Interaktivni dokazi

Žufić, Ivan

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ivan Žufić

INTERAKTIVNI DOKAZI

Diplomski rad

Voditelj rada:
doc. dr. sc. Marko Horvat

Zagreb, travanj 2022.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

> „Što intuitivno očekujemo od procedure za dokazivanje teorema? Kao prvo, da je moguće 'dokazati' istinit teorem. Kao drugo, da je nemoguće 'dokazati' neistinit teorem. Kao treće, da je komuniciranje dokaza efikasno, na sljedeći način: Nije bitno koliko vremena dokazivač provede dokazivajući, bitno je da verifikator može to brzo provjeriti."
>
> *Goldwasser, Micali, Rackoff, 1985.*

Matematički dokazi su dosta slični onome što u računarstvu označava NP. Dokazivač iznosi cijeli dokaz verifikatoru, koji nakon toga provjerava njegovu ispravnost. Nije bitno kako je dokazivač došao do dokaza i koliko je vremena pritom potrošio, bitno je da je dokaz točan i provjerljiv u razumnoj količini vremena.

Što se događa ako u zadovoljavajuće metode dokazivanja uključimo još neke tehnike? Primjerice, ima smisla dozvoliti interakciju dokazivača i verifikatora. Također, možemo verifikatoru omogućiti generiranje slučajnih vrijednosti, primjerice bacanjem novčića. Uz to, u mnogim situacijama nije nužno da smo sasvim uvjereni u dokaz, dovoljno je da smo uvjereni uz neku veoma veliku vjerojatnost.

Omogućavanjem tih metoda dolazimo do klase koju zovemo interaktivni dokazi. Ispostavlja se da je klasa problema rješivih interaktivnim dokazima ekvivalentna s PSPACE, klasi problema rješivih s Turingovim strojevima koji koriste polinomno mnogo memorije. Za klasu PSPACE smatra se da je veća od NP. Drugim riječima, omogućavanjem tih metoda zbilja dobivamo znatno veću klasu. Također, interaktivni dokazi su u praksi korišteni kao osnova za neke od kriptografskih tehnika.

Na početku ćemo navesti standardne definicije, koje načelno prate [9], [10].

U drugom poglavlju formalno ćemo definirati interaktivne dokaze. Proći ćemo kroz primjer interaktivnih dokaza neizomorfnosti grafova te razmotriti što se mijenja ako su slučajne vrijednosti koje verifikator koristi javne. Ovo poglavlje prati [2].

U trećem poglavlju dokazujemo da je klasa IP jednaka klasi PSPACE. Dokaz prati [5], [2].

U četvrtom poglavlju bavimo se nekim praktičnim primjenama interaktivnih dokaza u kriptografiji te konstruktivno dokazujemo da ako postoje jednosmjerne funkcije — funkcije koje znamo efikasno izračunati, ali ne znamo efikasno invertirati — onda je bilo koji jezik u NP također u klasi *zero-knowledge dokaza*, varijanti interaktivnih dokaza u kojima verifikator iz interakcije ne saznaje ništa novo, osim da riječ pripada jeziku.

Poglavlje 1

Osnovni pojmovi i rezultati

1.1 Turingovi strojevi

Alfabet je neki konačan neprazan skup, koji obično označavamo sa Σ. Elementi alfabeta su *znakovi*. *Riječ* definiramo kao konačan niz znakova. Skup svih riječi alfabeta Σ označavamo sa Σ^*. *Jezik* je bilo koji podskup od Σ^*.

Definicija 1.1. ***Turingov stroj*** *je uređena sedmorka* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$*, gdje redom vrijedi:*

- Q *je konačan skup čije elemente nazivamo* stanja*;*
- Σ *je konačan skup čije elemente nazivamo* ulazni znakovi*. Pretpostavljamo da* Σ *ne sadrži* prazan znak *koji označavamo sa* $\sqcup$*;*
- Γ *je konačan skup koji nazivamo* alfabet *Turingovog stroja. Pretpostavljamo da je prazan znak* $\sqcup$ *element od* Γ *te da je* $\Sigma \subseteq \Gamma$*;*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ *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*.*

Turingov stroj ćemo ponekad također zvati deterministički *Turingov stroj.*

Definicija 1.2. *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* $I_1, \ldots, I_m$ *u skupu* $\{L, D, S\}$ *tako da vrijedi:*

- $r_0 = q_0$ *i* s_0 *je prvi lijevi znak riječi* w*;*
- *za svaki* $j \in 0, \ldots, m-1$ *vrijedi* $\delta(r_j, s_j) = (r_{j+1}, s_{j+1}, I_{j+1})$ *i* $r_j \notin q_{DA}, q_{NE}$*;*
- $r_m = q_{DA}$*.*

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.*

Nadalje ćemo reći da neki Turingov stroj *stane* s danim ulazom ako postoje nizovi kao u prethodnoj definiciji, pri čemu dodatno dopuštamo $r_m = q_{NE}$. Ako Turingov stroj T prepoznaje riječ w, tj. stroj T s ulazom w stane 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 stane u konačno mnogo koraka u stanju q_{NE}, tada još kažemo da Turingov stroj *odbija* riječ w.

Definicija 1.3. ***Nedeterministički Turingov stroj*** $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$ *je uređena sedmorka gdje* $Q, \Sigma, \Gamma, q_0, q_{DA}$ *i* q_{NE} *imaju isto značenje kao kod deterministčkih Turingovih strojeva. Međutim, funkcija prijelaza je oblika* $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$*.*

Definicija 1.4. *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 znakova* $I_1, \ldots, I_m \in \{L, D, S\}$ *tako da vrijedi:*

- $r_0 = q_0$ *i* s_0 *je prvi lijevi znak riječi* w*;*
- *za svaki* $j \in \{0, \ldots, m-1\}$ *vrijedi* $(r_{j+1}, s_{j+1}, I_{j+1}) \in \delta(r_j), s_j)$ *i* $r_j \notin \{q_{DA}, q_{NE}\}$*;*
- $r_m = q_{DA}$*.*

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

Definicija 1.5. *Za neki jezik* $L \subseteq \Sigma^*$ *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}*.*

Definicija 1.6. *Neka je* T *neki Turingov stroj (deterministički ili nedeterministički), te su* q *i* q' *dva njegova stanja,* s *i* s' *znakovi 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 1.7. *Neka je T neki deterministički Turingov stroj koji stane za svaki ulazni podatak.* ***Vremenska složenost deterministočkog stroja*** *T je funkcija $time_T : \mathbb{N} \to \mathbb{N}$, gdje je $time_T(n)$ maksimalan broj koraka koje stroj T napravi za ulazne podatke 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 koje stroj N napravi za ulazne podatke duljine n.*

Definicija 1.8. *Neka su $f, g : \mathbb{N} \to \mathbb{R}^+$. Kažemo da je funkcija g* ***asimptotska gornja međa*** *za funkciju f ako postoje $c > 0$ i $n_0 \in \mathbb{N}$ takvi da za svaki $n \geq n_0$ vrijedi $f(n) \leq c \cdot g(n)$. Oznaka: $f(n) \sim O(g(n))$.*

Definicija 1.9. *Neka je T deterministički ili nedeterministički Turingov stroj. Kažemo da je stroj T* ***vremenski polinoman*** *ako postoji neki polinom f tako da vrijedi $time_T(n) = O(f(n))$.*

Definicija 1.10. *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.*

Definicija 1.11. *Klasu svih jezika koji su odlučivi na nekom determinističkom Turingovom stroju vremenske složenosti $O(n^k)$, za neki $k \in \mathbb{N}$, označavamo* ***PTIME****, ili kratko* ***P****.*

Definicija 1.12. *Sa* ***NPTIME****, ili kratko* ***NP****, označavamo klasu svih jezika koji su odlučivi na nekom nedeterminističkom Turingovom stroju vremenske složenosti $O(n^k)$, za neki $k \in \mathbb{N}$.*

Teorem 1.13 (O certifikatu)**.** *Za svaki jezik L vrijedi:*

$$L \in NP \iff (\exists R \in P)(\exists k \in \mathbb{N})(L = \{x : (\exists c)(|c| = O(|x|^k) \wedge R(x, c))\})$$

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

U ovom radu certifikate koristimo jer se njima možemo u polinomnoj složenosti uvjeriti da neka riječ pripada nekom NP jeziku.

Definicija 1.14. *Neka su Σ_1 i Σ_2 proizvoljni alfabeti. Kažemo da je neka funkcija f iz Σ_1^* u Σ_2^* vremenski polinomno izračunljiva ako postoji vremenski polinomno složen Turingov stroj koji za svaku riječ $w \in \Sigma_1^*$ kao ulazni podatak na traku ispisuje $f(w)$. Kažemo da je jezik $L_1 \subseteq \Sigma_1^*$* ***vremenski polinomno reducibilan*** *na jezik $L_2 \subseteq \Sigma_2^*$ ako postoji vremenski polinomno izračunljiva funkcija $f : \Sigma_1^* \to \Sigma_2^*$ takva da za svaki $w \in \Sigma_1^*$ vrijedi:*

$$w \in L_1 \iff f(w) \in L_2$$

Definicija 1.15. *Neka je T neki deterministički Turingov stroj koji staje za svaki ulaz.* ***Prostorna složenost*** *deterističkog Turingovog stroja T je funkcija $space_T : \mathbb{N} \to \mathbb{N}$, gdje je $space_T(n)$ maksimalan broj elemenata trake koje stroj T koristi, za svaki ulazni podatak duljine n. Kažemo da je T* ***prostorno polinoman*** *ako postoji polinom f takav da vrijedi $space_T(n) = O(f(n))$.* ***PSPACE*** *je klasa prostorno polinomnih jezika.*

Definicija 1.16. *Jezik L je* ***PSPACE-potpun*** *ako vrijedi:*

- *L pripada klasi PSPACE*
- *svaki jezik L' iz klase PSPACE je vremenski polinomno reducibilan na L.*

1.2 Matematička logika

Definicija 1.17. ***Alfabet logike sudova*** *je unija skupova A_1, A_2 i A_3, pri čemu je:*

- $A_1 = \{P_0, P_1, P_2, \dots\}$ *prebrojiv skup čije elemente nazivamo* ***propozicijske varijable****,*
- $A_2 = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ *skup* ***logičkih veznika****,*
- $A_3 = \{(,)\}$ *skup* ***pomoćnih znakova*** *(zagrade).*

Definicija 1.18. ***Atomarna formula*** *je svaka propozicijska varijabla. Pojam* ***formule logike sudova*** *definiramo rekurzivno:*

1. *svaka atomarna formula je formula;*
2. *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 (ispuštamo zagrade kad nema mogućnosti zabune);*
3. *riječ alfabeta logike sudova je formula ako je nastala primjenom konačno mnogo koraka 1 i 2.*

Definicija 1.19. *Svako preslikavanje sa skupa svih propozicijskih varijabli u skup $\{0, 1\}$, tj. $I : \{P_0, P_1, \dots\} \to \{0, 1\}$, nazivamo* ***(totalna) interpretacija****. Ako je preslikavanje definirano na podskupu skupa propozicijskih varijabli tada kažemo da je to* ***parcijalna interpretacija****.*

Definicija 1.20. *Neka je I interpretacija (totalna ili parcijalna). Tada* ***vrijednost interpretacije*** *I na proizvoljnoj formuli definiramo rekurzivno ovako:*

- $I(\neg A) = 1$ *ako i samo ako* $I(A) = 0$*;*
- $I(A \wedge B) = 1$ *ako i samo ako* $I(A) = 1$ *i* $I(B) = 1$*;*

- $I(A \vee B) = 1$ *ako i samo ako* $I(A) = 1$ *ili* $I(B) = 1$*;*
- $I(A \to B) = 1$ *ako i samo ako* $I(A) = 0$ *ili* $I(B) = 1$*;*
- $I(A \leftrightarrow B) = 1$ *ako i samo ako* $I(A) = I(B)$*.*

Definicija 1.21. *Za formulu F logike sudova kažemo da je **ispunjiva** ako postoji interpretacija I takva da vrijedi $I(F) = 1$.*

Definicija 1.22. *Atomarnu formulu i njezinu negaciju nazivamo **literal**. Formulu oblika $A_1 \wedge A_2 \wedge \cdots \wedge A_n$ nazivamo **konjunkcija**, a formulu oblika $A_1 \vee A_2 \vee \cdots \vee A_n$ 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.*

Definicija 1.23. *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:*

$$k\text{–}SAT = \{F : F \text{ je ispunjiva } k\text{–}KNF\}$$

Definicija 1.24. ***#3SAT*** $= \{(F, K) : F$ *je 3-KNF formula koja ima točno K interpretacija I za koje vrijedi $I(F) = 1$}.*

Zanima nas baš 3-SAT jer je 2-SAT u P, 3-SAT u NP te se k-SAT za $k > 3$ može jednostavno reducirati na 3-SAT.

1.3 Teorija grafova

Definicija 1.25. ***Graf*** *G je uređeni par (V, E), gdje je V neprazan konačan skup čije elemente zovemo **vrhovi**, a skup $E \subseteq V \times V$ skup čije elemente zovemo **bridovi**. Ako je $(a, b) \in E$, kažemo da su vrhovi a i b **povezani** te da je brid (a, b) **incidentan** s vrhovima a i b. Ponekad pišemo $(a, b) \in G$ umjesto $(a, b) \in E$.*

Definicija 1.26. *Neka su G_1 i G_2 grafovi. Ako G_1 i G_2 imaju n vrhova te postoji bijekcija $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ takva da za svaki brid (a, b) vrijedi $(a, b) \in G_1 \iff (\pi(a), \pi(b)) \in G_2$, kažemo da su G_1 i G_2 **izomorfni** i da je π **izomorfizam** s G_1 na G_2.*

Definicija 1.27. *Neka je G graf. Ako je π izomorfizam sa G na G, onda kažemo i da je π **automorfizam** sa G na G. Skup svih automorfizama grafa G označavamo s $aut(G)$.*

1.4 Vjerojatnost i statistika

Neka $\mathbb{P}[E]$ označava vjerojatnost događaja E.

Teorem 1.28 (Bonferronijeva nejednakost)**.** *Za sve događaje $A_1, \ldots, A_n$ vrijedi:*

$$\mathbb{P}\left[\bigcup_{i=1}^{n} A_i\right] \geq \sum_{i=1}^{n} \mathbb{P}[A_i] - \sum_{1 \leq i < j \leq n} \mathbb{P}[A_i \cap A_j].$$

Dokaz nalazimo u [2].

Ako imamo pošteni novčić i bacimo ga 100 puta, očekujemo da će pismo pasti 50 puta. Međutim, hoće li nas jako iznenaditi ako pismo padne 60 puta? Općenito, ako imamo n nezavisnih (ne nužno jednako distribuiranih) varijabli nad $\{0, 1\}$, kolika je vjerojatnost da je njihova suma barem k? O toj vjerojatnosti govori Chernoffova ograda.

Teorem 1.29 (Chernoffova ograda)**.** *Neka su $X_1, \ldots, X_n$ nezavisne slučajne varijable nad $\{0, 1\}$ i neka je $\mu = \sum_{i=1}^{n} E[X_i]$. Tada za svaki $\delta > 0$ vrijedi:*

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \geq (1+\delta) \cdot \mu\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \leq (1-\delta) \cdot \mu\right] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}$$

Često se koristi i sljedeći korolar:

Korolar 1.30. *Uz oznake i pretpostavke za Chernoffovu ogradu, za svaki $c > 0$ vrijedi:*

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i - \mu\right| \geq c \cdot \mu\right] \leq 2 \cdot e^{-min\{c^2/4, c/2\}\mu}.$$

Posebno, ova vjerojatnost je ograničena s $2^{-O(n)}$, gdje konstanta za O ovisi o c.

Dokaze nalazimo u [2].

Poglavlje 2

Sustav interaktivnih dokaza

Sustav interaktivnih dokaza (engl. *Interactive proof system*, nadalje *IPS*) je apstraktni stroj koji dokazivanje prikazuje kao interakciju između dva sudionika, jednoga koji izlaže dokaz i drugoga koji provjerava dokaz. Te sudionike ćemo redom zvati *dokazivač* i *verifikator*. Za dokazivača pretpostavljamo da ima neograničene resurse za računanje, ali i da je možda zlonamjeran te da mu se ne može vjerovati. Verifikator je pouzdan, ali ima ograničene resurse, tj. mora biti polinomne vremenske složenosti. Od svakog IPS zahtijevamo:

- *Potpunost*: ako je izjava istinita, dokazivač može uvjeriti verifikatora u njenu istinitost uz vjerojatnost barem $\frac{2}{3}$.
- *Ispravnost*: ako je izjava neistinita, dokazivač ne može uvjeriti verifikatora da je izjava istinita, osim uz vjerojatnost od najviše $\frac{1}{3}$.

Vrijednosti $\frac{2}{3}$ i $\frac{1}{3}$ u gornjoj definiciji nisu presudne, pokazat ćemo da se može postići proizvoljno dobru vjerojatnost, odnosno da ih smijemo zamijeniti redom s $1 - \epsilon$ te ϵ, za bilo koji $0 < \epsilon < \frac{1}{2}$.

2.1 Vjerojatnosno provjeravanje

Vjerojatnosti u definicijama potpunosti i ispravnosti bi za slučaj determinističkog verifikatora mogle biti 0 i 1, ali bismo htjeli verifikatoru omogućiti generiranje slučajnih bitova i donošenje odluka na temelju tih slučajnih bitova. Time može doći do eventualne pogreške u odlukama. Motivacija za omogućavanje verifikatorovih grešaka je to da u praksi neke probleme (npr. u kriptografiji; o tome će biti više riječi u posljednjem poglavlju) ne možemo deterministički riješiti, ali nam je rješenje koje funkcionira uz veoma veliku vjerojatnost zadovoljavajuće. Slučajne bitove možemo promatrati kao bacanje poštenog novčića. Novčić može pasti na bilo koju stranu te mi možemo modelirati svoje odluke na

temelju strane koja padne. Također, možemo baciti novčić i javno obznaniti rezultat bacanja, a možemo i rezultat zadržati za sebe. To može utjecati na odluke drugih, ali ispostavit će se da zapravo ne radi veliku razliku. Na kraju ovog poglavlja pokazat ćemo da je klasa IPS koji podržava privatno bacanje novčića jednaka klasi koja podržava javno bacanje i koristi dvije dodatne iteracije interakcije dokazivača i verifikatora.

Prije nego što definiramo konkretne IPS koji nas zanimaju, potrebne su nam još neke definicije.

Definicija 2.1. ***Vjerojatnosni Turingov stroj*** *je Turingov stroj koji ima dvije funkcije prijelaza, δ_1 i δ_2 te neovisno, u svakom koraku uz vjerojatnost $\frac{1}{2}$, primijenjuje funkciju prijelaza δ_1, a inače primjenjuje δ_2.*

Ostale definicije za vjerojatnosni Turingov stroj su analogne onima za nedeterministički Turingov stroj.

Premda na prvi pogled vjerojatnosni i nedeterministički stroj izgledaju veoma slično, u svojoj osnovi su veoma različiti. Glavna razlika između njih je kako promatramo stablo svih izračunavanja. Nedeterministički strojevi prihvaćaju ulaznu riječ ako postoji grana koja je prihvaća, dok vjerojatnosni promatraju vjerojatnost prihvaćanja — vjerojatnost prihvaćanja prati broj grana koje prihvaćaju. Također, vjerojatnosni strojevi se, poput deterministčkih, baziraju na efikasnim i time praktičnim idejama, dok nedeterministički zbilja prolaze kroz cijelo stablo stanja, koje je eksponencijalno veliko.

Nadalje, želimo definirati što točno znači da dokazivač i verifikator komuniciraju. Odmah ćemo definirati vjerojatnosnu verziju interakcije, jer nas ona zanima u ovom radu. Broj koraka interakcije označavat ćemo s k, a broj slučajno bačenih novčića s m. r_i su rezultati bacanja, a a_i poruke kojima strojevi komuniciraju. S $r_{a\ldots b}$ označavamo podriječ od r počevši od a-tog do b-tog elementa, uključivo.

Definicija 2.2. *Neka su $f, g : \{0,1\}^* \to \{0,1\}^*$, te k i m prirodni brojevi (koji mogu ovisiti o dužini inputa). Neka je $r \in \{0,1\}^m$ uniformno slučajno odabran niz iz $\{0,1\}^m$.* ***Interakcija u k koraka s privatnim novčićima*** *funkcija f i g na ulazu $x \in \{0,1\}^*$, koju označavamo $\langle f, g\rangle$ je niz riječi $a_1, \ldots, a_k \in \{0,1\}^*$ za koje postoji monotono rastući niz $b_1, \ldots, b_k$ prirodnih brojeva ne većih od m za koji vrijedi:*

- $a_1 = f(x, r_1)$
- $a_2 = g(x, a_1)$
- $\ldots$
- $a_{2i+1} = f(x, r_{1\ldots b_{2i}}, a_1, \ldots, a_{2i})$ *za* $2i < k$
- $a_{2i+2} = g(x, a_1, \ldots, a_{2i+1})$ *za* $2i + 1 < k$

Izlaz *funkcije f na kraju interakcije definiramo kao $f(x, r_{1\ldots m}, a_1, \ldots, a_k)$ i označavamo s $\texttt{out}_f\langle f, g\rangle(x)$. Pretpostavljamo da je ta vrijednost u $\{0, 1\}$.*

Funkcija f ovisi o r, pa je izlaz zapravo slučajna varijabla.

Premda funkcija f ima r kao parametar, funkcija g ga nema. Varijanta u kojoj i funkcija g u $(2i)$-tom koraku interakcije ima $r_{1\ldots b_{2i-1}}$ kao parametar zove se **interakcija u k koraka s javnim novčićima**. Napomenimo da funkcija g ne „vidi budućnost", tj. nema za parametre bitove od r koje funkcija f nije imala u prethodnoj interaciji interakcije. Riječ r ćemo ponekad zvati **privatni bitovi** ako g nema pristup njima, odnosno **javni bitovi** ako ih g dobiva kao parametar.

2.2 Klasa IP

Definicija 2.3. *Neka je $f : \mathbb{N} \to \mathbb{N}$ neka polinomno izračunljiva funkcija.* ***Jezik L je u IP[f]*** *ako postoji polinomni vjerojatnosni Turingov stroj V koji može imati interakciju u k koraka s javnim novčićima s funkcijom $P : \{0, 1\}^* \to \{0, 1\}^*$ takvu da vrijedi:*

- Potpunost*: $x \in L \to \exists P(\mathbb{P}(\texttt{out}_V\langle P, V\rangle(x) = 1) \geq \frac{2}{3})$.*
- Ispravnost*: $x \notin L \to \forall P(\mathbb{P}(\texttt{out}_V\langle P, V\rangle(x) = 1) \leq \frac{1}{3})$.*

gdje sve vjerojatnosti proizlaze iz odabira r. Sad definiramo $\boldsymbol{IP} = \bigcup_{c\geq 1} \boldsymbol{IP}[n^c]$.

Ranije smo spomenuli da vrijednost $\frac{1}{3}$ nema nikakav velik značaj za mogućnost pogreške; možemo uzeti bilo koju pozitivnu vrijednost manju od $\frac{1}{2}$, o čemu govori sljedeći teorem:

Teorem 2.4. *Klasa IP se ne mijenja ako u definiciji potpunosti i ispravnosti zamijenimo vrijednosti $\frac{2}{3}$ i $\frac{1}{3}$ s vrijednostima $1 - \epsilon$ i ϵ, redom, za bilo koji $0 < \epsilon < \frac{1}{2}$.*

Dokaz. Verifikator može ponoviti cijeli proces interakcije proizvoljan broj puta, neka je to n iteracija. Nakon toga, prihvatit će riječ ako i samo ako je prihvatio riječ u preko pola iteracija. Ako je $x \in L$, onda ga dokazivač koji ga može u svakoj iteraciji uvjeriti s vjerojatnošću barem $\frac{2}{3}$ u ovoj varijanti može uvjeriti s vjerojatnošću barem $1 - 2^{-O(n)}$, zbog Chernoffove ograde. Ako $x \notin L$, onda je nakon n iteracija vjerojatnost za pogrešku najviše $2^{-O(n)}$, zbog Chernoffove ograde. Valja napomenuti da ograda na vjerojatnost od $\frac{1}{3}$ da dokazivač uvjeri verifikatora vrijedi neovisno o broju iteracija obavljenih u prošlosti, jer ne postoji dokazivač koji može uvjeriti verifikatora s vjerojatnošću većom od $\frac{1}{3}$, pa posebno to vrijedi i za dokazivače koji znaju iteracije obavljene u prošlosti. □

2.3 GNI ∈ IP

Definicija 2.5. ***Problem neizomorfnosti grafova*** *(engl.* Graph Non-Isomorphism*, nadalje* GNI*) je problem odlučivanja vrijedi li za dva grafa da* nisu *izomorfni.*

Problem neizomorfnosti grafova nam je posebno zanimljiv zato što trenutno nije poznato pripada li klasi NP. Međutim, taj problem pripada klasi IP, što ćemo sad i pokazati. Na prvi pogled može izgledati prirodnije promatrati izomorfnost grafova. Međutim, izomorfnost grafova je očito u NP — jer izomorfni grafovi imaju certifikat za to (dovoljno je da certifikat sadrži permutaciju preko koje su izomorfni) — a želimo pokazati da je NP prava potklasa od IP, uz pretpostavku da je NP prava potklasa od PSPACE (trenutno se smatra da to vrijedi).

Teorem 2.6. *GNI ∈ IP.*

Dokaz. Neka su dani grafovi G_1 i G_2 za koje želimo odrediti jesu li neizomorfni. Ponovimo sljedeću proceduru 2 puta:

1. Verifikator uniformno slučajno odabire indeks i iz $\{1, 2\}$.
2. Neka je H graf nastao slučajnim permutiranjem indeksa čvorova od G_i.
3. Verifikator šalje dokazivaču grafove G_1, G_2 i H i pita ga koji je od grafova, G_1 ili G_2 iskorišten za generiranje H. Drugim riječima, verifikator pita dokazivača koja je vrijednost indeksa i.
4. Dokazivač šalje verifikatoru indeks j pripadnog grafa.

Nakon svih koraka procedure, verifikator prihvaća dokaz ako i samo ako je u svakom koraku vrijedilo $i = j$.

Provjerimo *potpunost* i *ispravnost* ovog protokola:

- *Potpunost*: Ako su G_1 i G_2 neizomorfni, onda postoji dokazivač koji će u svakom koraku procedure ispitati je li H izomorfan s G_1, je li H izomorfan s G_2, i kad se točno jedna od tih provjera ispostavi istinitom, vratiti verifikatoru indeks te provjere. Taj dokazivač će u svakom koraku vratiti ispravan odgovor i time uvjeriti verifikatora.
- *Ispravnost*: Ako G_1 i G_2 nisu neizomorfni, onda ne postoji dokazivač koji može odrediti koji je od grafova korišten za generiranje H, pa je u svakom koraku vjerojatnost vraćanja ispravnog odgovora $\frac{1}{2}$. Nakon 2 koraka, vjerojatnost ispravnog odgovora u svim koracima, i time uvjeravanja verifikatora u neispravan dokaz, je $\frac{1}{4}$, što je zbilja manje od potrebnih $\frac{1}{3}$. Štoviše, jednostavno možemo dobiti proizvoljno malu vjerojatnost povećavajući broj koraka procedure.

□

2.4 Klasa AM

U klasi IP, dokazivač nema pristup verifikatorovim slučajnim bitovima. Drugim riječima, verifikatorovi bitovi su tajni. Nameće se pitanje što se događa ako su ti bitovi javni. Time dolazimo do *Arthur-Merlin protokola*, odnosno pripadne klase AM:

Definicija 2.7. *Klasu AM[k] definiramo slično kao IP[k], uz razliku da ne koristimo interakciju s privatnim nego s javnim novčićima. Također definiramo AM = AM[2].*

Napomenimo da, iako su sad bitovi javni, dokazivač nema pristup bitovima kojima verifikator nije imao pristup u prethodnoj iteraciji interakcije. Drugim riječima, dokazivač „ne vidi budućnost".

Možda je iznenađujuće, ali ispostavlja se da javnost bitova zapravo ne igra preveliku ulogu. Naime, može se dokazati da je $IP[k] = AM[k+2]$, odnosno da se korištenjem javnih bitova i dva dodatna koraka interaktivne komunikacije može postići isto što i korištenjem privatnih bitova.

U nastavku ćemo prvo navesti dokaz za $GNI \in AM$. Iz tog dokaza nazire se ideja dokaza za $IP[k] = AM[k+2]$, čiju ćemo skicu navesti nakon toga. Potpuni dokaz za $IP[k] = AM[k+2]$ može se naći u [5]

GNI ∈ AM

U ovoj točki pokazat ćemo da je GNI ∈ AM. Pristupit ćemo tom problemu s kvantitativnog aspekta. Intuitivno (sve ovo ćemo kasnije definirati precizno), želimo procijeniti koliko postoji grafova koji su izomorfni s bar jednim od dva grafa za koje ispitujemo jesu li neizomorfni; ako jesu izomorfni, taj broj će biti znatno manji nego ako nisu. Za procjenu te vrijednosti, koristit ćemo protokol za procjenu veličine skupova. Taj protokol koristi hash funkcije sa skupa svih grafova, te će verifikator pitati dokazivača imaju li neke slučajne vrijednosti praslike. Na temelju toga verifikator će procijeniti veličinu skupa.

Neka $x \in_R S$ označava da je x uniformno slučajno odabran element iz S.

Definicija 2.8. *Neka je $\mathcal{H}_{n,k}$ klasa funkcija sa $\{0,1\}^n$ u $\{0,1\}^k$. Kažemo da je $\mathcal{H}_{n,k}$* ***klasa u parovima nezavisnih hash funkcija*** *ako za sve $x, x' \in \{0,1\}^n, y, y' \in \{0,1\}^k$ gdje je $x \neq x'$ vrijedi $\mathbb{P}[h(x) = y \wedge h(x') = y'] = 2^{-2k}$, gdje je $h \in_R \mathcal{H}_{n,k}$.*

Drugim riječima, ako su x i x' neke dvije riječi iz $\{0,1\}^n$ i h iz $\mathcal{H}_{n,k}$ slučajno odabran, onda je $(h(x), h(x'))$ uniformna slučajna varijabla na $\{0,1\}^k \times \{0,1\}^k$.

S $GF(2^n)$ označavamo konačno polje s 2^n elemenata. Neka je $f(x)$ neki ireducibilan polinom stupnja n nad Z_2. Elementi polja $GF(2^n)$ su polinomi stupnja manjeg od n nad Z_2. Zbrajanje i množenje definiramo kao standardno zbrajanje i množenje polinoma u $Z_2[x]$, i promatramo ih modulo $f(x)$.

Skup $\{0,1\}^n$ možemo poistovjetiti sa konačnim poljem $GF(2^n)$ te odabrati $\mathcal{H}_{n,n}$:

Teorem 2.9. *Za svaki n, definiramo klasu $\mathcal{H}_{n,n}$ da bude $\{h_{a,b} : a, b \in GF(2^n)\}$, te da za svaki a, b vrijedi $h_{a,b}(x) = ax + b$. Tada je $\mathcal{H}_{n,n}$* klasa po parovima neovisnih hash funkcija.

Dokaz. Neka su $x \neq x' \in GF(2^n)$ te neka su $y, y' \in GF(2^n)$. Tada je $h_{a,b}(x) = y$ i $h_{a,b}(x') = y'$ ako i samo ako je $ax+b = y$ i $ax'+b = y'$. To vrijedi ako i samo ako je $a = (y-y')(x-x')^{-1}$, što je jedinstveno definirano ($x-x' \neq 0$, pa ima multiplikativni inverz). Sada je i $b = y-ax$. Dakle, par (a, b) je jedinstveno određen za fiksne x, x', y, y', pa je vjerojatnost odabira točno tih a, b zbilja $\frac{1}{2^{2n}}$. □

Slijedi *Goldwasser-Sipserov protokol za donju ogradu veličine skupa* [5]: Neka je $S \subseteq \{0,1\}^m$ skup za koji možemo provjeriti članstvo nekog elementa. Neka je K neki broj koji znaju i verifikator i dokazivač. Cilj dokazivača je uvjeriti verifikatora da vrijedi $|S| \geq K$, te verifikator treba uz veliku vjerojatnost odbaciti tvrdnju ako je $|S| \leq \frac{K}{2}$. Neka je k prirodni broj takav da je $2^{k-2} < K \leq 2^{k-1}$.

- Verifikator slučajno odabire funkciju $h : \{0,1\}^m \to \{0,1\}^k$ iz $\mathcal{H}_{m,k}$, slučajno odabire $y \in \{0,1\}^k$, te šalje h i y dokazivaču.
- Nakon toga dokazivač pokušava pronaći $x \in S$ takav da je $h(x) = y$. Dokazivač šalje verifikatoru taj x i certifikat da je $x \in S$.
- Na kraju verifikator koriseći cerfitikat provjerava da je $x \in S$. Ako jest, prihvaća dokaz, a ako nije, odbija ga.

Dokazivač može uvjeriti verifikatora ako i samo ako postoji $x \in S$ takav da je $h(x) = y$. Pokažimo da postoji velika razlika u vjerojatnostima prihvaćanja za događaje koji nas zanimaju, tj. za $|S| \geq K$ i $|S| < \frac{K}{2}$. Intuitivno, ako je $|S| \geq K$, onda će slučajno odabrana vrijednost y imati znatno veću (barem p nasuprot $\frac{3p}{4}$) vjerojatnost da ima prasliku u S nego ako je $|S| \leq \frac{K}{2}$. Ponavljanjem postupka mnogo puta, verifikator može postati proizvoljno dobro siguran da zna razabrati vrijedi li $|S| \geq K$ ili $|S| < \frac{K}{2}$.

Lema 2.10. *Neka je $S \subseteq \{0,1\}^m$ takav da je $|S| \leq \frac{2^k}{2}$. Neka su $p = \frac{|S|}{2^k}$, $h \in_R \mathcal{H}_{m,k}$ te $y \in_R \{0,1\}^k$. Tada vrijedi*

$$p \geq \mathbb{P}[\exists_{x \in S} : h(x) = y] \geq \frac{3p}{4}.$$

Dokaz. Gornja ograda očito vrijedi, jer najviše $|S|$ (od ukupno 2^k) vrijednosti koje y može poprimiti imaju prasliku u S.

Za dokaz donje ograde, neka je $y \in \{0,1\}^k$ proizvoljan, te neka je E_x događaj u kojem je $h(x) = y$. Slijedi da je $\mathbb{P}[\exists x \in S : h(x) = y] = \mathbb{P}[\bigcup_{x \in S} E_x]$ pa po Bonferronijevoj nejednakosti dobivamo da ta vjerojatnost iznosi barem

$$\sum_{x \in S} \mathbb{P}[E_x] - \frac{1}{2} \sum_{x \neq x' \in S} \mathbb{P}[E_x \cap E_{x'}].$$

$\mathcal{H}$ je klasa po parovima neovisnih hash funkcija, pa za $x \neq x'$ vrijedi $\mathbb{P}[E_x] = 2^{-k}$ i $\mathbb{P}[E_x \cap E_{x'}] = 2^{-2k}$, pa je tražena vjerojatnost barem

$$\frac{|S|}{2^k} - \frac{1}{2} \cdot \frac{|S|^2}{2^{2k}} = \frac{|S|}{2^k}\left(1 - \frac{|S|}{2^{k+1}}\right) \geq \frac{3p}{4}.$$

□

Teorem 2.11. *GNI ∈ AM.*

Dokaz. Neka su G_1 i G_2 grafovi koji sadrže n čvorova svaki (za grafove koji sadrže različit broj čvorova, rješenje je trivijalno). Svaki od njih ima najviše $n!$ različitih izomorfnih grafova. Taj broj je strogo manji od $n!$ ako graf ima neki netrivijalan automorfizama a nama je jednostavnije dokazivanje ako ih je točno $n!$, pa uzmimo sljedeću definiciju skupa S:

$$S = \{(H, \Pi) : H \text{ je izomorfan barem jednom od grafova } G_1, G_2, \text{ te je } \Pi \in aut(H)\}$$

Ako su G_1 i G_2 izomorfni, onda je $|S| = n!$, a ako nisu izomorfni, onda je $|S| = 2n!$.

Sada možemo iskoristiti Goldwasser-Sipserov protokol za donju granicu veličine skupa, to jest verifikator može provesti nekoliko iteracija tog protokola, i prihvatiti dokaz ako je udio prihvaćenih dokaza barem $\frac{5}{8} \cdot \frac{K}{2^k}$. Koristeći Chernoffovu ogradu, zaključujemo da je konstantan broj iteracija dovoljan za ispunjavanje uvjeta potpunosti i ispravnosti.

Broj potrebnih rundi interakcije između dokazivača i verifikatora ostaje 2, jer verifikator može sve svoje upite, parove (h, y), poslati odjednom. □

Odnos IP i AM

Promotrimo kako su povezani protokoli koje smo koristili za GNI ∈ IP i GNI ∈ AM. Zanima nas distribucija poruka koje verifikator šalje dokazivaču u ovisnosti o slučajnom bitu koji je iskoristio. Ako su grafovi izomorfni, slučajni bit (za odabir iz kojeg grafa će nastati graf H) nije otkriven slanjem bilo koje poruke, a ako grafovi nisu izomorfni, onda poruka otkriva skriveni bit. Skup vrijednosti koje poslana poruka može imati je dvostruko veći ako su grafovi izomorfni.

O interakciji s javnim novčićima zapravo možemo razmišljati kao o interakciji u kojoj dokazivač pokušava uvjeriti verifikatora da bi u interakciji s privatnim novčićima dokazivač uspio uvjeriti verifikatora u ispravnost dokaza s velikom vjerojatnošću.

Precizno o odnosu IP i AM govori Goldwasser-Sipserov teorem.

Teorem 2.12 (Goldwasser-Sipser)**.** *Neka je* $k : \mathbb{N} \to \mathbb{N}$ *polinomno izračunljiva funkcija. Tada je* $IP[k] = AM[k + 2]$.

Ideja dokaza $IP[k] = AM[k + 2]$ je slična ideji dokaza $GNI \in AM$. Trebamo koristiti više koraka interakcije, i dokazivač treba uvjeriti verifikatora da je skup slučajnih bitova za koje će dokazivač uvjeriti verifikatora zaista velik. Cijeli dokaz nalazi se u [5].

Poglavlje 3

IP = PSPACE

U ovom potpoglavlju konačno ćemo smjestiti klasu IP u hijerarhiju složenosti klasa.

Teorem 3.1. *IP = PSPACE*

Prije samog dokaza, uvest ćemo neke nove tehnike i rezultate. Za početak, definirat ćemo aritmetizaciju, sumcheck protokol, pokazati da su #3SAT i TQBF (od engl. True Quantified Boolean Formula) u IP. Tada će rezultat slijediti iz činjenice da za TQBF vrijedi da je PSPACE-potpun.

3.1 Aritmetizacija

Ideja aritmetizacije je prevesti logičke izraze u aritmetičke. Aritmetizaciju koristimo kad nam je praktičnije baratati aritmetičkim izrazima nego logičkim. Logički izraz s n predikata $p_1, \ldots, p_n$ preslikat ćemo u polinom u n varijabli, $x_1, \ldots, x_n$. Istinu i laž zamijenit ćemo vrijednostima 1 i 0, redom. Računat ćemo nad nekim konačnim poljem $\mathbb{F}$. Logičke operacije zamijenit ćemo aritmetičkim izrazima na sljedeći način:

- $a \wedge b \mapsto a \cdot b$
- $a \vee b \mapsto 1 - (1 - a) \cdot (1 - b)$
- $\neg a \mapsto 1 - a$
- $a \vee b \vee c \mapsto 1 - (1 - a) \cdot (1 - b) \cdot (1 - c)$

Ovim postupkom bilo koju formulu u 3CNF koja sadrži m elementarnih disjunkcija možemo prikazati kao polinom u n varijabli čija je vrijednost 1 ako predikati ispunjavaju formulu, a 0 inače.

3.2 Sumcheck protokol

Ovaj protokol osnova je za dokaz da su #3SAT i TQBF u IP. Ukratko, verifikator će nastojati prebrojati interpretacije za koje je dana formula istinita tako što će formulu aritmetizirati i pokušati odrediti sumu aritmetizirane formule po svim vrijednostima propozicijskih varijabli. Od dokazivača će zahtijevati računanje te sumacije, ali da mu pritom jednu varijablu prepusti kao parametar i vrati pripadni polinom u toj varijabli. Verifikator će zatim provjeriti zbroj vrijednosti polinoma u 0 i 1, odabrati neku slučajnu vrijednost (ne nužno 0 ili 1!) za tu varijablu, te nastaviti rekurzivno dalje provjeravati je li dani polinom ispravan.

Neka su dani polinom $f(x_1, \ldots, x_n)$ stupnja k takav da je $k \leq n^3$ (ako je k prevelik, možemo „ubaciti" dodatne varijable među varijable polinoma bez da ih uključimo u izraz), vrijednost S, te prost broj $p \in [2^n, 2^{n+1}]$. Dat ćemo interaktivni dokaz za tvrdnju

$$S = \sum_{a_1 \in \{0,1\}} \cdots \sum_{a_n \in \{0,1\}} f(a_1, \ldots, a_n) \tag{3.1}$$

gdje se sve operacije evaluiraju modulo p.

Primijetimo da uvrštavanjem vrijednosti $a_2, \ldots, a_n$ u varijable $x_2, \ldots, x_n$, dobivamo polinom

$$g(x_1) = \sum_{a_2 \in \{0,1\}} \cdots \sum_{a_n \in \{0,1\}} f(x_1, a_2, \ldots, a_n) \tag{3.2}$$

stupnja k, za koji vrijedi: ako je tvrdnja (3.1) istinita, onda je $g(0) + g(1) = S$. Određivanje polinoma g je vremenski zahtjevno, ali dovoljno je da to određivanje obavi dokazivač; to se vidi iz sumcheck protokola koji glasi:

1. Ako je $n = 1$, verifikator provjerava je li $g(0) + g(1) = S$. Ako jest, verifikator prihvaća dokaz, a ako nije, odbija ga.

2. Inače, tj. kad vrijedi $n \geq 2$, verifikator zahtijeva od dokazivača da mu vrati $g(x_1)$, kako je definiran u (3.2).

3. Dokazivač šalje verifikatoru polinom h.

4. Verifikator evaluira $h(0) + h(1)$, ako taj zbroj ne iznosi S, verifikator odbija dokaz. Inače, verifikator uniformno slučajno odabire a, i rekurzivno provjerava da vrijedi

$$h(a) = \sum_{a_2 \in \{0,1\}} \cdots \sum_{a_n \in \{0,1\}} f(a, a_2, \ldots, a_n) \tag{3.3}$$

Provjerimo vrijede li potpunost i ispravnost.

Potpunost nije teško pokazati: ako (3.1) vrijedi, pošteni dokazivač će u trećem koraku vratiti upravo polinom g, i verifikator će prihvatiti dokaz.

Dokaz ispravnosti je nešto teži. Pretpostavimo da (3.1) ne vrijedi. Dokažimo da je tada vjerojatnost da verifikator odbije dokaz barem

$$\left(1 - \frac{k}{p}\right)^n \tag{3.4}$$

Ako iskoristimo $k \leq n^3$ i $p \geq 2^n$, onda kad je $n \geq 100$ dobivamo

$$\left(1 - \frac{k}{p}\right)^n \geq 1 - 10^{-15}.$$

što je zadovoljavajuća ograda za uvjet ispravnosti, jer je veća od $\frac{2}{3}$. Ogradu od $1 - 10^{-15}$ dobili smo uvrštavanjem $n = 100$. Deriviranjem možemo pokazati da tvrdnja vrijedi i za $n > 100$. Slijedi induktivni dokaz tvrdnje (3.4).

Ako je $n = 1$, tvrdnja očito vrijedi, jer verifikator može evaluirati $g(0)$ i $g(1)$, te provjeriti je li suma jednaka S.

Pretpostavimo sad da je $n \geq 2$ i da tvrdnja vrijedi za sve polinome u $n - 1$ ili manje varijabli. Ako dokazivač vrati neki polinom h različit od g, onda je razlika s ta dva polinoma različita od nulpolinoma. Polinom s je stupnja najviše k, pa ima najviše k nultočaka, dakle polinomi h i g imaju najviše k točaka u kojima se podudaraju. Iz toga slijedi da kad verifikator uniformno slučajno odabere vrijednost a, s vjerojatnošću barem $1 - \frac{k}{p}$ će vrijediti $g(a) \neq h(a)$, te će dokazivač trebati dokazati netočnu tvrdnju za $n - 1$. Po pretpostavci indukcije, vjerojatnost da to ne uspije je barem $\left(1 - \frac{k}{p}\right)^{n-1}$. Dakle, vjerojatnost da verifikator odbije tvrdnju je

$$\mathbb{P}(\mathtt{out}_V\langle P, V\rangle(x) = 0) \geq \left(1 - \frac{k}{p}\right) \cdot \left(1 - \frac{k}{p}\right)^{n-1} = \left(1 - \frac{k}{p}\right)^n$$

□

Korolar 3.2 (#3SAT ∈ IP)**.** *Neka je P neka 3-KNF logička formula, te je S broj interpretacija za koje je P istinita. Neka je G neki nenegativni cijeli broj. Problem odlučivanja je li S = G nazivamo #3SAT. Tada vrijedi #3SAT ∈ IP.*

Dokaz. Neka je P logička formula navedenog oblika. Aritmetizacijom formule P dobivamo polinom f. Sada možemo provjeriti je li G ispravna vrijednost primjenom sumcheck protokola na polinom f i vrijednost G. □

3.3 TQBF ∈ IP

Pokazali smo da je #3SAT ∈ IP. Preostalo je definirati TQBF i pokazati da je u IP, iz čega će slijediti da je PSPACE u IP, jer je TQBF PSPACE-potpun.

Definicija 3.3. *Kvantificirana logička formula (engl.* Quantified Boolean Formula*, nadalje* QBF*) je logička formula oblika*

$$P = \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_n \phi(x_1, \ldots, x_n)$$

gdje je Φ *neka formula logike sudova u 3-KNF.*

Definicija 3.4. *TBQF je jezik koji se sastoji od svih istinitih QBF.*

Da bismo provjerili je li P u TQBF, dovoljno je provjeriti vrijedi li

$$1 \leq \prod_{a_1 \in \{0,1\}} \sum_{a_2 \in \{0,1\}} \prod_{a_3 \in \{0,1\}} \cdots \sum_{a_n \in \{0,1\}} f(a_1, \ldots, a_n) \tag{3.5}$$

gdje je f formula nastala aritmetizacijom od ϕ.

Teorem 3.5 (Stockmeyer)**.** *TQBF je PSPACE-potpun.*

Dokaz Stockmeyerovog teorema može se pronaći u [8].

Sumcheck protokol primijenili smo za dokaz da je #3SAT $\in$ IP, a veoma sličan protokol primijenit ćemo kako bismo pokazali da je TQBF $\in$ IP.

Teorem 3.6. *TQBF* $\in$ *IP.*

Objasnit ćemo ideju dokaza gornjeg teorema, a čitatelj može pogledati detalje u [2]

Jedna jednostavna ideja je primijeniti gotovo jednak protokol onome za #3SAT $\in$ IP, uz razliku da kad se pojavi produkt evaluiramo $h(0) \cdot h(1)$ umjesto $h(0) + h(1)$. Međutim, problem nastaje time što množenje polinoma povećava stupanj polinoma. Nakon k koraka, polinom može imati stupanj 2^k, što je previše i za samu komunikaciju između dokazivača i verifikatora. Naime, poruke kojima oni komuniciraju smiju biti najviše polinomne dužine u n, a 2^k je više od toga.

Ovo možemo zaobići tako što ćemo pobliže pogledati strukturu polinoma koje šaljemo. Promijenit ćemo formulu P u neku njoj logički ekvivalentnu formulu P' koja aritmetizacijom ne postaje prevelika. Ići ćemo po kvantifikatorima s lijeve strane prema desnoj. Svaki put kad naiđemo na kvantifikator $\forall$ uz varijablu x_i, uzmemo sufiks formule koji počinje s tim kvantifikatorom. Neka je taj sufiks ψ, oblika $\psi = \forall x_i \phi'(x_1, \ldots, x_n)$. Želimo osigurati da se varijable $x_1, \ldots, x_i$ ne pojavljuju u ϕ. To ćemo postići koristeći pomoćne varijable. Konkretno, zamijenit ćemo ψ sa $\psi' = \forall x_i \exists x'_1 \ldots \exists x'_i (x_1 = x'_1) \wedge \ldots (x_i = x'_i) \wedge \phi(x'_1, \ldots, x'_i, x_{i+1}, \ldots, x_n)$. Novonastala formula P' imat će najviše $\mathbb{O}(n^2)$ varijabli, i može se pokazati da će polinomi koji se prenose u sumcheck protokolu biti stupnja najviše 2, pa zbilja možemo koristiti sumcheck protokol.

Još je bitno napomenuti da je u ovom protokolu moguće da je $S \equiv 0 \pmod{p}$, ali uz slučajno odabrani prost broj p iz intervala $[2^n, 2^{2n}]$, vjerojatnost za to je također malena.

3.4 IP = PSPACE

Preostalo je pokazati jednostavniji smjer dokaza, da je IP podskup od PSPACE. Ideja dokaza je koristeći PSPACE stroj (Turingov stroj koji koristi polinomno mnogo memorije) odrediti dokazivač P koji s najvećom vjerojatnošću uvjerava verifikatora i reproducirati njihovu interakciju. Odrediti P možemo simuliranjem svih mogućih interakcija dokazivača i verifikatora te slučajnih bitova koje verifikator koristi. To će biti moguće obaviti u polinomno memorije jer se verifikator izvršava u polinomnom vremenu, pa možemo proći kroz prostor dokazivača u polinomno prostora.

Teorem 3.7. *IP ⊆ PSPACE*

Dokaz. Neka je L neki jezik u IP. Tada postoji verifikator V za L. Pokažimo da postoji PSPACE algoritam koji odlučuje L tako što izračunava vjerojatnost

$$p = \max_P \{\mathbb{P}(\mathtt{out}_V\langle P, V\rangle(x) = 1)\}$$

Ako je $p \geq \frac{2}{3}$, onda po definiciji od IP vrijedi $x \in L$. Ako je $p \leq \frac{1}{3}$, onda $x \notin L$.

Pokažimo kako izračunati vrijednost p. Postoji polinom f takav da se za input x duljine $n = |x|$ verifikator V zaustavlja nakon najviše $f(n)$ koraka. Uz to, V pri svom izvršavanju koristi najviše $f(n)$ slučajnih bitova. Svaki odgovor koji P vraća je duljine najviše $f(n)$. Cijeli prostor interakcija možemo onda prikazati kao stablo s korijenom koje je dubine najviše $f(n)$, te ga možemo obići uz polinomno memorije. Pritom brojimo listove u kojima V prihvaća x i P daje optimalne odgovore, kao i sve listove u kojima P daje optimalne odgovore. Omjer tih vjerojatnosti je p. Valja napomenuti da te vrijednosti mogu postati eksponencijalne, ali je njihov zapis polinomno dug, pa možemo računati s njima. □

Korolar 3.8. *PSPACE = IP.*

Dokaz. Iz teorema 3.7 znamo da je IP potklasa od PSPACE. Iz teorema 3.6 znamo da je TQBF element od IP, iz tog teorema i Stockmeyerovog teorema, koji pokazuje da je TQBF PSPACE-potpun, slijedi i da je PSPACE u IP. Dakle, vrijedi IP = PSPACE. □

Poglavlje 4

Primjena interaktivnih dokaza

Glavna primjena interaktivnih dokaza je u računalnoj sigurnosti. Primjerice, možemo imati dva korisnika koji žele razmijeniti neke povjerljive podatke putem interneta. To mogu biti bankovni i medicinski podaci, ali i uobičajene poruke kojima svakodnevno komuniciramo s drugima. Problem komunikacije putem interneta je da komunikacija sama po sebi nije privatna. Štoviše, veoma je jednostavno pratiti i snimati sav mrežni promet računala koristeći Wireshark ili slične programe. Zato ćemo u nastavku pretpostavljati da su sve poruke kojima korisnici komuniciraju javne. Ono što pruža privatnost je mogućnost šifriranja poruka. U pozadini šifriranja je činjenica da se neki problemi, poput faktoriziranja velikih brojeva, trenutno ne znaju efikasno riješiti, pa je teško dekriptirati šifriranu poruku.

U ovom poglavlju promotrit ćemo neke od problema koji se javljaju: problem autentikacije, kako privatno komunicirati javnim kanalom, kako pošteno baciti novčić preko telefona, te kako nekoga uvjeriti u istinitost neke tvrdnje bez da sazna bilo što novo o toj tvrdnji, osim da je istinita (takozvani zero-knowledge dokaz). Nadalje, navest ćemo neka praktična rješenja navedenih problema. Primjerice, objasnit ćemo RSA algoritam te jednu proceduru autentikacije korisnika koja je slična handshake proceduri TLS protokola. Na kraju poglavlja navest ćemo jedan zanimljiv teorijski rezultat, zero-knowledge dokaz za bilo koji problem u NP.

4.1 Osnovni pojmovi i problemi

U ovom poglavlju pretpostavit ćemo da imamo dvije korisnice, nazovimo ih Ankica i Brankica, koje žele privatno komunicirati javnim kanalom. Ankica će započeti, a Brankica nastaviti interakciju. Uz njih, imamo još jednog korisnika, nazovimo ga Edvard, koji želi doći do njihovih podataka ili zloupotrijebiti njihovu interakciju na neki drugi način.

Kao i kod pisama koja nam dolaze na kućnu adresu, smatramo da korisnici vide samo sadržaj poruka koje su stigle do njih, ne i tko ih je *zapravo* poslao (morat će to zaključiti iz

sadržaja jedne ili više poruka).

Kad kažemo *poruka*, obično mislimo na neki općeniti podatak. U ovom ćemo poglavlju, jednostavnosti radi, pretpostaviti da je poruka zapravo neki broj. Taj broj može biti ogroman, pa smatramo da može kodirati čitav blok teksta.

Za Edvarda pretpostavljamo da:

- može računati kao polinoman vjerojatnosan Turingov stroj,
- vidi sve poruke koje se šalju kanalom,
- može zaustaviti slanje bilo koje poruke te
- može samostalno slati poruke Ankici i Brankici.

Za Ankicu i Brankicu pretpostavljamo da se ponašaju slično kao verifikator u prijašnjim poglavljima. Konkretno, pretpostavljamo da:

- mogu računati kao vjerojatnosni Turingovi strojevi,
- mogu uvjeriti jedna drugu u istinite tvrdnje ako se Edvard ne upliće te
- ne mogu biti uvjerene u neistinite tvrdnje (međusobno ili od strane Edvarda) osim uz zanemarivu vjerojatnost.

Pretpostavljamo i da su svi korisnici upoznati s protokolima koji se koriste. Također ćemo pretpostaviti da su Ankica i Brankica odabrale svaka po jedan podatak koji je svima javno dostupan. Svaki od ta dva podatka naziva se *javni ključ*, a služit će za šifriranje poruka.

Problemi koje ćemo rješavati u ovom poglavlju su sljedeći:

1. Ankica želi poslati podatak Brankici, ali bez da Edvard sazna koji je to podatak.
2. Ankica želi postaviti pitanje Brankici i dobiti Brankičin odgovor. Posebno, Ankica želi biti sigurna da je odgovor poslala Brankica, a ne Edvard.
3. Ankica i Brankica žele slučajno odabrati jedan bit, bez pretpostavke da vjeruju jedna drugoj.
4. Ankica želi uvjeriti Brankicu da neka riječ pripada nekom jeziku, ali bez da Brankica sazna bilo koju dodatnu novu informaciju (koju sama nije mogla izračunati).

4.2 RSA

Prvi problem, problem slanja tajne poruke javnim kanalom, riješit ćemo koristeći RSA algoritam ([7]). On se sastoji se od četiri faze:

1. generiranje javnog ključa,
2. slanje javnog ključa,
3. šifriranje poruke,
4. dešifriranje poruke.

Najprije Brankica generira dva prosta broja, neka su to p i q te nalazi umnožak $n = p \cdot q$. Brojevi p i q trebaju biti veliki i slučajno odabrani. U praksi, zadovoljavajuće je uzeti 1024-bitne proste brojeve. Ključno je da p i q ostanu tajni. Napomenimo da Brankica može generirati proste brojeve jer se slučajni brojevi mogu se brzo generirati, prostost broja može se efikasno provjeriti te prostih brojeva ima dovoljno (za brojeve do k, okvirno svaki $\ln(k)$-ti je prost). Dakle, dovoljno je da Brankica slučajno generira brojeve dok ne naiđe na dva prosta broja.

Drugi korak je slanje javnog ključa. Mi pretpostavljamo da je svačiji broj n javno dostupan. Inače, jednom kad korisnici krenu komunicirati, mogu broj n poslati i u poruci.

Nakon toga slijedi šifriranje poruke. Neka je m poruka koju Ankica želi poslati. Pretpostavljamo da je $m < n$. Inače, Ankica može poslati nekoliko poruka. Ankica tada računa $c = m^e \bmod n$, gdje je e neki dogovoreni eksponent. Ovo se može brzo računati koristeći modularno potenciranje. c je šifrirana poruka koju Ankica šalje Brankici. U praksi se često koristi $e = 65537$, jer je dovoljno velik da je otporan na napade u kojima je poruka malen broj, a istodobno dovoljno malen da potenciranje ne zahtjeva puno množenja.

Na kraju, Brankica dešifrira c. Brankica želi invertirati modularno potenciranje (za složeni modul), no to se u praksi ne može efikasno napraviti za $n = p \cdot q$ bez da se zna p ili q. Međutim, Brankica zna faktorizaciju broja n, i može napraviti sljedeće:

- Neka je $l = \phi(n)$, gdje je ϕ Eulerova funkcija. Tada je $l = (p-1) \cdot (q-1)$ jer su p i q prosti.
- Neka je $d = e^{-1} \bmod l$. Multiplikativni inverz modulo l može se efikasno naći koristeći prošireni Euklidov algoritam.
- Konačno, Brankica računa $c^d \equiv (m^e)^d \equiv m^{e \cdot d} \equiv m \pmod{n}$. Zadnja kongruencija vrijedi zbog Eulerovog teorema [3].

Ideja kojom se postiže sigurnost RSA algoritma je korištenje funkcije koju je jednostavno izračunati za bilo koji ulazni podatak, ali je vremenski veoma zahtjevno invertirati

nepoznati izlazni podatak bez poznavanja faktorizacije broja n. U teorijskom računarstvu, funkcija s tim svojstvima naziva se *jednosmjerna funkcija* (engl. *one-way function*).

Definicija 4.1. *Neka su n, m proizvoljni prirodni brojevi. Funkcija $f : \{0,1\}^n \to \{0,1\}^m$ je **jednosmjerna** ako se f može izračunati u polinomnom vremenu, ali bilo koji polinomni vjerojatnosni Turingov stroj F koji pokušava invertirati f ne uspijeva to napraviti osim sa zanemarivom vjerojatnošću. Preciznije, f je jednosmjerna ako za svaki vjerojatnosni Turingov stroj F, za sve prirodne brojeve c, i za sve dovoljno velike prirodne brojeve n vrijedi:*

$$\mathbb{P}[f(F(f(x))) = f(x)] < n^{-c}$$

gdje je $x \in_R \{0,1\}^n$.

Ne zna se je li modularno potenciranje jednosmjerna funkcija, kao ni druge slične funkcije korištene u kriptografiji, poput SHA256. Štoviše, nije poznato postoji li ijedna jednosmjerna funkcija. Međutim, praktična važnost te i drugih kriptografskih funkcija je dobra motivacija za definiranje i razmatranje jednosmjernih funkcija. Jednu posljedicu postojanja jednosmjerne funkcije pokazuje sljedeći teorem.

Teorem 4.2. *Ako postoji jednosmjerna funkcija f, onda je $P \neq NP$.*

Skica dokaza. Neka je f jednosmjerna funkcija. Promotrimo problem traženja inverza od f. Po definiciji jednosmjerne funkcije, Turingovim strojem ne možemo odrediti inverz u polinomnom vremenu. Međutim, za dane x i y, lako je provjeriti vrijedi li $f(x) = y$. Time smo dobili problem koji je u NP i nije u P, pa vrijedi P $\neq$ NP. □

Postoji razni problemi koji se mogu javiti u ovisnosti o porukama koje Ankica šalje Brankici, odnosno odabiru p i q. Primjerice, ako je malen skup vrijednosti S iz kojeg Ankica bira podatak x čiju hash vrijednost $f(x)$ želi poslati, onda Edvard može proći po svim vrijednostima $y \in S$, za svaku vrijednost odrediti $f(y)$, i time kad dobije $f(x) = f(y)$ invertirati f. Ovo se može jednostavno izbjeći dodavanjem dovoljno dugog slučajno generiranog prefiksa na kraj poruke.

Uz to, na kvantnim računalima je n moguće faktorizirati u polinomnom vremenu koristeći Shorov algoritam. Međutim, kvantno računanje je veoma nestabilno zbog fizikalnih ograničenja, pa je RSA i dalje u praksi siguran od napada koji koriste kvantna računala [1]. Primjerice, 2012. godine je, primjenom Shorovog algoritma, uspješno faktoriziran broj 21 [6], dok je 2019. godine bezuspješno prošao pokušaj faktoriziranja broja 35 [1].

4.3 Autentikacija

Sljedeći problem kojim se bavimo je problem autentikacije: kako Ankica i Brankica mogu znati da komuniciraju međusobno unatoč Edvardovom uplitanju? Želimo izbjeći sve si-

tuacije u kojima Edvard može saznati neke privatne informacije ili može predstaviti neku svoju informaciju kao Ankičinu ili Brankičinu. U tim situacijama želimo da primatelj modificirane poruke shvati da je nešto pošlo po zlu.

Napomenimo da je moguće da Edvard jednostavno ne propušta slanje nemodificiranih poruka. Tada Ankica i Brankica neće moći komunicirati, ali tu ne mogu ništa napraviti. To je IPS gdje dokazivač koji se ne pridržava protokola često neće uvjeriti verifikatora da neka riječ zbilja pripada jeziku.

Prvo trebamo protokol kojim će se Ankica i Brankica uvjeriti da zbilja komuniciraju međusobno. Možemo pretpostaviti da su njihovi javni ključevi generirani kao RSA javni ključevi iz prethodnog potpoglavlja. Jedino što ih zapravo razlikuje od Edvarda je to što znaju faktorizirati svoje javne ključeve, i time dešifrirati poruke šifrirane tim javnim ključevima. Zato Ankica može poslati Brankici jednokratnu slučajno generiranu poruku (nadalje *nonce*, od engl. *number used only once*) šifriranu Brankičinim javnim ključem i zahtijevati da joj Brankica pošalje nazad dešifrirani nonce. Brankica se na sličan način može uvjeriti da komunicira s Ankicom, a ne s Edvardom.

Jednostavnosti radi, pretpostavimo da su Ankičin i Brankičin javni ključ 1024-bitni RSA ključevi kao oni opisani u prethodnom poglavlju i da je poruka znatno manja od toga, primjerice da stane u 512 bitova. Konkretan protokol glasi:

- Ankica generira 128-bitni nonce (jednokratnu slučajnu odabranu vrijednost) m_A.
- Ankica šifrira m_A koristeći Brankičin javni ključ n_B i eksponent e te dobiva vrijednost $c_A = m_A^e \bmod n_B$.
- Ankica šalje Brankici vrijednost c_A.
- Brankica dešifrira c_A (čime saznaje m_A) i generira svoj 128-bitni nonce m_B.
- Brankica generira vrijednost m kao uređeni par vrijednosti (m_A, m_B) (primjerice, kodira ga u obliku $m = m_A \cdot 2^{128} + m_B$).
- Brankica šifrira m koristeći Ankičin javni ključ n_A i eksponent e te dobiva vrijednost $c_B = m^e \bmod n_A$.
- Brankica šalje Ankici vrijednost c_B.
- Ankica dešifrira c_B i provjerava da je m_A ispravan.
- Ankica Brankici šalje dešifrirani m_B šifriran s n_B.
- Brankica dešifrira pristiglu poruku i provjerava njenu ispravnost.

Jednom kad se Ankica i Brankica uvjere da komuniciraju međusobno, dogovorit će se oko parametara za komunikaciju, ako postoje. Primjerice, ako koriste RSA za komunikaciju u oba smjera, ne trebaju nikakve dodatne parametre. Ipak u praksi se često koriste drugi algoritmi, koji zahtijevaju nešto više dogovaranja. Svaka od tih poruka u sebi mora sadržavati dešifrirani nonce prethodne poruke i novi nonce. U praksi, primjerice u TLS protokolu se zahtjeva i više: potpisuje se i šalje transkript čitave prethodno obavljene komunikacije.

Ako Ankica i Brankica nastave koristiti RSA, nastavit će na isti način, slanjem jedne po jedne poruke, svaki put uz pripadne nonce vrijednosti. Dobar zadatak za vježbu je uvjeriti se zašto su nam potrebni u svakoj poruci, te što ako imamo poruku koja je preduga (tj. vrijednost m je veća od n).

4.4 Bacanje novčića preko telefona

Sljedeći problem je problem odabira slučajne vrijednosti javnim kanalom bez pretpostavke da Ankica i Brankica vjeruju jedna drugoj. Svaka od njih može zasebno odabrati neku slučajnu vrijednost. Najjednostavnije rješenje bilo bi da Ankica pošalje Brankici neku vrijednost, a Brankica se složi s tom vrijednosti. To rješenje nije zadovoljavajuće, jer Brankica ne vjeruje Ankici da je zbilja odabrala slučajno rješenje.

Za rješenje ovog problema, prisjetimo se jednosmjernih funkcija. Ankica i Brankica mogu se dogovoriti oko odabira jednosmjerne funkcije f koja uređeni par (a, p) preslikava u vrijednost h.

Jednostavnosti radi, pretpostavimo da žele dobiti cijeli broj između 0 i $n - 1$, gdje je n neki prirodni broj. Slijedi protokol odabira slučajne vrijednosti:

1. Ankica i Brankica slučajno odabiru svaka po jedan prirodni broj između 0 i $n - 1$. Neka su ti brojevi a i b, redom.
2. Ankica odabire i prirodni broj p.
3. Ankica izračunava $h = f(a, p)$ i šalje Brankici vrijednost h.
4. Brankica šalje Ankici vrijednost b.
5. Ankica šalje Brankici vrijednosti a i p.
6. Brankica provjerava vrijedi li $h = f(a, p)$.
7. Vrijednost $v = (a + b) \bmod n$ postaje slučajno odabrana vrijednost koja se traži u problemu.

Glavna ideja protokola je da Ankica odabere neku vrijednost a i zapečati je. Za vrijeme obavljanja nekog dijela interakcije, vrijednost a je privatna. Međutim, nakon toga Ankica može otkriti tu vrijednost. Bitno je napomenuti da Brankica ne može efikasno invertirati funkciju f i time prijevremeno saznati vrijednost a, ali može na kraju efikasno provjeriti da je $h = f(a, b)$.

Vrijednost p je ovdje iz praktičnih razloga: ako je n malen, Brankica može proći po svim vrijednostima manjim od n i provjeriti koja se vrijednost preslikava u h. Primjerice, ovo je velik problem kad je $n = 2$, to jest kad Ankica i Brankica trebaju samo jednu binarnu vrijednost. Tada vrijednost p onemogućuje Brankici iteriranje po svim mogućim vrijednostima.

Ovaj protokol (izuzev 4. i 7. koraka) je osnova *sheme pečaćenja bitova*.

Definicija 4.3. Shema pečaćenja bitova *(engl.* bit-commitment scheme*) je protokol između dva vjerojatnosna polinomna Turingova stroja S i R (pošiljatelj i primatelj) u dvije faze u kojem S može zapečatiti neku privatnu vrijednost $a \in \{0, 1\}$ na način da vrijedi:*

- *Ulaz: zajednički ulaz za oba stroja je vrijednost n (tzv.* sigurnosni parametar*). Privatni ulaz stroja S je binarna vrijednost a.*
- Tajnost*: Na kraju prve faze, R nema nikakvo znanje o vrijednosti a. Drugim riječima, za bilo koji vjerojatnosni polinomni Turingov stroj R^* i za bilo koji $n \in \mathbb{N}$, vrijednosti $\mathtt{out}_{R^*}\langle S(0), R^*\rangle(1^n)$ i $\mathtt{out}_{R^*}\langle S(1), R^*\rangle(1^n)$ ne mogu se efikasno razlučiti.*
- Jedinstvenost*: Postoji točno jedna vrijednost koju primatelj može prihvatiti kao originalno zapečaćenu vrijednost. Ovo mora vrijediti čak i ako je pošiljatelj zlonamjeran.*
- Održivost*: Ako se pošiljatelj i primatelj pridržavaju protokola, na kraju druge faze primatelj saznaje vrijednost a i može provjeriti njenu ispravnost.*

4.5 Zero-knowledge dokazi za NP

Kao što smo spomenuli ranije, ponekad će Ankica htjeti uvjeriti Brankicu u neku tvrdnju, primjerice da neka riječ pripada nekom jeziku, ali na način da Brankica ne sazna ništa što sama nije mogla izračunati, osim da riječ zbilja pripada tom jeziku. Ovakvu vrstu interaktivnih dokaza nazivamo zero-knowledge dokazima. U ovom potpoglavlju dokazat ćemo da takav sustav može prepoznati bilo koji jezik u NP, uz pretpostavku da postoje jednosmjerne funkcije. Prvo ćemo navesti definiciju zero-knowledge dokaza. Nakon toga, navest ćemo protokol za prepoznavanje 3-obojivosti grafova koji koristi zero-knowledge dokaz. Problem 3-obojivost grafova je NP-potpun, pa bilo koji problem u NP možemo odlučiti koristeći taj protokol.

Definicija koju ćemo navesti koristi se za jezike u klasi NP. Zero-knowledge dokazi mogu se koristiti i izvan NP, ali time se ne bavimo u ovom poglavlju, u kojem ćemo navesti samo neke osnove zero-knowledge dokaza. Oni su, kao i neke druge teme na kojima se bazira moderna kriptografija, opširnije obrađeni u [4].

Definicija 4.4. *Neka je (P, V) IPS za jezik L. Kažemo da je P* ***zero-knowledge dokaz*** *ako za svaki vjerojatnosni polinomni Turingov stroj V^* postoji vjerojatnosni polinomni Turingov stroj M^* takav da za svaki $x \in L$ i pripadni certifikat u vrijedi da su slučajne varijable $\mathtt{out}_{V^*}\langle P(x,u), V^*(x)\rangle$ i $M^*(x)$ jednako distribuirane. Turingov stroj M^* nazivamo* ***simulatorom*** *za V^* jer simulira izlaz interakcije V^* i dokazivača. Napomenimo da V^* može biti bilo koji algoritam, ne nužno algoritam za određivanje 3-obojivosti.*

Za kraj navodimo zero-knowledge dokaz za 3-obojivost grafova. Ukratko, neka je graf G 3-obojiv. Neka je ϕ neko 3-bojanje grafa G. Neka je τ bojanje nastalo slučajnim permutiranjem boja od ϕ. Prvo će dokazivač napraviti pečaćenje svakog čvora bojanja τ. Nakon toga će verifikator odabrati dva susjedna čvora tog grafa, te će dokazivač otkriti boje tih čvorova i otpečatiti ih. Verifikator će provjeriti da su te boje različite i da su bile ispravno zapečaćene. Graf ne može imati više od n^2 bridova, pa je za neispravna bojanja u svakom koraku ponavljanja postupka vjerojatnost nalaženja jednako obojanih susjednih čvorova barem $\frac{1}{n^2}$. Postupak se ponavlja n^{10} puta da bi se verifikator uz veliku vjerojatnost uvjerio u postojanje bojanja.

Slijedi zero-knowledge protokol za 3-obojivost. Pretpostavljamo da je 3-obojiv graf G na ulazu i verifikatora i dokazivača te da je pripadno 3-bojanje ϕ samo na ulazu dokazivača. Neka je n broj čvorova u G.

1. Dokazivač slučajno odabire neku permutaciju $\pi : \{1,2,3\} \to \{1,2,3\}$.
2. Dokazivač generira 3-bojanje τ tako da za svaki čvor (a, c) iz ϕ koji u grafu G označava čvor a te je boje c u bojanje τ doda čvor $(a, \pi(c))$ koji i dalje označava čvor a, ali je boje $\pi(c)$. Drugim riječima, τ nastaje tako da se čvorovima u ϕ permutiraju boje koristeći permutaciju π.
3. Dokazivač pečati svaki čvor u τ te ih sve šalje verifikatoru. Pritom u svakom čvoru njegovoj boji dodaje neki slučajno generirani sufiks binarnih znamenaka duljine n.
4. Verifikator odabire dva susjedna čvora a i b iz G te ih šalje dokazivaču. Odabire ih uniformno slučajno iz skupa svih bridova iz G.
5. Dokazivač šalje verifikatoru boje c_a i c_b čvorova a i b te njihove slučajno generirane sufikse.
6. Verifikator provjerava da vrijedi $c_a \neq c_b$ te da c_a i c_b odgovaraju zapečaćenim vrijednostima.

7. Prvih šest koraka ponavljamo n^{10} puta. Ako je svaki put u šestom koraku vrijedilo $c_a \neq c_b$ i vrijednosti su odgovarale zapečaćenim vrijednostima, verifikator prihvaća dokaz, a inače ga odbija.

Provjerimo vrijede li potpunost i ispravnost.

Potpunost očito vrijedi. Ako je graf 3-obojiv, onda dokazivač prati protokol i u svakom ponavljanju šestog koraka protokola vrijedit će svi uvjeti, jer je graf 3-obojiv.

Ispravnost također očito vrijedi. Ako graf nije 3-obojiv, onda u svakom ponavljanju protokola verifikator s vjerojatnošću barem $\frac{1}{n^2}$ da bira čvorove koji su iste boje ili neispravno zapečaćeni. Ograda $\frac{1}{n^2}$ vrijedi jer graf ima manje od n^2 bridova. Dakle, ponavljamo proces n^{10} puta, i vjerojatnost uspješnog pronalaska greške je u svakom koraku barem $\frac{1}{n^2}$, pa je ukupna vjerojatnost pronalaska greške barem $\frac{2}{3}$.

Dokazali smo da je to IPS, preostalo je dokazati da je zero-knowledge, za što ćemo navesti grubu skicu dokaza tako da konstruiramo simulator. Neka je V^* neki (možda zlonamjerni) vjerojatnosni polinomni Turingov stroj. Stroj M^* konstruiramo tako da i u potpunosti simulira rad stroja V^*, i da simulira interakciju strojeva V^* i P. M^* simulira rad stroja V^* tako što prati sve vrijednosti na svim trakama koje V^* koristi. Pritom, kad V^* generira slučajne vrijednosti, onda ih generira i M^*, čime distribucije vrijednosti traka ostaju jednake. Preostalo je simulirati interakciju V^* i P. M^* nastoji „pogoditi" koji brid (a, b) će V^* odabrati za provjeru. M^* može uniformno slučajno obojati sve čvorove. To uglavnom neće biti ispravno bojanje, ali ako su u njemu a i b različito obojani, V^* ne može uočiti razliku, te distribucija vrijednosti koje traka može poprimiti ostaje nepromijenjena. Ako su u njemu a i b jednako obojani, onda M^* može simulaciju premotati nazad do zadnjeg generiranja bojanja i probati ponovo. Svi neuspjeli pokušaji se zanemaruju. M^* ponavlja pokušaje dok neki od njih ne uspije. Kad uspije, distribucija vrijednosti traka ostaje nepromijenjena. Napomenimo da je za tri boje očekivani broj premotavanja za svaki pogodak konstantan.

Potpuniji dokaz je tehničke prirode, preostalo je precizno raspisati ove ideje. Formalni dokaz, kao i više o zero-knowledge dokazima, može se pronaći u [4].

Bibliografija

[1] M. Amico, Z. H. Saleem i M. Kumph, *Experimental study of Shor's factoring algorithm using the IBM Q Experience*, Physical Review A **100** (2019), br. 1.

[2] S. Arora i B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[3] A. Dujella, *Teorija brojeva*, Školska knjiga, 2019.

[4] O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001.

[5] S. Goldwasser i M. Sipser, *Private coins versus public coins in interactive proof systems*, STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing (1986), 59–68.

[6] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, Xiao Qi Z. i J. L. O'Brien, *Experimental realization of Shor's quantum factoring algorithm using qubit recycling*, Nature Photonics **6** (2012), br. 11, 773–776.

[7] R. Rivest, A. Shamir i L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, CACM **21** (1978), 120–126.

[8] L. J. Stockmeyer i A. R. Meyer, *Word problems requiring exponential time*, STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing (1973), 1–9.

[9] M. Vuković, *Složenost algoritama*, (2019), `https://www.pmf.unizg.hr/_download/repository/SA-skripta-2019%5B1%5D.pdf`.

[10] V. Čačić, *Komputonomikon*, (2021), `https://web.math.pmf.unizg.hr/~veky/izr/Komputonomikon.pdf`.

Sažetak

U ovom radu opisujemo IP, klasu problema rješivih u polinomnom vremenu koristeći sustave interaktivnih dokaza.

Sustavi interaktivnih dokaza koriste interakciju dva stroja, dokazivača i verifikatora, za odlučivanje pripadnosti neke riječi nekom formalnom jeziku. Dokazivač je stroj s neograničenim vremenskim i memorijskim računskim mogućnostima, dok je verifikator Turingov stroj kojem dodatno omogućavamo generiranje slučajnih bitova. Dokazivač pokušava uvjeriti verifikatora u pripadnost riječi jeziku, dok verifikator pokušava provjeriti istinitost te tvrdnje. Za razliku od mnogih drugih klasa, dopuštamo verifikatoru malu mogućnost pogreške.

Omogućavanje generiranja slučajnih bitova i dopuštanje pogrešaka možda ne izgleda značajno, ali ispostavlja se da je klasa IP ekvivalentna klasi PSPACE, što i dokazujemo u ovom radu. Razmatramo i varijantu interaktivnih dokaza u kojoj su verifikatorovi slučajno generirani bitovi javno dostupni. U posljednjem poglavlju navodimo neke primjene interaktivnih dokaza u kriptografiji.

Summary

In this paper, we present IP, the class of problems solvable in polynomial time using interactive proof systems.

Interactive proof systems use the interaction of two machines, a prover and a verifier, to check if some word is in some formal language. The prover is a machine with unlimited computing power, while the verifier is a Turing machine which can generate random bits. The prover's task is to convince the verifier that the word is indeed in the language, while the verifier's task is to check the truthfulness of that claim. We allow a small chance of the verifier failing in its task.

Enabling random bit generation and allowing mistakes might look insignificant, but it turns out that the class IP is equivalent to the class PSPACE, which we prove in this paper. We also consider a variant of interactive proofs in which the verifier's randomly generated bits are publicly available to the prover. In the final chapter, we mention some applications of interactive proofs in cryptography.

Životopis

Rođen sam 1996. u Puli. Po završetku osnovne škole upisujem prirodoslovno-matematički smjer Gimnazije Pula. 2015. godine upisujem preddiplomski studij Matematika na zagrebačkom Prirodoslovno-matematičkom fakultetu. 2019. godine završavam preddiplomski studij i upisujem diplomski studij Računarstvo i matematika na istom fakultetu.