Sveučilište u Zagrebu
PMF–Matematički odsjek

Mladen Vuković

Složenost algoritama

predavanja i vježbe

21. ožujka 2019.

Predgovor

Ovaj nastavni materijal namijenjen je prije svega studentima diplomskog studija Računarstvo i matematika Prirodoslovno–matematičkog fakulteta u Zagrebu. Kolegij *Složenost algoritama* trenutno je obavezni kolegij na drugoj godini diplomskog studija Računarstvo i matematika s tjednom satnicom 3+0. Navedeni kolegij predajem od akademske godine 2009./10. Sadržaj kolegija većinom je pokriven izvrsnim Sipserovim udžbenikom [23]. Jedan od ciljeva ovog materijala je naglasiti veze sa sadržajima koji se predaju u kolegijima *Matematička logika*, *Oblikovanje i analiza algoritama*, *Interpretacija programa* i *Izračunljivost*.

Algoritam je jedan od osnovnih pojmova u računarstvu i matematici. Istaknimo ovdje neke algoritme koje su studenti trebali upoznali tijekom studija: Euklidov i Hornerov algoritam (kolegij *Elementarna matematika 1*); algoritmi za pretraživanje (kolegiji *Programiranje 1 i 2*); algoritmi za sortiranje (kolegiji: *Programiranje 1, Diskretna matematika, Kombinatorika, Strukture podataka i algoritmi*); algoritmi na grafovima (kolegiji: *Diskretna matematika, Oblikovanje i analiza algoritama*); osnovni Gaussov algoritam (kolegij *Numerička matematika*); QR algoritam (kolegij *Iterativne metode*); algoritmi normaliziranih rezova – obrada slike (kolegij *Matematičko modeliranje pretraživača*); Dijkstrin algoritam, algoritmi za poravnanje parova nizova i Viterbijev algoritam (kolegij *Bioinformatika*); simpleks metoda (kolegij *Uvod u optimizaciju*); algoritmi za zbrajanje, oduzimanje, množenje i dijeljenje prirodnih brojeva, brza Fourierova transformacija (kolegij *Oblikovanje i analiza algoritama*); heuristički algoritmi za rješavanje problema `SAT` i `TSP` (kolegij *Metaheuristike*); ...

U kolegiju *Matematička logika* posebno se naglasi važna razlika logike sudova i logike prvog reda: prva je odlučiva, a druga nije (Churchov teorem). Na taj način se želi istaknuti da je nužno formalno definirati pojam algoritma. U kolegiju *Izračunljivost* definira se pojam algoritma, tj. formalno se definira intuitivni pojam izračunljive funkcije. U kolegiju *Interpretacija programa* među ostalim razmatraju se jezici i gramatike, te konačni automati. U kolegiju *Oblikovanje i analiza algoritama* razmatraju se neki konkretni algoritmi, te se određuje njihova vremenska složenost.

Poglavlje pod naslovom Dodatak sadrži dokaze nekih teorema i rješenja zadataka koji su usko povezani s ostalim dijelovima ovog nastavnog materija, ali nisu ispredavani, a onda nisu ni nužni za polaganje kolegija. Sadržaj Dodatka velikim dijelom čine studentski seminari.

Za greške u ovom nastavnom materijalu kriv sam samo ja. Biti ću zahvalan svakome tko me upozori na grešku bilo kakve vrste.

Zagreb, prosinac, 2018. Mladen Vuković

Sadržaj

Poglavlje 1

Uvod

Ovaj Uvod ima nekoliko ciljeva. Prvo ćemo navesti neku motivaciju za proučavanje složenosti algoritama. Zatim ćemo ponoviti neke pojmove vezane uz izračunljivost. Jedna točka ovog Uvoda biti će posvećena konačnim automatima i jezicima. Pošto je gotovo u svim udžbenicima iz teorije složenosti osnovni model izračunavanja Turingov stroj (u kolegiju Izračunljivost to je bio RAM–stroj), jednu točku ćemo posvetiti Turingovim strojevima. U nekim dijelovima skripte koristit ćemo i neke pojmove matematičke logike. Pojmove iz logike ponovit ćemo neposredno ispred dijela skripte gdje ćemo ih koristiti.

1.1 Motivacija

The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) odabrao je 2000. godine 7 važnih, a dugo vremena neriješenih problema, ponudivši nagradu od po milijun dolara za rješenje svakog od njih. Ti problemi zovu se *Milenijski problemi*. Jedan od tih problema naziva se "P = NP?" ili "P vs. NP". Riječ je o problemu iz teorije algoritama, a koji, vrlo pojednostavljeno rečeno, glasi: *Postoje li, za naizgled složene algoritme, jednostavniji algoritmi koji rješavaju iste probleme?*

Sada navodimo primjer s mrežne stranice CMI–a kojim želimo na jednostavan način ilustrirati problem "P=NP?" .

Primjer 1.1. *Pretpostavimo da želimo organizirati smještaj za grupu od 400 studenata. Prostor je ograničen, te će samo 100 studenata dobiti sobu. Da ne bi bilo prejednostavno, dekan je dao listu parova "nekompatibilnih" studenata: onih koji ne mogu zajedno biti u domu (tj. ako jedan dobije smještaj, drugi ga ne smije dobiti).*

Ako želimo eksplicitno rješenje postavljenog problema moramo razmotriti sve 100 člane podskupove skupa studenata, te provjeriti koji od njih (ako ijedan!) zadovoljava uvjet "nekompatibilnosti".

Znamo da traženih podskupova ima $\binom{400}{100}$, *što je otprilike* $3 \cdot 10^{27}$ *skupova.*

Današnjem najbržem računalu trebalo bi oko 30 000 godina da na taj način riješi problem.

Razmatrani problem je jedan od tipičnih tzv. NP problema, jer je za neko ponuđeno rješenje jednostavno provjeriti zadovoljava li dane uvjete, ali do samog rješenja općenito nije jednostavno doći.

Sljedećom tablicom želimo naglasiti koliko su veliki neki brojevi. Kasnije, kada ćemo razmatrati vremensku složenost nekih algoritama, bilo bi dobro prisjetiti se ove tablice, i to posebno, kada ćemo imati situaciju da je neki algoritam eksponencijalne vremenske složenosti.

Pretpostavimo da neko računalo izvršava milijun instrukcija u sekundi. U sljedećoj tablici su istaknuta okvirna vremena koja bi bila potrebna računalu za izvršiti $f(n)$ instrukcija za različite vrijednosti n.

$f(n)$	$n = 10^3$	$n = 10^5$	$n = 10^6$
$\log_2 n$	10^{-5} sec	$1,7 \cdot 10^{-5}$ sec	$2 \cdot 10^{-5}$ sec
n	10^{-3} sec	0,1 sec	1 sec
$n \cdot \log_2 n$	0,01 sec	1,7 sec	20 sec
n^2	1 sec	3 sata	12 dana
n^3	17 min	32 godine	317 stoljeća
2^n	10^{285} stoljeća	$10^{10\ 000}$ godina	$10^{100\ 000}$ godina

Naglasit ćemo još jednom nepraktičnost algoritama čija je vremenska složenost eksponencijalna. Ako bi imali računalo koje u jednoj sekundi izvršava 100 milijuna instrukcija, njemu bi za izvršavanje 2^{60} instrukcija trebalo oko 365 godina.

Želimo još istaknuti da ćemo osim vremenske složenosti promatrati i prostornu složenost.

1.2 Izračunljivost – kratki sažetak

U ovoj točki ćemo kratko ponoviti osnovne pojmove i teoreme vezane uz teoriju izračunljivosti. To su sljedeći pojmovi: RAM–izračunljiva funkcija, primitivno rekurzivne funkcije i relacije, (parcijalno) rekurzivne funkcije, rekurzivni skupovi, indeks funkcije, Kleenijev teorem o normalnoj formi, teorem rekurzije, Riceov teorem, Churchova teza, aritmetička hijerarhija i rekurzivno prebrojivi skupovi. Sve detalje o navedenim pojmovima možete naći u [26].

Jedno od osnovih pitanja u teoriji izračunljivosti je što znači da je neka funkcija $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ izračunljiva. Jedan način stroge definicije izračunljive funkcije dan je pomoću RAM–stroja.

RAM–stroj (eng. random access machine) je model idealiziranog računala. Sastoji se od trake s registrima, spremnika za program i brojača. Registri na traci su redom označeni sa $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ Pretpostavljamo da registara ima prebrojivo beskonačno mnogo. Na početku rada RAM–stroja, te u svakom trenutku rada, u svakom registru može biti upisan proizvoljni prirodni broj. U spremniku za program prije početka rada stroja upisuje se program. Program za RAM–stroj je konačan niz instrukcija.

Postoje četiri tipa instrukcija za RAM–stroj: INC $\mathcal{R}_k$, DEC $\mathcal{R}_k, m$, GO TO m i STOP. Opišimo kratko svaku instrukciju. Kada stroj izvodi instrukciju INC $\mathcal{R}_k$ tada jednostavno broj u registru $\mathcal{R}_k$ poveća za jedan, te broj u brojaču poveća za jedan. Kada izvodi instrukciju DEC $\mathcal{R}_k, m$ tada stroj prvo ispita da li se u registru $\mathcal{R}_k$ nalazi broj različit od nule. Ako je u $\mathcal{R}_k$ zapisan broj različit od nule, tada se taj broj smanjuje za jedan, a broj u brojaču se poveća za jedan. Ako je u registru $\mathcal{R}_k$ zapisana nula, tada se samo broj u brojaču promijeni u m. Kada stroj izvodi instrukciju GO TO m, tada se jednostavno broj u brojaču promijeni u m. Instrukcija STOP znači bezuvjetno zaustavljanje rada stroja. Rad RAM–stroja s programom P i ulaznim podacima $\vec{x} = (x_1, \ldots, x_k)$ (na početku rada stroja u registru $\mathcal{R}_i$ je zapisan prirodan broj x_i, za $i = 1, \ldots, k$) nazivamo P–izračunavanje sa $\vec{x}$. Kažemo da program P za RAM–stroj izračunava funkciju $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ ako za sve $\vec{x} \in \mathbb{N}^k$ vrijedi:

> $\vec{x} \in S$ ako i samo ako P–izračunavanje sa $\vec{x}$ stane, te je na kraju rada u registru $\mathcal{R}_0$ zapisan broj $f(\vec{x})$.

Kažemo da je neka funkcija $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ RAM–izračunljiva ako postoji program P za RAM–stroj koji je izračunava.

Kodiranje RAM–programa je svaka injekcija $P \mapsto e$ koja je izračunljiva, te postoji program Q za RAM–stroj tako da Q–izračunavanje sa x odgovara na pitanje je li x kod nekog RAM–programa, te određuje u tom slučaju o kojem programu se radi.

Drugi način stroge definicije pojma izračunljive funkcije je pomoću parcijalno rekurzivnih funkcija. Klasa svih parcijalno rekurzivnih funkcija je najmanja klasa funkcija koja sadrži sve inicijalne funkcije (nul–funkciju, funkciju sljedbenik, te sve projekcije) i zatvorena je na kompoziciju, primitivnu rekurziju i μ–operator. Za parcijalno rekurzivnu funkciju kažemo da je primitivno rekurzivna ako je totalna, te ju je moguće definirati bez korištenja μ–operatora. Za parcijalno rekurzivnu funkciju kažemo da je rekurzivna ako je totalna. Za skup (ili relaciju) $S \subseteq \mathbb{N}^k$ kažemo da je (primitivno) rekurzivan ako je karakteristična funkcija χ_S (primitivno) rekurzivna.

Za sve $e, k \in \mathbb{N}$ sa $\{e\}^k$ označavamo parcijalnu funkciju iz $\mathbb{N}^k$ koja je definirana sa:

$$\{e\}^k(\vec{x}) \simeq \begin{cases} \text{izlazni rezultat}, & \text{ako je } e \text{ kod nekog programa } P, \\ & \text{te } P\text{–izračunavanje sa } \vec{x} \text{ stane}; \\ \text{nedefinirano}, & \text{inače}. \end{cases}$$

Kažemo da za neku funkciju $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ postoji indeks e ako vrijedi $f \simeq \{e\}^k$.

Kleenijev teorem o normalnoj formi za parcijalno rekurzivne funkcije. Postoji primitivno rekurzivna funkcija U, te za svaki $k \in \mathbb{N} \setminus \{0\}$ postoji primitivno rekurzivna relacija T_k, tako da za svaku k–mjesnu parcijalno rekurzivnu funkciju φ postoji $e \in \mathbb{N}$ tako da za svaki $\vec{x} \in \mathbb{N}^k$ vrijedi $\varphi(\vec{x}) \simeq U(\mu y T_k(e, \vec{x}, y))$.

Istaknimo neke posljedice Kleenijevog teorema:

a) funkcija je RAM–izračunljiva ako i samo ako funkcija je parcijalno rekurzivna;

b) za funkciju postoji indeks ako i samo ako funkcija je parcijalno rekurzivna;

c) za svaku $(k+1)$–mjesnu parcijalno rekurzivnu funkciju G postoji $e \in \mathbb{N}$ tako da za svaki $\vec{x} \in \mathbb{N}^k$ vrijedi: $G(e, \vec{x}) \simeq \{e\}(\vec{x})$. (Teorem rekurzije)

d) ako je $\emptyset \neq S \subsetneqq \mathbb{N}$ koji ima svojstvo da $i \in S$ i $\{i\} \simeq \{j\}$ povlači $j \in S$, tada S nije rekurzivan skup. (Riceov teorem)

Churchova teza: Svaka izračunljiva funkcija je parcijalno rekurzivna.

Halting problem: Postoji li RAM–stroj koji za svaki program P i svaki $\vec{x}$ određuje hoće li P–izračunavanje sa $\vec{x}$ stati?
Halting problem je nerješiv, tj. ne postoji traženi RAM–stroj (tj. program za RAM–stroj). Istaknimo da je dokaz nerješivosti halting problema jednostavna posljedica egzistencije funkcije koja nije izračunljiva:

$$f(x) \simeq \begin{cases} \{x\}(x)+1, & \text{ako je } \{x\}(x) \text{ definirano;} \\ 0, & \text{inače.} \end{cases}$$

(Uočite da za definiciju funkcije f koristimo dijagonalni postupak, a njena neizračunljivost slijedi iz Churchove teze).

Konačan niz kvantifikatora oblika $Q_1 y_1 \ldots Q_n y_n$, gdje su $Q_i \in \{\forall, \exists\}$, nazivamo prefiks. Kažemo da je prefiks Π_n^0 ako je alternirajući, sadrži n kvantifikatora i prvi kvantifikator slijeva je $\forall$. Kažemo da je prefiks Σ_n^0 ako je alternirajući, sadrži n kvantifikatora i prvi kvantifikator slijeva je $\exists$. Kažemo da je neka relacija R jedna Π_n^0 relacija ako postoji rekurzivna relacija P i Π_n^0 prefiks $Q_1 y_1 \ldots Q_n y_n$, tako da vrijedi: $R(\vec{x})$ ako i samo ako $Q_1 y_1 \ldots Q_n y_n P(\vec{x}, y_1, \ldots, y_n)$. Ponekad ćemo pisati $R \in \Pi_n^0$, te govoriti da je Π_n^0 oznaka klase svih Π_n^0 relacija. Slično definiramo pojam Σ_n^0 relacije. Kažemo da je relacija Δ_n^0 ako je istovremeno Π_n^0 i Σ_n^0.

Teorem o aritmetičkoj hijerarhiji. Za svaki $n > 0$ postoji Π_n^0 relacija koja nije Σ_n^0, te postoji Σ_n^0 relacija koja nije Π_n^0.

Kažemo da je relacija $S \subseteq \mathbb{N}^k$ rekurzivno prebrojiva ako je S domena neke parcijalno rekurzivne funkcije. Kratko ćemo reći da je to RE–relacija (eng. *recursively enumerable*). Istaknimo neke činjenice o rekurzivno prebrojivim relacijama:
a) relacija R je RE–relacija ako i samo ako je $R \in \Sigma_1^0$;
b) postoji RE–relacija koji nije rekurzivna;
c) funkcija f je parcijalno rekurzivna ako i samo ako je graf od f jedna RE–relacija. (Teorem o grafu)
d) relacija R je rekurzivna ako i samo ako su relacije R i $\neg R$ rekurzivno prebrojive. (Postov teorem)
e) skup $A \subseteq \mathbb{N}$ je rekurzivno prebrojiv ako i samo ako skup A je slika neke parcijalno rekurzivne funkcije.

1.3 Konačni automati i jezici – kratki sažetak

Konačni automati su "konačne aproksimacije" Turingovih strojeva. Oni su najjednostavniji modeli računanja. Ovdje ćemo kratko ponoviti osnovne činjenice o konačnim automatima. Prije konačnih automata navodimo definicije osnovnih pojmova vezanih uz jezike.

Neka je Γ neki alfabet (tj. proizvoljan neprazan skup). Elemente skupa Γ nazivamo simboli. Svaki konačan niz elemenata od Γ nazivamo riječ. Sa Γ^* označavamo skup svih riječi skupa Γ. Pretpostavljamo da za svaki alfabet Γ skup Γ^* sadrži praznu riječ, tj. riječ koja ne sadrži niti jedan element od Γ. Praznu riječ označavamo sa $\sqcup$. Ako je alfabet Γ konačan ili prebrojiv skup, tada je skup svih riječi Γ^* prebrojiv skup. Svaki podskup skupa Γ^* nazivamo jezik. Duljina neke riječi α je broj simbola koje sadrži. Duljinu riječi α označavamo sa $|\alpha|$.

Konačni automat je uređena petorka $M = (Q, \Gamma, \delta, q_0, F)$, gdje je redom:

- Q konačan skup, čije elemente nazivamo stanja;
- Γ je konačan skup koji nazivamo alfabet konačnog automata;
- funkciju δ nazivamo funkcija prijelaza, pri čemu je $\delta : Q \times (\Gamma \cup \{\sqcup\}) \to Q$;
- $q_0 \in Q$ je istaknuti element skupa stanja koji nazivamo početno stanje;
- $F \subseteq Q$ nazivamo skup prihvaćajućih stanja.

Neformalno govoreći, možemo reći da se konačni automat sastoji od četiri dijela: trake na koju se upisuje ulazna riječ, glave koja čita redom simbole ulazne riječi, memorije koja se može naći u jednom od konačno mnogo stanja, te kontrolne jedinice koja ovisno o trenutnom stanju i pročitanom simbolu prebacuje automat u novo stanje. Opišimo kratko rad konačnog automata. Na početku rada automat je u početnom stanju q_0, te je glava za čitanje na početnom simbolu ulazne riječi. Nakon startanja automata, kontrolna jedinica mijenja stanje ovisno o trenutnom stanju i pročitanom simbolu. Zatim se glava pomiče na idući simbol riječi. Ako glava pređe preko zadnjeg simbola ulazne riječi, tada se automat zaustavi.

Neka je $M = (Q, \Gamma, \delta, q_0, F)$ konačan automat. Kažemo da automat M prihvaća neku riječ $\alpha \in \Gamma^*$ ako automat M s riječi α kao ulazom stane u nekom od prihvaćajućih stanja iz zadanog skupa F. Skup svih riječi koje prihvaća neki zadani konačni automat M označavamo sa $L(M)$. Kažemo da je neki jezik L regularan ako postoji konačan automat M tako da vrijedi $L = L(M)$. (Regularni jezici se mogu zadati i pomoću regularnih izraza; vidi zadatak 2 na strani 148).

Konačni automati koje smo do sada razmatrali nazivaju se i deterministički konačni automati. Nedeterministički konačni automat je uređena petorka $(Q, \Gamma, \delta, q_0, F)$, gdje sve komponente imaju isto značenje kao i kod determinističkog automata, jedino

je funkcija prijelaza sada zadana kao $\delta : Q \times (\Gamma \cup \{\sqcup\}) \to \mathcal{P}(Q)$. Nedeterministički konačni automat, osim što može preći u više stanja, može obavljati i tzv. ϵ–prijalaze, tj. ne mora se glava pomicati na sljedeći simbol ulazne riječi, a stroj može prijeći u novo stanje. Važno je istaknuti da za svaki nedeterministički konačni automat N postoji deterministički konačni automat M tako da vrijedi $L(N) = L(M)$.

Lema o pumpanju za regularne jezike. Neka je L regularni jezik. Tada postoji $k \in \mathbb{N}$ tako da za svaku riječ $z \in L$ koja ima najmanje k simbola, postoje riječi u, v, w tako da je $z = uvw,\ \ v \neq \sqcup,\ \ |uv| \leq k$, te za sve $n \in \mathbb{N}$ vrijedi $uv^n w \in L$.

Primjenom leme o pumpanju lako je provjeriti da jezik $\{0^n 1^n : n \in \mathbb{N}\}$ nije regularan.

1.4 Turingov stroj

Kako odrediti vrijeme izvođenja nekog algoritma? Ako bi mjerili vrijeme izvođenja neke implementacije algoritma, morali bi voditi računa o kojem programskom jeziku se radi. Zatim, morali bi znati koji prevodioc programskog jezika koristimo. Naravno, ne bi smjeli zaboraviti istaknuti koje računalo smo koristili. Računala se razvijaju velikom brzinom i novi modeli dolaze na tržište svakodnevno. Sigurno se slažete da bi bilo vrlo teško odrediti programski jezik, njegov prevodioc i vrstu računala koji bi nam služili za određivanje vremenske složenosti svakog algoritma. Kako bi se izbjegla potreba stalnog biranja idealne platforme za testiranja algoritama, analiza algoritama bazira se na teorijskom računalu. Najčešće se razmatraju Turingovi strojevi. Pojam Turingovog stroja definirao je 1936. godine Alan Turing.

Slika 1.1: Alan Turing (1912.–1954.)

Prije formalne definicije, dajemo intuitivni opis Turingovog stroja. Možemo reći da je Turingov stroj automat koji manipulira simbolima na beskonačnoj traci, pri čemu su

ti simboli iz fiksiranog konačnog alfabeta. Traka je podijeljena u registre. U svakom registru može biti zapisan samo jedan simbol. Traka je beskonačna slijeva i zdesna. U jednom trenutku glava stroja može promatrati samo jedan registar. Turingov stroj ima samo konačan broj stanja. Turingov stroj može izvršavati četiri vrste akcije:

a) pomaknuti glavu za jedan registar lijevo;

b) pomaknuti glavu za jedan registar desno;

c) pročitati simbol iz registra kojeg promatra;

d) napisati simbol (iz danog skupa simbola) u registar (odnosno, izbrisati postojeći simbol i napisati novi).

Turingov stroj možemo zadati tablicom iz koje se može iščitati skup stanja i dozvoljenih simbola. U sljedećem primjeru upravo dajemo definiciju traženog Turingovog stroja pomoću tablice.

Primjer 1.2. *Na traci se nalazi zapisan prirodan broj veći od nule u unarnom zapisu (npr. broj 5 je zapisan kao IIIII). Definirat ćemo Turingov stroj koji određuje binarni zapis tog broja. Pretpostavljamo da se glava stroja na početku nalazi na krajnjem lijevom znaku.*

Prvo dajemo opis rada stroja po koracima, a onda ćemo ga točno zadati tablicom.

1. *pomak na zadnji desni znak I;*

2. *obrisati zadnji desni znak I;*

3. *pomak na prvo lijevo prazno mjesto;*

4. *na prazno mjesto pišemo 1;*

5. *pomak na zadnji desni znak I;*

6. *obrisati zadnji desni znak I;*

7. *pomak lijevo do prve znamenke zdesna;*

8. *ako je znamenka 0 tada je treba obrisati i napisati 1, te primijenimo korak 10.; ako je znamenka 1 tada je treba obrisati i napisati 0, te pomaknuti se jedno mjesto lijevo;*

9. *ako je opet pročitana neka znamenka (0 ili 1) tada se vratimo na korak 8.;*

10. *vraćamo se na korak 5. ako još ima znakova I, a inače stroj stane.*

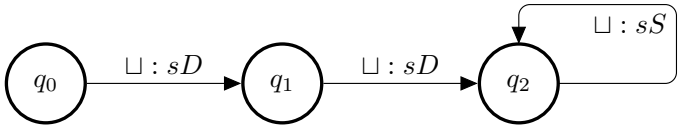
Sada traženi Turingov stroj zadajemo tablicom.

	q_0	q_1	q_2
0	$0Dq_0$	$0Sq_{STOP}$	$1Dq_0$
1	$1Dq_0$	$1Sq_{STOP}$	$0Lq_2$
$\sqcup$	$\sqcup Lq_1$	$\sqcup Sq_{STOP}$	$1Dq_0$
I	IDq_0	$\sqcup Lq_2$	ILq_2

Primijetimo da dani Turingov stroj ima četiri moguća stanja: q_0, q_1, q_2 *i* q_{STOP} *(završno stanje). Skup dozvoljenih simbola je* $\{I,0,1\}$. *Slovo D označava da se glava stroja pomiče desno u sljedećem koraku, slovo L označava da se glava stroja pomiče lijevo, a slovo S označava da glava stroja ostaje na istom registru na traci.*

Turingovi strojevi mogu se zadavati i grafovima. To pokazujemo u sljedećem primjeru.

Primjer 1.3. *Dajemo primjer Turingovog stroja koji na praznu traku napiše tri ista simbola: sss. Prazno mjesto označavamo sa* $\sqcup$. *Traženi Turingov stroj je zadan sljedećim grafom.*



Slika 1.2: Turingov stroj zadan grafom.

Najčešće ćemo Turingove strojeve zadavati neformalnim opisima. U sljedećem primjeru navodimo jednu takvu situaciju.

Primjer 1.4. *Opišimo Turingov stroj koji provjerava je li zapisani prirodni broj na traci palindrom, tj. broj koji je jednak bez obzira ako ga čitamo s lijeva ili zdesna. Primjerice, neki palindromi su 22, 121, 5335 i 478 874. Pretpostavljamo da je broj na traci zapisan u dekadskom zapisu, te da se na početku rada stroja glava nalazi na prvoj lijevoj znamenci.*

Stroj pročita prvu lijevu znamenku. Neka je to neki $k \in \{0,1,\ldots,9\}$. *Tada stroj pređe u stanje* q_k, *te obriše prvu lijevu znamenku. Nakon toga glava za čitanje se miče do zadnje desne znamenke (stroj ostaje u stanju* q_k*). Ako je posljednja desna znamenka različita od k, stroj staje u stanju* q_{NE}. *Ako je posljednja desna znamenka jednaka k, tada je stroj obriše, pomakne se na prvu znamenku lijevo, te se ponovi čitav postupak.*

Sada navodimo formalnu definiciju Turingovih strojeva, i to one vrste koju ćemo najčešće koristiti.

Definicija 1.5. **Turingov stroj** *je uređena sedmorka* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, *gdje je redom:*

- Q *konačan skup čije elemente nazivamo* **stanja**;
- Σ *je konačan skup, čije elemente nazivamo* **ulazni simboli**. *Pretpostavljamo da* Σ *ne sadrži "prazan simbol" kojeg označavamo sa* $\sqcup$;
- Γ *je konačan skup kojeg nazivamo* **alfabet Turingovog stroja**. *Pretpostavljamo da je prazan simbol* $\sqcup$ *element od* Γ, *te* $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ *je proizvoljna funkcija koju nazivamo* **funkcija prijelaza**;
- q_0 *je element iz* Q *i nazivamo ga* **početno stanje**;
- q_{DA} *je element skupa* Q *i nazivamo ga* **stanje prihvaćanja**;
- q_{NE} *je element skupa* Q *i nazivamo ga* **stanje odbijanja**.

Uočimo da je funkcija δ totalna funkcija. To, primjerice, znači da je definirano $\delta(q_{DA}, s)$, tj. čini se da stroj nastavlja raditi i nakon završnog stanja. Naglasimo ponovno da je to formalna definicija, a intuitivni opis stroja je nešto drugo. Istaknimo još da slovo L označava pomak glave za čitanje jedno mjesto lijevo na traci, D pomak desno na traci, a S znači da glava za čitanje ostaje na istom mjestu.

Primijetite da smo definirali da Turingov stroj ima samo dva završna stanja: q_{DA} i q_{NE}. Ponekad se takvi strojevi nazivaju odlučitelji. Kako bi mogli razmatrati primjerice probleme optimizacije, ili pak izračunljivost funkcija, tada se razmatraju strojevi koji se nazivaju pretvarači (ako se zaustave tada je riječ koja je preostala na traci zapravo izlazni rezultat). Za funkcije oblika $f : S \subseteq \mathbb{N}^k \to \mathbb{N}$ se definira pojam Turing–izračunljivosti na sasvim analogni način kao i RAM–izračunljivost. Važno je naglasiti da se klase RAM–izračunljivih i Turing–izračunljivih funkcija poklapaju (vidi zadatke u Dodatku na strani 150).

Definicija 1.6. *Kažemo da* **Turingov stroj** $T = (Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$ **prepoznaje neku riječ** $w \in \Gamma^*$ *ako postoji konačan niz parova* $(r_0, s_0), \ldots, (r_m, s_m) \in Q \times \Gamma$, *te konačan niz simbola* $I_1, \ldots, I_m \in \{L, D, S\}$ *tako da vrijedi:*

a) $r_0 = q_0$ *i* s_0 *je prvi lijevi simbol riječi* w;

b) za svaki $j \in \{0, \ldots, m-1\}$ *imamo* $\delta(r_j, s_j) = (r_{j+1}, s_{j+1}, I_{j+1})$ *i* $r_j \notin \{q_{DA}, q_{NE}\}$;

c) $r_m = q_{DA}$.

Za proizvoljan Turingov stroj T *sa* $L(T)$ *označavamo skup svih riječi koji* T *prepoznaje. Kažemo da neki Turingov stroj* T **prepoznaje jezik** L *ako vrijedi* $L = L(T)$. *Za neki jezik* L *kažemo da je* **Turing–prepoznatljiv**, *ili samo kratko* **prepoznatljiv**, *ako postoji Turingov stroj koji ga prepoznaje.*

Svaki konačan jezik je Turing–prepoznatljiv. Jezik $\{0^n1^n : n \in \mathbb{N}\}$ je Turing–prepoznatljiv.

Nadalje ćemo reći da neki **Turingov stroj staje** s nekim danim ulazom, ako postoje nizovi kao u prethodnoj definiciji, pri čemu je $r_m = q_{DA}$ ili $r_m = q_{NE}$. Ako takvi nizovi ne postoje, reći ćemo da **Turingov stroj radi vječno** s danim ulazom.

Ako Turingov stroj T prepoznaje riječ w, tj. stroj T s ulazom w staje u konačno mnogo koraka u završnom stanju q_{DA}, tada kažemo još da Turingov stroj T **prihvaća riječ** w. Ako Turingov stroj T s ulazom w staje u konačno mnogo koraka u stanju q_{NE}, tada još kažemo da Turingov stroj **odbija riječ** w.

Primijetimo da ako Turingov stroj T prepoznaje jezik L, te je $w \in \Gamma^* \setminus L$, tada Turingov stroj T koji na početku kao ulazni podatak ima na traci zapisanu riječ w uopće ne mora stati, tj. ne mora nužno završiti sa stanjem q_{NE}.

Definicija 1.7. *Za neki jezik $L \subseteq \Gamma^*$ kažemo da je* **Turing–odlučiv**, *ili kratko* **odlučiv**, *ako postoji Turingov stroj T koji ga prepoznaje, te za svaku riječ $w \in \Gamma^* \setminus L$ stroj T s ulazom w staje u završnom stanju q_{NE}. Ako neki jezik nije odlučiv tada još kažemo da je* **neodlučiv**.

U literaturi se Turing–prepoznatljivi jezici nazivaju i rekurzivno prebrojivi jezici. Razlog tome je rekurzivna prebrojivosti skupa $\{f(w) : w \in L\}$ za svaki Turing–prepoznatljiv jezik L, gdje je f proizvoljno kodiranje riječi jezika L s prirodnim brojevima. Turing–odlučivi jezici se nazivaju i rekurzivni jezici.

Znamo da svaki rekurzivno prebrojiv podskup od $\mathbb{N}$ ne mora biti rekurzivan. To znači da postoje jezici koji su prepoznatljivi, ali nisu i odlučivi (vidi zadatak 4 na strani 15).

Primjer 1.8. *Lako je pokazati da su sljedeći jezici odlučivi.*

- *Skup svih riječi zadanog konačnog alfabeta.*
- $\{x \in \Gamma^* : |x|$ *je prim broj* $\}$, *gdje je* Γ *proizvoljan konačan skup.*
- *Jezik kojeg prepoznaje neki konačan automat, tj. regularni jezik.*
- *Jezik koji je generiran s konteksno neovisnom gramatikom.*

Kako bismo mogli navesti neke primjere neodlučivih jezika moramo prvo definirati kodiranje Turingovih strojeva. Neka je $T = (Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$ proizvoljan Turingov stroj. Neka je $Q = \{q_0, q_1, q_2, \ldots, q_m, q_{m+1}, q_{m+2}\}$, pri čemu je $q_{m+1} = q_{DA}$ i $q_{m+2} = q_{NE}$. Definiramo kodiranje stroja T pomoću nizova nula i jednica ovako:

- svakom stanju q_i pridružujemo niz od $i + 1$ nula;
- simbolu L (oznaka za pomak glave stroja u lijevo) pridružujemo 0, simbolu D pridružujemo 00, a simbolu S pridružujemo 000;

- ako je $\Gamma = \{s_1, \ldots, s_n\}$, tada simbolu s_i pridružujemo niz od i nula.

Ako je $\delta(q, s) = (q', s', I)$, pri čemu $q, q' \in Q$, $s, s' \in \Gamma$, te $I \in \{L, D, S\}$, tada uređenu petorku (q, s, q', s', I) nazivamo **konfiguracija** Turingovog stroja T. Svakoj konfiguraciji (q, s, q', s', I) pridružujemo niz nula i jedinica na sljedeći način:

"kod stanja q" 1 "kod simbola s" 1 "kod stanja q'" 1 "kod simbola s'" 1 "kod simbola I".

Budući da je svaki Turingov stroj jedinstveno određen konačnim nizom konfiguracija $k_1, \ldots, k_p$ (skupovi Q i Γ su konačni), te možemo zadati da je niz konfiguracija uređen leksikografski, tada Turingovom stroju T na jedinstveni način pridružujemo sljedeći kod:

1 "kod konfiguracije k_1" 1 "kod konfiguracije k_2" 1 ... 1 "kod konfiguracije k_p" 1

Primijetite da prilikom kodiranja nismo uzeli u obzir skup Σ, tj. nismo ga kodirali. Razlog tome je to što nam u daljnim razmatranjima skup Σ neće biti bitan.

Kod Turingovog stroja T označavamo sa $\langle T \rangle$. Notacija sa šiljatim zagradama se na prirodan način proširuje. Ako su T i T' dva Turingova stroja, tada $\langle T, T' \rangle$ označava njihov kod iz kojeg se mogu rekonstruirati oba stroja. Ako je T stroj i w neka riječ, tada s $\langle T, w \rangle$ označavamo kod stroja T koji na ulazu ima riječ w.

Postoji više načina dokazivanja neodlučivosti nekog jezika. Neodlučivost nekog jezika možemo dokazivati dijagonalnim postupkom. Zatim, dokaz neodlučivosti nekog jezika možemo provesti svođenjem njegove odlučivosti na odlučivost nekog jezika za koji smo prije dokazali da je neodlučiv. Za dokaz neodlučivosti nekog jezika ponekad je vrlo zgodno primijeniti Riceov teorem. Prije teorema navodimo u sljedećoj lemi jedan jezik čiju neodlučivost ćemo koristiti u dokazu Riceovog teorema. Lema se lako dokazuje primjenom dijagonalnog postupka.

Lema 1.9. *Jezik* $A_{TM} = \{\langle T, w \rangle : T$ *je Turingov stroj koji prihvaća riječ* $w\}$ *je neodlučiv.*

Dokaz. Pretpostavimo da stroj A odlučuje jezik A_{TM}. Ako završno stanje stroja A s ulazom $\langle T, w \rangle$ označimo s $A(\langle T, w \rangle)$, tada vrijedi

$$A(\langle T, w \rangle) = \begin{cases} q_{DA}, & \text{ako stroj } T \text{ prihvaća riječ } w; \\ q_{NE}, & \text{inače.} \end{cases}$$

Definiramo novi Turingov stroj D koji s ulazom $\langle T \rangle$ radi ovako:

1. prvo simulira rad stroja A s ulazom $\langle T, \langle T \rangle \rangle$;
2. ako stroj A prihvati riječ $\langle T, \langle T \rangle \rangle$ tada definiramo da stroj D odbije riječ $\langle T \rangle$.

3. ako stroj A odbije riječ $\langle A, \langle A \rangle\rangle$ tada definiramo da stroj D prihvaća riječ $\langle T \rangle$.

Kratko možemo zapisati rad stroja D ovako:

$$D(\langle T \rangle) = \begin{cases} q_{DA}, & \text{ako stroj } T \text{ ne prihvaća riječ } \langle T \rangle; \\ q_{NE}, & \text{ako stroj } T \text{ prihvaća riječ } \langle T \rangle. \end{cases}$$

(Primijetite da prethodni skraćeni zapis ne može biti definicija stroja D. Egzistenciju stroja D osigurava pretpostavka o egzistenciji stroja A.)
Tada posebno imamo:

$$D(\langle D \rangle) = \begin{cases} q_{DA}, & \text{ako stroj } D \text{ ne prihvaća riječ } \langle D \rangle; \\ q_{NE}, & \text{sko stroj } D \text{ prihvaća riječ } \langle D \rangle. \end{cases}$$

Time je dobivena kontradikcija. Q.E.D.

Primijetite da je u prethodnom dokazu korišten dijagonalni postupak. G. Cantor je primjenom dijagonalnog postupka dokazao da je skup $\mathbb{R}$ neprebrojiv. K. Gödel je koristio dijagonalni postupak prilikom dokaza nepotpunosti Peanove aritmetike. Egzistencija funkcije $f : \mathbb{N} \to \mathbb{N}$ koja nije izračunljiva također se dokazuje primjenom dijagonalnog postupka (vidi stranu 4).

Teorem 1.10. (H. G. Rice, 1951.) *Neka je L proizvoljan jezik koji ima sljedeća svojstva:*

a) za sve Turingove strojeve T_1 i T_2, za koje je $L(T_1) = L(T_2)$, vrijedi:

$$\langle T_1 \rangle \in L \quad \textit{ako i samo ako} \quad \langle T_2 \rangle \in L;$$

b) postoje Turingovi strojevi T_1 i T_2 takvi da je $\langle T_1 \rangle \in L$ i $\langle T_2 \rangle \notin L$.

Tada je jezik L neodlučiv.

Dokaz. Pretpostavimo da je jezik L odlučiv. Cilj nam je dokazati da bi tada jezik A_{TM}, koji je definiran u prethodnoj lemi, trebao biti odlučiv. U tu svrhu ćemo konstruirati stroj S koji će odlučivati jezik A_{TM}. Neka je Γ konačan alfabet takav da je $L \subseteq \Gamma^*$. Neka je T Turingov stroj koji odlučuje jezik L. Tada za svaku riječ $x \in \Gamma^*$ vrijedi:

$$T(x) = \begin{cases} q_{DA}, & \text{ako } w \in L; \\ q_{NE}, & \text{ako } w \in \Gamma^* \setminus L. \end{cases}$$

Označimo sa $T_\emptyset$ neki Turingov stroj za koji vrijedi $L(T_\emptyset) = \emptyset$. Ako $\langle T_\emptyset \rangle \in L$ tada primijetimo da za jezik $\Gamma^* \setminus L$ redom vrijedi:

(1) ako su S_1 i S_2 Turingovi strojevi takvi da $L(S_1) = L(S_2)$ tada iz uvjeta a) iz iskaza teorema imamo sljedeće ekvivalencije:

$$\langle S_1 \rangle \in \Gamma^* \setminus L \;\Leftrightarrow\; \langle S_1 \rangle \notin L \;\Leftrightarrow\; \langle S_2 \rangle \notin L \;\Leftrightarrow\; \langle S_2 \rangle \in \Gamma^* \setminus L;$$

(2) budući da iz b) imamo da postoji stroj T_2 tako da vrijedi $\langle T_2\rangle \notin L$, tada $\langle T_2\rangle \in \Gamma^* \setminus L$;

(3) za stroj $T_\emptyset$ imamo $\langle T_\emptyset\rangle \notin \Gamma^* \setminus L$;

(4) budući da smo pretpostavili da je jezik L odlučiv, tada je i jezik $\Gamma^* \setminus L$ također odlučiv.

Dakle, ako $\langle T_\emptyset\rangle \in L$ tada jezik $\Gamma^* \setminus L$ ispunjava uvjete teorema, te za njega vrijedi $\langle T_\emptyset\rangle \notin \Gamma^* \setminus L$, onda u daljnim razmatranjima promatramo jezik $\Gamma^* \setminus L$. Radi kraćeg zapisa jezika pretpostavit ćemo da vrijedi $\langle T_\emptyset\rangle \notin L$.

Neka je T_1 neki Turingov stroj za koji vrijedi $\langle T_1\rangle \in L$ (takav postoji zbog uvjeta b) u iskazu teorema).

Za svaki stroj M koji na ulazu ima riječ w definiramo stroj M_w. Rad stroja M_w koji na ulazu ima riječ $x \in \Gamma^*$ definiramo ovako:

$$M_w(x) = \begin{cases} q_{DA}, & \text{ako } M(w) = q_{DA} \text{ i } T_1(x) = q_{DA}; \\ q_{NE}, & \text{ako } M(w) = q_{NE}; \\ \text{radi vječno}, & \text{inače.} \end{cases}$$

(Uočimo da ako stroj M s ulazom w ne staje u konačno koraka tada stroj M_w sa svakim ulazom radi bez zaustavljanja. Zatim, ako stroj M s ulazom w staje u stanju q_{DA}, ali stroj T_1 s ulazom x radi bez zaustavljanja, tada ni stroj M_w s ulazom x ne staje u konačno mnogo koraka.)

Definiramo Turingov stroj S koji na ulazu ima riječ $\langle M, w\rangle$, gdje je M Turingov stroj, a $w \in \Gamma^*$, na sljedeći način:

$$S(\langle M, w\rangle) = \begin{cases} q_{DA}, & \text{ako } T(\langle M_w\rangle) = q_{DA}; \\ q_{NE}, & \text{ako } T(\langle M_w\rangle) = q_{NE}. \end{cases}$$

(Uočite da stroj S sa svakim ulazom staje u konačno mnogo koraka, jer po pretpostavci stroj T odlučuje jezik L, pa on staje u konačno mnogo koraka za svaki ulaz.)

Tvrdimo da stroj S odlučuje jezik A_{TM}. U tu svrhu promatramo dva slučaja: $\langle M, w\rangle \in A_{TM}$ i $\langle M, w\rangle \notin A_{TM}$.

Pretpostavimo prvo da za neki stroj M i neku riječ w vrijedi $\langle M, w\rangle \in A_{TM}$. Iz definicije jezika A_{TM} tada slijedi da stroj M prihvaća riječ w. Iz definicije stroja M_w tada imamo $L(M_w) = L(T_1)$. Budući da je $\langle T_1\rangle \in L$, tada iz uvjeta a) iz iskaza teorema slijedi $\langle M_w\rangle \in L$. Budući da stroj T odlučuje jezik L tada stroj T prihvaća riječ $\langle M_w\rangle$. Iz definicije stroja S slijedi da tada stroj S prihvaća riječ $\langle M, w\rangle$.

Promotrimo sada slučaj kada je M Turingov stroj i w riječ tako da vrijedi $S(\langle M, w\rangle) = q_{DA}$. Želimo dokazati $\langle M, w\rangle \in A_{TM}$, tj. stroj M prihvaća riječ w, odnosno da vrijedi $M(w) = q_{DA}$. Iz $S(\langle M, w\rangle) = q_{DA}$ i definicije stroja S slijedi $T(\langle M_w\rangle) = q_{DA}$. Budući da po pretpostavci stroj T odlučuje jezik L tada $\langle M_w\rangle \in L$. Ako bi vrijedilo $M(w) = q_{NE}$ ili stroj M radi s ulazom w beskonačno tada $L(M_w) = \emptyset$. Tada imamo

$L(M_w) = \emptyset = L(T_\emptyset)$ i $\langle M_w \rangle \in L$. Iz uvjeta b) iz iskaza teorema slijedi $\langle T_\emptyset \rangle \in L$, što je kontradikacija. To znači da mora vrijediti $M(w) = q_{DA}$.

Promotrimo sada slučaj kada za neki stroj M i neku riječ w vrijedi $\langle M, w \rangle \notin A_{TM}$. Tada iz definicije jezika A_{TM} slijedi da stroj M ne prihvaća riječ w. Tada su moguće dvije situacije: stroj M s ulazom w radi bez zaustavljanja ili vrijedi $M(w) = q_{NE}$. Ako stroj M s ulazom w radi bez zaustavljanja, tada stroj M_w ne staje u konačno mnogo koraka niti za jedan ulaz, pa je $L(M_w) = \emptyset$. Ako vrijedi $M(w) = q_{NE}$, tada iz definicije stroja M_w slijedi $L(M_w) = \emptyset$. Zaključujemo da u obje situacije vijedi $L(M_w) = \emptyset$, tj. $L(M_w) = L(T_\emptyset)$. Budući da $\langle T_\emptyset \rangle \notin L$ tada iz uvjeta a) iz iskaza teorema slijedi $\langle M_w \rangle \notin L$. Tada stroj T odbija riječ $\langle M_w \rangle$, a onda i stroj S odbija riječ $\langle M, w \rangle$.

Time smo dokazali da stroj S odlučuje jezik A_{TM}, tj. jezik A_{TM} je odlučiv. Time je dobivena kontradikcija s prethodnom lemom. Q.E.D.

Turingove strojeve koji smo do sada razmatrali nazivaju se i **jednotračni deterministički Turingovi strojevi**. Mogu se promatrati i sljedeće varijante Turingovog stroja: višetračni, stroj s višedimenzionalnim trakama, stroj s više glava za čitanje i pisanje, te nedeterministički Turingov stroj. Prilikom rješavanja nekih problema zgodnije je upotrijebiti neku varijantu (najčešće višetračnu ili nedeterminističku verziju).

Kao što smo već naveli, Turingove strojeve koje smo do sada razmatrali nazivamo deterministički Turingovi strojevi. Kod nedeterminističkih Turingovih strojeva funkcija prijelaza poprima vrijednosti u partitivnom skupu. Sada dajemo njihovu formalnu definiciju.

Definicija 1.11. **Nedeterministički Turingov stroj** *je uređena sedmorka* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, *gdje simboli* Q, Σ, Γ, q_0, q_{DA} *i* q_{NE} *imaju isto značenje kao kod determinističkih Turingovih strojeva. No, sada je promijenjena kodomena funkcije prijelaza, tj. imamo* $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$.

Nedeterminizam se najbolje očituje u tome što stroj u nekom stanju i trenutno pročitanim simbolom može postupati na bitno različite načine. Najbolje je mogućnost postupanja na različite načine zamišljati kao grananja.

Definicija 1.12. *Kažemo da* **nedeterministički Turingov stroj** $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$ **prepoznaje riječ** $w \in \Gamma^*$ *ako postoji konačan niz parova* $(r_0, s_0), \ldots, (r_m, s_m) \in Q \times \Gamma$, *te konačan niz simbola* $I_1, \ldots, I_m \in \{L, D, S\}$ *tako da vrijedi:*

a) $r_0 = q_0$ *i* s_0 *je prvi lijevi simbol riječi* w;

b) za svaki $j \in \{0, \ldots, m-1\}$ *imamo* $(r_{j+1}, s_{j+1}, I_{j+1}) \in \delta(r_j, s_j)$ *i* $r_j \notin \{q_{DA}, q_{NE}\}$;

c) $r_m = q_{DA}$.

Za proizvoljan nedeterministički Turingov stroj N sa $L(N)$ označavamo skup svih riječi koji stroj N prepoznaje. Kažemo da neki nedeterministički Turingov stroj N **prepoznaje jezik** *L ako vrijedi $L = L(N)$.*

Možemo kratko reći da neki nedeterministički Turingov stroj N prepoznaje neku riječ w ako taj stroj s ulaznim podatkom w ima svojstvo da barem jedna grana izračunavanja stane u konačno mnogo koraka u završnom stanju q_{DA}.

Reći ćemo da neki nedeterministički stroj s nekim ulazom staje ako sve grane izračunavanja stanu u nekom završnom stanju.

Za svaki nedeterministički Turingov stroj N postoji deterministički Turingov stroj T tako da vrijedi $L(N) = L(T)$. To ćemo posebno istaknuti u teoremu 2.18.

Kada u daljnjim izlaganjima napišemo samo "Turingov stroj", mislimo na jednotračni deterministički Turingov stroj.

Zadaci

1. Neka je $L \subseteq \Gamma^*$ odlučiv jezik. Dokažite da je tada i jezik $\Gamma^* \setminus L$ odlučiv.

2. Neka su L_1 i L_2 odlučivi jezici. Dokažite da su tada jezici $L_1 \setminus L_2$, $L_1 \cap L_2$ i $L_1 \cup L_2$ također odlučivi.

3. Dokažite da su sljedeći jezici odlučivi:

   a) $\{\langle T\rangle : T$ je Turingov stroj koji ima manje od 78 stanja$\}$;

   b) $\{\langle T\rangle : T$ je Turingov stroj koji ima točno 153 stanja$\}$;

   c) $\{\langle T\rangle : T$ je Turingov stroj čiji alfabet sadrži manje od 1000 simbola$\}$;

   d) $\{\langle T\rangle : T$ je Turingov stroj čiji alfabet sadrži točno 33 simbola$\}$;

   e) $\{\langle T\rangle : T$ je Turingov stroj čiji alfabet sadrži više od 17 simbola$\}$;

   f) $\{\langle T_1, T_2\rangle : T_1$ i T_2 su Turingovi strojevi koji imaju jednak broj stanja$\}$;

   g) $\{\langle T_1, T_2\rangle : T_1$ i T_2 su Turingovi strojevi pri čemu je kardinalnost alfabeta stroja T_1 manja od kardinalnosti alfabeta stroja $T_2\}$.

4. Dokažite nerješivost halting problema za Turingove strojeve, tj. dokažite da jezik $HALT_{TM} = \{\langle M, w\rangle : M$ je Turingov stroj s ulaznom riječi w koji staje$\}$ nije odlučiv. Dokažite da je navedeni jezik prepoznatljiv.
   Uputa. Pretpostavite da je jezik $HALT_{TM}$ odlučiv, te tada definirajte Turingov stroj koji odlučuje jezik A_{TM}.

5. Dokažite da sljedeći jezici nisu odlučivi:

   a) $L_1 = \{\langle T, w\rangle : T$ je Turingov stroj koji ne prihvaća riječ $w\}$;

   b) $L_2 = \{\langle T, w\rangle : T$ je Turingov stroj koji odbija riječ $w\}$;

c) $L_3 = \{\langle T, w\rangle : T$ je Turingov stroj koji radi beskonačno s ulazom $w\}$.

Upute. a) Očito $L_1 = \{0,1\}^* \setminus A_{TM}$. Ako bi jezik L_1 bio odlučiv tada bi i jezik A_{TM} također bio odlučiv.
b) Za svaki Turingov stroj T definiramo pripadni "komplementarni" Turingov stroj $\overline{T}$ ovako:

$$\overline{T}(w) = \begin{cases} q_{DA}, & \text{ako } T(w) = q_{NE}; \\ q_{NE}, & \text{ako } T(w) = q_{DA}. \end{cases}$$

(Primijetite da oba stroja T i $\overline{T}$ s nekim ulazom staju, ili pak oba rade bez zaustavljanja.) Očito za svaki Turingov stroj T i svaku riječ w vrijedi: $\langle T, w\rangle \in A_{TM}$ ako i samo ako $\langle \overline{T}, w\rangle \in L_2$. To znači da iz odlučivosti jezika L_2 slijedi i odlučivost jezika A_{TM}.
c) Očito vrijedi $L_3 = \{0,1\}^* \setminus HALT_{TM}$.

6. Dokažite da jezik $E_{TM} = \{\langle T\rangle : T$ je Turingov stroj za koji vrijedi $L(T) = \emptyset\}$ nije odlučiv.

7. Dokažite da jezik $EQ_{TM} = \{\langle T_1, T_2\rangle : T_1$ i T_2 su Turingovi strojevi za koje vrijedi $L(T_1) = L(T_2)\}$ nije odlučiv.
Uputa. Neka je $T_\emptyset$ neki Turingov stroj za koji vrijedi $L(T_\emptyset) = \emptyset$. Očito za svaki Turingov stroj T vrijedi: $\langle T\rangle \in E_{TM}$ ako i samo ako $\langle T_\emptyset, T\rangle \in EQ_{TM}$. Budući da iz prethodnog zadatka znamo da jezik E_{TM} nije odlučiv, tada iz prethodne ekvivalencije slijedi da ni jezik EQ_{TM} nije odlučiv.

8. Dokažite da sljedeći jezici nisu odlučivi:

   a) $\{\langle T\rangle : T$ je Turingov stroj takav da $L(T) \leq 373\}$;

   b) $\{\langle T\rangle : T$ je Turingov stroj takav da $L(T) = 1001\}$;

   c) $\{\langle T\rangle : T$ je Turingov stroj takav da $L(T) \geq 561\}$

   d) $\{\langle T\rangle : T$ je Turingov stroj za koji je $L(T)$ konačan skup$\}$;

   e) $\{\langle T\rangle : T$ je Turingov stroj za koji je $L(T)$ beskonačan skup$\}$.

9. Dokažite da jezik $L = \{\langle T_1, T_2\rangle : T_1$ i T_2 su Turingovi strojevi takvi da vrijedi $L(T_1) \subseteq L(T_2)\}$ nije odlučiv.
Rješenje. Neka je T_0 neki Turingov stroj koji prihvaća samo riječ 0, tj. vrijedi $L(T_0) = \{0\}$. Označimo $L_0 = \{\langle T\rangle : T$ je Turingov stroj i $0 \in L(T)\}$. Očito za svaki Turingov stroj T vrijedi: $\langle T\rangle \in L_0$ ako i samo ako $\langle T_0, T\rangle \in L$. To znači da je za neodlučivost jezika L dovoljno dokazati da jezik L_0 nije odlučiv. U tu svrhu koristimo Riceov teorme. Očito postoje Turingovi strojevi T i T' takvi da vrijedi $0 \in L(T)$ i $0 \notin L(T')$, tj. takvi da je $\langle T\rangle \in L_0$ i $\langle T'\rangle \notin L_0$. Neka su T_1 i T_2 Turingovi strojevi takvi da je $L(T_1) = L(T_2)$ i vrijedi $\langle T_1\rangle \in L_0$. Tada vrijedi $0 \in L(T_1)$. Iz $L(T_1) = L(T_2)$ slijedi $0 \in L(T_2)$, a onda i $\langle T_2\rangle \in L_0$.

10. Dokažite da jezik $L = \{\langle T_1, T_2\rangle : T_1 \text{ i } T_2 \text{ su Turingovi strojevi i } L(T_1) \cap L(T_2) = \emptyset\}$ nije odlučiv.

11. Dokažite da jezik $L = \{\langle T_1, T_2\rangle : T_1 \text{ i } T_2 \text{ su Turingovi strojevi i } L(T_1) \cup L(T_2) = \{0,1\}^*\}$ nije odlučiv.

12. Dokažite da sljedeći jezici nisu odlučivi:

    a) $\{\langle T_1, T_2\rangle : T_1 \text{ i } T_2$ su Turingovi strojevi takvi da postoji riječ koju oba stroja prihvaćaju$\}$;

    b) $L_2 = \{\langle T_1, T_2\rangle : T_1 \text{ i } T_2$ su Turingovi strojevi takvi da postoji riječ koju oba stroja ne prihvaćaju$\}$;

    c) $L_3 = \{\langle T_1, T_2\rangle : T_1 \text{ i } T_2$ su Turingovi strojevi takvi da postoji riječ koju oba stroja odbijaju$\}$.

    Uputa. a) Označimo dani jezik sa L. Neka je T_0 neki stroj koji prihvaća samo riječ 0. Označimo $L' = \{\langle T\rangle : T$ je Turingov stroj koji prihvaća riječ 0$\}$. Primjenom Riceovog teorema lako je dokazati da je jezik L' neodlučiv. Očito za svaki Turingov stroj T vrijedi: $\langle T\rangle \in L'$ ako i samo ako $\langle T_0, T\rangle \in L$.

13. Dokažite da sljedeći jezici nisu odlučivi:

    a) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $L(T)$ regularan$\}$;

    b) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $\{0,1\}^* \setminus L(T)$ regularan$\}$;

    c) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $L(T)$ prepoznatljiv$\}$;

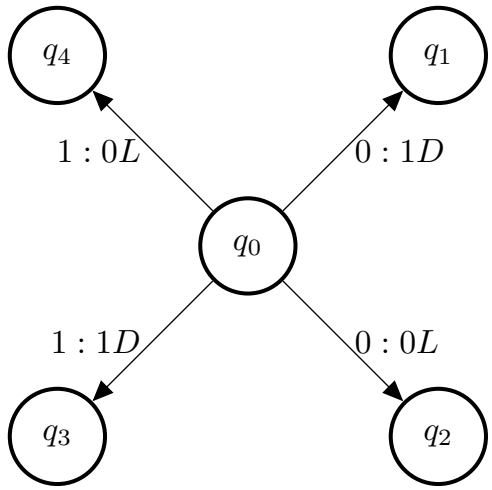
    d) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $\{0,1\}^* \setminus L(T)$ prepoznatljiv$\}$;

    e) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $L(T)$ odlučiv$\}$;

    f) $\{\langle T\rangle : T$ je Turingov stroj za koji je jezik $\{0,1\}^* \setminus L(T)$ odlučiv$\}$.

    Rješenje. a) Označimo dani jezik sa L. Primjenjujemo Riceov teorem. Neka je T Turingov stroj koji odlučuje jezik $\{0^n1^n : n \in \mathbb{N}\}$ (takav stroj postoji!). Iz leme o pumpanju znamo da jezik $L(T)$ nije regularan. Iz toga slijedi $\langle T\rangle \notin L$. Neka je T' neki Turingov stroj za koji vrijedi $L(T') = \{0,1\}^*$. Budući da je $\{0,1\}^*$ regularan jezik tada imamo $\langle T'\rangle \in L$. Neka su T_1 i T_2 Turingovi strojevi za koje vrijedi $L(T_1) = L(T_2)$ i $\langle T_1\rangle \in L$. Tada je jezik $L(T_1)$ regularan. Iz $L(T_1) = L(T_2)$ slijedi da je i jezik $L(T_2)$ regularan, tj. vrijedi $\langle T_2\rangle \in L$.

14. Odredite što radi nedeterministički Turingov stroj koji je zadan sljedećom slikom.



Slika 1.3: Primjer nedeterminističkog Turingovog stroja

Poglavlje 2

Vremenska složenost

U ovom poglavlju definirat ćemo osnovne pojmove vezane uz vremensku složenost nekog Turingovog stroja. Ako je T neki jednotračni Turingov stroj (deterministički ili nedeterministički), te su q i q' dva njegova stanja, s i s' simboli alfabeta, te $I \in \{L, D, S\}$, tada uređenu petorku (q, q', s, s', I) nazivamo **korak stroja** T ako vrijedi $\delta(q, s) = (q', s', I)$, odnosno $(q', s', I) \in \delta(q, s)$ ako se radi o nedeterminističkom stroju.

Definicija 2.1. *Neka je T neki deterministički Turingov stroj koji staje za svaki ulazni podatak.* **Vremenska složenost deterističkog** *stroja T je funkcija $time_T : \mathbb{N} \to \mathbb{N}$, gdje je $time_T(n)$ maksimalan broj koraka koje stroj T napravi za svaki ulazni podatak duljine n.*

Neka je N neki nedeterministički Turingov stroj koji staje za svaki ulazni podatak. **Vremenska složenost nedeterminističkog** *stroja N je funkcija $time_N : \mathbb{N} \to \mathbb{N}$, gdje je $time_N(n)$ maksimalan broj koraka, uzimajući u obzir sve grane, koje stroj N napravi za svaki ulazni podatak duljine n.*

Ako je $time_T$ vremenska složenost stroja T (determinističkog ili nedeterminističkog) tada kažemo da stroj T radi u vremenu $time_T$, tj. da je T $time_T$**–vremenski složen Turingov stroj**.

Vremensku složenost nekog problema (ili jezika) ne možemo definirati kao vremensku složenost najefikasnijeg Turingovog stroja koji rješava taj problem. Dva su barem razloga za to: moguće je ubrzati izračunavanje primjenom većeg alfabeta ili moguće je ubrzati izračunavanje primjenom više traka. Sljedeći teorem upravo govori da je odlučivanje o nekom jeziku uvijek moguće proizvoljno linearno ubrzati. Dokaz teorema je dan u Dodatku.

Teorem 2.2. (O linearnom ubrzanju) [1] *Neka je L proizvoljan odlučiv jezik. Za svaki Turingov stroj T koji odlučuje jezik L i prirodan broj $m \neq 0$ postoji Turingov stroj S koji odlučuje jezik L, te za svaki $n \in \mathbb{N}$ vrijedi:*

$$time_S(n) \leq \frac{time_T(n)}{m} + n.$$

[1] U literaturi na engleskom jeziku teorem se naziva *Linear Speed–Up Theorem*.

Prirodno se postavlja pitanje jesu li moguća i veća ubrzanja od linearnog. Djelomični odgovor na to pitanje daje sljedeći teorem.

Teorem 2.3. (Speed–Up teorem)
Za svaku rekurzivnu funkciju $g : \mathbb{N} \to \mathbb{N}$ postoji odlučiv jezik L tako da za svaki Turingov stroj T koji odlučuje jezik L postoji Turingov stroj S tako da za svaki $n \in \mathbb{N}$ vrijedi: $g(time_S(n)) \leq time_T(n)$.

Dokaz možete pogledati u [7]. Primijetimo da je teorem o linearnom ubrzanju primjenjiv na sve odlučive jezike. Prethodni pak teorem govori da postoji jezik koji je moguće odlučivati na proizvoljno brzom stroju. Za proizvoljni jezik teško je očekivati bolje od linearnog ubrzanja.

2.1 Notacija veliko O

Točno vrijeme trajanja rada nekog Turingovog stroja i potrebnog prostora je unaprijed većinom gotovo nemoguće izračunati, stoga ga procjenjujemo. Jedan prikladan oblik procjene u kojem pokušavamo ocijeniti vrijeme trajanja rada stroja je trajanje rada za velike ulazne podatke, koji zovemo asimptotska analiza. Često je teško, a nekad i nemoguće, izvesti egzaktnu formulu za broj operacija nekog algoritma. Zato proučavamo asimptotsko ponašanje broja operacija kad veličina ulaznih podataka neograničeno raste. Obično se složenost algoritama uspoređuje s nekom od sljedećih standardnih funkcija:

$$\log_2 n, \quad n, \quad n \log_2 n, \quad n^2, \quad n^3, \quad 2^n.$$

Sljedećom tablicom želimo naglasiti koliko su zapravo velike (približne) vrijednosti nekih funkcija.

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
5	3	5	15	25	125	32
10	4	10	40	100	10^3	10^3
100	7	100	700	10^4	10^6	10^{30}
1000	10	10^3	10^4	10^6	10^9	10^{300}

Definicija 2.4. *Neka su $f, g : \mathbb{N} \to \mathbb{R}^+$ dvije funkcije. Kažemo da je funkcija g* **asimptotska gornja međa** *za funkciju f ako postoji $c > 0$ i $n_0 \in \mathbb{N}$ tako da za svaki $n \geq n_0$ vrijedi $f(n) \leq cg(n)$. Oznaka: $f(n) = O(g(n))$.*

Propozicija 2.5. *Ako vrijedi $f_1(n) = O(g_1(n))$ i $f_2(n) = O(g_2(n))$ tada je $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.*

Dokaz. Iz $f_i(n) = O(g_i(n))$ slijedi da postoji $n_i \in \mathbb{N}$ i $c_i > 0$ tako da za svaki $n \geq n_i$ vrijedi $f_i(n) \leq c_i g_i(n)$, za $i \in \{1, 2\}$. Neka je $n_0 = \max\{n_1, n_2\}$ i $c_0 = \max\{c_1, c_2\}$. Tada za svaki $n \geq n_0$ vrijedi: $f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \leq c_0(g_1(n) + g_2(n))$.
Q.E.D.

Propozicija 2.6. *Ako vrijedi* $f_1(n) = O(g_1(n))$ *i* $f_2(n) = O(g_2(n))$ *tada je* $f_1(n) + f_2(n) = O(\max\{g_1(n),\ g_2(n)\})$.

Dokaz. Neka za svaki $n \geq n_1$ vrijedi $f_1(n) \leq c_1 g_1(n)$ i neka za svaki $n \geq n_2$ vrijedi $f_2(n) \leq c_2 g_2(n)$. Neka je $n_0 = \max\{n_1, n_2\}$ i $c_0 = c_1 + c_2$. Tada za svaki $n \geq n_0$ vrijedi: $f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \leq (c_1 + c_2) \max\{g_1(n), g_2(n)\} = c_0 \cdot \max\{g_1(n), g_2(n)\}$.
Q.E.D.

Propozicija 2.7. *Ako vrijedi* $f_1(n) = O(g_1(n))$ *i* $f_2(n) = O(g_2(n))$ *tada je* $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.

Dokaz. Neka za svaki $n \geq n_1$ vrijedi $f_1(n) \leq c_1 g_1(n)$ i neka za svaki $n \geq n_2$ vrijedi $f_2(n) \leq c_2 g_2(n)$. Neka je $n_0 = \max\{n_1, n_2\}$ i $c_0 = c_1 \cdot c_2$. Tada za svaki $n \geq n_0$ vrijedi: $f_1(n) \cdot f_2(n) \leq c_1 g_1(n) \cdot c_2 g_2(n) = c_0 \cdot g_1(n) \cdot g_2(n)$. Q.E.D.

Propozicija 2.8. *Ako je* $p : \mathbb{N} \to \mathbb{R}^+$ *polinom stupnja* k *tada vrijedi* $p(n) = O(n^k)$.

Dokaz. Neka je $p(n) = a_0 + a_1 n + \ldots + a_k n^k$, gdje je $a_k \neq 0$. Označimo $c = |a_0| + |a_1| + \ldots + |a_k|$. Tada za svaki $n \in \mathbb{N} \setminus \{0\}$ vrijedi:

$$\begin{aligned} p(n) &= |p(n)| \leq |a_0| + |a_1| \cdot n + \ldots + |a_k| \cdot n^k \\ &= \left[|a_0|/n^k + |a_1|/n^{k-1} + \ldots + |a_k| \right] \cdot n^k \\ &\leq (|a_0| + |a_1| + \ldots + |a_k|) n^k = c n^k. \end{aligned}$$

Q.E.D.

Propozicija 2.9. *Vrijedi* $\log_2 n = O(n)$.

Dokaz. Indukcijom po n lako je dokazati da vrijedi $\log_2 n \leq n$. Raspisujemo samo korak indukcije: $\log_2(n+1) \leq \log_2(n+n) = \log_2(2n) = \log_2 n + 1 \leq n + 1$, pri čemu smo za zadnju nejednakost koristili induktivnu pretpostavku. Q.E.D.

Definicija 2.10. *Neka je* T *deterministički ili nedeterministički Turingov stroj. Kažemo da je stroj* T*:*

- **(vremenski) polinomni** *ako postoji neki polinom* f *tako da vrijedi* $time_T(n) = O(f(n))$;

- **(vremenski) eksponencijalni** *ako postoji eksponencijalna funkcija g tako da vrijedi $time_T(n) = O(g(n))$.*

U istom smislu ćemo koristiti nazive (vremenski) polinomni i eksponencijalni algoritam.

Napomena 2.11. *Kada govorimo o rješivosti nekog problema na Turingovim strojevima tada pod pripadnim jezikom podrazumijevamo skup svih rješenja tog problema. U daljnjem tekstu govorit ćemo da je neki problem rješiv u polinomnom (eksponencijalnom) vremenu na (ne)determinističkom Turingovom stroju, ako je pripadni jezik takav.*

U sljedećem primjeru navodimo tri algoritma za jedan odlučivi jezik. Vremenska složenost svakog od tih algoritama je različita. Na taj način želimo ponovno naglasiti da je nemoguće definirati vremensku složenost problema. No, sljedeći primjer ukazuje na još jednu važnu činjenicu: vremenske složenosti promatranih algoritama ipak imaju nešto zajedničko, tj. nisu sasvim nepovezane (polinomne su).

Primjer 2.12. *Jezik $L = \{0^k 1^k \ : \ k \in \mathbb{N}\}$ je odlučiv. Navest ćemo tri Turingova stroja koji odlučuju dani jezik, a imaju međusobno različitu vremensku složenost. Pretpostavljamo da je na traci svakog stroja na početku rada zapisana neka riječ $w \in \{0,1\}^*$. Neka je $|w| = n$.*

- *Označimo sa T_1 jednotračni Turingov stroj čiji rad možemo opisati pomoću sljedećih koraka:*

  1. *Glava stroja prijeđe preko cijele riječi te provjeri da li se simbol 0 negdje nalazi desno od simbola 1. Ako se neki simbol 0 nalazi desno od nekog simbola 1, tada stroj odmah staje u stanju q_{NE}.*
  2. *Stroj ponavlja sljedeće sve dok ima simbola 0 i 1 na traci:*
     a) *Glava stroja prelazi po traci te briše uzastopce jedan simbol 0 i jedan simbol 1;*
     b) *Ako na traci postoji još neki simbol 0 nakon što su obrisani svi simboli 1, ili pak postoji simbol 1 kada su obrisani svi simboli 0, tada stroj staje, i to u stanju q_{NE}. Inače stroj završava u stanju q_{DA}.*

  Kako bi odredili vremensku složenost stroja T_1 analiziramo posebno svaki dio rada stroja. U prvom dijelu rada stroj prolazi jednom preko ulazne riječi. Stroju za taj prvi dio treba n koraka. Za vraćanje glave na prvi lijevi simbol ulazne riječi stroju treba novih n koraka. Dakle, prije početka rada u drugom dijelu stroj je napravio $2n$ koraka, tj. vremenska složenost je $O(n)$.

  U drugom dijelu stroj ponavlja pregledavanje i brisanje simbola 0 i 1. Za svaki pregled treba $O(n)$ koraka. Budući da se u svakom pregledu brišu dva simbola

(jedna 0 i jedna 1) tada su nužna $n/2$ pregleda. Dakle, ukupno potrebno vrijeme za taj drugi dio rada je $(n/2) \cdot O(n)$, tj. $O(n^2)$.

U završnom koraku stroj još jednom treba proći po čitavoj riječi kako bi mogao odlučiti da li će prihvatiti ili odbaciti ulaznu riječ w. Za to mu treba još $O(n)$ koraka.

Rezimirajmo: ukupno vrijeme rada stroja T_1 za neku riječ duljine n alfabeta $\{0,1\}$ iznosi: $O(n) + O(n^2) + O(n) = O(n^2)$.

- *Sada opisujemo rad jednotračnog Turingovog stroja koji odlučuje isti jezik, ali mu je vremenska složenost $O(n \log_2 n)$. Navodimo korake rada stroja.*

  1. *Glava stroja prijeđe preko cijele riječi te provjeri da li se simbol 0 negdje nalazi desno od simbola 1. Ako se neki simbol* 0 *nalazi desno od nekog simbola* 1, *tada stroj odmah staje u stanju q_{NE}.*
  2. *Stroj ponavlja sljedeće korake sve dok ima simbola 0 ili 1 na traci.*
     a) *Glava stroja prelazi preko riječi koja je trenutno na traci. Ako je ukupan broj simbola 0 i 1 neparan broj, tada stroj stane u stanju q_{NE}.*
     b) *Glava stroja prelazi preko riječi koja je trenutno na traci i briše svaki drugi simbol 0 počevši od prvog, te briše svaki drugi simbol 1 počevši od prvog.*
  3. *Ako nema više simbola 0 i 1 na traci tada stroj stane u stanju q_{DA}, a inače staje u stanju q_{NE}.*

  U svakom koraku (1., 2a), 2b), 3.) stroj radi $O(n)$ koraka. Koraci 1. i 3. se izvode samo jednom. Korak 2b) se ponavlja najviše $1 + \log_2 n$ puta. Ukupan broj koraka rada ovog stroja približno je jednak: $O(n) + (1 + \log_2 n) \cdot O(n)$, tj. $O(n \log_2 n)$.

- *Na kraju navodimo primjer dvotračnog Turingovog stroja (kasnije ćemo formalno definirati višetračne strojeve) koji odlučuje dani jezik i ima vremensku složenost $O(n)$. Navodimo korake rada stroja.*

  1. *Glava stroja prijeđe preko cijele riječi te provjeri da li se simbol 0 negdje nalazi desno od simbola 1. Ako se neki simbol* 0 *nalazi desno od nekog simbola* 1, *tada stroj odmah staje u stanju q_{NE}.*
  2. *Glava stroja prelazi preko simbola 0 na prvoj traci sve dok ne dođe do prvog simbola 1. Istovremeno kopira simbole 0 na drugu traku.*
  3. *Glava stroja prelazi preko svih simbola 1 na prvoj traci do kraja. Za svaki pročitani simbol 1 na prvoj traci obriše simbol 0 na drugoj traci. Ako su svi simboli 0 na drugoj traci obrisani prije nego što je pročitan zadnji simbol 1 na prvoj traci, tada stoj završi u stanju q_{NE}.*

4. *Ako su sve nule na drugoj traci obrisane stroj staje u stanju q_{DA}. Ako je neki simbol 0 preostao na drugoj traci, tada stroj završava u stanju q_{NE}.*

Očito je vremenska složenost definiranog Turingovog stroja jednaka $O(n)$. Može se pokazati da je to najbolje moguće vrijeme (vidi zadatak 1 na strani 151).

Primjer 2.13. *Odredimo približnu vremensku složenost sljedećih dijelova programa.*

Petlja
```
for (int i=0; i<n; i++)
    sum=sum-i;
```
izvršava se u $O(n)$ koraka.

Petlja
```
for int i=0; i<n*n; i++)
    sum=sum+i;
```
izvršava se u $O(n^2)$ koraka.

Program
```
i=1;
while (i<n) {
    sum=sum+i;
    i=i* 2
}
```
izvršava se u $O(\log_2 n)$ koraka.

Program
```
sum=0
for(int i=0; i<n; i++)
    for(int j=0; j<n; j++)
      sum+=i*j;
```
izvršava se u $O(n^2)$ koraka.

Program
```
i=1;
while(i<=n) {
    j=1;
    while(j<=n) {
      j=j*2;
    }
    i=i+1;
}
```
izvršava se u $O(n \log_2 n)$ koraka.

Zadaci

1. Ispitajte vrijede li sljedeće tvrdnje:

a) $3n + 7 = O(n)$	g) $\log n = O(n)$
b) $n = O(3n + 7)$	h) $n = O(\log n)$
c) $n^2 + n = O(n^2)$	i) $n^k = O(2^n)$
d) $n^2 = O(n^2 + n)$	j) $2^n = O(\log n)$
e) $n = O(0.1n^2)$	k) $\sqrt{n} = O(n)$
f) $0.1n^2 = O(n)$	l) $n = O(n \log \log n)$

Rješenje. Sve tvrdnje su istinite, osim tvrdnji f), h) i j).

2. Dokažite da vrijedi $2^{n+1} = O(3^n/n)$.
Uputa. Indukcijom dokažite da za svaki $n \geq 7$ vrijedi $2^{n+1} \leq 3^n/n$.

3. Dokažite da za sve $a, b > 1$ vrijedi $\log_a n = O(\log_b n)$.

4. Vrijedi li $3^n = O(2^n)$? Vrijedi li $(2n)! = O(n!^2)$?
Rješenje. Ne vrijedi niti jedno od navedenog.

5. Vrijedi li $1 + 2 + \ldots + n = O(n^2)$? Vrijedi li $1^2 + 2^2 + \ldots + n^2 = O(n^3)$?

6. Provjerite vrijedi li, tj. dokažite ili nađite protuprimjere za sljedeće tvrdnje:

   a) Ako je $f(n) = O(g(n))$ tada je $g(n) = O(f(n))$;

   b) Ako postoji $n_0 \in \mathbb{N}$ takav da je za svaki $n \geq n_0$ ispunjeno $\log_2(g(n)) \geq 1$ i $f(n) = O(g(n))$, tada vrijedi $\log_2(f(n)) = O(\log_2(g(n)))$;

   c) Za svaku funkciju $f : \mathbb{N} \to \mathbb{R}^+$ vrijedi $f(n) = O(f^2(n))$.

   Rješenje b). Iz $f(n) = O(g(n))$ slijedi da postoji $c > 0$ i $n_1 \in \mathbb{N}$ tako da za svaki $n \geq n_1$ vrijedi $f(n) \leq cg(n)$. Neka je $n_2 = \max\{n_0, n_1\}$. Tada za svaki $n \geq n_2$ vrijedi $\log_2(g(n)) \geq 1$, tj. $g(n) \geq 2$. Neka je $k_0 \in \mathbb{N}$ najmanji prirodni broj za koji vrijedi $c \leq 2^{k_0}$. Tada za svaki $n \geq n_2$ vrijedi:

$$f(n) \leq cg(n) \leq 2^{k_0} g(n) \leq g^{k_0}(n) \cdot g(n) = g^{k_0+1}(n),$$

   tj. $\log_2(f(n)) \leq (k_0 + 1)\log_2(g(n))$.

7. Za proizvoljne funkcije $f, g : \mathbb{N} \to \mathbb{R}^+$ bili smo definirali što znači $f(n) = O(g(n))$. No, često se oznaka $O(g(n))$ koristi i za klasu svih funkcija f za koje vrijedi $f(n) = O(g(n))$, tj. $O(g(n)) = \{f : \ f(n) = O(g(n))\}$. Kada uzmemo u obzir tu definiciju, tada je jasno što znači $O(f(n)) = O(g(n))$, $O(f(n)) + O(g(n)) = O(h(n))$, $O(f(n)) \cdot O(g(n)) = O(h(n))$, $2^{O(f(n))} = O(g(n))$ i slično. Dokažite da vrijedi:

   a) $O(n) + O(n) + O(n) = O(n)$;

   b) $O(n^2) + O(n) = O(n^2)$;

   c) $(n/2) \cdot O(n) = O(n^2)$;

   d) $O(n) + (1 + \log_2 n) \cdot O(n) = O(n \log_2 n)$.

8. Za proizvoljnu funkciju $f : \mathbb{N} \to \mathbb{R}^+$, takvu da je $f(n) \geq n$, odredite odnos klasa funkcija $O(2^{f(n)})$ i $2^{O(f(n))}$.
Rješenje. Neka je $c > 0$ proizvoljan. Tada očito vrijedi: $c2^{f(n)} = 2^{\log_2 c} \cdot 2^{f(n)} = 2^{\log_2 c + f(n)}$. Neka je n_0 najmanji prirodan broj za koji vrijedi $n_0 \geq \log_2 c$. Tada za svaki $n \geq n_0$ vrijedi: $\log_2 c + f(n) \leq n_0 + f(n) \leq n + f(n) \leq f(n) + f(n)$.

To znači da je za svaki $n \geq n_0$ ispunjeno $c2^{f(n)} \leq 2^{2f(n)}$. Time smo dokazali da vrijedi $O(2^{f(n)}) \subseteq 2^{O(f(n))}$.

Neka $c_1 > 1$ i $c_2 > 0$ proizvoljni. Tada za svaki $n \in \mathbb{N}$ vrijedi:

$$2^{c_1 f(n)} \leq c_2 2^{f(n)} \;\Leftrightarrow\; 2^{(c_1-1)f(n)} \leq c_2,$$

Iz pretpostavke $n \leq f(n)$ i posljednjeg slijedi da $2^{O(f(n))} \subseteq O(2^{f(n)})$ povlači $2^{(c_1-1)n} \leq c_2$, što je nemoguće.

9. Vrijedi li općenito da $f(n) = O(g(n))$ povlači $2^{f(n)} = O(2^{g(n)})$?
Rješenje. Ne. Uzmimo $f(n) = 2n$. Tada očito vrijedi $f(n) = O(n)$. U drugu ruku, $2^{f(n)} = O(2^n)$ sada zapravo znači $2^{2n} = O(2^n)$. Ovo posljednje je ekvivalentno sa: $(\exists c > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) 4^n \leq c \cdot 2^n$, odnosno $2^n \leq c$, što očito nije istina.

10. Neka su $f, g : \mathbb{N} \to \mathbb{R}^+$ funkcije za koje vrijedi $g(n) = O(f(n))$. Dokažite da tada imamo $O(f(n) + g(n)) = O(f(n))$.

11. Dokažite da za sljedeće algoritme pretraživanja i sortiranja vrijede sljedeće procjene vremenske složenosti:

    - određivanje maksimuma zadanog konačnog skupa realnih brojeva: $O(n)$;
    - linearno pretraživanje: $O(n)$;
    - binarno pretraživanje sortiranog niza: $O(\log_2 n)$;
    - bubble sort, selection sort, insertion sort: $O(n^2)$;
    - merge sort, heap sort, quick sort: $O(n \log_2 n)$.

    Rješenje. Ovdje dajemo jedan algoritam za određivanje maksimuma. Kraj svake instrukcije navodimo potreban broj koraka.

procedure	$maximum\ (a_1, \ldots, a_n)$	
	$max := a_1$	1
	$i := 2$	1
	while $i \leq n$	n
	if $max < a_i$ **then**	$n-1$
	$max := a_i$	od 0 do $n-1$
	$i := i + 1$	$n-1$
	return max	1

Ukupan broj koraka može najmanje biti $3n + 1$, a najviše $4n$.

Sada navodimo opis algoritma za linearno pretraživanje, te kraj svake instrukcije navodimo potreban broj koraka.

procedure	*linear_search* $(k, a_1, \ldots, a_n)$	
	$loc := 0$	1
	$i := 1$	1
	while $i \leq n \ \wedge \ loc = 0$	od 1 do $n+1$
	if $k = a_i$ **then**	od 1 do n
	$loc := i$	0 ili 1
	else $i := i + 1$	od 0 do n
	return loc	1

Ukupan broj koraka može najmanje biti 7, a najviše $3n + 4$.

Dajemo i opis algoritma za bubble sort.

procedure	*bubble_sort* $(a_1, \ldots, a_n)$	
	$i := 1$	1
	while $i < n$	n
	$j := 1$	$n-1$
	while $j < n$	$(n-1)n$
	if $a_j > a_{j+1}$	$(n-1)^2$
	$temp := a_j$	0 do $(n-1)^2$
	$a_j := a_{j+1}$	0 do $(n-1)^2$
	$a_{j+1} := temp$	0 do $(n-1)^2$
	$j := j + 1$	$(n-1)^2$
	$i := i + 1$	$n-1$

Ukupan broj koraka može najmanje biti $3n^2 - 2n + 1$, a najviše $6n^2 - 8n + 4$.

2.2 Teoremi o redukciji

U ovoj točki dokazujemo dva teorema koja smo već bili najavili. Prvi teorem govori da se svaki višetračni Turingov stroj može simulirati s nekim jednotračnim strojem, a drugi teorem tvrdi da se svaki nedeterministički Turingov stroj može simulirati s nekim deterministčkim Turingovim strojem. Važno je naglasiti da se u oba teorema navodi kakva je vremenska složenost novog stroja. U oba teorema zahtijevamo da vrijedi $f(n) \geq n$. To je zato jer zahtijevamo da svaki Turingov stroj pročita barem ulaznu riječ.

Definicija 2.14. *Neka je* $k \in \mathbb{N} \setminus \{0, 1\}$. ***k*–tračni Turingov stroj** *je uređena osmorka* $(k, Q, \Gamma, \Sigma, \delta, q_0, q_{DA}, q_{NE})$, *gdje simboli* Q, Γ, Σ, q_0, q_{DA} *i* q_{NE} *imaju isto značenje kao u definiciji jednotračnog Turingovog stroja, a funkcija prijelaza je sada dana sa:* $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, \ D, \ S\}^k$.

Kada kažemo **"višetračni Turingov stroj"** *mislimo na neki k–tračni Turingov stroj.*

Pojmovi kao što su **višetračni Turingov stroj prepoznaje riječ**, **višetračni Turingov stroj prihvaća**, odnosno **odbija neku riječ**, te **višetračni Turingov stroj prepoznaje jezik**, definiraju se sasvim na isti način kao za jednotračne Turingove strojeve, pa ih nećemo ovdje navoditi.

Definicija 2.15. *Kažemo da su dva Turingova stroja S i T (jednotračni, višetračni i nedeterministički; mogu biti iste vrste, a mogu biti i različite vrste)* **ekvivalentna** *ako prepoznaju iste jezike, tj. ako vrijedi $L(S) = L(T)$.*

Teorem 2.16. (O redukciji višetračnog Turingovog stroja na jednotračni) *Neka je $f : \mathbb{N} \to \mathbb{R}^+$ funkcija takva da je $f(n) \geq n$, za svaki $n \in \mathbb{N}$. Tada za svaki višetračni Turingov stroj vremenske složenosti $f(n)$ postoji ekvivalentan $O(f^2(n))$–vremenski složen jednotračni Turingov stroj.*

Dokaz. Neka je $M = (k, Q_M, \Sigma_M, \Gamma_M, \delta_M, q_0, q_{DA}, q_{NE})$ neki k–tračni Turingov stroj čija je vremenska složenost $f(n)$.

Definiramo alfabet stroja S ovako: $\Gamma_S := \Gamma_M \cup \{\dot{s} : s \in \Gamma_M\} \cup \{\sharp\}$. Simbol $\sharp$ je novi simbol, tj. ne pripada skupu Γ_M. Simbol $\sharp$ treba poslužiti kao separator između sadržaja raznih traka stroja M koje kopiramo na jednu traku stroja S. Za svaki simbol $s \in \Gamma_M$ u alfabetu stroja S imamo simbol $\dot{s}$, čime označavamo da se glava stroja M upravo nalazi nad tim simbolom.

Neka je $w = w_1 \dots w_n$ ulazna riječ stroja M koja je zapisana na prvoj traci. Funkciju prijelaza stroja S nećemo formalno definirati, već ćemo dati opis stroja S, te istovremeno navoditi njegovu vremensku složenost. Na početku rada stroj S kopira sljedeću riječ: $\sharp\, \dot{w}_1\, w_2\, \dots\, w_n\, \sharp\, \dot{\sqcup}\, \sharp\, \dot{\sqcup}\, \sharp \dot{\sqcup} \dots\, \dot{\sqcup}\, \sharp$. (Sa $\sqcup$ smo označili praznu riječ, tj. "blank".) Sa $\dot{w}_1$ je označeno da se na prvoj traci stroja M glava za čitanje nalazi na simbolu w_1. Sa $\dot{\sqcup}$ je označeno da su sve trake, osim prve, stroja M prazne, tj. da se glave za čitanje nalaze na praznom simbolu.

Analizirajmo vremensku složenost prvog koraka rada stroja S. Stroj S treba zapisati na traku n–simbola w_i, zatim $(k+1)$–simbol $\sharp$, te $(k-1)$–simbol $\dot{\sqcup}$. To znači da stroju S za prvi korak treba $n + (k+1) + (k-1) = n + 2k$ pomaka po traci. Budući da je broj k konstantan (ne ovisi o broju n) tada je vremenska složenost prvog koraka $O(n)$. Za simulaciju jednog koraka rada stroja M stroj S mora dva puta prijeći po aktivnom dijelu trake. U prvom prolazu stroj S određuje položaj virtualnih glava, tj. određuje k–torku $\dot{s}_1 \dots \dot{s}_k$ kako bi znao argument funkcije prijelaza stroja M. U drugom prolazu po svojoj traci stroj S izvršava ono što mu nalaže funkcija δ_M. Trajanje jednog prolaza po aktivnom dijelu trake stroja S jednako je sumi duljina aktivnih dijelova traka stroja M (uvećano za $k+1$ separatora $\sharp$.) No, aktivni dio niti jedne od k traka stroja M ne može biti dulji od $f(n)$. Prilikom simuliranja nekog koraka rada stroja M može biti nužno pomicanje sadržaja trake stroja S udesno (to je slučaj kada bi virtualna glava trebala doći na neki simbol $\sharp$). Takvih pomicanja može biti najviše k. Dakle, ukupno vrijeme simuliranja jednog koraka stroja M je $O(f(n))$.

Sumiranjem dobivamo vremensku složenost stroja S:

$$\begin{array}{ccccc} O(n) & + & f(n) & \cdot & O(f(n)) \\ \uparrow & & \uparrow & & \uparrow \\ \text{inicijaliziranje} & & \text{broj} & & \text{simulacija} \\ \text{trake stroja } S & & \text{koraka} & & \text{jednog} \\ & & \text{stroja } M & & \text{koraka od } M \end{array}$$

što ukupno iznosi $O(f^2(n))$. Q.E.D.

Očita posljedica prethodnog teorema je da ne moramo promatrati pojam Turing–prepoznatljivih i Turing–odlučivih jezika u odnosu na višetračne strojeve. Tu činjenicu koristimo u dokazu sljedećeg teorema.

Teorem 2.17. *Jezik $L \subseteq \Gamma^*$ je Turing–odlučiv ako i samo ako su jezici L i $\Gamma^* \setminus L$ Turing–prepoznatljivi.*

Dokaz. Pretpostavimo prvo da je jezik L Turing–odlučiv. Neka je T Turingov stroj koji ga prepoznaje, te za svaku riječ $w \in \Gamma^* \setminus L$ stroj staje u završnom stanju q_{NE}. Budući da stroj T prepoznaje jezik L, tada je L Turing–prepoznatljiv. Definiramo stroj T' koji se poklapa u svemu sa strojem T, osim što je $q'_{DA} = q_{NE}$ i $q'_{NE} = q_{DA}$. Tada očito stroj T' prepoznaje jezik $\Gamma^* \setminus L$.

Pretpostavimo sada da su jezici L i $\Gamma^* \setminus L$ Turing–prepoznatljivi. Neka je T_1 Turingov stroj koji prepoznaje jezik L, a neka je T_2 Turingov stroj koji prepoznaje jezik $\Gamma^* \setminus L$. Definiramo dvo–tračni Turingov stroj T koji na jednoj traci simulira rad stroja T_1, a na drugoj traci stroja T_2. Zatim, definiramo da ako za neku ulaznu riječ neki od strojeva T_1 i T_2 stanu, tada stane i stroj T. Primijetimo da će za svaku riječ iz Γ^* stroj T stati. Ako za neku riječ w stroj T_1 završi u stanju prihvaćanja, tada definiramo da stroj T stane u stanju q_{DA}. Ako za neku riječ stroj T_2 završi u stanju prihvaćanja, tada definiramo da stroj T stane u stanju q_{NE}. Ako pak za neku riječ stroj T_1 prvi završi u stanju odbijanja, tada definiramo da stroj T stane u stanju q_{NE}. Ako za neku riječ stroj T_2 stane u stanju odbijanja, tada definiramo da stroj T stane u stanju q_{DA}. Očito stroj T odlučuje jezik L. Q.E.D.

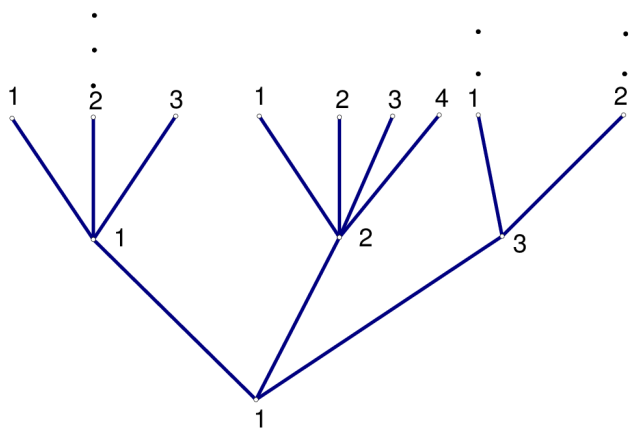
Teorem 2.18. (O redukciji nedeterminističkog stroja na deterministički) *Za svaki nedeterministički Turingov stroj N postoji njemu ekvivalentan determinstički Turingov stroj T.*

Štoviše, ako je N nedeterministički Turingov stroj vremenske složenosti $f(n)$, gdje je $f : \mathbb{N} \to \mathbb{R}^+$ funkcija takva da je $f(n) \geq n$, za svaki $n \in \mathbb{N}$, tada postoji njemu ekvivalentan $2^{O(f(n))}$–vremenski složen deterministički Turingov stroj.

Dokaz. Radi jednostavnosti izlaganja pretpostavljamo da je N neki jednotračni nedeterministički Turingov stroj. Konstruirat ćemo prvo 3–tračni deterministički Turingov stroj T koji je ekvivalentan stroju N. Trake stroja T redom nazivamo: ulazna traka, traka za simulaciju, traka s adresama.

Stroj T treba simulirati sve grane izračunavanja stroja N. No, stroj T ne simulira redom rad svake grane, već to radi po "nivoima".

Broj grananja nekog čvora je broj svih neposrednih sljedbenika tog čvora. Označimo sa b maksimalan broj grananja stroja N. Uočimo da je $b > 1$, jer za $b = 1$ stroj N bi zapravo bio deterministički stroj. Svakom čvoru u stablu izračunavanja stroja N pridružujemo adresu koja je riječ alfabeta $\Sigma_b = \{1, 2, \ldots, b\}$. To ilustriramo na sljedećoj slici.



Slika 2.1: Stablo izračunavanja

Primjerice, niz 331 označava čvor koji se nalazi na grani koja počinje s korijenom stabla (koji je označen s 1), zatim se grana nastavlja sa čvorom koji je označen sa 3, te konačno dolazi do trećeg nivoa gdje je oznaka 3.

Na skupu Σ_b^* promatramo uređaj $\prec$ koji je definiran ovako:

$$w \prec v \Leftrightarrow \begin{cases} |w| < |v| \\ \text{ili} \\ |w| = |v| \ \text{ i } \ w \ \text{ je ispred } \ v \ \text{ u odnosu na leksikografski uređaj.} \end{cases}$$

Svaki čvor stabla izračunavanja je zapravo jedan uređeni par koji određuje neke konfiguracije stroja N.

U svakom trenutku rada stroja T treća traka sadrži neku riječ alfabeta Σ_b. Ta riječ reprezentira čvor koji je oznaka neke konfiguracije stroja N čiji rad se simulira na drugoj traci. Opišimo malo detaljnije rad stroja T po koracima:

1. Na početku rada na prvoj traci stroja T zapisana je riječ w, a druga i treća traka su prazne. Prazna riječ na trećoj traci znači da se prvo treba simulirati dio konfiguracije stroja N koji je definiran korijenom stabla (koji je označen sa 1);

2. Riječ w s prve trake se kopira na drugu;
(na taj način je ulazna riječ stalno zapisana na prvoj traci, pa se može koristiti za simuliranje svake grane izračunavanja).

3. Druga traka se koristi za simulaciju jedne grane izračunavanja stroja N. Prije svakog koraka prvo se pročita simbol na trećoj traci kako bi se odredilo koji izbor stanja i simbola treba napraviti. Ako na trećoj traci više nema simbola ili ako simbol ne predstavlja niti jednu konfiguraciju stroja N, prekine se izvršavanje grane i prelazi se na korak 4. Ako se naiđe na konfiguraciju koja odbija ulaz, također se ide na korak 4. Ako se naiđe na konfiguraciju koja prihvaća ulaz, tada definiramo da i stroj T stane i prihvaća ulaz;

4. Zamijeniti riječ na trećoj traci s riječi koja neposredno slijedi u odnosu na uređaj $\prec$ na skupu Σ_b^*. Prelazi se na korak 2. kako bi se simulirao jedan korak sljedeće grane računanja stroja N.

Dakle, stroj T prelazi redom po čvorovima iste visine. Očito je stroj T ekvivalentan stroju N.

Pretpostavimo sada da je N nedeterministički stroj vremenske složenosti $f(n)$. Provedimo sada analizu vremenske složenosti stroja T.

Za svaki ulaz duljine n duljina proizvoljne grane izračunavanja stroja N je najviše $f(n)$. Ukupan broj listova u stablu izračunavanja stroja N je najviše $b^{f(n)}$ (ima najviše $f(n)$ nivoa, te svaki novi nivo ima najviše b–puta više čvorova u odnosu na prethodni nivo.)

Dokažimo indukcijom da za svaki $b > 1$ i svaki $k \in \mathbb{N}$ vrijedi $1+b+b^2+\ldots+b^k \leq 2b^k$. Baza indukcije je trivijalno ispunjena jer $1 + b \leq b + b = 2b$. Pretpostavimo da tvrdnja vrijedi za neki $k \in \mathbb{N}$. Tada imamo:

$$\begin{aligned} 1 \ + \ b \ + \ b^2 \ + \ \ldots \ + \ b^k \ + \ b^{k+1} \ &\leq \ 2b^k \ + \ b^{k+1} \\ &= \ b^{k+1}(2/b \ + \ 1) \\ &\leq \ b^{k+1}(1+1) = 2b^{k+1}, \end{aligned}$$

pa smo provjerili i korak indukcije.

Iz dokazane pomoćne tvrdnje slijedi da je ukupan broj čvorova u stablu manji od dvostrukog broja listova. Iz toga slijedi da broj svih čvorova u stablu možemo ograničiti sa $O(b^{f(n)})$. Vrijeme potrebno za prolaz od korijena stabla izračunavanja stroja N do nekog čvora je $O(f(n))$. Budući da ispitujemo sve puteve od korijena do svakog čvora tada je ukupno vrijeme rada stroja T manje ili jednako $f(n) \cdot 2b^{f(n)}$. Redom vrijede sljedeće nejednakosti za svaki $n \in \mathbb{N}$:

$$f(n) \cdot 2b^{f(n)} \leq 2b^{f(n)} \cdot b^{f(n)} \leq 2b^{2f(n)} = 2^{1+(\log_2 b)2f(n)},$$

pa je stroj T vremenske složenosti $2^{O(f(n))}$.

Primjenom teorema o redukciji višetračnog stroja na jednotračni slijedi da postoji jednotračni Turingov stroj S koji je ekvivalentan stroju T (a onda je ekvivalentan i stroju N) čija je vremenska složenost jednaka:

$$\left(2^{O(f(n))}\right)^2 \;=\; 2^{2O(f(n))} \;=\; 2^{O(f(n))}.$$

Q.E.D.

Napomena 2.19. *Definirali smo pojam odlučivog jezika u odnosu na jednotračne deterministične Turingove strojeve. Prethodna dva teorema imaju za posljedicu da nije potrebno posebno razmatrati odlučive jezike u odnosu na višetračne, odnosno nedeterminističke, Turingove strojeve.*

2.3 Primjeri algoritama

U ovoj točki navodimo nekoliko algoritma i njihove vremenske složenosti. To su algoritmi koje ćemo kasnije više puta koristiti. Prvo navodimo algoritme za aritmetičke operacije s prirodnim brojevima.

2.3.1 Operacije s prirodnim brojevima

Prvo razmatramo jedan algoritam za zbrajanje dva prirodna broja koji su dani u binarnom zapisu. Promatramo Turingov stroj s 3 trake. Na drugoj i trećoj traci na početku rada stroja neka se nalaze sumandi zapisani u binarnom zapisu, te na početku i kraju označeni simbolom $\sqcup$. Prva traka neka bude na početku prazna (na njoj će Turingov stroj ispisati rezultat zbrajanja). Bez smanjenja općenitosti možemo pretpostaviti da su oba sumanda duljine n (ako je jedan od sumanda manje duljine Turingov stroj ga može dopuniti do duljine n s vodećim nulama u jednom prolazu, dakle u vremenu $O(n)$). Neka se glave stroja na početku rada nalaze na krajnjim desnima znamenkama sumanda. Sljedećom tablicom zadajemo Turingov stroj koji zbraja dva zadana broja.

Trake 2 i 3 / stanje	q_0 (početno stanje)	q_1
(0,0)	0 L L L q_0	1 L L L q_0
(0,1) ili (1,0)	1 L L L q_0	0 L L L q_1
(1,1)	0 L L L q_1	1 L L L q_1
($\sqcup$,$\sqcup$)	$\sqcup$ S S S q_{STOP}	1 S S S q_{STOP}

Tablica 2.1: Turingov stroj koji zbraja dva zadana sumanda.

Tablica u prvom stupcu sadrži uređene parove koje glave čitaju s traka 2 i 3. Primjerice konfiguracija 1 L L L q_0 znači da na prvoj traci glava napiše simbol 1, sve tri glave pomakne za jedno mjesto u lijevo, te stroj prelazi u stanje q_0. Konfiguracija 1 S S S q_{STOP} pak znači da se na prvoj traci piše simbol 1, sve tri glave ostaju na mjestu, te stroj staje. Vidljivo je da stroj u jednom prolazu obavlja zbrajanje (najviše $n+1$ koraka) stoga je $time_T(n) = n+1$, tj. $time_T(n) = O(n)$. Dakle, navedeni algoritam za zbrajanje prirodnih brojeva je polinoman.

Za oduzimanje, množenje i dijeljenje s ostatkom također postoje polinomni algoritmi (vidi [5]).

2.3.2 Euklidov algoritam

Neka su a i b dva prirodna broja takva da je $a \leq b$. Euklidovim algoritmom određuje se najveći zajednički djeljitelj brojeva a i b, kojeg kratko označavamo sa $nzd(a,b)$. Euklidov algoritam glasi:

- Ako je $a = 0$, onda je najveći zajednički djelitelj od a i b jednak $nzd(a,b) = b$.
- Ako je $a > 0$, cjelobrojno dijelimo b sa a, i ostatak tog dijeljenja označimo sa r. Tada je $nzd(a,b) = nzd(r,a)$, pa je stoga dovoljno odrediti najveći zajednički djelitelj od r i a.

Budući da je $r < a$, ova rekurzija staje nakon konačno mnogo iteracija i vraća $nzd(a,b)$.

Propozicija 2.20. *Euklidov algoritam se izvodi u polinomnom vremenu. Točnije, treba mu* $O(\log_2 a \ + \ \log_2 b)$ *aritmetičkih operacija s ulazom* (a,b).

Dokaz. Promotrimo izvršavanje Euklidovog algoritma:

$$b := q_0 a \ + \ r_0$$

$$a := q_1 r_0 \ + \ r_1$$

$$r_0 := q_2 r_1 \ + \ r_2$$

$$\vdots$$

Primijetimo da u prvom koraku vrijedi $b \geq a + r_0 > 2r_0$, pa je stoga

$$r_0 < \frac{b}{2}, \quad \text{i nadalje} \quad a r_0 < \frac{ab}{2}.$$

Na isti način dobivamo da u drugom koraku vrijedi $r_0 r_1 < \frac{r_0 a}{2}$. Zaključujemo da za svaki i vrijedi $r_i r_{i+1} < \frac{r_i r_{i-1}}{2}$. Iz čega slijedi da za $k = \lceil \log_2(ab) \rceil$ vrijedi:

$$r_{k-2}r_{k-1} < \frac{ab}{2^k} = \frac{ab}{2^{\lceil \log_2(ab) \rceil}} \leq \frac{ab}{2^{\log_2(ab)}} = \frac{ab}{ab} = 1.$$

Znači, nakon najviše $\log_2(ab)$ koraka, produkt odgovarajućih r_{k-2} i r_{k-1} je strogo manji od 1, tj. jednak je 0 pa algoritam završava. Kako se svaki korak algoritma sastoji od elementarnih računskih operacija (zbrajanje, oduzimanje, množenje i dijeljenje s ostatkom prirodnih brojeva) za koje smo naveli u prethodnom primjeru da su polinomne, slijedi da je ukupno vrijeme algoritma polinomno. Q.E.D.

Promotrimo sada jednu drugu varijantu Euklidovog algoritma. To je verzija koja je, navodno, sličnija izvornoj Euklidovoj ideji od algoritma maloprije. Koraci tog algoritma su sljedeći:

- Ako $a = 0$, tada $nzd(a, b) = b$.
- Ako $a < b$, tada zamijenimo a i b.
- U slučaju $0 < a \leq b$, iznova definiramo b kao $b := b - a$.

Ovaj algoritam daje isti rezultat kao i prvi algoritam, međutim nije polinomne složenosti, već eksponencijalne. U to se možemo osvjedočiti već u slučaju kada je $a = 1$: za $nzd(1, b)$ nam treba b iteracija, što je eksponencijalno veliko obzirom na $\lceil \log b \rceil + 1$, broj znamenki ulaza. Problem se lako uočava na primjeru: za $a = 8$ i $b = 28$, prvi korak prvog algoritma će biti $28 = q \cdot a + r = 3 \cdot 8 + 4$. Drugi će algoritam za isti rezultat napraviti ovo:

$$b := b - a = 28 - 8 = 20$$

$$b := 20 - 8 = 12$$

$$b := 12 - 8 = 4$$

Dakle, za jedan korak prvog algoritma, drugi potroši q koraka (u našem primjeru je to 3 koraka). Zato je vremenska složenost algoritama toliko različita.

Treba navesti još jedno bitno svojstvo Euklidovog algoritma. On, naime, ne vraća samo najveći zajednički djelitelj dvaju brojeva već i brojeve p i q takve da je $nzd(a, b) = pa + qb$. To svojstvo ima više primjena: u dijeljenju u polju klase reziduala modulo m te, na primjer, kod izračunavanja cijelog broja x o kojem znamo samo ostatke cjelobrojnog dijeljenja s relativno prostim brojevima $m_1, ..., m_k$ (koristi se Kineski teorem o ostatcima).

2.3.3 Množenje matrica

Neka su $A = [a_{ij}]$ i $B = [b_{ij}]$ dvije matrice reda n. Ako označimo $A \cdot B = [c_{ij}]$ tada je $c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$. Uočimo da je veličina ulaznog podatka $2n^2$. Standardni algoritam

za množenje matrica, tj. ako jednostavno množimo matrice koristeći definiciju, treba nam $O(n^3)$ množenja. Postoje brži algoritmi za množenje matrica (npr. Strassenov algoritam).

2.4 Univerzalni Turingov stroj

U ovoj točki definiramo pojam univerzalnog Turingovog stroja koji ćemo koristiti kasnije. Zapravo, mi smo već implicitno koristili univerzalne Turingove strojeve kada smo bili rekli da za svaki Turingov stroj M i ulaznu riječ w postoji neki Turingov stroj koji za ulaz ima $\langle M, w\rangle$, te simulira rad stroja M s ulazom w. No, ovdje nam je važno istaknuti kolika je vremenska složenost univerzalnog Turingovog stroja.

Nadalje ćemo prilikom zadavanja Turingovog stroja isticati samo one dijelove koji ga razlikuju od nekog drugog. Primjerice, $M = (5, Q_M, \Gamma_M, \delta_M)$ je oznaka za 5–tračni Turingov stroj čiji je skup stanja Q_M, alfabet trake je Γ_M, te je funkcija prijelaza δ_M. Nismo posebno istaknuli skup Σ_M ulaznih simbola, početno stanje q_0 te završna stanja q_{DA} i q_{NE}.

Definicija 2.21. *Neka je $T = (k+1, Q_T, \Gamma, \delta_T)$ neki $(k+1)$–tračni Turingov stroj, a $S = (k, Q_S, \Gamma, \delta_S)$ neki k–tračni Turingov stroj (imaju isti alfabet Γ). Neka je $p \in \Gamma^*$ proizvoljna riječ. Kažemo da stroj T* **simulira stroj S s programom** *p ako za proizvoljnu riječ $w \in \Gamma^*$ vrijedi:*

- *stroj T s ulazom w na prvoj traci i p na zadnjoj traci stane ako i samo ako stroj S s ulazom w stane;*
- *ako su oba stroja stala, tada na prvih k traka imaju isti sadržaj, te su stala u istom završnom stanju.*

Definicija 2.22. *Neka je $k \in \mathbb{N} \setminus \{0\}$ i T neki $(k+1)$–tračni Turingov stroj nad alfabetom Γ. Kažemo da je T* **univerzalni Turingov stroj** *ako za svaki k–tračni Turingov stroj S nad alfabetom Γ postoji riječ $p \in \Gamma^*$ tako da stroj T simulira stroj S s programom p.*

Teorem 2.23. (O egzistenciji univerzalnog Turingovog stroja)
Za svaki prirodan broj $k \geq 1$ i svaki konačan alfabet Γ postoji $(k+1)$–tračni univerzalni Turingov stroj T.

Ako je S neki k–tračni Turingov stroj s alfabetom Γ koji staje za svaki ulaz, te je $p \in \Gamma^*$ program s kojim stroj T simulira rad stroja S, tada vrijedi: $time_T(n) \leq |p| \cdot time_S(n)$, za svaki $n \in \mathbb{N}$.

Dokaz. Ovdje ćemo dati samo skicu dokaza. Detaljan dokaz teorema možete pronaći, primjerice, u [7].

Neka su zadani $k \geq 1$ i konačan alfabet Γ. Neka je S neki k–tračni Turingov stroj s alfabetom Γ. Po dogovoru smatramo da je na početku rada stroja S ulazni podatak

w zapisan na prvoj traci, glava za čitanje se nalazi na prvom lijevom simbolu riječi w, a sve ostale trake stroja S su prazne.

Opisat ćemo rad stroja T (nećemo detaljno definirati njegov alfabet, a ni skup stanja i funkciju prijelaza). Prije početka rada stroja T na $(k+1)$. traci se zapiše kod početnog stanja stroja S, te kod stroja S. Zatim, na prvoj traci stroja T se zapiše riječ w. Glava za čitanje na prvoj traci stroja T se nalazi na prvom lijevom simbolu riječi w. Tada startamo stroj T. On će prvo na $(k+1)$. traci potražiti konfiguraciju stroja S čije stanje je početno, na prvoj traci stroja S glava za čitanje je na prvom lijevom simbolu riječi w, a na svim ostalim trakama glave su nad praznim simbolima. U daljnjem dijelu koda konfiguracije očita se u koje stanje stroj S prelazi. Kod tog novog stanja se tada zapiše na $(k+1)$. traci stroja T na mjestu koda početnog stanja stroja S. Zatim se na $(k+1)$. traci stroja T očita što će se napisati na svakoj od prvih k traka, te u kojem smjeru će se pomaknuti svaka glava. Analogno dalje.

Želimo naglasiti da na prvih k traka stroj T simulira rad stroja S. Budući da za svaki korak stroja S stroj T na $(k+1)$. traci prelazi preko koda stroja S, tada očito vrijedi: $time_T(n) \leq |\langle S \rangle| time_S(n)$, za svaki $n \in \mathbb{N}$.

Ako je p neki program s kojim stroj T simulira rad stroja S, očito mora vrijediti $|\langle S \rangle| \leq |p|$. Q.E.D.

Kako ne postoji jedinstveno kodiranje Turingovog stroja, tako ne postoji ni jedinstveni univerzalni stroj. Uz mnoštvo mogućnosti kreiranja takvog stroja, postavlja se pitanje najmanjeg (u smislu veličine alfabeta i skupa stanja) univerzalnog stroja. Prvi pokušaj kreiranja čim manjeg stroja dogodio se 1962. godine, kada je M. Minsky definirao univerzalni Turingov stroj s alfabetom od 4 simbola i sa 7 stanja. Danas je poznat univerzalni Turingov stroj s 2 stanja i alfabetom od 5 simbola.

Za $(n+1)$–mjesnu parcijalno rekurzivnu funkciju G kažemo da je univerzalna funkcija ako za svaku n–mjesnu parcijalno rekurzivnu funkciju f postoji $y \in \mathbb{N}$ tako da vrijedi $f(\vec{x}) \simeq G(\vec{x}, y)$. Vrijedi: za svaki $n \in \mathbb{N}$ postoji $(n+1)$–mjesna univerzalna funkcija. Alternativna definicija univerzalnog Turingovog stroja može biti: univerzalni Turingov stroj je Turingov stroj koji izračunava neku univerzalnu funkciju.

2.5 Klase vremenske složenosti

U ovoj točki definiramo za svaku funkciju $f : \mathbb{N} \to \mathbb{R}^+$ pripadne klase vremenske složenosti. Navest ćemo teoreme koji ističu netrivijalnost klasa vremenske složenosti. Posebno razmatramo klase `PTIME` i `NPTIME`.

Definicija 2.24. *Neka je $f : \mathbb{N} \to \mathbb{R}^+$ proizvoljna funkcija. Klasa vremenske složenosti* **DTIME**$(f(n))$ *je skup svih jezika koji su odlučivi s nekim $O(f(n))$–vremenski složenim jednotračnim deterministističkim Turingovim strojem.*

Klasa vremenske složenosti **NTIME**$(f(n))$ *je skup svih jezika koji su odlučivi s nekim $O(f(n))$–vremenski složenim nedeterminističkim Turingovim strojem.*

Iz definicije odmah slijedi tvrdnja sljedeće propozicije.

Propozicija 2.25. *Za svaku funkciju* $f : \mathbb{N} \to \mathbb{R}^+$ *vrijedi:*

$$\mathrm{DTIME}(f(n)) \subseteq \mathrm{NTIME}(f(n)).$$

Prirodno se postavlja pitanje je li, primjerice, skup $\mathrm{DTIME}(n^3) \setminus \mathrm{DTIME}(n^2)$ neprazan. Zatim, možemo pitati postoji li odlučiv jezik L tako da za njega vrijedi $L \in \mathrm{DTIME}(2^n)$ i $L \notin \mathrm{DTIME}(n^5)$. Odgovor na oba pitanja je potvrdan. O tome govori teorem o vremenskoj hijerahiji. Kako bi mogli izreći taj teorem prvo definiramo tzv. "malo o" notaciju.

Definicija 2.26. *Neka su* $f, g : \mathbb{N} \to \mathbb{R}^+$ *proizvoljne funkcije. Pišemo* $f(n) \sim o(g(n))$ *ako je* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. *Kažemo da je* **Turingov stroj** T **vremenske složenosti** $o(f(n))$ *ako vrijedi* $time_T(n) \sim o(f(n))$. *Kažemo da je neki* **jezik** L **odlučiv u vremenu** $o(f(n))$ *ako postoji Turingov stroj vremenske složenosti* $o(f(n))$ *koji ga odlučuje.*

Sljedeći teorem ističe netrivijalnost klasa vremenske složenosti. Teorem se dokazuje primjenom dijagonalnog postupka. Dokaz teorema je dan u Dodatku. Prije iskaza teorema navodimo definiciju još jednog pojma.

Definicija 2.27. *Neka je* $f : \mathbb{N} \to \mathbb{N}$ *funkcija koja je veća ili jednaka* $O(n \log_2 n)$. *Kažemo da je funkcija* f **vremenski konstruktibilna** *ako je funkcija koja riječ* 1^n *preslikava u binarnu reprezentaciju od* $f(n)$ *Turing–izračunljiva na nekom stroju vremenske složenosti* $O(f(n))$.

Teorem 2.28. (O vremenskoj hijerahiji)
Za svaku vremenski konstruktibilnu funkciju f *postoji jezik* L *tako da vrijedi* $L \in \mathrm{DTIME}(f(n))$, *ali* L *nije odlučiv niti na jednom Turingovom stroju* T *za koji vrijedi* $time_T(n) \sim o(f(n)/\log_2 f(n))$.

Dokažite da je za svaki $k \geq 1$ funkcija $n \mapsto n^k$ vremenski konstruktibilna (vidi zadatak 5 na strani 41). Tada tvrdnja sljedećeg korolara jednostavno slijedi iz prethodnog teorema.

Korolar 2.29. *Za svaki* $k \in \mathbb{N}$ *vrijedi* $\mathrm{DTIME}(n^k) \subsetneq \mathrm{DTIME}(n^{k+1})$.

Sljedeći teorem također pokazuje netrivijalnost klasa složenosti $\mathrm{DTIME}(f(n))$. Iz njega primjerice slijedi da za svaki $k \in \mathbb{N}$ postoji odlučivi jezik L tako da $L \notin \mathrm{DTIME}(n^k)$. Zatim, iz sljedećeg teorema slijedi da ne postoji rekurzivna funkcija f tako da za svaki odlučivi jezik L vrijedi $L \in \mathrm{DTIME}(f(n))$.

Teorem 2.30. (O netrivijalnosti vremenske hijerahije)
Za svaku rekurzivnu funkciju $f : \mathbb{N} \to \mathbb{N}$ postoji odlučiv jezik L tako da vrijedi $L \notin$ DTIME$(f(n))$.

Kako bi dokazali prethodni teorem definirat ćemo prvo pojam potpuno vremenski konstruktibilne funkcije, te dokazati dvije leme. Nakon toga ćemo dati dokaz teorema.

Definicija 2.31. *Za funkciju $f : \mathbb{N} \to \mathbb{N}$ kažemo da je* **potpuno vremenski konstruktibilna** *ako postoji Turingov stroj koji za svaki ulaz duljine n radi točno $f(n)$ koraka.*

Potpuno vremenski konstruktibilna funkcija f očito je Turing–izračunljiva, te za svaki $n \in \mathbb{N}$ mora zadovoljavati $f(n) \geq n$, jer smatramo da Turingov stroj treba barem pročitati ulazni podatak i još jedan znak koji označava kraj ulaza. Zatim, svaka potpuno vremenski konstruktibilna funkcija f za koju vrijedi da $f(n)$ je veće od $O(n \log_2 n)$ očito je vremenski konstruktibilna.

Primjer 2.32. *Pokažimo da je funkcija $f(n) = n^2$ potpuno vremenski konstruktibilna. U tu svrhu opisujemo 2–tračni Turingov stroj kojemu je na ulazu riječ duljine n. Ako je riječ duljine 0 ili jedan, tada je jasno kako stroj treba postupiti. Opisujemo rad stroja kada je $n \geq 2$. Stroj prvo kopira riječ s ulaza na drugu traku, s time da u prvom koraku na prvoj traci briše prvi znak ulazne riječi (ukupno n koraka, a na ulazu je ostala riječ duljine $n-1$). Za svaki od preostalih $n-1$ znakova na prvoj traci stroj pročita cijelu riječ na drugoj traci (uočimo da mu za to treba $n(n-1)$ koraka). Dakle, ukupno stroju treba: $n + (n-1) \cdot n = n^2$ koraka.*

Svaka potpuno vremenski konstruktibilna funkcija je rekurzivna. Obrat ne vrijedi. Primjerice, niti jedna rekurzivna funkcija za koju je $f(n) < n$ nije potpuno vremenski konstruktibilna. Pomoću potpuno vremenski konstruktibilnih funkcija lako je ograničiti vremensku složenost drugih Turingovih strojeva. Turingov stroj T čija je vremenska složenost točno $f(n)$ možemo "ugraditi" u druge Turingove strojeve, te stroj T koristimo kao sat. Konkretnije, nekom Turingovom stroju M "dodamo" stroj T tako da se na dodatnim trakama izvršava jedan korak stroja T, dok se na preostalim trakama izvrši jedan korak stroja M.

Lema 2.33. *Ako je funkcija $f : \mathbb{N} \to \mathbb{N}$ potpuno vremenski konstruktibilna, tada je i funkcija f^4 potpuno vremenski konstruktibilna.*

Dokaz. Pokažimo prvo da je funkcija f^2 potpuno vremenski konstruktibilna. Neka je T neki Turingov stroj koji za svaki ulaz duljine n radi točno $f(n)$ koraka. Stroj T modificiramo na sljedeći način:

1. Stroju T dodamo još dvije trake koje su na početku prazne. Nove trake u daljnjem razmatranju nazivamo prva i druga traka.

2. Pokrenemo stroj T s nekim ulazom duljine n. Za svaki korak rada neka se na prvoj traci zapiše jedan simbol različit od praznog. (Uočimo: time je izvršeno $f(n)$ koraka, te je na prvoj traci zapisano $f(n)$ simbola.)

3. Nakon što je stroj T stao, riječ s prve trake se kopira na drugu traku, s time da se u prvom i drugom koraku na prvoj traci obrišu prvi i drugi simbol. (Uočimo: time je napravljeno novih $f(n)$ koraka, te je na prvoj traci riječ duljine $f(n)-2$, a na drugoj traci je riječ duljine $f(n)$.)

4. Za svaki od $f(n) - 2$ simbola na prvoj traci, pročita se cijela riječ na drugoj traci.

Ukupan broj koraka rada upravo upisanog stroja je: $f(n)+f(n)+(f(n)-2)\cdot f(n) = f^2(n)$. Time smo dokazali da je funkcija f^2 potpuno vremenski kontruktibilna.

Primijenimo li opisani postupak na funkciju f^2 dobivamo traženu tvrdnju. Q.E.D.

Lema 2.34. *Za svaku rekurzivnu funkciju $f : \mathbb{N} \to \mathbb{N}$ postoji potpuno vremenski konstruktibilna funkcija g takva da za svaki $n \in \mathbb{N}$ vrijedi $f(n) \leq g(n)$.*

Dokaz. Neka je T_1 Turingov stroj koji za svaku riječ w na ulazu daje kao izlaz njenu duljinu $|w|$ u binarnom zapisu.

Po pretpostavci funkcija f je rekurzivna, pa je onda i Turing–izračunljiva (vidi zadatke u Dodatku). Neka je T_2 Turingov stroj koja izračunava funkciju f (ako je na ulazu broj n u binarnom zapisu, tada stroj T_2 na izlazu ima broj $f(n)$ u binarnom zapisu).

Neka je T_3 stroj koji za ulaz $n \in \mathbb{N}$ u binarnom zapisu, kao izlaz daje unarni zapis broja n. Uočimo da za svaki $n \in \mathbb{N}$ vrijedi $time_{T_3}(n) \geq n$.

Definiramo funkciju $g : \mathbb{N} \to \mathbb{N}$ ovako:

$$g(n) = time_{T_1}(n) \ + \ time_{T_2}(n) \ + \ time_{T_3}(f(n)).$$

Očito za svaki $n \in \mathbb{N}$ vrijedi $f(n) \leq g(n)$. Označimo sa $T_1+T_2+T_3$ Turingov stroj nastao "spajanjem" tri prije opisana Turingova stroja. Tada za svaku riječ w, takvu da je $|w| = n$, stroj $T_1 + T_2 + T_3$ radi točno $g(n)$ koraka. Q.E.D.

Sada dajemo dokaz teorema 2.30. Primjenom leme 2.34. slijedi da možemo pretpostaviti da je f potpuno vremenski konstruktibilna funkcija. Neka je Γ proizvoljan konačni alfabet. Neka je T dvotračni univerzalni Turingov stroj nad alfabetom Γ (takav postoji; vidi teorem 2.23.). Definiramo jezik:

$$L = \{w \in \Gamma^* : \text{ za ulaz } (w, w) \text{ stroj } T \text{ staje u najviše } f^4(|w|) \text{ koraka}\}.$$

Jedan Turingov stroj koji odlučuje jezik L dobivamo malom modifikacijom stroja T. Iz leme 2.33. slijedi da je f^4 potpuno vremenski konstruktibilna funkcija. Tada postoji Turingov stroj T' koji za svaki ulaz duljine n radi točno $f^4(n)$ koraka. Stroju T

ugradimo stroj T' kao "štopericu". Ako za neku riječ w stroj T stane u najviše $f^4(|w|)$ koraka (bez obzira u kojem stanju), tada definiramo da modificirani stroj prihvaća riječ w. Ako pak za neku riječ w novi stroj ne stane u najviše $f^4(|w|)$ koraka, tada definiramo da taj novi stroj odbija riječ w.

Pretpostavimo da vrijedi $L \in \mathrm{DTIME}(f(n))$. Tada postoji neki k–tračni Turingov stroj složenosti $O(f(n))$ koji odlučuje jezik L. Tada iz teorema 2.16. slijedi da postoji ekvivalentni 1–tračni Turingov stroj M složenosti $c \cdot f^2(n)$, za neki $c > 0$. Tada za svaki $n > c$ vrijedi: $cf^2(n) < nf^2(n) \leq f(n) \cdot f^2(n) = f^3(n)$.

Malo modificiramo stroj M tako da novi stroj odlučuje jezik L, te za svaku riječ $w \in \Gamma^*$ radi najviše $f^3(|w|)$ koraka:

- ako je $w \in \Gamma^*$ za koju je $|w| \geq c$, tada koristimo stroj M;
- Riječi $w \in \Gamma^*$ za koje je $|w| < c$ ima konačno mnogo. Modifikacijom funkcije prijelaza stroja M možemo u linearnom vremenu za takve riječi odlučiti pripadaju li jeziku L.

Lako je vidjeti da primjenom stroja M možemo definirati jednotračni Turingov stroj S koji ima sljedeća dva svojstva:

a) $w \in L$ ako i samo ako stroj S s ulazom w ne staje;

b) ako je za neku riječ w stroj S stao, tada je napravio najviše $f^3(|w|)$ koraka.

Neka je $p \in \Gamma^*$ tako da univerzalni Turingov stroj T s programom p simulira rad stroja S.

Pretpostavimo prvo $p \in L$. Tada stroj S s ulazom p ne staje. U drugu ruku, iz definicije jezika L slijedi da stroj T s ulazom (p, p) staje. Budući da stroj T simulira stroj S s programom p, tada bi i stroj S s ulazom p trebao stati, čime je dobivena kontradikcija.

Dakle, mora vrijediti $p \notin L$. Iz definicije stroja S tada slijedi da stroj S s ulazom staje. Iz teorema 2.23. slijedi:

$$time_T(|p|) \leq |p| \cdot time_S(|p|) \leq |p| \cdot f^3(|p|) \leq f(|p|) \cdot f^3(|p|) = f^4(|p|).$$

To znači da stroj T s ulazom (p, p) staje u najviše $f^4(|p|)$ koraka. Iz definicije jezika L tada slijedi $p \in L$. Time je dobivena kontradikcija. Q.E.D.

Sada navodimo još jedan teorem koji ističe netrivijalnost klasa vremenske složenosti. Dokaz teorema možete pogledati primjerice u [7]. Skica dokaza je dana u Dodatku.

Teorem 2.35. (O praznini) [2]
Neka je g rekurzivna funkcija, takva da za svaki $n \in \mathbb{N}$ vrijedi $n \leq g(n)$. Tada postoji rekurzivna funkcija f tako da za svaki $n \in \mathbb{N}$ vrijedi $n \leq f(n)$, te imamo

$$\mathrm{DTIME}((g \circ f)(n)) = \mathrm{DTIME}(f(n)).$$

[2] U literaturi na engleskom jeziku ovaj teorem se naziva Gap theorem.

Zadaci

1. Dokažite da je jezik A_{TM} Turing–prepoznatljiv. Dokažite da jezik $\{0,1\}^* \setminus A_{TM}$ nije Turing prepoznatljiv.

2. Dokažite korolar 2.29.

3. Dokažite da postoje rekurzivne funkcija f i g tako da vrijedi:

   a) $\mathrm{DTIME}(f(n)) = \mathrm{DTIME}(f^2(n))$;

   b) $\mathrm{DTIME}(g(n)) = \mathrm{DTIME}(2^{g(n)})$.

4. Dokažite da je funkcija $n \mapsto n^3$ potpuno vremenski konstruktibilna.
   Rješenje. Neka je T Turingov stroj s tri trake. Na početku rada stroja na prvoj traci je zapisana neka riječ $w_1 \dots w_n$, a druga i treća traka su prazne. Stroj prvo prepiše riječ na drugu i treću traku, te na prvoj traci u prvom koraku obriše simbol w_1. Tako je stroj napravio n koraka. Zatim, za svaki simbol s prve trake stroj pročita riječ s druge trake, te za svaki simbol s druge trake pročita riječ na trećoj traci. To je ukupno: $(n-1) \cdot n \cdot n = n^3 - n^2$ koraka. Nakon toga za svaki simbol na prvoj traci pročita čitavu riječ na drugoj traci (na trećoj traci ne radi ništa). To je još: $(n-1) \cdot n = n^2 - n$ koraka. Ukupno je stroj napravio: $n \ + \ (n^3 - n^2) \ + (n^2 - n) = n^3$ koraka.

5. Dokažite da je za svaki $k \in \mathbb{N}$ funkcija $n \mapsto n^k$ potpuno vremenski konstruktibilna.
   Uputa. Za svaki $k \geq 2$ vrijedi:

$$n^k = n \ + \ \sum_{i=1}^{k-1} (n-1)n^i.$$

6. Dokažite da je funkcija $n \mapsto 2^n$ potpuno vremenski konstruktibilna.
   Rješenje. Neka je T Turingov stroj s tri trake. Pretpostavljamo da je ulazna riječ $w_1 \dots w_n$ zapisana na prvoj traci, te da su druga i treća traka prazne na početku rada stroja. Ako ulazna riječ nije prazna, tada pretpostavljamo da je na početku rada glava za čitanje na prvoj traci nad simbolom w_1. Ovdje opisujemo korake rada stroja T.

   a) Ako je $n = 0$, tj. na prvoj traci je na početku zapisana prazna riječ, tada se glava za čitanje pomakne jedno mjesto udesno i stroj stane.

   b) Ako je $n \neq 0$ tada stroj radi sljedeće:

      1. Simbol w_1 se kopira na drugu traku, te se istovremeno glava na prvoj traci pomakne jedno mjesto udesno. (Time je stroj napravio jedan korak.)

2. Stroj ponavlja sljedeće sve dok na prvoj traci glava ne dođe do praznog mjesta:

   (i) na prvoj traci se glava za čitanje pomakne jedno mjesto udesno;

   (ii) ako je glava na prvoj traci bila nad nekim simbolom s parnim indeksom, tj. na nekom simbolu oblika w_{2k}, tada se dva puta kopira riječ s druge trake na treću. Prilikom drugog kopiranja obriše se riječ s druge trake;

   (iii) ako je glava na prvoj traci bila nad nekim simbolom s neparnim indeksom, tj. na nekom simbolu oblika w_{2k+1}, tada se dva puta kopira riječ s treće trake na drugu traku. Prilikom drugog kopiranja obriše se riječ s treće trake.

   Primijetimo: ako je na drugoj (trećoj) traci bila zapisana neka riječ duljine k, tada će nakon izvršenja ovog dijela rada stroja na trećoj (drugoj) traci biti zapisana riječ duljine $2k$, a druga (treća) traka će biti prazna.

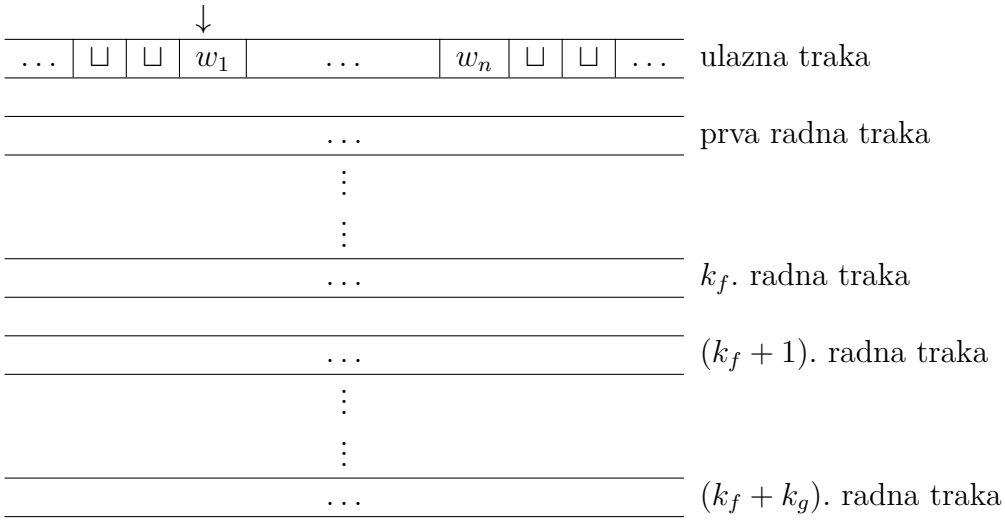
3. Kada je glava na prvoj traci došla do praznine, tada se glava pomakne još jedno mjesto u desno, te stroj stane.

Ukupan broj koraka stroja T koji na ulazu ima riječ duljine n, pri čemu je $n \geq 1$, jednak je: $1 + 2 + 4 + \ldots + 2^{n-1} + 1 = 2^n$.

7. Neka su $f, g : \mathbb{N} \to \mathbb{N}$ potpuno vremenski konstruktibilne funkcije. Dokažite da je tada i funkcija $f + g$ potpuno vremenski konstruktibilna.

Rješenje. Neka su T_f i T_g Turingovi strojevi takvi da za svaki ulaz duljine n stroj T_f radi točno $f(n)$ koraka, a stroj T_g radi točno $g(n)$ koraka. Označimo s Γ_f alfabet stroja T_f, a s Γ_g alfabet stroja T_g. Alfabet Γ_{f+g} stroja T_{f+g} neka je zadan sa $\Gamma_{f+g} = \Gamma_f \cup \Gamma_g \cup \{p\}$, pri čemu $p \notin \Gamma_f \cup \Gamma_g$ (simbol p će nam služiti kao oznaka da se radi o prvom simbolu ulazne riječi). Neka stroj T_f ima k_f traka, a stroj T_g neka ima k_g traka. Definirat ćemo stroj T_{f+g} koji ima $k_f + k_g + 1$ trake.

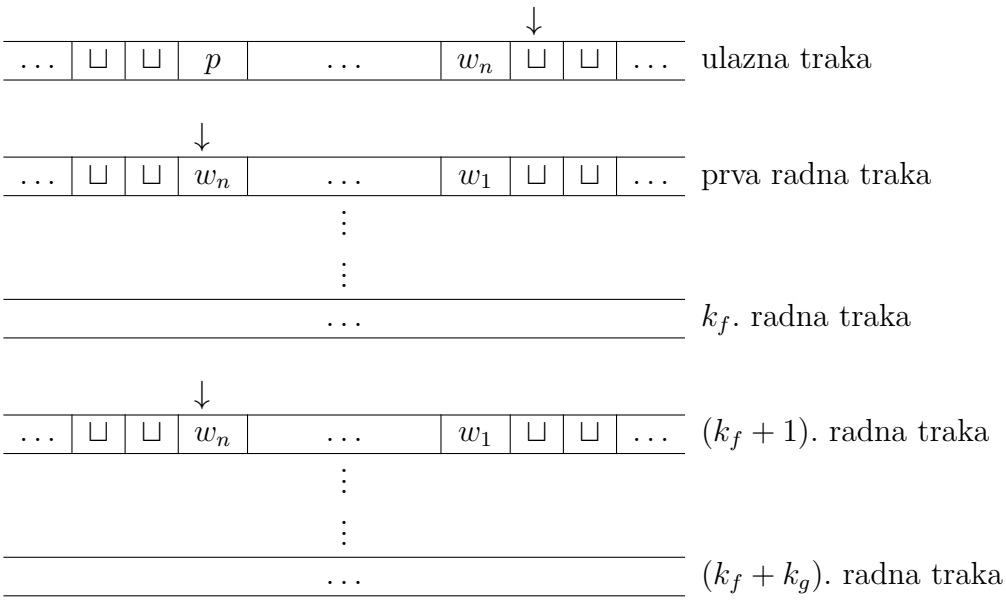
Ulazni podatak $w_1 \ldots w_n$ (neka riječ duljine n) na početku rada stroja T_{f+g} zapisana je na traci koju ćemo nazivati ulazna traka, te se glava za čitanje nalazi na prvom lijevom znaku ulazne riječi. Pretpostavljamo da su ostale trake stroja T_{f+g} prazne.



Izgled traka stroja T_{f+g} prije početka rada

Na početku rada stroja T_{f+g} ulazni podatak s prve trake kopira se na prvu radnu traku, te na $(k_f + 1)$. traku i to zdesna na lijevo. U prvom koraku se umjesto simbola w_1 na ulaznoj traci zapiše simbol p. To znači da se nakon n koraka glava na prvoj traci nalazi na prvoj praznini zdesna iza ulaznog podatka, a na prvoj radnoj i $(k_f + 1)$. traci glave se nalaze na prvom simbolu slijeva.

(Napomena. Ne možemo pretpostavljati da će nakon simulacije rada stroja T_f na ulaznoj traci biti zapisana početna ulazna riječ).



Izgled traka stroja T_{f+g} nakon prvih n koraka

Sada stroj T_{f+g} istovremeno simulira prvih n koraka rada stroja T_f i stroja T_g. Simulacija stroja T_f se odvija na prvih k_f radnih traka, a simulacija rada stroja T_g se odvija na zadnjih k_g radnih traka. Uočimo da nam brojenje točno n koraka omogućava riječ na ulaznoj traci (primijetite da je važno što smo posebno označili prvi lijevi simbol ulazne riječi). Nakon n koraka istovremene simulacije rada oba stroja, prekidamo simulaciju rada stroja T_g, te nastavljamo samo simulaciju rada stroja T_f. Kada završi simulacija stroja T_f stroj T_{f+g} je ukupno napravio $n + f(n)$ koraka. Tada pokrećemo nastavak simulacije rada stroja T_g, te do kraja te simulacije stroj T_{f+g} napravi još $g(n) - n$ koraka (primijetite da je $g(n) - n \geq 0$, jer je po pretpostavci zadataka funkcija g potpuno vremenski konstruktibilna, pa je posebno $g(n) \geq n$ za svaki $n \in \mathbb{N}$).

Stroj T_{f+g} je napravio ukupno: $n + f(n) + g(n) - n = f(n) + g(n)$ koraka.

8. Objasnite zašto se u definiciji vremenski konstruktibilne funkcije zahtijeva da je $f(n)$ veće ili jednako $O(n \log_2 n)$.

Rješenje. Kako bi Turingov stroj mogao izračunati binarnu reprezentaciju od $f(n)$, mora mu biti poznata binarna reprezentacija broja n. To znači da Turingov stroj mora ulaz, koji je oblika 1^n (niz od n jedinica, tj. unarna reprezentacija broja n), pretvoriti u binarnu reprezentaciju od n. Promotrimo sljedeći Turingov stroj S (zbog jednostavnosti, pretpostavimo da je ulaz dan u obliku ♯111...1♯):

S = na ulazu je ♯1^n♯ (glava je na prvom lijevom znaku ♯):

(a) Dok glava stroja nije došla do drugog znaka ♯ (tj. kraja ulaznog niza), pamti parnost broja jedinica, te svaku drugu jedinicu (počevši od prve) zamijeni sa znakom & (znakovi & su ignorirani u sljedećem prolazu). Neka znak 0 predstavlja paran broj jedinica, a znak 1 neparan broj jedinica.

(b) Ako se u ulaznom nizu pojavila barem jedna jedinica, zapiši parnost na prvo slobodno mjesto prije prvog znaka ♯ (prvo slobodno mjesto s lijeve strane ulaza) i vrati se na prvi znak ♯. Ako u ulaznom nizu više nema jedinica (tj. ako su ostali samo znakovi &), onda prekini s radom.

(c) Vrati se na korak 1.

Opisani stroj ulaz 1^n pretvara u binarnu reprezentaciju od n, te je zapisuje s lijeve strane ulaznog niza. U prvom koraku stroj mora proći kroz cijeli ulazni niz počevši od prvog znaka ♯ za što mu treba $n+1$ pomak glave. U drugom koraku stroj mora: vratiti na prvo slobodno mjesto prije prvog znaka ♯, zapisati parnost, te se ponovno vratiti na prvi znak ♯. Za to mu treba: n + 1 pomaka da dođe do prvog znaka ♯, zatim (u najgorem slučaju) $\log_2 n$ pomaka (jer toliko znamenaka ima broj n u binarnom zapisu) da dođe do prvog slobodnog mjesta prije prvog

znaka ♯, te naposljetku još (u najgorem slučaju) $\log_2 n$ pomaka da ponovno dođe do prvog znaka ♯. Dakle, u drugom koraku ukupno imamo $(n+1)+\log_2 n+\log_2 n$ pomaka. U trećem koraku ponavljamo prvi korak i eventualno drugi. Budući da je nakon svakog završetka prvog koraka broj jedinica u nizu duplo manji, prva dva koraka ćemo ponavljati $\log_2 n$ puta. Još trebamo jedan prolaz kroz niz da se uvjerimo da u nizu više nema jedinica, tj. da su ostali samo znakovi &. Za to nam treba $n+1$ pomak. Dakle, zaključujemo da nam za pretvorbu broja n u binarnu reprezentaciju ukupno treba $(n+1+\log_2 n+\log_2 n)\log_2 n+n+1$ pomaka, što je $O(n\log_2 n)$. Upravo to je i razlog zašto se u definiciji vremenski konstruktibilnih funkcija zahtjeva da je funkcija f veća ili jednaka $O(n\log_2 n)$.

2.5.1 Klasa PTIME

U ovoj točki razmatramo klasu problema za koje postoji efikasan ("brzi") algoritam za rješavanje. To je klasa problema koji su odlučivi na nekom polinomno složenom Turingovom stroju. Osim definicije dat ćemo i nekoliko važnih primjera koje ćemo kasnije koristiti.

Definicija 2.36. *S* PTIME*, ili samo kratko s* P*, označavamo klasu svih jezika koji su odlučivi na nekom deterministčkom Turingovom stroju vremenske složenosti* $O(n^k)$*, za neki* $k \in \mathbb{N}$*. Dakle, vrijedi*

$$P = \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(n^k).$$

Vjeruje se da je klasa P invarijantna u odnosu na izbor teorijskog modela izračunavanja (u našem slučaju to je Turingov stroj). Za svaka dva poznata modela izračunavanja pokazalo se da postoji konstanta c koja ima sljedeće svojstvo: ako prvi stroj napravi t koraka, tada drugi stroj napravi $O(t^c)$ koraka (vidi zadatak 2). Neka razmatranja upućuju na to da model izračunavanja koji se bazira na kvantnoj mehanici, nije polinomno ekvivalentan Turingovom stroju.

Komentirat ćemo kratko što zapravo znači "efikasno izračunavanje". Nadamo se da se slažete, da izračunavanja koja se izvršavaju u linearnom ili "kvadratnom" vremenu, smatramo efikasnim. Zatim, prirodno je zahtijevati da je klasa efikasnih algoritama zatvorena za kompoziciju. Cobham je 1974. dokazao da je najmanja klasa problema koja sadrži sve probleme linearne i kvadratne složenosti, te je zatvorena za kompoziciju, upravo klasa P.[3]

[3] Teško ćemo se složiti da klasa $\mathrm{DTIME}(n^{100})$ sadrži samo probleme koje možemo "efikasno" riješiti. Međutim, kada se dokazuje u praksi da neki problem pripada klasi P, obično se pronađe algoritam složenosti n^3 ili n^5. Više puta se znalo dogoditi da se za neki problem pronađe prvo algoritam velike polinomne složenosti (npr. n^{20}). No, najčesće je kasnije pronađen algoritam čija je složenost najviše n^5.

Primjer 2.37. *Jezik* `RELPRIM` *definiramo kao* $\{\langle x, y\rangle : x \text{ i } y \text{ su relativno prosti prirodni brojevi }\}$. *Jezik* `RELPRIM` *možemo odlučiti s Euklidovim algoritmom, za koji smo pokazali da je polinomne složenosti. To znači da je vremenska složenost jezika* `RELPRIM` *također polinomna. Dakle,* `RELPRIM` $\in$ `P`.

Primjer 2.38. *Navedimo neke probleme koji pripadaju klasi* `P`.

1. *zbrajanje, oduzimanje, množenje i dijeljenje s ostatkom prirodnih brojeva (vidi stranu 32 i knjigu [5]);*
2. *određivanje najveće zajedničke mjere dvaju prirodnih brojeva (Euklidov algoritam i njegova složenost, je dan na strani 33);*
3. *ispitivanje povezanosti grafa; traženje najkraćeg puta u grafu (o problemima na grafovima ćemo pričati kasnije);*
4. 2–`SAT`; `SAT`–Horn *(O tome ćemo također govoriti kasnije);*
5. *problem egzistencije rješenja sistema linearnih algebarskih jednadžbi.*

Napomena 2.39. *Neka su* $b_1,\ b_2 \geq 2$ *proizvoljni prirodni brojevi. Za svaki* $n \in \mathbb{N}$ *broj znamenki u prikazu broja* n *u bazi* b_i *je jednak* $\lceil \log_{b_i} n \rceil + 1$. *Budući da vrijedi* $\log_{b_1} n = \dfrac{\log_{b_2} n}{\log_{b_2} b_1}$, *tada se duljine prikaza razlikuju za konstantni faktor. Iz toga zaključujemo da je klasa* `P` *invarijantna za odabir baze u kojoj ćemo prezentirati prirodne brojeve kao ulazne podatke.*

Sada navodimo dva primjera algoritma na grafovima koji su polinomne vremenske složenosti. Prvo navodimo sve potrebne definicije u vezi grafova.

Definicija 2.40. **Usmjereni graf** *je uređeni par* (G, R), *gdje je* G *proizvoljan neprazan konačan skup, a* $R \subseteq G \times G$. *Ako je relacija* R *simetrična tada kažemo još da je graf* **neusmjeren**. *Elemente skupa* G *nazivamo* **čvorovi**, *a elemente skupa* R **bridovi**.

Za graf (G', R') *kažemo da je* **podgraf** *grafa* (G, R) *ako je* $G' \subseteq G$, *te je* $R' \subseteq R \cap G' \times G'$.

Kažemo da postoji **put** *između čvorova* v *i* w *ako postoji konačan niz čvorova* $b_1, \ldots, b_n$ *tako da vrijedi:* $vRb_1Rb_2 \ \ldots \ b_nRw$.

Sada prvo razmatramo problem egzistencije puta između dva čvora u grafu. U tu svrhu definiramo sljedeći jezik:

$$\texttt{PROBLEM PUT} = \{(G, x, y) : G \text{ je graf},\ x, y \in G,\ \text{ postoji put od } x \text{ do } y\}.$$

Lema 2.41. *Vrijedi:* `PROBLEM PUT` $\in$ `P`.

Dokaz. Navodimo jedan algoritam koji za dani graf G i čvorove x i y u polinomnom vremenu određuje da li postoji put od čvora x do čvora y. Ovdje su koraci algoritma:

(1) označi čvor x;

(2) za svaki brid $\{a, b\}$ u kojem je čvor a označen, označi i čvor b;

(3) ako je neki novi čvor označen, vrati se na korak (2);

(4) prihvati ulaz (G, x, y) ako je čvor y označen, a inače ga odbij.

Odredimo vremensku složenost navedenog algoritma. Prvi i četvrti korak se izvode samo jednom, te za njih treba najviše $O(n)$ koraka (sa n smo označili duljinu ulazne riječi koja određuje graf). U drugom koraku provjerava se svaki brid. U grafu sa m čvorova ima približno najviše $O(m^2)$ bridova (ako se radi o potpuno povezanom grafu, tada je svaki čvor povezan sa svim čvorovima, pa ima $2m \cdot (m-1) + m$ bridova). Zatim, u drugom koraku se za svaki brid provjerava je li prvi čvor označen, te se možda označava drugi čvor. Dakle, broj koraka u (2) je $2 \cdot O(m^2)$, tj. $O(m^2)$ koraka. Korak (2) se ponavlja najviše onoliko puta koliko ima čvorova, tj. m puta. Time zaključujemo da drugi korak treba najviše $O(m^3)$ ponavljanja. Budući da očito vrijedi $m = O(\sqrt{n})$, tada je vremenska složenost navedenog algoritma jednaka: $O(n) + O(n\sqrt{n}) + O(n) = O(n\sqrt{n})$. Q.E.D.

Definicija 2.42. *Za graf (G, R) kažemo da je* **povezan** *ako između svaka dva čvora postoji put.* **Ciklus** *u grafu je put koji ima isti početak i kraj.* **Stablo** *je povezan graf bez ciklusa.* **Razapinjajuće stablo** *grafa (G, R) je podgraf koji je stablo i čiji skup čvorova je G. Neka je (G, R) graf i $f : R \to \mathbb{R}^+$. Tada uređenu trojku (G, R, f) nazivamo* **težinski graf**.

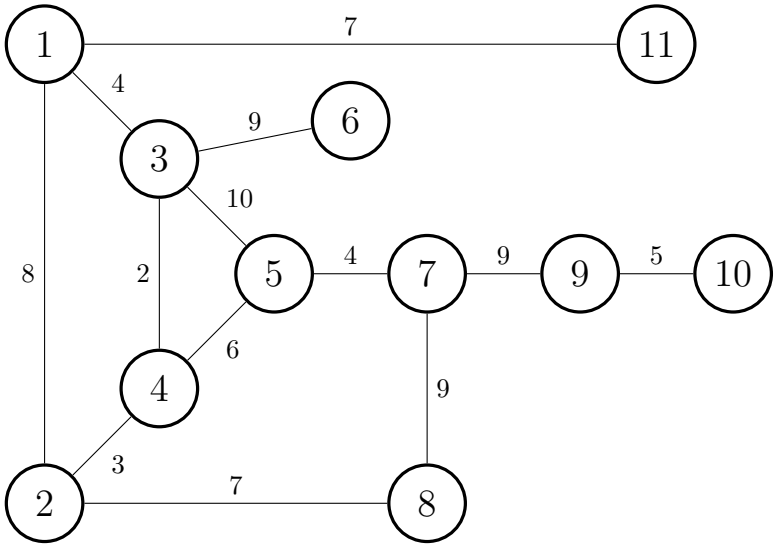
Sada razmatramo jedan polinomni algoritam za određivanje minimalnog razapinjujućeg stabla zadanog težinskog grafa.[4] Problem minimalnog razapinjajućeg stabla sastoji se od određivanja razapinjujućeg stabla čija je ukupna suma težina njegovih bridova minimalna, ako je zadan neki težinski graf. Primijetimo da problem minimalnog razapinjujućeg stabla nije problem odluke. Sada dajemo korake **Primovog algoritma** kojim se određuje minimalno razapinjajuće stablo. Neka je zadan težinski graf (G, R, f). Primov algoritam se sastoji od sljedećih koraka:

1. Iz skupa čvorova G proizvoljno odaberemo početni čvor v, te definiramo stablo $T = \{v\}$.

2. Ponavljamo sljedeći korak sve dok u stablu T nemamo točno $k(G)$ čvorova: stablu T dodajemo brid minimalne težine s jednim krajem u skupu T, a drugim u skupu $G \setminus T$.

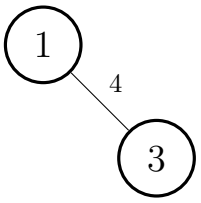
[4] Primjenom Zornove leme lako je dokazati da u svakom povezanom grafu postoji razapinjajuće stablo. Zornova lema glasi: Ako je $(A, <)$ parcijalno uređen skup u kojem za svaki neprazni lanac postoji gornja međa, tada skup A sadrži barem jedan maksimalni element.

Kao jedan primjer izvođenja Primovog algoritma analizirat ćemo određivanje razapinjujućeg stabla grafa kojeg zadajemo na sljedećoj slici.



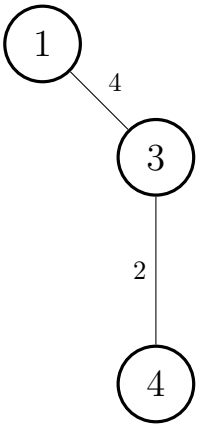
Slika 2.2: Težinski graf

Izabiremo jedan čvor grafa, i njega inicijalno postavljamo u traženo razapinjujuće stablo. Neka je to čvor 1. Sada tražimo brid koji ima najmanju težinu, a da mu je jedan čvor u skupu $G \setminus T$, a drugi u skupu $T = \{1\}$. Bridovi koji su kandidati su: $\{1, 2\}$, $\{1, 3\}$ i $\{1, 11\}$. Budući da brid $\{1, 3\}$ ima najmanju težinu, njega dodajemo.



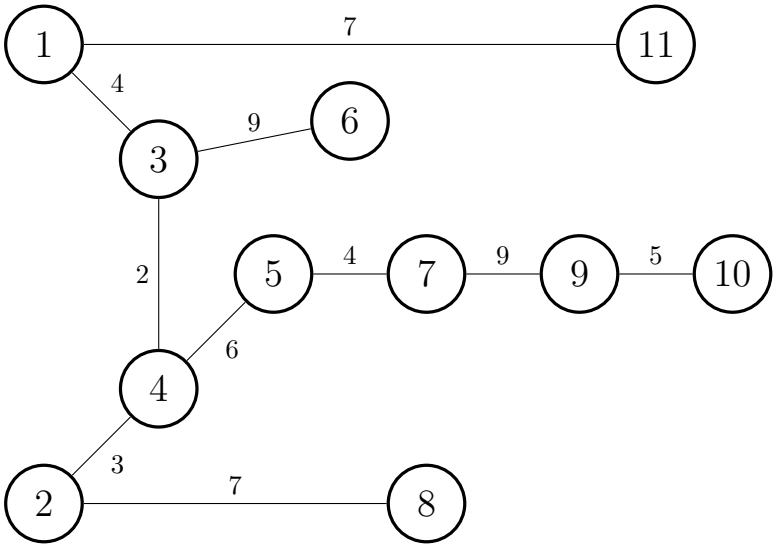
Slika 2.3: Početak primjene algoritma

Sada promatramo bridove $\{1,2\}$, $\{1,11\}$, $\{3,4\}$, $\{3,5\}$ i $\{3,6\}$. Brid najmanje težine je brid $\{3,4\}$, pa njega dodajemo.



Slika 2.4: Nakon drugog koraka

Uzastopnim dodavanjem čvora dobivamo sljedeće minimalno razapinjajuće stablo:



Slika 2.5: Minimalno razapinjujuće stablo

Vremenske složenost Primovog algoritma je $O(n^2)$, gdje je n broj čvorova grafa.

Napomena 2.43. *Označimo s* `PRIMES` *problem ispitivanje primarnosti prirodnog broja. Indijski matematatičari Agrawal, Kayal i Saxena su 2004. godine u časopisu Annals of Mathematics objavili članak u kojem su dokazali da vrijedi* $\texttt{PRIMES} \in \texttt{P}$.

Sada ćemo u grubim crtama navesti korake Agrawal, Kayal, Saxenovog algoritma. Za svaki $r \in \mathbb{N}$ *i svaki* $n \in \mathbb{Z}$ *za koje je* $nzd(n, r) = 1$, *sa* $o_r(n)$ *označavamo najmanji prirodan broj* k *koji ima svojstvo* $n^k \equiv 1 \ (mod\ r)$. *Sa* Φ *označavmo Eulerovu funkciju koja svakom prirodnom broju pridružuje broj svih prirodnih brojeva koji su manji od* n, *te su relativno prosti sa* n.

Agrawal, Kayal, Saxenov algoritam:

1. *Ako je* $n = a^b$ *za neki* $a \in \mathbb{N}$ *i* $b > 1$, *tada algoritam završava s porukom "SLOŽEN";*

2. *Nađi najmanji* r *takav da je* $o_r(n) > \log_2 n$;

3. *Ako je* $1 < nzd(a, n) < n$ *za neki* $a \leq r$, *tada algoritam završava s porukom "SLOŽEN";*

4. *Ako je* $n \leq r$, *tada algoritam završava s porukom "PRIM";*

5. *Za* $a = 1$ *do* $\lceil \sqrt{\Phi(r)} \log n \rceil$ *napravi:*
   ako $((X + a)^n \not\equiv X^n + a \ (mod\ X^r - 1, n))$, *tada algoritam završava s porukom "SLOŽEN";*

6. *Algoritam završava s porukom "PRIM".*

Još neke detalje o algoritmima za ispitivanje primarnosti možete naći u knjizi [5].

Definicija 2.44. *Označimo sa* EXPTIME *klasu svih jezika koji su odlučivi na nekom Turingovom stroju vremenske složenosti* $O(2^{n^k})$, *za neki* $k \in \mathbb{N}$. *Dakle, vrijedi:*

$$\mathtt{EXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k}).$$

Iz teorema o vremenskoj hijerarhiji, tj. iz teorema 2.28., slijedi tvrdnja sljedećeg korolara.

Korolar 2.45. *Vrijedi* $\mathtt{P} \subsetneq \mathtt{EXPTIME}$.

Dokaz. Očito vrijedi $\mathtt{P} \subseteq \mathtt{EXPTIME}$. Kako bi dokazali da vrijedi prava inkluzija, pokažimo da za svaki $k \in \mathbb{N}$ vrijedi $\mathrm{DTIME}(n^k) \subsetneq \mathrm{DTIME}(2^n)$. Iz zadatka 6 znamo da je funkcija $n \mapsto 2^n$ vremenski konstruktibilna. Iz teorema o vremenskoj hijerahiji, tj. teorema 2.28., slijedi da postoji odlučiv jezik $L \in \mathrm{DTIME}(2^n)$ tako da jezik L nije odlučiv niti na jednom Turingov stroju T za koji vrijedi $time_T(n) \sim o(2^n / \log_2 2^n)$. Budući da za svaki $k \in \mathbb{N}$ očito vrijedi $n^k \sim o(2^n/n)$ tada $L \notin \mathtt{PTIME}$. Q.E.D.

Zadaci

1. Dokažite da je klasa P zatvorena za konačne unije.
   Rješenje. Neka su $L_1,\ L_2 \in \mathtt{P}$. Neka je $T_i = (k_i, Q_i, \Gamma, \delta_i)$ neki k_i–tračni Turingov stroj koji odlučuje jezik L_i u vremenu $f_i(n)$, gdje je f_i neki polinom, za $i = 1, 2$. Definiramo Turingov stroj $T = (k_1 + k_2 + 1, Q, \Gamma, \delta)$ koji odlučuje jezik $L_1 \cup L_2$ tako da je $Q = Q_1 \cup Q_2$ (pretpostavljamo da je $Q_1 \cap Q_2 = \emptyset$), a funkciju prijelaza δ posebno opisujemo. Stroj T ima jednu ulaznu traku (važno je da se ulazni podatak ne mijenja na toj traci). Stroj T prvo kopira ulaznu riječ s ulazne trake na prvu radnu traku. Tada stroj T "simulira" rad stroja T_1 koristeći prvih k_1 radnih traka. Ako nakon toga "simulacija" stroja T_1 završi u stanju q_{DA} tada definiramo da stroj T također završi u stanju q_{DA}. Ako "simulacija" stroja T_1 završi u stanju q_{NE} tada stroj T kopira riječ s ulazne trake na $(k_1 + 1)$. radnu traku. Nakon toga stroj T pokrene "simulaciju" stroja T_2. Ako "simulacija" stroja T_2 završi u stanju q_{DA} tada definiramo da stroj T također završi u stanju q_{DA}. Ako "simulacija " stroja T_2 završi u stanju q_{NE} tada definiramo da stroj T također završi u stanju q_{NE}. Očito stroj T odlučuje jezik $L_1 \cup L_2$. Uočimo još da vrijedi: $time_T(n) \leq 2n + f_1(n) + f_2(n)$, pa je stroj T polinomne složenosti.

2. Dokažite da je klasa P zatvorena za komplemente.
   Uputa. Ako stroj T odlučuje neki jezik $L \subseteq \Gamma^*$ u vremenu $f(n)$, onda stroj $\overline{T}$ koji je jednak stroju T osim što je $q^{\overline{T}}_{DA} = q^T_{NE}$ i $q^{\overline{T}}_{NE} = q^T_{DA}$ odlučuje jezik $\Gamma^* \setminus L$ u istom vremenu $f(n)$.

3. Dokažite da je klasa P zatvorena za konačne presjeke.

4. Dokažite da je klasa P zatvorena za simetrične razlike.

5. Dokažite da je klasa P zatvorena za konkatenacije.
   Neka su $L_1,\ L_2 \in$ P proizvoljni. Neka su T_1 i T_2 deterministički Turingovi strojevi polinomne vremenske složenosti takvi da je $L(T_1) = L_1$ i $L(T_2) = L_2$. Neka je vremenska složenost stroja T_1 jednaka $O(n^k)$, a stroja T_2 neka je $O(n^l)$. Definirat ćemo deterministički Turingov stroj polinomne vremenske složenosti koji odlučuje jezik $L_1 * L_2$.
   Stroj T za svaku ulaznu riječ w na prvoj traci razmatra sve podriječi α i β za koje vrijede $w = \alpha * \beta$. Za svaki takav rastav pobriše prvo sve na drugoj i trećoj traci, te napiše riječ α na drugu traku i riječ β na treću traku. Tada se simulira rad stroja T_1 s ulazom α, te rad stroja T_2 s ulazom β. Ako za neki par (α, β) oba stroja prihvate ulaze, tada definiramo da i stroj T prihvaća ulaz. Inače definiramo da stroj T odbija ulaz. Očito stroj T odlučuje jezik $L_1 * L_2$.

   Dokažimo još da je vremenska složenost stroja T polinomna. Za svaki par (α, β) složenost rada stroja T je jednaka: $|w| + time_{T_1}(|\alpha|) + time_{T_2}(|\beta|)$. Tada ukupan broj koraka rada stroja T s riječi w na ulazu možemo ocijeniti sa sljedećim

nejednakostima:

$$\begin{aligned}
\sum_{m=0}^{|w|}\Big(|w| \;+\; & time_{T_1}(m) + time_{T_2}(|w|-m)\Big) \\
&= \sum_{m=0}^{|w|}|w| \;+\; \sum_{m=0}^{|w|} time_{T_1}(m) \;+\; \sum_{m=0}^{|w|} time_{T_2}(|w|-m) \\
&= |w|\cdot(|w|+1) \;+\; \sum_{m=0}^{|w|} time_{T_1}(m) \;+\; \sum_{m=0}^{|w|} time_{T_2}(|w|-m) \\
&\leq K_0|w|^2 \;+\; \sum_{m=0}^{|w|} c_m m^k \;+\; \sum_{m=0}^{|w|} s_m(|w|-m)^l \\
&\leq K_0|w|^2 \;+\; K_1|w|^k \;+\; K_2|w|^l,
\end{aligned}$$

gdje su $K_0, c_0, \ldots, c_{|w|},\ s_0, \ldots, s_{|w|}$, te K_1 i K_2 neki pozitivni realni brojevi. Iz toga slijedi da je vremenska složenost stroja T polinomna.

6. Neki pojmovi vezani uz grafove su definirani već u definicijama 2.40. i 2.42. Za podgraf (G', R') grafa (G, R) kažemo da je **klika** ako je $R' = G' \times G'$. Za kliku (G', R') grafa (G, R) kažemo da je k**–klika**, ako sadrži točno k čvorova, gdje je $k \geq 2$. `PROBLEM KLIKA` sastoji se od određivanja da li zadani graf sadrži kliku zadane veličine, tj. formalno:

$$\texttt{PROBLEM KLIKA} = \{\langle G, k\rangle \;:\; G \text{ je graf koji sadrži neku } k\text{–kliku}\}.$$

`PROBLEM KLIKA` ćemo razmatrati kasnije (vidi stranu 61). Za svaki $k \geq 2$ može se promatrati i sljedeći problem:

$$k\text{–}\texttt{KLIKA} = \{G \;:\; G \text{ je graf koji sadrži } k\text{–kliku}\}$$

Dokažite da za svaki $k \geq 2$ vrijedi k–`KLIKA` $\in$ `P`.
Rješenje. Neka je $k \geq 2$ zadan, te neka je G neki graf sa n vrhova. Označimo sa A algoritam koji kao ulazni podatak prima graf G te radi sljedeće:

1. Za svaki k člani podskup S od G :

   provjeri čine li vrhovi iz S jednu k–kliku u grafu G. Ako je S jedna k–klika, tada algoritam staje u stanju q_{DA}, a inače se bira novi k–člani podskup od G.

2. Ako su ispitani svi k–člani podskupovi od G, te niti jedan od njih nije k–klika od G, tada algoritam staje u stanju q_{NE}.

Očito algoritam A ispituje sadrži li graf G neku k–kliku. Odredimo vremensku složenost algoritma A. Prvi korak algoritma se izvodi $\binom{n}{k}$ puta, tj. njegova vremenska složenost je $O(n^k)$. Drugi korak se može provesti u vremenu $O(k^2 n)$

(za svaki od vrhova izabranog k članog podskupa S provjerava se je li povezan s ostalim odabranim bridovima; svaki vrh grafa G je incidentan s najviše $n-1$ bridova). To znači da je vremenska složenost algoritma A jednaka: $O(n^k) \cdot O(k^2 n)$, tj. $O(n^{k+1})$.

7. Dokažite da problem povezanosti grafa pripada klasi P.

2.5.2 Klasa NPTIME

Nažalost, za mnoge važne probleme nije poznat polinoman algoritam. No, mnogi od tih problema ipak imaju jedno zajedničko svojstvo: odlučivi su na nekom nedeterminističkom Turingovom stroju polinomne vremenske složenosti. U ovoj točki razmatramo upravo tu klasu problema.

Definicija 2.46. *Sa NPTIME, ili samo kratko s NP, označavamo klasu svih jezika koji su odlučivi na nekom nedeterminističkom Turingovom stroju vremenske složenosti $O(n^k)$, za neki $k \in \mathbb{N}$. Dakle, vrijedi:*

$$NP = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k).$$

Bili smo već istaknuli da za svaku funkciju $f : \mathbb{N} \to \mathbb{R}^+$ vrijedi $\mathrm{DTIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$. Iz toga očito slijedi tvrdnja sljedećeg korolara.

Korolar 2.47. *Vrijedi $P \subseteq NP$.*

Sljedeći teorem jako dobro opisuje klasu NP: za svaki problem iz te klase lako je provjeriti je li nešto njegovo rješenje.

Teorem 2.48. (O certifikatu)
Za svaki jezik L vrijedi:

$$\begin{aligned} L \in NP \Leftrightarrow \quad & (\exists R \in P)(\exists\ k \in \mathbb{N}) \\ & L = \{x : (\exists c)(|c| = O(|x|^k) \wedge R(x,c))\}. \end{aligned}$$

Riječ c nazivamo **certifikat** *za riječ x, a jezik koji sadrži sve certifikate nazivamo certifikat za jezik L.*

Dokaz. Dokažimo prvo implikaciju $\Leftarrow$. Konstruiramo nedeterministički Turingov stroj ovako: za ulaz x prvo "pogađamo" certifikat. U prvom koraku stroj se razgrana na sve moguće riječi duljine jedan. U drugom koraku se razgrana na sve moguće riječi duljine dva. Itd. ... U $|x|^k$ koraka će biti generirane sve riječi zadanog alfabeta čija je duljina najviše $|x|^k$. Po pretpostavci za $x \in L$ postoji certifikat čija je duljina najviše $|x|^k$. Zatim, po pretpostavci u polinomnom vremenu može se ispitati vrijedi li $R(x,c)$.

Dokažimo sada obratnu implikaciju. Pretpostavimo da postoji nedeterministički Turingov stroj T koji odlučuje jezik L u polinomnom vremenu. Traženu relaciju R definiramo ovako:

$(x, c) \in R$ ako i samo ako riječ c predstavlja put u stablu računanja stroja T, ako stroj T prihvaća riječ x.

Po pretpostavci stroj T odlučuje riječ x u polinomnom vremenu, pa posebno vrijedi $|c| = O(|x|^k)$. Lako je vidjeti da vrijedi $R \in \mathtt{P}$. Q.E.D.

Primjer 2.49. *Iz prethodnog teorema slijedi da je za dokaz da za neki jezik L vrijedi $L \in$ NP, dovoljno naći certifikat za svaku riječ jezika L. Označimo* SLOŽEN$=\{n \in \mathbb{N} : n$ *nije prim broj*$\}$. *Očito je certfikat za svaki složeni prirodan broj jedan njegov djeljitelj. Ako je zadan prirodan broj n, te otkrijemo jedan djeljitelj d od n, tada je vrlo lako provjeriti dijeli li stvarno d broj n. Dakle,* SLOŽEN $\in$ NP.

Kao što smo već bili naveli još u uvodu ovog materijala, problem odnosa klasa P i NP, tj. problem "P vs NP", je za sada neriješeni problem. Problem je star više od 40 godina. Istaknimo neka razmišljanja o problemu "P vs NP" imajući na umu karakterizaciju klase NP pomoću certifikata.

a) lakše je utvrditi je li neki odgovor točan, nego dati odgovor;

b) daleko lakše je provjeriti je li dokaz nekog teorema točan, nego dokazati teorem;

c) lakše je provjeriti je li je neki put u grafu potpun, nego utvrditi postoji li potpun put.

Primjer 2.50. *Sada navodimo neke primjere problema koji pripadaju klasi* NP.

- *problem ispunjivosti formula logike sudova, tj. problem* SAT *(tom problemu je posvećeno sljedeće poglavlje);*
- *problem faktorizacije prirodnih brojeva;*
- *problem cjelobrojnog linearnog programiranja;*
- *određivanje je li je graf 3–obojiv (taj problem ćemo razmatrati kasnije);*
- *problem egzistencije Hamiltonovog puta u grafu (Hamiltonov put je onaj koji sadrži sve čvorove u grafu, ali samo jednom; taj problem ćemo također razmatrati kasnije);*
- *problem egzistencije nezavisnog skupa u grafu (nezavisan skup je potpuno nepovezan podgraf; taj problem ćemo također razmatrati kasnije);*
- *problem egzistencije klike u grafu (klika je potpuno povezan podgraf; taj problem ćemo također razmatrati kasnije);*

- *problem trgovačkog putnika, tj.* `PROBLEM TSP` *(eng. traveling salesman problem; vidi zadatak 6.4 na strani 152);*
- *Minesweeper*
  (vidi V. Kojić, Minesweeper problem je `NP`*–potpun, math.e, 12 (2008); http://e.math.hr)*
- *problemi u vezi igara: potapanje brodova, tetris, gomilice, Sokoban.*

U sljedećem primjeru želimo istaknuti da se razmatra i složenost problema koji su definirani nad $\mathbb{R}$ i $\mathbb{C}$ (za više detalja vidite [3]).

Primjer 2.51. (Hilbertov Nullstellensatz)
Neka su $f_1, \ldots, f_k \in \mathbb{C}[X_1, \ldots, X_n]$ proizvoljni polinomi nad poljem $\mathbb{C}$. Koji su nužni i dovoljni uvjeti da ti polinomi imaju zajedničku nul–točku? Odgovor na to pitanje daje Hilbertov Nullstellensatz:

> *Skup polinoma $\{f_1, \ldots, f_k\}$ ima zajedničku nul–točku ako i samo ako ne postoje polinomi $g_1, \ldots, g_k \in \mathbb{C}[X_1, \ldots, X_k]$ tako da vrijedi $\sum_{i=1}^{k} g_i \cdot f_i = 1$.*

Odnosno, postoji nul–točka konačnog skupa polinoma $f_1, \ldots, f_k$ ako i samo ako je ideal $I = I(f_1, \ldots, f_k)$ pravi ideal. (Ako postoje polinomi $g_1, \ldots, g_k$ takvi da je $\sum g_i \cdot f_i = 1$, tada imamo $1 \in I$, jer iz definicije ideala vrijedi $I \cdot \mathbb{C}[X_1, \ldots, X_n] \subseteq I$. To znači da je tada $I = \mathbb{C}[X_1, \ldots, X_k]$), tj. ideal I nije pravi ideal.)

Uočite da baš nema smisla pričati o Turingovim strojevima koji operiraju s kompleksnim brojevima! Definiraju se posebni strojevi koji operiraju s kompleksnim brojevima. Postavljena je hipoteza da ne postoji polinomni algoritam koji odlučuje Hilbertov Nullstellensatz nad $\mathbb{C}$.

Zadaci

1. Dokažite da je klasa `NP` zatvorena za konačne unije i konkatenacije.
2. Dokažite da postoji odlučiv jezik L tako da vrijedi $L \notin$ `NP`.
   Rješenje. Neka je Γ proizvoljan konačan alfabet. Definiramo jezik L ovako:

   $$L = \{\langle T\rangle : T \text{ je jednotračni deterministički Turingov stroj nad alfabetom } \Gamma \text{ koji ne prihvaća riječ } \langle T\rangle \text{ u najviše } 2^{2^{|\langle T\rangle|}} \text{ koraka}\}.$$

   Iz teorema 2.23. slijedi da postoji univerzalni Turingov stroj nad alfabetom Γ. Funkcija $n \mapsto 2^{2^n}$ je potpuno vremenski konstruktibilna (vidi zadatak 6 na strani 41). Modifikacijom stroja U dodavanjem "štoperice" koja broji najviše 2^{2^n} koraka, dobivamo stroj koji odlučuje jezik L. To znači da je jezik L odlučiv.

Pretpostavimo da vrijedi $L \in \texttt{NP}$. Neka je N neki nedeterminsitički Turingov stroj vremenske složenosti n^k, za neki $k \in \mathbb{N}$, koji odlučuje jezik L. Iz teorema 2.18. o redukciji nedeterminističkog Turingovog stroja na deterministički slijedi da postoji deterministički Turingov stroj M koji odlučuje jezik L i vremenske složenosti je $2^{O(n^k)}$. Tada postoji $c > 0$ i $n_0 \in \mathbb{N}$ tako da za svaki $n \geq n_0$ vrijedi $time_M(n) \leq 2^{cn^k}$. Očito postoji $n_1 \geq n_0$ tako da za svaki $n \geq n_1$ vrijedi $cn^k \leq 2^n$. Tada imamo da za svaki $n \geq n_1$ vrijedi: $time_M(n) \leq 2^{2^n}$.

Bez smanjenja općenitosti možemo pretpostaviti da je $|\langle M \rangle| \geq n_1$, jer inače možemo skupu stanja stroja M dodati nova stanja koja se uopće ne koriste, pa se time dovoljno poveća duljina koda stroja M, ali se vremenska složenost stroja M ne mijenja. Tada imamo:

$$\begin{aligned}
\langle M \rangle \in L &\Leftrightarrow \langle M \rangle \in L(M) \\
&\Leftrightarrow \text{stroj } M \text{ prihvaća riječ } \langle M \rangle \text{ u vremenu } time_M(|\langle M \rangle|) \\
&\Leftrightarrow \text{stroj } M \text{ prihvaća riječ } \langle M \rangle \text{ u najviše } 2^{2^{|\langle M \rangle|}} \text{ koraka} \\
&\Leftrightarrow M \notin L
\end{aligned}$$

Time je dobivena kontradikcija.

Poglavlje 3

NP–potpunost

U ovom poglavlju posebno ćemo razmatrati problem ispunjivosti formula logike sudova, tj. problem SAT (eng. satisfiability). Prvo ćemo ponoviti definicije nekih pojmova, te istaknuti neke činjenice. Centralni dio ovog poglavlja je Cook–Levinov teorem koji govori da je svaki NP–problem moguće svesti na problem SAT. Na kraju ćemo razmotriti još neke varijante problema SAT, te opisati Davis–Putnamov algoritam za rješavanje problema SAT.

3.1 Osnovne definicije

Prisjetimo se prvo nekih pojmova iz logike sudova, te istaknimo neke činjenice koje ćemo kasnije trebati.

Definicija 3.1. *Alfabet logike sudova je unija skupova* A_1, A_2 *i* A_3, *pri čemu je:*

$$A_1 = \{P_0,\ P_1,\ P_2,\ \ldots\} \text{ prebrojiv skup čije elemente nazivamo propozicionalne varijable,}$$

$$A_2 = \{\neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow\} \text{ skup logičkih veznika,}$$

$$A_3 = \{(,\ )\} \text{ skup pomoćnih simbola (zagrade).}$$

Definicija 3.2. *Atomarna formula je svaka propozicionalna varijabla. Pojam formule logike sudova definiramo rekurzivno:*

a) svaka atomarna formula je formula;

b) ako su A *i* B *formule tada su i riječi* $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ *i* $(A \leftrightarrow B)$ *također formule;*

c) riječ alfabeta logike sudova je formula ako je nastala primjenom konačno mnogo koraka uvjeta a) i b).

Definicija 3.3. *Svako preslikavanje sa skupa svih propozicionalnih varijabli u skup* $\{0,1\}$, *tj.* $I : \{P_0, P_1, \ldots\} \to \{0,1\}$, *nazivamo totalna interpretacija ili kratko interpretacija. Ako je preslikavanje definirano na podskupu skupa propozicionalnih varijabli tada kažemo da je to parcijalna interpretacija.*

Definicija 3.4. *Neka je* I *interpretacija (totalna ili parcijalna). Tada vrijednost interpretacije* I *na proizvoljnoj formuli definiramo rekurzivno ovako:*

$$\begin{aligned} I(\neg A) = 1 \quad & \textit{ako i samo ako} \quad I(A) = 0; \\ I(A \wedge B) = 1 \quad & \textit{ako i samo ako} \quad I(A) = 1 \textit{ i } I(B) = 1; \\ I(A \vee B) = 1 \quad & \textit{ako i samo ako} \quad I(A) = 1 \textit{ ili } I(B) = 1; \\ I(A \to B) = 1 \quad & \textit{ako i samo ako} \quad I(A) = 0 \textit{ ili } I(B) = 1; \\ I(A \leftrightarrow B) = 1 \quad & \textit{ako i samo ako} \quad I(A) = I(B). \end{aligned}$$

Preglednije je vrijednost interpretacije na formulama definirati pomoću tablica koje se nazivaju semantičke tablice. Tada se vrijednosti interpretacije za složenije formule mogu definirati kao u sljedećoj tablici.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \to Q$	$P \leftrightarrow Q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Tablica 3.1: Vrijednosti interpretacije za složenije formule.

Definicija 3.5. *Za formulu* F *logike sudova kažemo da je* **ispunjiva** *ako postoji interpretacija* I *tako da vrijedi* $I(F) = 1$.

Kažemo da su formule A *i* B *logički ekvivalente, i pišemo* $A \Leftrightarrow B$, *ako za svaku interpretaciju* I *vrijedi* $I(A) = I(B)$.

Definicija 3.6. *Atomarnu formulu i njezinu negaciju nazivamo* **literal**. *Formulu oblika* $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ *nazivamo konjunkcija, a formulu oblika* $A_1 \vee A_2 \vee \ldots \vee A_n$ *nazivamo disjunkcija, gdje su* A_i *proizvoljne formule.*

Elementarna konjunkcija je konjunkcija literala, a elementarna disjunkcija je disjunkcija literala. **Konjunktivna normalna forma**, *ili kratko* **knf**, *je konjunkcija elementarnih disjunkcija.* **Disjunktivna normalna forma**, *ili kratko* **dnf**, *je disjunkcija elementarnih konjunkcija.*

Teorem o normalnim formama u logici sudova. Za svaku formulu F logike sudova postoji konjunktivna normalna forma A i disjunktivna normalna forma B tako da vrijedi $F \Leftrightarrow A$ i $F \Leftrightarrow B$.

Sve detalje o logici sudova, kao i dokaz prethodnog teorema, možete vidjeti u [25].

Sada navodimo jedan algoritam za određivanje konjunktivne normalne forme formule, te ističemo njegovu vremensku složenost.

Algoritam za određivanje konjunktivne normalne forme zadane formule F:

1. Primjenom ekvivalencija $(A \to B) \Leftrightarrow (\neg A \vee B)$ i $(A \leftrightarrow B) \Leftrightarrow ((\neg A \vee B) \wedge (\neg B \vee A)$ eliminiraju se logički veznici $\to$ i $\leftrightarrow$ iz formule.

2. Primjenom de Morganovih zakona i pravila dvojne negacije dobiva se formula u kojoj je svaka negacija uz neku propozicionalnu varijablu.

3. Primjenom distributivnosti veznika $\vee$ prema $\wedge$, tj. primjenom ekvivalencije $(A \vee (B \wedge C)) \Leftrightarrow ((A \vee B) \wedge (A \vee C))$, dobiva se konjunktivna normalna forma koja je logički ekvivalentna formuli F.

Navedeni algoritam je eksponencijalne vremenske složenosti jer primjenom pravila za distributivnost duljina potformula se skoro dva puta poveća.

Želimo istaknuti da postoji i polinomni algoritam za određivanje knf (dobivena knf F' nije nužno logički ekvivalentna početnoj formuli F, ali je ispunjivost fromule F ekvivalentna ispunjivosti formule F'). Neke detalje možete vidjeti u zadatku 3, a svi detalji su navedeni u [20] i [21].

Problem SAT (eng. satisfiability) glasi: *odrediti je li zadana knf ispunjiva.* Odnosno, točnije:

$$\texttt{SAT} = \{F : F \text{ je ispunjiva knf}\}.$$

Semantičke tablice su npr. jedan algoritam za ispitivanje ispunjivosti formule. No, znamo ako formula sadrži n različitih propozicionalnih varijabli, tada semantička tablica sadrži 2^n redaka. To znači da semantičke tablice nisu polinomni algoritam, već eksponencijalni. Ako bi računalo za svaki redak semantičke tablice potrošilo jednu mikrosekundu trebalo bi mu više od godinu dana da napravi semantičku tablicu za formulu koja ima 50 varijabli. U točkama koje slijede istaknut ćemo važnost problema SAT, te navesti neke varijante tog problema.

Knf koja u svakoj svojoj elementarnoj disjunkciji sadrži točno k literala (za neki $k \in \mathbb{N} \setminus \{0\}$), nazivamo **$k$–knf**. Problem k–SAT se sastoji od ispitivanja ispunjivosti k–knf, odnosno točnije:

$$\textbf{k–SAT} = \{F : F \text{ je ispunjiva } k\text{–knf}\}.$$

3.2 Polinomna reducibilnost

Neka su Γ_1 i Γ_2 proizvoljni alfabeti.

Definicija 3.7. *Kažemo da je neka* **funkcija** $f : \Gamma_1^* \to \Gamma_2^*$ **vremenski polinomno izračunljiva** *ako postoji polinomno vremenski složen Turingov stroj koji za svaku riječ $w \in \Gamma_1^*$ kao ulazni podatak na traku ispisuje $f(w)$.*

Definicija 3.8. *Kažemo da je* **jezik** $L_1 \subseteq \Gamma_1^*$ **polinomno reducibilan na jezik** $L_2 \subseteq \Gamma_2^*$, *ako postoji vremenski polinomno izračunljiva funkcija $f : \Gamma_1^* \to \Gamma_2^*$ takva da za svaki $w \in \Gamma_1^*$ vrijedi:*

$$w \in L_1 \qquad \textit{ako i samo ako} \qquad f(w) \in L_2.$$

Ako je jezik L_1 polinomno reducibilan na jezik L_2 tada to označavamo sa $L_1 \leq_p L_2$.

Propozicija 3.9. *Ako $L_1 \leq_p L_2$ i $L_2 \leq_p L_3$ tada vrijedi $L_1 \leq_p L_3$.*

Propozicija 3.10. *Ako $L_1 \in$ P i $\emptyset \neq L_2 \neq \Gamma^*$ tada $L_1 \leq_p L_2$.*

Propozicija 3.11. *Ako $L_2 \in$ P (odnosno $L_2 \in$ NP) i $L_1 \leq_p L_2$ tada $L_1 \in$ P (odnosno $L_1 \in$ NP).*

Dokaz. Neka je $L_2 \in$ NP. Tada postoji $k \in \mathbb{N}$ i nedeterministički Turingov stroj N vremenske složenosti n^k koji odlučuje jezik L_2. Budući da je jezik L_1 polinomno reducibilan na jezik L_2 tada postoji vremenski polinomno izračunljiva funkcija $f : {\Gamma_1}^* \to \Gamma_2^*$ tako da za svaku riječ $w \in {\Gamma_1}^*$ vrijedi: $w \in L_1$ ako i samo ako $f(w) \in L_2$. Budući da je f vremenski polinomno izračunljiva tada postoji Turingov stroj T vremenske složenosti n^m, za neki $m \in \mathbb{N}$, koji za ulaz $w \in \Gamma_1^*$ na traku ispisuje riječ $f(w)$, gdje je $n = |w|$.

Definiramo nedeterministički Turingov stroj S ovako:

a) prvo se simulira rad stroja T koji za ulaz $w \in \Gamma_1^*$ na traku ispiše riječ $f(w)$;

b) zatim se simulira rad stroja N;

c) definiramo da stroj S prihvaća riječ w ako i samo ako stroj N prihvaća riječ $f(w)$, odnosno stroj S odbija riječ w ako i samo ako stroj N odbija riječ $f(w)$.

Očito stroj S odlučuje jezik L_1, tj. vrijedi $L(S) = L_1$. Primijetimo da je stroj S polinomne složenosti, jer su strojevi N i T polinomne vremenske složenosti, tj. točnije: stroj S je vremenske složenosti $O(n^p)$, gdje je $p = \max\{k, m\}$. Q.E.D.

Definicija 3.12. *Kažemo da je neki jezik L* **NP–potpun** *ako zadovoljava sljedeća dva uvjeta:*

a) jezik L pripada klasi `NP`*;*

b) svaki jezik $L' \in$ `NP` *je polinomno reducibilan na jezik L.*

Propozicija 3.13. *Neka je L_1 neki* `NP`*–potpun jezik i $L_2 \in$* `NP`*. Ako $L_1 \leq_p L_2$ tada je i jezik L_2* `NP`*–potpun.*

Sada dajemo dva primjera u kojima pokazujemo polinomnu reducibilnost problema 3–`SAT` na jedan problem s grafovima, te jednog problema s grafovima na problem `SAT`. Neke pojmove vezane uz grafove već smo bili definirali u definicijama 2.40. i 2.42. na strani 46. `PROBLEM KLIKA` definirali smo u zadatku 6 na strani 52.

Teorem 3.14. *Problem 3–*`SAT` *je polinomno reducibilan na* `PROBLEM KLIKA`*.*

Dokaz. Neka je F neka 3–knf koja sadrži k elementarnih disjunkcija (a onda sadrži točno $3k$ literala). Za formulu F kontruiramo graf (G, R) ovako: skup čvorova sadrži sve literale od F (više čvorava može biti označeno istim literalom), a svi čvorovi su povezani bridom osim ova dva izuzetka:

a) nikoja dva literala koja pripadaju nekoj istoj elementarnoj disjunkciji od F nisu povezana bridom;

b) nikoja dva komplemantarna literala, tj. literali oblika P i $\neg P$, nisu povezani bridom.

Tvrdimo da vrijedi: $F \in$ `SAT` ako i samo ako graf G sadrži k–kliku.
Pretpostavimo prvo da je knf F ispunjiva. Neka je I neka interpretacija za koju vrijedi $I(F) = 1$. Tada u svakoj elementarnoj disjunkciji od F postoji literal koji je istinit za interpretaciju I. U svakoj elementarnoj disjunkciji izaberemo točno jedan literal koji je istinit za interpretaciju I. Očito da svi ti istiniti literali tvore jednu k–kliku u grafu G (svi su međusobno povezani bridovima jer ne pripadaju istoj elementarnoj disjunkciji, te ne mogu biti suprotni literali budući da su istiniti za istu interpretaciju).

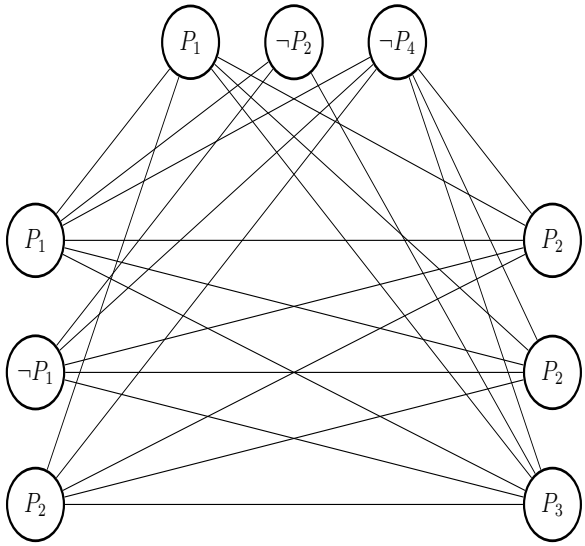
Pretpostavimo sada da graf G sadrži neku k–kliku G'. Definiramo parcijalnu interpretaciju $I : Var(F) \to \{0, 1\}$ ovako:

$$I(P_i) = \begin{cases} 1, & \text{ako je } \ P_i \in G'; \\ 0, & \text{inače.} \end{cases}$$

Budući da po pretpostavci formula F sadrži k elementarnih disjunkcija, te je G' jedna k–klika, tada u svakoj elementarnoj disjunkciji postoji literal L tako da vrijedi $I(L) = 1$. Iz toga očito slijedi $I(F) = 1$, tj. formula F je ispunjiva. Q.E.D.

Za ilustraciju dajemo graf koji je pridružen sljedećoj formuli:

$$(P_1 \vee \neg P_1 \vee P_2) \wedge (P_1 \vee \neg P_2 \vee \neg P_4) \wedge (P_2 \vee P_2 \vee P_3).$$



Slika 3.1: Primjer grafa koji je pridružen formuli

Definicija 3.15. *Kažemo da je graf* (G, R) k**–obojiv** $(k \in \mathbb{N} \setminus \{0\})$ *ako postoji particija* $B_1, \ldots, B_k$ *(dopuštamo da su neki skupovi* B_i *prazni) skupa* G *tako da ne postoje* $a, b \in G$, *te* $j \in \{1, \ldots, k\}$ *za koje vrijedi* $\{a, b\} \in R$ *i* $a, b \in B_j$. `PROBLEM` k`-OBOJIVOSTI` *sastoji se od određivanja je li zadani graf* k*–obojiv.*

U sljedećem poglavlju navest ćemo da se problem 3–`SAT` može polinomno reducirati na `PROBLEM 3-OBOJIVOSTI`. U sljedećem primjeru pokazujemo kako se `PROBLEM 3-OBOJIVOSTI` za graf sa 4 čvora može svesti na problem 3–`SAT`. Primjer bi trebao poslužiti za lakše razumijevanje dokaza Cook–Levinovog teorema koji dajemo odmah nakon primjera.

Primjer 3.16. *Neka je* $(\{1, 2, 3, 4\}, R)$ *proizvoljan graf. Definirat ćemo knf* F *tako da vrijedi: graf* G *je 3–obojiv ako i samo ako formula* F *je ispunjiva. U tu svrhu promatramo sljedeći skup propozicionalnih varijabli:*

$$\{P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}, P_{31}, P_{32}, P_{33}, P_{41}, P_{42}, P_{43}\}.$$

Intutivno značenje je sljedeće: ako je I *interpretacija takva da za neku varijablu* P_{jk} *vrijedi* $I(P_{jk}) = 1$, *tada je čvor* j *obojan* k*–tom bojom. Sada redom definiramo elementarne disjunkcije koje će graditi traženu knf* F. *Za svaki* $i = 1, 2, 3, 4$ *neka je* $C_i \equiv P_{i1} \vee P_{i2} \vee P_{i3}$. *(Intuitivno značenje formule* C_i, *tj. činjenice* $I(C_i) = 1$, *je da je čvor* i *obojan barem jednom od tri boje.) Zatim, za svaki* $i = 1, 2, 3, 4$ *definiramo:* $T_i \equiv \neg P_{i1} \vee \neg P_{i2}$, $U_i \equiv \neg P_{i1} \vee \neg P_{i3}$, $V_i \equiv \neg P_{i2} \vee \neg P_{i3}$. *Neka je formula* F_1 *definirana sa:*

$$F_1 \equiv \left(\bigwedge_{i=1}^{4} C_i \right) \wedge \left(\bigwedge_{i=1}^{4} T_i \right) \wedge \left(\bigwedge_{i=1}^{4} U_i \right) \wedge \left(\bigwedge_{i=1}^{4} V_i \right).$$

(Ako za neku interpretaciju I vrijedi $I(F_1) = 1$, tada to intuitivno znači da je svaki čvor grafa obojan s jednom i samo jednom bojom.)

Moramo još definirati formulu koja će izražavati činjenicu da nikoja dva čvora istog brida nisu obojana istom bojom. Iz tog razloga za svaki brid $e = \{u, v\}$ grafa i svaku boju $j \in \{1, 2, 3\}$ definiramo formulu: $D_{ej} \equiv \neg P_{uj} \vee \neg P_{vj}$. Neka je

$$F_2 \equiv \bigwedge_{e \in R,\ j=1,2,3} D_{ej}.$$

Konačno, neka je $F \equiv F_1 \wedge F_2$. Nije teško vidjeti da dobivena formula F ima tražena svojstva. Primijetimo još da se konstrukcija formule F može provesti u polinomnom vremenu u odnosu na zadani početni graf G.

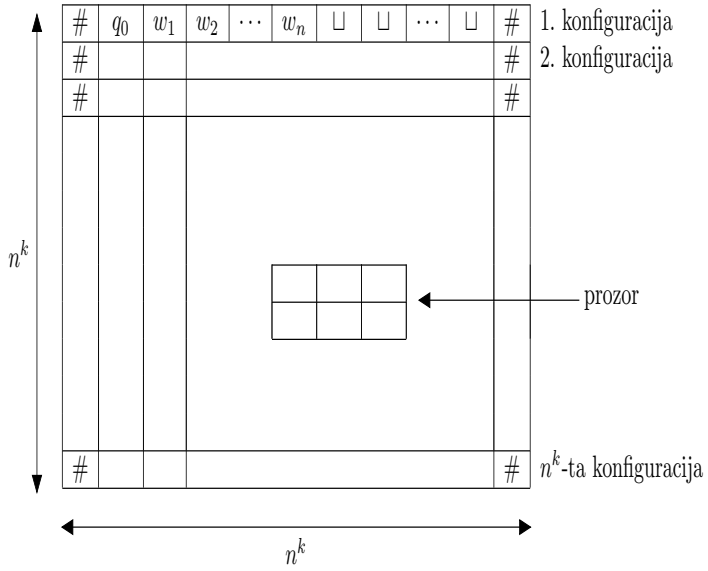
3.3 Cook–Levinov teorem

U ovoj točki dokazujemo teorem za koji se sigurno može reći da je sam temelj teorije složenosti. Teorem jednostavno govori da se svaki NP problem može polinomno reducirati na problem SAT. To znači da kada bi imali neki polinomni algoritam na deterministčkom Turingovom stroju koji bi odlučivao problem SAT, tada bi za svaki NP problem postojao polinomni algoritam na deterministčkom Turingovom stroju.

Teorem 3.17. (S. Cook, L. Levin, 1971.) *Problem SAT je NP–potpun.*

Dokaz. Kako bi dokazali da je SAT $\in$ NP iz teorema o certifikatu slijedi da je za svaku knf F logike sudova dovoljno naći certifikat c, tako da postoji deterministički Turingov stroj T koji u polinomnom vremenu za (F, c) određuje je li F ispunjiva formula. Naravno, za ispunjivu formulu F certifikat c je niz 0 i 1 za koju je $I(F) = 1$. Očito je $|c| \leq |F|$, tj. $|c| = O(|F|)$. Traženi deterministički Turingov stroj ima dvije trake. Na prvoj traci je zapisana formula F, a na drugoj traci je zapisan certifikat. Stroj jednim prolaskom preko formule propozicionalnim varijablama pridružuje vrijednosti 0 ili 1 (čitamo ih s druge trake). Drugim prolaskom stroj provjerava da li se u svakoj elementarnoj disjunkciji nalazi barem jedna jedinica. Za to mu ukupno treba $O(|F|)$ koraka.

Preostalo je dokazati kako se proizvoljan jezik iz klase NP može polinomno reducirati na problem SAT. Neka je $L \in$ NP proizvoljan jezik, te neka je $N = (\Gamma, Q, \delta)$ neki jednotračni nedeterministički Turingov stroj koji odlučuje jezik L u vremenu n^k, za neki $k \in \mathbb{N}$. Budući da stroj N za ulaz duljine n radi najviše n^k koraka, tada na traci može pristupiti najviše do n^k različitih registara. Definirat ćemo formulu Φ logike sudova koja će imati sljedeće svojstvo: Turingov stroj N prihvaća riječ w ako i samo ako formula Φ je ispunjiva. Sada uvodimo notaciju i ilustracije koje bi nam trebale pomoći u definiciji polinomne redukcije. Pretpostavljamo da su registri trake Turingov stroja N označeni cijelim brojevima, te da je na početku rada stroja glava na nultoj ćeliji. Na sljedećoj slici prikazujmo $n^k \times n^k$ tablicu čiji su redovi konfiguracije



Slika 3.2: Tablica koja predstavlja jednu granu izračunavanja

jedne grane stroja N koji na ulazu ima riječ $w = w_1 \dots w_n$. Sa $\sqcup$ označavamo prazno mjesto na traci.

Svaka konfiguracija stroja sastoji se od tri objekta: trenutnog stanja, ukupnog sadržaja trake i lokacije glave. Iz praktičnih razloga svaka konfiguracija počinje i završava znakom $\sharp$, tako da su prvi i zadnji stupac u tablici ispunjeni znakom $\sharp$. Prvi red u tablici je početna konfiguracija stroja N, a svaki sljedeći redak tablice dobiven je u skladu s funkcijom prijelaza δ stroja N. Tablica prihvaća riječ w ako bilo koji redak tablice predstavlja konfiguraciju stroja koja prihvaća w. Svaka tablica za stroj N koja prihvaća riječ w odgovara jednoj grani izračunavanja stroja N koja prihvaća w. Stoga je pitanje da li stroj N prihvaća riječ w ekvivalentno pitanju postoji li tablica za N koja prihvaća riječ w. Neka je $C = Q \cup \Gamma \cup \{\sharp\}$. Za sve $i, j \in \{1, \dots, n^k\}$ i svaki $s \in C$ sa $x_{i,j,s}$ označavamo propozicionalnu varijablu. Formulama logike sudova opisat ćemo rad Turingov stroja N.

Svaki od $(n^k)^2$ dijelova tablice nazivat ćemo ćelija. Sadržaj ćelije u i–tom retku i j–tom stupcu označavamo sa $cel(i, j)$ (to je neki element skupa C). Sadržaj svake ćelije reprezentiramo propoziconalnim varijablama. Ako varijabla $x_{i,j,s}$ poprimi vrijednost 1, tada vrijedi $cel(i, j) = s$. Tražena formula Φ je konjunkcija formula Φ_{cel}, Φ_{start}, Φ_{pomak} i $\Phi_{prihvat}$. Sada redom definiramo svaku formulu. Formula Φ_{cel} izražava da svaka ćelija sadrži samo jedan simbol:

$$\Phi_{cel} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C,\ s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right].$$

Formula Φ_{start} izražava da prvi redak tablice sadrži početnu konfiguraciju stroja N koji na ulazu ima riječ $w = w_1 \dots w_n$:

$$\Phi_{start} = x_{1,1,\sharp} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \dots\ x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\sharp}.$$

Formula $\Phi_{prihvat}$ izražava da u tablici postoji prihvaćajuća konfiguracija, a definirana je formalno ovako:

$$\Phi_{prihvat} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{DA}}.$$

Formula Φ_{pomak} izražava da svaki redak tablice dobivamo primjenom funkcije prijelaza δ. To ćemo osigurati tako da svaka mala tablica dimenzije 2×3 bude u skladu s funkcijom δ. Takve male tablice koje su u skladu s funkcijom δ ćemo nazivati legalni prozor. (Primijetite da je dovoljno i nužno promatrati samo prozore veličine 2×3.) Konfiguracije stroja N ćemo zapisivati u obliku uqv, gdje je q trenutno stanje, a u i v su riječi koje se nalaze na traci tako da je uv trenutno stanje trake, a glava stroja se nalazi na prvom simbolu riječi v.

Navodimo dva primjera legalnih prozora. Neka su $a, b, c \in \Gamma$, $q_1, q_2 \in Q$, te

$$\delta(q_1, a) = \{(q_1, a, D)\} \qquad \text{i} \qquad \delta(q_1, b) = \{(q_2, c, L),\ (q_2, a, D)\}.$$

Navodimo dva primjera legalnih prozora koji su inducirani gore zadanom funkcijom prijelaza.

a	q_1	b
q_2	a	c

Primjer jednog legalnog prozora

Prvi redak prozora izražava da je stroj u stanju q_1, te da se glava za čitanje nalazi nad simbolom b. Drugi redak prozora izražava da je stroj prešao u stanje q_2, te da je obrisan simbol b i napisan je na njegovo mjesto simbol c. Na kraju se glava za čitanje pomaknula lijevo nad simbol a. To je legalni prozor jer smo bili zadali da vrijedi $(q_2, c, L) \in \delta(q_1, b)$.

a	q_1	b
a	a	q_2

Primjer drugog legalnog prozora

Primijetimo da je prvi redak prozora jednak kao i u prvom primjeru. Drugi redak izražava da je stroj zapisao simbol a na mjestu gdje je bio simbol b, prešao je u stanje q_2, te se je glava za čitanje pomaknula desno. To je legalni prozor jer smo bili definirali da vrijedi $(q_2, a, D) \in \delta(q_1, b)$.

Sada ćemo dati primjer jednog ilegalnog prozora.

a	b	a
a	a	a

Primjer jednog ilegalnog prozora

Dani prozor je ilegalni, jer za promjenu simbola b sa simbolom a glava stroja bi trebala biti iznad simbola b. To znači da bi u ćeliji ispred b trebalo biti zapisano neko stanje stroja.

Formula Φ_{pomak} osigurava da su svi prozori u tablici legalni. Svaki prozor sadrži 6 ćelija koje mogu biti postavljene na konačan broj načina kako bi tvorile legalan prozor. To povlači da je dobro definirana sljedeća formula:

$$\Phi_{pomak} = \bigwedge_{1\leq i<n^k,\ 1<j<n^k} \Big(\text{prozor } \ (i,j) \ \text{ je legalan}\Big).$$

Tekst "prozor (i, j) je legalan" možemo zamijeniti formulom logike sudova. Ako sa $a_1, \ldots, a_6$ označavamo sadržaj ćelija prozora, tada činjenicu "prozor (i, j) je legalan" možemo izraziti formulom logike sudova ovako:

$$\bigvee_{a_1,\ldots,a_6 \text{ je legalan prozor}} (x_{i,j-1,a_1} \ \wedge x_{i,j,a_2} \ \wedge \ x_{i,j+1,a_3} \ \wedge$$

$$x_{i+1,j-1,a_4} \ \wedge x_{i+1,j,a_5} \ \wedge \ x_{i+1,j+1,a_6}).$$

Iz konstrukcije formule Φ odmah slijedi da vrijedi: $w \in L$ ako i samo ako $\Phi \in$ SAT. Dokažimo još da je za svaku riječ $w \in \Gamma^*$ pripadnu formulu Φ moguće konstruirati u polinomnom vremenu. Označimo sa p broj elemenata skupa $C(= Q \cup \Gamma \cup \{\sharp\})$. Primijetimo da broj p ne ovisi o duljini ulazne riječi, već je određen strojem N. Tada je broj svih propozicionalnih varijabli jednak $n^k \cdot n^k \cdot p$, tj. $O(n^{2k})$. Formula Φ_{cel} je duljine $O(n^{2k})$, formula Φ_{start} je duljine $O(n^k)$, a formule $\Phi_{prihvat}$ i Φ_{pomak} su duljine $O(n^{2k})$. Tada je formula Φ duljine $O(n^{2k})$. To znači da je svaki jezik $L \in$ NP polinomno reducibilan na problem SAT. Q.E.D.

3.4 Verzije problema SAT

U ovoj točki prvo ćemo razmatrati k–SAT probleme. Već smo bili definirali probleme k–SAT, ali ćemo ovdje opet ponoviti tu definiciju. Ako u konjunktivnoj normalnoj formi svaka elementarna disjunkcija sadrži točno k literala tada govorimo o k–SAT problemu. Na kraju točke ćemo razmatrati PROBLEM Horn–SAT.

Teorem 3.18. *Vrijedi: 1–*SAT $\in P$.

Dokaz. Formula jezika 1–SAT su oblika $Q_1 \wedge \ldots \wedge Q_n$, gdje su Q_i literali. Takva formula je ispunjiva ako ne sadrži istovremeno neku propoziconalnu varijablu P i njenu negaciju. Kako bi ispitali sadrži li formula neku varijablu i njenu negaciju trebaju nam dvije "ugniježđene" petlje, pa algoritam ima vremensku složenost $O(n^2)$. Q.E.D

Teorem 3.19. *Vrijedi: 2–SAT* $\in$ *P.*

Teorem dokazujemo pomoću sljedećih dviju lema.

Lema 3.20. *Neka je F neka 2–knf sastavljena od propozicionalnih varijabli $P_1, \ldots, P_n$. Neka je G graf sa skupom vrhova: $V(G) = \{P_1, \ldots, P_n, \neg P_1, \ldots, \neg P_n\}$. Za svaki literal A definiramo $\overline{A} \equiv \neg P$, ako je A propozicionalna varijabla P, odnosno $\overline{A} \equiv P$, ako je $A \equiv \neg P$. Za svaku elementarnu disjunkciju $A \vee B$ od F definiramo da usmjereni graf G sadrži bridove $(\overline{A}, B)$ i $(\overline{B}, A)$.[1] Tada postoji deterministički Turingov stroj polinomne vremenske složenosti koji odgovara na pitanje postoji li u grafu G ciklus između neke propozicionalne varijable i njene negacije.*

Dokaz. Prvo moramo iz zadane 2–knf konstruirati graf G, tj. treba konstruirati skup vrhova i skup bridova. Za konstruiranje skupa vrhova potrebno je samo jednom proći po formuli F i identificirati sve propozicionalne varijable, a za konstruiranje skupa bridova treba opet proći po cijeloj formuli F i prema pravilu navedenom u iskazu leme treba spojiti vrhove označene odgovarajućim varijablama. Dakle, oba skupa je moguće konstruirati u polinomnom vremenu.

Nakon konstrukcije grafa G, za svaki $i = 1, \ldots, n$ treba provjeriti da li postoji put između vrha P_i i vrha $\neg P_i$, te da li postoji put između vrha $\neg P_i$ i vrha P_i. U lemi 2.41. na strani 46 pokazali smo da postoji polinomni algoritam za ispitivanje postojanja puta u grafu između dva zadana vrha. Q.E.D.

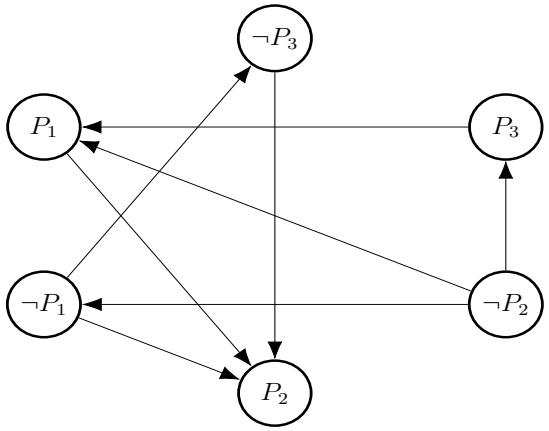
Sljedećim primjerom želimo ilustrirati kako za jednu zadanu 2–knf izgleda pripadni graf koji je definiran u prethodnoj lemi.

Primjer 3.21. *Neka je $F \equiv (P_1 \vee P_2) \wedge (P_1 \vee \neg P_3) \wedge (\neg P_1 \vee P_2) \wedge (P_2 \vee P_3)$. Neka je $V(G) = \{P_1, P_2, P_3, \neg P_1, \neg P_2, \neg P_3\}$. Kako bi definirali bridove grafa G kao u prethodnoj lemi, primijetimo prvo da za svaku elementarnu disjunkciju formule F redom imamo sljedeće bridove:*

elementarna disjunkcija	*bridovi*
$(P_1 \vee P_2)$	$(\neg P_1, P_2)$ *i* $(\neg P_2, P_1)$
$(P_1 \vee \neg P_3)$	$(\neg P_1, \neg P_3)$ *i* (P_3, P_1)
$(\neg P_1 \vee P_2)$	(P_1, P_2) *i* $(\neg P_2, \neg P_1)$
$(P_2 \vee P_3)$	$(\neg P_2, P_3)$ *i* $(\neg P_3, P_2)$.

Na sljedećoj slici ilustriramo graf G.

[1] Primijetite da vrijedi $(\overline{A} \vee B) \Leftrightarrow (A \to B)$ i $(\overline{B} \vee A) \Leftrightarrow (B \to A)$.



Slika 3.3: Graf G pridružen danoj formuli

Lema 3.22. *Neka je F neka 2–knf i G graf kao u prethodnoj lemi. Tada vrijedi: formula F je ispunjiva ako i samo ako u grafu G ne postoji ciklus koji sadrži neku propozicionalnu varijablu i njenu negaciju.*

Dokaz. Neka je F ispunjiva 2–knf. Neka je I neka interpretacija za koju vrijedi $I(F) = 1$. Pretpostavimo da postoji propozicionalna varijabla P tako da u grafu G postoji ciklus koji sadrži vrhove P i $\neg P$. Tada postoje literali $P = y_0,\ y_1,\ \ldots,\ y_s = \neg P,\ y_{s+1},\ \ldots,\ y_{s+t} = P$ tako da je za svaki $i \in \{0, 1, \ldots, s+t-1\}$ formula $y_i \to y_{i+1}$ ekvivalentna nekoj elementarnoj disjunkciji od F. Iz $I(F) = 1$ slijedi $I(y_i \to y_{i+1}) = 1$, za svaki i. Ako bi vrijedilo $I(P) = 0$ tada bi imali $I(\neg P) = 1$. No, tada iz $1 = I(\neg P \to y_{s+1}) = I(y_{s+1} \to y_{s+2}) = \ldots = I(y_{s+t-1} \to P)$ slijedi $I(P) = 1$, što je suprotno pretpostavci. Ako bi vrijedilo $I(P) = 1$ tada zbog $1 = I(P \to y_1) = I(y_1 \to y_2) = \ldots = I(y_{s-1} \to \neg P)$ slijedi $I(\neg P) = 1$, tj. $I(P) = 0$, što je opet suprotno pretpostavci.

Dokažimo sada obrat. Pretpostavimo da niti za jednu varijablu P formule F ne postoji ciklus u pripadnom grafu G između vrhova P i $\neg P$. Dokazat ćemo da je tada nužno formula F ispunjiva. Za svaku varijablu P od F za koju u grafu G ne postoji put od P do $\neg P$, a ni put od $\neg P$ do P, u graf G dodajemo brid $(P, \neg P)$. Označimo sa G' novi graf. Uočite da niti u novom grafu G' ne postoji neka varijabla P tako da graf G' sadrži ciklus između P i $\neg P$. Definiramo $I : Var(F) \to \{0, 1\}$ ovako:

$$I(P) = \begin{cases} 1, & \text{ako postoji put od } \neg P \text{ do } P; \\ 0, & \text{ako postoji put od } P \text{ do } \neg P. \end{cases}$$

Tvrdimo da vrijedi $I(F) = 1$. Pretpostavimo suprotno. Tada postoji barem jedna elementarna disjunkcija E od F tako da $I(E) = 0$. Označimo sa P i Q varijable koje nastupaju u E. Formula E može biti jednaka samo jednoj od sljedeće četiri formule: $P \vee Q$, $P \vee \neg Q$, $\neg P \vee Q$ i $\neg P \vee \neg Q$. Promotrimo slučaj kada je $E \equiv P \vee Q$.

Iz $I(E) = 0$ slijedi $I(P) = I(Q) = 0$. Iz definicije interpretacije I slijedi da postoje putovi u grafu G' od vrha P do vrha $\neg P$, te od vrha Q do vrha $\neg Q$. Budući da je $E \equiv P \vee Q$ elementarna disjunkcija u F tada iz definicije grafa G slijedi da su $(\neg P, Q)$ i $(\neg Q, P)$ bridovi u G, a onda i u G'. Time dobivamo da postoji ciklus između P i $\neg P$:

$$\text{put od } P \text{ do } \neg P; \quad \text{brid } (\neg P, Q); \quad \text{put od } Q \text{ do } \neg Q; \text{ brid } (\neg Q, P).$$

Na taj način je dobivena kontradikcija s činjenicom da graf G' ne sadrži ciklus između P i $\neg P$ niti za jednu varijablu P. Na sasvim analogni način bi se razmatrala preostala tri slučaja za elementarnu disjunkciju E. Q.E.D.

Za razliku od problema 1–SAT i 2–SAT, za problem 3–SAT još nije poznat polinomni algoritam za deterministički Turingov stroj. No, problem 3–SAT ima jedno važno svojstvo koje iskazujemo u sljedećem teoremu.

Teorem 3.23. *Problem 3–*SAT *je NP–potpun.*

Dokaz. Budući da je 3–SAT specijalni slučaj problema SAT očito vrijedi 3–SAT $\in$ NP. Iz propozicije 3.13. slijedi da za dokaz NP–potpunosti problema 3–SAT još treba dokazati da je problem SAT polinomno reducibilan na problem 3–SAT. Neka je $F(P_1, \ldots, P_n)$ proizvoljna konjunktivna normalna forma. Dokazujemo da postoji konjunktivna normalna forma $F'(P_1, \ldots, P_n, Q_1, \ldots, Q_m),\ m \geq 0$, tako da svaka elementarna disjunkcija sadrži točno tri literala, te za svaku parcijalnu interpretaciju $I : \{P_1, \ldots, P_n\} \to \{0, 1\}$ vrijedi:

$$I(F) = 1 \text{ ako i samo ako postoji proširenje } I' \text{ od } I \text{ tako da } I'(F') = 1.$$

Primijetimo da iz ove tvrdnje posebno slijedi da za svaku formulu F postoji 3–knf F' tako da vrijedi:

formula F je ispunjiva ako i samo ako F' je ispunjiva.

Prvo ćemo dokazati prethodno navedenu tvrdnju za jedan specijalan slučaj, tj. za formulu C koja je elementarna disjunkcija. Tvrdimo da postoji konjunktivna normalna forma C' koja ima sljedeća svojstva:

(i) svaka elementarna disjunkcija od C' sadrži točno tri literala;

(ii) za sve parcijalne interpretacije $I : Var(C) \to \{0, 1\}$ vrijedi: $I(C) = 1$ ako i samo ako postoji proširenje I' od I tako da vrijedi $I'(C') = 1$.

Varijable koje dolaze u formuli C označimo sa $P_1,\ P_2,\ \ldots$, a novo uvedene varijable u C' ćemo označavati sa $Q_1,\ Q_2,\ \ldots$ Neka je $C \equiv A_1 \vee \ldots \vee A_l$, gdje su A_i literali. Promatramo slučajeve obzirom na broj l.

(a) $l = 1$, tj. $C \equiv A_1$.
Tada definiramo da je formula C' jednaka:

$$(A_1 \vee Q_1 \vee Q_2) \wedge (A_1 \vee \neg Q_1 \vee Q_2) \wedge (A_1 \vee Q_1 \vee \neg Q_2) \wedge (A_1 \vee \neg Q_1 \vee \neg Q_2).$$

(b) $l = 2$, tj. $C \equiv A_1 \vee A_2$.
Tada definiramo:

$$C' \equiv (A_1 \vee A_2 \vee Q_1) \wedge (A_1 \vee A_2 \vee \neg Q_1).$$

(c) $l = 3$.
Tada neka je jednostavno C' upravo formula C.

(d) $l > 3$.
Neka je:

$$C' \equiv (A_1 \vee A_2 \vee Q_1) \wedge \bigwedge_{i=1}^{l-4} (\neg Q_i \vee A_{i+2} \vee Q_{i+1}) \wedge (\neg Q_{l-3} \vee A_{l-1} \vee A_l).$$

(Ako je $l = 4$ tada u C' imamo "praznu" konjunkciju, tj. konjunkciju oblika $\bigwedge_{i=1}^{0}$. Po definiciji smatramo da je takva konjunkcija jednaka tautologiji $Q_1 \vee \neg Q_1$. Odnosno, za slučaj $l = 4$ definiramo formulu C' ovako:

$$C' \equiv (A_1 \vee A_2 \vee Q_1) \wedge (\neg Q_1 \vee A_3 \vee A_4)\ ).$$

Preostalo je dokazati tvrdnju (*ii*). U slučaju c) to je trivijalno. Lako je vidjeti da za slučajeve a) i b) možemo uzeti proizvoljna proširenja I'. Preostalo je jedino razmotriti slučaj d). Pretpostavimo prvo da vrijedi $I(C) = 1$. Tada postoji literal A_p tako da je $I(A_p) = 1$. Proširenje I' od I definiramo po slučajevima obzirom na p.

(d_1) Ako je $p = 1$ ili $p = 2$ tada definiramo $I'(Q_1) = \ldots = I'(Q_{l-3}) = 0$.

(d_2) Ako je $p = l - 1$ ili $p = l$ tada definiramo $I'(Q_1) = \ldots = I'(Q_{l-3}) = 1$.

(d_3) Za broj p, za koji vrijedi $3 \leq p \leq l - 2$, definiramo

$$I'(Q_1) = \ldots = I'(Q_{p-2}) = 1, \qquad I'(Q_{p-1}) = \ldots = I'(Q_{l-3}) = 0.$$

Lako je provjeriti da u svim slučajevima vrijedi $I'(C') = 1$. Obrat tvrdnje dokazujemo obratom po kontrapoziciji. Neka je I interpretacija za koju vrijedi $I(C) = 0$, tj. $I(A_1) = \ldots = I(A_l) = 0$. Ako je I' proširenje od I tako da je $I'(C') = 1$ tada mora biti $I'(Q_1) = \ldots = I'(Q_{l-3}) = 1$. No, tada je i $I'(\neg Q_{l-3} \vee A_{l-1} \vee A_l) = 0$,

što povlači $I'(C') = 0$. Time je tvrdnja zadatka potpuno dokazana za slučaj kada je F elementarna disjunkcija. Promotrimo sada općenit slučaj, tj. kada je F proizvoljna konjunktivna normalna forma. Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$, gdje su C_i elementarne disjunkcije. Konstruiramo formule C_i' kao prije. (Konstrukcije možemo provesti tako da su skupovi novo uvedenih varijabli međusobno disjunktni). Tada definiramo $F' \equiv C_1' \wedge \ldots \wedge C_s'$. Lako je provjeriti da formula F' ima tražena svojstva. Time smo dokazali da se svaka zadaća iz SAT transformira u neku zadaću iz 3–SAT. Nije teško vidjeti da se ova transfomacija može realizirati u polinomnom vremenu.

Q.E.D.

Sada navodimo još jednu verziju problema SAT koja pripada klasi P. To je klasa ispunjivih Hornovih formula. Prvo dajemo potrebne definicije, te navodimo neke primjere.

Definicija 3.24. **Hornova klauzula** *je svaka elementarna disjunkcija s najviše jednim pozitivnim literalom.* **Hornova formula** *je svaka konjunkcija Hornovih klauzula.*

Navedimo neke primjere Hornovih klauzula: $\neg P_2 \vee \neg P_3 \vee \neg P_4 \vee P_2$, $\neg P_2 \vee \neg P_3 \vee \neg P_7$, $\neg P_4$, P_5.

Primijetimo da svaka Hornova klauzula može biti zapisana u ekvivalentnom obliku pomoću kondicionala:

$$\begin{aligned} \neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n \vee Q &\Leftrightarrow (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \to Q \\ \neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n &\Leftrightarrow (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \to \bot \\ P &\Leftrightarrow \top \to P. \end{aligned}$$

U daljnjim razmatranjima pretpostavljat ćemo da su Hornove klauzule napisane u kondicionalnom obliku. Hornove formule imaju važnu ulogu u intuicionističkoj logici i logičkom programiranju. Označimo: PROBLEM Horn–SAT$= \{F : F$ je ispunjiva Hornova formula$\}$.

Teorem 3.25. *PROBLEM Horn–SAT pripada klasi P.*

Dokaz prethodnog teorema je dan u Dodatku.

Zadaci

1. Neka $L_1 \subseteq \Gamma_1^*$ i $L_2 \subseteq \Gamma_2^*$ jezici tako da vrijedi: $L_1 \in$ P i $\emptyset \neq L_2 \neq \Gamma_2^*$. Dokažite da tada vrijedi $L_1 \leq_p L_2$.
   Rješenje. Neka $a \in L_2$ i $b \in \Gamma_2^* \setminus L_2$ proizvoljni. Definiramo funkciju $f : \Gamma_1^* \to \Gamma_2^*$ ovako:
$$f(w) = \begin{cases} a, & \text{ako } w \in L_1; \\ b, & \text{ako } w \in \Gamma^* \setminus L_1. \end{cases}$$

Budući da $L_1 \in \mathtt{P}$, tada je očito funkcija f vremenski polinomno izračunljiva. Zatim, očito vrijedi za svaki $w \in \Gamma_1^*$:

$$w \in L_1 \text{ ako i samo ako } f(w) \in L_2.$$

No, to upravo znači $L_1 \leq_p L_2$.

2. Dokažite da za svaki jezik $L \in \mathtt{P}$ vrijedi $L \leq_p$ PROBLEM PUT.
Rješenje. Neka je $L \in \mathtt{P}$ proizvoljan jezik. Tada postoji deterministički Turingov stroj T polinomne vremenske složenosti koji odlučuje jezik L. Označimo sa Γ alfabet stroja T. Neka $(G_1, u_1, v_1) \in$ PROBLEM PUT i $(G_2, u_2, v_2) \notin$ PROBLEM PUT proizvoljni. Definiramo funkciju f s domenom Γ^* ovako:

$$f(w) = \begin{cases} (G_1, u_1, v_1), & \text{ako } w \in L; \\ (G_2, u_2, v_2), & \text{ako } w \notin L. \end{cases}$$

Očito vrijedi: $w \in L$ ako i samo $f(w) \in$ PROBLEM PUT. Malom modifikacijom stroja T lako je dobiti da je funkcija f polinomno izračunljiva.

3. Dokažite da postoji polinomni algoritam koji za svaku formulu F logike sudova određuje konjunktivnu normalnu formu G tako da vrijedi: formula F je ispunjiva ako i samo ako formula G je ispunjiva.
Uputa: vidi [20].

4. Neka je $L \in$P neki NP–potpun jezik. Dokažite da je tada nužno P=NP.
Uputa. Neka je $L' \in$NP proizvoljan jezik. Budući da je jezik L jedan NP–potpun jezik, tada vrijedi $L' \leq_p L$. Klasa P je zatvorena "na dolje", pa slijedi $L' \in$P.

5. Neka je DOUBLE–SAT= $\{F : F$ je knf za koju postoje najmanje dvije interpretacije za koju je ona istinita$\}$. Dokažite da je ovaj jezik NP–potpun.
Rješenje. Očito vrijedi DOUBLE–SAT$\in$NP, jer su upravo dvije interpertacije za koju je zadana knf istinita jedan certifikat za nju. Dokažimo da je dani problem i NP–težak. Neka je $F(P_1, \ldots, P_n)$ neka knf. Tada definiramo knf F' ovako:

$$F' \equiv F \wedge (P_{n+1} \vee \neg P_{n+1})$$

Ako $F \in$ SAT tada očito $F' \in$ DOUBLE–SAT. Ako $F \notin$ SAT, tada ne postoji interperatacija I takva da vrijedi $I(F) = 1$, pa $F \notin$ DOUBLE–SAT. Dakle, vrijedi:

$$F \in \text{SAT ako i samo ako } F' \in \text{ DOUBLE–SAT}.$$

Za svaku knf F formulu F' je moguće konstruirati u polinomnom vremenu.

6. Problem XSAT (eng. Exact satisfiability) glasi: za danu knf F treba odrediti da li postoji interpretacija I takva da vrijedi $I(F) = 1$ i da je točno jedan literal u svakoj elementarnoj disjunkciji istinit. Dokažite da je problem XSAT jedan

NP–potpun problem.
Uputa. Za svaki literal Q uvodimo oznaku $\overline{Q}$ ovako:

$$\sim Q \equiv \begin{cases} \neg Q, & \text{ako je } Q \text{ prop. varijabla;} \\ P, & \text{ako } Q \equiv \neg P. \end{cases}$$

Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$ neka 3–knf (C_i je elementarna disjunkcija). Ako je $C_i \equiv Q_1 \vee Q_2 \vee Q_3$, te su P_1, P_2, P_3 i P_4 varijable koje ne nastupaju u formuli F, te ih još nismo upotrijebili za definiciju formule F', tada definiramo:

$$C_i' \equiv (\sim Q_1 \vee P_1 \vee P_2) \wedge (Q_2 \vee P_2 \vee P_3) \wedge (\sim Q_3 \vee P_3 \vee P_4)$$

Tada definiramo 3–knf F' ovako: $F' \equiv \bigwedge_{i=1}^{s} C_i'$.

7. Problem **MAX–2–SAT** glasi: za zadanu 2–knf F i prirodan broj k, potrebno je ispitati postoji li interpretacija I takva da je za nju istinito barem k elementarnih disjunkcija formule F. Dokažite da je problem MAX–2–SAT jedan NP–potpun problem.

8. **Booleov sklop** (eng. Boolean circuit), ili samo kratko sklop, je usmjereni graf $C = (V, E)$, gdje je $V = \{1, \ldots, m\}$ skup čvorova koji se ovdje nazivaju vrata. U grafu C svi bridovi su oblika (i, j) gdje je $i < j$. Sva vrata u grafu imaju ulazni stupanj jednak $0, 1$ ili 2. Svaka vrata i imaju definiranu vrstu $s(i)$, gdje je $s(i) \in \{\top, \bot, \wedge, \vee, \neg\} \cup \{P_1, \ldots, P_n\}$. Ulazni stupanj vrata i označavamo sa $st_{ul}(i)$, a definiran je ovako:

$$st_{ul}(i) = \begin{cases} 0, & \text{ako je } s(i) \in \{\top, \bot\} \cup \{P_1, \ldots, P_n\}; \\ 1, & \text{ako je } s(i) = \neg; \\ 2, & \text{ako je } s(i) \in \{\wedge, \vee\}. \end{cases}$$

Vrata bez ulaznih bridova zovu se ulazne vrijednosti sklopa C. Vrata m (koji nema izlaznih bridova) zovemo izlaznim vratima kruga.

Neka je $X(C)$ skup svih propozicionalnih varijabli koje se pojavljuju u sklopu C. Kažemo da je interpretacija I adekvatna za sklop C ako je definirana za svaki $P \in X(C)$. Za takvu interpretaciju I istinosna vrijednost vrata i definirana je rekurzivno ovako:

- ako je $s(i) = \top$ onda je $I(i) = 1$, a ako je $s(i) = \bot$ onda je $I(i) = 0$.
- ako je $s(i) \in X(C)$ onda je $I(i) = I(s(i))$.
- ako je $s(i) = \neg$ onda postoje jedinstvena vrata $j < i$ takva da je $(j, i) \in E$. Tada definiramo $I(i) = 1 - I(j)$.
- ako je $s(i) = \vee$ onda postoje dva ulazna brida (j, i) i (k, i). Tada definiramo $I(i) = 1$ ako i samo ako $I(j) = 1$ ili $I(k) = 1$.

- ako je $s(i) = \wedge$ onda postoje dva ulazna brida (j, i) i (k, i). Tada definiramo $I(i) = 1$ ako i samo $I(j) = I(k) = 1$.

Vrijednost interpretacije na sklopu C definiramo sa: $I(C) = I(m)$, gdje su m izlazna vrata sklopa.

PROBLEM SKLOP-SAT (eng. `CIRCUIT SAT`) glasi: za dani sklop C treba odrediti da li postoji interpretacija I adekvatna za sklop C takva da je $I(C) = 1$. Dokažite da je `PROBLEM SKLOP-SAT` jedan `NP`–potpun problem.

9. **PROBLEM VRIJEDNOST SKLOPA** (eng. `CIRCUIT VALUE`) glasi: za dani sklop C, čija niti jedna vrata nisu propozicionalna varijabla, treba odrediti postoji li interpretacija I adekvatna za C takva da je $I(C) = 1$. Dokažite da vrijedi `PROBLEM VRIJEDNOST SKLOPA` $\in$ `P`.

10. Problem **NAESAT** (eng. Not-All-Equal satisfiability) glasi: za danu knf F treba odrediti postoji li interpretacija I takva da vrijedi $I(F) = 1$ i da je barem jedan literal u svakoj elementarnoj disjunkciji neistinit. Dokažite da je `NAESAT` jedan `NP`–potpun problem.

3.5 Davis–Putnamov algoritam

Davis–Putnamov algoritam je algoritam za rješavanje `SAT` problema baziran na metodi rezolucije. Razvili su ga Martin Davis i Hilary Putnam 1960. godine, a kao njegovo poboljšanje 1962. godine Martin Davis, Hilary Putnam, George Logemann i Donald W. Loveland predstavljaju Davis–Putnam–Logemann–Loveland algoritam. Ovdje ćemo predstaviti Davis–Putnamov algoritam. U tu svrhu moramo prvo definirati neke pojmove.

Ako je Q neki literal tada sa $|Q|$ označavamo propozicionalnu varijablu koja je pridružena literalu Q, a sa $\overline{Q}$ označavamo pripadni dualni literal, tj.

$$\overline{Q} = \begin{cases} \neg Q, & \text{ako je } Q \text{ propozicionalna varijabla;} \\ P, & \text{ako je } Q \equiv \neg P, \text{ gdje je } P \text{ propozicionalna varijabla.} \end{cases}$$

Neka je F neka knf, tj. $F \equiv C_1 \wedge \ldots \wedge C_s$, gdje su C_i elementarne disjunkcije. U sljedećim razmatranjima pretpostavljamo da su elementarne disjunkcije zapisane u skupovnom obliku, tj. umjesto $C \equiv Q_1 \vee \ldots Q_k$, gdje su Q_j literali, promatrat ćemo $C = \{Q_1, \ldots, Q_k\}$. Skup $\{C_1, \ldots, C_s\}$ nazivat ćemo skup klauzula formule F, te ćemo ga označavati sa C_F.

Primjer 3.26. *Neka je* $F \equiv (P_1 \vee P_3 \vee \neg P_5) \wedge (P_2 \vee \neg P_4) \wedge (\neg P_3 \vee \neg P_4 \vee \neg P_5)$. *Tada je skup klauzula formule* F *jednak* $\{\{P_1, P_3, \neg P_5\},\ \{P_2, \neg P_4\},\ \{\neg P_3, \neg P_4, \neg P_5\}\}$.

Neka je F neka knf, te C i D neke klauzule od F. Kažemo da je definirana rezolucija za klauzule C i D ako postoji literal Q tako da vrijedi $Q \in C$ i $\overline{Q} \in D$. U tom slučaju definiramo $Res(C, D) = C \setminus \{Q\} \cup D \setminus \{\overline{Q}\}$.

Za proizvoljnu knf F i proizvoljan skup klauzula Σ definiramo:

$$res_F(\Sigma) = C_F \cup \{Res(C,D) : \ C, D \in C_F \cup \Sigma, \ \text{rezolucija je definirana za klazule } C \text{ i } D, \text{ te } Res(C,D) \text{ nije tautologija}\}.$$

Očito je res_F monotoni operator. Primjenom Knaster–Tarskijevog teorema (vidi primjerice [27]) slijedi da postoji fiksna točka tog operatora. Fiksnu točku označavamo sa $Res(F)$. To je najmanje zatvorenje knf F po rezoluciji. Sljedeći teorem je osnova Davis–Putnamovog algoritma.

Teorem o potpunosti rezolucije. Neka je F proizvoljna knf. Formula F je ispunjiva ako i samo $\emptyset \notin Res(F)$.

Kako bi mogli napisati pseudokod Davis–Putnamovog algoritma moramo uvesti još neke pojmove. Za literal Q koji nastupa u knf F kažemo da je čist za F ako dualni literal $\overline{Q}$ ne nastupa u formuli F. Neka je Σ neki skup klauzula, te P neka propozicionalna varijabla. Definiramo novi skup kaluzula:

$$Res(\Sigma, P) = \begin{cases} \Sigma \setminus \{Q\}, \text{ ako je } |Q| = P, \ \text{ te je literal } Q \text{ čist za } \Sigma; \\ \\ \{C \cup D : C \cup \{P\} \in \Sigma, \ D \cup \{\neg P\} \in \Sigma, \text{ te } C \cup D \text{ nije tautologija}\} \\ \cup \{C \in \Sigma : \text{ varijabla } P, \ \text{ a ni literal } \neg P, \text{ se ne pojavljuju} \\ \quad \text{u klauzuli } C\}, \text{ inače.} \end{cases}$$

Davis–Putnamov algoritam koji na ulazu ima skup klauzula $\Sigma := C_F$ neke knf F glasi:

1. Ako je $\Sigma = \emptyset$ tada je formula F ispunjiva.
2. Ako je $\emptyset \in \Sigma$ tada formula F nije ispunjiva.
3. Ako je $\Sigma \neq \emptyset$ i $\emptyset \notin \Sigma$, tada za svaku varijablu P koja se pojavljuje u formuli F primijeni: $\Sigma := Res(\Sigma, P)$.

Vremenska složenost danog algoritma je $O(1,696^n)$, a malim modifikacijama dobivamo algoritam složenosti $O(1,618^n)$. Kako bi ilustrirali što takvo poboljšanje znači navodimo sljedeće potencije:

$$\begin{aligned} 2^{100} &\approx 1\ 267\ 650\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000 \\ 1,696^{100} &\approx 87\ 616\ 270\ 000\ 000\ 000\ 000\ 000 \\ 1,618^{100} &\approx 790\ 408\ 700\ 000\ 000\ 000\ 000. \end{aligned}$$

Davis–Putnamov algoritam je efikasan za formule koje nemaju više od 500 varijabli. Nažalost, u mnogim primjenama pojavljuju se formule s više od 1000 varijabli.

Iz tog razloga osim egzaktnih metoda razmatraju se i stohastičke metode, primjerice GSAT i WalkSAT. Više detalja o Davis–Putnamovom algoritmu i njegovim verzijama možete naći u [19], [20] i [21].

3.6 Algoritmi na grafovima

U ovoj točki promatrat ćemo nekoliko problema u vezi grafova, te ćemo dokazati njihovu NP–potpunost. Sve pojmove vezane uz grafove već smo bili definirali na stranama 46 i 52.

Istaknimo još da smo u prethodnom poglavlju dokazali da je problem 3–SAT polinomno reducibilan na PROBLEM KLIKA (vidi teorem 3.14. na strani 61). Zatim, bili smo dokazali da je problem 3–SAT jedan NP–potpun problem (vidi teorem 3.23. na strani 69). Primjenom propozicije 3.13. slijedi tada tvrdnja sljedećeg teorema.

Teorem 3.27. *PROBLEM KLIKA je NP–potpun problem.*

Neka je (G, E) neusmjereni graf. **Pokrivač bridova** grafa G je $W \subseteq G$ tako da za svaki brid $\{a, b\} \in E$ vrijedi $\{a, b\} \cap W \neq \emptyset$. Dakle, pokrivač bridova sadrži barem jedan kraj svakog brida grafa. Za pokrivač bridova W kažemo da je **k–pokrivač bridova** ako skup W ima točno k članova.

PROBLEM POKRIVAČ BRIDOVA sastoji se od određivanja da li zadani graf sadrži pokrivač bridova zadane veličine, tj. formalno:

$$\text{PROBLEM POKRIVAČ BRIDOVA} = \{\langle G, k\rangle \ : \ \begin{array}{l} G \text{ je graf koji sadrži} \\ k\text{–pokrivač bridova}\}. \end{array}$$

Propozicija 3.28. *Vrijedi: PROBLEM POKRIVAČ BRIDOVA $\in$ NP.*

Dokaz. Neka je (G, R) proizvoljni graf i $k \in \mathbb{N}$. Za proizvoljni podskup W od G definiramo certifikat T ovako:

1. Ako je $|W| \neq k$, odbaci.
2. Za svaki vrh $v \in W$:
3. Za svaki brid $\{u, w\} \in R$: ako je $(u = v)$ ili $(w = v)$ označi brid $\{u, w\}$
4. Za svaki brid $(u, w) \in R$ ako $\{u, w\}$ nije označeno, odbaci.
5. Prihvati.

Očito stroj T prihvaća (G, R, k, W) ako i samo ako je $|W| = k$ i svaki brid iz R ima bar jednu točku u W. Složenost je očito $O(|W| \times |R|)$. Dakle konstruirali smo certifikat koji je polinoman, stoga slijedi PROBLEM POKRIVAČ BRIDOVA $\in$ NP. Q.E.D.

Teorem 3.29. *Vrijedi: 3–SAT $\leq_p$ PROBLEM POKRIVAČ BRIDOVA.*

Dokaz. Neka je F neka 3–knf. Označimo sa n broj propozicionalnih varijabli formule F, a sa s broj elementarnih disjunkcija formule F. Neka je $k := n + 2s$. Definiramo graf (G, R) sa $2n + 3s$ čvorova koji će imati sljedeće svojstvo:

formula F je ispunjiva ako i samo ako graf G sadrži k–člani pokrivač bridova.

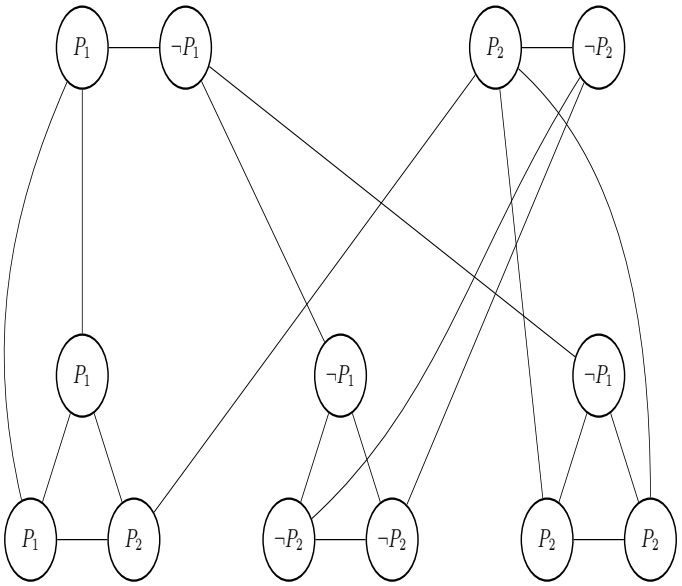
Uočite da je k strogo manji od broja čvorova. Čvorovi grafa G su označeni:

- literalima formule F (to je $3s$ čvorova);
- sa P i $\neg P$, za svaku propozicionalnu varijablu P koja se pojavljuje u formuli F (to je $2n$ čvorova).

Primjerice, za formulu $F \equiv (P_1 \vee P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_2) \wedge (\neg P_1 \vee P_2 \vee P_2)$ čvorovi grafa G su označeni redom sa: P_1, P_1, P_2, $\neg P_1$, $\neg P_2$, $\neg P_2$, $\neg P_1$, P_2, P_2, P_1, $\neg P_1$, P_2 i $\neg P_2$. Definiramo bridove grafa G ovako:

- sve čvorove označene literalima iz jedne elementarne disjunkcije međusobno spojimo;
- svaki par dodatnih čvorova oblika P i $\neg P$ međusobno spojimo;
- svaki dodatni čvor Q spojimo sa svakim čvorom iz svake elementarne disjunkcije koji je također označen sa Q.

Za formulu $F \equiv (P_1 \vee P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_2) \wedge (\neg P_1 \vee P_2 \vee P_2)$ dajemo pripadni graf.



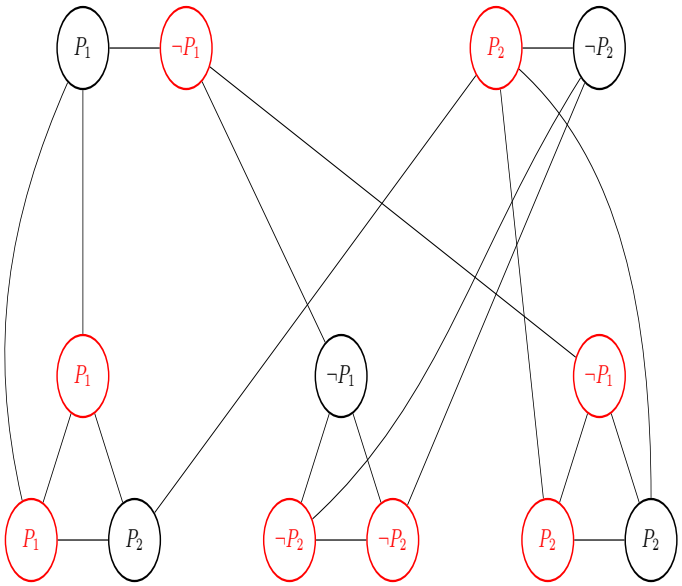
Slika 3.4: Pripadni graf za zadanu formulu

Tvrdimo da za svaku 3–knf F vrijedi:

formula F je ispunjiva ako i samo ako graf G sadrži k–člani pokrivač bridova.

Pretpostavimo prvo da je formula F ispunjiva. Neka je I neka interpretacija za koju vrijedi $I(F) = 1$. Definiramo skup W (traženi k–pokrivač bridova). Skup W sadrži sve dodatne čvorove Q za koje vrijedi $I(Q) = 1$. Budući da je $I(F) = 1$ tada u svakoj elementarnoj disjunkciji postoji literal koji je istinit za interpretaciju I. Iz svake elementarne disjunkcije odaberemo po jedan literal koji je istinit za I. Skup W sadrži i sve čvorove iz elementarnih disjunkcije koje nismo odabrali zbog istinitosti.

Za ispunjivu formulu $F \equiv (P_1 \vee P_1 \vee P_2) \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_2) \wedge (\neg P_1 \vee P_2 \vee P_2)$ i interpretaciju I koja je definirana sa $I(P_1) = 0$ i $I(P_2) = 1$, čvorovi skupa W su na sljedećoj slici označeni crveno.



Slika 3.5: Graf s istaknutim čvorima koji pripadaju skupu W

Uočimo da skup W sadrži k čvorova ($2s$ čvorova iz elementarnih disjunkcija, te n iz skupa dodatnih čvorova: definirali smo $k = 2s + n$). Ako je (u, v) neki brid grafa G tada su moguća tri slučaja:

- čvorovi u i v pripadaju istoj elementarnoj disjunkciji.
  Tada iz definicije skupa W slijedi da je barem jedan čvor element iz W.

- čvor u je iz neke elementarne disjunkcije, a v je dodatni čvor (tada su u i v označeni istim literalima).
  Ako je $I(v) = 1$ tada iz defincije skupa W slijedi $v \in W$. Ako pak je $I(v) = 0$, tada je i $I(u) = 0$. Budući da je u literal iz elementarne disjunkcije tada je $u \in W$.

- oba čvora u i v su dodatni čvorovi. Tada je $u \Leftrightarrow v$.
  Tada vrijedi $I(u) = 1$ ili $I(v) = 1$. Iz definicije skupa W tada slijedi $u \in W$ ili $v \in W$.

Dakle, W je k–pokrivač bridova grafa G.

Pretpostavimo sada da graf G sadrži k–pokrivač bridova W. Iz definicije grafa G slijedi da W sadrži točno jedan dodatni čvor svakog para P i $\neg P$. Definiramo interpretaciju $I : Var(F) \to \{0, 1\}$ ovako:

$$I(P) = \begin{cases} 1, \text{ ako } P \in W; \\ \\ 0, \text{ ako } \neg P \in W. \end{cases}$$

Lako je provjeriti da vrijedi $I(F) = 1$. Q.E.D.

Korolar 3.30. *PROBLEM POKRIVAČ BRIDOVA je NP–potpun.*

Dokaz. Propozicija 3.28. i teorem 3.29.

Prilikom razmatranja problema SAT, već smo bili definirali PROBLEM k-OBOJIVOSTI grafa (vidi definiciju 3.15. na strani 62), te smo naveli da je problem 3–SAT polinomno reducibilan na PROBLEM 3-OBOJIVOSTI. U sljedećoj propoziciji ističemo još jednu vezu problema obojivosti grafa i matematičke logike. Pojmove i tvrdnje iz matematičke logike možete pronaći u [25].

Propozicija 3.31. *Neka je G prebrojiv skup. Graf (G, R) je k–obojiv ako i samo ako je svaki konačan podgraf od (G, R) k–obojiv.*[2]

Dokaz. Ako je graf (G, R) k–obojiv tada je očito i svaki konačan podgraf od (G, R) k–obojiv. Dokažimo obrat. Pretpostavimo da je svaki konačan podgraf od (G, R) k–obojiv. Promotrimo logiku sudova čiji je skup propozicionalnih varijabli jednak $\{1, \ldots, k\} \times G$. Uočite da je to prebrojiv skup, jer je po pretpostavci skup G prebrojiv. Označimo sa S skup koji sadrži sljedeće formule:

(1) $(i, a) \to \neg(j, a)$, za sve $i \neq j$ i svaki $a \in G$
(uočite da ova formula izražava uvjet da je svaki čvor grafa obojen najviše jednom bojom);

(2) $(1, a) \vee (2, a) \vee \ldots \vee (k, a)$, za svaki $a \in G$
(ova formula izražava uvjet da je svaki čvor grafa obojen barem s jednom bojom);

[2]Danu tvrdnju su dokazali N. G. de Bruijn i P. Erdös. Korištenje teorema kompaktnosti zamjenjuje dosta složenu primjenu teorema Tychonoffa ili Königove leme. Teorem Tychonoffa govori da je produkt proizvoljne familije kompaktnih topoloških prostora ponovo kompaktan topološki prostor.

(3) $(i,a) \to \neg(i,b)$, za svaki $i \leq k$ i svaki $(a,b) \in R$
(ovom formulom je izrečeno da povezani čvorovi ne mogu biti obojeni istom bojom).

Budući da je po pretpostavci svaki konačan podgraf od (G,R) k–obojiv tada je svaki konačan podskup od S ispunjiv. Tada iz teorema kompaktnosti slijedi da je i skup formula S ispunjiv. Neka je I interpretacija takva da je $I(S) = 1$. Sada definiramo traženu particiju $B_1, \ldots, B_k$ skupa G sa: $a \in B_i$ ako i samo ako $I(i,a) = 1$.[3] Q.E.D.

Napomena 3.32. *Za graf kažemo da je planarni ako se grafički može predočiti tako da se svi njegovi bridovi sijeku samo u čvorovima. Poznati problem četiri boje tvrdi da je svaki planarni graf 4–obojiv. Problem četiri boje je pozitivno riješen 1976. godine uz veliku pomoć računala. O problemu četiri boje možete čitati npr. u knjizi D. Veljan, Kombinatorika s teorijom grafova, Školska knjiga, Zagreb, 1989.*

Teorem 3.33. *PROBLEM* 3*-OBOJIVOSTI je NP–potpun.*

Dokaz prethodnog teorema je dan u Dodatku.

Zadaci

1. PROBLEM SUMA PODSKUPA glasi: za zadani konačni multiskup $S \subseteq \mathbb{N}$ i $t \in \mathbb{N}$ treba odrediti da li postoji $T \subseteq S$ tako da vrijedi $\sum_{x \in T} x = t$. Dokažite da je PROBLEM SUMA PODSKUPA jedan NP–potpun problem.

2. PROBLEM PARTICIJA glasi: za zadani multiskup S prirodnih brojeva odrediti da li postoji $T \subseteq S$ tako da vrijedi $\sum_{x \in T} x = \sum_{x \in S \setminus T} x$. Dokažite da je PROBLEM PARTICIJA jedan NP–potpun problem.
Rješenje. Lako je vidjeti da vrijedi PROBLEM PARTICIJA $\in$ NP. Dokažimo da je PROBLEM SUMA PODSKUPA polinomno reducibilan na PROBLEM PARTICIJA. Neka je S neki multiskup prirodnih brojeva, te $t \in \mathbb{N}$. Označimo $s = \sum_{x \in S} x$. Promotrimo slučaj kada je $2t \geq s$. Neka je S' multiskup definiran sa $S' = S \cup \{2t - s\}$ (ovo nije obična unija budući da se radi o multiskupovima). Očito je S' moguće dobiti iz S i t u polinomnom vremenu. Tvrdimo da vrijedi:

   postoji $T \subseteq S$ takav da je $\sum_{x \in T} x = t$ ako i samo ako postoji $T' \subseteq S'$ tako da $\sum_{x \in T'} x = \sum_{x \in S' \setminus T'} x$.

[3] Preporučamo da svakako pročitate članak *V. Kovač, Kromatski broj ravnine – neriješeni problem o bojenju*, Hrvatski matematički elektronski časopis math.e, 6 (2005).

Treba naglasiti da $S' \setminus T'$ nije obična skupovna razlika budući da se radi o multiskupovima.

Pretpostavimo prvo da postoji $T \subseteq S$ takav da vrijedi $\sum_{x\in T} x = t$. Neka je T_1 multiskup kojeg smo dobili iz S izbacujući sve elemente iz T. Tada imamo $\sum_{x\in T_1} x = \sum_{y\in S} y - \sum_{z\in T} z = s - t$. Neka je T' multiskup kojeg smo dobili tako da smo iz S' izbacili sve elemente od T. Tada vrijedi:

$$\sum_{x\in T'} x = \sum_{x\in T_1} x + (2t - s) = (s - t) + (2t - s) = t = \sum_{x\in T} x = \sum_{x\in S'\setminus T'} x.$$

Pretpostavimo sada da postoji $T' \subseteq S'$ tako da vrijedi $\sum_{x\in T'} x = \sum_{x\in S'\setminus T'} x$. No, budući da $\sum_{x\in S'} x = s+(2t-s) = 2t$, tada $\sum_{x\in T'} x = \sum_{x\in S'\setminus T'} x = t$. Ako $(2t-s) \in T'$ tada za T možemo uzeti $S'\setminus T'$, ako pak $2t-s) \in S'\setminus T'$ tada definiramo $T = T'$. Sasvim analogno se dokazuje slučaj kada je $2t < s$.

Po prethodnom zadatku znamo da je `PROBLEM SUMA PODSKUPA` jedan `NP`–potpun problem. Iz propozicije 3.13. tada slijedi da je i `PROBLEM PARTICIJA` jedan `NP`–potpun problem.

3. Neka su $a_1, \ldots, a_n, b \in \mathbb{Z}$. Nejednadžbu oblika $a_1x_1 + \ldots + a_nx_n \leq b$ nazivamo linearna diofantska nejednadžba. **PROBLEM LDN** (linearne diofantske nejednadžbe) glasi: za dani sistem linearnih diofantskih nejednadžbi treba odrediti ima li cjelobrojno rješenje. Dokažite da je ovaj problem `NP`–potpun.
Uputa. Lako je pokazati da vrijedi `PROBLEM LDN` $\in$ `NP`. Dokazujemo da vrijedi 3–`SAT` $\leq_p$ `PROBLEM LDN`. Neka je $F(P_1, \ldots, P_k)$ neka 3–knf. Definiramo sistem linearnih diofantskih nejednadžbi ovako:

$$(*)\begin{cases} 0 \leq x_i \leq 1, \quad \text{za svaki} \quad i \in \{1, \ldots, k\} \\ x_{i_1} + x_{i_2} + x_{i_3} \geq 1, \quad \text{ako je} \quad P_{i_1} \vee P_{i_2} \vee P_{i_3} \quad \text{elem. dis. od} \quad F \\ x_{i_1} + x_{i_2} + (1 - x_{i_3}) \geq 1, \quad \text{ako je} \quad P_{i_1} \vee P_{i_2} \vee \neg P_{i_3} \quad \text{elem. dis. od} \quad F \\ x_{i_1} + (1 - x_{i_2}) + (1 - x_{i_3}) \geq 1, \quad \text{ako je} \quad P_{i_1} \vee \neg P_{i_2} \vee \neg P_{i_3} \\ \qquad\qquad \text{elementarna disjunkcija od} \quad F \\ (1 - x_{i_1}) + (1 - x_{i_2}) + (1 - x_{i_3}) \geq 1, \quad \text{ako je} \quad \neg P_{i_1} \vee \neg P_{i_2} \vee \neg P_{i_3} \\ \qquad\qquad \text{elementarna disjunkcija od} \quad F. \end{cases}$$

Očito vrijedi: sistem $(*)$ ima cjelobrojno rješenje ako i samo ako formula F je ispunjiva. Provjerite da je u polinomnom vremenu moguće za zadanu 3–knf F konstruirati sistem $(*)$.

4. PROBLEM BLOKIRAJUĆI SKUP glasi:

   > *Neka je* $\{A_1, \dots A_n\}$ *skup konačnih skupova, te* $m \in \mathbb{N}$. *Treba odrediti postoji li skup* A *koji ima najviše* m *elemenata, te da za svaki* $i \in \{1, \dots, n\}$ *vrijedi* $A \cap A_i \neq \emptyset$.

   Dokažite da je ovaj problem NP–potpun.
   Uputa. Lako je pokazati da ovaj problem pripada klasi NP. Dokazujemo da vrijedi 3–SAT $\leq_p$ PROBLEM BLOKIRAJUĆI SKUP. Neka je $F \equiv C_1 \wedge \dots \wedge C_s$ neka 3–knf, te neka $Var(F) = \{P_1, \dots, P_k\}$. Za svaki $i \in \{1, \dots, s\}$ neka se skup A_i sastoji od svih literala elementarne disjunkcije C_i. Zatim, neka je za svaki $j \in \{1, \dots, k\}$ skup A_{s+j} jednak $\{P_j, \neg P_j\}$. Definiramo još $m = k$. Tada vrijedi:

   > za skup $\{A_1, \dots, A_s, A_{s+1}, \dots, A_{s+k}\}$ postoji skup A s najviše m elemenata koji ima svojstvo $A \cap A_i \neq \emptyset$, za svaki $i \in \{1, \dots, s+k\}$ ako i samo ako formula F je ispunjiva.

   Provjerite da je u polinomnom vremenu moguće za zadanu 3–knf F konstruirati skup $\{A_1, \dots, A_{s+k}\}$.

5. Dokažite da vrijedi PROBLEM POKRIVAČ BRIDOVA $\leq_p$ PROBLEM KLIKA.
   Rješenje. Neka je $G = (V, R)$ neki graf. Komplement grafa G označavamo sa $G^c = (V, R^c)$, pri čemu je $R^c = \{\{x, y\} : x, y \in V,\ \{x, y\} \notin R\}$.

   Ako graf G sadrži neki k–pokrivač bridova S, tj. vrijedi $\{x, y\} \cap S \neq \emptyset$ za svaki $\{x, y\} \in T$, te je $|S| = k$, tada definiramo $S^c = V \setminus S$. Tada imamo: $|S^c| = |V| - |S| = |V| - k$.

   Tvrdimo da vrijedi: graf G sadrži neki k–pokrivač bridova S ako i samo ako komplement G^c sadrži $(|V| - k)$–kliku.

   Pretpostavimo prvo da graf G sadrži neki k–pokrivač bridova S. Tada za svaki dvočlani skup $\{x, y\}$ za koji $\{x, y\} \cap S = \emptyset$ očito vrijedi $\{x, y\} \notin R$, tj. $\{x, y\} \in R^c$. Dakle, za sve $x, y \in S^c$ imamo $\{x, y\} \in R^c$. To upravo znači da je S^c jedna $(|V| - k)$–klika grafa G^c.

   Analogno se dokazuje obrat.

6. Dokažite da vrijedi PROBLEM KLIKA $\leq_p$ PROBLEM POKRIVAČ BRIDOVA.

7. Neka je (G, R) neusmjereni graf. Za skup $N \subseteq G$ kažemo da je **nezavisan skup** ako za svaki $u, v \in N$ vrijedi $\{u, v\} \notin R$. PROBLEM NEZAVISAN SKUP glasi: za zadani neusmjereni graf G i prirodan broj k odrediti da li graf G sadrži nezavisan skup veličine k. Dokažite da vrijedi PROBLEM KLIKA $\leq_p$ PROBLEM NEZAVISAN SKUP, te PROBLEM NEZAVISAN SKUP $\leq_p$ PROBLEM KLIKA.

8. PROBLEM IS-KLIKA glasi: za zadani neusmjereni graf G i prirodan broj k odrediti sadrži li G kliku veličine k ili nezavisan skup veličine k. Dokažite da je

PROBLEM IS-KLIKA jedan NP–potpun problem.
Uputa. Neka je $G = (V, E)$ graf, te k prirodan broj. Označimo $n := |V|$. Neka je $W = \{w_1, \ldots, w_n\}$ takav da je $W \cap V = \emptyset$. Neka je $V' = W \cup V$ i $E' = E$, te $G' = (V', E')$. Tvrdimo da vrijedi:

> graf G sadrži nezavisan skup veličine k ako i samo ako graf G' sadrži kliku veličine $n + k$ ili nezavisan skup veličine $n + k$.

9. Neka je (G, R) neusmjereni graf. Za $D \subseteq G$ kažemo da je **dominantan skup** ako za svaki $v \in G \setminus D$ postoji $u \in D$ tako da je $\{u, v\} \in R$. PROBLEM DOMINANTAN SKUP glasi: za zadani neusmjereni graf G i prirodan broj m odrediti sadrži li graf G dominantan skup veličine m. Dokažite da je PROBLEM DOMINANTAN SKUP jedan NP–potpun problem.

Rješenje. Dokazujemo da je problem 3–SAT polinomno reducibilan na PROBLEM DOMINANTAN SKUP. Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$ neka 3–knf. Neka su $P_1, \ldots, P_k$ sve varijable koje nastupaju u formuli F. Neka je skup vrhova zadan sa:

$$G = \{P_1, \ldots, P_k, \neg P_1, \ldots, \neg P_k, 1, \ldots, k, C_1, \ldots, C_s\}.$$

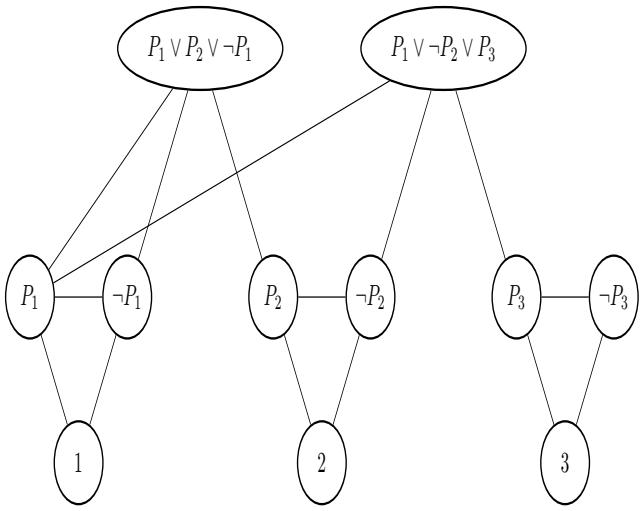
Definiramo skup bridova R ovako:

- $\{P_i, \neg P_i\}$, $\{P_i, i\}$, $\{\neg P_i, i\}$ $\in R$, za svaki $i \in \{1, \ldots, k\}$;
  (time je svaka trojka čvorova $P_i, \neg P_i$, i povezana u trokut)
- ako literal Q nastupa u nekoj elementarnoj disjunkciji C_i formule F tada definiramo $\{Q, C\} \in R$.

Definiramo $m = k$. Tvrdimo da vrijedi: formula F je ispunjiva ako i samo ako pripadni graf G sadrži dominantan skup veličine m. Pretpostavimo prvo da je formula F ispunjiva. Neka je I interpretacija takva da je $I(F) = 1$. Neka je D skup svih literala Q za koje vrijedi $I(Q) = 1$. Uočimo da skup D sadrži točno k elemenata. Lako je vidjeti da je D dominantan skup za graf G.

Pretpostavimo sada da je D dominantan skup za graf G koji sadrži m vrhova. Lako je vidjeti da za svaki $i \in \{1, \ldots, k\}$ vrijedi da P_i i $\neg P_i$ nisu istovremeno elementi od D. Definiramo interpretaciju I ovako: $I(P_i) = 1$ ako je $P_i \in D$, odnosno $I(P_i) = 0$ inače. Lako je vidjeti da za svaki $j \in \{1, \ldots, s\}$ vrijedi $I(C_j) = 1$, tj. $I(F) = 1$.

Za ilustraciju na sljedećoj slici dajemo pripadni graf za formulu $F \equiv (P_1 \vee P_2 \vee \neg P_1) \wedge (P_1 \vee \neg P_2 \vee P_3)$.



10. PROBLEM POVEZAN DOMINANTAN SKUP glasi: za zadani neusmjereni graf G i prirodan broj k odrediti sadrži li graf G dominantan skup veličine najviše k tako da je podgraf G' induciran sa D povezan. Dokažite da je PROBLEM POVEZAN DOMINANTAN SKUP jedan NP–potpun problem.
Uputa. PROBLEM POKRIVAČ BRIDOVA je moguće polinomno reducirati na PROBLEM POVEZAN DOMINANTAN SKUP. Neka je $G = (V, E)$ graf, te k prirodan broj. Konstruiramo graf $G' = (V', E')$ na sljedeći način:

    1. Skup vrhova G' sadrži sve vrhove grafa G i svaka dva takva vrha su povezana bridom;

    2. Za svaki brid $\{x, y\} \in E$, graf G sadrži vrh v_{xy} koji je povezan s vrhovima x i y.

    Tada vrijedi:

    graf G ima pokrivač bridova veličine najviše k ako i samo ako graf G' ima povezan dominantan skup veličine najviše k.

Poglavlje 4

NP–teški problemi

U ovom poglavlju razmatrat ćemo probleme koji su "skoro" NP–potpuni. Ti problemi imaju svojstvo da se svaki problem iz klase NP može polinomno reducirati na njih, a da oni sami možda uopće ne pripadaju klasi NP. Prvo ćemo razmatrati problem egzistencije cjelobrojnih nultočaka diofantskih jednadžbi. Nakon toga ćemo navesti neke probleme optimizacije. O NP–teškim problemima možete puno više naći u knjizi [13].

Definicija 4.1. *Za neki jezik L kažemo da je* **NP–težak** *ako je svaki jezik iz klase NP polinomno reducibilan na jezik L.*

Očito je svaki NP–potpun problem ujedno i NP–težak. No, NP–težak problem ne mora pripadati klasi NP. Upravo je takav problem egzistencije cjelobrojnih nultočaka diofantskih jednadžbi. Sada ćemo dokazati da je to NP–težak problem.

Neka je $p(x_1, \ldots, x_n)$ polinom s cjelobrojnim koeficijentima, tj. $p \in \mathbb{Z}[X_1, \ldots, X_n]$. Jednadžba oblika $p(x_1, \ldots, x_n) = 0$ naziva se diofantska jednadžba. Takve su primjerice sljedeće jednadžbe $x^2 + y^2 - z^2 = 0$ i $6x^{18} - x + 3 = 0$.

Postavlja se pitanje ima li dana diofantska jednadžba cjelobrojna rješenja. Za neke posebne slučajeve odavno su poznati algoritmi pomoću kojih se određuje da li dana diofantska jednadžba ima cjelobrojno rješenje. To su primjerice diofantske jednadžbe s jednom nepoznanicom, te linearne diofantske jednadžbe s dvije nepoznanice. Na drugom svjetskom kongresu matematičara u Parizu 1900. godine njemački matematičar David Hilbert održao je predavanje pod naslovom *Matematički problemi*. Iznio je 23 problema za koje je smatrao da su ključni za razvitak matematike u novom mileniju.

Deset Hilbertov problem glasi:

> *Neka je zadana proizvoljna diofantska jednadžba s proizvoljnim brojem nepoznanica. Postoji li algoritam pomoću kojeg je nakon konačno mnogo koraka moguće odrediti ima li ta jednadžba rješenje u skupu cijelih brojeva?*

J. V. Matijašević je 1969. godine dokazao da traženi algoritam ne postoji. To ističemo u sljedećem teoremu.

Teorem 4.2. *Neka je* $Dif = \{p : p \in \mathbb{Z}[X_1, \ldots, X_n]$ *koji ima cjelobrojnu nultočku*$\}$. *Problem* Dif *je neodlučiv. Posebno,* $Dif \notin NP$.

Dokaz prethodnog teorema možete pogledati u [4].

Teorem 4.3. *Problem 3–*SAT *je polinomno reducibilan na problem* Dif.

Dokaz. Moramo naći vremenski polinomno izračunljivu funkciju f takvu da za proizvoljnu 3–knf F vrijedi:

$$F \in 3\text{–}\mathtt{SAT} \qquad \text{ako i samo ako} \qquad f(F) \in Dif.$$

Za danu 3–knf formulu F konstruirat ćemo polinom $p_F \in \mathbb{Z}[\vec{X}]$ koji će imati cjelobrojnu nultočku ako i samo ako je formula F ispunjiva. Kako bi mogli definirati polinom p_F prvo ćemo definirati polinom q_F, i to prvo za slučaj kada je F elementarna disjunkcija s tri literala.

Neka je Q literal u kojem se pojavljuje propozicionalna varijabla P_i (tj. $Q \equiv P_i$ ili $Q \equiv \neg P_i$). Tada definiramo $q_Q \in \mathbb{Z}[X]$ ovako:

$$q_Q(x_i) = \begin{cases} x_i, & \text{ako je } Q \equiv P_i; \\ 1 - x_i, & \text{ako je } Q \equiv \neg P_i. \end{cases}$$

Radi jednostavnijeg zapisa, u daljnjem tekstu pretpostavljamo da se u literalu Q_i pojavljuje propozicionalna varijabla P_i. Neka je $C \equiv Q_1 \vee Q_2 \vee Q_3$ elementarna disjunkcija. Budući da je $C \Leftrightarrow \neg(\neg Q_1 \wedge \neg Q_2 \wedge \neg Q_3)$, tada definiramo:

$$q_C(x_1, x_2, x_3) = 1 - \prod_{i=1}^{3}(1 - q_{Q_i}(x_i)).$$

Za svaku interpretaciju I i varijablu P_i uvodimo oznaku $p_i = I(P_i)$.

Pomoćna tvrdnja. Neka je $C \equiv Q_1 \vee Q_2 \vee Q_3$ elementarna disjunkcija. Tada za svaku interpretaciju I vrijedi $I(C) = q_C(p_1, p_2, p_3)$.
Dokaz pomoćne tvrdnje. Pretpostavimo prvo da je I interpretacija takva da je $I(C) = 1$. Bez smanjenja općenitosti možemo pretpostaviti da je $I(Q_1) = 1$. Promatramo dva slučaja: $Q_1 \equiv P_1$ i $Q_1 \equiv \neg P_1$. Ako je $Q_1 \equiv P_1$ tada je $p_1 = I(P_1) = I(Q_1) = 1$, te je $q_{Q_1}(x_1) = x_1$. Zatim, vrijedi:

$$q_C(x_1, x_2, x_3) = 1 - \prod_{i=1}^{3}(1 - q_{Q_i}(x_i)) = 1 - (1 - x_1) \cdot \prod_{i=2}^{3}(1 - q_{Q_i}(x_i)).$$

Tada je

$$q_C(p_1, p_2, p_3) = q_C(1, p_2, p_3) = 1 - (1 - 1) \cdot \prod_{i=2}^{3}(1 - q_{Q_i}(p_i) = 1 = I(C).$$

Ako je $Q_1 \equiv \neg P_1$ tada je $p_1 = I(P_1) = 1 - I(Q_1) = 1 - 1 = 0$, te je $q_{Q_1}(x_1) = 1 - x_1$. Time imamo:

$$\begin{aligned} q_C(x_1, x_2, x_3) &= 1 - \prod_{i=1}^{3}(1 - q_{Q_i}(x_i)) = 1 - [1 - (1 - x_1)] \cdot \prod_{i=2}^{3}(1 - q_{Q_i}(x_i)) \\ &= 1 - x_1 \cdot \prod_{i=2}^{3}(1 - q_{Q_i}(x_i)). \end{aligned}$$

Tada je

$$q_C(p_1, p_2, p_3) = q_C(0, p_2, p_3) = 1 - 0 \cdot \prod_{i=2}^{3}(1 - q_{Q_i}(p_i) = 1 = I(C).$$

Promotrimo sada slučaj kada je $I(C) = 0$. Tada je $I(Q_1) = I(Q_2) = I(Q_3) = 0$. Neka je $A \subseteq \{1, 2, 3\}$ takav da za svaki $i \in A$ vrijedi $Q_i \equiv P_i$. Slično, neka je $B \subseteq \{1, 2, 3\}$ takav da za svaki $j \in B$ vrijedi $Q_j \equiv \neg P_j$ (možda je neki od skupova A ili B prazan). Tada očito vrijedi:

$$q_C(x_1, x_2, x_3) = 1 - \prod_{i \in A}(1 - x_i) \cdot \prod_{i \in B} x_j.$$

Budući da je $p_i = 0$ za svaki $i \in A$, te je $p_j = 1$ za svaki $j \in B$, tada je

$$q_C(p_1, p_2, p_3) = 1 - \prod_{i \in A}(1 - 0) \cdot \prod_{j \in B} 1 = 1 - 1 = 0 = I(C).$$

Time je pomoćna tvrdnja potpuno dokazana.

Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$ neka 3–knf, te $Var(F) = \{P_1, \ldots, P_n\}$. Tada definiramo $q_F \in \mathbb{Z}[X_1, \ldots, X_n]$ ovako: $q_F = q_{C_1} \cdot q_{C_2} \cdot \ldots \cdot q_{C_s}$. Iz dokazane pomoćne tvrdnje lako slijedi da za sve interpretacije I vrijedi: $I(F) = q_F(p_1, \ldots, p_n)$.

Ovo posljednje povlači da ako je F ispunjiva 3–knf tada polinom $1 - q_F$ ima barem jednu cjelobrojnu nultočku. No, obrat ne mora vrijediti. Pokažimo to primjerom. Neka je $F \equiv (P_1 \vee P_1 \vee P_1) \wedge (\neg P_1 \vee \neg P_1 \vee \neg P_1) \wedge (P_2 \vee P_2 \vee P_2)$. Očito je F antitautologija, tj. nije ispunjiva. No, budući da očito vrijedi

$$q_F(x_1, x_2) = 1 - [1 - (1 - x_1)^3] \cdot [1 - x_1^3] \cdot [1 - (1 - x_2)^3],$$

tada polinom $1 - q_F$ ima cjelobrojne nultočke koje čak ne pripadaju skupu $\{0, 1\}^2$ (primjerice $(0, 2)$ i $(1, -33)$ su cjelobrojne nultočke polinoma $1 - q_F$). Kako bi vrijedila i nužnost definiramo prvo polinom $r \in \mathbb{Z}[X_1, \ldots, X_n]$ ovako:

$$r(x_1, \ldots, x_n) = \sum_{i=1}^{n}(x_i(1 - x_i))^2.$$

Očito za svaki $\vec{p} \in \{0, 1\}^n$ vrijedi $r(\vec{p}) = 0$, a za svaki $\vec{y} \in \mathbb{Z}^n \setminus \{0, 1\}^n$ vrijedi $r(\vec{y}) > 0$. Sada konačno definiramo polinom $p_F \in \mathbb{Z}[X_1, \ldots, X_n]$ sa: $p_F = (1 - q_F)^2 + r$.

Ako je F ispunjiva tada znamo da polinom $1 - q_F$ ima cjelobrojnu nultočku $\vec{p} \in \{0,1\}^n$. Istu nultočku ima i polinom $(1 - q_F)^2$. Budući da je $r(\vec{p}) = 0$, za sve $\vec{p} \in \{0,1\}^n$, tada je očito $p_F(\vec{p}) = 0$. Ako je F antitautologija tada za sve $\vec{p} \in \{0,1\}^n$ vrijedi $(1 - q_F)(\vec{p}) = 1$, te $r(\vec{p}) = 0$, pa je $p_F(\vec{p}) \neq 0$. Za $\vec{y} \in \mathbb{Z}^n \setminus \{0,1\}^n$ očito vrijedi $(1 - q_F)^2(\vec{y}) \geq 0$, te $r(\vec{y}) > 0$, pa je opet $p_F(\vec{y}) \neq 0$.

Ostaje još pokazati da je konstrukcija polinoma p_F vremenski izvediva u polinomnom broju koraka u ovisnosti o duljini formule F. Lako se vidi da s jednim prolaskom kroz formulu F možemo dobiti polinom q_F. Pri prolasku kroz formulu kako čitamo varijable, odmah spremamo one različite koje onda na kraju koristimo za dopisivanje r člana u polinomu p_F. Q.E.D.

Iz prethodnog teorema i NP–potpunosti problema 3–SAT slijedi tvrdnja sljedećeg korolara.

Korolar 4.4. *Jezik* $Dif = \{p : p \in \mathbb{Z}[X_1, \ldots, X_n]$ *koji ima cjelobrojnu nultočku*$\}$ *je NP–težak.*

Poglavlje 5

Prostorna složenost

Do sada smo razmatrali vremensku složenost algoritama. Za efikasnost algoritma svakako je važno koliko memorijskog prostora je potrebno za izvršenje algoritma. Nakon navođenja osnovnih definicija dokazat ćemo najvažniji teorem u vezi prostorne složenosti. To je Savitchev teorem.

5.1 Osnovne definicije i činjenice

U ovoj uvodnoj točki dajemo osnovne definicije, te ističemo teorem o linearnoj kompresiji prostora i prostornoj hijerahiji. Ako drugačije ne istaknemo, Turingovi strojeve koje razmatramo su jednotračni.

Definicija 5.1. *Neka je T neki deterministički Turingov stroj koji staje za svaki ulaz.* **Prostorna složenost determinističkog** *Turingovog stroja T je funkcija $space_T : \mathbb{N} \to \mathbb{N}$, gdje je $space_T(n)$ maksimalan broj registara na traci po kojima glava stroja T čita, za svaki uzlazni podatak duljine n.*

Neka je N nedeterministički Turingov stroj koji ima svojstvo da za svaki ulaz izračunavanje na svakoj grani stane. **Prostorna složenost nedeterminističkog** *Turingovog stroja N je funkcija $space_N : \mathbb{N} \to \mathbb{N}$, gdje je $space_N(n)$ maksimalan broj registara na traci po kojima glava stroja N čita, za svaki ulazni podatak duljine n.*

Ako je $space_T$ prostorna složenost stroja T (determinističkog ili nedeterminističkog) tada kažemo da stroj T treba prostor $space_T$, tj. da je T jedan $space_T$**–prostorno složen Turingov stroj**. Kažemo još da je stroj T **prostorno polinoman** ako postoji polinom f tako da vrijedi $space_T(n) = O(f(n))$.

Kako bi definirali prostorne klase složenosti moramo prvo definirati prostorne klase za svaku nenegativnu funkciju.

Definicija 5.2. *Za svaku funkciju $f : \mathbb{N} \to \mathbb{R}^+$ definiramo pripadne prostorne klase složenosti ovako:*

DSPACE $(f(n)) = \{L : L$ *je jezik odlučiv na nekom* $O(f(n))$ *prostorno složenim deterministič̆kim Turingovim strojem*$\}$.

NSPACE $(f(n)) = \{L : L$ *je jezik odlučiv na nekom* $O(f(n))$ *prostorno složenim nedeterministič̆kim Turingovim strojem*$\}$.

Definicija 5.3. *Neka je* **PSPACE**$=\bigcup_{k\in\mathbb{N}} \mathrm{DSPACE}(n^k)$, *te neka je* **NPSPACE**$=\bigcup_{k\in\mathbb{N}} \mathrm{NSPACE}(n^k)$.

Primjer 5.4. *U prethodnom poglavlju bili smo dokazali da je problem* `SAT` *jedan* `NP`*–potpun problem. U ovom primjeru želimo naglasiti da problem* `SAT` *može biti riješen algoritmom linearne prostorne složenosti. Označimo s* T *Turingov stroj koji na ulazu ima formulu* F *logike sudova. Stroj* T *generira redom interpretacije s domenom* $Var(F)$, *te određuje* $I(F)$. *Ako za neku interpretaciju* I *vrijedi* $I(F) = 1$, *tada stroj* T *prihvaća formulu* F, *a inače je odbaci. Stroj* T *očito koristi samo linearan prostor jer za svaku interpretaciju propozicionalnih varijabli koristi isti dio trake.*

U poglavlju 2 bili smo naveli teorem o linearnom ubrzanju, tj. teorem 2.2. Sada ističemo njegov prostorni analogon.

Teorem 5.5. (O linearnoj kompresiji prostora)
Neka je L *proizvoljan odlučiv jezik. Za svaki Turingov stroj* T *koji odlučuje jezik* L *i svaki* $m \in \mathbb{N} \setminus \{0\}$ *postoji Turingov stroj* S *koji također odlučuje jezik* L, *te za svaki* $n \in \mathbb{N}$ *vrijedi:*

$$space_S(n) \leq \frac{space_T(n)}{m} \; + \; n.$$

Teorem o prostornoj hijerahiji govori da ako koristimo više registara tada ćemo moći odlučiti više jezika. Kako bi mogli točno formulirati teorem prvo moramo definirati jednu posebnu vrstu funkcija.

Definicija 5.6. *Neka je* $f : \mathbb{N} \to \mathbb{N}$ *funkcija tako da je za svaki* n *(dovoljno velik!) ispunjeno da je* $f(n)$ *barem* $O(\log_2 n)$. *Kažemo da je funkcija* f **prostorno konstruktibilna** *ako je funkcija koja riječ* 1^n *u unarnom zapisu preslikava u binarni zapis broja* $f(n)$ *Turing–izračunljiva funkcija prostorne složenosti* $O(f(n))$.

Funkcije $\log_2 n$, $n \log_2 n$ i n^2 su prostorno konstruktibilne funkcije.

Teorem 5.7. (O prostornoj hijerarhiji)
Za svaku prostorno konstruktibilnu funkciju $f : \mathbb{N} \to \mathbb{N}$ *postoji jezik* L *koji je odlučiv na nekom Turingovom stroju prostorne složenosti* $O(f(n))$, *ali jezik* L *nije* $o(f(n))$*-prostorno odlučiv.*

Dokaz prethodnog teorema možete vidjeti u [23].

Korolar 5.8. *Za svake dvije funkcije $f, g : \mathbb{N} \to \mathbb{N}$ takve da imamo $f(n) = o(g(n))$, te je funkcija g prostorno konstruktibilna, vrijedi* $\mathrm{DSPACE}(f(n)) \subsetneqq \mathrm{DSPACE}(g(n))$.

Napomena 5.9. *Ako je $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija tada je za svaki $n \in \mathbb{N}$ vrijedi da je $f(n)$ barem $O(n \log_2 n)$, te je funkcija koja 1^n preslikava u binarni zapis od $f(n)$ Turing–izračunljiva u vremenu $O(f(n))$. No, tada očito za svaki $n \in \mathbb{N}$ vrijedi da je $f(n)$ barem $O(\log_2 n)$, te je funkcija koja 1^n preslikava u binarni zapis od $f(n)$ Turing–izračunljiva u prostoru $O(f(n))$. Dakle, svaka vremenski konstruktibilna funkcija je prostorno konstruktibilna.*

Budući da je za svaki $k \in \mathbb{N} \setminus \{0\}$ funkcija $n \mapsto n^k$ vremenski konstruktibilna (bili smo naveli da je čak potpuno vremenski konstruktibilna), tada je ona i prostorno konstruktibilna. Iz prethodnog korolara posebno slijedi $\mathrm{DSPACE}(n^k) \subsetneqq \mathrm{DSPACE}(n^{k+1})$.

Označimo $\texttt{EXPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{DSPACE}(2^{n^k})$.

Korolar 5.10. *Vrijedi: $\texttt{PSPACE} \subsetneqq \texttt{EXPSPACE}$.*

Dokaz. Očito za svaki $k \in \mathbb{N}$ vrijedi $\mathrm{DSPACE}(n^k) \subseteq \mathrm{DSPACE}(2^n)$. Iz toga slijedi: $\texttt{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{DSPACE}(n^k) \subseteq \mathrm{DSPACE}(2^n)$.

Funkcija $n \mapsto 2^{n^2}$ je potpuno vremenski konstruktibilna, a onda je i prostorno konstruktibilna. Očito vrijedi $\lim_{n \to \infty} \frac{2^n}{2^{n^2}} = 0$. Iz teorema o prostornoj hijerahiji slijedi $\mathrm{DSPACE}(2^n) \subsetneqq \mathrm{DSPACE}(2^{n^2})$. Time imamo:

$$\texttt{PSPACE} \subseteq \mathrm{DSPACE}(2^n) \subsetneqq \mathrm{DSPACE}(2^{n^2}) \subseteq \texttt{EXPSPACE}.$$

Q.E.D.

5.2 Savitchev teorem

Prije iskaza i dokaza Savitchevog teorema moramo uvesti još neke pojmove, te navesti neke činjenice. Prilikom dokaza Cook–Levinovog teorema već smo bili govorili o pojmu konfiguracije Turingovog stroja. Ovdje ćemo ponoviti neke pojmove i oznake u vezi s time, te navesti neke nove.

Konfiguracija Turingovog stroja $T = (\Gamma, Q, \delta)$ je određena s trenutnim stanjem, sadržajem trake, te pozicijom glave na traci. Ako je stroj u stanju q, na traci je zapisana riječ oblika uv i glava je pozicionirana na prvi znak riječi v, tada konfiguraciju zapisujemo kao uqv. Kažemo da iz konfiguracije C_1 proizlazi konfiguracija C_2 ako Turingov stroj može iz C_1 po pravilima definiranim u funkciji prijelaza δ doći u konfiguraciju C_2 u jednom koraku.

Početna konfiguracija za Turingov stroj T s ulazanom riječi w je riječ $q_0 w$. Prihvaćajuća konfiguracija je konfiguracija sa stanjem q_{DA}, a odbijajuća konfiguracija je konfiguracija sa stanjem q_{NE}.

Lema 5.11. *Neka je $f : \mathbb{N} \to \mathbb{R}^+$ funkcija takva da je $f(n) \geq n$, za svaki $n \in \mathbb{N}$. Neka je T Turingov stroj prostorne složenosti $f(n)$. Tada T ima maksimalno $2^{O(f(n))}$ različitih konfiguracija.*

Dokaz. Neka je $T = (\Gamma, Q, \delta)$ i neka je $|Q| = q$ i $|\Gamma| = g$. Tada imamo $q \cdot f(n) \cdot g^{f(n)}$ različitih konfiguracija, jer imamo q različitih stanja, glava stroja se može nalaziti na $f(n)$ mogućih mjesta i $g^{f(n)}$ je broj svih mogućih riječi koje mogu biti zapisane na traci kada se koristi $f(n)$ različitih registara. Očito vrijedi

$$q \cdot f(n) \cdot g^{f(n)} = 2^{log_2(q \cdot f(n) \cdot g^{f(n)})} = 2^{\log_2 q + \log_2 f(n) + f(n) \cdot \log_2 g}.$$

Sada procjenjujemo izraz u eksponentu. Vrijedi:

$$\log_2 q + \log_2 f(n) + f(n) \cdot \log_2 g = O(1) + O(\log_2 f(n)) + O(f(n)) = O(f(n))$$

(Uočimo da za zadnju navedenu jednakost koristimo da $f(n) \geq n$ povlači $\log_2 f(n) = O(f(n))$.) Dakle, $2^{O(f(n))}$ je asimptotska gornja međa za broj konfiguracija stroja T. Q.E.D.

Prilikom izvršavanja, stroj T iz leme, u najgorem slučaju, prođe kroz sve moguće konfiguracije i to kroz svaku samo jednom (inače bi stroj ušao u petlju i radio vječno) pa je maksimalno vrijeme izvršavanja jednako maksimalnom broju konfiguracija, tj. $2^{O(f(n))}$.

Teorem 5.12. (W. Savitch, 1970.) *Ako je $f : \mathbb{N} \to \mathbb{N}$ takva da za svaki $n \in \mathbb{N}$ vrijedi $f(n) \geq n$, tada imamo:*

$$\text{NSPACE}(f(n)) \subseteq \text{DSPACE}(f^2(n)).$$

Dokaz. Neka je $L \in \text{NSPACE}(f(n))$, te neka je N nedeterministički Turingov stroj prostorne složenosti $f(n)$ koji odlučuje jezik L. Konstruirat ćemo deterministički Turingov stroj T koji odlučuje isti jezik. Cilj nam je simulirati stroj N pomoću determinističkog stroja koristeći što manje prostora, ne obazirući se na utrošeno vrijeme. Ako bi pokušali simulirati sve grane izračunavanja, ne pazeći pri tome da koristimo isti registar više puta, trebalo bi nam $2^{O(f(n))}$ registara. Umjesto simuliranja svih grana izračunavanja stroja N, stroj T će rješavati problem dostiživosti koji glasi:

> Neka su dane dvije konfiguracije C_1 i C_2 nedeterminističkog stroja N i broj $t \in \mathbb{N}$. Treba odrediti da li stroj N može doći iz konfiguracije C_1 do konfiguracije C_2 u najviše t koraka.

Definiramo rekurzivnu proceduru DOSEG koja na ulazu ima tri parametra: dvije konfiguracije C_1 i C_2 nedeterminističkog stroja N i prirodan broj t. Procedura prihvaća ulaz ako je konfiguracija C_2 dostiživa iz konfiguracije C_1 u najviše t koraka. U suprotnom procedura odbija ulaz. Sada dajemo pseudokod procedure DOSEG.

DOSEG = Na ulazu su konfiguracije C_1, C_2 i prirodan broj t :

1. Ako je $t = 1$
   - Ako je $C_1 = C_2$, prihvati.
   - Ako iz C_1 proizlazi C_2, prihvati.
2. Ako je $t > 1$ tada za svaku konfiguraciju C_i stroja N :
   a) izvedi proceduru DOSEG s ulazom $(C_1, C_i, \lceil t/2 \rceil)$;
   b) izvedi proceduru DOSEG s ulazom $(C_i, C_2, \lceil t/2 \rceil)$;
   c) ako su oba koraka a) i b) završila s prihvaćanjem, tada ova procedura također prihvaća početni ulaz.
3. Ako je procedura došla do ovog koraka, tada procedura odbija ulaz.

Sada možemo definirati deterministički stroj T. Prije toga malo modificiramo stroj N. Ako stroj N prihvati neki ulaz w, tada neka prije zaustavljanja potpuno očisti traku. Tu konfiguraciju označavamo sa $C_{prihvat}$. Neka je sa $C_{start}(w)$ označena početna konfiguracija. Iz prethodne leme znamo da je broj svih različitih konfiguracija stroja N omeđen sa $2^{O(f(n))}$, ako je na ulazu riječ w duljine n. To znači da postoji konstanta $d > 0$ takva da stroj N za bilo koji ulaz duljine n nema više od $2^{d \cdot f(n)}$ konfiguracija. Iz toga slijedi da ako stroj N prihvaća ulaz, tada do prihvaćajuće konfiguracije može doći za manje od $2^{d \cdot f(n)}$ koraka.

Deterministički stroj T s ulazom w, pri čemu je $|w| = n$, definiramo kao implementaciju procedure DOSEG s ulazom $(C_{start}(w), C_{prihvat}, 2^{d \cdot f(n)})$.

Primijetimo da ako stroj N prihvaća riječ w tada postoji put od startne konfiguracije do konfiguracije $C_{prihvat}$ koji traje manje od $2^{d \cdot f(n)}$ koraka. U tom slučaju procedura DOSEG u stroju T će prihvatiti riječ w, tj. stroj T prihvaća ulaz w. Dakle, stroj T korektno simulira rad stroja N (točnije, vrijedi: $L = L(N) = L(T)$).

Analizirajmo prostornu složenost stroja T. Neka je $b \geq 2$ neka baza u kojoj prikazujemo brojeve u stroju T. Zatim, neka je alfabet Γ stroja T jednak $\{0, 1, \ldots, b-1\}$.

Rad stroja T se sastoji zapravo od jednog poziva rekurzivne procedure DOSEG. Prostornu složenost procedure DOSEG analiziramo u tri koraka.

a) Dokažimo prvo da svaki nivo rekurzije koristi najviše $O(f(n))$ registara.
   Jedan nivo rekurzije procedure DOSEG koristi prostor za zapis konfiguracija C_1 i C_2, te broja t. Svaka konfiguracija zauzima maksimalno $O(f(n))$ registara.

Veličina zapisa broja t je broj njegovih znamenki u prikazu u bazi b, a to je jednako $\lceil \log_b t \rceil + 1$. Izbor baze ne igra ulogu jer se duljine prikaza za različite baze razlikuju za konstantni faktor, pa možemo uzeti da je prostor potreban za zapis broja t na stroju T omeđen sa $O(\log_2 t)$. Najveći broj koji stroj T koristi (za ulaz duljine n) je $2^{d \cdot f(n)}$ za koji treba $O(\log_2 2^{d \cdot f(n)}) = O(f(n))$ prostora. Ukupno, jedan nivo rekurzije koristi najviše $O(f(n)) + O(f(n)) + O(f(n)) = O(f(n))$ registara.

b) Dokažimo sada da je broj nivoa rekurzije u stroju T za bilo koji ulaz w duljine n jednak $O(f(n))$. Prvi poziv rekurzivne procedure DOSEG je s brojem $t = 2^{d \cdot f(n)}$. Pri svakom sljedećem prelasku na novi nivo broj t postaje za pola manji. Na novi nivo prelazimo sve dok ne bude $t = 1$. Dakle, ukupno će biti najviše $\log_2 2^{d \cdot f(n)} = d \cdot f(n) = O(f(n))$ nivoa.

c) Uzimajući u obzir a) i b) zaključujemo da je prostorna složenost procedure DOSEG jednaka $O(f(n)) \cdot O(f(n))$, tj. $O(f^2(n))$.

Još jedna stvar koju moramo napomenuti je da bi stroj T trebao znati vrijednosti funkcije f, tj. trebao bi znati $f(n)$. Umjesto nekih dodatnih pretpostavki u teoremu, uvodimo malu modifikaciju u definiciji stroja T. Stroj T će uzimati za $f(n)$ redom vrijednosti $n, n+1, n+2, \ldots$ (počinjemo od n jer je po pretpostavci teorema $f(n) \geq n$, a završavamo kada više nema konfiguracija čija je duljina veća od trenutne vrijednosti $f(n)$). Za svaku vrijednost $f(n) = i$, osim provjeravanja dostiživosti neke prihvaćajuće konfiguracije, stroj T treba još provjeriti je li od startne konfiguracije dostiživa neka konfiguracija čija je veličina barem $i + 1$ (pretpostavili smo da je duljina jedne konfiguracije $f(n)$). Ukratko, modificirani stroj T prihvaća ulaznu riječ ako je dostiživa prihvaćajuća konfiguracija. Zatim, stroj odbija ulaznu riječ ako postoji konfiguracija C čija je duljina barem $i+1$, a C nije dostiživa iz startne konfiguracije. Inače, stroj nastavlja rad, pri čemu uzima $f(n) = i + 1$. Q.E.D.

Korolar 5.13. *Vrijedi:* `PSPACE`=`NPSPACE`.

5.3 Veza vremenskih i prostornih klasa složenosti

Ako neki deterministički Turingov stroj s ulazom duljine n napravi $f(n)$ koraka, tada njegova glava za čitanje ne može pristupiti više od $f(n)$ registara. Iz te jednostavne činjenice slijedi tvrdnja sljedeće propozicije.

Propozicija 5.14. *Neka je $f : \mathbb{N} \to \mathbb{R}$ proizvoljna funkcija, tako da za svaki $n \in \mathbb{N}$ vrijedi $f(n) \geq n$. Tada vrijedi* $\mathrm{DTIME}(f(n)) \subseteq \mathrm{DSPACE}(f(n))$.

Korolar 5.15. *Vrijedi* `PTIME` $\subseteq$ `PSPACE`.

Dokaz. Neka je $L \in \mathrm{PTIME}$ proizvoljan jezik. Tada postoji $k \in \mathbb{N}$ tako da je $L \in \mathrm{DTIME}(n^k)$. Iz prethodne propozicije slijedi $L \in \mathrm{DSPACE}(n^k)$. Q.E.D.

Teorem 5.16. *Neka je $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija. Tada vrijedi:* $\mathrm{NTIME}(f(n)) \subseteq \mathrm{DSPACE}(f(n))$.

Dokaz. Neka je $L \in \mathrm{NTIME}(f(n))$ proizvoljan jezik. Neka je N neki nedeterministički Turingov stroj koji odlučuje jezik L u vremenu $f(n)$. Tada najduža grana izračunavanja stroja N ima duljinu najviše $f(n)$. Ideja dokaza je simulirati izračunavanje redom po svakoj grani, koristeći pri tome onoliko prostora koliko treba kod izračunavanja na najdužoj grani.

Neka je Γ_N alfabet a Q_N je skup stanja stroja N. Fiksirajmo neke linearne uređaje na skupovima Q_N, Γ_N i $\{L, D, S\}$. Tada je na skupu $Q_n \times \Gamma_N \times Q_N \times \Gamma_N \times \{L, D, S\}$ definiran leksikografski uređaj. Uočite da je svaka konfiguracija (q, s, q', s', I) stroja N element upravo navedenog skupa. Neka je $k = |Q_N \times \Gamma_N \times Q_N \times \Gamma_N \times \{L, D, S\}|$.

Opisat ćemo dvotračni deterministički Turingov stroj T koji će simulirati rad stroja N, te čija će prostorna složenosti biti $O(f(n))$. Alfabet stroja Γ_T neka sadrži alfabet Γ_N, te skup novih simbola $\{s_1, \ldots, s_k\}$. Smatramo da je svakom simbolu s_i pridružen i–ti član skup $Q_n \times \Gamma_N \times Q_N \times \Gamma_N \times \{L, D, S\}$.

Prva traka stroja T će biti radna, a druga će služiti za zapisivanje niza simbola s_i. Tu drugu traku ćemo nazivati adresna traka.

Neka na ulazu stroj T ima neku riječ $w = w_1 \ldots w_n$. Budući da je po pretpostavci funkcija f vremenski konstruktibilna, stroj T prvo na drugoj traci izračuna $f(|w|)$ uočite da stroj T za računanje $f(|w|)$ ne treba više od $O(f(|w|))$ prostora). Nakon toga se označi početak i kraj riječi $f(|w|)$, a samu riječ $f(|w|)$ obriše.

Stroj T na drugoj traci redom (obzirom na leksikografski uređaj) generira riječi alfabeta $\{s_1, \ldots, s_k\}$ čija duljina nije veća od $f(|w|)$ (za to koristi oznake na adresnoj traci kreirane pomoću riječi $f(|w|)$). Za svaku riječ na adresnoj traci stroj T provjerava radi li se o nizu simbola koji predstavljaju neku granu izračunavanja stroja N, te istovremeno simulira na radnoj traci prijelaz iz jedne u drugu konfiguraciju stroja N. Opisat ćemo to malo detaljnije. Pretpostavimo da je u nekom trenutku rada stroja T na adresnoj traci zapisana riječ $s_{i_1} \ldots s_{i_j}$, a na radnoj straci je zapisana samo riječ w. Prvo se provjerava je li s_{i_1} simbol neke konfiguracije oblika (q_0, w_1, q', s', I'), gdje smo sa q_0 označili početno stanje stroja N. Ako se ne radi o konfiguaciji tog oblika, tada se na adresnoj traci generira sljedeća riječ u odnosu na leksikografski uređaj. Ako s_{i_1} jeste simbol neke "početne" konfiguracije, tada se prvo provjerava je li s_{i_2} simbol koji označava konfiguraciju oblika (q', s', q'', s'', I''). Ako s_{i_2} nije oznaka takve konfiguracije, tada se na adresnu traku generira sljedeća riječ. Ako s_{i_2} jeste simbol neke takve konfiguracije, tada stroj T na radnoj traci simulira korak stroja N prilikom prijelaza iz (q_0, w_1) u (q', s', I'). Analogno dalje.

Budući da po pretpostavci stroj N odlučuje jezik L tada za svaki ulaz, a onda i za w, sve grane izračunavanja od N staju u konačno mnogo koraka. Ako neka grana izračunavanja stroja N prihvaća riječ w, tada definiramo da i stroj T prilikom simulacije te grane staje, i prihvaća riječ w. Ako niti jedna grana od N ne prihvaća riječ w, tada definiramo da stroj T staje i odbija riječ w.

Rezimirajmo: upravo definirani deterministički stroj T redom isprobava sve moguće grane izračunavanja stroja N. Pri tome stroj T koristi onoliko prostora koliko i

N koristi najviše za jednu granu izračunavanja (što je najviše $f(n)$). No, još moramo uračunati prostor upotrijebljen na adresnoj traci koji iznosi $O(f(n))$ jer duljina jedne grane izračunavanja stroja N je najviše $f(n)$.

Dakle, prostorna složenost deterministčkog stroja T je $f(n) + f(n) = O(f(n))$. (Primijetimo da smo definirali prostornu složenost za jednotračne Turingove strojeve, a ovdje računamo prostornu složenost dvotračnog stroja.) Q.E.D.

Korolar 5.17. *Vrijedi:* $\mathtt{NPTIME} \subseteq \mathtt{PSPACE}$.

Napomena 5.18. *Označimo s* NPC *klasu svih* NP*–potpunih problema. Ako pretpostavimo da vrijedi* $\mathtt{P} \neq \mathtt{NP}$*, tada očito* $\mathtt{NPC} \subseteq \mathtt{NP} \setminus \mathtt{P}$*, te je dokazano* $\mathtt{NPC} \neq \mathtt{NP} \setminus \mathtt{P}$*. Označimo* $\mathtt{NPI} = [\mathtt{NP} \setminus \mathtt{P}] \setminus \mathtt{NPC}$ *(eng. intermediate in difficulty). Dakle, uz pretpostavku* $\mathtt{P} \neq \mathtt{NP}$ *slijedi* $\mathtt{NPI} \neq \emptyset$*. No, nije poznat niti jedan problem koji pripada klasi* NPI*. Vjeruje se da sljedeća tri problema pripadaju klasi* NPI*:*

PROBLEM IZOMORFIZMA GRAFOVA*: Neka su* $G = (V, E)$ *i* $G' = (V', E')$ *grafovi. Jesu li* G *i* G' *izomorfni?*

PROBLEM ODREĐIVANJA SLOŽENOSTI BROJA*: Neka je* $n \in \mathbb{N}$*. Postoje li* $j, k \in \mathbb{N} \setminus \{0, 1\}$ *takvi da je* $n = j \cdot k$*?*

PROBLEM LINEARNOG PROGRAMIRANJA*: Neka je* $\{\vec{v}_i : i = 1, \ldots, m\}$ *skup vektora. Neka je* $D = \{d_1, \ldots, d_m\}$ *i* $C = \{c_1, \ldots, c_n\}$*, te neka je* $B \in \mathbb{N}$*. Postoji li vektor* $X = (x_1, \ldots, x_n)$ *takav da je* $\vec{v}_i \cdot X \leq d_i$*, za svaki* $i = 1, \ldots, m$*, te je* $C \cdot X \geq B$*?*
Više detalja o klasi NPI *možete pronaći u [17].*

5.4 PSPACE–potpunost

U prethodnim razmatranjima uvjerili smo se koliko je važan pojam NP–potpunosti. Sada definiramo njegov "prostorni" analogon.

Definicija 5.19. *Za jezik* L *kažemo da je* **PSPACE–potpun** *ako vrijedi:*

a) jezik L *pripada klasi* PSPACE*;*

b) svaki jezik $L' \in \mathtt{PSPACE}$ *je vremenski polinomno reducibilan na jezik* L*.*

Napomena 5.20. *Želimo naglasiti da se u prethodnoj definiciji traži vremenska reducibilnost, a ne prostorna. Pokušat ćemo objasniti što je tome razlog. Potpuni problemi su važni jer su primjeri najtežih problema iz klase složenosti koja se promatra. Potpuni problemi imaju svojstvo da ako nađemo efikasan način rješavanja potpunog problema, tada možemo efikasno rješavati sve probleme iz klase. Redukcija svakako mora biti laka, i to laka u odnosu na složenost tipičnih problema iz klase. Ako bi redukcija bila teška, tada efikasno rješavanje potpunog problema neće inducirati efektivno lako rješavanje svih problema iz klase složenosti. Možemo formulirati*

sljedeće pravilo za definiranje potpunih problema: redukcija mora biti više limitirana nego što je model koji se koristi za definiciju klase složenosti.

Ako bi u definiciji PSPACE–potpunog problema zahtijevali prostornu polinomnu složenost, tada ne bi znali je li i vremenska složenost nužno polinomna. No, budući da u definiciji zahtijevamo vremensku polinomnu složenost, tada će očito i prostorna složenost bili polinomna.

Kako bi mogli navesti jedan primjer PSPACE–potpunog jezika, prvo moramo još navesti neke pojmove. Alfabet kvantifikacijske propozionalne logike TQBF osim alfabeta klasične logike sudova sadrži i kvantifikatore $\forall$ i $\exists$. Formule logike TQBF definiramo rekurzivno ovako:

a) svaka propozicionalna varijabla je formula;

b) ako su F i G formule tada su i riječi $\neg F$, $F \wedge G$, $F \vee G$, $F \to G$ i $F \leftrightarrow G$ također formule;

c) ako je F formula i p propozicionalna varijabla tada su i riječi $\forall pF$ i $\exists pF$ formule.

Formula $\forall pF$ je očito ekvivalentna formuli $F(\bot/p) \wedge F(\top/p)$, a formula $\exists pF$ je ekvivalentna formuli $F(\bot/p) \vee F(\top/p)$. To znači da se logika TQBF može svesti na klasičnu logiku. No, očito je da za tu interpretaciju treba eksponencijalno vrijeme. Za neku TQBF–formulu kažemo da je zatvorena ako je svaka njena propozicionalna varijabla u dosegu nekog kvantifikatora.

Teorem 5.21. (L. Stockmeyer, 1974.) *Problem određivanja istinitosti zatvorenih formula logike TQBF je PSPACE–potpun.*

Dokaz prethodnog teorema možete pronaći u [23].

5.5 Klase L i NL

Do sada smo razmatrali vremenske i prostorne granice koje su najmanje linearne, tj. zahtijevali smo da funkcije složenosti f imaju svojstvo $f(n) \geq n$. Sada razmatramo tzv. sublinearne međe, tj. funkcije složenosti za koje ne vrijedi nužno $f(n) \geq n$.

Za razmatranje vremenske složenosti sublinearne međe nisu nikako dobre, jer tada nismo osigurali niti da stroj pročita čitavu ulaznu riječ. No, možemo razmatrati sublineanu prostornu složenost, ako modificiramo definiciju stroja kojeg koristimo, te onda i promijenimo definiciju prostorne složenosti. Dozvolit ćemo da stroj može pročitati čitav ulaz, ali to nećemo brojati pod prostornu složenost. Važno će nam samo biti što stroj radi na radnoj traci na kojoj nisu ulazni podaci.

Prilikom razmatranja sublinearnih prostornih klasa složenosti razmatramo Turingove strojeve koji imaju dvije trake. Jedna traka služi samo da se na njoj upiše ulazna riječ, te ne dozvoljavamo da se po toj traci piše. Dakle, na prvoj traci podaci se mogu

samo čitati. Na drugoj traci, koju još nazivamo radna traka, moguće je čitanje i pisanje podataka. Prilikom definicije prostorne složenosti brojimo samo registre na radnoj traci. Sada možemo definirati sublinearne klase složenosti.

Definicija 5.22.

$$\begin{aligned} \mathtt{L} &= \mathrm{DSPACE}(\log n) \\ \mathtt{NL} &= \mathrm{NSPACE}(\log n) \end{aligned}$$

Nije teško pokazati da vrijedi $\mathtt{NL} \subsetneqq \mathtt{PSPACE}$. Štoviše, poznato je da vrijedi i $\mathtt{NL} \subseteq \mathtt{P}$. Istaknimo ovdje sve veze između klasa složenosti koje smo mi proučavali, a koje su danas poznate.

$$\mathtt{NL} \subseteq \mathtt{P} \subseteq \mathtt{NP} \subseteq \mathtt{PSPACE}{=}\mathtt{NPSPACE} \subseteq \mathtt{EXPTIME}.$$

Budući da znamo da vrijedi $\mathtt{P} \subsetneqq \mathtt{EXPTIME}$, tada je barem jedna od gornjih inkluzija prava, ali se ne zna koja.

5.6 Komplementi klasa

Za jezik $L \subseteq \Gamma^*$ označimo sa $\overline{L}$ njegov komplement $\Gamma^* \setminus L$. Za svaku klasu složenosti $\mathcal{C}$ možemo definirati klasu co–$\mathcal{C}$ kao $\{\overline{L} \mid L \in \mathcal{C}\}$.

Lako se vidi da za sve klase složenosti $\mathcal{C}$ definirane deterističkim Turingovim strojevima vrijedi $\mathcal{C} =$ co–$\mathcal{C}$. Primjerice, očito vrijedi P=co–P. No, zbog "asimetije nedeterminizma" ne možemo na isti način izvesti da vrijedi NP=co–NP. Zatim, očito pretpostavka P=NP povlači NP=co–NP. Otvoreni je problem jesu li klase složenosti definirane nedeterminističkim vremenski složenim Turingovim strojevima zatvorene na komplement. Specijalno, pitanje o jednakosti klasa NP i co–NP predstavlja jedno od najpoznatijih otvorenih problema teorije računske složenosti.

Primjer 5.23. *Budući da je* $\mathtt{SAT}^c = \{F : F \text{ je antitautologija}\}$, *tada očito jezik* $\{F : F \text{ je antitautologija}\}$ *pripada klasi co–NP. Označimo* $\mathtt{TAUT} = \{F : F \text{ je tautologija logike sudova}\}$. *Budući da za svaku formulu* F *vrijedi:*

$$F \in \mathtt{TAUT} \text{ ako i samo ako } \neg F \notin \mathtt{SAT},$$

te vrijedi $\mathtt{SAT} \in \mathtt{NP}$, *tada je* $\mathtt{TAUT} \in$ co–NP. *Na sličan način dobivamo da je problem* TAUT *jedan co–NP–potpun problem.*

Teorem 5.24. (N. Immerman; R. Szelepcsényi, 1987.)
Ako je $f(n) \geq \log n$ *tada vrijedi* $\mathrm{NSPACE}(f(n)) =$ co–$\mathrm{NSPACE}(f(n))$. *Štoviše, vrijedi* NL=co–NL.

Dokaz prethodnog teorema možete pronaći u [14], [16], [22] i [23].

Zadaci

1. Dokažite da je klasa PSPACE zatvorena za konačne unije i presjeke, te na operaciju komplementa.

2. Dokažite da za svaki $n \in \mathbb{N}$ vrijedi $\mathrm{DTIME}(n) \subsetneqq$ NPSPACE.
   Rješenje. Iz teorema o vremenskoj hijerahiji znamo da za svaki $k \geq 2$ vrijedi: $\mathrm{DTIME}(n) \subsetneqq \mathrm{DTIME}(n^k)$ (jer je funkcija $n \mapsto n^k$ vremenski konstruktibilna, te vrijedi $\lim_{n\to\infty} n/n^k = 0$). Očito vrijedi $\mathrm{DTIME}(n^k) \subseteq \mathrm{NSPACE}(n^k)$ (ako neki stroj napravi najvise $O(n^k)$ koraka, tada može koristiti najviše $O(n^k)$ registara). Iz svega toga slijedi: $\mathrm{DTIME}(n) \subsetneqq \mathrm{DTIME}(n^k) \subseteq \mathrm{NSPACE}(n^k) \subseteq \bigcup_{j\in\mathbb{N}} \mathrm{NSPACE}(n^j) =$NPSPACE.

3. Dokažite da vrijedi $\mathrm{NTIME}(n) \subsetneqq$ PSPACE.
   Rješenje. Iz teorema o vremenskoj hijerahiiji slijedi da za svaki $k \in \mathbb{N} \setminus \{0, 1\}$ vrijedi $\mathrm{NTIME}(n) \subsetneqq \mathrm{NTIME}(n^k)$ (tu se koristi da je $n = o(n^k)$ i da je funkcija $n \mapsto n^k$ potpuno vremenski konstruktibilna). Očito vrijedi $\mathrm{NTIME}(n^k) \subsetneqq$ NSPACE(n^k) za svaki $k \in \mathbb{N}$. Savitchev teorem povlači da vrijedi $\mathrm{NSPACE}(n^k) \subseteq \mathrm{DSPACE}(n^{2k})$ za svaki $k \in \mathbb{N}$. Rezimirajmo: $\mathrm{NTIME}(n) \subsetneqq \mathrm{NTIME}(n^k) \subseteq \mathrm{NSPACE}(n^k) \subseteq \mathrm{DSPACE}(n^{2k}) \subseteq$ PSPACE.

4. Označimo s K klasu svih jezika koji su odlučivi na nekom dvotračnom determinističkom Turingovom stroju čija je prostorna složenost jednaka $O(\log_2 n)$ (prostornu složenost mjerimo samo na radnoj traci; na jednoj traci se nalaze ulazni podaci, te na njoj ne promatramo prostornu složenost). Dokažite da vrijedi K $\subseteq$ PTIME.
   Rješenje. Ako je T neki deterministički Turingov stroj prostorne složenosti $O(\log_2 n)$, tada je broj svih njegovih konfiguracija najviše $2^{O(\log_2 n)}$ (vidi lemu 5.11.). No, za $f \in 2^{O(\log_2 n)}$ vrijedi $f(n) \leq 2^{c\cdot n}$, za neki $c > 0$. Budući da $2^{c\cdot\log_2 n} = 2^{\log_2 n^c} = n^c$, slijedi da je T polinomne vremenske složenosti.

5. Dokažite teorem o linearnoj kompresiji prostora, tj. teorem 5.5.

6. Neka je $f : \mathbb{N} \to \mathbb{R}^+$ tako da vrijedi $f(n) \geq \log_2 n$ za svaki $n \in \mathbb{N}$. Dokažite da za svaki jezik $L \in \mathrm{DSPACE}(f(n))$ postoji $m \in \mathbb{N}$ tako da vrijedi:

   $$L \in \mathrm{DTIME}\ ((m+1)^{f(n)}).$$

   Uputa. Vidi [7].

7. Dokažite da postoji rekurzivna funkcija f za koju vrijedi:
   $\mathrm{DTIME}(f(n)) =$DSPACE$f(n))$.
   Uputa. Primijenite teorem o praznini i prethodni zadatak.

8. Neka je $f : \mathbb{N} \to \mathbb{N}$ vremenski konstruktibilna funkcija. Dokažite da tada vrijedi:
$$\mathrm{NSPACE}(f(n)) \subseteq \bigcup_{c>0} \mathrm{DTIME}(2^{c \cdot f(n)}).$$

   Uputa. Neka je $L \in \mathrm{NSPACE}(f(n))$, te neka je N neki nedeterministički Turingov stroj koji odlučuje jezik L. Primijetimo da je svaka grana izračunavanja stroja N duljine najviše $f(n)$, ako je duljina ulaza n.

   Neka je w neka riječ koja je ulaz stroja N. Definiramo deterministički Turingov stroj T koji simulira rad stroja N, te je vremenska složenost stroja T jednaka $O(2^{cf(n)})$, za neki $c > 0$. Definiramo usmjereni graf G čiji su čvorovi sve konfiguracije stroja N. (Primijetimo da tu koristimo vremensku konstruktibilnost funkcije f kako bi prilikom generiranja niza konfiguracija imali gornju među.) Iz leme 5.11. znamo da je $|G| = 2^{O(f(n))}$. Zatim, definiramo da između dvije konfiguracije C_1 i C_2 postoji brid ako C_2 proizlazi iz C_1. Očito vrijedi: stroj N prihvaća riječ w ako i samo ako postoji put od početne konfiguracije do neke završne konfiguracije. Primjenom Primovog algoritma (vidi stranu 47; definiramo da je težina svakog brida, primjerice, jednaka 1) slijedi da je vremenska složenost stroja T jednaka $O(|G|^2)$.

9. Dokažite da vrijedi: $\mathtt{NPSPACE} \subseteq \mathtt{EXPTIME}$.

10. Dokažite da iz pretpostavke $\mathtt{PTIME}{=}\mathtt{PSPACE}$ slijedi $\mathtt{PTIME}{=}\mathtt{NPTIME}$.

11. Dokažite da je jezik $\mathtt{TAUT}$ jedan co–$\mathtt{NP}$–potpun jezik.
    Rješenje. Budući da vrijedi: $F \in \mathtt{TAUT}$ ako i samo ako $\neg F \notin \mathtt{SAT}$, tada vrijedi $\mathtt{TAUT} \in$ co–$\mathtt{NP}$ (jer je $\mathtt{SAT} \in \mathtt{NP}$).

    Pokažimo još da je svaki jezik iz klase co–$\mathtt{NP}$ polinomno reducibilan na jezik $\mathtt{TAUT}$. Označimo sa $\mathcal{A}$ neki konačan alfabet klasične logike sudova (umjesto prebrojivog skupa propozicionalnih varijabli u alfabetu imamo, primjerice, dekadske znamenke kako bi mogli kodirati propozicionalne varijable). Neka je $L \in$ co–$\mathtt{NP}$ proizvoljan jezik, i $L \subseteq \Gamma^*$, gdje je Γ proizvoljan konačan alfabet. Tada je $\Gamma^* \setminus L \in \mathtt{NP}$. Budući da je jezik $\mathtt{SAT}$ jedan $\mathtt{NP}$–potpun jezik, tada vrijedi $\Gamma^* \setminus L \leq_p \mathtt{SAT}$. To znači da postoji vremenski polinomno izračunljiva funkcija $f : \Gamma^* \to \mathcal{A}^*$, te takva da za sve riječi $w \in \Gamma^*$ vrijedi: $w \in \Gamma^* \setminus L$ ako i samo ako $f(w) \in \mathtt{SAT}$. Ovo posljednje je ekvivalentno s: $w \in L$ ako i samo ako $\neg f(w) \in \mathtt{TAUT}$.

    Očito je funkcija $g : \Gamma^* \to \mathcal{A}^*$, koja je definirana sa $g(w) = \neg f(w)$, polinomno vremenski izračunljiva, te za sve riječi w vrijedi: $w \in L$ ako i samo ako $g(w) \in \mathtt{TAUT}$. Time smo pokazali da vrijedi $L \leq_p \mathtt{TAUT}$, tj. da je jezik $\mathtt{TAUT}$ jedan co–$\mathtt{NP}$–potpun jezik.

12. Neka je $L \in$ NP neki jezik koji je co–NP potpun. Dokažite da je tada nužno co–NP=NP.
Rješenje. Primijetimo prvo da pošto je jezik L jedan co–NP potpun jezik tada je posebno $L \in$ co–NP.

U dokazu ćemo koristiti sljedeću pomoćnu tvrdnju:

> Neka su L_1 i L_2 jezici i $\mathcal{C}$ neka klasa složenosti ($\mathcal{C}$ je NP ili co–NP.) Ako je jezik L_2 jedan $\mathcal{C}$–potpun jezik, te vrijedi $L_1 \leq_p L_2$ tada vrijedi $L_1 \in \mathcal{C}$.

Dokaz te tvrdnje je sasvim analogan dokazu propozicije 3.11., pa ga ovdje ispuštamo.

Označimo s Γ pripadni alfabet jezika L. Neka je $L_1 \in$ NP proizvoljan jezik nad alfabetom Γ_1. Tada je $L_1^c \in$ co–NP, pa imamo $L_1^c \leq_p L$. To znači da postoji vremenski polinomno izračunljiva funkcija $f : \Gamma_1^* \to \Gamma^*$ tako da za sve riječi $w \in \Gamma_1^*$ vrijedi:

$$w \in L_1^c \text{ ako i samo ako } f(w) \in L.$$

Budući da je T vremenski polinomno izračunljiva tada postoji deterministički Turingov stroj T polinomne vremenske složenosti koji za ulaz $w \in \Gamma_1*$ izračunava $f(w)$. Neka je $w_0 \in \Gamma^* \setminus L$ proizvoljna riječ (takva postoji; inače ...) Definiramo funkciju $g : \Gamma_1^* \to \Gamma^*$ ovako:

$$g(w) = \begin{cases} f(w), & \text{ako je } w \in L_1; \\ w_0, & \text{ako je} w \in L_1^c. \end{cases}$$

Očito za sve riječi $w \in \Gamma_1^*$ vrijedi:

$$w \in L_1 \text{ ako i samo ako } g(w) \in L,$$

te je g vremenski polinomno izračunljiva. Time smo dokazali da vrijedi $L_1 \leq_p L$. Iz pomoćne tvrdnje slijedi $L_1 \in$ co–NP.

Dokažimo sada obratnu inkluziju. Neka je $L_2 \in$ co–NP proizvoljan jezik. Budući da je po pretpostavci jezik L co–NP potpun tada vrijedi $L_2 \leq_p L$. No, po pretpostavci zadatka je $L \in$ NP. Iz pomoćne tvrdnje slijedi $L_2 \in$ NP.

13. Neka je DP= $\{L :$ postoji $L_1 \in$ NP i postoji $L_2 \in$ co–NP takvi da je $L = L_1 \cap L_2\}$. Zatim, definiramo sljedeće jezike:

SAT-UNSAT= $\{(F, G) :$ F, G su 3–knf, takve da je F ispunjiva, a G nije ispunjiva$\}$;

CRITICAL-SAT= $\{F :$ F je knf koja nije ispunjiva, ali brisanjem bilo koje njene elementarne disjunkcije dobivena formula postaje ispunjiva$\}$;

UNIQUE SAT= $\{F :$ F je knf za koju postoji jedinstvena interpretacija I takva da je $I(F) = 1\}$.

Dokažite da sva tri navedena problema pripadaju klasi DP, te da su prva dva DP–potpuna.
Uputa. Vidi [22].

Poglavlje 6

Dodatak

Ovo poglavlje se sastoji od četiri dijela. Prvo su dani dokazi nekih teorema koji su prije bili iskazani. Zatim su dana rješenja nekih težih zadataka. Većina ovog sadržaja čine studentski seminari. Treći dio posvećen je deskriptivnoj teoriji složenosti. U četvrtom dijelu ovog poglavlja dani su zadaci koji nisu izravno vezani uz teoriju složenosti.

6.1 Dokazi nekih teorema

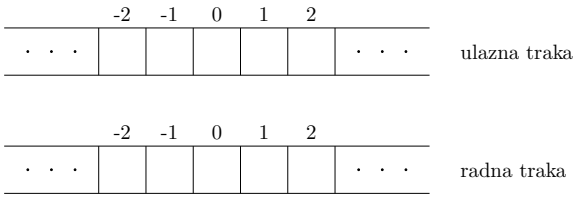
U ovoj točki dajemo dokaze nekih teorema koji su prije bili samo iskazani. Želimo istaknuti da su dokazi izdvojeni jer nisu ispredavani.

6.1.1 Teorem o linearnom ubrzanju

Na strani 20 iskazan je teorem o linearnom ubrzanju. Ovdje dajemo prvo vrlo detaljan dokaz s dosta ilustracija na način da traženi stroj ima veći broj traka, a onda dajemo drugu verziju dokaza u kojoj ne mijenjamo broj traka, ali povećavamo alfabet.

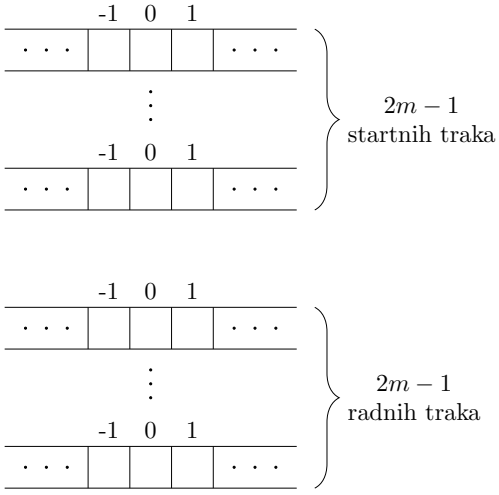
Teorem 6.1. (O linearnom ubrzanju)
Neka je L proizvoljan odlučiv jezik. Za svaki Turingov stroj T koji odlučuje jezik L i prirodan broj $m \neq 0$ postoji Turingov stroj S koji odlučuje jezik L, te za svaki $n \in \mathbb{N}$ vrijedi $time_S(n) \leq \frac{1}{m} time_T(n) + n$.

Dokaz. Bez smanjenja općenitosti možemo pretpostaviti da je T neki 2–tračni Turingov stroj. Trake stroja T redom nazivamo: ulazna traka i radna traka. Pretpostavljamo da su registri traka stroja T označeni cijelim brojevima.



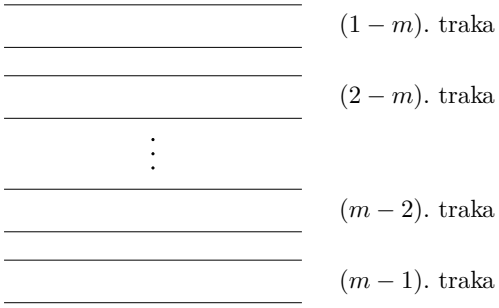
Slika 6.1: Sve trake stroja T

Traženi stroj S ima sljedeće trake: jednu ulaznu traku, $2m-1$ startnih traka i $2m-1$ radnih traka. Pretpostavljamo da su sve trake stroja S označene cijelim brojevima.



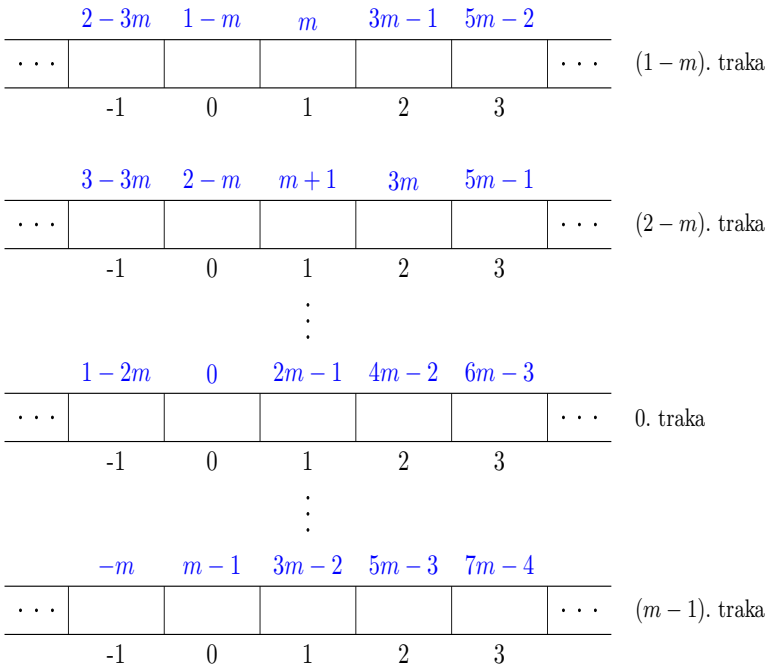
Slika 6.2: Startne i radne trake stroja S

Startne i radne trake stroja S označimo redom brojevima: $1-m,\ \ 2-m,\ \ \ldots,$ $m-2,\ \ m-1$.



Slika 6.3: Numeracija startnih, odnosno radnih, traka stroja S

Na i–toj traci (startnoj ili radnoj) j–ti registar označimo sa: $j \cdot (2m-1) + i$.



Slika 6.4: Označavanje registara stroja S

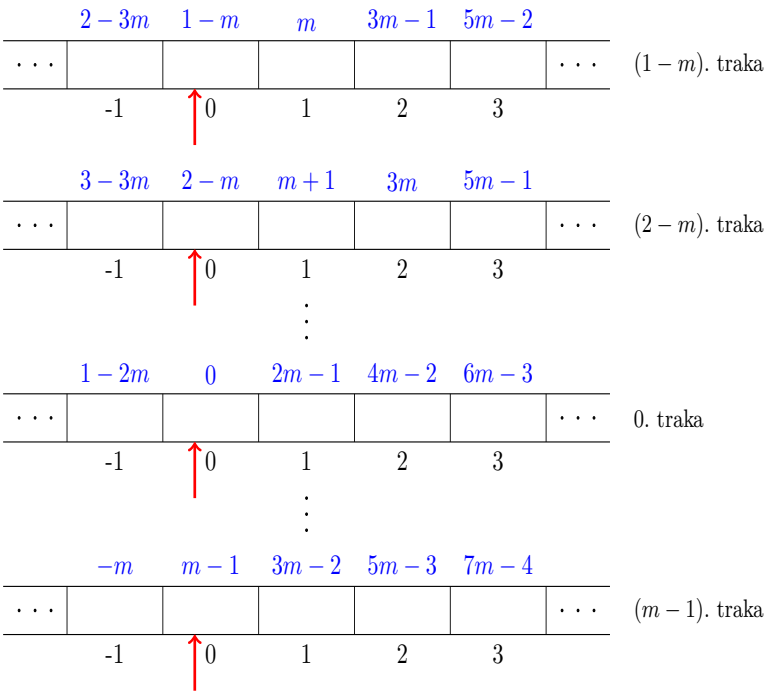
Uočimo da je "razmak" između susjednih registara na jednoj traci $2m-1$. Označimo $B = \{1-m,\ 2-m,\ 3-m, \ldots,\ m-3,\ m-2,\ m-1\}$. Zatim, za svaki $i \in B$ označimo $A_i = \{j \cdot (2m-1) + i : j \in \mathbb{Z}\}$. Lako je vidjeti da vrijede sljedeće dvije tvrdnje.

Pomoćna tvrdnja 1. Za sve $i_1 \neq i_2$ vrijedi $A_{i_1} \cap A_{i_2} = \emptyset$.

Pomoćna tvrdnja 2. Vrijedi $\bigcup_{i \in B} A_i = \mathbb{Z}$.

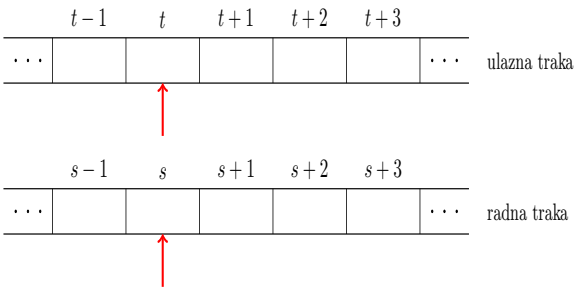
Ovakvim indeksiranjem postignuto je da i–tom registru ulazne trake stroja T odgovara točno jedan i–ti registar neke startne trake stroja S, te i–tom registru radne trake stroja T odgovara i–ti registar neke radne trake stroja S.

Na početku rada stroj S sadržaj i–tog registra svoje ulazne trake kopira u i–ti regisatar odgovarajuće startne trake. Nakon toga sve glave na j–toj startnoj traci stroj S postavi na j–ti registar .



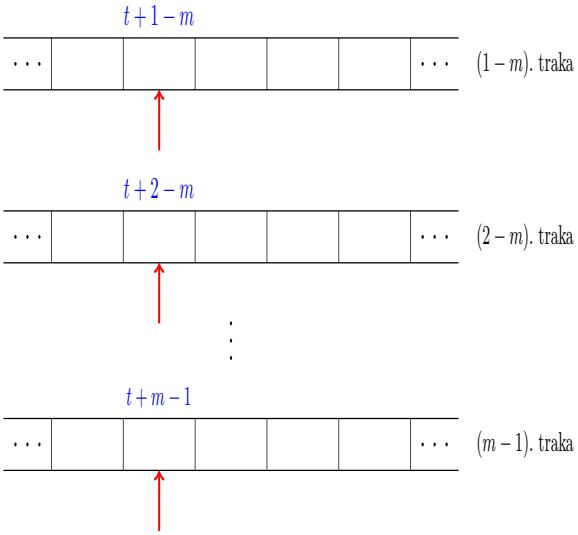
Slika 6.5: Položaj glava stroja S nakon kopiranja ulaza

Cilj nam je jednim korakom stroja S simulirati m uzastopnih koraka rada stroja T. Iz tog razloga promatramo rad stroja T u "segmentima" od po m uzastopnih koraka. Pretpostavimo da se nakon $m \cdot k$ $(k \in \mathbb{N})$ koraka rada stroja T glava na njegovoj ulaznoj traci nalazi na nekom t–tom registru $(t \in \mathbb{Z})$. Zatim, pretpostavimo da se glava na radnoj traci stroja T nalazi na s–tom registru $(s \in \mathbb{Z})$.



Slika 6.6: Položaj glava stroja T nakon $m \cdot k$ koraka

Neka su glave startnih traka stroja S pozicionirane redom na registre sa sljedećim indeksima: $t+1-m,\ t+2-m,\ \ldots,\ t+m-2,\ t+m-1$

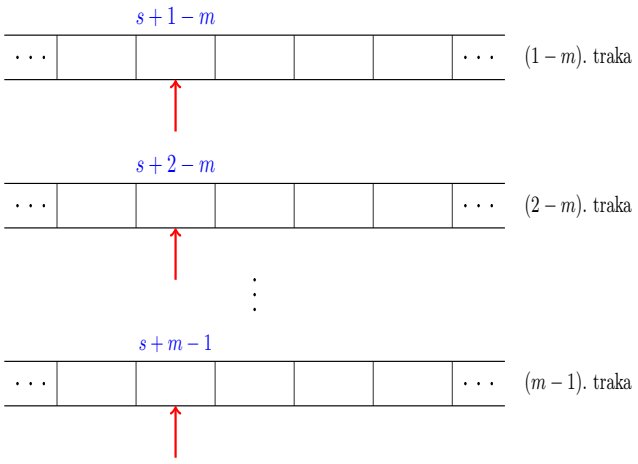


Slika 6.7: Položaj glava startnih traka stroja S nakon $m \cdot k$ koraka

Neka su glave radnih traka stroja S pozicionirane redom na registre sa sljedećim indeksima: $s+1-m,\ s+2-m,\ \ldots,\ s+m-2,\ s+m-1$

Pretpostavljamo da se sadržaj i–tog registra ulazne trake stroja T poklapa sa sadržajem i–tog registra odgovarajuće startne trake stroja S. Zatim, pretpostavljamo da se sadržaj i–tog registra radne trake stroja T poklapa sa sadržajem i–tog registra odgovarajuće radne trake stroja S. Opišimo kako će izgledati konfiguracija stroja S nakon sljedećih m koraka stroja T. Pretpostavimo da su nakon m koraka glave stroja T pozicionirane ovako:

- na ulaznoj traci je glava na nekom $(t+i)$–tom registru;



Slika 6.8: Položaj glava radnih traka stroja S nakon $m \cdot k$ koraka

- na radnoj traci je glava na nekom $(t + j)$–tom registru,

gdje su i, j prirodni brojevi za koje vrijedi $0 < i, j \leq m - 1$. (Radi određenosti promatramo slučaj kada stroj u m koraka napravi više koraka udesno nego lijevo, pa onda vrijedi $0 < i, j$.)

Primijetimo da je glava radne trake stroja T tijekom m uzastopnih koraka mogla promijeniti samo sadržaje registara čije su oznake u sljedećem intervalu: $s+1-m,\ s+2-m,\ \ldots,\ s+m-2,\ s+m-1$

Stroj S mora simulirati m uzastopnih koraka stroja T u jednom koraku. To radi na sljedeći način. Stroj S pomoću svoje funkcije prijelaza odredi:

- koje će biti stanje stroja T nakon m koraka;
- koje znakove treba zapisati na trake da bi sadržaj na njima odgovarao sadržaju traka stroja T, te ih zapiše
  (primijetimo da stroj S to može napraviti u jednom koraku jer su mu glave na registrima čiji indeksi pripadaju intervalima koje smo naveli);
- pomakne glave na trakama (to ćemo posebno opisati).

Stroj S treba pomaknuti glave na startnim trakama tako da pokrivaju sljedeći interval indeksa registara:

$$t + i + 1 - m,\ t + i + 2 - m,\ \ldots,\ t + i + m - 1,$$

odnosno, stroj S treba pomaknuti glave na radnim trakama tako da pokrivaju sljedeći interval indeksa registara:

$$s + j + 1 - m,\ s + j + 2 - m,\ \ldots,\ s + j + m - 1$$

(Stroj S na taj način "predviđa" m pomaka stroja T.)

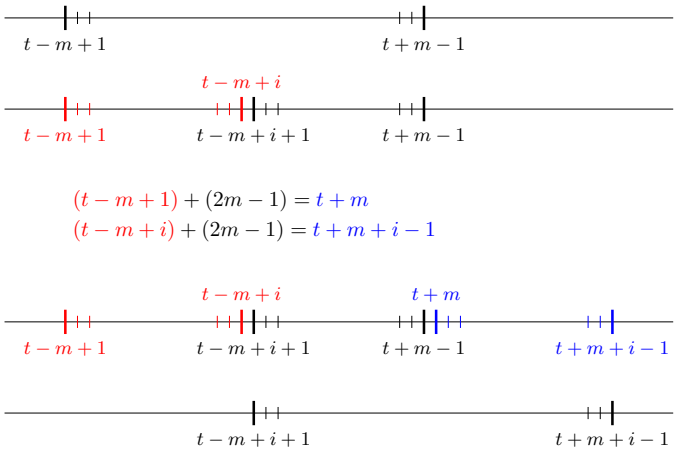
U tu svrhu stroj S pomakne samo glave na startnim trakama na registrima s oznakama:

$$t-m+1,\ t-m+2,\ \ldots,\ t-m+i,$$

odnosno, samo glave na radnim trakama nad sljedećim registrima:

$$s-m+1,\ s-m+2,\ \ldots,\ s-m+j$$

za jedno mjesto udesno. (PAZI! Registri su na trakama stroja S "udaljeni" za $2m-1$).



Slika 6.9: Intervali u kojima se mogu nalaziti glave stroja S

Ovako konstruirani Turingov stroj S radi m–puta manje koraka nego stroj T, a odlučuje isti jezik L. Kako čitanje ulaznih podataka nije moguće ubrzati, slijedi tvrdnja teorema. Primijetimo još samo da je time dokazan slučaj kada stroj T ima samo jednu radnu traku. Ako bi stroj T imao k radnih traka tada bi stroj S morao imati $k\cdot(2m-1)$ radnih traka. Q.E.D.

Uočimo da u prethodnom (pre)detaljnom dokazu teorema o linearnom ubrzanju nije mijenjan alfabet početnog stroja, ali je dozvoljeno da se za ubrzanje upotrebljava stroj s više traka. Sada dajemo dokaz istog teorema u kojem proširujemo početni alfabet, ali za ubrzanje koristimo stroj koji ima jednaki broj traka kao početni stroj.

Druga verzija dokaza teorema o linearnom ubrzanju (ne povećava se broj traka, već se povećava alfabet).
Neka je $T=(Q,\Gamma,\delta)$ jednotračni Turingov stroj koji odlučuje jezik L. Definirat ćemo Turingov stroj $S=(Q',\Gamma',\delta')$ koji ima tražena svojstva. Želimo da funkcija δ' u jednom koraku simulira m koraka stroja T. Budući da želimo da i stroj S bude jednotračni, proširit ćemo alfabet Γ i skup stanja Q.

Za svaku riječ $w \in \Gamma^*$ čija je duljina manja ili jednaka m, definiramo novo stanje koje označavamo sa q_w. Neka je $Q \cup \{q_w : w \in \Gamma^*,\ |w| \leq m\}$ podskup od Q' (kasnije ćemo potpuno definirati skup stanja Q'.)

Za svaki simbol $s \in \Gamma$ definiramo novi simbol $\overline{s}$ koji će označavati gdje se trenutno nalazi glava stroja T. Neka je $\overline{\Gamma} = \{\overline{s} : s \in \Gamma\}$. Alfabet Γ' stroja S definiramo ovako:

$$\Gamma' = \Gamma \cup \Gamma^m \cup \overline{\Gamma} \times \Gamma^{m-1} \cup \Gamma \times \overline{\Gamma} \times \Gamma^{m-2}$$

$$\cup\ \Gamma^2 \times \overline{\Gamma} \times \Gamma^{m-3} \cup \ldots \cup \Gamma^{m-2} \times \overline{\Gamma} \times \Gamma \cup \Gamma^{m-1} \times \overline{\Gamma}$$

Opišimo funkciju prijelaza δ'. Najprije pročitamo što je zapisano na traci, te ponovno zapišemo "istu" riječ, ali koristeći se simbolima alfabeta $\Gamma' \setminus \Gamma$. To možemo ovako: ako je stroj bio u stanju q_w i pročitao je simbol a, tada ili prelazimo u stanje q_{wa}, ako je $|w| < m$, ili zapisujemo simbol koji predstavlja riječ w, te stroj prelazi u stanje $q_\sqcup$ (sa $\sqcup$ označavamo praznu riječ).

Nakon što stroj S pročita ulaz (za što nam treba n koraka), svi ulazni simboli su zamijenjeni novim simbolima.

Nakon toga slijedi simulacija rada stroja T. Učitamo simbol, te nam on govori gdje se nalazi glava stroja T. Sljedećih m koraka stroja T možemo simulirati s jednim korakom stroja S. Naime, budući da jedan simbol stroja S predstavlja m simbola stroja T, simulacijom m koraka stroja T biti će potrebno izmijeniti najviše dva simbola na traci stroja S. Također, pomicanjem glave stroja T možemo simulirati pomicanjem crtice iznad simbola na traci stroja S i micanjem glave stroja S.

Dakle, simuliranjem m koraka stroja T, stroj S će napraviti 2 koraka. Zato je $2 \cdot time_S(n) \leq 1/m \cdot time_T(n)$, za svaki $n \in \mathbb{N}$. Q.E.D.

6.1.2 Teorem o vremenskoj hijerahiji

Teorem o vremenskoj hijerarhiji ističe ono što intuitivno očekujemo od Turingovog stroja: ako stroju damo više vremena, on će moći riješiti više problema. Primjerice, očekujemo da će Turingov stroj vremenske složenosti n^3 odlučivati više jezika nego Turingov stroj vremenske složenosti n^2. Drugim riječima, teorem osigurava da niti u jednom trenutku neće doći do kolapsa između klasa složenosti, odnosno da će za danu klasu uvijek postojati jezik koji joj ne pripada.

Možemo se pitati, koliko vremena trebamo dati Turingovom stroju da budemo sigurni da će se broj jezika koje on odlučuje povećati? Ispostavlja se da se njegova vremenska složenost mora povećati barem za logaritamski faktor. To je ujedno i glavna razlika u odnosu na teorem o prostornoj hijerarhiji koji ističe da bilo koje asimptotsko povećanje prostora osigurava nove jezike koji su odlučivi u tom prostoru. Naglasimo samo da je ovaj rezultat posljedica korištenja deterministčkog jednotračnog Turingovog stroja. Otvoreni je problem može li se pod istim pretpostavkama dobiti jača verzija teorema. Postupak dijagonalizacije, na kojemu se temelji dokaz teorema, danas je jedini način na koji znamo razdvajati klase vremenske složenosti.

Na strani 37 dan je iskaz ovog teorema. Sada dajemo malo izmijenjeni, ali ekvivalentan iskaz, a onda i dokaz.

Teorem 6.2. (O vremenskoj hijerarhiji)
Za svaku vremenski konstruktibilnu funkciju $f : \mathbb{N} \to \mathbb{N}$ *postoji jezik* L *tako da vrijedi* $L \in \mathrm{DTIME}(f(n)\log_2 f(n))$, *ali jezik* L *nije odlučiv u vremenu* $o(f(n))$.

Dokaz. Moramo pronaći jezik L takav da je L odlučiv na nekom deterministčkom Turingovom stroju vremenske složenosti $O(f(n)\log_2 f(n))$, te L ne smije biti odlučiv niti na jednom deterministčkom Turingovom stroju M za kojeg vrijedi $time_M(n) \sim o(f(n))$.

Jezik L definiramo tako što definiramo Turingov stroj D koji ga odlučuje, tj. vrijedit će $L = L(D)$. Ako je M neki Turingov stroj koji odlučuje jezik $L(M)$ tako da vrijedi $time_M(n) \sim o(f(n))$, stroj D mora osigurati da se jezik L razlikuje od jezika $L(M)$ na barem jednom mjestu. Ideja je da stroj D koristimo za simulaciju rada proizvoljnog stroja M s posebno određenim ulazom, te pritom brojimo korake simulacije. Ako se stroj M zaustavi u manje ili jednako $o(f(n))$ koraka u stanju q_{DA}, tada ćemo definirati da stroj D staje u stanju q_{NE}, i obratno.

Što nam omogućava da će Turingov stroj D imati dovoljno vremena da "zapamti" broj koraka simulacije stroja M? Budući da vrijedi $time_M(n) \sim o(f(n))$, iz toga slijedi da je $\lim_{n\to\infty} time_M(n)/f(n) = 0$. Dakle, mora postojati prirodan broj n_0, tako da za svaki prirodan broj $n \geq n_0$ vrijedi da je $time_M(n) < f(n)$. To znači da ako Turingov stroj M odlučuje jezik $L(M)$ u $time_M(n)$ koraka, tada broj koraka simulacije stroja M možemo ograničiti s $f(n)$, za dovoljno velik ulazni niz znakova ($n \geq n_0$). Stroj D će imati dovoljno vremena da izračuna vrijednost funkcije f jer je po pretpostavci teorema f vremenski konstruktibilna, pa ju je moguće izračunati u vremenu $O(f(n))$, a to je manje od pretpostavljene vremenske složenosti rada stroja D, tj. od $O(f(n)\log_2 f(n))$. Prema tome, možemo pretpostaviti da Turingov stroj D simulira rad stroja M u najviše $f(n)$ koraka. Ako se unutar $f(n)$ koraka stroj M zaustavi u nekom stanju, tada ćemo definirati da se stroj D zaustavlja u suprotnom stanju.

Bitno je naglasiti da je vremenska granica simuliranja rada stroja M važna jer stroju D treba ostati dovoljno vremena da može odlučiti kako se jezik koji odlučuje razlikuje od jezika $L(M)$. U slučaju da se stroj M ne zaustavi u $f(n)$ koraka, stroj D možda neće imati dovoljno vremena da odluči suprotno. Međutim, stroj D se tada nema potrebu ponašati drukčije od stroja M, pa će sve takve jezike ignorirati. Mi ćemo definirati da se stroj D u tom slučaju uvijek zaustavi u stanju q_{NE}.

Bili smo već spomenuli da se jezik L mora razlikovati od jezika $L(M)$ na barem jednoj riječi, za svaki Turingov stroj M za čiju vremensku složenost vrijedi $time_M(n) \sim o(f(n))$. U tu svrhu ćemo primijeniti postupak dijagonalizacije.

Bez smanjenja općenitosti možemo pretpostaviti da je alfabet svih Turingovih strojeva koje razmatramo jednak $\{0, 1\}$, jer je taj skup dovoljan za kodiranje. Svih Turingovih strojeva nad alfabetom $\{0, 1\}$ koji staju za svaki ulaz ima prebrojivo mnogo, pa ih možemo poredati u niz: $M_1, M_2, \ldots$ U sljedećoj tablici retci predstavljaju Turingove strojeve, a stupci njihove opise (kodove). Element u i–tom retku i j–tom

stupcu je primjer odlučivanja Turingovog stroja M_i kada mu je na ulazu dan njegov kod $\langle M \rangle$.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\dots$
M_1	$\mathbf{q}_{NE}$	q_{DA}	q_{DA}	$\dots$
M_2	q_{DA}	$\mathbf{q}_{DA}$	q_{NE}	$\dots$
M_3	q_{NE}	q_{NE}	$\mathbf{q}_{NE}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Posebno smo označili elemente na dijagonali tablice, jer će upravo to biti mjesta na kojima će se razlikovati strojevi D i M_i, za svaki $i \in \mathbb{N}$.

Primijetimo da, iako simuliramo rad Turingova stroja M u $f(n)$ koraka, još uvijek se može dogoditi da stroj D odluči isti jezik kao i neki stroj M. Do toga može doći u slučaju kada asimptotsko ponašanje funkcije $time_M$ još nije došlo do izražaja. U prethodnom dijelu zaključili smo da funkciju $time_M$ možemo ograničiti funkcijom f kada vrijedi da je $n \geq n_0$, za neki $n_0 \in \mathbb{N}$. No, u slučaju kada je $n < n_0$ može se dogoditi da vrijedi $time_M(n) > f(n)$, pa Turingov stroj M ne staje u najviše $f(n)$ koraka. Budući da ćemo definirati da stroj D simulira prvih $f(n)$ koraka rada stroja M, tada ćemo definirati da stroj D u tom slučaju staje u stanju q_{NE}. Naravno, problem nastaje kada i stroj M staje u stanju q_{NE}. Taj problem lako rješavamo tako da ulazni podatak $\langle M \rangle$ "produljimo" koliko treba. Umjesto $\langle M \rangle$ kao ulazni podatak uzet ćemo riječ $\langle M \rangle 10^\star$, gdje nam simbol "1" služi kao separator između koda stroja M i konačnog niza nula. Broj simbola "0" mora biti toliki da vrijedi $|\langle M \rangle 10^\star| \geq n_0$.

Sada definiramo rad stroja D :

$D =$ "na ulazu je riječ w, te je $n = |w|$":

1. Izračunaj $f(n)$ koristeći činjenicu da je funkcija f vremenski konstruktibilna i pohrani vrijednost $f(n)$ u unarnom zapisu na poseban dio trake. Ovaj dio trake na kojem je pohranjen broj $f(n)$ nazvat ćemo binarni brojač.

2. Binarni brojač smanji za jedan nakon svakog koraka rada stroja D u sljedećim fazama 3, 4 i 5. Ako vrijednost binarnog brojača u bilo kojem trenutku dođe do nule, tada završi u stanju q_{NE}.

3. Ako ulazna riječ w nije oblika $\langle M \rangle 10^\star$ za neki Turingov stroj M, završi u stanju q_{NE}.

4. Simuliraj rad Turingovog stroja M s ulazom w.

5. Ako stroj M završi rad u stanju q_{DA} tada neka stroj D stane u stanju q_{NE}. Ako pak stroj M stane u stanju q_{NE}, tada neka stroj D završi rada u stanju q_{DA}.

Svaka od faza 1, 2, 3, 4, 5 i 6 mora se moći izvršiti u vremenu $O(f(n)log_2 f(n))$. Da bismo proveli efikasnu simulaciju, Turingov stroj D ćemo promatrati kao poseban slučaj stroja koji ima više traka, ali samo jednu glavu koja se paralelno kreće po

svim trakama. Takva reprezentacija jednotračnog Turingovog stroja D je moguća jer znamo da se svaki k–tračni Turingov stroj može efikasno simulirati jednotračnim strojem uz povećanje alfabeta. Malo preciznije, ako je D_1 neki k–tračni Turingov stoj i Γ_1 njegov alfabet, onda postoji jednotračni Turingov stroj D_2 s alfbetom Γ_2^k takav da stroj D_2 može simulirati stroj D_1 uz konstantno povećanje vremena koje ovisi samo o broju traka i alfabetu. Ideja je da svaku k–torku simbola stroja D_1 reprezentiramo jednim simbolom na traci stroja D_2. To znači da stroj D_2 možemo promatrati kao da ima k paralelnih traka i jednu glavu koja se paralelno kreće duž svih traka. U dokaz ove tvrdnje se nećemo upuštati, on se može pogledati u [1].

Mi ćemo promatrati Turingov stroj D koji ima tri trake. Prva traka je radna traka i ona se koristi za simulaciju rada stroja M. Na drugoj traci je pohranjeno trenutno stanje stroja M kao i njegova funkcija prijelaza. Treća traka služi kao binarni brojač, tj. na njoj je pohranjena vrijednost $f(n)$. Možemo pretpostaviti da su na početku glava stroja nalazi na prvom znaku svake od riječi.

Funkcija f je po pretpostavci teorema vremenski konstruktibilna, pa se njena vrijednost može izračunati u vremenu $O(f(n))$. Također, moguće je u jednom prolazu provjeriti je li w riječ oblika $\langle M\rangle 10^*$. Stroj D treba samo krenuti od desnog kraja riječi i provjeriti slijedi li nakon nula (ako ih ima) jedinica pa kod stroja M. Još je potrebno provjeriti je li simulaciju stroja M kao i ažuriranje brojača moguće efikasno provesti.

Svaki put kad stroj D simulira jedan korak rada stroja M, on prvo uzme simbol s prve trake na kojeg trenutno pokazuje glava (dakle „glava" stroja M). Zatim pomiče glavu da bi pročitao trenutno stanje s druge trake. Kada stroj D sazna te informacije, on prolazi kroz funkciju prijelaza zapisanu na drugoj traci dok ne naiđe na podudaranje sa stanjem i simbolom (argumentima funkcije). Konačno, stroj D uzima vrijednosti funkcije prijelaza, zapisuje trenutno stanje na drugoj traci, te ažurira prvu traku. Ako je trenutno stanje q_{DA} ili q_{NE} stroj D završava s radom, inače ponavlja postupak. Kod simulacije stroja M, problem može nastati u slučaju da glava „ode" predaleko u jednu stranu. Budući da imamo samo jednu glavu za čitanje i pisanje može se dogoditi da njeno kretanje oduzme previše vremena i počme ovisiti o duljini ulazne riječi w. To se ne smije dogoditi jer bi rezultiralo da stroj D radi više od $O(f(n)\log_2 f(n))$ koraka (moramo uzeti u obzir i ažuriranje binarnog brojača na trećoj traci).

Da bi simulacija bila efikasna, stroj D mora sadržaj druge trake držati blizu trenutne pozicije glave stroja M na prvoj traci. To radi tako da svaki put kada promijeni poziciju glave na prvoj traci on prebaci čitav sadržaj druge trake za jedno mjesto u smjeru u kojem je pomaknuo glavu. Budući da sadržaj druge trake ne ovisi o duljini ulaza od stroja M (jer je na njoj samo opis od M), pomak dodaje samo konstantni faktor vremenu simulacije, tj. faktor $|\langle M\rangle|$. Nadalje, zbog toga jer stroj D uvijek drži sadržaj druge trake blizu glave stroja M, vrijeme pretraživanja funkcije prijelaza i ažuriranje prve trake također je konstantno. Dakle, ako stroj M radi $g(n)$ koraka, stroj D ga može simulirati u vremenu $O(g(n))$.

Binarni brojač sadrži vrijednost $f(n)$ u binarnom obliku koju Turingov stroj D

nakon svakog koraka simulacije stroja M smanjuje za jedan. To radi tako da pođe od najmanje značajnog simbola i ako je on jednak 1, na njegovo mjesto piše 0 i prestaje s operacijom smanjenja za jedan. Ako je on jednak 0, na njegovo mjesto piše 1 i vrši prijenos simbola. Očito je ažuriranje brojača moguće provesti u jednom prolasku i za to nam treba najviše $\log_2 f(n)$ koraka (jer je to duljina broja $f(n)$ u binarnom zapisu). Još trebamo izvršiti pomak sadržaja i ove trake u smjeru pomaka glave iz istog razloga kojeg smo gore naveli. To je dodatnih $\log_2 f(n)$ koraka, pa ažuriranje treće trake uzima $O(\log_2 f(n))$ koraka.

Dakle, budući da stroj M simuliramo u $f(n)$ koraka, ukupno vrijeme rada stroja iznosi $O(f(n) \log_2 f(n))$. To znači da je jezik A odlučiv u vremenu $O(f(n) \log_2 f(n))$.

Da bi pokazali da jezik A nije odlučiv u vremenu $o(f(n))$ pretpostavimo suprotno. Neka je M neki Turingov stroj koji odlučuje jezik A u vremenu $g(n) \sim o(f(n))$. Iz gornjeg dijela dokaza znamo da stroj D može simulirati stroj M u $dg(n)$ koraka, gdje je d neka konstanta. Budući da vrijedi $g(n) \sim o(f(n))$, tada postoji prirodan broj n_0 takav da za svaki $n \geq n_0$ vrijedi $dg(n) < f(n)$. To znači da će stroj D simuliranje stroja M privesti kraju kad god je duljina ulaznog niza veća ili jednaka n_0. Ako sada pokrenemo simulaciju stroja M s ulazom $\langle M \rangle 10^{n_0}$ slijedi da će simulacija završiti, te da će stroj D odlučiti suprotno od onoga što stroj M odluči. Dakle, došli smo do kontradikcije s pretpostavkom da je jezik A odlučiv u vremenu $o(f(n))$. Zaključujemo da jezik A nije odlučiv u vremenu $o(f(n))$. Q.E.D.

6.1.3 Teorem o praznini

Na strani 41 iskazan je sljedeći teorem.

Teorem 6.3. (O praznini)
Neka je g rekurzivna funkcija, takva da za svaki $n \in \mathbb{N}$ vrijedi $n \leq g(n)$. Tada postoji rekurzivna funkcija f tako da za svaki $n \in \mathbb{N}$ vrijedi $n \leq f(n)$, te imamo

$$\mathrm{DTIME}((g \circ f)(n)) = \mathrm{DTIME}(f(n)).$$

Skica dokaza. Neka je $T = (2, \Sigma, \Gamma, \delta)$ 2–tračni univerzalni Turingov stroj. Za ulaz (x, y) sa $time_T(x, y)$ označimo potreban broj koraka. Tvrdnja teorema slijedi iz sljedeće dvije pomoće tvrdnje.

Pomoćna tvrdnja 1. Postoji jedinstvena rekurzivna funkcija $f : \mathbb{N} \to \mathbb{N}$ tako da za svaki $n > 0$ i sve $x, y \in \Sigma^*$ takve da je $|x|, |y| \leq n$ vrijedi:

$$time_T(x, y) \leq f(n) \qquad \text{ili} \qquad time_T(x, y) \geq g^3(f(n))$$

Pomoćna tvrdnja 2. Za funkciju f iz tvrdnje 1 vrijedi:

$$\mathrm{DTIME}((g \circ f)(n)) = \mathrm{DTIME}(f(n))$$

6.1.4 PROBLEM Horn–SAT

Na strani 71 dan je sljedeći teorem. Ovdje dajemo detaljan dokaz.

Teorem 6.4. *PROBLEM Horn–SAT pripada klasi P.*

Dokaz. Neka je s HORN označen sljedeći algoritam:

1. Ako se konstanta $\top$ pojavljuje u formuli F tada je označi u svim nastupima u formuli F.
2. Ponavljaj sve dok se pojavljuju novi označeni simboli:

   za svaku Hornovu klauzulu $P_{i_1} \wedge \ldots P_{i_m} \rightarrow Q$ koja je konjunkt u formuli F, i za koju su već svi simboli P_{i_j} označeni, označi i simbol Q u svim klauzuluma u kojima se pojavljuje.
3. Ako je logička konstanta $\bot$ označena, tada algoritam staje u stanju q_{NE} (odnosno, algoritam odbacuje formulu F). Ako simbol $\bot$ nije označen tada algoritam staje u stanju q_{DA} (odnosno, algoritam prihvaća formulu F kao ispunjivu).

Preostalo je dokazati da je algoritam polinomne vremenske složenosti, te da je korektan. Neka je $\{P_1, \ldots, P_n\}$ skup svih propozicionalnih varijabli koje nastupaju u formuli F. Očito se petlja u algoritmu HORN ponavlja najviše $n+1$ puta (ima n varijabli, te moguće se označi i simbol $\bot$). Iz toga direktno slijedi da je algoritam HORN polinomne vremenske složenosti.

Prije dokaza korektnosti primijetimo prvo da se za neku Hornovu klauzulu F može dogoditi da primjenom algoritma HORN nije označen niti jedan simbol. Posebno, tada nije označen ni simbol $\bot$. Algoritam je u tom slučaju korektan, jer za interpretaciju $I \equiv 0$ očito vrijedi $I(F) = 1$, pa je formula F ispunjiva.

Kako bi dokazali korektnost algoritma HORN s ulazom F, dokazujemo indukcijom po broju izvršavanja petlje u algoritmu sljedeću tvrdnju:

$$(*) \left\{ \begin{array}{l} \text{ako je } P \text{ neka označena propozicionalna varijabla} \\ \text{tijekom rada algoritma HORN, tada za svaku interpretaciju } I \\ \text{za koju je } I(F) = 1, \text{ vrijedi } I(P) = 1. \end{array} \right.$$

Baza indukcije: prije početka izvršavanja petlje u algoritmi mogu biti označene samo logičke konstante $\top$ koje se pojavljuju u formuli tj. niti jedna propozicionalna varijabla ne može biti označena.

Neka je $k \in \mathbb{N}$ tako da tvrdnja $(*)$ vrijedi ako je u algoritmu HORN bilo k izvršavanja petlje. Ako je nastupilo i $(k+1)$.–to izvršavanje petlje, tada postoji barem jedna Hornova klauzula $H \equiv P_{i_1} \wedge \ldots P_{i_m} \rightarrow Q$ koja je konjunkt u formuli F, za koju su već svi simboli P_{i_j} označeni do k–tog izvršavanja petlje. Neka je I proizvoljna interpretacija I za koju je $I(F) = 1$. Iz pretpostavke indukcije slijedi $I(P_{i_j}) = 1$ za

svaki $j \in \{1, \ldots, m\}$. Budući da je H jedan konjunkt formule F, te je $I(F) = 1$, tada je $I(H) = 1$. Očito je tada i $I(Q) = 1$.

Dokažimo sada korektnost algoritma HORN. Promotrimo prvo slučaj kada je algoritam za neku Hornovu formulu F završio u stanju q_{NE}. Tada je logički simbol $\perp$ bio označen tijekom rada algoritma. To znači da postoji klauzula $H \equiv P_{i_1} \wedge \ldots P_{i_m} \rightarrow \perp$ koja je konjunkt u formuli F, za koju su već svi simboli P_{i_j} označeni. Ako bi formula F bila ispunjiva, tada bi postojala interpretacija I takva da je $I(F) = 1$, pa iz dokazane tvrdnje $(*)$ slijedi $I(P_{i_j}) = 1$ za svaki j. Budući da je H jedan konjunkt u formuli F, te je $I(F) = 1$, tada je $I(H) = 1$. No, iz $I(P_{i_1} \wedge \ldots \wedge P_{i_m}) = 1$, slijedi $I(\perp) = 1$, što je nemoguće. Dakle, ako je algoritmom HORN za Hornovu formulu F označena konstanta $\perp$, tada F nije ispunjiva.

Promotrimo sada slučaj kada je za Hornovu formulu F algoritam HORN završio tako da logička konstanta $\perp$ nije označena. Definiramo interpretaciju $I : Var(F) \rightarrow \{0, 1\}$ ovako:

$$I(P) = \begin{cases} 1, & \text{ako je varijabla } P \text{ označena tijekom rada algoritma;} \\ 0, & \text{inače.} \end{cases}$$

Pretpostavimo da je $I(F) = 0$. Tada postoji Hornova klauzula H od F tako da je $I(H) = 0$. Neka je $H \equiv P_{i_1} \wedge \ldots P_{i_m} \rightarrow Q$. Tada je $I(P_{i_1} \wedge \ldots P_{i_m}) = 1$ i $I(Q) = 0$. Iz definicije interpretacije I slijedi da su svi simboli P_{i_j} označeni. No, tada iz definicije algoritma HORN slijedi da je i simbol Q označen. Simbol Q ne može biti varijabla, jer iz $I(Q) = 0$ i definicije interpretacije I slijedi da Q nije označena varijabla. Simbol Q ne može biti logička konstanta $\perp$, jer upravo promatramo slučaj kada simbol $\perp$ nije označen. Simbol Q ne može biti logička konstanta $\top$, jer smo bili zaključili da je $I(Q) = 0$. Time je dobivena kontradikcija. Dakle, treba vrijediti $I(F) = 1$, tj. formula F treba biti ispunjiva ako algoritam HORN završi tako da simbol $\perp$ nije označen. Q.E.D.

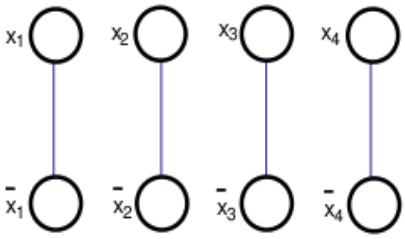
6.1.5 PROBLEM 3-OBOJIVOSTI

PROBLEM 3-OBOJIVOSTI nekoliko puta smo razmatrali u primjerima. Sada dajemo detaljan dokaz da je taj problem NP–potpun. Teorem je iskazan na strani 80. Tamo su dane i definicije svih pojmova.

Teorem 6.5. *PROBLEM 3-OBOJIVOST je NP–potpun.*

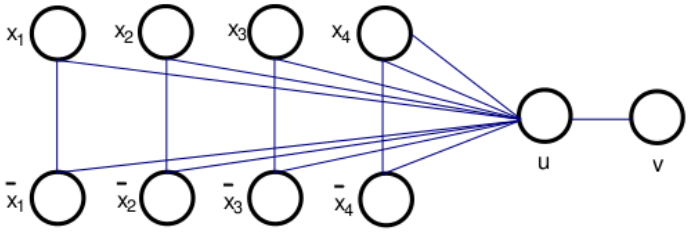
Dokaz. Neka je F neka 3–knf. Konstruirat ćemo graf G za nju koji je moguće obojati s tri boje ako i samo ako je F ispunjiva formula. Konstrukciju grafa G provodimo u tri koraka. Za čvorove grafa G prvo uzmemo sve literale koje određuju varijable formule F, te spojimo svaku varijablu s njenom negacijom.

Za formulu $F = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$ taj prvi dio grafa G izgleda:



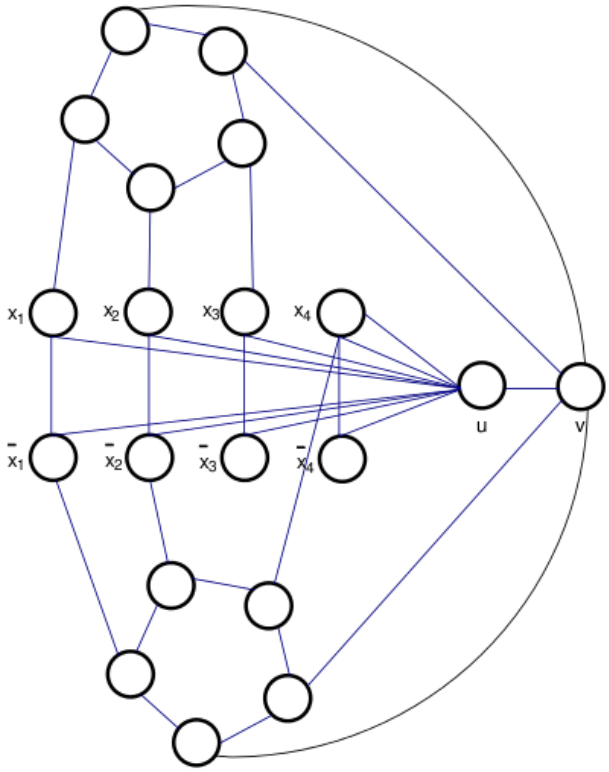
U drugom koraku dodamo još dva nova čvora U i V, i spojimo ih jedan s drugim. Nadalje, spojimo U sa svim literalima.

Za formulu $F = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$ nakon drugog koraka graf G izgleda:



U ovom dokazu dio grafa koji je u obliku peterokuta nazivamo pentagon. U trećem koraku konstrukcije grafa za svaku elementarnu disjunkciju $D = Z_{i_1} \vee Z_{i_2} \vee Z_{i_3}$ grafu G dodajemo pentagon. Spojimo dva proizvoljna susjedna vrha svakog pentagona sa čvorom V. U svakom pentagonu, koji je pridružen elementarnoj disjunkciji D, preostala tri vrha spojimo sa Z_{i_1}, Z_{i_2} i Z_{i_3}.

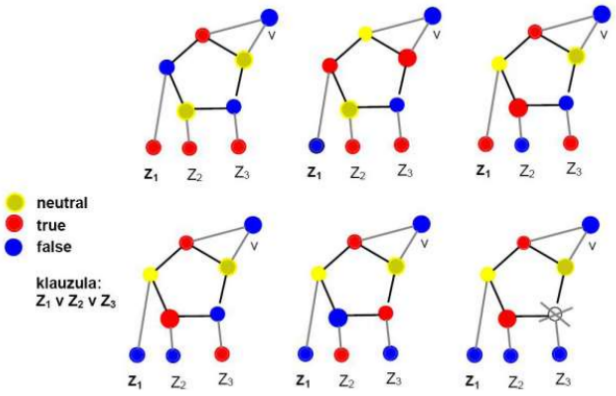
Za formulu $F = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$ graf G izgleda:



Tvrdimo da je tako konstruirani graf G moguće obojati s tri boje ako i samo ako je F ispunjiva formula. Ono što igra ključnu ulogu u ovom dokazu, a lako je za provjeriti jest da:

$(*)$ ako su za neku elementarnu disjunkciju $Z_{i_1} \vee Z_{i_2} \vee Z_{i_3}$ čvorovi Z_{i_1}, Z_{i_2}, Z_{i_3} i V obojeni s dvije boje, tada se pripadni pentagon može na "legalan" način obojati s tri boje ako i samo ako barem jedan od čvorova Z_{i_1}, Z_{i_2}, Z_{i_3} je obojan drugačije nego čvor V

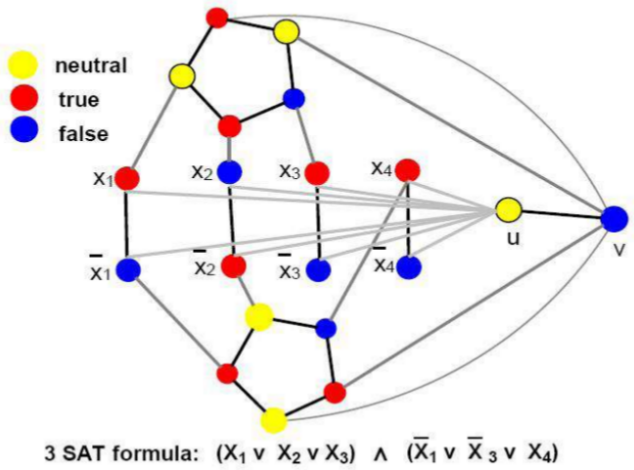
Na sljedećoj slici su "legalno" obojani pentagoni, pa slika može poslužiti kao pomoć prilikom dokaza prije navedene tvrdnje $(*)$.



Ako osiguramo da je čvor V obojan bojom koja označava neistinitost, tada graf G nije 3–obojiv ako i samo ako postoji elementarna disjunkcija koja se sastoji isključivo od neistinitih literala. Pretpostavimo da je F ispunjiva formula. Neka je I neka interpretacija za koju vrijedi $I(F) = 1$. U pripadnom grafu G obojimo s crvenom bojom sve one literale L za koje je $I(L) = 1$, a plavom bojom obojimo neistinite literale.

Sada još trebamo obojati čvorove U i V. To ćemo napraviti tako da žutom bojom obojimo vrh U, jer je on spojen s čvorovima koji su obojeni i u plavo i u crveno. Vrh V ćemo obojiti plavom bojom. Obzirom da svaka elementarna disjunkcija mora sadržavati bar jedan literal obojen crvenom bojom, ovakvo bojanje se može proširiti na vrhove pripadnog pentagona.

Za ispunjivu formulu $F = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$ i interpretaciju $I(x_1) = 1$, $I(x_2) = 0$, $I(x_3) = 1$, $I(x_4) = 1$ (uočite da je $I(F) = 1$) bojenje pripadnog grafa izgleda:



Obratno, pretpostavimo da je pripadni graf G formule F moguće obojati s tri boje (uzimamo da su te boje crvena, plava i žuta). Pretpostavimo da je V plave boje, a čvor U žute (moraju biti različito obojeni jer su spojeni). Tada čvorovi koji se odnose na literale moraju biti obojani samo s plavom i crvenom (jer je žuti čvor U spojen sa svakim literalom). Čvorovi s varijablama i njenim negacijama tada moraju biti obojani tako da je jedan vrh crven, a drugi plavi. Budući da su i vrhovi svakog pentagona legalno obojani, te je čvor V obojan plavo, tada iz tvrdnje $(*)$ slijedi da svaka elementarna disjunkcija sadrži jedan crveni čvor. Ukoliko crvene čvorove smatramo "istinitim" tada dobivamo interpretaciju za koju je formula F istinita. Q.E.D.

6.2 Rješenja nekih zadataka

1. Sada dokazujemo da je problem XSAT jedan NP–potpun problem (vidi zadatak 6 na strani 73).

   > *Problem XSAT (eng. Exact satisfiability) glasi: za danu knf F treba odrediti da li postoji interpretacija I takva da vrijedi $I(F) = 1$ i da je točno jedan literal u svakoj elementarnoj disjunkciji istinit.*

   Na sličan način definiramo problem 3–XSAT, samo što zahtijevamo da se u svakoj elementarnoj disjunkciji od F nalaze točno tri literala. Ovdje ćemo dokazati jaču tvrdnju: problem 3–XSAT je jedan NP–potpun problem. Uočimo da iz toga odmah slijedi da je i problem XSAT također NP–potpun jer je svaki 3–XSAT problem ujedno i XSAT problem.

   Jasno je da vrijedi 3-XSAT $\in$ NP jer za certifikat možemo uzeti parcijalnu interpretaciju. Tada algoritam za provjeru računa istinost formule i pritom provjerava je li u svakoj elementarnoj disjunkciji točno jedan literal istinit. Taj algoritam je očito polinomne vremenske složenosti. (Isti argument možemo ponoviti prilikom dokaza da vrijedi XSAT $\in$ NP).

   Dokažimo sada da je problem 3–SAT polinomno reducibilan na problem 3-XSAT. Iz toga odmah slijedi da je problem 3-XSAT jedan NP–težak problem.

   Za svaki literal Q uvodimo oznaku $\overline{Q}$ ovako:

   $$\sim Q \equiv \begin{cases} \neg Q, & \text{ako je } Q \text{ prop. varijabla;} \\ P, & \text{ako } Q \equiv \neg P. \end{cases}$$

   (Na taj način se izbjegavaju dvojne negacije, koje ne smiju doći u elementarnim disjunkcijama).

   Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$ neka 3–knf (C_i je elementarna disjunkcija). Definirat ćemo knf F' tako da vrijedi:

   $$F \in \text{3–SAT ako i samo ako } F' \in \text{3–XSAT} \qquad (*)$$

   Ako je $C_i \equiv Q_1 \vee Q_2 \vee Q_3$, te su P_1, P_2, P_3 i P_4 varijable koje ne nastupaju u formuli F, te ih još nismo upotrijebili za definiciju formule F', tada definiramo:

   $$C_i' \equiv (\sim Q_1 \vee P_1 \vee P_2) \wedge (Q_2 \vee P_2 \vee P_3) \wedge (\sim Q_3 \vee P_3 \vee P_4)$$

   Tada definiramo 3–knf F' ovako:

   $$F' \equiv \bigwedge_{i=1}^{s} C_i'.$$

Tvrdimo da za svaku interpretaciju $I : Var(F) \to \{0,1\}$ vrijedi:

$$(**)\begin{cases} I(F)=1 \quad \text{ako i samo ako} & \text{postoji proširenje } I' \text{ interpretacije } I \\ & \text{tako da vrijedi } I'(F')=1, \text{ te je u svakoj} \\ & \text{elementarnoj disjunkciji od } F' \text{ istinit} \\ & \text{točno jedan literal.} \end{cases}$$

Iz prethodne tvrdnje očito slijedi tvrdnja $(*)$.

Sada dokazujemo tvrdnju $(**)$. Neka je $I : Var(F) \to \{0,1\}$ proizvoljna interpretacija. Pretpostavimo prvo da vrijedi $I(F)=1$. Za svaki $i \in \{1,\ldots,s\}$ označimo redom:

$$\begin{aligned} C'_{i1} &\equiv \sim Q_1 \vee P_1 \vee P_2 \\ C'_{i2} &\equiv Q_2 \vee P_2 \vee P_3 \\ C'_{i3} &\equiv \sim Q_3 \vee P_3 \vee P_4 \end{aligned}$$

Za svaki $i \in \{1,\ldots,s\}$ definiramo traženo proširenje I' od I obzirom na slučajeve ovako:

(a) ako je $I(Q_2)=1$ tada definiramo:

$$\begin{aligned} &I'(P_1)=I(Q_1); \\ &I'(P_2)=I'(P_3)=0; \\ &I'(P_4)=I(Q_3) \end{aligned}$$

Lako je vidjeti da vrijedi $I'(C'_{i1})=I'(C'_{i2})=I'(C'_{i3})=1$, te je u svakoj od te tri elementarne disjunkcije istinit točno jedan literal za interpretaciju I'. To ilustriramo u sljedećim tablicama:

C_{i1}		
$\sim Q_1$	P_1	P_2
$1-I(Q_1)$	$I(Q_1)$	0

C_{i2}		
Q_2	P_2	P_3
1	0	0

C_{i3}		
$\sim Q_3$	P_3	P_4
$1-I(Q_3)$	0	$I(Q_3)$

(b) ako je $I(Q_2)=0$ tada promatramo sljedeća tri podslučaja:

2.a) Neka je $I(Q_1)=I(Q_3)=1$. Tada definiramo: $I'(P_1)=I'(P_3)=1$ i $I'(P_2)=I'(P_4)=0$.
Ovaj slučaj ilustriramo sljedećim tablicama.

C_{i1}		
$\sim Q_1$	P_1	P_2
0	1	0

C_{i2}		
Q_2	P_2	P_3
0	0	1

C_{i3}		
$\sim Q_3$	P_3	P_4
0	1	0

Dakle, vrijedi $I'(C'_{i1})=I'(C'_{i2})=I'(C'_{i3})=1$, te je u svakoj od te tri elementarne disjunkcije istinit točno jedan literal za interpretaciju I'.

2.b) Neka je $I(Q_1) = 1$ i $I(Q_3) = 0$. Tada definiramo: $I'(P_1) = I'(P_3) = I'(P_4) = 0$ i $I'(P_2) = 1$. Opet ćemo sve ilustrirati tablicama.

C_{i1}		
$\sim Q_1$	P_1	P_2
0	0	1

C_{i2}		
Q_2	P_2	P_3
0	1	0

C_{i3}		
$\sim Q_3$	P_3	P_4
1	0	0

Iz tablica vidimo da vrijedi $I'(C'_{i1}) = I'(C'_{i2}) = I'(C'_{i3}) = 1$, te je u svakoj od te tri elementarne disjunkcije istinit točno jedan literal za interpretaciju I'.

2.c) Neka je $I(Q_1) = 0$ i $I(Q_3) = 1$. Tada definiramo: $I'(P_1) = I'(P_2) = I'(P_4) = 0$ i $I'(P_3) = 1$. I ovaj slučaj prikazujemo tablično.

C_{i1}		
$\sim Q_1$	P_1	P_2
1	0	0

C_{i2}		
Q_2	P_2	P_3
0	0	1

C_{i3}		
$\sim Q_3$	P_3	P_4
0	1	0

Očito je $I'(C'_{i1}) = I'(C'_{i2}) = I'(C'_{i3}) = 1$, te je u svakoj od te tri elementarne disjunkcije istinit točno jedan literal za interpretaciju I'.

Dokažimo sada drugi smjer u tvrdnji $(**)$. Neka je $I'(F') = 1$, te neka je u svakoj elementarnoj disjunkciji od F' istinit točno jedan literal. Označimo sa I restrikciju interpretacije I' na skup $Var(F)$. Pretpostavimo da $I(F) = 0$. Tada postoji neki $i \in \{1, \ldots, s\}$ takav da je $I(C_i) = 0$. Ako je $C_i \equiv Q_1 \vee Q_2 \vee Q_3$ tada je $I(Q_1) = I(Q_2) = I(Q_3) = 0$. No, iz $I'(F') = 1$ posebno slijedi $I'(C'_i) = 1$, tj. $I'(C'_{i1}) = I'(C'_{i2}) = I'(C'_{i3}) = 1$. Sada iz $I'(C'_{i2}) = 1$ i $I'(Q_2) = I(Q_2) = 0$ slijedi $I'(P_2) = 1$ ili $I'(P_3) = 1$ (ali ne vrijedi $I'(P_2) = I'(P_3) = 1$). Budući da je $I'(\sim Q_1) = 1 - I'(Q_2) = 1 - I(Q_2) = 1$, te analogno $I'(\sim Q_3) = 1$, slijedi da su u barem jednoj od elementarnoj disjunkciji C'_{i1} i C'_{13} istinita dva literala za interpretaciju I', što je suprotno pretpostavci.

2. Dokazujemo da je problem `MAX–2–SAT` jedan `NP`–potpun problem (vidi zadatak 7 na strani 73).

Ponovimo prvo formulaciju problema `MAX–2–SAT`:

> *za zadanu 2–knf F i prirodan broj k, potrebno je ispitati postoji li interpretacija I takva da je za nju istinito barem k elementarnih disjunkcija formule F.*

Kako je problem `MAX–2–SAT` specijalni slučaj problema `SAT`, te vrijedi `SAT` $\in$ `NP`, tada očito vrijedi `MAX–2–SAT` $\in$ `NP`.

Sada dokazujemo da vrijedi 3–`SAT`$\leq_p$ `MAX–2–SAT`. U tu svrhu za svaku 3–knf F koja se sastoji od m elementarnih disjunkcija definiramo 2–knf G_F tako da vrijedi:

formula F je ispunjiva ako i samo ako postoji interpretacija I za koju je istinito najviše $7m$ elementarnih disjunkcija formule G_F.

Za svaku formulu A definiramo formulu $\sim A$ ovako:

$$\sim A = \begin{cases} B, & \text{ako } A \equiv \neg B; \\ \neg A, & \text{inače.} \end{cases}$$

Neka je $F \equiv C_1 \wedge \ldots \wedge C_m$ neka 3–knf gdje su C_i elementarnje disjunkcije s točno tri literala. Neka je $W = \{Q_1, \ldots, Q_m\}$ skup propozicionalnih varijabli tako da je $Var(F) \cap W = \emptyset$. Za svaku elementarnu disjunkciju $C_i \equiv C_{i1} \vee C_{i2} \vee C_{i3}$ (C_{i1}, C_{i2} i C_{i3} su literali) definiramo 2–knf G_i ovako:

$$\begin{aligned} G_i \equiv \; & \underbrace{C_{i1} \wedge C_{i2} \wedge C_{i3} \wedge Q_i}_{(a)} \wedge \\ & \underbrace{(\sim C_{i1} \vee \sim C_{i2}) \wedge (\sim C_{i2} \vee \sim C_{i3}) \wedge (\sim C_{i3} \vee \sim C_{i1})}_{(b)} \wedge \\ & \underbrace{(C_{i1} \vee \neg Q_i) \wedge (C_{i2} \vee \neg Q_i) \wedge (C_{i3} \vee \neg Q_i)}_{(c)}. \end{aligned}$$

Neka je $G_F \equiv G_1 \wedge \ldots \wedge G_m$. Uočimo da je formula G_F jedna 2–knf. Pokažimo da je formula G_F antitautologija. Pretpostavimo suprotno. Neka je I interpretacija za koju je $I(G_F) = 1$. Tada je posebno i $I(G_1) = 1$. No, iz $I(G_1) = 1$ slijedi da je dio formule G_1 koji smo označili sa (a) istinit za interpretaciju I. To znači da vrijedi $I(C_{11}) = I(C_{12}) = I(C_{13}) = I(Q_1) = 1$. Tada je $I(\sim C_{i1}) = I(\sim C_{i2}) = 0$, pa je $I(\sim C_{i1} \vee \sim C_{i2}) = 0$. Iz ovog posljednjeg slijedi da je je $I(G_1) = 0$, što je suprotno pretpostavci.

Pokažimo da za svaku interpretaciju I te za svaki $i \in \{1, \ldots m\}$ postoji najviše 7 elementarnih disjunkcija u formuli G_i koje su istinite za interpretaciju I. Neka je I proizvoljna interpretacija. Promatramo sljedeće slučajeve:

a) $I(C_{i1}) = I(C_{i2}) = I(C_{i3}) = 1$.
Tada su istinite tri elementarne disjunkcije iz skupine (a), te su istinite sve tri elementarne disjunkcije iz skupine (c). Svaka elementarna disjunkcija iz dijela formule G_i koji smo označili sa (b) je neistinita.

Ovisno o vrijednosti $I(Q_i)$ imamo da je za ovaj slučaj istinito 6 ili 7 elementarnih disjunkcija formule G_i.

b) istinita su točno dva literala iz skupa $\{C_{i1}, C_{i2}, C_{i3}\}$ za interpretaciju I.
Tada su istinite dvije elementarne disjunkcije iz skupine (a) i dvije iz skupine (b), tj. četiri ukupno.

Ako je $I(Q_i) = 1$ tada u skupini (a) imamo još jednu istinitu elementarnu disjunkciju, a u skupini (c) dvije.

Ako je $I(Q_i) = 0$ onda imamo tri istinite elementarne disjunkcije u skupini (c).

To znači da bez obzira na vrijednost interpretacije I na varijabli Q_i u ovom slučaju imamo točno 7 istinitih elementarnih disjunkcija u formuli G_i.

c) istinit je točno jedan literal iz skupa $\{C_{i1}, C_{i2}, C_{i3}\}$ za interpretaciju I.
Tada je u skupini (a) istinit jedan literal, a u skupini (b) tri.

Ako je $I(Q_i) = 1$ tada je u skupini (a) istinita još jedna elementarna disjunkcija, a u skupini (c) su sve tri neistinite. To znači da je u ovom slučaju istinito 5 elementarnih disjunkcija.

Ako je $I(Q_i) = 0$ tada su u skupini (c) istinite 3 elementarne disjunkcije, pa je u ovom slučaju ukupno 7 elementarnih disjunkcija istinito.

d) $I(C_{i1}) = I(C_{i2}) = I(C_{i3}) = 0$.
Tada su istinite sve 3 elementarne disjunkcije iz skupine (b).

Ako je $I(Q_i) = 1$ tada je iz skupine (a) istinita još jedna elementarna disjunkcija. U ovom slučaju imamo ukupno 4 istinite elementarne disjunkcije formule G_i.

Ako je $I(Q_i) = 0$ tada su istinite sve 3 elementarne disjunkcije iz skupine (c). Tada je u tom slučaju istinito ukupno 6 elementarnih disjunkcija.

Pretpostavimo prvo da je 3–knf F ispunjiva, tj. $F \in$3–SAT. Neka je I interpretacija takva da je $I(F) = 1$. Iz prethodnih razmatranja slijedi da je tada za interpretaciju I istinito maksimalno $3m$ elementarnih disjunkcija formule G_F. To znači da je $(G_F, 3m) \in$MAX–2–SAT.

Lako je vidjeti da vrijedi i obrat, tj. ako je za svaku interpretaciju I istinito maksimalno $7m$ elementarnih disjunkcija formule G_F tada je formula F ispunjiva.

Zatim, očito je za svaku knf F formulu G_F moguće konstruirati u polinomnom vremenu (u odnosu na duljinu riječi F).

3. Dokazujemo da vrijedi PROBLEM VRIJEDNOST SKLOPA $\in$ P (vidi zadatak 9 na strani 74). PROBLEM VRIJEDNOST SKLOPA (eng. CIRCUIT VALUE) glasi:

> *za dani sklop C, čija niti jedna vrata nisu propozicionalna varijabla, treba odrediti da li postoji interpretacija I adekvatna za C takva da je $I(C) = 1$.*

Koristeći se algoritmom E. W. Dijkstre možemo formulu prebaciti u postfix oblik u kojem je pogodna za evaluaciju. Koraci algoritma su sljedeći:

(a) Pročitaj sljedeći znak.
(b) Ako je znak 0 ili 1 ispiši ga.
(c) Ako je znak operator,

- sve dok stog nije prazan i operator na vrhu stoga ima veći ili jednak prioritet od pročitanog, ispiši i ukloni operator na vrhu stoga, zatim
- stavi znak na stog.

(d) Ako je znak "(" onda ga stavi na stog.

(e) Ako je znak ")" onda ispiši i uklanjaj sve operatore sa stoga dok ne naiđeš na zbak "(". Ukloni znak ")" sa vrha stoga.

(f) Ako ima još znakova za proçitati, onda idi na korak 1., inače ispisuj i uklanjaj ostatak stoga.

Napomenimo samo da se stog trivijalno simulira dodatnom trakom na Turingovom stroju. Operatori poredani silazno po prioritetu su $\neg$, $\wedge$, $\vee$. Dodatno, smatramo da "(" ima najmanji prioritet. Za evaluaciju se postavimo na početak postfix izraza i pratimo sljedeće korake:

(a) Pročitaj sljedeći znak.

(b) Ako je znak 0 ili 1 onda ga stavi na stog.

(c) Ako je znak n–aran operator onda uzmi prvih n elemenata s vrha stoga i evaluiraj operator nad tim argumentima. Rezultat stavi na stog.

(d) Ako ima još znakova za pročitati onda idi na korak 1., a inače ispiši rezultat s vrha stoga kao konačno rješenje.

Algoritam je očito polinomne vremenske složenosti i rješava dani problem.

4. Dokazujemo da je problem `NAESAT` jedan `NP`–potpun problem (vidi zadatak 10 na strani 74).

> *Problem* `NAESAT` *(eng. Not-All-Equal satisfiability) glasi: za danu knf F treba odrediti da li postoji interpretacija I takva da vrijedi $I(F) = 1$ i da je barem jedan literal u svakoj elementarnoj disjunkciji neistinit.*

Problem `NAESAT`, kao i svaka druga varijanta problema `SAT`, je iz klase `NP`. Naime, iz teorema o certifikatu slijedi da je za svaku knf dovoljno pronaći certifikat i onda je lako, u polinomnom vremenu, provjeriti je li pojedini literal istinit za svaku elementarnu disjunkciju.

Preostalo je još izabrati neki `NP`–potpun jezik i vidjeti je li on polinomno reducibilan na jezik `NAESAT`. U tu svrhu ćemo uzeti `PROBLEM SKLOP–SAT` koji je `NP`–potpun.

Za dani sklop C želimo konstruirati knf $R(C)$ tako da vrijedi: $R(C) \in$ `NAESAT` ako i samo ako je sklop C ispunjiv. To nije teško za napraviti budući da su formule i sklopovi zapravo dva različita načina reprezentiranja propozicionalnih funkcija, pa je međusobna konverzija jednostavna. Skup varijabli koje nastupaju

u $R(C)$ će sadržavati sve varijable sklopa C, i uz to za svaka vrata g sklopa C imat ćemo dodatnu varijablu u $R(C)$ koju također označavamo sa g.

Za svaka vrata sklopa C konstruirat ćemo formulu koja će biti podformula od $R(C)$.

Ako su g vrata čija je vrsta varijabla P_i, onda formuli $R(C)$ dodajemo dvije elementarne disjunkcije: $(\neg g \vee P_i)$ i $(g \vee P_i)$. Uočimo da za svaku interpretaciju I, za koju su obje navedene elementarne disjunkcije istinite, vrijedi $I(g) = I(P_i)$, odnosno $(\neg g \vee P_i)$ i $(g \vee P_i)$ je konjunktivna normalna forma formule $g \leftrightarrow P_i$.

Ako su g vrata čija je vrsta logička konstanta $\top$, tada jednostavno dodajemo elementarnu disjunkciju (g). Ako su g vrata čija je vrsta logička konstanta $\bot$, tada formuli $R(C)$ dodajemo $(\neg g)$.

Ako su g vrata čija je vrsta $\neg$, te su vrata h prethodnik od g u sklopu C, onda konstruiramo formulu koja će izražavati $g \leftrightarrow \neg h$, tj. formuli $R(C)$ dodajemo $(\neg g \vee \neg h) \wedge (g \vee h)$.

Ako su g vrata čija je vrsta $\vee$, te h_1 i h_2 prethodnici od g u sklopu C, tada u $R(C)$ dodajemo sljedeću knf: $(\neg h_1 \vee g) \wedge (\neg h_2 \vee g) \wedge (h_1 \vee h_2 \vee \neg g)$ (primijetimo da je dana knf ekvivalentna formuli $g \leftrightarrow (h_1 \vee h_2)$).

Slično, ako su g vrata čija je vrsta $\wedge$, te su h_1 i h_2 prethodnici od g u sklopu C, onda konstruiramo knf za $g \leftrightarrow (h_1 \wedge h_2)$. Ta formula se može zapisati u konjunktivnoj normalnoj formi kao $(\neg g \vee h) \wedge (\neg g \vee h_2) \wedge (\neg h_1 \vee \neg h_2 \vee g)$, i ona se dodaje u $R(C)$.

Ako su g izlazna vrata sklopa C, tada u $R(C)$ dodajemo elementarnu disjunkciju (g). Do sada definirani dio formule $R(C)$ ćemno nazivati početna formula i označavati sa $R_0(C)$. Očito za svaku interpretaciju I adekvatnu za formulu $R_0(C)$ vrijedi: $I(C) = 1$ ako i samo ako $I(R_0)(C) = 0$.

Sada još svim elementarnim disjunkcijama u formuli $R_0(C)$ s jednim ili dva literala dodajemo još jedan literal isti za sve elementarne disjunkcije kojeg označavamo sa P. Smatramo da je P nova propozicionalna varijabla. Na taj način smo dobili da je formula $R(C)$ jedna 3–knf.

Očito je da se ova redukcija može napraviti u polinomnom vremenu jer sve što trebamo učiniti je proći po svim vratima sklopa, te prema navedenom postupku za svaka vrata načiniti knf, te na kraju u određene elementarne disjunkcije dodati još jedan literal kako bi dobili finalnu knf.

Tvrdimo da vrijedi: $R(C) \in$ `NAESAT` ako i samo ako sklop C je ispunjiv. Pretpostavimo prvo da postoji interpretacija I za koje su sve elementarne disjunkcije formule $R(C)$ istinite, te u svakoj elementarnoj disjunkciji postoji barem jedan literal Q tako da vrijedi $I(Q) = 0$. Označimo sa I^c suprotnu interpretaciju od I. Očito vrijedi $I^c(R(C)) = 1$.

Za novu dodanu varijablu P očito vrijedi $I(P) = 1$ ili $I^c(P) = 1$. Iz toga slijedi da će barem za jednu od tih interpretacija biti istinite sve one elementarne

dijsunkcije od $R(C)$ u koje smo dodavali varijablu P. No, tada je ispunjiv i sklop C, jer formula s uklonjenim varijablama P odgovara upravo formuli koja se dobije redukcijom iz sklopa, pa kako je ona ispunjiva, ispunjiv je i sklop.

Dokažimo sada obrat. U tu svrhu pretpostavimo da postoji interpretacija I takva da je $I(C) = 1$. Tada postoji interpretacija I_1 za koju vrijedi $I_1(P) = 0$ i $I_1(R_0(C)) = 1$.

Sada tvrdimo da u svakoj elementarnoj disjunkciji formule $R(C)$ postoji literal Q tako da je $I_1(Q) = 0$. Da bismo to dokazali koristimo činjenicu da početna formula $R_0(C)$ ima točno određeni oblik koji je ranije definiran. Imamo točno definirane grupe elementarnih disjunkcija koje odgovaraju određenoj vrsti vrata sklopa iz kojeg je ta formula konstruirana redukcijom. Vrata čija je vrsta $\top$, $\bot$, $\neg$,, varijabla, odnosno ako su izlazna vrata, se sastoje od elementarnih disjunkcija s po jednim ili dva literala, pa je tim elementarnim disjunkcijama dodan literal P, a kako je on $I_1(P) = 0$, tada će svaka takva elementarna disjunkcija sadržavati literal koji nije istinit za interpretaciju I_1.

Za vrata čija je vrsta $\vee$ bili smo definirali da se u formulu $R_0(C)$ dodaju sljedeće elementarne disjunkcije: $(\neg h_1 \vee g)$, $(\neg h_2 \vee g)$ i $(h_1 \vee h_2 \vee \neg g)$. Tada su u formulu $R(C)$ dodane sljedeće elementarne disjunkcije:

$$\begin{array}{ll} (1) & (\neg h_1 \vee g \vee P); \\ (2) & (\neg h_2 \vee g \vee P); \\ (3) & (h_1 \vee h_2 \vee g). \end{array}$$

Prve dvije elementarne disjunkcije sadrže literal P, pa onda sadrže literal koji je neistinit za interpretaciju I_1.. Treća elementarna disjunkcija također mora sadržavati neki literal koji je neistini za interpretaciju I_1, jer bi u suprotnom prve dvije elementarne disjunkcije bile neistinite, što nije moguće zbog pretpostavke da vrijedi $I_1(R_0(C)) = 1$.

Za vrata čija je vrsta $\wedge$ bili smo definirali da se dodaju sljedeće tri elementarne disjunkcije: $(\neg g \vee h)$, $(\neg g \vee h_2)$ i $(\neg h_1 \vee \neg h_2 \vee g)$ u početnu formulu $R_0(C)$, odnosno sljedeće tri elementarne disjunkcije u formulu $R(C)$:

$$\begin{array}{ll} (1) & (\neg g \vee h \vee P); \\ (2) & (\neg g \vee h_2 \vee P); \\ (3) & (\neg h_1 \vee \neg h_2 \vee g). \end{array}$$

Uočimo da u prve dvije elementarne disjunkcije postoji litral koji je neistinit za interpretaciju I_1 : to je literal P. Ako bi u trećoj elementarnoj disjunkciji svi literali bili istiniti, tj. ako bi vrijedilo $I_1(\neg h_1) = I_1(\neg h_2) = I_1(g) = 1$, tada je lako vidjeti da bi prve dvije elementarne disjunkcije bile neistinite za interpretaciju I_1. To je u kontradikciji s činjenicom $I_1(R_0(C)) = 1$.

5. U zadatku 1 na strani 80 formuliran je `PROBLEM SUMA PODSKUPA` koji glasi:

za zadani konačni multiskup $S \subseteq \mathbb{N}$ i $t \in \mathbb{N}$ treba odrediti da li postoji $T \subseteq S$ tako da vrijedi $\sum_{x \in T} x = t$.

Dokažimo da je `PROBLEM SUMA PODSKUPA` jedan `NP`–potpun problem.
Rješenje. Očito vrijedi `PROBLEM SUMA PODSKUPA` $\in$ `NP` (podskup T koji ima traženo svojstvo je jedan certifikat). Dokažimo da je problem 3–`SAT` moguće polinomno reducirati na `PROBLEM SUMA PODSKUPA`. Neka je $F \equiv C_1 \wedge \ldots \wedge C_s$ neka 3–knf (svaka formula C_i je elementarna disjunkcija koja se sastoji točno od tri literala). Za svaku propozicionalnu varijablu P_i koja nastupa u formuli F definirat ćemo dva prirodna broja koja ćemo označavati sa p_i i n_i. Za svaku elementarnu disjunkciju C_j definirat ćemo dva prirodna broja koja ćemo označavati sa c_j i d_j.

Pretpostavimo da su u formuli F pojavljuju varijable $P_1, \ldots, P_k$. Za svaku varijablu P_i i svaku elementarnu disjunkciju C_j definiramo:

$$\delta_{ij} = \begin{cases} 1, & \text{ako se literal } P_i \text{ pojavljuje u elementarnoj disjunkciji } C_j, \\ 0, & \text{inače} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1, & \text{ako se literal } \neg P_i \text{ pojavljuje u elementarnoj disjunkciji } C_j, \\ 0, & \text{inače.} \end{cases}$$

Zatim za svaku propozicionalnu varijablu P_i definiramo brojeve p_i i n_i u dekadskom zapisu ovako: $p_i = 1 \underbrace{0 \ \ldots \ 0}_{k-i \text{ puta}} \delta_{i1} \, \delta_{i2} \, \ldots \, \delta_{is}$ i $n_i = 1 \underbrace{0 \ \ldots \ 0}_{k-i \text{ puta}} \gamma_{i1} \, \gamma_{i2} \, \ldots \, \gamma_{is}$.

Za svaku elementarnu disjunkciju C_j, $j \in \{1, \ldots, s\}$, definiramo brojeve c_j i d_j ovako: $c_j = d_j = 1 \underbrace{0 \ \ldots \ 0}_{s-j \text{ puta}}$.

Na kraju definiramo broj t ovako: $t = \underbrace{1 \ \ldots \ 1}_{k \text{ puta}} \underbrace{3 \ \ldots \ 3}_{s \text{ puta}}$. Upravo definirane brojeve pregledno prikazujemo u sljedećoj tablici.

	P_1	P_2	P_3	P_4	...	P_k	C_1	C_2	...	C_s
p_1	1	0	0	0	...	0	δ_{11}	δ_{12}	...	δ_{1s}
n_1	1	0	0	0	...	0	γ_{11}	γ_{12}	...	γ_{1s}
p_2		1	0	0	...	0	δ_{21}	δ_{22}	...	δ_{2s}
n_2		1	0	0	...	0	γ_{21}	γ_{22}	...	γ_{2s}
p_3			1	0	...	0	δ_{31}	δ_{32}	...	γ_{3s}
n_3			1	0	...	0	γ_{31}	γ_{32}	...	γ_{3s}
$\vdots$					$\vdots$				$\vdots$	
p_k						1	δ_{k1}	δ_{k2}	...	δ_{ks}
n_k						1	γ_{k1}	γ_{k2}	...	γ_{ks}
c_1							1	0	...	0
d_1							1	0	...	0
c_2								1	...	0
d_2								1	...	0
$\vdots$									$\vdots$	
c_s										1
d_s										1
t	1	1	1	1	...	1	3	3	...	3

Uočimo da je broj t tako odabran da izborom podskupa T možemo dobiti da je suma svake kolone u tablici (bilo koje P_i–te kolone i bilo koje C_j–te kolone) upravo jednaka znamenki broja t u toj koloni.

Radi ilustracije navodimo kako izgledaju brojevi p_i i n_i, te c_j i d_j, za formulu:

$$F \equiv (P_1 \vee P_2 \vee P_4) \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_3) \wedge (\neg P_2 \vee P_3 \vee \neg P_4).$$

Tada je:

$$\begin{aligned} C_1 &\equiv P_1 \vee P_2 \vee P_4, \\ C_2 &\equiv \neg P_1 \vee \neg P_2 \vee \neg P_3, \\ C_3 &\equiv \neg P_2 \vee P_3 \vee \neg P_4. \end{aligned}$$

Pripadne brojeve za formulu F dajemo u sljedećoj tablici

	P_1	P_2	P_3	P_4	C_1	C_2	C_3
p_1	1	0	0	0	1	0	0
n_1	1	0	0	0	0	1	0
p_2		1	0	0	1	0	0
n_2		1	0	0	0	1	1
p_3			1	0	0	0	1
n_3			1	0	0	1	0
p_4				1	1	0	0
n_4				1	0	0	1
c_1					1	0	0
d_1					1	0	0
c_2						1	0
d_2						1	0
c_3							1
d_3							1
t	1	1	1	1	3	3	3

Za danu 3–knf $F \equiv C_1 \wedge \ldots \wedge C_s$ u kojoj se pojavljuju varijable $P_1, \ldots, P_k$ definiramo multiskup prirodnih brojeva S ovako:

$$S = \{p_i,\ n_i : i = 1, \ldots, k\} \cup \{c_j, d_j : j = 1, \ldots, s\}.$$

Tvrdimo da vrijedi: formula F je ispunjiva ako i samo ako postoji $T \subseteq S$ tako da vrijedi $\sum\limits_{x \in T} x = t$.

Pretpostavimo prvo da je formula F ispunjiva. Neka je I neka interpretacija koja ima svojstvo $I(F) = 1$. Definiramo $T \subseteq S$ koji će imati traženo svojstvo. Za svaki $i \in \{1, \ldots, k\}$ definiramo $p_i \in T$ ako je $I(P_i) = 1$, odnosno $n_i \in T$ ako je $I(P_i) = 0$. Zatim, za svaki $j \in \{1, \ldots, s\}$ definiramo $c_j \in T$, a možda i $d_j \in T$, pazeći da suma u C_j–toj koloni bude 3. Očito je $\sum\limits_{x \in T} x = t$.

Pretpostavimo sada da postoji $T \subseteq S$ tako da vrijedi $\sum\limits_{x \in T} x = t$. Definiramo interpretaciju I ovako:

$$I(P_i) = \begin{cases} 1, & \text{ako je } \ p_i \in T; \\ 0, & \text{ako je } \ n_i \in T. \end{cases}$$

Uočimo da je interpretacija I dobro definirana jer nemoguće je da za neki i vrijedi $p_i, n_i \in T$, budući da bi tada suma u P_i–toj koloni u tablici bila 2, a ne jedan (znamenka broja t u P_i–toj koloni je 1). Zatim, za svaki $i \in \{1, \ldots, k\}$ mora vrijediti $p_i \in T$ ili $n_i \in T$, jer inače bi suma u P_i–toj koloni u tablici bila 0. Primijetimo još da je $I(F) = 1$. Razlog tome je što je suma u svakoj C_j–toj koloni jednaka 3. Suma znamenaka brojeva c_j i d_j u C_j–toj koloni je najviše

2. To znači da još barem 1 mora za sumu 3 u C_j–toj koloni doći od nekog p_i ili n_i. Tada za svaki $j \in \{1, \ldots, s\}$ postoji $i \in \{1, \ldots, k\}$ tako da $\delta_{ij} = 1$ ili $\gamma_{ij} = 1$. Ako je $\delta_{ij} = 1$ tada je $p_i \in T$, pa je $I(P_i) = 1$. Zatim, $\delta_{ij} = 1$ povlači da se literal P_i pojavljuje u elementarnoj disjunkciji C_j. Time dobivamo da $\delta_{ij} = 1$ povlači $I(C_j) = 1$. Analogno zaključujemo ako za neki $i \in \{1, \ldots, k\}$ postoji j tako da je $\gamma_{ij} = 1$.

Na kraju još samo kratko komentirajmo da je opisana redukcija polinomne vremenske složenosti. Za svaku 3–knf F tablica brojeva p_i, n_i, c_j i d_j je veličine oko $(k + s)^2$. Svaku znamenku u tablici možemo lako odrediti pomoću formule F u polinomnom vremenu.

Iz teorema 3.23. znamo da je problem 3–SAT jedan NP–potpun problem. Iz propozicije 3.13. tada slijedi da je PROBLEM SUMA PODSKUPA jedan NP–potpun problem.

6. Na strani 82 u zadatku 7 definiran je sljedeći PROBLEM NEZAVISAN SKUP:

> *za zadani neusmjereni graf G i prirodan broj k odrediti da li graf G sadrži nezavisan skup veličine k.*

Dokažimo da je to NP–potpun problem. U teoremu 6.5. dokazali smo da je PROBLEM 3-OBOJIVOST jedan NP–potpun problem. Dokazujemo da vrijedi PROBLEM 3–OBOJIVOST $\leq_p$ PROBLEM NEZAVISAN SKUP. Neka je G proizvoljan graf s n čvorova. Označimo s H graf koji sadrži tri kopije grafa G, te su "izomorfni" čvorovi povezani bridom. Tri kopije grafa G u grafu H označimo s G_1, G_2 i G_3. Tvrdimo da vrijedi:

graf G je 3–obojiv ako i samo ako u grafu H postoji n–nezavisan podskup.

Pretpostavimo prvo da je graf G moguće obojati s 3 boje. Tada postoje $B_1, B_2, B_3 \subseteq G$ u parovima disjunktni tako da nikoja dva čvora iz B_i, za svaki $i \in \{1, 2, 3\}$, nisu povezana bridom. (Možemo zamisliti da su qv cvorovi iz B_1 obojani crvenom bojom, i B_2 plavom, a iz B_3 zelenom.) Ovo bojanje grafa G prirodno definira bojanje svakog od grafova G_1, G_2 i G_3. Za svaki $i \in \{1, 2, 3\}$ neka su $B_{i1}, B_{i2}, B_{i3} \subseteq G_i$ u parovima disjunktni te nikoja dva čvora iz B_{ij}, za svaki $j \in \{1, 2, 3\}$, nisu povezana bridom. (Možemo zamišljati da su čvorovi svakog skupa B_{i1} obojani crveno, skupa B_{i2} plavo, a čvorovi skupa B_{i3} su obojani zeleno). Tada je, primjerice,

$$B_{11} \cup B_{22} \cup B_{33},\ B_{12} \cup B_{23} \cup B_{31},\ B_{13} \cup B_{21} \cup B_{32}$$

jedno pravilno 3–bojanje grafa H. (Možemo zamisliti da su sada svi čvorovi skupa $B_{11} \cup B_{22} \cup B_{33}$ obojani crveno, skupa $B_{12} \cup B_{23} \cup B_{31}$ plavo, a čvorovi skupa $B_{13} \cup B_{21} \cup B_{32}$ su obojani zeleno). Neka je $B = B_{11} \cup B_{22} \cup B_{33}$. Očito

je $card(B) = n$, te nikoja dva čvora iz skupa B nisu povezana bridom. To znači da graf H sadrži jedan n–nezavisni podskup.

Dokažimo obrat. Neka graf H sadrži jedan n–nezavisni podskup B. Označimo $B'_i = G_i \cap B$, za svaki $i \in \{1, 2, 3\}$. Zatim, označimo s B_i izomorfnu kopiju skupa B'_i u grafu G, za svaki $i \in \{1, 2, 3\}$. Dokažimo da su skupovi B_1, B_2, B_3 u parovima disjunktni. Pretpostavimo da $x \in B_1 \cap B_2$. Označimo $x_1 \in B'_1$ i $x_\in B'_2$ izomorfne kopije čvora x. Budući da čvorovi x_1 i x_2 imaju isti orginal u grafu G tada iz definicije grafa H slijedi da su povezani bridom, a to je nemoguće jer $x_1, x_2 \in B$, te je B nezavisan skup. Budući da su B_1, B_2, B_3 međusobno disjunktni, te je $B'_1 \cup B'_2 \cup B'_3 = B$ jedan n–nezavisni podskup od H, tada je $B_1 \cup B_2 \cup B_3$ jedan n–člani podskup od G. Budući da je $card(G) = n$ tada je $G = B_1 \cup B_2 \cup B_3$. Očito nikoja dva čvora iz B_i, za svaki $i \in \{1, 2, 3\}$, nisu povezana bridom. To znači da B_1, B_2, B_3 definira jedno pravilono 3–bojanje grafa G. Dakle, graf G je 3–obojiv.

7. Dokazujemo da je `PROBLEM IS-KLIKA` jedan `NP`–potpun problem (vidi zadatak 8 na strani 83).

   `PROBLEM IS–KLIKA` glasi:

   > *za zadani neusmjereni graf G i prirodan broj k odrediti da li G sadrži kliku veličine k ili nezavisan skup veličine k.*

   Najprije dokazujemo da dani problem pripada klasi `NP`, a nakon toga dajemo polinomnu redukciju s `PROBLEM NEZAVISAN SKUP` za kojeg iz prošlog zadatka znamo da je `NP`–potpun.

   Tvrdnja 1. Vrijedi: `PROBLEM IS–KLIKA` $\in$ `NP`.
   Neka je $G = (V, E)$ graf, te k prirodan broj. Za proizvoljan $U \subseteq V$ možemo provjeriti je li certifikat ovako:

   1. Ako skup vrhova U sadrži kliku veličine k, prihvati.
   2. Ako skup vrhova U sadrži nezavisan skup veličine k, prihvati.
   3. Odbaci.

   Očito gornja procedura prihvaća (G, k, U) ako i samo ako skup U sadrži kliku ili nezavisan skup veličine k. Budući da `PROBLEM KLIKA` i `PROBLEM NEZAVISAN SKUP` pripadaju klasi `NP`, vremenska složenost 1. i 2. koraka je polinomna, tj. provjera certifikata je polinomna, pa je time tvrdnja dokazana.

   Tvrdnja 2. Vrijedi: `PROBLEM NEZAVISAN SKUP` $\leq_p$ `PROBLEM IS–KLIKA`.
   Neka je $G = (V, E)$ graf, te k prirodan broj. Označimo $n := |V|$. Konstruiramo graf $G' = (V', E')$ na sljedeći način:

- graf G' sadrži sve vrhove grafa G;
- graf G' sadrži sve bridove grafa G;
- grafu G' dodamo n novih izoliranih vrhova.

Odnosno, malo formalnije neka je $W = \{w_1, \ldots, w_n\}$ takav da je $W \cap V = \emptyset$. Zatim, neka je $V' = W \cup V$ i $E' = E$. Tvrdimo da vrijedi:

> graf G sadrži nezavisan skup veličine k ako i samo ako graf G' sadrži kliku veličine $n + k$ ili nezavisan skup veličine $n + k$.

Pretpostavimo prvo da graf G sadrži nezavisan skup U veličine k. Kako je $U \subseteq V$ i $W \setminus V = \emptyset$, iz definicije grafa G' slijedi da je skup $U \cup W$ nezavisan skup grafa G' veličine $n + k$.

Dokažimo sada obrat. Pretpostavimo da graf G' sadrži podskup U veličine $n + k$ koji je nezavisan skup ili je klika. Kako je $n + k > n = |V|$, i skup W je nezavisan, to U ne može biti klika. Dakle, U je nezavisan skup veličine $n + k$. Kako je skup W nezavisan skup veličine n, vrijedi $|U \cap V| \geq k$, drugim riječima barem k vrhova skupa U mora se nalaziti medu vrhovima originalnog grafa G. Stoga postoji $U' \subseteq U \cap V$, $|U'| \geq k$, te vrijedi da je U' nezavisan skup u grafu G (jer je svaki podskup nezavisnog skupa ponovno nezavisan).

8. Dokazujemo da je `PROBLEM POVEZAN DOMINANTAN SKUP` jedan `NP`–potpun problem (vidi zadatak 10 na strani 84).

   Kako bi iskazali `PROBLEM POVEZAN DOMINANTAN SKUP` ponavljamo prvo dvije definicije. Neka je $G = (V, E)$ neusmjereni graf. Za $D \subseteq V$ kažemo da je dominantan skup ako za svaki vrh $v \in V \setminus D$ postoji vrh $u \in D$ susjedan vrhu v, tj. takav da je $\{u, v\} \in E$.

   Neka su $G = (V, E)$ i $G' = (V', E')$ grafovi. Za graf G' kažemo da je induciran sa $V' \subseteq V$ ako je $E' = \{\{u, v\} : u, v \in V'\} \subseteq E$.

   Sada možemo iskazati `PROBLEM POVEZAN DOMINANTAN SKUP`:

   > *Za zadani neusmjereni graf G i prirodan broj k treba odrediti da li graf G sadrži dominantan skup D veličine najviše k tako da je podgraf koji je induciran sa D povezan.*

   Prvo ćemo pokazati da je `PROBLEM POVEZAN DOMINANTAN SKUP` u klasi `NP`, a zatim da je problem `PROBLEM POKRIVAČ BRIDOVA` polinomno reducibilan na `PROBLEM POVEZAN DOMINANTAN SKUP`. Kako znamo da je `PROBLEM POKRIVAČ BRIDOVA` jedan `NP`–potpun problem (vidi korolar 3.30.), time će slijediti tvrdnja zadatka.

   Tvrdnja 1. Vrijedi `PROBLEM POVEZAN DOMINANTAN SKUP` $\in$ `NP`.
   Neka je $G = (V, E)$ proizvoljan graf i $k \in \mathbb{N}$. Za proizvoljni podskup $U \subseteq V$ možemo ovako provjeriti je li to certifikat:

1. Ako je $|U| > k$ odbaci.
2. Ako graf $(U, (U \times U) \cap E)$ nije povezan, odbaci.
3. Za svaki vrh $v \in V \setminus U$ napravi sljedeće:
   (a) Prođi po svim bridovima incidentnim sa v
   (b) Ako ne postoji brid $\{v, u\}$ za neki $u \in U$, odbaci.
4. Prihvati.

Očito prethodna procedura prihvaća (G, k, U) ako i samo ako je $|U| \leq k$, te vrijedi da svaki vrh $v \in V \setminus U$ ima susjedan vrh iz U. Znamo da je `PROBLEM PUT` $\in$ `P`, pa je vremenska složenost 2. koraka polinomna (jer je $|U| \leq k$). Složenost 3. koraka je najviše $O(|V| \times |E|)$. Dakle, provjera je li U certifikat je polinomne vremenske složenosti.

Tvrdnja 2. Vrijedi `PROBLEM POKRIVAČ BRIDOVA` $\leq_p$ `PROBLEM POVEZAN DOMINANTAN SKUP`.
Neka je $G = (V, E)$ graf, te k prirodan broj. Konstruiramo graf $G' = (V', E')$ na sljedeći način:

1. Skup vrhova G' sadrži sve vrhove grafa G i svaka dva takva vrha su povezana bridom;

2. Za svaki brid $\{x, y\} \in E$, graf G sadrži vrh v_{xy} koji je povezan s vrhovima x i y. Takav vrh ćemo nazivati dodatni vrh.

To možemo formalno zadati ovako:

$$V' = V \cup \{v_{xy} : \{x, y\} \in E\}$$

$$E' = \{\{x, y\} : x, y \in V\} \cup \{\{x, v_{xy}\},\ \{y, v_{xy}\} : \{x, y\} \in E\}$$

Dokazat ćemo da vrijedi sljedeća tvrdnja:

> graf G ima pokrivač bridova veličine najviše k ako i samo ako graf G' ima povezan dominantan skup veličine najviše k.

Pretpostavimo prvo da graf G ima pokrivač bridova U. Možemo pretpostaviti da je $U \neq \emptyset$. (Zaista, kada bi $U = \emptyset$ to bi značilo da je graf G potpuno nepovezan graf. Po konstrukciji bi tada graf G' imao iste vrhove kao i G, te bi bio potpuno povezan. Kako je k prirodan broj (posebno $k \geq 1$), tada bi svaki jednočlan podskup od V predstavljao povezan dominantan skup veličine najviše k, te bi tvrdnja vrijedila.) Pokazat ćemo da je U i dominantan skup u grafu G'. Za svaki vrh $w \in V \setminus U$, po definiciji postoje dvije mogućnosti:

a) $w \in V$. Kako je $U \neq \emptyset$, to postoji $u \in U \subseteq V$ koji je po konstrukciji povezan s vrhom w. Dakle, za vrh w postoji susjedan vrh iz U.

b) w je dodatni vrh, tj. $w = v_{xy}$ za neki brid $\{x, y\} \in E$. Kako je U pokrivač bridova, slijedi da je $x \in U$ ili $y \in U$, pa iz definicije grafa G' slijedi da vrh w ima susjedni vrh iz U.

Time smo pokazali da je U dominantan skup u grafu G', a kako smo u grafu G' dodali sve bridove između vrhova iz V, te kako je $U \subseteq V$, to je U ujedno i povezan dominantan skup u grafu G'.

Pretpostavimo sada da graf G' sadrži povezan dominantan skup U veličine najviše k. Kako su jedini susjedi vrha v_{xy} vrhovi x i y, slijedi da barem jedan vrh trokuta $\{v_{xy}, x, y\}$ mora biti element skupa U. Možemo pretpostaviti da niti jedan od dodatnih vrhova v_{xy} nije u U. (Zaista, ako $v_{xy} \in U$, tada možemo maknuti b_{xy} iz U i zamijeniti ga sa x. Jedini vrhovi koji bi time mogli postati "nepokriveni" su sam vrh v_{xy} i njegovi susjedi, ali oni su "pokriveni" baš s vrhom x.)

Iz 2. koraka definicije grafa G' zaključujemo da skup U mora sadržavati barem jedan kraj svakog brida, te je stoga on i pokrivač bridova grafa G.

Preostalo je još komentirati da je konstrukcija grafa G' polinomna. Prvi korak konstrukcije je vremenske složenosti $O(|V|)$, dok je drugi korak vremenske složenosti $O(|E| \cdot |V|)$.

9. Na strani 101 dan je zadatak u vezi klase DP= $\{L$: postoji $L_1 \in$NP i postoji $L_2 \in$ co–NP takvi da je $L = L_1 \cap L_2\}$. Treba dokazati da jezici SAT-UNSAT, CRITICAL-SAT i UNIQUE SAT pripadaju klasi DP.

Ponovimo definiciju prvog jezika:

SAT-UNSAT= $\{(F, G)$: F, G su 3–knf, F je ispunjiva, a G nije ispunjiva $\}$.

U svrhu dokaza označimo: $L_1 = \{(F, G) : F, G$ su 3-knf, F je ispunjiva$\}$ i $L_2 = \{(F; G) : F, G$ su 3-knf, G nije ispunjiva$\}$. Tada očito vijedi: SAT-UNSAT $= L_1 \cap L_2$. Očito je jezik L_1 ekvivalentan problemu 3–SAT. Nedeterministički Turingov stroj za L_1 treba samo izbrisati drugi dio uređenog para s ulaza, što traje $O(n)$ vremena, i dalje se ponašati kao nedeterministički Turingov stroj za problem 3–SAT, pa vrijedi $L_1 \in$ NP. Dalje, jezik L_2^c je ekvivalentan jeziku L_1, pa vrijedi $L_2^c \in$ NP, a onda i $L_2 \in$ co–NP. Zaključujemo da je SAT-UNSAT u klasi DP.

Drugi jezik koji promatramo je definiran ovako:

CRITICAL-SAT= $\{F$: F je knf koja nije ispunjiva, ali brisanjem bilo koje njene elementarne disjunkcije dobivena formula postaje ispunjiva $\}$.

Kako bi dokazali da dani jezik pripada klasi DP uzmimo neku knf $F \equiv F_1 \wedge \ldots \wedge F_k$, gdje je svaka formula F_i jedna elementarna disjunkcija. Za svaki $i \in \{1, \ldots, k\}$ sa F_i označimo formulu dobivenu iz F brisanjem njene i–te elementarne disjunkcije. Označimo $L_2 = \{F : F$ je knf, za svaki i je formula F_i ispunjiva$\}$. Očito vrijedi CRITICAL-SAT $=$ SAT$^c \cap L_2$. Znamo da vrijedi SAT$^c \in$ co–NP.

Kako bismo vidjeli da je $L_2 \in$ NP dovoljno je pokazati da za svaku knf $F \in L_2$ postoji certifikat. Kako je $|F_i| \leq |F| = n$ i $F_i \in$ SAT, za svaku knf F_i postoji certifikat c_i duljine $O(n^k)$. Primijetimo da je broj različitih formula F_i manji od n jer je svaka elementarna disjunkcija kodirana barem jednim znakom. Tada konkatenacijom pripadnih certifikata c_i za elementarnu disjunkciju F_i dobivamo certifikat za knf F (u odnosu na L_2) duljine $O(n^{k+1})$. Polinomni deterministički Turingov stroj koji uzima certifikat i formulu F konstruira sve knf F_i (najviše ih je n) i za svakog od njih uzima pripadni certifikat ci te se dalje ponaša kao determinstički Turingov stroj za problem SAT iz teorema o certifikatu – to vodi na povećanje vremenske složenosti za jedan red veličine, što je i dalje polinomno. Slijedi da vrijedi $L_2 \in$ NP, pa jezik CRITICAL-SAT pripada klasi DP.

Treći jezik koji promatramo je definiran sa:

$$\text{UNIQUE SAT} = \{F : \ F \text{ je knf za koju postoji jedinstvena interpretacija } I \text{ takva da je } I(F) = 1\}.$$

Kako bi dokazali da i taj jezik pripada klasi DP, prvo se prisjetimo da je na strani 72 u zadatku 5 istaknuto da je problem DOUBLE–SAT$= \{F : F$ je knf za koju postoje najmanje dvije interpretacije za koju je ona istinita$\}$ jedan NP–potpun problem. Tada posebno imamo DOUBLE–SAT$^c \in$ co–NP. Iz toga slijedi UNIQUE-SAT$=$DOUBLE–SAT$^c \cap$ SAT, tj. UNIQUE-SAT $\in$ DP.

6.3 Deskriptivna teorija složenosti

Deskriptivna teorija složenosti nastala je razvojem teorije konačnih modela. Primjenom matematičke logike pokušava se opisati neke probleme u teoriji složenosti, te ih riješiti. Sadržaj smo podijelili na tri dijela: Teorija konačnih modela, Hvatanje klasa složenosti i Logike fiksnih točaka. U prvom dijelu želimo istaknuti pojmove i činjenice koje su nam nužne za daljnje izlaganje. U drugim dijelu želimo posebno istaknuti Faginov teorem koji daje "logičku" karakterizaciju klase NP. U trećem dijelu definiramo tri logike fiksne točke koje "logički" karakteriziraju klasu P.

Ovdje nećemo ponavljati pojmove kao što su: signatura, σ–struktura, logika prvog reda i slično. Zatim, pretpostavljamo da su čitaocu poznati teorem kompaktnosti, teorem potpunosti i Löwenheim, Skolemovi teoremi za logiku prvog reda. Svi detalji mogu se pogledati, primjerice, u [25].

6.3.1 Teorija konačnih modela

Teorija konačnih modela – skraćeno FMT (eng. Finite Model Theory), uključuje: klasičnu teoriju modela, kombinatoriku i teoriju složenosti. Neki dijelovi FMT jako su povezani s računarstvom, posebno deskriptivna teorija složenosti. Zapravo, smatra se da bi FMT trebala biti logika za računarstvo. Dokazano je da mnogi klasični rezultati teorije modela ne vrijede na klasi svih konačnih struktura. Primjerice, ne vrijede teoremi kompaktnosti i potpunosti, te mnogi teoremi o očuvanju i lema interpolacije.

Propozicija 6.6. *Teorem kompaktnosti ne vrijedi za FMT, tj. postoji skup S rečenica logike prvog reda čiji svaki konačan podskup ima konačan model, ali za S ne postoji konačan model.*

Skica dokaza. Za svaki $n \in \mathbb{N}$ definiramo formule φ_n ovako:

$$\varphi_n \equiv \text{kardinalni broj nosača strukture je različit od } n.$$

Zatim definiramo skup formula $S = \{\varphi_n : n \in \mathbb{N}\}$.

Svaki konačan podskup S' od S ima konačan model (ako $\varphi_n \in S \setminus S'$ tada je svaka struktura s točno n elemenata model za S'). No, očito skup formula S nema konačan model. Q.E.D.

Teorem potpunosti ne vrijedi za FMT. Iz teorema potpunosti za logiku prvog reda slijedi da je skup V svih valjanih rečenica rekurzivno prebrojiv. Nevaženje teorema potpunosti za FMT slijedi iz Trachtenbrotovog teorema.

Teorem 6.7. (B. Trachtenbrot, 1950.) *Skup V_{fin} svih rečenica koje su istinite na svim konačnim strukturama nije rekurzivno prebrojiv.*

Skica dokaza. Označimo: S_{fin} = skup svih rečenica za koje postoji konačan model. Lako je dokazati da je skup S_{fin} rekurzivno prebrojiv. Iz nerješivosti halting problema slijedi da skup $\{\langle T\rangle \ : \ T$ je Turingov stroj koji staje za svaki ulaz$\}$ nije rekurzivan. Za svaki Turingov stroj T moguće je konstruirati rečenicu φ_T tako da vrijedi: Turingov stroj T staje ako i samo ako $\varphi_T \in S_{fin}$.

Iz toga slijedi da skup S_{fin} nije rekurzivan. Očito vrijedi: $S^c_{fin} = \{\varphi : \neg\varphi \in V_{fin}\}$. Pretpostavimo da je skup V_{fin} rekurzivno prebrojiv. Tada je očito skup S^c_{fin} rekurzivno prebrojiv. Iz Postovog teorema slijedi da je skup S_{fin} rekurzivan, čime je dobivena kontradikcija. Q.E.D.

Definicija 6.8. *Logički sustav (ili logika) $\mathcal{L}$ je uređeni par $(L, \vDash)$ funkcije L i binarne relacije $\vDash$. Svakom skupu nelogičkih simbola σ funkcija L pridružuje $L[\sigma]$, skup σ–rečenica od $\mathcal{L}$. Pri tome zahtijevamo da budu ispunjena sljedeća četiri uvjeta:*

a) ako $\sigma_0 \subseteq \sigma_1$ tada $L[\sigma_0] \subseteq L[\sigma_1]$;

b) ako $\mathfrak{A} \vDash \varphi$ tada postoji σ tako da $\mathfrak{A}$ σ-struktura i $\varphi \in L[\sigma]$;

c) ako $\mathfrak{A} \vDash \varphi$ *i* $\mathfrak{A} \cong \mathfrak{B}$ *(tj.* $\mathfrak{A}$ *i* $\mathfrak{B}$ *su izomorfne strukture), tada* $\mathfrak{B} \vDash \varphi$;

d) ako $\sigma_0 \subseteq \sigma_1$, $\varphi \in L[\sigma_0]$ *i* $\mathfrak{A}$ *je* σ_1*–struktura, tada vrijedi:* $\mathfrak{A} \vDash \varphi$ *ako i samo ako* $\mathfrak{A}|_{\sigma_0} \vDash \varphi$ *(pri čemu je s* $\mathfrak{A}|_{\sigma_0}$ *označen* σ_0*–redukt strukture* $\mathfrak{A}$*).*

Primjer 6.9. *Primjerice, u slučaju logike prvog reda, koju označavamo sa FO, (eng. first-order) funkcija L iz prethodne definicije svakom skupu nelogičkih simbola* σ *pridružuje skup* σ*–rečenica prvog reda, dok je* $\vDash$ *uobičajena relacija istinitosti između struktura i rečenica prvog reda koja predstavlja Tarskijevu semantiku logike prvog reda.*

Primjer 6.10. *Sa* $\mathcal{L}_{\infty\omega}$ *označavamo beskonačnu logiku koja se definira na isti način kao i logika prvog reda uz razliku što se dopuštaju disjunkcije proizvoljnog skupa formula. Formalno, skup* σ*–formula logike* $\mathcal{L}_{\infty\omega}$ *dan je sljedećom rekurzivnom definicijom:*

a) sve atomarne σ*–formule logike prvog reda su* σ*-formule logike* $\mathcal{L}_{\infty\omega}$*;*

b) ako je φ *neka* σ*-formula, tada je i* $\neg\varphi$ *jedna* σ*–formula;*

c) ako je φ *neka* σ*–formula, a x varijabla, tada je i* $\exists x\varphi$ *jedna* σ*–formula;*

d) ako je Ψ *proizvoljan skup* σ*–formula, tada je i* $\bigvee \Psi$ *jedna* σ*–formula.*

Semantika logike $\mathcal{L}_{\infty\omega}$ dobiva se proširenjem definicije semantike logike prvog reda na način da se $\bigvee \Psi$ interpretira kao disjunkcija po svim formulama iz Ψ, tj. $\mathfrak{A} \vDash \bigvee \Psi$ ako i samo ako za neki $\psi \in \Psi$ je $\mathfrak{A} \vDash \psi$. Uvedemo li oznaku $\bigwedge \Psi := \neg \bigvee \{\neg\psi \mid \psi \in \Psi\}$, tada se $\bigwedge \Psi$ intepretira kao konjunkcija po svim formulama iz Ψ. Löwenheim–Skolemov teorem vrijedi za beskonačnu logiku, a teorem kompaktnosti ne.

Primjer 6.11. *Logika drugog reda, koju označavamo sa SO (eng. second-order), predstavlja proširenje logike prvog reda u kojoj je dopuštena kvantifikacija relacijskih varijabli (koje su uz individualne varijable također dio alfabeta SO logike). Formalno, skup* σ*–formula SO logike definira se rekurzivno koristeći pravila za formiranje formula logike prvog reda uz sljedeći dodatak:*

- *ako je X neka k mjesna relacijska varijabla, i* $t_1, \ldots t_k$ *termi, tada je* $Xt_1 \ldots t_k$ *formula logike SO;*
- *ako je* φ *neka SO–formula i X relacijska varijabla, tada je i* $\exists X\varphi$ *formula.*

Neka je dana *SO*–formula $\varphi(\vec{x}, \vec{X})$, σ–struktura $\mathfrak{A}$, $\vec{a} \in |\mathfrak{A}|^k$, te konačan niz relacija $\vec{R}$ nad $|\mathfrak{A}|$. Tada pišemo $\mathfrak{A} \vDash \varphi[\vec{a}, \vec{R}]$ ako i samo ako $\vec{a}$ zajedno sa $\vec{R}$ zadovoljava $\varphi(\vec{x}, \vec{X})$ na strukturi $\mathfrak{A}$. Za logiku *SO* ne vrijedi teorem kompaktnosti, a ni Löwenheim–Skolemov teorem "na dolje".

Primjer 6.12. *Slaba logika drugog reda, koju označavamo sa SO^w, razlikuje se od logike drugog reda SO samo u definiciji istinitosti formula oblika $\exists X\varphi$.*

Neka je $\mathfrak{M}$ neka σ–struktura, te v neka valuacija na $\mathfrak{M}$. Neka je X neka n–mjesna relacijska varijabla, te neka je φ neka formula logike drugog reda. Tada za logiku SO^w definiramo:

$$\mathfrak{M} \models_v \exists X\varphi \quad \text{ako i samo ako} \quad \text{postoji konačan } S \subseteq |\mathfrak{M}|^n \text{ tako da } \mathfrak{M} \models_v \varphi[S].$$

Za logiku SO^w ne vrijedi teorem kompaktnosti, ali vrijedi Löwenheim–Skolemov teorem.

Primjer 6.13. *Logički sistem L_Q^N dobivamo dodavanjem alfabetu logike prvog reda novog kvantifikatora Qx, te je interpretacija formule oblika $Qx\varphi$ definirana sa: "postoji neprebrojivo mnogo x koji zadovoljavaju formulu φ."*

Logički sistem L_Q^N je izražajniji od logike prvog reda. Npr. pojam "najviše prebrojiv" možemo definirati formulom $\neg Qx(x = x)$. Za logiku L_Q^N ne vrijedi teorem kompaktnosti, a ni Löwenheim–Skolemov teorem.

Primjer 6.14. *Logički sistem L_Q^P dobivamo dodavanjem alfabetu logike prvog reda novog kvantifikatora Qx. Interpretacija formule oblika $Qx\varphi$ je definirana sa: "postoji prebrojivo mnogo x koji zadovoljavaju formulu φ."*

Za logiku L_Q^P ne vrijedi teorem kompaktnosti, ali vrijedi Löwenheim–Skolemov teorem.

Primjer 6.15. *Kod logičkog sistema L_Q^K interpretacija formule oblika $Qx\varphi$ je definirana sa: "postoji samo konačno mnogo x koji zadovoljavaju formulu φ."*

Za sistem L_Q^K ne vrijedi teorem kompaktnosti, ali vrijedi Löwenheim–Skolemov teorem.

Primjer 6.16. *Kod logičkog sistema L_Q^D interpretacija formule oblika $Qx\varphi$ je definirana sa: "postoje barem dva različita x koji zadovoljavaju formulu φ."*

Za sistem L_Q^D ne vrijedi teorem kompaktnosti, a ni Löwenheim–Skolemov teorem.

U sljedećoj tablici rezimiramo, te navodimo neke nove, činjenice o logikama koje smo definirali.

Logički sistem	Löwenheim–Skolemov teorem	Teorem kompaktnosti	
		Općenito	za prebrojive skupove formula
FO	+	+	+
SO	–	–	–
SO^{W}	+	–	–
$L_{\infty\omega}$	+	–	–
L_Q^N	–	–	+
L_Q^P	+	–	–
L_Q^K	+	–	–
L_Q^D	–	–	–

Provedena razmatranja, odnosno dani primjeri, prije svega trebali bi ilustrirati definiciju pojma logike koji ćemo dalje koristiti. No, želimo naglasiti da su ti primjeri motivacija za proučavanja koja se nazivaju apstraktna teorija modela. Osnovni rezultat apstraktne teorije modela je Lindströmov prvi teorem (vidi [25]).

6.3.2 Hvatanje klasa složenosti

Sada želimo navesti veze između pojednih logika i klasa složenosti. U tu svrhu uvodimo još neke pojmove.

Domena $\mathcal{D}$ je svaka klasa konačnih struktura koja je zatvorena za izomorfizme. Želimo naglasiti da se u nekoj domeni mogu nalaziti strukture različitih signatura. Ako je $\mathcal{D}$ neka domena i σ signatura, tada sa $\mathcal{D}[\sigma]$ označavamo klasu svih σ–struktura koje pripadaju domeni $\mathcal{D}$.

Definicija 6.17. *Problem ispunjivosti za neku logiku $\mathcal{L}$ na domeni $\mathcal{D}$ definiran je na sljedeći način:*

> *za danu rečenicu ψ logike $\mathcal{L}$ potrebno je odrediti postoji li struktura $\mathfrak{A} \in \mathcal{D}$ takva da vrijedi $\mathfrak{A} \vDash \psi$.*

Iako je problem ispunjivosti od fundamentalnog značaja u mnogim područjima logike i njezine primjene, a isto tako neki njegovi specijalni slučajevi (primjerice SAT i TQBF) od neprijeporne važnosti u teoriji računske složenosti, pokazuje se kako problem ispunjivosti općenito ipak ne igra odlučujuću ulogu u teoriji konačnih modela.

S druge strane, pokazat će se kako središnje mjesto u teoriji konačnih modela zauzima problem verifikacije modela kojeg opisujemo sljedećom definicijom.

Definicija 6.18. *Problem verifikacije modela za logiku $\mathcal{L}$ i domenu $\mathcal{D}$ definiran je na sljedeći način:*

> *za danu rečenicu ψ logike $\mathcal{L}$ i strukturu $\mathfrak{A} \in \mathcal{D}$ potrebno je odrediti vrijedi li $\mathfrak{A} \vDash \psi$.*

U daljnjim razmatranjima koristit ćemo naziv "klasa složenosti". Pod time mislimo jednostavno na neki skup jezika.

Sada nam je cilj definirati najvažniji pojam ove točke, a to je pojam hvatanja klase složenosti s nekom logikom. U tu svrhu prvo ćemo definirati kodiranje klasa struktura, te ćemo objasniti što znači da neka klasa struktura pripada nekoj klasi složenosti.

Za danu signaturu σ sa $Ord[\sigma]$ označavamo klasu svih uređenih konačnih σ–struktura, tj.

$$Ord[\sigma] = \{(\mathfrak{A}, <) \mid \mathfrak{A} \in Fin[\sigma] \text{ i } < \text{ je linearni uređaj na } |\mathfrak{A}|\},$$

gdje smo sa $Fin[\sigma]$ označili klasu svih konačnih σ–struktura.

Za svaki $k \in \mathbb{N}$ i $(\mathfrak{A}, <) \in Ord[\sigma]$, pri čemu je $\mid \mathfrak{A} \mid = n$, smatramo da je skup $\mid \mathfrak{A} \mid^k$ uređen leksikografski, te ga poistovjećujemo sa skupom $\{0, 1, \ldots, n^k - 1\}$.

Funkciju $code : Ord[\sigma] \to \Sigma^*$ (gdje je Σ neki konačan alfabet) nazivamo funkcijom kodiranja ukoliko ona identificira izomorfne strukture, polinomno je omeđena, definabilna je u logici prvog reda, te omogućava efikasnu evaluaciju istinitosnih vrijednosti atomarnih formula. Funkciju kodiranja formalno definiramo u sljedećoj definiciji.

Definicija 6.19. *Neka je Σ neki konačan alfabet i σ neka signatura. Funkcija kodiranja je svaka funkcija $code : Ord[\sigma] \to \Sigma^*$ koja zadovoljava sljedeća svojstva:*

a) $code(\mathfrak{A}, <) = code(\mathfrak{B}, <)$ *ako i samo ako* $(\mathfrak{A}, <) \cong (\mathfrak{B}, <)$;

b) $|code(\mathfrak{A}, <)| \leq p(card(|\mathfrak{A}|))$ *za neki polinom* p;

c) Za svaki $k \in \mathbb{N}$ *i svaki simbol* $s \in \Sigma$ *postoji formula logike prvog reda* $\beta_s(x_1, \ldots, x_k)$ *nad signaturom* $\sigma \cup \{<\}$ *takva da za svaku uređenu strukturu* $(\mathfrak{A}, <) \in Ord[\sigma]$ *i sve* $\vec{a} \in |\mathfrak{A}|^k$ *vrijedi sljedeća ekvivalencija:*

$$(\mathfrak{A}, <) \vDash \beta_s[\vec{a}] \quad \textit{ako i samo ako} \quad s \textit{ je } \vec{a}\textit{-ti simbol riječi } code(\mathfrak{A}, <);$$

d) Za dani kod $code(\mathfrak{A}, <)$*, te relacijski simbol* $R \in \sigma$ *i odgovarajuću reprezentaciju konačnog niza* $\vec{a}$ *moguće je efikasno odlučiti da li vrijedi* $\mathfrak{A} \vDash R\vec{a}$.

Primjer 6.20. *Sada ćemo opisati jednu često korištenu funkciju kodiranja nad alfabetom* $\Sigma = \{0, 1\}$. *Neka je* $\mathfrak{A} = (|\mathfrak{A}|, R_1, \ldots, R_t)$ *neka* σ*–struktura kardinaliteta* n*, te* $<$ *linearni uređaj na* $|\mathfrak{A}|$. *Označimo sa* p *maksimalnu mjesnost relacije* $R_1, \ldots, R_t$. *Svakoj relaciji* R *mjesnosti* k *pridružujemo sljedeću riječ:*

$$\chi(R) = w_0 \ldots w_{n^k-1} 0^{n^p - n^k} \in \{0, 1\}^{n^p},$$

gdje je $w_i = 1$ *ukoliko se* i*–ta* k*–torka iz* $|\mathfrak{A}|^k$ *nalazi u* R*, odnosno* $w_i = 0$ *ukoliko to nije slučaj. Funkciju kodiranja sada definiramo ovako:*

$$code(\mathfrak{A}, <) = 1^n 0^{n^l - n} \chi(R_1) \ldots \chi(R_t).$$

Nije teško pokazati da na ovaj način definirana funkcija kodiranja zadovoljava sve uvjete iz definicije.

U nastavku teksta za svaku signaturu σ odaberimo neku funkciju kodiranja, te smatramo da je svaka uređena struktura nad signaturom σ kodirana upravo tom funkcijom kodiranja. Svakoj neuređenoj strukturi $\mathfrak{A}$ pridružujemo skup:

$$\{code(\mathfrak{A}, <) \mid \text{relacija } < \text{ je linearni uređaj na } |\mathfrak{A}|\}.$$

Stoga kada kažemo da Turingov stroj M odlučuje klasu σ–struktura $\mathcal{K}$ zapravo mislimo da M odlučuje skup svih kodova struktura u $\mathcal{K}$, odnosno jezik dan s: $\{code(\mathfrak{A}, <) \mid \mathfrak{A} \in \mathcal{K}$ i $<$ je linearni uređaj na $|\mathfrak{A}|\}$.

Na taj način ima smisla govoriti o složenosti klase struktura $\mathcal{K}$, promatrajući pri tome u kojoj klasi složenosti se nalazi pripadni skup kodova. Na isti način moći ćemo govoriti i o složenosti klase modela neke logičke rečenice.

Vidjeli smo kako se klase struktura mogu kodirati pomoću jezika, no, isto tako svaki se jezik $L \subseteq \Sigma^*$ može smatrati klasom struktura nad signaturom $\{<\} \cup \{P_a \mid a \in \Sigma\}$, gdje su P_a unarni relacijski simboli.

Naime, svaka riječ $w = w_0 \dots w_{m-1} \in \Sigma^*$ opisuje strukturu $\mathfrak{A}_w$ čiji je nosač $\{0, \dots, m-1\}$, standardnom interpretacijom relacije $<$, te interpretacijom od P_a definiranom sa $P_a^{\mathfrak{A}_w} = \{i \mid w_i = a\}$.

Definicija 6.21. *Neka je $\mathcal{L}$ logika, $\mathcal{C}$ klasa složenosti i $\mathcal{D}$ neka domena konačnih struktura. Kažemo da logika $\mathcal{L}$ hvata klasu složenosti $\mathcal{C}$ na domeni $\mathcal{D}$ ako vrijedi:*

a) za svaku signaturu σ i svaku rečenicu $\psi \in \mathcal{L}[\sigma]$ problem verifikacije modela za ψ na $\mathcal{D}[\sigma]$ nalazi se u klasi složenosti $\mathcal{C}$.

b) za svaku klasu struktura $\mathcal{K} \subseteq \mathcal{D}[\sigma]$ koja se nalazi u klasi složenosti $\mathcal{C}$ postoji rečenica $\psi \in \mathcal{L}[\sigma]$ takva da je $\mathcal{K} = \{\mathfrak{A} \in \mathcal{D}[\sigma] \mid \mathfrak{A} \vDash \psi\}$.

Egzistencijalna formula logike drugog reda, tj. Σ_1^1–formula, je formula oblika: $\exists X_1 \dots \exists X_n \varphi$, gdje su X_i relacijske varijable, a φ je formula logike prvog reda. Egzistencijalna logika drugog reda, koju označavamo sa Σ_1^1, je fragment logike drugog reda u kojoj dozvoljavamo samo Σ_1^1–formule. Sada nam je cilj dokazati Faginov teorem koji govori da logika Σ_1^1 hvata klasu NP na domeni svih konačnih struktura. Prvo dajemo nekoliko primjera Σ_1^1–formula koje "opisuju" neke NP probleme.

Primjer 6.22. *Svojstvo 3–obojivosti grafa moženo izraziti sljedećom Σ_1^1–formulom:*

$$\begin{aligned}\exists R \exists B \exists G \Big(\quad & \forall x (Rx \vee Bx \vee Gx) \wedge \\ & \forall x \Big(\neg(Rx \wedge Bx) \ \wedge \ \neg(Bx \wedge Gx) \wedge \\ & \neg(Rx \wedge Gx) \Big) \wedge \\ & \forall x \forall y \Big(Exy \to (\neg(Rx \wedge Ry) \wedge \\ & \neg(Bx \wedge By) \ \wedge \ \neg(Gx \wedge Gy) \Big) \Big)\end{aligned}$$

Prisjetimo se još da znamo da vrijedi PROBLEM *3–*OBOJIVOSTI $\in$ NP*. Štoviše, to je jedan* NP*–potpun problem.*

Primjer 6.23. *Činjenicu da je neki graf Hamiltonov*[1] *možemo izraziti sljedećom* Σ_1^1 *formulom:*

$$\exists R\Bigg(\quad R \;\; \textit{je linearni uređaj} \;\; \wedge \\ \forall x E(x, x+1) \wedge E(max, min) \Bigg)$$

Teorem 6.24. (R. Fagin, 1973.) *Logika* Σ_1^1 *hvata klasu* NP *na domeni svih konačnih struktura. Odnosno, vrijedi* $\Sigma_1^1 = NP$.

Skica dokaza. Prvo uvodimo neke relacije.

- Neka je $X_q := \{\vec{t} \in |\mathfrak{A}|^k \mid$ u trenutku $\vec{t}$ stroj M se nalazi u stanju $q\}$, za svako stanje $q \in Q$;
- Neka je $Y_s := \{(\vec{t}, \vec{a}) \in |\mathfrak{A}|^k \times |\mathfrak{A}|^k \mid$ u trenutku $\vec{t}$ imamo da $\vec{a}$–ti element trake sadržava simbol $s\}$, za svaki simbol $s \in \Sigma$;
- Neka je $Z := \{(\vec{t}, \vec{a}) \in |\mathfrak{A}|^k \times |\mathfrak{A}|^k \mid$ u trenutku $\vec{t}$ glava od M nalazi se na poziciji $\vec{a}\}$.

Formula $START$ osigurava da je konfiguracija stroja M u trenutku $\vec{t} = 0$ početna konfiguracija $C_0(\mathfrak{A}, <)$ s kodom $code(\mathfrak{A}, <)$ na ulazu. Pri tome, obzirom da je *code* funkcija kodiranja, $code(\mathfrak{A}, <)$ je reprezentiran formulom prvog reda $\beta_\sigma(\vec{x})$, pa stavljamo:

$$\text{START} := X_{q_0}(\vec{0}) \wedge Z(\vec{0}, \vec{0}) \wedge \bigwedge_{s \in \Sigma} \Big(\beta_s(\vec{x}) \to Y_s(\vec{0}, \vec{x})\Big).$$

Formula $COMPUTE$ opisuje tranziciju stroja M iz jedne konfiguracije u iduću, a dobivamo je kao konjunkciju formula $NOCHANGE$ i $CHANGE$. Formula $NOCHANGE$ izražava da se sadržaji ćelija iznad kojih se ne nalazi glava od M ne mijenjaju pri prelasku iz jedne konfiguracije u iduću, dok $CHANGE$ osigurava da su relacije X_q, Y_s i Z usklađene s tranzicijskom funkcijom stroja M:

$$NOCHANGE := \bigwedge_{s \in \Sigma} \Big(Y_s(\vec{t}, \vec{x}) \wedge (\vec{y} \neq \vec{x}) \wedge Z(\vec{t}, \vec{y}) \to Y_s(\vec{t}+1, \vec{x})\Big),$$

$$CHANGE := \bigwedge_{q \in Q,\ s \in \Sigma} \Big(PRE_{q,s} \to \bigvee_{(q', s', m) \in \delta(q,s)} POST_{q', s', m}\Big).$$

Pri tome smo sa $PRE_{q,s}$ i $POST_{q',s',m}$ označili slijedeće formule:

[1] Za graf (G, R), gdje je $G = \{v_1, \ldots, v_n\}$, kažemo da je Hamiltonov ako postoji permutacija π skupa svih čvorova tako da vrijedi $v_{\pi(1)} R \ldots R v_{\pi(n)} R v_{\pi(1)}$.

$$PRE_{q,s} \;:=\; X_q(\vec{t}) \wedge Z(\vec{t},\vec{x}) \wedge Y_s(\vec{t},\vec{x}),$$

$$POST_{q',s',m} \;:=\; X_{q'}(\vec{t}+1) \wedge Y_{s'}(\vec{t}+1,\vec{x}) \wedge Z(\vec{t}+1,\vec{x}+m).$$

Na koncu imamo još formulu: $END := \neg X_{q_{NE}}(\vec{t})$, kojom se osigurava prihvaćanje stroja M isključujući mogućnost da se nađe u odbacujućem stanju.

Sada definiramo $\psi_M := START \;\wedge\; COMPUTE \;\wedge\; END$.

Tvrdnja 1. Ako stroj M prihvaća $code(\mathfrak{A},<)$, tada je $(\mathfrak{A},<) \vDash \exists\overline{X}\psi_M$.
Ova tvrdnja slijedi direktno iz konstrukcije rečenice ψ_M, budući za bilo koje prihvaćajuće izračunavanje stroja M na kodu $code(\mathfrak{A},<)$ intendirana interpretacija od $\overline{X}$ zadovoljava ψ_M.

Tvrdnja 2. Ako je $(\mathfrak{A},<,\overline{X}) \vDash \psi_M$, tada stroj M prihvaća $code(\mathfrak{A},<)$. Q.E.D.

Univerzalna formula logike drugog reda, tj. Π^1_1–formula, je svaka formula oblika $\forall R_1 \ldots \forall R_n \varphi$, gdje su X_i relacijske varijable, a φ je formula logike prvog reda. Univerzalna logika drugog reda, koju označavamo sa Π^1_1, je fragment logike drugog reda u kojoj dozvoljavamo samo Π^1_1–formule. Sljedeći korolar je očita posljedica Faginovog teorema.

Korolar 6.25. *Logika Π^1_1 hvata klasu* CO–NP *na klasi svih konačnih struktura.*

Zanimljivo je da Cook–Levinov teorem kojeg smo bili već prije dokazali, sada slijedi kao jednostavan korolar Faginovog teorema.

Sada dajemo neke rezultate o hvatanju klase složenosti P. Bili smo već definirali pojam Hornove klauzule u logici sudova. Sada to isto radimo za logiku prvog reda.

Definicija 6.26. *Hornova klauzula nad skupom relacijskih simbola $R_1, \ldots, R_m$ je formula oblika*

$$\beta_1 \wedge \ldots \wedge \beta_n \to H,$$

gdje je za svaki $j \in \{1,\ldots,n\}$ svaka formula β_j ili atomarna formula oblika $R_k\bar{z}$ za neko $k \in \{1,\ldots,m\}$ ili formula prvog reda koja ne sadrži $R_1,\ldots,R_m$, dok je H ili atomarna formula oblika $R_k\bar{z}$ ili logička konstanta $\bot$.

Definicija 6.27. *Hornova logika drugog reda, koju označavamo s* SO–HORN*, je skup svih rečenica oblika:*

$$Q_1 R_1 \ldots Q_m R_m \forall y_1 \ldots \forall y_s \bigwedge_{i=1}^{t} C_i,$$

gdje su $Q_i \in \{\exists,\forall\}$, R_i relacijske varijable, a C_i Hornove klauzule nad $R_1,\ldots,R_m$.

Definicija 6.28. *Sa* Σ_1^1–HORN *označavamo egistencijalni fragment logike* SO–HORN*, tj. skup svih* SO–HORN*–rečenica u kojima su svi kvantifikatori drugog reda egzistencijalni.*

Teorem 6.29. (E. Grädel, 1991.) *Logike* SO–HORN *i* Σ_1^1–HORN *hvataju klasu* `P` *na klasi svih uređenih konačnih struktura.*

Kako bi se dokazao Grädelov teorem, prvo se pokaže da ukoliko je Turingov stroj deterministički, tada se formula konstruirana u dokazu Faginovog teorema može prebaciti u ekvivalentnu Σ_1^1–formulu. Detalje dokaza možete vidjeti u [14].

Pretpostavka o uređenosti struktura je nužna, jer se može pokazati da linearne uređaje nije moguće definirati Hornovim formulama.

6.3.3 Logike fiksne točke

Logike fiksne točke su proširenja logike prvog reda koja dopuštaju rekurzivne definicije. U ovoj točki definirat ćemo tri primjera logike fiksne točke: IFP, PFP i LFP. Na kraju točke ćemo istaknuti dva teorema iz kojih se vidi važnost logika fiksne točke.

Primjer 6.30. *Logika prvog reda nije zatvorena na rekurzivne definicije. Kako bi to pokazali, promotrimo proizvoljan neusmjereni graf* $G = (V, E)$*. Za proizvoljan* $v \in V$ *rekurzivno definiramo niz podskupova skupa čvorova* V *ovako:*

$$C_0(v) = \{v\},$$

$$C_{n+1} = \{x : \exists y \in C_n(v),\ \{x, y\} \in E\}.$$

Za svaki $n \in \mathbb{N}$ *skup* $C_n(v)$ *je definabilan u logici prvog reda na strukturi* G*. No, očito da skup* $\cup_{n\in\mathbb{N}} C_n(v)$ *(tzv. komponenta povezanosti čvora* v*) nije definabilan u logici prvog reda na strukturi* G*. (Primijetimo da je* $\cup_{n\in\mathbb{N}} C_n(v)$ *fiksna točka prije definirane operacije).*

Neka je σ neka signatura, te neka je P neki k–mjesni relacijski simbol koji ne pripada σ. Neka je $\varphi(x_1, \ldots, x_k)$ neka $\sigma \cup \{P\}$–formula. Za svaku $\sigma \cup \{P\}$–strukturu $\mathfrak{M}$ definiramo novu $\sigma \cup \{P\}$–strukturu $\mathcal{O}_{\varphi,P}(\mathfrak{M})$ ovako:

$$|\mathcal{O}_{\varphi,P}(\mathfrak{M})| := |\mathfrak{M}|;$$

$$s^{\mathcal{O}_{\varphi,P}(\mathfrak{M})} := s^{\mathfrak{M}}, \quad \text{za svaki} \quad s \in \sigma;$$

$$P^{\mathcal{O}_{\varphi,P}(\mathfrak{M})} := P^{\mathfrak{M}} \ \cup\ \{\vec{a} \in |\mathfrak{M}|^k :\ \mathfrak{M} \models \varphi[\vec{a}]\}.$$

Neka je $\mathfrak{N}$ neka σ–struktura, te P neki k–mjesni relacijski simbol, takav da $P \notin \sigma$. Strukturu $\mathfrak{N}$ možemo promatrati i kao $\sigma \cup \{P\}$–strukturu ako definiramo $P^{\mathfrak{N}} = \emptyset$. Za proizvoljnu $\sigma \cup \{P\}$–formulu φ definiramo niz struktura ovako:

$$\mathfrak{N}_0 := \mathfrak{N};$$

$$\mathfrak{N}_{k+1} := \mathcal{O}_{\varphi,P}(\mathfrak{N}_k).$$

Ako je $\mathfrak{N}$ konačna struktura tada očito postoji $k \in \mathbb{N}$ tako da vrijedi $\mathfrak{N}_{k+1} = \mathfrak{N}_k$. Ako je $\mathfrak{N}$ beskonačna struktura tada općenito ne mora postojati ordinalni broj α tako da vrijedi $\mathfrak{N}_\alpha = \mathfrak{N}_{\alpha+1}$. Ako za operator $\mathcal{O}_{\varphi,P}$ postoji fiksna točka tada je označavamo sa $\mathfrak{N}_f$.

Inflatorna logika fiksne točke – IFP
Logika IFP je proširenje FO. U definiciji pojma IFP–formule dodaje se i sljedeće pravilo:

> za svaki k–mjesni relacijski simbol, takav da $P \notin \sigma$, za svaku $\sigma \cup \{P\}$–formulu $\varphi(x_1, \ldots, x_k)$, te za sve σ–terme $t_1, \ldots, t_k$ riječ $[\mathrm{ifp}_{\varphi,P}](t_1, \ldots, t_k)$ je σ–formula logike IFP.

Za svaku σ–strukturu $\mathfrak{N}$ i valuaciju v definiramo:

$$\mathfrak{N} \models_v [\mathrm{ifp}_{\varphi,P}](t_1, \ldots, t_k) \qquad \text{ako i samo ako}$$

$$\text{postoji fiksna točka } \mathfrak{N}_f \text{ operatora } \mathcal{O}_{\varphi,P},$$

$$\text{te } (v(t_1), \ldots, v(t_k)) \in \{\vec{a} \in |\mathfrak{N}_f|,\ \mathfrak{N}_f \models P(x_1, \ldots, x_k)[\vec{a}]\}.$$

(Ova logika se naziva inflatorna jer vrijedi $P^{\mathfrak{N}_0} \subseteq P^{\mathfrak{N}_1} \subseteq P^{\mathfrak{N}_2} \subseteq \ldots$).

Parcijalna logika fiksne točke – PFP
Logiku PFP definiramo na sasvim analogni način kao IFP, pri čemu umjesto operatora $\mathcal{O}_{\varphi,P}$ promatramo operator $\mathcal{O}^{\mathrm{pfp}}_{\varphi,P}$, gdje samo naglašavamo:

$$P^{\mathcal{O}^{\mathrm{pfp}}_{\varphi,P}(\mathfrak{M})} := \{\vec{a} \in |\mathfrak{M}|^k :\ \mathfrak{M} \models \varphi[\vec{a}]\}.$$

Logika najmanje fiksne točke – LFP
Logiku LFP definiramo na sasvim analogni način kao IFP, pri čemu umjesto operatora $\mathcal{O}_{\varphi,P}$ promatramo operator $\mathcal{O}^{\mathrm{lfp}}_{\varphi,P}$, gdje je φ pozitivna formula, tj. φ je ekvivalentna nekoj formuli u konjunktivnoj normalnoj formi, pri čemu niti u jednoj elementarnoj disjunkciji ne nastupa literal oblika $\neg P(\vec{t}$. Lako je provjeriti da za pozitivne fomule φ i svaku σ–strukturu $\mathfrak{N}$ vrijedi: $P^{\mathfrak{N}_k} \subseteq P^{\mathfrak{N}_{k+1}}$, te za svaku σ–strukturu $\mathfrak{N}$ postoji (najmanja i najveća) fiksna točka operatora $\mathcal{O}^{\mathrm{lfp}}_{\varphi,P}$.

Sada ističemo dva važna teorema o logikama fiksne točke. Dokaze tih teorema možete pronaći u [14].

Teorem 6.31. (N. Immerman; M. Y. Vardi, 1982.)
Logika najmanje fiksne točke LFP *hvata klasu* `P` *na klasi svih uređenih konačnih struktura.*

Teorem 6.32. (S. Abiteboul, V. Vianu, 1991.)
Logika parcijalne fiksne točke PFP *hvata klasu* `PSPACE` *na klasi svih uređenih konačnih struktura.*

Na samom kraju navodimo jedan otvoreni problem deskriptivne teorije složenosti.

Otvoreni problem: Postoji li logika koja hvata klasu `P` na klasi svih konačnih struktura?

6.4 Još zadataka

Konačni automati, regularni jezici i gramatike

1. Dokažite da je klasa svih regularnih jezika zatvorena za uniju, konkatenaciju i operaciju $\star$. (Ako su A i B jezici tada sa $A \circ B$ označavamo njihovu konkatenaciju. Ako je A neki jezik tada je $A^\star = \{x_1 x_2 \dots x_k : k \geq 0 \text{ i svaki } x_i \in A\}$.)

2. Rekurzivno simultano definiramo pojam regularnog izraza R i pripadnog jezik $L(R)$. Svaki element α zadanog alfabeta je regularni izraz. U tom slučaju definiramo $L(\alpha) = \{\alpha\}$. Prazan skup je regularni izraz, te definiramo $L(\emptyset) = \emptyset$. Prazna riječ $\sqcup$ je regularni izraz, te definiramo $L(\sqcup) = \{\sqcup\}$. Ako su R_1 i R_2 regularni izrazi, tada su i riječi $(R_1 \cup R_2)$, $(R_1 \circ R_2)$ i $(R^\star)$ regularni izrazi. Zatim, redom definiramo: $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$, $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$, te $L(R^\star) = L(R)^\star$. Dokažite da je neki jezik L regularan ako i samo postoji regularan izraz R tako da vrijedi $L = L(R)$.

3. Dokažite lemu o pumpanju za regularne jezike.

4. Dokažite da jezik $\{w$: riječ w sadrži jednak broj simbola 0 i 1 $\}$ nije regularan.

5. Dokažite da jezik $\{1^{n^2} : n \geq 0\}$ nije regularan.

6. Dokažite da ne vrijedi obrat leme o pumpanju, tj. dokažite da postoji jezik koji nije regularan a ipak se može "pumpati".

7. Konteksno neovisna gramatika je uređena četvorka (V, Σ, P, S), gdje je redom: V konačan skup čije elemente nazivamo varijable, Σ je konačan skup disjunktan sa skupom V čije elemente nazivamo završni znakovi, P je konačan skup čiji elementi su produkcije (produkcija je riječ oblika $A \to w_1 w_2 \dots w_k$, gdje je $A \in V$ i $w_i \in V \cup \Sigma$), S je element skupa V koji nazivamo početna varijabla.

Jezik generiran s nekom konteksno neovisnom gramatikom nazivamo konteksno neovisan jezik. Kažemo da je neka konteksno neovisna gramatika u Chomskyjevoj normalnoj formi ako je svaka njena produkcija jednog od oblika: $A \to BC$, $A \to a$ i $S \to \sqcup$. Dokažite da se svaki konteksno neovisan jezik može generirati pomoću gramatike koja je u Chomskyjevoj normalnoj formi.

8. Neka su A i B konteksno neovisni jezici. Dokažite da su tada i jezici $A \cup B$, $A \circ B$ i $A^\star$ konteksno neovisni.

9. Neka je R neka gramatika u Chomskyjevoj normalnoj formi. Dokažite da je tada svaku riječ $w \in L(R)$ moguće generirati iz početne varijable primjenom $2|w| - 1$ produkcija.

10. Definirajte pojmove determinističkog i nedeterminističkog potisnog automata. Dokažite da je neki jezik konteksno neovisan ako i samo postoji nedeterministički potisni automat koji prepoznaje jezik.

11. Dokažite da je svaki regularan jezik konteksno neovisan. Navedite primjer konteksno neovisnog jezika koji nije regularan.
Uputa. Znamo da jezik $\{0^n 1^n : n \in \mathbb{N}\}$ nije regularan. Definirajte potisni automat koji ga prepoznaje.

12. Dokažite lemu o pumpanju za konteksno neovisne jezike, tj. dokažite da za svaki konteksno neovisan jezik L postoji $p \in \mathbb{N}$ tako da za sve riječi $\alpha \in L$ čija je duljina barem p postoje riječi u, v, x, y, z tako da vrijedi redom: $\alpha = uvxyz$, $|vy| > 0$, $|vxy| \leq p$, te za svaki $k \in \mathbb{N}$ vrijedi $uv^k xy^k z \in L$. Primjenom leme dokažite da jezik $\{0^n 1^n 2^n : n \geq 0\}$ nije konteksno neovisan.

13. Dokažite da presjek konteksno neovisnih jezika nije nužno konteksno neovisni jezik.
Uputa. Dokažite da su jezici $\{0^i 1^i 2^j : i, j \in \mathbb{N}\}$ i $\{0^i 1^j 2^j : i, j \in \mathbb{N}\}$ konteksno neovisni. Njihov presjek je $\{0^n 1^n 2^n : n \geq 0\}$, a po prethodnom zadatku taj jezik nije konteksno neovisan.

14. Dokažite da komplement konteksno neovisnog jezika nije nužno konteksno neovisan.
Uputa. Pretpostavimo suprotno. Iz prethodnog zadatka slijedi da postoje konteksno neovisni jezici A i B takvi da jezik $A \cap B$ nije konteksno neovisan. Neka je $S = A^c \cup B^c$. Budući da su po pretpostavci A^c i B^c konteksno neovisni, tada iz zadatka 8 slijedi da je i jezik S konteksno neovisan. Tada iz pretpostavke slijedi da je i $S^c = A \cap B$ konteksno neovisan, čime je dobivena kontradikcija.

15. Neka je A konteksno neovisan jezik, a B regularan jezik. Dokažite da je $A \cap B$ također konteksno neovisan.

16. * Dokažite da se regularni jezici mogu opisati homomorfizmima na monoidima (vidi [12] i [15]).

Turing–izračunljive funkcije

1. Dokažite da je svaka inicijalna funkcija Turing–izračunljiva.
2. Dokažite da je klasa svih Turing–izračunljivih funkcija zatvorena za kompoziciju.
3. Dokažite da je klasa svih Turing–izračunljivih funkcija zatvorena za primitivnu rekurziju.
4. Dokažite da je klasa svih Turing–izračunljivih funkcija zatvorena za μ–operator.
5. Dokažite da je svaka parcijalno rekurzivna funkcija Turing–izračunljiva.
6. Dokažite da je svaka Turing–izračunljiva funkcija parcijalno rekurzivna.
   Uputa. Detaljno kodirajte rad Turingovog stroja.

(Ne)odlučivi jezici

1. Definirajte kodiranje konačnih automata. Neka je za konačni automat M sa $\langle M\rangle$ označen njegov kod. Dokažite da su sljedeći jezici odlučivi:
   a) $\{(\langle M\rangle, w) : M$ je konačni automat koji prihvaća riječ $w\}$;
   b) $\{\langle M\rangle : M$ je konačni automat koji ne prihvaća niti jednu riječ $\}$;
   c) $\{(\langle A\rangle, \langle B\rangle) : A$ i B su konačni automati tako da $L(A) = L(B)\}$;
   d) $\{\langle N, w\rangle : N$ je nedeterministički konačni automat koji prihvaća riječ $w\}$.
2. Dokažite da je svaki konteksno neovisan jezik odlučiv.
   (Definicija konteksno neovisne gramatike je dana u zadatku 7 na strani 149).
3. Označimo $L_{CFG} = \{\langle R, w\rangle : R$ je konteksno neovisna gramatika koja generira riječ $w\}$. Dokažite da je jezik L_{CFG} odlučiv.
   Uputa. Koristite zadatak 9 sa strane 149.
4. Dokažite da sljedeći jezik nije odlučiv:

   $\{\langle G, H\rangle : G$ i H su konteksno neovisne gramatike takve da $L(G) = L(H)\}$.

5. Definirajte verziju RAM–strojeva koji prepoznaju i odlučuju jezike. Dokažite da za svaki takav RAM–stroj postoji ekvivalentan Turingov stroj, i obratno.
   Uputa. Vidi [7].
6. Definirajte pojam ABAK–računala koje prepoznaje i odlučuje jezike. Dokažite da za svako takvo ABAK–računalo postoji ekvivalentan Turingov stroj, i obratno.
   Uputa. Vidi [2].

Vremenska složenost

1. Neka je L odlučiv jezik za koji postoji jednotračni Turingov stroj koji ga odlučuje, te čija je vremenska složenost strogo manja od $O(n \log_2 n)$. Dokažite da je tada jezik L regularan.

2. Neka je $T : \mathbb{N} \to \mathbb{R}^+$ funkcija za koju vrijedi $T(n) \geq n$, za svaki $n \in \mathbb{N}$. Ako je f Turing–izračunljiva na nekom Turingovom stroju vremenske složenosti $T(n)$, tada je funkcija f RAM–izračunljiva ne nekom RAM–stroju vremenske složenosti $O(T^2(n))$.
   Uputa: vidi [10].

NP–teški problemi optimizacije

Sada definiramo jedan problem iz kombinatorne optimizacije za čiju jednu verziju ćemo pokazati da je NP–teška. Važno je naglasiti da to nije problem odluke već se kao rezulat traži maksimalna vrijednost određene funkcije.

PROBLEM RUKSAKA glasi:

> *Dano je n predmeta s vrijednostima p_i i težinama w_i, gdje je $i \in \{1, \ldots, n\}$. Ruksak ima težinski kapacitet m, gdje je $m \in [0, +\infty\rangle$. Treba odrediti skup predmeta koji će stati u ruksak tako da ukupna vrijednost bude maksimalna.*

Označimo sa PROBLEM RUKSAKA* sljedeću verziju prethodnog problema:

> *Dano je n predmeta. Zadana je težina $w_i \in \mathbb{N}$ i vrijednost $p_i \in \mathbb{N}$ pojedinog predmeta. Zatim, neka je $m \in \mathbb{N}$ broj koji označava maksimalni težinski kapacitet ruksaka. Neka je $M \in \mathbb{N}$ proizvoljan. Treba odrediti je li optimalna vrijednost predmeta koji stanu u ruksak veća od M.*

Očito je PROBLEM RUKSAKA* problem odluke. Dokažimo da je PROBLEM RUKSAKA* jedan NP–težak problem. U tu svrhu dokazujemo da vrijedi PROBLEM PARTICIJA $\leq_p$ PROBLEM RUKSAKA*. PROBLEM PARTICIJA smo definirali u zadatku 2 na strani 80, te znamo iz navedenog zadatka da je problem PROBLEM PARTICIJA jedan NP–potpun problem.
Neka je $S = \{a_1, \ldots, a_n\}$ neki multiskup prirodnih brojeva. Označimo $s = a_1 + \ldots + a_n$. Definiramo jednu instancu PROBLEMA RUKSAKA* ovako:

- broj predmeta je n;
- težinski kapacitet ruksaka je $m = \lfloor s/2 \rfloor$, tj. $m = \max\{k \in \mathbb{N} : k \leq s/2\}$.
- težina i vrijednost pojedinog predmeta je zadana sa $w_i = p_i = a_i, \;\; i = 1, \ldots, n$.

- neka je $M = \lceil s/2 \rceil$ $(= \min\{k \in \mathbb{N} : s/2 \leq k\})$.

Tvrdimo da za multiskup S postoji podskup T tako da vrijedi $\sum_{x \in T} x = \sum_{x \in S \setminus T} x$ ako i samo ako je optimalna vrijednost predmeta koji stanu u ruksak veća od M.
Pretpostavimo prvo da postoji multiskup $T \subseteq S$ tako da vrijedi $\sum_{x \in T} x = \sum_{x \in S \setminus T} x$. Tada je očito broj s paran, te vrijedi $s/2 = \lfloor s/2 \rfloor = \lceil s/2 \rceil$. To znači da predmeti koji odgovaraju elementima iz skupa T imaju težinu $\lfloor s/2 \rfloor$ (dakle, stanu u ruksak), te im je ukupna vrijednost $\lceil s/2 \rceil$.
Pretpostavimo sada da postoji $R \subseteq \{1, \ldots, n\}$ tako da je:

$$\sum_{i \in R} w_i = \sum_{i \in R} a_i \leq \lfloor s/2 \rfloor$$

(to znači da predmeti stanu u ruksak), te

$$\sum_{i \in R} t_i = \sum_{i \in R} a_i \geq \lceil s/2 \rceil$$

(to znači da im je ukupna vrijednost barem $\lceil s/2 \rceil$). Lako je vidjeti da je tada nužno $\lfloor s/2 \rfloor = \lceil s/2 \rceil = s/2$. Iz toga direktno slijedi da je $\sum_{i \in R} a_i = \sum_{j \in \{1,\ldots,n\} \setminus R} a_j$.

Sada ćemo definirati još tri problema kombinatorne optimizacije. Za svaki od njih može se dokazati da je NP–težak.

PROBLEM RASPOREDA (poslova na identične strojeve) glasi:

> *Zadano je n nezavisnih poslova i m identičnih strojeva. Vrijeme izvođenja i–tog posla na bilo kojem stroju iznosi t_i. Treba odrediti raspored izvršavanja poslova na strojevima tako da vrijeme potrebno da se izvrše svi poslovi bude minimalno. Pri tome se niti jedan posao ne smije cjepkati, a jedan stroj ne može obavljati više od jednog posla u jednom trenutku.*

PROBLEM OPTIMALNO BOJENJE GRAFA glasi:

> *Zadan je neusmjereni graf. Vrhovima grafa treba pridružiti boje tako da su susjedni vrhovi uvijek u različim bojama. Pri tome ukupan broj upotrijebljenih boja mora biti minimalan.*

PROBLEM TRGOVAČKOG PUTNIKA, ili kratko PROBLEM TSP (eng. traveling salesman problem) glasi:

> *Zadan je potpuni neusmjereni graf čiji bridovi imaju zadane duljine. Treba odrediti Hamiltonov ciklus[2] minimalne duljine. Duljina ciklusa jednaka je zbroju duljina svih bridova ciklusa.*

[2]Hamiltonov ciklus u grafu je onaj ciklus koji sadrži sve čvorove grafa, ali samo jednom.

Bibliografija

[1] S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[2] G. S. Boolos, J. P. Burgess, R. C. Jeffrey, *Computability and Logic*, Fourth Edition, Cambridge University Press, 2002.

[3] L. Blum, F. Cucker, M. Shub, S. Smale, *Complexity and Real Computation*, Springer, 1998.

[4] M. Davis, *Hilbert's Tenth Problem is Unsolvable*, The American Mathematical Monthly, 80 (1973), 233–269

[5] A. Dujella, M. Maretić, *Kriptografija*, Element, Zagreb, 2007.

[6] H. D. Ebbinghaus, J. Flumm, *Finite model theory*, Springer–Verlag, 1999.

[7] P. Gács, L. Lovász, *Complexity of Algorithms*, skripta, Boston University, 2010. http://www.cs.bu.edu/faculty/gacs/courses/cs332/Home.html

[8] M. R. Garey, D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Bell Telephone Laboratories, 1979.

[9] E. Grädel et al., *Finite Model Theory and Its Applications*, Springer, 2007.

[10] J. Håstad, *Complexity Theory*, skripta, Royal Institute of Technology, Stockholm, 2009.
http://www.nada.kth.se/∼johanh/complexitylecturenotes.pdf

[11] S. Homer, A. L. Selman, *Computability and Complexity Theory*, Springer–Verlag, 2001.

[12] J. E. Hopcroft, J. D. Ulman, *Introduction to Automata Theory, Language and Computation*, Addison-Wesley Publishing, 1979.

[13] J. Hromkovič, *Algorithmics for Hard Problems*, Springer, 2004.

[14] N. Immerman, *Descriptive Complexity*, Springer, 1999.

[15] D. C. Kozen, *Automata and Computation*, Springer–Verlag, 1997.

[16] D. C. Kozen, *Theory of Computation*, Springer, 2006.

[17] S. G. Krantz, *Handbook of Logic and Proof Techniques for Computer Science*, Springer, 2002.

[18] L. Libkin, *Elements of Finite Model Theory*, Springer, 2004.

[19] V. W. Marek, *Introduction to Mathematics of Satisfiability*, Chapman&Hall/CRC, 2009.

[20] M. Mihelčić, *Davis–Putnamov algoritam*, diplomski rad, PMF–Matematički odsjek, Zagreb, 2011.
http://web.math.pmf.unizg.hr/∼vukovic/Diplomski-radovi/Mihelcic-Davis-Putnamov-algoritam.pdf

[21] M. Mihelčić, T. Lolić, *Problem ispunjivosti logičke formule (SAT)*, math.e (elektronski časopis), 21, 2012.
http://e.math.hr/math_e_article/br21/mihelcic_lolic

[22] C. H. Papadimitrou, *Computational Complexity*, Addison–Wesley, 1994.

[23] M. Sipser, *Introduction to the Theory of Computation*, Third edition, Cengage Learning, 2013.

[24] J. Talbot, D. Welsh, *Complexity and Cryptography, An Introduction*, Cambridge University Press, 2006.

[25] M. Vuković, *Matematička logika*, Element, Zagreb, 2009.

[26] M. Vuković, *Izračunljivost*, predavanja, PMF–MO, Zagreb, 2013.
https://www.math.pmf.unizg.hr/sites/default/files/pictures/izn-skripta-2009.pdf

[27] M. Vuković, *Teorija skupova*, predavanja, PMF–MO, Zagreb, 2015.
https://www.math.pmf.unizg.hr/sites/default/files/pictures/ts-skripta-2015.pdf

Indeks