1. tragu. Na početku rada stroja, pretpostavljamo kako se glava stroja nalazi na poziciji koja omogućava pristup registru početnog znaka ulazne riječi. Na taj način se istovremeno pristupa registrima ostalih tragova koji se nalaze točno ispod početka ulazne riječi.*

TS s više tragova možemo zamišljati kao TS s jednom trakom koja se sastoji $k > 1$ redova (tragova). Svaki od redova ima svoju abecedu, Γ_i. U ovom slučaju, $\Gamma_i = \{0, 1, \sqcup\}$. Svaka ćelija zapravo stupac. TS može čitati i pisati po svakom redu posebno, ali još uvijek postoji samo jedna glava za čitanje i pisanje, jedna pozicija (Slika 1) [RS16, Kal16].



(a) Na početku prijelaza (b) Nakon prijelaza s pomakom ulijevo

(c) Nakon prijelaza bez pomaka glave (d) Nakon prijelaza s pomakom udesno

Slika 1: Pomak glave Turingovog stroja s jednom trakom i pet tragova u jednom prijelazu. Vertikalna linija označena s G predstavlja glavu stroja.

Vašno svojstvo višetragovnog TS-a je da redukcija višetragovnog TS-a na jednotragovni povećava vremensku složenost samo za konstantni faktor.

Teorem 1.2 (Redukcija višetragovnog Turingovog stroja na jednotragovni [Sip06b, J.68])**.** *Neka je* $f : \mathbb{N} \to \mathbb{N}$, $k \in \mathbb{N}$.
Za svaki k-tragovni TS K vremenske složenosti $f(n)$ *postoji jednotragovni TS* K' *vremenske složenosti* $d \cdot f(n)$, $d \in \mathbb{R}^+$, *koji mu je ekvivalentan.*

Kažemo da je K' nastao redukcijom višetragovnog TS-a K na jednotragovni. Glavna ideja dokaza upravo spomenutog teorema leži u definiciji abecede stroja K'. Abeceda stroja K' definira se kao Kartezijev produkt svih Γ_i.

Primjetimo da ukoliko je abeceda svakoga traga stroja K jednaka $A := \{0, 1, \sqcup\}$, tada K' ne će imati abecedu jednaku A. Ipak, K' je moguće reducirati u ekvivalentni K'' sa željenom abecedom (A). Redukcija se izvodi kodiranjem znakove abecede K' znakovima željene abecede. Moguće je da će nekoliko znakova željene abecede predstavljati jedan znak stare abecede. Prilikom čitanja jednog znaka stare abecede u novoj abecedi, novi stroj troši za konstantni faktor više vremena. Konstantni faktor ovisi o duljini najduljeg koda znaka stare abecede u novoj abecedi što je spomenuto i u službenom pojašnjenju dokaza teorema o vremenskoj složenosti iz literature [Sip06a], [Gha12]. Dakle, klase vremenske složenosti stroja K i K'' se podudaraju.

Napomena 1.2. *Od sada pa nadalje, ukoliko postoji dokaz za višetragovni TS K smatra se da isti dokaz vrijedi i za jednotragovni TS K'' iste abecede i klase vremenske složenosti. Nakon dokazane tvrdnje za K, korsitimo stroj K misleći pritom na K''.*

Prije dokaza teorema 1.1 o vremenskoj hijerarhiji dokazuju se 3 leme. One za cilj imaju pojednostaviti dokaz teorema čineći ga što čitljivijim i lakšim za razumijevanje. Kroz 3 leme definiraju se 4 TS-a koja će se koristiti u dokazu.

Ideja dokaza teorema je jednostavna, definirati TS D koji je vremenske složenosti $O(f(n))$, a da se jezik koji on odlučuje ne može odlučiti ni na jednom Turingovom stroju vremenske složenosti $o(\frac{f(n)}{log_2(f(n))})$. Kako bi se pokazalo da jezik $L(D)$ nije odlučiv u $o(\frac{f(n)}{log_2(f(n))})$ koristi se postupak dijagonalizacije. Primjer korištenja postupka dijagonalizacije može se naći u literaturi [Sip06a]: dokaz neodlučivosti jezika A_{TM}.

Krećemo s jednom jednostavnom lemom.
Budući da iskaz teorema implicitno zahtjeva izračunavanje funkcije koja ovisi o duljini riječi jezika $\mathbf{L} := L(D)$, potrebno ju je moći izračunati i to u vremenu $O(f(n))$. Točnije, želimo TS T_1 koji može izračunati duljinu ulazne riječi w u vremenu koje ne prelazi $O(f(n))$. T_1 će se kasnije koristiti u definiciji stroja D omogućavajući mu isto.
T_1 se definira sljedećom lemom.

Lema 1.3. *Postoji Turingov stroj T_1 linearne vremenske složenosti koji za ulaz w izračunava duljinu ulaza u unarnom zapisu.*

Dokaz. Definiramo Tringov stroj T_1 opisom rada stroja:
T_1 = *"na ulazu se nalazi w"*
Dok ne dođe do kraja ulazne riječi radi sljedeće:

1. zamijeni trenutni znak s 1,

2. napravi pomak udesno.

Primjetimo kako svako izvođenje 1. i 2. odgovara jednom prijelazu stroja T_1 i da se 1. i 2. izvršava točno onoliko puta kolika je duljina riječi na ulazu. Zaključujemo, $time_{T_1}(n) = O(n)$. □

Nadalje, potreban je stroj koji izračunava vremenski konstruktibilnu funkciju $f(n)$, a da ne prelazi vremensku složenost $O(f(n))$. Postojanje traženog stroja slijedi iz definicije vremenski konstruktibilne funkcije. Označimo s T_2 jedan takav stroj.

Ukoliko stroj D definiramo na način da prvo pokrenemo T_1 pa T_2, dobivamo stroj vremenske složenosti $O(f(n))$. Na taj način, jezik koji odlučuje D je sigurno iz $DTIME(f(n))$, ali potrebno je još i više. Potrebno je osigurati da $L(D)$ nije moguće odlučiti u vremenu $o(\frac{f(n)}{log_2(f(n))})$. Dakle, stroj D treba raditi dovoljno dugo, ali ne i predugo kako ne bi prešao $O(f(n))$. U tu svrhu definira se brojač koji će brojati ponavljanja određenog skupa slijednih

koraka TS-a D i paziti da mu složenost bude veća od $o(\frac{f(n)}{log_2(f(n))})$, a manja od $O(f(n))$.

Budući da se želi koristiti brojač koraka, potrebno je u sveukupno vrijeme rada TS-a uračunati i mijenjanje vrijednosti brojača. Jedno mijenjanje vrijednosti brojača sastoji se od jednog oduzimanja jedinice od broja u brojaču vremenske složenosti $O(log_2(t))$ gdje je t trenutna vrijednost brojača u binarnom zapisu [Bea04, Tre04].

Gornja granica za složenost stroja je $O(f(n))$ što daje gornju ogradu na broj mijenjanja brojača stroja, $O(\frac{f(n)}{log_2(f(n))})$. Neka je broj mijenjanja brojača točno $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$. Dakle, prilikom konstrukcije stroja D potrebno je obratiti pažnju da vremenska složenost skupa slijednih koraka čije ponavljanje brojač prebrojava ne prelazi $O(log_2(f(n)))$.

Budući da je početna vrijednost brojača različita za svaki ulaz, stroj D ju mora biti u mogućnosti izračunati. Također, potrebno je zahtjevati da vremenska složenost njezinog izračunavanja ne prelazi $O(f(n))$. Ukoliko izračun početne vrijednosti brojača prelazi $O(f(n))$, sveukupna vremenska složenosti stroja D prelazi dopuštenu granicu.

Da je gornji zahtjev moguće ostvriti dokazuje sljedeća Lema 1.4 za $n = f(n)$.

Lema 1.4. *Postoji Turingov stroj T_3 koji za ulaz $\mathbf{n}$ izračunava binarni zapis od $\lceil n/log_2(n) \rceil$ u vremenu $O(n)$.*

Dokaz. Ulazna abeceda stroja je jednaka $\{0, 1\}$ pa koristeći $\mathbf{n}$ za argument matematičkih funkcija, interpretiramo ga kao broj u binarnom zapisu.

Definicija željenog stroja je sljedeća. Prvo se definira stroj T s 4 traga. Prva 3 traga koriste za dijeljenje, dok se 4. koristi za čuvanje rezultata dijeljenja. 4. trag je izlazni trag stroja. Na početku rada stroja, na prvom tragu se nalazi ulazna riječ n.
Za vrijeme rada, stroj T radi sljedeće:

1. Prepiše ulaz na 2. trag što je složenosti $O(log_2(n))$ i ostavi glavu za

čitanje i pisanje na kraju ulazne riječi,

2. Na drugom tragu zapiše duljinu ulaza u binarnom obliku tako da kraj riječi na 2. tragu odgovara kraju riječi na 1. tragu što je složenosti $O(log_2(n))$. Na drugom tragu se sada nalazi najviše $\lceil log_2(log_2(n))\rceil$ znakova.,

3. U ravnini zadnjeg znaka riječi na 1. tragu na 4. trag zapiše 0. To je najviše $O(log_2(n))$ koraka.

4. Dok god je zadnjih $\lceil log_2(log_2(n))\rceil$ znakova na 1. tragu manje od zapisa na 2. tragu:

   (a) pozicionira se na kraj riječi na 1. tragu (indirektno i na 2.) što je složenosti $O(log_2(n))$,

   (b) oduzme 2. trag od 1. zapisujući rezultat na 3. trag. Složenost je jedan prolaz po riječi na 1. tragu, tj. $O(log_2(n))$,

   (c) obriše riječ na 1. tragu,

   (d) prepiše riječ s 3. traga na 1. trag,

   (e) broj zapisan na 4. tragu poveća za 1,

5. U ovom trenutku, na 4. tragu se nalazi rezultat željenog dijeljenja. Ukoliko se na 1. tragu nalazi broj različit od 0, povećaj broj na 4. tragu za 1. Gornja ograda složenosti ovoga koraka je $O(log_2(n))$.

Svi koraci, osim koraka 4 opisa rada stroja su složenosti najviše $O(log_2(n))$. Provjera uvjeta prilikom ponavljanja koraka 4 je složenosti $O(log_2(n))$, dok u svakom ponavljanju koraka 4, svaki podkorak ima složenost $O(log_2(n))$. Korak 4 se ponavlja najviše $\frac{n}{log_2(n)}$ puta.

Dakle, složenost stroja T je $O(log_2(n)+log_2(n)+log_2(n)+(n/log_2(n))(log_2(n)+log_2(n)+log_2(n)+log_2(n))+log_2(n)) = O(n)$. Definiramo $T_3 := T''$.

□

Kako u dokazu *Teorema o vremenskoj hijerarhiji* želimo koristiti postupak dijagonalizacije, želimo da stroj D prihvaća ulaze koji sadrže kodove Turingovih strojeva koje će kasnije simulirati.

Prilikom dokazivanja neodlučivosti A_{TM}-a [Sip06a], konstruira se stroj $D_{A_{TM}}$ koji prihvaća jezik $L_{A_{TM}}$ koji se sastoji od kodova određenih Turingovih strojeva. Budući da složenost stroja $D_{A_{TM}}$ nije bitna, $D_{A_{TM}}$ za ulaz $\langle M \rangle$ simulira cjelokupni rad stroja M. Ipak, za stroj D, vremenska složenost je bitna. Ograničavajući vrijeme (broj koraka) simulacije, omugućava se držanje vremenske složenosti stroja D ispod željene granice. U tu svrhu koristimo brojač. Brojač broji koliko se koraka TS-a definiranog ulazom simuliralo zaustavljajući simulaciju kada dosegne 0.

U dokazu *Teorema o vremenskoj hijerarhiji* bitno je moći konstruirati stroj D na način da je za $L(D)$ lako pronaći riječ duljine veće od neke donje granice. Budući da za $L_{A_{TM}}$ nije jednostavno pronaći takvu riječ, oblik riječi koje će stroj D prihvaćati definiramo malo drugačije, s $\langle M \rangle 01^*$. Za riječi oblika $\langle M \rangle 01^*$ lako je pronaći riječ koja je dulja od neke donje granice $n_d \in \mathbb{N}$, npr. $\langle M \rangle 01^{n_d}$.

Promotrimo sada kolika je vremenska složenost simulacije konačnog broja koraka Turingovog stroja definiranog na stranici 6 s $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$.

Lema 1.5. [2] *Neka je M Turingov stroj nad abecedom $\Sigma = \{0,1\}$, $\langle M \rangle$ njegov kod u abecedi Σ i $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija. Postoji 3-tragovni Turingov stroj T_4 koji za ulaz $w \in \Sigma^*$, $n := |w|$, $w = \langle M \rangle 01^*$ na prvom tragu i $k := \left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ na 2. tragu, simulaira k koraka rada stroja M s ulazom $\langle M \rangle 01^*$ u vremenu $O(k \cdot log_2(f(n)))$.*
Broj k je dan u binarnom zapisu.

Dokaz. Definiramo T_4 opisom rada stroja. Na početku rada stroja T_4, 3. trag je prazan. 3. trag se koristi za pamćenje informacije o funkciji prijelaza stroja M koja je zapisana na odgovarajućem mjestu u $\langle M \rangle$.

[2] Za ulaz oblika $\langle M \rangle 10^*$, D simulira M u vremenu $d \cdot g(n)$ gdje je d neka konstanta, $g(n)$ vremenska složenost stroja N.

T_4 radi sljedeće:

1. Prije početka simulacije stroja M, jednim prolazom po w (ulaznoj riječi), pronalazi kod funkcije prijelaza stroja M koji zapisuje na 3. trag. Prepisivanje koda funkcije prijelaza s 1. na 3. trag je složenosti $O(n)$. Ispred koda funkcije prijelaza za M zapisuje početno stanje stroja M.

2. Pomiče sadržaje 2. i 3. traga tako da im je početak poravnat s početkom riječi na 1. tragu. Pomicanje sadržaja 2. i 3. traga omogućava lakše praćenje složenosti stroja T_4. Pomicanje sadržaja 2. traga je složenosti
$log_2\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil\right) \cdot \left(n - log_2\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil\right)\right) \leq log_2(n) \cdot (n - log_2(n)) = O(log_2(n) \cdot n)$. Kako vrijedi $O(log_2(n)) \leq f(n)$ po vremenskoj konstruktibilnosti funkcije f, prepisivanje 2. traga je složenosti $O(f(n))$.

   Budući da prepisivanje sadržaja 3. traga ne ovisi o ulazu za M, već samo o stroju M i veličini koda za M, $d = |\langle M \rangle|$, pomicanje sadržaja 3. traga je vremenske složenosti $O(d \cdot d) = O(d^2)$. Naime, kraj od $\langle M \rangle$ nalazi se najviše d mjesta od početka ulazne riječi pa se početak koda funkcije prijelaza nalazi najviše d mjesta od početka ulazne riječi. Dakle, sveukupna duljina riječi na 3. tragu ne prelazi d.

3. Pomiče glavu na početak riječi na 1. tragu i započinje simulaciju. T_4 prilikom simulacije jednog koraka stroja M radi sljedeće:

   - Temeljem znak $\alpha \in \{0, 1, \sqcup\}$ koji glava čita na prvom tragu i stanja q_l koje se nalazi na početku riječi 3. traga, stroj prolazi po 3. tragu tražeći odgovarajući prijelaz. Potrebno je pronaći prijelaz koji odgovara prijelazu stroja M iz stanja q_l čitajuću α. U skladu s pronađenim prijelazom mijenja sadržaje 1. i 3. traga. Složenost ovog koraka ovisi samo o d, tj. konstantna je.

Napomena 1.3. *Budući da abeceda tragova stroja M ima konačno mnogo znakova, T_4 je u mogućnosti čuvati informaciju o upravo pročitanom znaku na 1. tragu (znaku koji predstavlja ulaz u funkciju prijelaza stroja M) u svojoj funkciji prijelaza. Temeljem tog stanja i početnog stanja koraka simulacije zapisanog na početku riječi 3. traga, stroj T_4 prolazi 3. tragom tražeći odgovarajući prijelaz. Pronađeni prijelaz definira novo stanje koje se zapisuje na početak 3. traga, brišući staro stanje. Prilikom pronalaska odgovarajućeg prijelaza, početno stanje koraka simulacije, T_4 ne može pamtiti u svojem stanju stroja. Naime, broj stanja stroja M nije unaprijed definiran, već ga definira ulazna riječ. Dakle, prolažeći 3. tragon, za svaki pronađeni prijelaz funkcije prijelaza potrebno je usporediti jednakost početnog stanja prijelaza i početno stanje koraka simulacije. Najveća udaljenost 2 spomenuta stanja na 3. tragu je d. Za svaki pronađeni prijelaz, radi se usporedba najviše d simbola udaljenih najviše d koraka. Budući da i prijelaza ima najviše d (strogo manje), složenost potrage odgovarajućeg prijelaza u svakom koraku simulacije je $O(d^3)$.*

Primijetimo kako različitih pomaka glave ima konačno mnogo pa ih T_4 pamti u svojoj funkciji prijelaza, kao i informaciju je li u stanju: a) "tražim odgovarajući prijelaz", b) "pronašao sam prijelaz, idem promijeniti trenutno čitani znak na 1. traci". c) "promijenio sam znak, idem pomaknuti glavu simulacije. Nadalje, znak riječi 1. traga nad kojem se glava stroja M nalazi prilikom koraka koji se simulira, T_4 označava s npr. točkom, tj. proširivanjem abecede na $\Sigma U \Sigma', \Sigma' = \{a^{\cdot} | a \in \Sigma\}$. Pomicanje glave stroja M simulira se pomicanjem točke u susjedni registar.

,

- Oduzme 1 broju zapisanom na 2. tragu. Oduzimanje je složenosti $O(log_2(n))$ što u ovom slučaju znači $O\left(log_2\left(\frac{f(n)}{log_2(f(n))}\right)\right) = O(log_2(f(n)))$

,

- Pomakne sadržaje 2. i 3. traga za jedno mjesto definirano upravo simuliranim korakom (prijelazom) u stranu u koju bi se stroj M pomaknuo izvršavajući upravo simulirani prijelaz. Složenost pomaka je $O(log_2(f(n)))$ za 2. trag i $O(d)$, $d = |\langle M \rangle|$ za 3. Na ovaj način se početci riječi na svim tragovima uvijek otprilike poklapaju. Pomicanje sadržaja 2. i 3. traga je bitna kako proces traženja budućih prijelaza ne bi trajao predugo.,
- Sada se T_4 nalazi u stanju koje odgovara znaku koji M čita nakon upravo simuliranog koraka. Stanje u kojem se M nalazi nakon upravo simuliranog koraka nalazi se na početku riječi 3. trake,
- Ako na 2. tragu nije zapisana 0, idi na 1. U suprotnome stani.

Simulacija jednog koraka stroja M je složenosti
$O(d^3) + O(log_2(f(n))) + O\left(log_2(f(n)) + d\right) + O(1) = O(log_2(f(n)))$.

4. Ukoliko je M prihvatio ulaz, definira se da T_4 prihvaća ulaz, ukoliko M ne prihvaća ulaz ili u $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka još nije stao, definira se da T_4 ne prihvaća ulaz.

Budući da se izvršava simulacija samo prvih $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka stroja M, stroj T_4 je složenosti $O\left(n + f(n) + \left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil \cdot log_2(f(n))\right) = O(f(n))$.

□

Gornjom lema pokazuje kako simulacija samo $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka proizvoljnog stroja M s ulazom $\langle M \rangle\, 01^*$ ne traje predugo, tj. ne više od $O(f(n))$.

Sada je sve spremno za dokaz teorema 1.1 o vremenskoj hijerarhiji koji slijedi.

Dokaz. Dokaz teorema provodi se u 2 faze:

1. Konstrukcija Turingovog stroj D koji radi u vremenu $\mathbf{L} \in DTIME(f(n))$, određivanje ulaza za D i definiranje $\mathbf{L} := L(D)$.

2. Dokaz kako $L(D)$ nije moguće odlučiti u vremenu $o(\frac{f(n)}{log_2(f(n))})$. Koristi se postupak dijagonalizacije.

Faza 1

Turingov stroj D definira se koristeći prije definirane strojeve T_1, T_2, T_3 i T_4. D ima jednu traku, a ona ima 3 traga. Pretpostavlja sa kako su na početku rada stroja svi tragovi osim prvoga prazni. Prvi trag sadrži ulaznu riječ, a glava stroja se nalazi na poziciji koja omogućava čitanje početnog simbola te riječi.

Stroj D prihvaća ulaze oblika $w = \langle M \rangle 01^*$, ali ne sve. Kako bi odredio koje riječi zadanog oblika će odbiti, a koje prihvatiti, D simulira $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka stroja M s ulazom w. Stroj D odbija ulaz w ako i samo ako M prihvaća ulaz w u manje od $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka.

Upravo ovakva definicija prihvaćanja stroja D je ključna za *Fazu 2* jer dovodi do odbacivanja mogućnosti odlučivanja $L(D)$ u vremenu $o\left(\frac{f(n)}{log_2(f(n))}\right)$.

Slijedi formalniji opis rada stroja D:
$D :=$ *"na ulazu se nalazi riječ* w, $n := |w|$*"*

1. Provjeri je li w oblika $\langle M \rangle 10^*$ gdje je M neki Turingov stroj. Ako w nije spomenutog oblika, definira se da D odbija ulaz. Provjera valjanosti ulaza je složenosti $O(n)$.

2. Prepiše ulaz na 2. trag što je složenosti $O(n)$.

3. Izvede T_1 na 2. tragu.
   Završetkom rada stroja T_1, jedino je sadržaj 2. traga promjenjen. Složenost izvedbe stroja T_1 je $O(n)$ prema lemi 1.3. Na 2. tragu se sada nalazi unarni zapis od n, BSOMP 1^n.

4. Izvede T_2 pa T_3 na 2. tragu.
   Izvedba T_2 je vremenske složenosti $O(f(n))$ jer je T_2 vremenske složenosti $O(f(n))$ za ulaz 1^n. Prije pokretanja stroja T_3, na drugom tragu se nalazi binarni zapis riječi $f(n)$. Prema lemi 1.4, T_3 za ulaz $f(n)$ u binarnom obliku, radi najviše $O(f(n))$ koraka. Zadaća stroja T_3 je zapisati početnu vrijednost brojača stroja D na 2.trag.

5. Postavi glavu stroja na poziciju koja odgovara početku zapisane riječi na 2. tragu što je složenosti $O(n)$.

6. Izvede T_4.
   Izvedba T_4 je vremenske složenosti $O\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil log_2(f(n))\right) = O\left(f(n)\right)$ prema lemi 1.5.

7. Ako T_4 (M u $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka) prihvati ulaz w, definira se da D ne prihvaća ulaz. Ukoliko T_4 ne prihvati ulaz w (M u $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka ne prihvati ulaz w ili uopće ne stane), definira se da D prihvaća ulaz w.

Primjetimo kako se u koraku 6 opisa rada stroja, simulira samo prvih $\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil\right)$ koraka rada stroja M s ulazom w.

Zaključno, D je vremenske složenosti $O\left(n + n + f(n) + n + f(n) + 1\right) = O\left(f(n)\right)$. Naime, po definiciji vremenski konstruktibilne funkcije vrijedi $f(n) \geq O\left(n \cdot log_2\left(n\right)\right)$. Definiramo $\mathbf{L}$:= $L(D)$.

Faza 2

Cilje ove faze je dokazati kako $\mathbf{L}$ nije moguće odlučiti u vremenu $o\left(\frac{f(n)}{log_2(f(n))}\right)$. Dokaz se provodi svođenjem na kontradikciju. Pretpostavimo suprotno, tj. $\mathbf{L}$ je moguće odlučiti u vremenu $o\left(\frac{f(n)}{log_2(f(n))}\right)$.
Tada postoji Turingov stroj N koji odlučuje $L(D)$ vremenske složenosti $g(n) = o\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil\right)$. Budući da je

$g(n) = o\left(\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil\right)$, tada $\lim_{n\to\infty} \frac{\frac{g(n)}{f(n)}}{log_2(f(n))} = 0$.
Po definiciji limesa funkcije, za svaki $\epsilon > 0, \exists n_0 \in \mathbb{N}$
t.d. $\forall n \in \mathbb{N}, n > n_0 \left| \frac{\frac{g(n)}{f(n)}}{log_2(f(n))} \right| < \epsilon$.
Neka je sada $\epsilon = 1$ i $n_d \in \mathbb{N}$ takav da $\forall n \in \mathbb{N}, n > n_d \left| \frac{\frac{g(n)}{f(n)}}{log_2(f(n))} \right| < 1$, tj. $g(n) < \frac{f(n)}{log_2(f(n))}$.

Stroj N s ulazom $w = \langle N \rangle 10^{n_N}$, $n_N > n_d$, završava u najviše $g(n), n = |w|$, koraka. Stroj D za ulaz w simulira prvih $\left\lceil \frac{f(n)}{log_2(f(n))} \right\rceil$ koraka stroja N. Kako je $n > n_d$ vrijedi $g(n) < \frac{f(n)}{log_2(f(n))}$. Dakle, D za ulaz oblika w izvršava simulaciju stroja N s ulazom w do kraja.

Neka je sada $w = \langle N \rangle 10^{n_N}$ ulaz za D.
Ukoliko D prihvati ulaz tada $w \in L(D)$. Ali, prema definiciji rada stroja D i jer se simulacija od N izvrši do kraja, stroj N odbija ulaz w. Kako N odlučuje $L(D)$, vrijedi $w \notin L(D)$ čime dobivamo kontradikciju.
Ukoliko D odbije ulaz tada $w \notin L(D)$. Prema definiciji rada stroja D i jer se simulacija od N izvrši do kraja, stroj N prihvaća ulaz w pa $w \in L(D)$ što je u kontradikciji s $w \notin L(D)$.
Zaključno, N ne odlučuje $L(D)$ što je u kontradikciji s polaznom pretpostavkom pa $L(D)$ nije odlučiv u $o\left(\frac{f(n)}{log_2(f(n))}\right)$. □

Za kraj, promotrimo detalj leme 1.5. Prilikom definiranja složenosti stroja T_4, pretpostavka o vremenskoj složenosti koraka simulacije nije u potpunosti razjašnjena. Za ulaz $w = \langle M \rangle 01^*$, pretpostavlja se da je duljina koda stroja M, u oznaci d, konstantna. Ipak, kôd stroja M je dio ulaza za T_4 i mijenja se u odnosu na ulaznu riječ, a time i njegova duljina. Budući da su jedini mogući ulazi koje D prihvaća oblika $\langle M \rangle 01^*$ pa $d < n$, može se zahtjevati da d bude dovoljno mali u odnosu na n kako bi se ostvarilo $d^3 \leq log_2(f(n))$. Za takve ulaze, složenost stroja T_4 ostaje dovoljno mala pa se ne narušava složenost

stroja D ni njegova primjena u *Fazi 2*. U tu svrhu uvodimo proširenje provjere ispravnosti ulaza stroja D. Nadopuna provjere ispravnosti ulaza stroja D izvršava se nakon izvedbe stroja T_2, u sklopu koraka 4 opisa rada stroja. U tom trenutku, na 2. tragu stroja D sadrži binarni zapis broja $f(n)$ i lako ga je usporediti s d^3. Ukoliko $d^3 > log_2(f(n))$, definira se da D odbija ulaz. Kako je d^3 moguće izračunati u $O(f(n))$ prema lemi u nastavku, nadopuna provjere ispravnosti ulaza ne povećava vremensku složenost stroja D iznad $O(f(n))$.

Lema 1.6. *Postoji 3-tragovni Turingov stroj T koji za ulaz d u binarnom zapisu na 1. tragu (dok su ostali prazni), izračunava d^3 u binarnom zapisu i njegova je vremenska složenost $O(dlog_2(d))$.*

Dokaz. T definiramo da radi sljedeće:

1. Prepiše d na 2. i 3. trag
   . Izvršava se jednim prolaskom po riječi na 1. tragu. Ono što glava čita na 1. tragu prepisuje se na 2. i 3. trag. Ovaj korak je složenosti $O(log_2(d))$. U ovom trenutku krajevi riječi na svim tragovim su poravnati.

2. Pribroji broj na 3. trag broju na 1. tragu
   Budući da su krajevi riječi na 3. i 1. tragu jedan ispod drugoga, zbrajanje je moguće izvršiti u vremenu $O(log_2(d))$ jedim prolazom po riječi na ulazu. Zbrajanje je linearno u duljini riječi koje se zbrajaju.

3. Oduzme 1 broju na 2. tragu.
   Primijetimo kako se ponavljanjem koraka opisa rada stroja 2 i 3 duljina riječi na 3. tragu ne mijenja, a na 1. tragu se najviše udvostruči ($log_2(d^2) = 2log_2(d)$) što ne mijenja klasu vremenske složenosti od 2. Složenost od 3 je složenost oduzimanja 2 broja duljina $O(d)$, tj. $O(log_2(d))$.

4. Ako broj na 2. tragu nije nula, ide na 2 .
   Provjera je složenosti najviše $O(log_2(n))$. Specijalan slučaj predstavlja situacija kada se glava stroja nalazi na kraju riječi. Tada se složenost smanjuje na $O(1)$.

5. Prepiše riječ na 3. tragu na 2. trag .
   Složenost je $log_2(d) = O(log_2(d))$.

6. Prepiše riječ na 1. tragu na 3. trag
   što je složenosti je $log_2(d^2) = 2log_2(d) = O(log_2(d))$.

7. Pribroji broj na 3. trag broju na 1. tragu ,

8. Oduzme 1 broju na 2. tragu što je složenoati $O(log_2(d))$.,

9. Ako je broj na 2. tragu nije nula, ide na 7. U suprotnome staje.

Složenosti od 7, 8 i 9 su analogne složenostima od 2, 3 i 4 jer veličina zapisa brojeva na svim tragovima nikada ne prelazi $O(log_2(d))$. Maksimum postiže na 1. tragu kada je $log_2(d^3) = 3log_2(d)$.

Budući da se prije prve izvedbe koraka 2 ili 7, na 2. tragu nalazi binarni zapis od d, 2, 3 i 4 i 7, 8 i 9 se ponavljaju točno d puta. Dakle, složenost upravo opisanog stroja je $O(log_2(d)) + d \cdot O(log_2(d) + log_2(d) + log_2(d)) + O(log_2(d)) + O(log_2(d)) + d \cdot O(log_2(d) + log_2(d) + log_2(d)) = O(d \cdot log_2(d))$. □

S T'' definiramo jednotragovni Turingov stroj ekvivalentan stroju T nastao redukcijom višetragovnog Turingovog stroja na jednotragovni abecede $\{0, 1, \sqcup\}$. Dakle, složenost od T'' je također $O(d \cdot log_2(d))$ i za ulaz d stroj izračunava d^3. Dakle, T'' možemo koristiti u nadopuni ispravnosti ulaza stroja D za izračun d^3. Korištenje T'' ne povećava vremansku složenost stroja D.

Kako bi dokaz teorema 1.1 o vremenskoj hijerarhiji bio u potpunosti u skladu s uvedenom modifikacijom, u *Fazi 2* zahtjevamo da n_N mora biti veći i od $|N|^3$.

2 Popis literature

[Bea04] Paul Beame. *Strange Turing Machines.* https://courses.cs.washington.edu/courses/cse532/04sp/root.pdf, 2004. stranica 5, zadnji paragraf, [Online; posjećeno 23. lipnja 2017.].

[FS66] F.C.Hennie and R. E. Stearns. *Two-Tape Simulation of Multitape Turing Machines* . http://www.dcs.fmph.uniba.sk/~fduris/VKTI/2tape.pdf, 1966. [Online; posjećeno 23. lipnja 2017.].

[FS16] Lance Fortnow and Rahul Santhanam. *New Non-Uniform Lower Bounds for Uniform Classes.* http://drops.dagstuhl.de/opus/volltexte/2016/5850/pdf/29.pdf, 2016. [Online; posjećeno 23. lipnja 2017.].

[Fü82] Martin Fürer. *The Tight Deterministic Time Hierarchy.* http://www.cse.psu.edu/~fhs/Significant%20research%20results.html, 1982. [Online; posjećeno 23. lipnja 2017.].

[Gha12] Kaveh Ghasemloo. *Errata for Introduction to the Theory of Computation, 3rd Edition.* http://math.mit.edu/~sipser/itoc-derrs3.1.html, 2012. [Online; posjećeno 23. lipnja 2017.].

[J.68] Hartmanis J. *Computational Complexity of One-Tape Turing Machine Computations.* http://cse.iitkgp.ac.in/~goutam/toc/readingMaterial/computational-complexity-one-tape-Hart.pdf, 1968. [Online; posjećeno 23. lipnja 2017.].

[Kal16] Ananth Kalyanaraman. *Turing Machines*. https://people.eecs.berkeley.edu/~luca/cs172/noteh.pdf, 2016. pages 18-21, [Online; posjećeno 23. lipnja 2017.].

[RS16] Eric Robles and Luke Skinker. *Strange Turing Machines*. http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/TuringMachines.pdf, 2016. [Online; posjećeno 23. lipnja 2017.].

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. *Separating Nondeterministic Time Complexity Classes* . http://pages.cs.wisc.edu/~jyc/710/seiferas.pdf, 1978. [Online; posjećeno 23. lipnja 2017.].

[Sip06a] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, second edition, 2006.

[Sip06b] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, second edition, 2006. Page: 370, second paragraph.

[Tre04] Luca Trevisan. *Notes on Hierarchy Theorems*. https://people.eecs.berkeley.edu/~luca/cs172/noteh.pdf, 2004. U Lemi 10, [Online; posjećeno 23. lipnja 2017.].

[Vuk16] Mladen Vuković. Složenost algoritama. https://www.math.pmf.unizg.hr/sites/default/files/pictures/sa-predavanja-2015-11.pdf, 2016. [Online; posjećeno 23. lipnja 2017.].

[wJP78] w. J. Paul. *On Time Hierarchies*. http://ac.els-cdn.com/002200007990028X/1-s2.0-002200007990028X-main.pdf?_tid=e876be7c-56c6-11e7-b6bb-00000aacb35e&acdnat=1498079976_17d3c54c3afd5c6b8f01e2fa938fe947, 1978. [Online; posjećeno 23. lipnja 2017.].

A Prilog: Neke druge inačice *Teorema o vremenskoj hijerarhiji*

U prilogu se iskazuju 4 drugačije inačice *Teorema o vremenskoj hijerarhiji.* Prvo donosimo iskaz $DTIME$ inačice *Teorema o vremenskoj hijerarhiji* o Turingovom stroj s dvije trake[Fü82, FS66] pa s k traka [wJP78]. Na kraju slijede 2 $NTIME$ inačice koje je moguće naći u [FS16], odnosno [SFM78].

Teorem A.1 (Teorem o vremenskoj hijerarhiji (2 trake))**.** *Neka je $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija. Tada postoji jezik $\boldsymbol{L}$ koji je odlučiv na dvotračnom deterministočkom Turingovom stroju vremenske složenosti $O(f(n))$, ali nije odlučiv ni na jednom dvotračnom deterministočkom Turingovom stroju vremenske složenosti $o(f(n))$.*

Teorem A.2 (Teorem o vremenskoj hijerarhiji (k traka))**.** *Neka je $k \in \mathbb{N}$, $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ t.d. $lim(inf)_{n\to\infty} \frac{f_1(n)\cdot log^*(f_1(n))}{f_2(n)} = 0$ i f_2 je vremenski konstruktibilna funkcija na nekom k-tračnom deterministočkom Turingovom stroju. Tada postoji jezik $\boldsymbol{L} \in DTIME_k(f_2(n))$, ali $\boldsymbol{L} \notin DTIME_k(f_1(n))$.*

S $DTIME_k(f(n))$ označavamo skup svih jezika koji su odlučivi na nekom k-tračnom Turingovom stroju vremenske složenosti $O(f(n))$. Također, $log^*(n) := min\left\{k | 2^{2^{2^{\cdots k_{times}}}} \geq n\right\}$, tj. najmanji k takav da vijedi $log_2^k(n) \leq 1$.

Teorem A.3 (Teorem o vremenskoj hijerarhiji (NTIME [FS16]))**.** *Neka su $d \geq 1$ i $d' > d$ neke konstante i $f : \mathbb{N} \to \mathbb{N}$, $f \in o(n^d)$, vremenski konstruktibilna funkcija. Tada vrijedi $NTIME(n^d) \not\subseteq \frac{NTIME(t)}{n^{\frac{1}{d'}}}$.*

Teorem A.4 (Teorem o vremenskoj hijerarhiji (NTIME 2 [SFM78]))**.** *Neka je $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija i $f_2 : \mathbb{N} \to \mathbb{N}$, $f_2(n+1) = o(f(n))$. Tada postoji jezik $\boldsymbol{L} \in NTIME(f(n))$, ali $\boldsymbol{L}$ nije iz $NTIME(f_2(n))$.*