# Binarni RAM--strojevi

**Miletić, Jurica**

# Binarni RAM--strojevi

**Miletić, Jurica**

**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO–MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Jurica Miletić

# BINARNI RAM-STROJEVI

Diplomski rad

Voditelj rada:
doc. dr. sc. Vedran Čačić

Zagreb, studeni 2020

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. _____

*Svojoj obitelji, prijateljima, mentoru i svima koji su mi pomogli na studentskom putu da ostvarim željeni cilj stjecanja znanja i primanja diplome kao potvrdu za to.*

# Sadržaj

# Uvod

U ovom diplomskom radu obrađuje se tema binarnih RAM-strojeva. Binarni RAM-stroj predstavlja model izračunavanja koji je definiran po uzoru na RAM-stroj, s tim da se u registrima binarnog RAM-stroja spremaju prirodni brojevi u binarnom zapisu. Samim time, instrukcije koje ćemo koristiti pri definiranju binarnog RAM-stroja bit će drugačije. Kroz rad ćemo binarni RAM-stroj zvati BRAM-stroj. Po uzoru na [1], definirat ćemo binmakro-stroj i ostale binmakro-izvedenice za BRAM-stroj.
Definirat ćemo šest tipova BRAM-instrukcija pomoću kojih ćemo izgraditi funkcijski binmakro te binmakro-programe za zbrajanje i množenje svih mjesnosti. Napravljena je i implementacija BRAM-instrukcija te izvedenih programa za funkcije zbrajanja i množenja u programskom jeziku Python, kako bi se i empirijski provjerila valjanost BRAM-programa i binmakroa koji će biti napisani u ovom radu.
Definirat ćemo pojmove poput BRAM-izračunljivosti funkcije, vremenske složenosti BRAM-algoritma i polinomne izračunljivosti funkcije na BRAM-stroju. To ćemo sve napraviti kako bismo pronašli totalno brojevne funkcije koje će biti izračunljive u polinomnom vremenu na BRAM-stroju.

## Oznake

Navest ćemo u uvodu neke oznake koje ćemo koristiti u radu i njihovo značenje. Za $a, b \in \mathbb{N}$ definiramo

$$[a..b] \coloneqq [a, b] \cap \mathbb{N} [a..b\rangle \coloneqq [a, b\rangle \cap \mathbb{N} \langle a..b\rangle \coloneqq \langle a, b\rangle \cap \mathbb{N}$$

Oznake $\lfloor \frac{a}{2} \rfloor$ i $a \bmod 2$ označavaju količnik i ostatak (cjelobrojnog) dijeljenja broja $a \in \mathbb{N}$ s 2.

# Poglavlje 1

# Binarna RAM-izračunljivost

Po uzoru na [1] model izračunavanja koji ćemo definirati bit će BRAM-stroj. Binarna RAM-izračunljivost je svojstvo funkcija koje mogu biti izračunate na binarnom RAM-stroju, te ćemo odsad te pojmove skraćeno nazivati BRAM-izračunljivost i BRAM-stroj.

**Definicija 1.1:** *BRAM-stroj* je matematički (idealizirani) stroj koji sadrži:

- *BRAM-program*: fiksni konačni niz instrukcija $P := (I_0, I_1, \ldots, I_{n-1})$;
- *registre*: za svaki $j \in \mathbb{N}$, registar $\mathcal{R}_j$, koji može sadržavati bilo koji prirodni broj (u binarnom zapisu);
- *programski brojač* (PC): još jedan „registar", koji u svakom trenutku izračunavanja sadrži broj iz intervala $[0..n]$.

‘ ◁

BRAM-program najčešće pišemo kao

$$P := \begin{bmatrix} 0.\ I_0 \\ 1.\ I_1 \\ \vdots \\ (n-1).\ I_{n-1} \end{bmatrix}, \text{ ili skraćeno } P := \Big[\ t.\ I_t\ \Big]_{t<n}$$

Broj instrukcija programa $P$ zovemo još *duljinom* programa $P$, i označavamo ga s $n_P$.

Sadržaj registara se može mijenjati za vrijeme izvršavanja programa, ovisno o instrukcijama. Početni sadržaj određen je ulaznim podacima. Irelevantni registri (koji se ne spominju u instrukcijama, niti služe za ulaz) sadrže vrijednost 0.

Sadržaj programskog brojača također se mijenja, tako da se izvršavanjem svake instrukcije poveća za 1, osim ako sama instrukcija kaže drugačije. Početna vrijednost programskog brojača je 0. U svakom trenutku sadržaj programskog brojača je redni broj instrukcije koja se trenutno izvršava, dok vrijednost $n$ označava kraj izvođenja programa.

**Definicija 1.2:** Svaka *BRAM-instrukcija* ima:

- (ako je dio BRAM-programa $P$) *redni broj*, element skupa $[0..n_P\rangle$;
- *tip*, koji može biti jedan od šest tipova: SHR, SHL, SHS, BZ, ODD, GO TO;
- (ako je tipa SHR, SHL, SHS, BZ ili ODD) *registar* na kojem djeluje ($\mathcal{R}_j$ za neki $j \in \mathbb{N}$);
- (ako je tipa BZ, ODD ili GO TO, te je dio BRAM-programa $P$) *odredište*: element skupa $[0..n_P]$.

$\triangleleft$

Dakle, BRAM-instrukcija može biti jednog od šest oblika, čiji efekti su:

SHR $\mathcal{R}_j$: Cjelobrojno dijeljenje sadržaja registra $\mathcal{R}_j$ s 2.

SHL $\mathcal{R}_j$: Množenje sadržaja registra $\mathcal{R}_j$ s 2.

SHS $\mathcal{R}_j$: Množenje sadržaja registra $\mathcal{R}_j$ s 2 i dodavanje jedinice.

BZ $\mathcal{R}_j, l$: Postavljanje PC na $l$ ako je sadržaj registra $\mathcal{R}_j$ jednak nuli.

ODD $\mathcal{R}_j, l$: Postavljanje PC na $l$ ako za sadržaj $r_j$ registra $\mathcal{R}_j$ vrijedi $r_j \bmod 2 = 1$, odnosno da je $r_j$ neparan.

GO TO $l$: Bezuvjetno postavljanje PC na $l$.

**Lema 1.3:** Skup $\mathcal{I}ns$ svih BRAM-instrukcija je prebrojiv.

*Dokaz.* Dokaz je analogan dokazu prebrojivosti RAM-instrukcija u [1].
Skup $\mathcal{I}ns_{\text{SHR}}$ svih instrukcija tipa SHR je prebrojiv: preslikavanje $f_1 : \mathbb{N} \to \mathcal{I}ns_{\text{SHR}}$ zadano s $f_1(j) := (\text{SHR } \mathcal{R}_j)$ je bijekcija. Analogno dokažemo prebrojivost skupova $\mathcal{I}ns_{\text{SHL}}$ i $\mathcal{I}ns_{\text{SHS}}$. Analogno, koristeći odredište (iako je broj odredišta ograničen za fiksni program $P$, svaka instrukcija GO TO $l$ se može pojaviti u *nekom* programu), skup $\mathcal{I}ns_{\text{GO TO}}$ je prebrojiv, a skupovi $\mathcal{I}ns_{\text{BZ}}$ i $\mathcal{I}ns_{\text{ODD}}$ su ekvipotentni s $\mathbb{N} \times \mathbb{N}$, pa su i oni prebrojivi. Sada je $\mathcal{I}ns$ prebrojiv kao (disjunktna) unija tih šestero prebrojivih skupova. $\square$

**Korolar 1.4:** Skup $\mathcal{P}rog$ svih BRAM-programa je prebrojiv.

*Dokaz.* Direktno iz činjenice da je $\mathcal{I}ns_{\text{SHR}}{}^{*} \subseteq \mathcal{P}rog \subseteq \mathcal{I}ns^{*}$, i leme 1.3. Iz teorije skupova znamo da je skup $A^{*}$ prebrojiv ako je $A$ prebrojiv, te tvrdnja u korolaru slijedi iz toga da $\mathcal{P}rog$ mora biti prebrojiv ako ima prebrojiv podskup i nadskup. □

S definiranim instrukcijama, programom i strojem, definirat ćemo izvršavanje programa na BRAM-stroju po uzoru na izvršavanje programa na RAM-stroju [1].

**Definicija 1.5:** Neka je $\mathcal{S}$ BRAM-stroj s programom $P$, registrima $\mathcal{R}_j$, $j \in \mathbb{N}$ te programskim brojačem PC. *Konfiguracija* BRAM-stroja $\mathcal{S}$ je bilo koje preslikavanje

$$c : \{\mathcal{R}_j \mid j \in \mathbb{N}\} \cup \{\text{PC}\} \to \mathbb{N} \tag{1.1}$$

takvo da je skoro svuda 0 (skup $c^{-1}[\mathbb{N}_{+}]$ je konačan), a $c(\text{PC}) \leq n_P$. Skraćeno je pišemo kao $c = (c(\mathcal{R}_0), c(\mathcal{R}_1), \ldots, c(\text{PC}))$. Konfiguracija $c$ je *završna* ako je $c(\text{PC}) = n_P$. *Početna konfiguracija* je konfiguracija $c$s za koju je $c(\text{PC}) = 0$. *Početna konfiguracija s ulazom* $\vec{x} = (x_1, x_2, \ldots, x_k) \in \mathbb{N}^k$ je $(0, x_1, x_2, \ldots, x_k, 0, 0, \ldots, 0)$ (u $\mathcal{R}_j$ je $x_j$ za $j \in [1..k]$, a svugdje drugdje su nule).
Za konfiguracije $c = (r_0, r_1, \ldots, pc)$ i $d = (r_0', r_1', \ldots, pc')$ istog BRAM-stroja s programom $P = (I_0, \ldots, I_{n_P-1})$, kažemo da $c$ *prelazi* u $d$ (po programu $P$, ili po instrukciji $I_{pc}$), i pišemo $c \rightsquigarrow d$, ako vrijedi jedno od sljedećeg:

1. $c$ je završna ($pc = n_P$) i $c = d$ — to skraćeno pišemo $c \circlearrowleft$;
2. $I_{pc} = \text{SHR}\ \mathcal{R}_j$ (za neki $j$), $r_j' = \lfloor \frac{r_j}{2} \rfloor$, $pc' = pc + 1$ te $r_i' = r_i$ za sve $i \neq j$;
3. $I_{pc} = \text{SHL}\ \mathcal{R}_j$ (za neki $j$), $r_j' = r_j \cdot 2$, $pc' = pc + 1$ te $r_i' = r_i$ za sve $i \neq j$;
4. $I_{pc} = \text{SHS}\ \mathcal{R}_j$ (za neki $j$), $r_j' = r_j \cdot 2 + 1$, $pc' = pc + 1$ te $r_i' = r_i$ za sve $i \neq j$;
5. $I_{pc} = \text{ODD}\ \mathcal{R}_j, l$ (za neke $j$ i $l$), $r_j \bmod 2 = 0$, $pc' = pc + 1$ te $r_i' = r_i$ za sve $i$;
6. $I_{pc} = \text{ODD}\ \mathcal{R}_j, l$ (za neke $j$ i $l$), $r_j \bmod 2 = 1$, $pc' = l$ te $r_i' = r_i$ za sve $i$;
7. $I_{pc} = \text{BZ}\ \mathcal{R}_j, l$ (za neke $j$ i $l$), $r_j \neq 0$, $pc' = pc + 1$ te $r_i' = r_i$ za sve $i$;
8. $I_{pc} = \text{BZ}\ \mathcal{R}_j, l$ (za neke $j$ i $l$), $r_j = 0$, $pc' = l$ te $r_i' = r_i$ za sve $i$;
9. $I_{pc} = \text{GO TO}\ l$ (za neki $l$), $pc' = l$ te $r_i' = r_i$ za sve $i$. ◁

**Lema 1.6:** Svaka konfiguracija prelazi u neku, jedinstvenu, konfiguraciju.

*Dokaz.* Dokazat ćemo ovo svojstvo za BRAM-stroj na analogan način kako i u [1] za RAM-stroj.
Neka je $\mathcal{S}$ BRAM-stroj s programom $(I_0, I_1, \ldots, I_{n-1})$ te $c = (r_0, r_1, \ldots, pc)$ proizvoljna

konfiguracija. Po definiciji je $pc \leq n$ — ako vrijedi jednakost, $c$ je završna pa po pravilu 1 iz definicije 1.5 prelazi u samu sebe (nijedno drugo pravilo nije primjenjivo jer $I_{pc}$ ne postoji). Ako je pak $pc < n$, pogledajmo tip od $I_{pc}$. Ako je to SHR, SHL, SHS, ili GO TO, pravila 1, 2, 3, odnosno 9, točno propisuju novu konfiguraciju u koju $c$ prelazi.
Inače, $I_{pc}$ je tipa ODD ili BZ, recimo ODD $\mathcal{R}_j, l$ ili BZ $\mathcal{R}_j, l$, i tada je opet nova konfiguracija jedinstveno određena, s obzirom na $r_j$. U slučaju da je $I_{pc}$ jednak ODD $\mathcal{R}_j, l$, ako je $r_j$ djeljiv s 2 ($r_j \bmod 2 = 0$) tada je primjenjivo samo pravilo 5. Inače, ako $r_j$ nije djeljiv s 2 ($r_j \bmod 2 = 1$), primjenjivo je samo pravilo 6. U slučaju da je $I_{pc}$ jednaka BZ $\mathcal{R}_j, l$, ako je $r_j \neq 0$, tada je primjenjivo samo pravilo 7. Inače, ako je $r_j = 0$, primjenjivo je samo pravilo 8. Svako od tih pravila (5–8) također jednoznačno određuje novu konfiguraciju. □

**Definicija 1.7:** Neka je $P$ BRAM-program i $c$ početna konfiguracija (stroja s programom $P$). *$P$-izračunavanje s $c$* je niz konfiguracija $(c_n)_{n\in\mathbb{N}}$, takvih da je $c_0 = c$ te za svaki $n \in \mathbb{N}$, $c_n$ prelazi u $c_{n+1}$. Kažemo da to izračunavanje *stane* ako postoji $n_0 \in \mathbb{N}$ takav da je $c_{n_0}$ završna. Također, kažemo da je $P$ *totalan* ako $P$-izračunavanje s $c$ stane za svaku početnu konfiguraciju $c$. ◁

**Propozicija 1.8:** Za svaki BRAM-program $P$, za svaku početnu konfiguraciju $c$ (stroja s programom $P$), postoji jedinstveno $P$-izračunavanje s $c$.

*Dokaz.* Za postojanje, induktivno definiramo

$$c_0 \coloneqq c, \tag{1.2}$$

$$c_{n+1} \coloneqq \text{jedinstvena konfiguracija u koju } c_n \text{ prelazi (prema lemi 1.6).} \tag{1.3}$$

Po Dedekindovom teoremu rekurzije, time je dobro definiran niz, i taj niz je po definiciji $P$-izračunavanje s $c$.

Za jedinstvenost, pretpostavimo da postoje dva $P$-izračunavanja s $c$, $(c_i)_{i\in\mathbb{N}}$ i $(c'_i)_{i\in\mathbb{N}}$. Kako je $c \neq c'$, postoji neki $i \in \mathbb{N}$ takav da je $c_i \neq c'_i$, a zbog dobre uređenosti od $\mathbb{N}$ postoji najmanji takav: označimo ga s $i_0$. Taj $i_0$ nije 0, jer je $c_0 = c$ = početna konfiguracija. Dakle, konfiguracija $c_{i_0-1} = c'_{i_0-1}$ prelazi u dvije različite konfiguracije $c_{i_0}$ i $c'_{i_0}$, što je kontradikcija s lemom 1.6. □

**Propozicija 1.9:** Za svaki BRAM-program $P$, za svaku početnu konfiguraciju $c$ (stroja s programom $P$) za koje $P$-izračunavanje s $c$ stane, postoji jedinstvena završna konfiguracija.

*Dokaz.* Pretpostavimo da je $(c_i)_{i\in\mathbb{N}}$ $P$-izračunavanje s $c$ u kojem postoje dvije završne konfiguracije, i označimo s $i_1$ i $i_2$ indekse na kojima se one prvi put pojavljuju. Bez smanjenja općenitosti (različitost je simetrična) možemo pretpostaviti $i_1 < i_2$. No, jer je $c_{i_1}$ završna, ona prelazi (samo) u samu sebe, pa indukcijom imamo

$$c_{i_1} = c_{i_1+1} = c_{i_1+2} = \cdots = c_{i_2}, \tag{1.4}$$

što je kontradikcija. □

**Korolar 1.10:** Skup $\mathcal{TP}rog$ svih totalnih BRAM-programa je prebrojiv.

*Dokaz.* Lako se vidi da je za svaki $k \in \mathbb{N}$, $\mathcal{I}ns^k_{\text{SHR}}$ totalan BRAM-program, jer za svaku početnu konfiguraciju $c$ (stroja s programom $\mathcal{I}ns^k_{\text{SHR}}$) u $k$ koraka dođemo do završne konfiguracije, odnosno $Ins^k_{\text{SHR}}$-izračunavanje stane u $k$ koraka. Sad tvrdnja korolara slijedi direktno iz činjenice da je $\mathcal{I}ns_{\text{SHR}}{}^* \subseteq \mathcal{TP}rog \subseteq \mathcal{P}rog$, i korolara 1.4. Iz teorije skupova znamo da je skup $A^*$ prebrojiv ako je $A$ prebrojiv, te tvrdnja u korolaru slijedi iz toga da $\mathcal{TP}rog$ mora biti prebrojiv ako ima prebrojiv podskup i nadskup. □

Ono što ćemo obrađivati u radu je izračunljivost totalnih brojevnih funkcija. To su funkcije $f : \mathbb{N}^k \to \mathbb{N}$, za bilo koji $k \in \mathbb{N}_+$. Taj $k$ je jedinstven za funkciju $f$, i zvat ćemo ga *mjesnost* funkcije. Totalnu brojevnu funkciju $f$ ćemo ponekad označavati s $f^k$ ako budemo željeli naznačiti njenu mjesnost. Sad ćemo definirati pojam *BRAM-algoritma* slično kao *RAM-algoritam* u [1]. Ovdje ćemo također imati pojam mjesnosti za BRAM-algoritam $P$ koji će određivati broj ulaznih podataka za početnu konfiguraciju pripadajućeg BRAM-stroja.

**Definicija 1.11:** *BRAM-algoritam* je uređen par BRAM-programa $P$ i mjesnosti $k \in \mathbb{N}_+$. Umjesto $(P, k)$ pišemo $P^k$.

Neka je $P^k$ BRAM-algoritam te $\vec{x} \in \mathbb{N}^k$. *$P$-izračunavanje s $\vec{x}$* je niz konfiguracija $(c_n)_{n\in\mathbb{N}}$, takvih da je $c_0$ početna konfiguracija (stroja s programom $P$) s ulazom $\vec{x}$ te za svaki $n \in \mathbb{N}$, $c_n$ prelazi u $c_{n+1}$. Kažemo da to izračunavanje *stane* ako postoji $n_0 \in \mathbb{N}$ takav da je $c_{n_0}$ završna. Također, kažemo da je $P^k$ *totalan* ako $P$-izračunavanje s $\vec{x}$ stane za svaki $\vec{x} \in \mathbb{N}^k$.

Neka je $P^k$ totalan BRAM-algoritam te $f^k$ totalna brojevna funkcija iste mjesnosti. Kažemo da $P^k$ *računa* funkciju $f$ ako za sve $\vec{x} \in \mathbb{N}^k$ $P$-izračunavanje s $\vec{x}$ stane u konfiguraciji oblika $(f(\vec{x}), \ldots, n_P)$. ◁

Navest ćemo tri posljedice determinizma koje će biti analogne kao i u slučaju determinizma kod RAM-stroja i RAM-izračunavanja u [1], a budući da dokazi tih posljedica ne ovise o načinu zapisa brojeva, ni o tipu instrukcija, dokazi će također biti analogni.

**Korolar 1.12:** Za svaki BRAM-algoritam $P^k$, za svaki ulaz $\vec{x} \in \mathbb{N}^k$,
postoji jedinstveno $P$-izračunavanje s $\vec{x}$.

*Dokaz.* Neka je $c$ početna konfiguracija s ulazom $\vec{x}$. Za postojanje, po propoziciji 1.8 imamo da je postoji jedinstveno $P$-izračunavanje s $c$, a po definiciji to je $P$-izračunavanje s $\vec{x}$.
Za jedinstvenost, pretpostavimo da postoje 2 $P$-izračunavanja s $\vec{x}$. Iz definicije 1.7 slijedi da su to također 2 $P$-izračunavanje s $c$, što je kontradikcija s propozicijom 1.8. □

**Korolar 1.13:** Za svaki BRAM program $P$, za svaki $\vec{x} \in \mathbb{N}^+$ za koje $P$-izračunavanje s $\vec{x}$ stane, postoji jedinstvena završna konfiguracija.

*Dokaz.* Pretpostavimo da je $(c_i)_{i\in\mathbb{N}}$ $P$-izračunavanje s $\vec{x}$ u kojem postoje dvije završne konfiguracije, i označimo s $i_1$ i $i_2$ indekse na kojima se one prvi put pojavljuju. Iz toga i definicije 1.7 slijedi da je to $P$-izračunavanje s $c_0$ u kojem postoje dvije završne konfiguracije što je u kontradikciji s propozicijom 1.9. □

**Korolar 1.14:** Svaki totalan BRAM-algoritam računa jedinstvenu totalnu brojevnu funkciju.

*Dokaz.* Neka je $k \in \mathbb{N}_+$ te $P^k$ totalan BRAM-algoritam. Definirajmo funkciju

$$f(\vec{x}) := c(\mathcal{R}_0), \text{ gdje je } c \text{ završna konfiguracija } P\text{-izračunavanja s } \vec{x} \in \mathbb{N}^k \qquad (1.5)$$

Budući da je $P^k$ totalan, iz toga slijedi da je $f : \mathbb{N}^k \to \mathbb{N}$ dobro definirana ($k$-mjesna) totalna brojevna funkcija, a $P^k$ računa $f$.

Za jedinstvenost, mjesnost funkcije je određena mjesnošću algoritma (uz prethodni dogovor da se prazne funkcije različitih mjesnosti razlikuju), njena domena je određena stajanjem izračunavanja (jedinstvenog zbog propozicije 1.12), a ona je $\mathbb{N}^k$ jer je $P^k$ totalan, a vrijednost funkcije u svakoj točki domene određena je završnom konfiguracijom (koja je jedinstvena zbog propozicije 1.13). □

Važna posljedica prethodnog rezultata je ograničenje broja izračunljivih totalnih brojevnih funkcija.

**Definicija 1.15:** Neka je $k \in \mathbb{N}_+$ te $f^k$ totalna brojevna funkcija. Kažemo da je $f^k$ *BRAM-izračunljiva* ako postoji totalan BRAM-algoritam $P^k$ koji je računa. Za svaki $k \in \mathbb{N}_+$, oznakom $\mathcal{C}omp_k$ označavamo skup svih BRAM-izračunljivih funkcija mjesnosti $k$. ◁

**Korolar 1.16:** Neka je $P$ BRAM-program. Ako je $P$ totalan, tada je BRAM-algoritam $P^k$ totalan za svaki $k \in \mathbb{N}_+$.

*Dokaz.* Neka je $P$ totalan i neka je $\vec{x} \in \mathbb{N}^+$ proizvoljan. Neka je $(c_n)_{n\in\mathbb{N}}$ $P$-izračunavanje s $\vec{x}$. Po definiciji 1.7 i propoziciji 1.8 je to također $P$-izračunavanje s $c_0$, jer je $c_0$ početna konfiguracija s ulazom $\vec{x}$ po definiciji. Budući da je $P$ totalan, slijedi da $P$-izračunavanje s $c_0$ stane, odnosno da postoji završna konfiguracija u $(c_n)_n$, iz čega slijedi da $P$-izračunavanje s $\vec{x}$ stane. Kako je $\vec{x}$ bio proizvoljan, slijedi da $P$-izračunavanje s $\vec{x}$ stane za svaki $\vec{x} \in \mathbb{N}^+$, odnosno da je BRAM-algoritam $P^k$ totalan za svaki $k \in \mathbb{N}_+$. □

**Korolar 1.17:** Skup $\mathcal{TA}lg_k$ svih BRAM-programa $P$ takvih da je $P^k$ totalan BRAM-algoritam je prebrojiv za svaki $k \in \mathbb{N}_+$.

*Dokaz.* Neka je $k \in \mathbb{N}_+$ proizvoljan. Iz korolara 1.16 slijedi da je za svaki totalan BRAM-program $P$, BRAM-algoritam $P^k$ totalan iz čega slijedi da je $\mathcal{TP}rog \subseteq \mathcal{TA}lg_k$. Također, budući da je $\mathcal{TA}lg_k$ skup BRAM-programa, slijedi da je $\mathcal{TP}rog \subseteq \mathcal{TA}lg_k \subseteq \mathcal{P}rog$. Iz

korolara 1.10 i 1.4 imamo da je $\mathcal{TA}lg_k$ prebrojiv jer ima prebrojiv podskup i nadskup. Kako je $k \in \mathbb{N}_+$ bio proizvoljan, slijedi tvrdnja u korolaru. □

**Teorem 1.18:** Za svaki $k \in \mathbb{N}_+$, skup $\mathcal{C}omp_k$ je prebrojiv. Skup $\mathcal{C}omp$ svih BRAM-izračunljivih totalnih brojevnih funkcija (svih mjesnosti) je također prebrojiv.

*Dokaz.* Neka je $k$ fiksna mjesnost. Preslikavanje sa skupa $\mathcal{TA}lg_k$ na skup $\mathcal{C}omp_k$, koje svakom $P \in \mathcal{TA}lg_k$ pridružuje totalnu brojevnu funkciju koju totalan BRAM-algoritam $P^k$ računa, je dobro definirano prema korolaru 1.14, i surjekcija je po definiciji 1.15. Iz toga je $\text{card}(\mathcal{C}omp_k) \leq \text{card}(\mathcal{TA}lg_k)$, što je $\aleph_0$ po korolaru 1.17.

Za drugu nejednakost, uočimo da su za sve $n \in \mathbb{N}$, konstantne totalne brojevne funkcije $\mathsf{C}^k_{2^n}$, zadane sa $\mathsf{C}^k_{2^n}(\vec{x}) := 2^n$, BRAM-izračunljive: doista, računaju ih totalni BRAM-algoritmi

$$P^k_n := \begin{bmatrix} 0.\ \text{SHS}\ \mathcal{R}_0 \\ 1.\ \text{SHL}\ \mathcal{R}_0 \\ 2.\ \text{SHL}\ \mathcal{R}_0 \\ \vdots \\ n.\ \text{SHL}\ \mathcal{R}_0 \end{bmatrix}^k \tag{1.6}$$

(što se može vidjeti indukcijom po $n$). Iz toga slijedi da je $\{\mathsf{C}^k_{2^n} \mid n \in \mathbb{N}\} \subseteq \mathcal{C}omp_k$, a kako je taj skup prebrojiv (sve funkcije $\mathsf{C}^k_{2^n}$ su različite jer je $2^i \neq 2^j$ za $i \neq j$), slijedi $\aleph_0 \leq \text{card}(\mathcal{C}omp_k)$, što zajedno s gornjim daje $\text{card}(\mathcal{C}omp_k) = \aleph_0$.

Sada je $\mathcal{C}omp = \bigcup_{k\in\mathbb{N}_+} \mathcal{C}omp_k$ prebrojiv kao unija prebrojivo mnogo prebrojivih skupova. □

Ono što nas zanima su totalne brojevne funkcije koje možemo izračunati u polinomnom vremenu na BRAM-stroju. Kako bismo odredili koje su to funkcije, moramo prvo definirati pojam polinomne izračunljivosti na BRAM-stroju.

## 1.1 Polinomna izračunljivost na BRAM-stroju

**Definicija 1.19:** Neka je $bLen : \mathbb{N} \to \mathbb{N}$ funkcija definirana sa:

$$bLen(n) = \begin{cases} \text{duljina binarnog zapisa od } n, & n > 0 \\ 0, & n = 0 \end{cases}$$

◁

**Napomena 1.20:** Funkciju *bLen* koristit ćemo za izračun duljine ulaza u BRAM-stroju. Smatramo da broj 0 nema duljinu, tj. nećemo mu dati istu duljinu kao i broju 1 (iako im je ista duljina binarnog zapisa), jer je na početku u svim registrima zapisan broj 0,

osim u konačno mnogo njih, a takvim registrima ne bismo htjeli dati značaj pri računanju vremenske složenosti programa, što ćemo naknadno definirati. ◁

**Definicija 1.21:** Neka je $\mathcal{S}$ BRAM-stroj. Za konfiguraciju $c$ BRAM-stroja $\mathcal{S}$ definiramo *duljinu konfiguracije* kao $bLen(c) = \sum_{i\in\mathbb{N}} bLen(c(\mathcal{R}_i))$. ◁

**Napomena 1.22:** *bLen* funkcija svaku konfiguraciju $c$ svakog BRAM-stroja $\mathcal{S}$ preslikava u $\mathbb{N}$ jer po definiciji 1.5 za preslikavanje $c$ je skup $c^{-1}[\mathbb{N}_+]$ konačan, tj. skoro svuda je nula, pa je suma $\sum_{i\in\mathbb{N}} bLen(c(\mathcal{R}_i))$ konačna. ◁

**Definicija 1.23:** Neka je $\mathcal{S}$ BRAM-stroj. Bilo koji prijelaz između konfiguracija zvat ćemo *korak*. ◁

**Korolar 1.24:** Neka je $\mathcal{S}$ BRAM-stroj. Neka je $(c_n)_{n\in\mathbb{N}}$ proizvoljan niz konfiguracija BRAM-stroja $\mathcal{S}$ takvih da je $c_i \rightsquigarrow c_{i+1}$ za svaki $i \in \mathbb{N}$. Broj koraka do konfiguracije $c_m$ je jednak $m$, za svaki $m \in \mathbb{N}$.

*Dokaz.* Neka je $m \in \mathbb{N}$ proizvoljan. Skup $S_m = \{c_t \rightsquigarrow c_{t+1} \mid t < m\}$ predstavlja skup prijelaza do konfiguracije $c_m$ te je očito kardinalnost tog skupa jednaka $m$. Dakle, slijedi da je broj koraka do konfiguracije $c_m$ jednak $m$, a budući da je $m \in \mathbb{N}$ bio proizvoljan, tvrdnja vrijedi za svaki $m \in \mathbb{N}$. □

**Definicija 1.25:** Neka je $Q$ binmakro-program. Kažemo da $P$ djeluje na registar $\mathcal{R}_i, i \in \mathbb{N}$ ako bar jedna od BRAM-instrukcija u $P$ djeluje na $\mathcal{R}_i$. ◁

**Definicija 1.26:** Neka je $P$ totalan BRAM-program.
*Vremenska složenost* BRAM-programa $P$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj koraka do prve završne konfiguracije u $P$-izračunavanju s početnom konfiguracijom $c$ duljine $n$, za svaku početnu konfiguraciju $c$ (stroja s programom $P$). ◁

**Napomena 1.27:** Definicija je dobra, jer za bilo koji totalan BRAM-program $P$ iz pravila 2–9 iz definicije 1.5 slijedi da će čitavo izračunavanje s bilo kojom konfiguracijom ostati isto ako je sadržaj registara na koje djeluje $P$ jednak. budući da je broj registara na koje $P$ djeluje konačan slijedi da je za svaki $n \in \mathbb{N}$ skup

$$\begin{aligned} S_{conf_n} = \{n_{0,c} \mid\ & (c_{n_c})_{n\in\mathbb{N}} \text{ je } P\text{-izračunavanje s } c,\\ & n_{0,c} \text{ je indeks završne konfiguracije u } (c_{n_c})_{n\in\mathbb{N}},\\ & c \text{ je proizvoljna početna konfiguracija (stroja s totalnim programom } P),\\ & bLen(c)) = n\} \subseteq \mathbb{N} \end{aligned}$$

konačan, totalno uređen, neprazan (jer postoji bar jedna početna konfiguracija stroja s programom $P$ duljine $n$, primjerice konfiguracija $(0, 2^{n-1}, 0, 0, \ldots, 0)$), pa postoji maksimalan član tog skupa i po korolaru 1.24 i definiciji 1.26 slijedi da je to $f(n)$. ◁

Za totalne brojevne funkcije nam neće biti dovoljna vremenska složenost BRAM-programa, tj. trebat će nam definicija vremenske složenosti za BRAM-algoritam, a prvo trebamo uvesti pojam *duljine* za ulaz.

**Definicija 1.28:** Neka je $k \in \mathbb{N}_+$. Za ulaz $\vec{x} \in \mathbb{N}^k$ oblika $\vec{x} = (x_1, x_2, \ldots, x_k)$ definiramo *duljinu ulaza* kao $bLen(\vec{x}) = \sum_{i=1}^{k} bLen(x_i) = n$. ◁

**Definicija 1.29:** Neka je $k \in \mathbb{N}$ te $P^k$ totalan BRAM-algoritam za koji $P$-izračunavanje stane za svaki $\vec{x} \in \mathbb{N}^k$. Vremenska složenost od $P^k$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj koraka do prve završne konfiguracije u $P$-izračunavanju s $\vec{x}$ za svaki $\vec{x} \in \mathbb{N}^k$ takav da je $bLen(\vec{x}) = n$. ◁

**Napomena 1.30:** Definicija je dobra jer je za svaki $k \in \mathbb{N}_+$ i za svaki $n \in \mathbb{N}$ skup

$$\begin{aligned} S_{input_n} = \{n_{0,\vec{x}} \mid\ & (c_{n_{\vec{x}}})_{n\in\mathbb{N}} \text{ je } P\text{-izračunavanje s } \vec{x},\\ & n_{0,\vec{x}} \text{ je indeks završne konfiguracije u } (c_{n_{\vec{x}}})_{n\in\mathbb{N}},\\ & \vec{x} \in \mathbb{N}^k,\\ & bLen(\vec{x})) = n\} \subseteq \mathbb{N}, \end{aligned}$$

totalno uređen, konačan i neprazan (jer za svaki $n \in \mathbb{N}$ postoji bar jedan $\vec{x} \in \mathbb{N}^k$ duljine $n$, primjerice $(2^{n-1}, 0, 0, \ldots, 0) \in \mathbb{N}^k$) skup pa postoji maksimalan element skupa $S_n$ i po korolaru 1.24 i definiciji 1.29 slijedi da je to $f(n)$. ◁

**Definicija 1.31:** Neka su $f$ i $g$ funkcije $f, g\colon \mathbb{N} \to \mathbb{N}$. Kažemo da je $f(n) = O(g(n))$ ako postoje $c, m_0 \in \mathbb{N}$ takvi da za svaki $m \in \mathbb{N}, n \geq m_0$ vrijedi

$$f(n) \leq c \cdot g(n).$$

Kada je $f(n) = O(g(n))$, kažemo da je $g(n)$ *gornja granica* za $f(n)$. ◁

**Korolar 1.32:** Neka je $P$ totalan BRAM-program i $g : \mathbb{N} \to \mathbb{N}$ proizvoljna funkcija. Ako je $P$ vremenske složenosti $O(g(n))$, tada je i BRAM-algoritam $P^k$ vremenske složenosti $O(g(n))$ za svaki $k \in \mathbb{N}_+$.

*Dokaz.* Neka je $k \in \mathbb{N}$ proizvoljan. Za početnu konfiguraciju $c$ s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$ iz definicije vidimo da vrijedi

$$bLen(\vec{x}) = \sum_{i=1}^{k} bLen(x_i) = \sum_{i=1}^{k} bLen(c(\mathcal{R}_i)) = \sum_{i\in\mathbb{N}} bLen(c(\mathcal{R}_i)) = bLen(c).$$

Iz napomena 1.27 i 1.30 slijedi da je $S_{input_n} \subseteq S_{conf_n}$ za svaki $n \in \mathbb{N}$. Iz toga je posebno $\max(S_{input_n}) \leq \max(S_{conf_n})$ za svaki $n \in \mathbb{N}$, iz čega slijedi tvrdnja u korolaru. □

**Definicija 1.33:** Neka je $v\colon \mathbb{N} \to \mathbb{N}$ proizvoljna funkcija. Definiramo *klasu vremenske složenosti*, TIME($v(n)$), da bude skup svih totalnih brojevnih funkcija za koje postoji BRAM-algoritmi vremenske složenosti $O(v(n))$ koji ih računaju. Za takve funkcije također kažemo da su BRAM-izračunljive u vremenu $O(v(n))$. ◁

**Lema 1.34:** Označimo s BP skup totalnih brojevnih funkcija koje su BRAM-izračunljive u polinomnom vremenu. Vrijedi

$$\mathrm{BP} = \bigcup_{t\in\mathbb{N}} \mathrm{TIME}(n^t).$$

Za takve funkcije također kažemo da su *polinomno* na BRAM-izračunljive.

*Dokaz.* Dokažimo:

- $\bigcup_{t\in\mathbb{N}} \mathrm{TIME}(n^t) \subseteq \mathrm{BP}$
  Neka je $f^k : \mathbb{N}^k \to \mathbb{N}$ za neki $k \in \mathbb{N}$, takav da $f^k \in \bigcup_{t\in\mathbb{N}} \mathrm{TIME}(n^t)$. Tada postoji $t \in \mathbb{N}$ takav da $f^k \in \mathrm{TIME}(n^t)$. Budući da je $n^t$ polinom, slijedi da je $f^k \in \mathrm{BP}$.

- $\mathrm{BP} \subseteq \bigcup_{t\in\mathbb{N}} \mathrm{TIME}(n^t)$
  Neka je $f^k : \mathbb{N}^k \to \mathbb{N}$ za neki $k \in \mathbb{N}$, takav da $f^k \in \mathrm{BP}$. Tada postoje $m \in \mathbb{N}, (a_1, a_2, \ldots, a_m) \in \mathbb{Z}^m$ takav da za polinom $g : \mathbb{N} \to \mathbb{N}$, $g(n) = a_m n^m + \ldots + a_1 n + a_0$, vrijedi $f^k \in TIME(g(n))$. Neka je $c = \max\{|a_i|;\ i \leq m\}$. Tada je za svaki $n \in \mathbb{N}$, $n \geq 1$
  $$\sum_{i=0}^{m} a_i n^i \leq \sum_{i=0}^{m} c \cdot n^i \leq \sum_{i=0}^{m} c \cdot n^{m+1} = c \cdot (m+1) \cdot n^{m+1}.$$
  Iz ovoga slijedi da je $g(n) = O(n^{m+1})$, što povlači da je pripadni BRAM-algoritam vremenske složenosti $O(n^{m+1})$, odnosno $f^k \in \mathrm{TIME}(n^{m+1})$, iz čega slijedi da je $f^k \in \bigcup_{t\in\mathbb{N}} \mathrm{TIME}(n^t)$.

□

Nakon što smo definirali pojam BRAM-izračunljivosti u polinomnom vremenu za brojevne funkcije, možemo početi određivati takve funkcije. Uzmimo za primjer familiju konstantnih funkcija svih mjesnosti, tj. $C = \{\mathsf{C}^k_m : \mathbb{N}^k \to \mathbb{N};\ \mathsf{C}^k_m(\vec{x}) = m \mid k \in \mathbb{N}_+, m \in \mathbb{N}\}$.

**Primjer 1.35:** Neka su $k \in N_+$ i $m \in \mathbb{N}$ proizvoljni te neka je $b = bLen(m)$. Zapišimo $m$ u binarnom zapisu $m = (m_{b-1} \cdots m_1 m_0)_2$. Zatim definirajmo funkciju $I : \{0,1\} \to \{\textsc{shl}\ \mathcal{R}_0, \textsc{shs}\ \mathcal{R}_0\}$ sa $I(0) = \textsc{shl}\ \mathcal{R}_0$ i $I(1) = \textsc{shs}\ \mathcal{R}_0$. Funkcija $I$ će nam poslužiti za definiciju BRAM-algoritma:

$$P^k_{\mathsf{C}_m} := \begin{bmatrix} 0.\ I(m_b) \\ \vdots \\ (b-2).\ I(m_1) \\ (b-1).\ I(m_0) \end{bmatrix}^k$$

S obzirom na prirodu funkcije $I$ lako se vidi da će nakon svakog $P^k_{\mathsf{C}_m}$-izračunavanja s $\vec{x} \in \mathbb{N}^k$ vrijednost registra $\mathcal{R}_0$ biti $(m_{b-1} \cdots m_1 m_0)_2$, tj. da $P^k_{\mathsf{C}_m}$ računa totalnu brojevnu funkciju $\mathsf{C}^k_m$. Budući da za svaki $\vec{x} \in \mathbb{N}^k$ broj koraka iznosi $b$, $P^k_{\mathsf{C}_m}$ je vremenske složenosti $O(1)$, što povlači da je $\mathsf{C}^k_m \in \mathrm{TIME}(1) \subseteq \mathrm{BP}$. Budući da su $k \in \mathbb{N}_+$ i $m \in \mathbb{N}$ bili proizvoljni, slijedi $C \subseteq \mathrm{BP}$. ◁

Pokazali smo za svaku konstantnu funkciju da je BRAM-izračunljiva u polinomnom vremenu, ali to su trivijalne funkcije koje nisu glavni predmet ovog rada. Za druge funkcije poput zbrajanja i množenja (svih mjesnosti), radi praktičnosti će nam trebati binmakro-stroj.

# Poglavlje 2

# Binmakro-izračunljivost

Definicije za binmakro-stroj kod BRAM-stroja bit će analogne onima kod RAM-stroja u [1], te ćemo i izvući iste zaključke koje će nam omogućiti korištenje binmakro-izračunljivosti u određivanju totalnih brojevnih funkcija koje su BRAM-izračunljive u polinomnom vremenu. Ubuduće ćemo koristiti binmakro-stroj kao skraćenicu za binmakro-stroj nakon njegove definicije

**Definicija 2.1:** *Binmakro-stroj* je matematički stroj koji sadrži:

- *binmakro-program*: fiksni konačni niz *binmakro-instrukcija* $Q := (I_0, I_1, \ldots, I_{n-1})$, svaka od kojih je jednog od dva oblika:
  - obična BRAM-instrukcija (tipa SHL, SHL, SHS, BZ, ODD ili GO TO), ili
  - *binmakro* oblika $P^*$, gdje je $P$ BRAM-program;

  Također, ako je binmakro-instrukcija oblika BZ, ODD ili GO TO, te je dio nekog binmakro-programa $Q$, onda njeno odredište mora element od $[0..n_Q]$
- registre $(\mathcal{R}_j)_{j\in\mathbb{N}}$, iste kao i kod BRAM-stroja;
- programski brojač PC, isti kao i kod BRAM-stroja;
- *pomoćni programski brojač* AC, čije moguće vrijednosti *ac* ovise o binmakro-instrukciji koja se trenutno izvršava ($I_{pc}$, gdje je *pc* vrijednost od PC):
  - ako je $I_{pc} = P^*$ za BRAM-program $P$, tada je $ac \in [0..n_P]$;
  - inače ($pc = n_Q$, ili $I_{pc} \in \mathfrak{Ins}$), $ac = 0$. ◁

**Korolar 2.2:** Skup $\mathcal{M}\mathfrak{Ins}$, svih binmakro-instrukcija je prebrojiv.

*Dokaz.* Po definiciji 2.1 svaka binmakro-instrukcija može biti BRAM-instrukcija ili binmakro oblika $P^*$, gdje je $P$ BRAM-program, slijedi da je

$$\mathcal{MI}ns = \mathcal{I}ns \cup \{P^* \mid P \in \mathcal{P}rog\}.$$

Po lemi 1.3 imamo da je $\mathcal{I}ns$ prebrojiv skup, a po korolaru 1.4 imamo da je skup $\mathcal{P}rog$ prebrojiv pa, budući da skup $\{P^* \mid P \in \mathcal{P}rog\}$ ne može biti veće kardinalnosti od $\mathcal{P}rog$, slijedi da je skup $\{P^* \mid P \in \mathcal{P}rog\}$ najviše prebrojiv. Iz toga slijedi da je skup $\mathcal{MI}ns$, kao unija prebrojivog i najviše prebrojivog skupa, prebrojiv skup. □

**Korolar 2.3:** Skup $\mathcal{MP}rog$ svih binmakro-programa je prebrojiv.

*Dokaz.* Skup $\mathcal{MP}rog = \{Q \in \mathcal{MI}ns^* \mid$ sva odredišta u $Q$ su manja ili jednaka $n_Q\}$ je svakako najviše prebrojiv, jer je skup $\mathcal{MI}ns^*$ prebrojiv (jer je skup $\mathcal{MI}ns$ prebrojiv), ali je također skup $\mathcal{I}ns^*_{\text{SHR}} \subseteq \mathcal{MP}rog$ prebrojiv, tako da iz toga slijedi da je $\mathcal{MP}rog$ prebrojiv. □

**Definicija 2.4:** *Konfiguracija binmakro-stroja* s programom $Q = (I_0, I_1, \ldots, I_{n_Q-1})$, registrima $\mathcal{R}_j$, $j \in \mathbb{N}$ te programskim brojačima PC i AC je bilo koje preslikavanje $c : \{\mathcal{R}_j \mid j \in \mathbb{N}\} \cup \{\text{PC}, \text{AC}\} \to \mathbb{N}$, takvo da je $c^{-1}[\mathbb{N}_+]$ konačan skup, $c(\text{PC}) \leq n_Q$, i još je $c(\text{AC}) = 0$ — osim u slučaju $I_{c(\text{PC})} = P^*$, kada je $c(\text{AC}) \leq n_P$. Skraćeno pišemo $c = (c(\mathcal{R}_0), c(\mathcal{R}_1), \ldots, c(\text{PC}), c(\text{AC}))$.

*Početna binmakro-konfiguracija* je binmakro-konfiguracija $c$ za koju je $c(\text{AC}) = 0$ i $c(\text{PC}) = 0$. Početna binmakro-konfiguracija s ulazom $\vec{x}$ definira se jednako kao i početna BRAM-konfiguracija: svuda osim na ulaznim registrima je 0, pa tako i na AC. Također, završna binmakro-konfiguracija definira se jednako kao i u BRAM-slučaju: uvjetom $c(\text{PC}) = n_Q$ (tada mora biti $c(\text{AC}) = 0$, jer $I_{c(\text{PC})}$ uopće ne postoji). ◁

**Definicija 2.5:** Za konfiguracije $c = (r_0, r_1, \ldots, pc, ac)$ i $d = (r'_0, r'_1, \ldots, pc', ac')$ istog binmakro-stroja s binmakro-programom $Q = (I_0, I_1, \ldots, I_{n_Q-1})$, kažemo da *c prelazi* u *d* (*po programu Q*), i pišemo $c \rightsquigarrow d$, ako vrijedi jedno od sljedećeg:

1. $c = d$, i $c$ je završna konfiguracija ($pc = n_Q$) — još pišemo $c \circlearrowleft$;
2. $ac = ac' = 0$, $I_{pc}$ je BRAM-instrukcija, a BRAM-konfiguracija $(r_0, r_1, \ldots, pc)$ nije završna ($pc < n_Q$) i prelazi u BRAM-konfiguraciju $(r'_0, r'_1, \ldots, pc')$ po programu $Q$ (odnosno njegovoj instrukciji s rednim brojem $pc$);
3. $pc' = pc$, $I_{pc}$ je binmakro $P^*$, a BRAM-konfiguracija $(r_0, r_1, \ldots, ac)$ nije završna ($ac < n_P$) i prelazi u BRAM-konfiguraciju $(r'_0, r'_1, \ldots, ac')$ po programu $P$ (odnosno njegovoj instrukciji s rednim brojem $ac$);
4. $pc' = pc + 1$, $I_{pc} = P^*$, BRAM-konfiguracija $(r_0, r_1, \ldots, ac)$ je završna ($ac = n_P$) i $ac' = 0$. ◁

**Korolar 2.6:** Svaka konfiguracija prelazi u neku, jedinstvenu, konfiguraciju.

*Dokaz.* Neka je $c$ proizvoljna binmakro-konfiguracija stroja s programom $Q$. Vidimo da pravila 1–4 obuhvaćaju sve slučajeve i pomoću leme 1.6 vidimo za svaki od tih slučajeva postoji jedinstvena binmakro-konfiguracija $d$ za koju je $c \rightsquigarrow d$. □

**Definicija 2.7:** Neka je $Q$ binmakro-program i $c$ početna konfiguracija (stroja s programom $Q$). *P-izračunavanje s c* je niz konfiguracija $(c_n)_{n\in\mathbb{N}}$, takvih da je $c_0 = c$ te za svaki $n \in \mathbb{N}$, $c_n$ prelazi u $c_{n+1}$. Kažemo da to izračunavanje *stane* ako postoji $n_0 \in \mathbb{N}$ takav da je $c_{n_0}$ završna. Također, kažemo da je $Q$ *totalan* ako $Q$-izračunavanje s $c$ stane za svaku početnu konfiguraciju $c$. ◁

**Propozicija 2.8:** Za svaki binmakro-program $Q$, za svaku početnu konfiguraciju $c$ (stroja s programom $Q$), postoji jedinstveno $Q$-izračunavanje s $c$.

*Dokaz.* Za postojanje, induktivno definiramo

$$c_0 \coloneqq c, \tag{2.1}$$

$$c_{n+1} \coloneqq \text{jedinstvena konfiguracija u koju } c_n \text{ prelazi (prema korolaru 2.6).} \tag{2.2}$$

Po Dedekindovom teoremu rekurzije, time je dobro definiran niz, i taj niz je po definiciji $Q$-izračunavanje s $c$.

Za jedinstvenost, pretpostavimo da postoje dva $Q$-izračunavanja s $c$, $(c_i)_{i\in\mathbb{N}}$ i $(c'_i)_{i\in\mathbb{N}}$. Kako je $c \neq c'$, postoji neki $i \in \mathbb{N}$ takav da je $c_i \neq c'_i$, a zbog dobre uređenosti od $\mathbb{N}$ postoji najmanji takav: označimo ga s $i_0$. Taj $i_0$ nije 0, jer je $c_0 = c$ = početna konfiguracija. Dakle, konfiguracija $c_{i_0-1} = c'_{i_0-1}$ prelazi u dvije različite konfiguracije $c_{i_0}$ i $c'_{i_0}$, što je kontradikcija s korolarom 2.6. □

**Propozicija 2.9:** Za svaki binmakro-program $Q$, za svaku početnu konfiguraciju $c$ (stroja s programom $Q$) za koje $Q$-izračunavanje s $c$ stane, postoji jedinstvena završna konfiguracija.

*Dokaz.* Pretpostavimo da je $(c_i)_{i\in\mathbb{N}}$ $Q$-izračunavanje s $c$ u kojem postoje dvije završne konfiguracije, i označimo s $i_1$ i $i_2$ indekse na kojima se one prvi put pojavljuju. Bez smanjenja općenitosti (različitost je simetrična) možemo pretpostaviti $i_1 < i_2$. No, jer je $c_{i_1}$ završna, ona prelazi (samo) u samu sebe, pa indukcijom imamo

$$c_{i_1} = c_{i_1+1} = c_{i_1+2} = \cdots = c_{i_2}, \tag{2.3}$$

što je kontradikcija. □

**Korolar 2.10:** Skup $\mathcal{MTP}rog$ svih totalnih binmakro-programa je prebrojiv.

*Dokaz.* Lako se vidi da je za svaki $k \in \mathbb{N}$, $\mathcal{I}ns^k_{\text{SHR}}$ totalan binmakro-program, jer za svaku početnu konfiguraciju $c$ (stroja s programom $\mathcal{I}ns^k_{\text{SHR}}$) u $k$ koraka dođemo do završne konfiguracije, odnosno $Ins^k_{\text{SHR}}$-izračunavanje stane u $k$ koraka. Sad tvrdnja korolara slijedi direktno iz činjenice da je ${\mathcal{I}ns_{\text{SHR}}}^* \subseteq \mathcal{MTP}rog \subseteq \mathcal{MP}rog$, i korolara 2.3. Iz teorije skupova znamo da je skup $A^*$ prebrojiv ako je $A$ prebrojiv, te tvrdnja u korolaru slijedi iz toga da $\mathcal{MTP}rog$ mora biti prebrojiv ako ima prebrojiv podskup i nadskup. □

Navest ćemo sada definicije *binmakro-algoritma*, *binmakro-izračunavanja s* $\vec{x} \in N^k$, $k \in \mathbb{N}_+$ i definiciju *binmakro-izračunljive* funkcije. Koristit ćemo se tim izrazima ubuduće dok budemo na analogan način kao u [1], dokazivali *ekvivalentnost* BRAM-izračunavanja i binmakro-izračunavanja, tj. da se svaki binmakro-stroj može simulirati BRAM-strojem. Razlog zašto nam ipak treba binmakro-stroj je taj što se mnogi programi jednostavnije pišu kao binmakro-programi, gdje imamo veći izbor „instrukcija" nego kod BRAM-programa, te nakon što uvedemo pojam *spljoštenja*, vidjet ćemo da kao instrukciju u binmakro-programu možemo efektivno koristiti bilo koji binmakro-program.

**Definicija 2.11:** *Binmakro-algoritam* je uređen par binmakro-programa $Q$ i mjesnosti $k \in \mathbb{N}_+$. Umjesto $(Q, k)$ pišemo $Q^k$.

Neka je $Q^k$ binmakro-algoritam te $\vec{x} \in \mathbb{N}^k$. *$Q$-izračunavanje s $\vec{x}$* je niz konfiguracija $(c_n)_{n\in\mathbb{N}}$, takvih da je $c_0$ početna konfiguracija (binmakro-stroja s programom $Q$) s ulazom $\vec{x}$ te, za svaki $n$, $c_n$ prelazi u $c_{n+1}$. Kažemo da to izračunavanje *stane* ako postoji $n_0 \in \mathbb{N}$ takav da je $c_{n_0}$ završna.

Neka je $Q^k$ totalan binmakro-algoritam te $f^k$ totalna brojevna funkcija iste mjesnosti. Kažemo da $Q^k$ *računa* funkciju $f$ ako za sve $\vec{x} \in \mathbb{N}^k$ $Q$-izračunavanje s $\vec{x}$ stane u konfiguraciji oblika $(f(\vec{x}), \ldots, 0, n_Q)$ ◁

**Korolar 2.12:** Za svaki binmakro-algoritam $Q^k$, za svaki ulaz $\vec{x} \in \mathbb{N}^k$, postoji jedinstveno $P$-izračunavanje s $\vec{x}$.
Za svaki binmakro-program $Q$, za svaki $\vec{x} \in \mathbb{N}^+$ za koje $Q$-izračunavanje s $\vec{x}$ stane, postoji jedinstvena završna konfiguracija.

*Dokaz.* Analognim dokazom kao i u korolarima 1.12 i 1.13, koristeći definiciju 2.7 i propozicije 2.8 i 2.9, slijedi da tvrdnje u korolaru vrijede. □

**Definicija 2.13:** Neka je $k \in \mathbb{N}_+$ te $f^k$ totalna brojevna funkcija. Kažemo da je $f^k$ *binmakro-izračunljiva* ako postoji binmakro-algoritam $P^k$ koji je računa. ◁

**Korolar 2.14:** Neka je $Q$ binmakro-program.
Ako je $Q$ totalan, tada je binmakro-algoritam $Q^k$ totalan za svaki $k \in \mathbb{N}_+$.

*Dokaz.* Neka je $Q$ totalan i neka je $\vec{x} \in \mathbb{N}^+$ proizvoljan. Neka je $(c_n)_{n\in\mathbb{N}}$ $Q$-izračunavanje s $\vec{x}$. Po definiciji 1.7 i propoziciji 1.8 je to također $Q$-izračunavanje s $c_0$, jer je $c_0$ početna konfiguracija s ulazom $\vec{x}$ po definiciji. Budući da je $Q$ totalan, slijedi da $Q$-izračunavanje s $c_0$ stane, odnosno da postoji završna konfiguracija u $(c_n)_n$, iz čega slijedi da $Q$-izračunavanje s $\vec{x}$ stane. Kako je $\vec{x}$ bio proizvoljan, slijedi da $Q$-izračunavanje s $\vec{x}$ stane za svaki $\vec{x} \in \mathbb{N}^+$, odnosno da je binmakro-algoritam $Q^k$ totalan za svaki $k \in \mathbb{N}_+$. □

## 2.1 Vanjska vremenska složenost binmakro-programa

**Definicija 2.15:** Neka je $\mathcal{S}$ binmakro-stroj. Za konfiguraciju $c$ binmakro-stroja $\mathcal{S}$ definiramo *duljinu konfiguracije* kao $bLen(c) = \sum_{i\in\mathbb{N}} bLen(c(\mathcal{R}_i))$. ◁

**Napomena 2.16:** *bLen* svaku konfiguraciju $c$ svakog binmakro-stroja $\mathcal{S}$ preslikava u $\mathbb{N}$ jer po definiciji 2.4 za preslikavanje $c$ je skup $c^{-1}[\mathbb{N}_+]$ konačan, tj. skoro svuda je nula, pa je suma $\sum_{i\in\mathbb{N}} bLen(c(\mathcal{R}_i))$ konačna. ◁

**Definicija 2.17:** Neka je $\mathcal{S}$ binmakro-stroj. Bilo koji prijelaz između konfiguracija binmakro-stroja zvat ćemo *korak*. ◁

**Napomena 2.18:** Definicija je dobra, jer iako imamo definiran pojam koraka i za BRAM-stroj, znat će se na koji pojam mislimo jer će oni biti vezani za prijelaze između konfiguracija BRAM- ili binmakro-stroja. ◁

**Korolar 2.19:** Neka je $\mathcal{S}$ binmakro-stroj. Neka je $(c_n)_{n\in\mathbb{N}}$ proizvoljan niz konfiguracija binmakro-stroja $\mathcal{S}$ takvih da je $c_i \rightsquigarrow c_{i+1}$ za svaki $i \in \mathbb{N}$. Broj koraka do konfiguracije $c_m$ je jednak $m$, za svaki $m \in \mathbb{N}$.

*Dokaz.* Analognim dokazom kao u 1.24 slijedi tvrdnja u korolaru. □

**Definicija 2.20:** Neka je $P^*$ binmakro. Kažemo da $P^*$ djeluje na registar $\mathcal{R}_i, i \in \mathbb{N}$ ako na njega djeluje BRAM-program $P$.
Neka je $Q$ binmakro-program. Kažemo da $Q$ djeluje na registar $\mathcal{R}_i, i \in \mathbb{N}$ ako bar jedna od binmakro-instrukcija u $Q$ djeluje na $\mathcal{R}_i$. ◁

**Definicija 2.21:** Neka je $Q$ totalan binmakro-program.
*Vremenska složenost* binmakro-programa $Q$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj koraka do prve završne konfiguracije u $Q$-izračunavanju s početnom konfiguracijom $c$ duljine $n$, za svaku početnu konfiguraciju $c$ (stroja s programom $Q$). ◁

**Napomena 2.22:** Analognim razmišljanjem kao u napomeni 1.27 dolazimo do zaključka da je definicija dobra. ◁

**Korolar 2.23:** Neka je $P$ totalan BRAM-program i $g : \mathbb{N} \to \mathbb{N}$ proizvoljna funkcija. Ako je $P$ vremenske složenosti $O(g(n))$, tada je i BRAM-algoritam $P^k$ vremenske složenosti $O(g(n))$ za svaki $k \in \mathbb{N}_+$.

*Dokaz.* Analognim razmišljanjem kao u korolaru 1.32 dolazimo do zaključka da vrijedi tvrdnja u korolaru. □

**Definicija 2.24:** Neka je $k \in \mathbb{N}$ te $Q^k$ totalan binmakro-algoritam za koji $Q$-izračunavanje stane za svaki $\vec{x} \in \mathbb{N}^k$. Vremenska složenost od $Q^k$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj koraka do prve završne konfiguracije u $Q$-izračunavanju s $\vec{x}$ za svaki $\vec{x} \in \mathbb{N}^k$ takav da je $bLen(\vec{x}) = n$. ◁

**Napomena 2.25:** Analognim razmišljanjem kao u napomeni 1.30 dolazimo do zaključka da je definicija dobra. ◁

**Definicija 2.26:** Neka je $Q$ binmakro-program. Za početnu konfiguraciju $c$ (stroja s programom $Q$) definiramo *vanjsko $Q$-izračunavanje* s $c$ kao podniz $(c_j)_{j\in V}$ $Q$-izračunavanja s $c$, $(c_n)_{n\in\mathbb{N}}$, gdje je $V = \{j \in \mathbb{N} \mid c_j(\text{AC}) = 0\}$.
Kažemo da vanjsko $Q$-izračunavanje s početnom konfiguracijom $c$ (stroja s programom $Q$) stane ako postoji $m_0$ takav je $c_{m_0}$ završna konfiguracija i nalazi se u vanjskom $Q$-izračunavanju s početnom konfiguracijom $c$. ◁

**Korolar 2.27:** Neka je $Q$ binmakro-program i $c$ proizvoljna konfiguracija (stroja s programom $Q$). $Q$-izračunavanje s $c$ stane ako i samo ako vanjsko $Q$-izračunavanje $c$ stane.

*Dokaz.* Tvrdnja proizlazi iz definicije vanjskog $Q$-izračunavanja s $c$, jer ako postoji završna konfiguracija u $Q$-izračunavanju s $c$, ona će se naći u vanjskom $Q$-izračunavanju s $c$, jer za nju joj je vrijednost od AC jednaka nula. U suprotnom, ako završna konfiguracija postoji u vanjskom $Q$-izračunavanju s $c$, znači da se mora nalaziti u $Q$-izračunavanju s $c$. □

**Definicija 2.28:** Za konfiguracije $c = (r_0, r_1, \ldots, pc, 0)$ i $d = (r'_0, r'_1, \ldots, pc', 0)$ istog binmakro-stroja s binmakro-programom $Q = (I_0, I_1, \ldots, I_{n_Q-1})$, kažemo da *$c$ vanjski prelazi* u $d$ (*po programu $Q$*), i pišemo $c \to d$, ako za neki $k \in \mathbb{N}$, $k > 1$ postoji konačan niz konfiguracija $(c_0, c_1, \cdots, c_k)$ takav da je $c_0 = c$, $c_k = d$ i $c_i \rightsquigarrow c_{i+1}$ za svaki $i < k$ i $c_i(\text{AC}) \neq 0$ za svaki $i \in [1..k\rangle$. ◁

**Definicija 2.29:** Neka je $\mathcal{S}$ binmakro-stroj. Bilo koji vanjski prijelaz između konfiguracija zvat ćemo *vanjski korak*. ◁

**Korolar 2.30:** Neka je $Q$ binmakro-program i $c$ početna konfiguracija (stroja s programom $P$). Neka je $(c_n)_{n\in\mathbb{N}}$ $Q$-izračunavanje s $c$ i neka je $(c_j)_{j\in V}$ vanjsko $Q$-izračunavanje s pripadajućim oznakama iz definicije 2.26. Za uzastopne članove $c_i, c_l$ u nizu $(c_j)_{j\in V}$, takvi da $i < l$, vrijedi $c_i \to c_l$.

*Dokaz.* Neka su $i, l \in V$, $i < l$, takvi da su $c_i, c_l$ uzastopni članovi u nizu $(c_j)_{j\in V}$. Budući da je $c_i, c_l \in (c_n)_{n\in\mathbb{N}}$, slijedi da je $(c_i, c_{i+1}, \ldots, c_l)$ konačan niz konfiguracija za koje je $c_t \rightsquigarrow c_{t+1}$ za $k \in [i, l\rangle$. Također, $\mathrm{v}c_t(\textsc{ac}) \neq 0$ za svaki $t \in \langle i..l\rangle$, jer bi u suprotnom vrijedilo $t \in V$ pa $c_i$ i $c_l$ ne bi bili uzastopni članovi niza $(c_j)_{j\in V}$. Slijedi $c_i \to c_l$. □

**Definicija 2.31:** Neka je $Q$ totalan binmakro-program.
*Vanjska vremenska složenost* totalnog binmakro-programa $Q$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj vanjskih koraka do prve završne konfiguracije u vanjskom $Q$-izračunavanju s početnom konfiguracijom $c$ duljine $n$, za svaku početnu konfiguraciju $c$ (stroja s programom $Q$). ◁

**Napomena 2.32:** Definicija vanjske vremenske složenosti za totalne binmakro-programe je dobra jer je za svaki binmakro-program $Q$ i za svaku početnu konfiguraciju $c$ (stroja s programom $Q$) vanjsko $Q$-izračunavanje izvedeno iz $Q$-izračunavanja, te je u svakom vanjskom $Q$-izračunavanju broj vanjskih koraka do završne konfiguracije manji ili jednak u odnosu na broj koraka u $Q$-izračunavanju do završne konfiguracije. ◁

**Definicija 2.33:** Neka je $Q^k$ totalan binmakro-algoritam.
*Vanjska vremenska složenost* totalnog binmakro-algoritma $Q^k$ je funkcija $f\colon \mathbb{N} \to \mathbb{N}$, gdje je $f(n)$ maksimalan broj vanjskih koraka do prve završne konfiguracije u vanjskom $Q$-izračunavanju s početnom konfiguracijom $c$ (stroja s programom $Q$) s ulazom $\vec{x}$, za svaki $\vec{x} \in \mathbb{N}^k$ duljine $n$. ◁

**Napomena 2.34:** Analognim razmišljanjem kao u napomeni 2.32 dolazimo do zaključka da je dobro definirana vanjska vremenska složenost za totalne binmakro-algoritme. ◁

**Definicija 2.35:** Neka je $P^*$ binmakro takav da je $P$ totalan BRAM-program. Kažemo da je $P^*$ *totalan* ako je $P$ totalan. Također, ako je $P^*$ totalan, tada definiramo vremensku složenost za $P^*$ da je jednaka vremenskoj složenosti od $P$. ◁

**Teorem 2.36:** Neka je $Q$ totalan binmakro-program s vanjske vremenske složenosti $O(n^t)$ za neki $t \in \mathbb{N}$, neka su svi binmakroi unutar programa $Q$ vremenske složenosti $O(n'^m)$ za neki $m \in \mathbb{N}$, te neka su registri tokom vanjskog $Q$-izračunavanja s proizvoljnom početnom konfiguracijom najviše veličine $O(n^l)$ za neki $l \in \mathbb{N}$. Tada je $Q$ polinomne vremenske složenosti.

*Dokaz.* Pogledajmo $Q$-izračunavanje s proizvoljnom početnom konfiguracijom $c$. Budući da je vanjska složenost od $Q$ jednaka $O(n^t)$, slijedi da se prilikom $Q$-izračunavanja binmakro-instrukcije izvršile najviše $O(n^t)$ puta, jer bi se inače u vanjskom-izračunavanju našlo više konfiguracija iz $Q$-izračunavanja zbog pravila 2 i 4 iz definicije 2.5 te zbog definicije 2.28. Iz toga slijedi da su se binmakroi u $Q$ izvršili najviše $O(n^t)$ puta, te im je ulaz bio duljine najviše $O(n^{l+1})$, iz čega slijedi da je vremenska složenost svakog binmakroa pri izvršavanju u $Q$ jednaka $O(n^{lm+m})$. Uzimajući u obzir koliko su se puta izvršili u $Q$, imamo da je sveukupna vremenska složenost svih binmakroa najviše $O(n^{t+lm+m})$. S pretpostavkom da binmakroa nema u $Q$ slijedi da je vremenska složenost od $Q$ jednaka vanjskoj vremenskoj složenosti od $Q$, odnosno $O(n^t)$. Kako je $O(n^t) = O(n^{t+lm+m})$, slijedi da je vremenska složenost od $Q$ jednaka $O(n^{t+lm+m})$, odnosno, polinomna je. □

**Teorem 2.37:** Neka je $Q^k$ totalan binmakro-algoritam vanjske vremenske složenosti $O(n^t)$ za neki $t \in \mathbb{N}$, neka su svi binmakroi unutar programa $Q$ vremenske složenosti $O(n'^m)$ za neki $m \in \mathbb{N}$, te neka su registri tokom vanjskog $Q$-izračunavanja za početnu konfiguraciju s proizvoljnim ulazom $x \in \mathbb{N}^k$ duljine $n$ najviše veličine $O(n^l)$ za neki $l \in \mathbb{N}$. Tada je $Q^k$ polinomne vremenske složenosti.

*Dokaz.* Analognim razmišljanjem kao u teoremu 2.36 dolazimo do zaključka da je $Q^k$ polinomne vremenske složenosti. □

## 2.2 Spljoštenje

Cilj nam je za naš binmakro-stroj opisati postupak spljoštenja, identičan postupku spljoštenja u [1], kako bismo dobili ekvivalentan BRAM-stroj koji bi prolazio kroz gotovo iste konfiguracije, gdje je razlika samo u tome što binmakro-stroj ima dva brojača u svojim konfiguracijama, za razliku od BRAM-stroja koji ima samo jedan.

**Definicija 2.38:** Neka je $Q$ binmakro-program. *Spljoštenje* od $Q$ je BRAM-program $Q^\flat$, dobiven iz $Q$ sljedećim postupkom:

Dok god postoji barem jedan binmakro u $Q$:

1. makni prvi binmakro iz $Q$: neka je to $i$. $P^*$;
2. u programu $Q$, svaki redni broj veći od $i$, i svako odredište veće od $i$, povećaj za $n_P - 1$ (tj. smanji za 1 ako je $P$ prazan program);
3. za svaku instrukciju programa $P$, dodaj u program $Q$ instrukciju istog tipa nad istim registrom (ako ga ima), kojoj su redni broj i odredište (ako ga ima) povećani za $i$. ◁

Dokaz iduće propozicije će biti isti kao u [1].

**Propozicija 2.39:** Za proizvoljni binmakro-program $Q$, postupak iz definicije 2.38 uvijek stane u konačno mnogo koraka, i pritom proizvede BRAM-program.

*Dokaz.* Pretpostavimo da postoji barem jedan binmakro u $Q$. Ako ne postoji, tada su sve instrukcije u binmakro-programu BRAM-instrukcije iz čega slijedi da je to BRAM-program. Neka je prvi takav binmakro $i.\ P^*$. U prvom koraku odmah uklanjamo binmakro $i.\ P^*$ iz $Q$, dok u drugom koraku ne mijenjamo broj binmakroa, eventualno njihove redne brojeve. U trećem koraku zapravo ne dodajemo nijedan binmakro, jer je za svaki binmakro $P^*$, $P$ BRAM-program, te su njegove instrukcije BRAM-instrukcije. Iz ovoga slijedi da se pri svakom prolazu kroz petlju broj binmakroa u $Q$ smanjuje za 1. Kako svaki binmakro-program ima konačno mnogo binmakroa, postupak će sigurno završiti (nakon najviše $n_Q$ prolaza kroz petlju). Kad završi, uvjet petlje neće biti ispunjen, dakle u $Q$ više neće biti binmakroa; drugim riječima, pretvorili smo $Q$ u BRAM-program. □

**Korolar 2.40:** Preslikavanje $^\flat$ je totalna surjekcija sa skupa $\mathcal{MP}rog$ na skup $\mathcal{P}rog$.

*Dokaz.* Surjektivnost slijedi iz činjenice da je $\mathcal{I}ns \subset \mathcal{MI}ns$ (dakle $\mathcal{P}rog \subset \mathcal{MP}rog$) te je $^\flat$ na BRAM-programima (tj. na binmakro-programima čije su instrukcije iz skupa $\mathcal{I}ns$) identiteta. Uvjet petlje već na početku nije ispunjen pa se program uopće ne mijenja. Dakle za svaki BRAM-program $P$ je $P^\flat = P$. □

**Definicija 2.41:** Za dva (binmakro- ili BRAM-) programa $P$ i $Q$ kažemo da su *ekvivalentni* ako za svaku mjesnost $k \in \mathbb{N}_+$, algoritmi $P^k$ i $Q^k$ računaju istu funkciju. ◁

**Teorem 2.42:** Za svaki binmakro-program $Q$, BRAM-program $Q^\flat$ je ekvivalentan s $Q$.

*Dokaz.* Neka je $Q$ proizvoljni binmakro-program. Prilikom spljoštenja binmakro-programa $Q$, definirat ćemo funkciju $v$ iz $\mathbb{N} \times \mathbb{N}$ u $\mathbb{N}$ sa

$$v(i,j) = \sum\nolimits_{t<i} n_t + j; \quad n_t = \begin{cases} 1, & I_t \in \mathcal{I}ns \\ n_P, & I_t = P^* \end{cases}, t < n_Q.$$

Ovom defincijom funkcije $v$ opisano je kako se točno transformiraju redni brojevi instrukcija pri spljoštenju binmakro-programa $Q$. Sad imamo da svaki prijelaz između BRAM-konfiguracija $(r_0, r_1, \ldots, v(pc, ac))$ i $(r'_0, r'_1, \ldots, v(pc', ac'))$ po programu $Q^\flat$ odgovara jednom ili više prijelaza između binmakro-konfiguracija $(r_0, r_1, \ldots, pc, ac)$ i $(r'_0, r'_1, \ldots, pc', ac')$ po programu $Q$. Iz toga slijedi da se pri izvršavanju programa i njegovog spljoštenja izvršavaju iste instrukcije, samo su im odredišta i redni brojevi transformirani po funkciji $v$. Promjene sadržaja registara koje te instrukcije proizvode su iste i odvijaju se na istim registrima, istim redom. To znači da ako počnemo od iste konfiguracije što se registara tiče, registri će mijenjati svoje vrijednosti na isti način prilikom izvršavanja $Q$ i $Q^\flat$, pa će posebno i sadržaj registra $\mathcal{R}_0$ biti isti. Štoviše, jer je $v(n_Q, 0) = n_{Q^\flat}$,

$Q^\flat$-izračunavanje s $\vec{x}$ će stati ako i samo ako $Q$-izračunavanje s $\vec{x}$ stane, i tada će u $\mathcal{R}_0$ biti isti broj. Posebno, oni za svaki $k \in \mathbb{N}_+$, $Q^k$ i $(Q^\flat)^k$ računaju istu funkciju, tj. $Q$ i $Q^\flat$ su ekvivalentni. □

**Korolar 2.43:** Za svaki totalni binmakro-program $Q$ vremenske složenosti $O(g(n))$ za neku funkciju $g : \mathbb{N} \to \mathbb{N}$, BRAM-program $Q^\flat$ je vremenske složenosti $O(g(n))$.

*Dokaz.* Slijedi iz teorema 2.42, definicije funkcije transformacije $v$ iz čega proizlazi da je broj koraka do završne konfiguracije manji u $Q^\flat$-izračunavanju nego u $Q$-izračunavanju s odgovarajućim konfiguracijama. Iz toga slijedi da je $Q^\flat$ vremenske složenosti $O(g(n))$ □

**Korolar 2.44:** Za svaki totalni binmakro-algoritam $Q^k$ vremenske složenosti $O(g(n))$ za neku funkciju $g : \mathbb{N} \to \mathbb{N}$, BRAM-program $Q^{\flat k}$ je vremenske složenosti $O(g(n))$.

*Dokaz.* Analognim razmišljanjem kao u korolaru 2.43 dolazimo do zaključka da je $Q^{\flat k}$ je vremenske složenosti $O(g(n))$. □

## 2.3 Primjeri binmakroa

Teorem 2.42 nam daje dvije važne posljedice. Prvu ćemo izraziti u obliku sljedećeg korolara, koji odgovara korolaru 1.28 iz [1].

**Korolar 2.45:** Neka je $k \in \mathbb{N}_+$ te $f^k$ totalna brojevna funkcija.
Tada je $f$ BRAM-izračunljiva ako i samo ako je binmakro-izračunljiva.

*Dokaz.* Za jedan smjer, ako je $f$ BRAM-izračunljiva, postoji BRAM-algoritam iste mjesnosti $P^k$ koji je računa. BRAM-program $P$ je i binmakro-program, a svejedno je na kojem će se stroju (BRAM- ili binmakro-) izvršavati jer je $P^\flat = P$. Drugim riječima, $P$ na binmakro-stroju također računa funkciju $f$, odnosno binmakro-algoritam $P^k$ računa $f$, pa je $f$ binmakro-izračunljiva.

Za drugi smjer, ako je $f$ binmakro-izračunljiva, postoji binmakro-algoritam $Q^k$ koji je računa. Po teoremu 2.42, $Q^\flat$ je ekvivalentan s $Q$, dakle za svaki $k$ (pa posebno i za mjesnost funkcije $f$), $Q^k$ i $(Q^\flat)^k$ (pišemo skraćeno $Q^{\flat k}$) računaju istu funkciju. Drugim riječima, BRAM-algoritam $Q^{\flat k}$ računa funkciju $f$, pa je ona BRAM-izračunljiva. □

**Napomena 2.46:** Druga posljedica teorema 2.42 je programska tehnika koja će bitno povećati izražajnost binmakro-programa koje pišemo. Rekli smo da je binmakro uvijek oblika $P^*$ gdje je $P$ BRAM-program, no zbog teorema 2.42 smijemo se ponašati kao da $P$ može biti i binmakro-program, koji koristi već napisane binmakroe. Formalno, pri tome mislimo na $(P^\flat)^*$, koji ima istu semantiku što se učinka na registre tiče. ◁

**Napomena 2.47:** Prije nego što krenemo računati vremensku složenost za pojedine binmakroe, definirat ćemo pojam *širine* BRAM-algoritma i binmakro-algoritma, analogno pojmu širine u [1]. Ovdje nam je trenutna motivacija za definiranje tog pojma drugačija, iz razloga što nas pri računanju vremenske složenosti nekog binmakroa interesiraju samo oni registri na koje binmakro utječe tijekom svog izvršavanja, jer oni uz ulaz utječu na konačan broj koraka, a kao što ćemo vidjeti, pojmom širine ćemo obuhvatiti sve takve registre.
Dakle, kako svaka BRAM-instrukcija djeluje na najviše jednom registru, čitav BRAM-program kao konačan niz instrukcija djeluje na konačno mnogo registara. To znači da za svaki BRAM-program $P$ postoji *širina* — najmanji broj $m_P \in \mathbb{N}$ takav da $P$ ne koristi nijedan registar $\mathcal{R}_i$ za $i \geq m_P$. Može biti i $m_P = 0$, ako program uopće ne koristi registre (ako je prazan, ili se sastoji samo od instrukcija tipa GO TO).
Za binmakro-program $Q$, možemo prirodno definirati $m_Q := m_{Q^\flat}$ — iako nam to zapravo neće trebati. Ali (RAM- i binmakro-) algoritmi $P^k$, pored registara koje koriste u instrukcijama, koriste i registre $\mathcal{R}_1$ do $\mathcal{R}_k$ za ulazne podatke. Moguće je da bude $m_P \leq k$, ako računamo funkciju koja ne ovisi o zadnjih nekoliko argumenata. Ipak, registar $\mathcal{R}_k$ jest bitan za postupak izračunavanja te funkcije na RAM-stroju jer, iako ga ne postavlja nijedna instrukcija, postavlja ga sam rad stroja koji u početnoj konfiguraciji u njega spremi argument $x_k$. Zato definiramo širinu algoritma kao $m_{P^k} := \max\{m_P, k+1\}$. Zbog $k \in \mathbb{N}_+$, uvijek je $m_{P^k} \geq 2$.
Također, iako ćemo računati vremensku složenost binmakroa, češće ćemo računati vremensku složenost binmakro-programa pomoću kojih su ti binmakro definirani, jer zbog korolara 2.43 će to biti dovoljno dokazati. ◁

**Napomena 2.48:** Ubuduće, kad budemo spominjali $r_i^{(j)}$, za $i, j \in \mathbb{N}$, prilikom analize algoritama (BRAM- ili binmakro-), to će nam označavati vrijednost registra $\mathcal{R}_i$ u trenutku kad je vrijednost programskog brojača $pc = j$. S napomenom da ćemo skraćeno koristiti $r_i$ umjesto $r_i^{(0)}$.
Također, prilikom dokazivanja semantike totalnog binmakro-programa za proizvoljnu početnu konfiguraciju, opisivat ćemo vanjsko izračunavanje vanjski korak po vanjski korak, te ćemo se služiti vanjskim prijelazima u svrhu toga. Navodit ćemo samo vrijednosti registara $r_i^{(j)}$ koji su promijenili svoju vrijednost nakon nekog vanjskog koraka, ili ako nema takvih navest ćemo promjenu vrijednosti programskog brojača $pc$.
Promjena vrijednosti programskog brojača će se uvijek dogoditi u izračunavanju binmakro-instrukcije totalnog binmakro-programa, jer u suprotnom slijedi da se izvršava jedna od BRAM-instrukcija s odredištem na istu tu instrukciju, zbog čega izračunavanje neće nikada stati pa onda taj binmakro-program nije totalan što je kontradikcija. Iz toga slijedi da je postupak s navođenjem stanja $pc$ pri svakom vanjskom prijelazu dobar za opis semantike nekog binmakro-programa. ◁

**Definicija 2.49:** Za $w \in \{0,1\}^*$ kažemo da je *binarna riječ*. S $\varepsilon$ ćemo označavati *praznu* binarnu riječ, odnosno binarnu riječ koja ne sadrži znamenke 0 i 1. ◁

Za početak ćemo definirati neke osnovne binmakroe koji će nam dobro doći pri stvaranju binmakro-programa koji će računati totalne brojevne funkcije.

**Definicija 2.50:** Za $i \in \mathbb{N}$ definirajmo binmakro

$$(\textsc{zero}\ \mathcal{R}_i) := \begin{bmatrix} 0.\ \textsc{bz}\ \mathcal{R}_i, 3 \\ 1.\ \textsc{shr}\ \mathcal{R}_i \\ 2.\ \textsc{go to}\ 0 \end{bmatrix}^*. \tag{2.4}$$

◁

**Propozicija 2.51:** Neka je $i \in \mathbb{N}$. Za proizvoljnu početnu konfiguraciju, binmakro (ZERO $\mathcal{R}_i$) ima semantiku $r_i' = 0$. Također, binmakro (ZERO $\mathcal{R}_i$) je vremenske složenosti $O(n)$.

*Dokaz.* Neka je $i \in \mathbb{N}$. Binmakro (ZERO $\mathcal{R}_i$) za početnu konfiguraciju u kojoj je $r_i = 0$, ima semantiku $r_i' = 0$, što je ujedno i završna konfiguracija zbog $pc = 3$ jer nema binmakroa unutar (ZERO $\mathcal{R}_i$) pa je uvijek $ac = 0$ (ubuduće ćemo smatrati da je dovoljno da je $pc = n_Q$, gdje je $Q$ binmakro, zbog dokazanih tvrdnji o spljoštenju u 2.42). Dalje, za početnu konfiguraciju u kojoj je $r_i \neq 0$ svaki prolaz kroz petlju, tj. čitanje instrukcija redom, ima semantiku $r_i \to r_i^{(1)} = \lfloor \frac{r_i}{2} \rfloor \to r_i = \lfloor \frac{r_i}{2} \rfloor$. Jer je $\lfloor \frac{x}{2} \rfloor < x$ za $x \in \mathbb{N}^+$, nakon konačno mnogo prolazaka kroz petlju dođemo do konfiguracije $r_i = 0$, a time dolazimo do prvog slučaja kada je sljedeća konfiguracija završna. Dakle, semantika ovog binmakroa za proizvoljan ulaz je $r_i' = 0$.
Za binmakro (ZERO $\mathcal{R}_i$), vidimo da je širina programa jednaka $i+1$ te da je jedini relevantan registar $\mathcal{R}_i$ pa možemo smatrati da je ulaz određen njegovom veličinom. Budući da cjelobrojno dijeljenje s 2 skrati duljinu zapisa vrijednosti registra $\mathcal{R}_i$ za 1 (osim u slučaju kad je vrijednost registra $\mathcal{R}_i$ jednaka 0, ali zbog 0. instrukcije, taj slučaj nemamo ovdje), a u spomenutoj petlji broj koraka je točno 2, imamo da je za ulaz duljine $n$ broj koraka iznosi točno $2n + 1$. Posebno, to vrijedi i za ulaz duljine 0, jer u tom slučaju imamo samo jedan korak. Sada imamo da je vremenska složenost binmakroa funkcija $v : \mathbb{N} \to \mathbb{N}$ zadana s $v(n) = 2n + 1$. Binmakro (ZERO $\mathcal{R}_i$) je vremenske složenosti $O(n)$. Također, vidimo da za svaki ulaz ovaj binmakro postavi vrijednost registra $\mathcal{R}_i$ na 0, tako da radi točno ono što se može iščitati iz njegovog naziva. □

**Napomena 2.52:** Ubuduće, kad budemo govorili da je neki binmakro vremenske složenosti $O(g(n))$, gdje je $g : \mathbb{N} \to \mathbb{N}$, zapravo ćemo misliti na njegovo spljoštenje, iz razloga što se jednak broj koraka izvede u programu na jednom i na drugom stroju. ◁

**Definicija 2.53:** Za različite $i, j \in \mathbb{N}$, definirajmo binmakro

$$(\text{BITMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j) \coloneqq \begin{bmatrix} 0.\ \text{ODD } \mathcal{R}_i, 4 \\ 1.\ \text{BZ } \mathcal{R}_j, 5 \\ 2.\ \text{SHL } \mathcal{R}_j \\ 3.\ \text{GO TO } 5 \\ 4.\ \text{SHS } \mathcal{R}_j \end{bmatrix}^{b*} \tag{2.5}$$

◁

**Propozicija 2.54:** Neka su $i, j \in \mathbb{N}$ različiti te neka su $w \in \{0,1\}^*$ i $b \in 0, 1$. Tada binmakro BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ za početnu konfiguraciju u kojoj je $b = r_i \bmod 2 \wedge r_j = (1w)_2$ ima semantiku $r'_j = (1wb)_2$.
Odnosno, binmakro (BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$) za početnu konfiguraciju u kojoj je $b = r_i \bmod 2 \wedge r_j \neq 0$, ima semantiku $r'_j = r_j \cdot 2 + b$. Dok za početnu konfiguraciju u kojoj je $r_i \bmod 2 = 1 \wedge r_j = 0$, binmakro BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ ima semantiku $r'_j = 1$.
Također, binmakro je BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ vremenske složenosti $O(1)$.

*Dokaz.* Neka su $i, j \in \mathbb{N}$ različiti te neka su $w \in \{0,1\}^*$. $(1w)_2$ nam zapravo označava sve binarne zapise prirodnih brojeva kojima je vodeća znamenka 1, a to uključuje sve brojeve osim nule. Za početnu konfiguraciju u kojoj je $r_i \bmod 2 = 1 \wedge r_j = (1w)_2$, čitanjem instrukcija možemo vidjeti da se dogodi sljedeće:

$$r_i \bmod 2 = 1 \wedge r_j = (1w)_2 \to r_j^{(4)} = r_j \cdot 2 + 1 \to r'_j = r_j \cdot 2 + 1.$$

Također, radi binarnog zapisa vrijednosti $r_j$ imamo

$$r'_j = r_j \cdot 2 + 1 \to r'_j = (1w)_2 \cdot 2 + 1 \to r'_j = (1w0)_2 + 1 \to r'_j = (1w1)_2.$$

Za početnu konfiguraciju u kojoj je $r_i \bmod 2 = 0 \wedge r_j = (1w)_2$, čitanjem instrukcija možemo vidjeti da se dogodi sljedeće:

$$r_i \bmod 2 = 1 \wedge r_j = (1w)_2 \to r_j^{(3)} = r_j \cdot 2 \to r'_j = r_j \cdot 2' + 0$$

Također, zbog binarnog zapisa vrijednosti $r_j$ imamo

$$r'_j = r_j \cdot 2 + 1 \to r'_j = (1w)_2 \cdot 2 \to r'_j = (1w0)_2.$$

Iz ovoga zaključujemo da za početnu konfiguraciju za koju je $b = r_i \bmod 2 \wedge r_j = (1w)_2$ ima semantiku $r'_j = (1wb)_2$, odnosno za početnu konfiguraciju u kojoj je $b = r_i \bmod 2 \wedge r_j \neq 0$, ima semantiku $r'_j = r_j \cdot 2 + b$.
Za početnu konfiguraciju u kojoj je $r_i \bmod 2 = 1 \wedge r_j = 0$, čitanjem instrukcija možemo vidjeti da ovaj binmakro ima semantiku

$$r_i \bmod 2 = 1 \wedge r_j = 0 \to r_j^{(4)} = 0 \cdot 2 + 1 = 1 \to r'_j = 1.$$

Posebno, u preostalom slučaju, tj. za početnu konfiguraciju u kojoj je $r_i \bmod 2 = 0 \wedge r_j = 0$, registri ne zabilježe promjenu, tj. s 0. instrukcije, prijeđe se na 1. binmakro-instrukciju, odakle se promijeni odredište na kraj programa, odnosno na završnu konfiguraciju.
Gledajući sva 4 slučaja, broj koraka je konstantan za svaki od slučaja pa slijedi da je binmakro BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ vremenske složenosti $O(1)$. □

**Definicija 2.55:** Za različite $i, j \in \mathbb{N}$ definirajmo binmakro

$$(\text{REVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_j) \coloneqq \left[\begin{array}{l} 0.\ \text{ZERO } \mathcal{R}_j \\ 1.\ \text{BZ } \mathcal{R}_i, 8 \\ 2.\ \text{SHS } \mathcal{R}_j \\ 3.\ \text{BITMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \\ 4.\ \text{SHR } \mathcal{R}_i \\ 5.\ \text{BZ } \mathcal{R}_i, 7 \\ 6.\ \text{GO TO } 3 \\ 7.\ \text{SHR } \mathcal{R}_j \end{array}\right]^{\flat *}. \tag{2.6}$$

◁

**Propozicija 2.56:** Neka su $i, j \in \mathbb{N}$ različiti te $w \in \{0,1\}^*$. Tada binmakro iz definicije 2.55 za početnu konfiguraciju u kojoj je $r_i = (1w)_2$ ima semantiku $r_i' = 0, r_j' = (1w^R)_2$, a za početnu konfiguraciju u kojoj je $r_i = 0$, binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) ima semantiku $r_j' = 0$.
Također, binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) vremenske složenosti $O(n)$.

*Dokaz.* Neka su $i, j \in \mathbb{N}$ različiti te $w \in \{0,1\}^*$. Za početak, vidimo da je (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) dobro definiran, jer su zadovoljeni uvjeti za sve binmakro-instrukcije unutar njega, posebno za (BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$), jer je $i$ i $j$ različiti. Za početnu konfiguraciju u kojoj je $r_i = 0$, praćenjem binmakro-instrukcija i pozivajući se na propoziciju 2.51, možemo vidjeti da binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) ima semantiku

$$r_i = 0 \to r_j^{(1)} = 0 \to r_j' = 0.$$

Za početnu konfiguraciju u kojoj je $r_i = (1w)_2$, odnosno $r_i \neq 0$, promotrimo vrijednost od $r_i$ u binarnom zapisu $(1b_{m-1} \cdots b_2 b_1)_2$, gdje je $m$ duljina binarnog zapisa od $r_i$. Praćenjem binmakro-instrukcija i pozivanjem na propozicije 2.51, za početnu konfiguraciju $r_i = (1b_{m-1} \cdots b_2 b_1)_2$ binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) ima semantiku

$$r_i = (1b_{m-1} \cdots b_2 b_1)_2 \to r_j^{(1)} = 0 \to r_j^{(3)} = 1.$$

Tu se događa grananje, odnosno, ako je $m = 1$, odnosno $w = \varepsilon$, dakle za $r_i = 1$, imamo da je po korolaru 2.53:

$$r_i^{(3)} = 1 \wedge r_j^{(3)} = 1 \to r_j^{(4)} = (11)_2 \to r_i^{(5)} = 0 \to r_j^{(8)} = 1 \to r_i' = 0 \wedge r_j' = 1.$$

Posebno je $r_i' = 0 \wedge r_j' = 1(w^R)_2$ (jer je $\varepsilon^R = \varepsilon$). U drugom slučaju, ako $m > 1$, odnosno postoji bar $b_1$ u zapisu $r_i$, imamo da vrijedi

$$r_i^{(4)} = (1b_{m-1}\cdots b_2b_1)_2 \wedge r_j^{(4)} = (1b_1)_2 \to r_i^{(5)} = (1b_{m-1}\cdots b_2)_2 \to r_i^{(3)} = (1b_{m-1}\cdots b_2)_2.$$

Tu dolazimo do petlje, u kojoj vidimo da smo iz $r_i^{(3)} = (1b_{m-1}\cdots b_2b_1)_2 \wedge r_i^{(3)} = 1$ došli u stanje registara $r_i^{(3)} = (1b_{m-1}\cdots b_2)_2 \wedge r_i^{(3)} = (1b_1)_2$. Iteriranim prolaskom kroz petlju i promatranjem slučaja u kojem je $r_i = 0$, lako se vidi da će semantika binmakroa u ovom slučaju biti $r_i = 0 \wedge r_j' = (1b_1b_2\cdots b_m)_2$, odnosno $r_i = 0 \wedge r_j' = (1w^R)_2$.
Promatrajući izvršavanje ovog binmakroa u slučaju $r_1 = (1b_{m-1}\cdots b_2b_1)_2$, vidimo da se petlja od 3. do 6. instrukcije izvrši $m-1$ puta, a po propoziciji 2.54 vidimo da su binmakro-instrukcije u petlji vremenske složenosti $O(1)$. Također, pomoću definicija 1.19 i 1.28 vidimo da je $m < bLen(r_i) \leq bLen(r_i) + bLen(r_j) = n$ pa zaključujemo da je taj dio programa vremenske složenosti $O(n)$. Uzimajući u obzir činjenicu da je konstantan broj koraka u ostatku izračunavanja, a po propoziciji 2.51 one su vremenske složenosti $O(n)$, te činjenicu da slučajevi $r_i = 1$ i $r_i = 0$ ne ulaze u petlju, dolazimo do zaključka da je binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) vremenske složenosti $O(n)$. □

**Definicija 2.57:** Za različite $i, j, k \in \mathbb{N}$ definirajmo binmakro

$$(\text{DREVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \text{ AND } \mathcal{R}_k) := \left[\begin{array}{l} 0.\ \text{ZERO } \mathcal{R}_j \\ 1.\ \text{ZERO } \mathcal{R}_k \\ 2.\ \text{BZ } \mathcal{R}_i, 12 \\ 3.\ \text{SHS } \mathcal{R}_j \\ 4.\ \text{SHS } \mathcal{R}_k \\ 5.\ \text{BITMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \\ 6.\ \text{BITMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_k \\ 7.\ \text{SHR } \mathcal{R}_i \\ 8.\ \text{BZ } \mathcal{R}_i, 10 \\ 9.\ \text{GO TO } 5 \\ 10.\ \text{SHR } \mathcal{R}_j \\ 11.\ \text{SHR } \mathcal{R}_k \end{array}\right]^{\flat *}. \tag{2.7}$$

◁

**Propozicija 2.58:** Neka su $i, j, k \in \mathbb{N}$ različiti te $w \in \{0,1\}^*$. Tada binmakro (DREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ AND $\mathcal{R}_k$) za početnu konfiguraciju u kojoj je $r_i = (1w)_2$ ima semantiku $r_i' = 0, r_j' = (1w^R)_2, r_k' = (1w^R)_2$, a za početnu konfiguraciju u kojoj je $r_i = 0$, binmakro (DREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ AND $\mathcal{R}_k$) ima semantiku $r_j' = 0, r_k' = 0$. Također, je binmakro (DREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ AND $\mathcal{R}_k$) vremenske složenosti $O(n)$.

*Dokaz.* Neka su $i, j, k \in \mathbb{N}$ različiti te $w \in \{0,1\}^*$. Za početak, vidimo da je binmakro (DREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ AND $\mathcal{R}_k$) dobro definiran, jer su zadovoljeni uvjeti za sve binmakro-instrukcije unutar njega, posebno za (BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$) i (BITMOVE $\mathcal{R}_i$ TO $\mathcal{R}_k$), jer je $i \neq j$

i $i \neq k$. Za početnu konfiguraciju u kojoj je $r_i = 0$, praćenjem binmakro-instrukcija i pozivajući se na propoziciju 2.51, možemo vidjeti da binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) ima semantiku

$$r_i = 0 \to r_j^{(1)} = 0 \to r_k^{(2)} = 0 \to r_j' = 0 \wedge r_k' = 0.$$

Za početnu konfiguraciju u kojoj je $r_i = (1w)_2$, odnosno $r_i \neq 0$, promotrimo vrijednost od $r_i$ u binarnom zapisu $(1b_{m-1} \cdots b_2 b_1)_2$, gdje je $m$ duljina binarnog zapisa od $r_i$. Praćenjem binmakro-instrukcija i pozivanjem na propozicije 2.51, za početnu konfiguraciju $r_i = (1b_{m-1} \cdots b_2 b_1)_2$ binmakro (REVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) ima semantiku

$$r_i = (1b_{n-1} \cdots b_2 b_1)_2 \to r_j^{(1)} = 0 \to r_k^{(2)} = 0 \to r_j^{(4)} = 1 \to r_k^{(5)} = 1.$$

Tu se događa grananje, odnosno, ako je $n = 1$, tj. $w = \varepsilon$, odnosno $r_i = 1$, imamo da je po korolaru 2.53

$$r_i^{(5)} = 1 \wedge r_k^{(5)} = 1 \to r_j^{(6)} = (11)_2 \to r_j^{(7)} = 1 \to r_i^{(8)} = 0,$$
$$r_i^{(8)} = 0 \to r_j^{(11)} = 1 \to r_k^{(12)} = 1 \to r_i' = 0 \wedge r_j' = 1 \wedge r_k' = 1.$$

Posebno je $r_i' = 0 \wedge r_j' = (1w^R)_2 \wedge r_k' = (1w^R)_2$, jer je $\varepsilon^R = \varepsilon$. U drugom slučaju, ako $m > 1$, odnosno bar postoji $b_1$ u izrazu od $r_i$, imamo da vrijedi

$$r_i^{(6)} = (1b_{n-1} \cdots b_2 b_1)_2 \wedge r_j^{(6)} = (1b_1)_2 \to r_k^{(7)} = (1b_1)_2 \to r_i^{(8)} = (1b_{n-1} \cdots b_2)_2,$$
$$r_i^{(8)} = (1b_{n-1} \cdots b_2)_2 \to r_i^{(5)} = (1b_{n-1} \cdots b_2)_2.$$

Tu dolazimo do petlje, u kojoj vidimo da se iz $r_i^{(5)} = (1b_{m-1} \cdots b_2 b_1)_2 \wedge r_j^{(5)} = 1 \wedge r_k^{(5)} = 1$ došli u stanje registara $r_i^{(5)} = (1b_{n-1} \cdots b_2)_2 \wedge r_i^{(5)} = (1b_1)_2 \wedge r_k^{(5)} = (1b_1)_2$. Ponavljanjem petlje i promatranjem slučaja u kojem je $r_i = 0$, lako se vidi da će semantika binmakroa u ovom slučaju biti $r_i = 0 \wedge r_j' = (1b_1 b_2 \cdots b_n)_2 \wedge r_j' = (1b_1 b_2 \cdots b_n)_2$, odnosno $r_i = 0 \wedge r_j' = (1w^R)_2 \wedge r_k' = (1w^R)_2$.
Promatrajući izvršavanje ovog binmakroa u slučaju $r_i = (1b_{m-1} \cdots b_2 b_1)_2$, vidimo da se petlja od 5. do 9. instrukcije izvrši $(m-1)$ puta, a po propoziciji 2.54 vidimo da su binmakro-instrukcije u petlji vremenske složenosti $O(1)$. Također, pomoću definicija 1.19 i 1.28 vidimo da je $m < bLen(r_i) \leq bLen(r_i) + bLen(r_j) + bLen(r_k) = n$ pa zaključujemo da je taj dio programa vremenske složenosti $O(n)$. Uzimajući u obzir činjenicu da je konstantan broj koraka u ostatku izračunavanja, a po propoziciji 2.51 svaka od njih je vremenske složenosti $O(n)$, te činjenicu što slučajevi $r_i = 1$ i $r_i = 0$ ne ulaze u petlju, dolazimo do zaključka da je binmakro (DREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ AND $\mathcal{R}_k$) vremenske složenosti $O(n)$.
Iz semantike ovog binmakroa, vidi se da on služi kao dvostruki REVERSE, te će nam kao takav dobro doći kasnije u kombinaciji s REVERSE binmakrom. □

Definirat ćemo u nastavku još dva specijalna tipa reverzibilnih binmakroa.

**Definicija 2.59:** Za različite $i, j \in \mathbb{N}$ definirajmo binmakro

$$(\text{ODDREVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_j) \coloneqq \left[\begin{array}{l} 0.\ \text{ODD } \mathcal{R}_i, 2 \\ 1.\ \text{GO TO } 7 \\ 2.\ \text{ZERO } \mathcal{R}_j \\ 3.\ \text{BITMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \\ 4.\ \text{SHR } \mathcal{R}_i \\ 5.\ \text{BZ } \mathcal{R}_i, 7 \\ 6.\ \text{GO TO } 3 \end{array}\right]^{\flat *}. \tag{2.8}$$

◁

**Propozicija 2.60:** Neka su $i, j \in \mathbb{N}$ različiti. Za početnu konfiguraciju u kojoj je $r_i = (w)_2$ neparan, binmakro ODDREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$ ima semantiku $r_i' = 0, r_j' = (w^R)_2$.

*Dokaz.* Neka su $i, j \in \mathbb{N}$ različiti. Formalno, binmakro (ODDREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) za konfiguraciju u kojoj je $r_i \bmod 2 = 0$, ne mijenja registre, odnosno jer je $r_i$ paran, nakon 0. binmakro-instrukcije u binmakrou se izvršava 1. binmakro-instrukcija koja radi prijelaz na završnu konfiguraciju, odnosno $pc' = 7$.
Neka je sad $w \in \{0,1\}^*$ takav da je $r_i = (w)_2$ i neka je $r_i \bmod 2 = 1$. Zapišimo $w$ u obliku $(b_m \cdots b_2 b_1)_2$, gdje je $m$ duljina binarnog zapisa od $r_i$. Tada praćenjem instrukcija i uz propoziciju 2.53 vidimo da vrijedi

$$r_i = (b_m \cdots b_2 b_1)_2 \to r_j^{(3)} = 0 \to r_j^{(4)} = b_1 \to r_i^{(5)} = (b_m \cdots b_2)_2 \to r_i^{(3)} = (b_m \cdots b_2)_2.$$

Primjećujemo da ovdje ulazimo u istu petlju od instrukcije 3. do 6. kao što je ona iz propozicije 2.56 pa lako zaključimo kako je semantika ove petlje

$$r_i^{(3)} = (b_m \cdots b_2 b_1)_2 \wedge r_j^{(3)} = 0 \to r_i^{(7)} = 0 \wedge r_j^{(7)} = (b_1 \cdots b_{m-1} b_m)_2,$$

iz čega slijedi da je $r_i' = 0 \wedge r_j' = (w^R)_2$, što je trebalo dokazati. Što se tiče vremenske složenosti, analogno razmišljanju u propoziciji 2.56, vidimo da je programski dio petlje vremenske složenosti $O(n)$. Uzimajući u obzir činjenicu da se ostatak binmakro-instrukcija izvrši konstantan broj puta, a po propoziciji 2.51 svaka od njih je vremenske složenosti $O(n)$, te činjenicu što slučaj kad je $r_i$ neparan ne ulazi u petlju, dolazimo do zaključka da je binmakro (ODDREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) vremenske složenosti $O(n)$. □

**Definicija 2.61:** Za različite $i, j \in \mathbb{N}$ definirajmo binmakro

$$(\text{SPECREVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_j) \coloneqq \left[\begin{array}{l} 0.\ \text{ODD } \mathcal{R}_i, 2 \\ 1.\ \text{GO TO } 6 \\ 2.\ \text{ZERO } \mathcal{R}_j \\ 3.\ \text{SHS } \mathcal{R}_j \\ 4.\ \text{ODDREVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \\ 5.\ \text{SHR } \mathcal{R}_j \end{array}\right]^{\flat *}. \tag{2.9}$$

◁

**Propozicija 2.62:** Neka su $i, j \in \mathbb{N}$ različiti. Tada binmakro
(SPECREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) za ulaz gdje je $r_i = (1w1)_2$ ima semantiku $r_i' = 0 \wedge r_j' = 1w^R$. Posebno, za ulaz gdje je $r_i = 1$, ovaj binmakro ima semantiku $r_i' = 0, r_j' = 1$. Također, binmakro (SPECREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) je vremenske složenosti $O(n)$.

*Dokaz.* Formalno, binmakro (SPECREVERSE $\mathcal{R}_i$ TO $\mathcal{R}_j$) za konfiguraciju u kojoj je $r_i \bmod 2 = 0$ ne utječe na registre iz istog razloga kao i binmakro ODDREVERSE. Za konfiguraciju u kojoj je $r_i = (1w1)_2$, praćenjem binmakro-instrukcija u programu, vidimo da binmakro ima semantiku

$$r_j^{(2)} = r_j \to r_j^{(3)} = 0 \to r_j^{(4)} = 1 \to r_i^{(5)} = 0 \wedge r_j^{(5)} = (1(w)^R1)_2 \to r_j^{(6)} = (1w^R)_2,$$

Dakle, za konfiguraciju u kojoj je $r_i = (1w1)_2$, ovaj binmakro ima semantiku $r_i' = 0 \wedge r_j' = (1w^R)_2$. Posebno, za slučaj $r_i = 1$, na isti način se vidi da ovaj binmakro ima semantiku $r_i' = 0 \wedge r_j' = 1$.
Što se tiče vremenske složenosti binmakroa SPECREVERSE, izvrši se konstantan broj binmakro-instrukcija koje su vremenske složenosti $O(n)$ iz čega zaključujemo da je binmakro SPECREVERSE vremenske složenosti $O(n)$. □

**Napomena 2.63:** Binmakroi SPECREVERSE i ODDREVERSE se čine dosta specifični s obzirom na svoju funkciju, a to je zato jer ćemo ih koristiti u posebnom slučaju pri izgradnji binmakroa za funkciju zbrajanja, dok će nam binmakro REVERSE poslužiti već sad pri izgradnji funkcijskog binmakroa. ◁

## 2.4 Funkcijski binmakro

U ovom potpoglavlju navest ćemo potrebne binmakroe za izgradnju funkcijskog binmakroa, dokazati njihovu vremensku složenost $O(n^k)$, za $k \in \mathbb{N}$ (vrijedit će čak $k = 1$) te ćemo kasnije i definirati i izgraditi sam funkcijski binmakro.

**Definicija 2.64:** Za različite $i, j, k \in \mathbb{N}$ definirajmo binmakro

$$(\text{REMOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \text{ USING } \mathcal{R}_k) := \left[\begin{array}{l} 0.\ \text{REVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_k \\ 1.\ \text{REVERSE } \mathcal{R}_k \text{ TO } \mathcal{R}_j \end{array}\right]^{\flat *}. \tag{2.10}$$

◁

**Propozicija 2.65:** Neka su $i, j, k \in \mathbb{N}$ različiti. Za proizvoljnu početnu konfiguraciju, binmakro (REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) ima semantiku $r_i' = 0 \wedge r_j' = r_i \wedge r_k' = 0$.
Također, binmakro (REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) je vremenske složenosti $O(n)$.

*Dokaz.* Formalno, može se vidjeti da binmakro (REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$), praćenjem binmakro-instrukcija, za ulaz s konfiguracijom gdje je $r_i = (1w)_2$ ima semantiku $r_i^{(1)} = 0 \wedge r_k^{(1)} = (1w^R)_2 \to r_k^{(1)} = 0 \wedge r_j^{(2)} = (1w)_2$. Za ulaz $r_i = 0$, (REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) pak ima semantiku $r_i^{(1)} = 0 \wedge r_k^{(1)} = 0 \to r_j^{(2)} = 0$.
Iz navedenog slijedi da binmakro za bilo koju konfiguraciju ima semantiku $r'_j = r_i, r'_k = 0$. Zbog toga možemo reći da binmakro (REMOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) premješta vrijednost iz $\mathcal{R}_i$ u $\mathcal{R}_j$ koristeći $\mathcal{R}_k$, što se da iščitati iz naziva samog binmakroa.
Što se tiče vremenske složenosti binmakroa, ona je jednaka $O(n)$, jer se za svaki ulaz izvedu dvije binmakro-instrukcije vremenske složenosti $O(n)$. □

**Definicija 2.66:** Za različite $i, j, k \in \mathbb{N}$ definirajmo binmakro

$$(\text{MOVE } \mathcal{R}_i \text{ TO } \mathcal{R}_j \text{ USING } \mathcal{R}_k) \coloneqq \left[\begin{array}{l} 0.\ \text{REVERSE } \mathcal{R}_i \text{ TO } \mathcal{R}_k \\ 1.\ \text{DREVERSE } \mathcal{R}_k \text{ TO } \mathcal{R}_j \text{ AND } \mathcal{R}_i \end{array}\right]^{\flat *}. \tag{2.11}$$
◁

**Propozicija 2.67:** Neka su $i, j, k \in \mathbb{N}$ različiti. Za proizvoljnu početnu konfiguraciju, binmakro (MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) ima semantiku $r'_j = r_i \wedge r'_k = 0$.
Također, binmakro (MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) je vremenske složenosti $O(n)$.

*Dokaz.* Formalno, može se vidjeti da binmakro (MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$), po propozicijama 2.56 i 2.58, za ulaz s konfiguracijom gdje je $r_i = (1w)_2$ ima semantiku

$$r_i^{(1)} = 0 \wedge r_k^{(1)} = (1w^R)_2 \to r_k^{(2)} = 0 \wedge r_j^{(2)} = (1w)_2 \wedge r_i^{(2)} = (1w)_2.$$

Za ulaz $r_i = 0$, binmakro (MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) pak ima semantiku

$$r_i^{(1)} = 0 \wedge r_k^{(1)} = 0 \to r_j^{(2)} = 0.$$

Iz navedenog slijedi da binmakro za bilo koju konfiguraciju ima semantiku $r'_j = r_i \wedge r_k = 0$. Zbog toga možemo reći da binmakro (MOVE $\mathcal{R}_i$ TO $\mathcal{R}_j$ USING $\mathcal{R}_k$) kopira vrijednost iz $\mathcal{R}_i$ u $\mathcal{R}_j$ koristeći $\mathcal{R}_k$, što nije možda intuitivno, ali kao što je navedeno u [1], moderne računalne arhitekture uglavnom terminološki prate arhitekturu `x86`, koja instrukciju za kopiranje podataka između registara (i drugih lokacija) zove `mov`.
Što se tiče vremenske složenosti binmakroa, ona je jednaka $O(n)$, jer se za svaki ulaz izvedu dvije binmakro-instrukcije vremenske složenosti $O(n)$. □

U preostalim binmakroima koje ćemo definirati za funkcijski binmakro, i u samom funkcijskom binmakrou, semantika i binmakro-instrukcije će biti gotovo istovjetne onima u [1], tako da se definicija tih binmakroa skoro pa i neće razlikovati u odnosu na one sa RAM-instrukcijama.

**Definicija 2.68:** Neka su $i, j, k, m \in \mathbb{N}$ takvi da $|i-j| \geq m$ i $k \geq \max\{i+n, j+n\}$. Definirajmo binmakro

$$(\text{MMOVE } m \text{ FROM } \mathcal{R}_i.. \text{ TO } \mathcal{R}_j.. \text{ USING } \mathcal{R}_k) \coloneqq \left[ \text{ t. REMOVE } \mathcal{R}_{i+t} \text{ TO } \mathcal{R}_{j+t} \text{ USING } \mathcal{R}_k \right]_{t<m}^{b*}. \quad (2.12)$$

◁

**Propozicija 2.69:** Neka su $i, j, k, m \in \mathbb{N}$ takvi da $|i - j| \geq m$ i $k \geq \max\{i + n, j + n\}$. Binmakro (MMOVE $m$ FROM $\mathcal{R}_i..$ TO $\mathcal{R}_j..$ USING $\mathcal{R}_k$) za proizvoljan ulaz ima semantiku $(\forall t < m)(r'_{j+t} = r_{i+t}) \wedge r'_k = 0$.
Također, vremenska složenost binmakroa (MMOVE $m$ FROM $\mathcal{R}_i..$ TO $\mathcal{R}_j..$ USING $\mathcal{R}_k$) je $O(n)$.

*Dokaz.* Za binmakro (MMOVE $m$ FROM $\mathcal{R}_i..$ TO $\mathcal{R}_j..$ USING $\mathcal{R}_k$) se može vidjeti, praćenjem instrukcija i s pomoću propozicije 2.67, da ima semantiku

$$(\forall t < m)(r_{j+t}^{(t+1)} = r_{i+t}^{(t)} \wedge r_{i+t}^{(t+1)} = 0) \wedge r_k^{(t+1)} = 0) \rightarrow (\forall t < m)(r'_{j+t} = r_{i+t} \wedge r'_{i+t} = 0) \wedge r'_k = 0$$

Za svaki $m \in \mathbb{N}_+$, vremenska složenost ovog binmakroa je $O(n)$, iz razloga što, iako se je $m$ izvršenja binmakro instrukcija složenosti $O(n)$, taj $m$ je tu konstantan, te možemo reći da je za svaki $m \in \mathbb{N}, m > 1$ binmakro vremenske složenosti $O(m \cdot n)$ što je jednako $O(n)$. □

**Definicija 2.70:** Neka je $k \in \mathbb{N}_+$ te $j_1, j_2, \ldots, j_k > k$ prirodni brojevi. Definiramo binmakro

$$(\text{ARGS } \mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}) \coloneqq \left[ \text{ t. MOVE } \mathcal{R}_{j_{t+1}} \text{ TO } \mathcal{R}_{t+1} \text{ USING } \mathcal{R}_0 \right]_{t<k}^{b*}. \quad (2.13)$$

◁

**Propozicija 2.71:** Binmakro (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$) je dobro definiran, njegova je semantika $(\forall t \in [1..k])(r'_t = r_{j_t}) \wedge r'_0 = 0$, te je vremenske složenosti $O(n)$.

*Dokaz.* Iz uvjeta definicije binmakroa (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$) vidimo da vrijedi $(\forall t \in [1..k])(0 < 1 \leq t \leq k < j_t)$, a to povlači

$$\begin{aligned} (\forall t \in [1..k])(0 < 1 \leq t \leq k < j_t) &\implies (\forall t \in [1..k])(0 < t < j_t) \\ (\forall t \in [1..k])(0 < t < j_t) &\implies (\forall t < k)(0 < t + 1 < j_{t+1}), \end{aligned}$$

iz čega vidimo da su za sve $t < k$ uvjeti za parametre od (MOVE $\mathcal{R}_{j_{t+1}}$ TO $\mathcal{R}_{t+1}$ USING $\mathcal{R}_0$) dobro definirani, iz čega slijedi da je (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$) dobro definiran.
Čitanjem binmakro-instrukcija redom vidimo da binmakro (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$) za početnu konfiguraciju s proizvoljnim ulazom ima semantiku $(\forall t < k)(r_{t+1}^{(t)} = r_{j_{t+1}} \wedge r_0^{(t)} = 0$, što, budući da se za svaku binmakro-instrukciju vrijednost registra $\mathcal{R}_0$ postavlja na 0 i da se $(\forall t < k)(\mathcal{R}_{t+1})$ obrađuje samo u jednoj binmakro-instrukciji u programu, povlači da je $(\forall t < k)(r'_{t+1} = r_{j_{t+1}} \wedge r'_0 = 0$.

Što se tiče vremenske složenosti ovog binmakroa (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$), vidimo da se $\forall k \in \mathbb{N}_+$, za proizvoljnu početnu konfiguraciju s proizvoljnim ulazom bude $k$ izvršenja binmakro-instrukcija vremenske složenosti $O(n)$, iz čega slijedi da je $\forall k \in \mathbb{N}_+$ vremenska složenost binmakroa (ARGS $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}, \ldots, \mathcal{R}_{j_k}$) jednaka $O(k \cdot n)$, a budući da je $k$ fiksiran, to je jednako $O(n)$. □

Sad imamo definirane sve potrebne binmakroe kako bismo napokon definirali funkcijski binmakro, koji će nam, kao u [1], omogućiti funkcijske pozive za bilo koju BRAM-izračunljivu funkciju (za koju imamo BRAM-program) na bilo kojim registrima kao argumentima, spremajući rezultat u po volji odabrani registar i čuvajući po volji velik početni komad memorije.

**Definicija 2.72:** Neka je $k \in \mathbb{N}_+$, $\mathsf{f}^k$ totalna brojevna funkcija te $P_{\mathsf{f}}^k$ totalan BRAM-algoritam koji računa $\mathsf{f}^k$. Neka su $m, j_0, j_1, \ldots, j_k \in \mathbb{N}$. Definiramo $b := 1 + \max\{m_{P_{\mathsf{f}}} - 1, m - 1, k, j_0, j_1, \ldots, j_k\}$ te pomoću tog broja definiramo *funkcijski binmakro*

$$(P_{\mathsf{f}}(\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}) \to \mathcal{R}_{j_0} \text{ USING } \mathcal{R}_m..) := \left[\begin{array}{l} 0.\ \text{MMOVE } b \text{ FROM } \mathcal{R}_0.. \text{ TO } \mathcal{R}_b.. \text{ USING } \mathcal{R}_{2b} \\ 1.\ \text{ARGS } \mathcal{R}_{b+j_1}, \mathcal{R}_{b+j_2}, \ldots, \mathcal{R}_{b+j_k} \\ 2.\ P_{\mathsf{f}}^* \\ 3.\ \text{REMOVE } \mathcal{R}_0 \text{ TO } \mathcal{R}_{b+j_0} \text{ USING } \mathcal{R}_{2b} \\ 4.\ \text{MMOVE } b \text{ FROM } \mathcal{R}_b.. \text{ TO } \mathcal{R}_0.. \text{ USING } \mathcal{R}_{2b} \end{array}\right]^{b*}. \quad \triangleleft \tag{2.14}$$

**Propozicija 2.73:** Semantika funkcijskog binmakroa, uz oznake iz definicije 2.72 te pokratu $\vec{r} := (r_{j_1}, r_{j_2}, \ldots, r_{j_k})$, glasi: $r'_{j_0} = \mathsf{f}(\vec{r}) \wedge (\forall t \in [b..2b])(r'_t = 0)$ (Specijalno, zbog $b \geq m$, za sve $t \in [0..m\rangle \setminus \{j_0\}$ je $r'_t = r_t$).
Također, ako je vremenska složenost od $P_{\mathsf{f}}^*$ jednaka $O(n^t)$, za neki $t \in \mathbb{N}_+$, tada je i vremenska složenost od $(P_{P_{\mathsf{f}}}(\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}) \to \mathcal{R}_{j_0} \text{ USING } \mathcal{R}_m..)$ jednaka $O(n^t)$.

*Dokaz.* Praćenjem instrukcija i po propozicijama 2.69 i 2.71, vidimo da vrijedi

$$(\forall t < b)(r^{(1)}_{b+t} = r_{j_t} \wedge r^{(1)}_t = 0) \wedge r^{(1)}_{2b} = 0 \to (\forall t \in [1..k])(r^{(2)}_t = r^{(1)}_{b+j_t} = r_{j_t}) \wedge r^{(2)}_0 = 0.$$

Iz toga slijedi da je po definiciji $P_{\mathsf{f}}^*$ i po propozicijama 2.65 i 2.69:

$$\begin{gathered} (\forall t \in [1..k])(r^{(2)}_t = r_{j_t}) \wedge r^{(2)}_0 = 0 \to r^{(3)}_0 = \mathsf{f}(\vec{r}) \wedge (\forall t \in [1..m_{P_{\mathsf{f}}}\rangle)(r^{(3)}_t = ?), \\ r^{(3)}_0 = \mathsf{f}(\vec{r}) \wedge (\forall t \in [1..m_{P_{\mathsf{f}}}\rangle)(r^{(3)}_t = ?) \to r^{(4)}_{b+j_0} = r^{(3)}_0 = \mathsf{f}(\vec{r}) \wedge r^{(4)}_0 = 0 \wedge r^{(4)}_{2b} = 0, \\ r^{(4)}_{b+j_0} = \mathsf{f}(\vec{r}) \wedge r^{(4)}_0 = 0 \wedge r^{(4)}_{2b} = 0 \to (\forall t < b)(r^{(5)}_t = r^{(4)}_{b+t} \wedge r^{(5)}_{b+t} = 0) \wedge r^{(5)}_{2b} = 0, \\ (\forall t < b)(r^{(5)}_t = r^{(4)}_{b+t} \wedge r^{(5)}_{b+t} = 0) \wedge r^{(5)}_{2b} = 0 \to r^{(5)}_{j_0} = \mathsf{f}(\vec{r}) \wedge (\forall t \in [0..b\rangle \setminus \{j_0\})(r^{(5)}_t = r_t). \end{gathered}$$

Iz ovoga slijedi da binmakro za $\vec{r} \in \mathcal{D}_{\mathsf{f}}$ ima semantiku

$$r'_{j_0} = \mathsf{f}(\vec{r}) \wedge (\forall t \in [0..b\rangle \setminus \{j_0\})(r'_t = r_t) \wedge (\forall t \in [b..2b])(r'_t = 0).$$

Posebno, zbog $b \geq m$, za sve $t \in [0..m\rangle \setminus \{j_0\}$ je $r'_t = r_t$.
Što se tiče vremenske složenosti, pretpostavimo da je vremenska složenost od $P^*_{\mathsf{f}}$ jednaka $O(n^t)$, za neki $t \in \mathbb{N}_+$. Po propozicijama 2.69, 2.71, 2.65, binmakro-instrukcije s rednim brojevima 0., 1., 3. i 4., su polinomne vremenske složenosti. Uzimajući u obzir da je vremenska složenost 2. binmakro-instrukcije (odnosno $P^*_{\mathsf{f}}$) jednaka $O(n^t)$ i da ima ukupno 5 koraka za proizvoljno vanjsko izračunavanje, što znači da je $(P_{P_{\mathsf{f}}}(\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}) \to \mathcal{R}_{j_0}$ USING $\mathcal{R}_m..)$ konstantne vanjske vremenske složenosti pa i su registri polinomne veličine u svakom koraku vanjskog izračunavanja, slijedi iz korolara 2.44 slijedi da je vremenska složenost od $(P_{P_{\mathsf{f}}}(\mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}) \to \mathcal{R}_{j_0}$ USING $\mathcal{R}_m..)$ polinomna. □

# Poglavlje 3

# Zbrajanje

**Lema 3.1:** Neka su $a, b \in \mathbb{N}$. Tada je

$$(a + b) \bmod 2 = (a \bmod 2 + b \bmod 2) \bmod 2.$$

*Dokaz.* Postoje $c, d \in \mathbb{N}$ takvi da je $a = c \cdot 2 + a \bmod 2$ i $b = d \cdot 2 + d \bmod 2$. Iz toga slijedi

$$\begin{aligned}(a + b) \bmod 2 &= (c \cdot 2 + a \bmod 2 + d \cdot 2 + d \bmod 2) \bmod 2\\ (c \cdot 2 + a \bmod 2 + d \cdot 2 + d \bmod 2) \bmod 2 &= ((c + d) \cdot 2 + a \bmod 2 + d \bmod 2) \bmod 2\\ ((c + d) \cdot 2 + a \bmod 2 + d \bmod 2) \bmod 2 &= (a \bmod 2 + b \bmod 2)\end{aligned}$$

□

**Definicija 3.2:** Definirajmo binmakro

$$(\textsc{bitAdd}) \coloneqq \left[\begin{array}{l} 0.\ \textsc{odd}\ \mathcal{R}_1, 6\\ 1.\ \textsc{odd}\ \mathcal{R}_2, 4\\ 2.\ \textsc{shl}\ \mathcal{R}_3\\ 3.\ \textsc{go to}\ 10\\ 4.\ \textsc{shs}\ \mathcal{R}_3\\ 5.\ \textsc{go to}\ 10\\ 6.\ \textsc{odd}\ \mathcal{R}_2, 9\\ 7.\ \textsc{shs}\ \mathcal{R}_3\\ 8.\ \textsc{go to}\ 10\\ 9.\ \textsc{shl}\ \mathcal{R}_3 \end{array}\right]^{\flat *}. \tag{3.1}$$

◁

**Propozicija 3.3:** Binmakro BITADD za početnu konfiguraciju s proizvoljnim ulazom ima semantiku $r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2$.
Također, vremenska složenost ovog binmakroa je $O(1)$.

*Dokaz.* Promotrimo 4 moguća slučaja za ulaz po parnosti $r_i$ i $r_j$:

- $r_1 \bmod 2 = 1 \wedge r_2 \bmod 2 = 1$. Praćenjem instrukcija u binmakrou i po lemi 3.1 vidimo da je

$$r_1 \bmod 2 = 1 \wedge r_2 \bmod 2 = 1 \to pc = 6 \to pc = 9 \to r_3^{(10)} = r_3' = r_3 \cdot 2,$$
$$r_3' = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (1 + 1) \bmod 2 = r_3 \cdot 2 + (r_1 \bmod 2 + r_2 \bmod 2) \bmod 2;$$
$$r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2.$$

- $r_1 \bmod 2 = 1 \wedge r_2 \bmod 2 = 0$. Praćenjem instrukcija u binmakrou i po lemi 3.1 vidimo da je

$$r_1 \bmod 2 = 1 \wedge r_2 \bmod 2 = 0 \to pc = 6 \to pc = 7 \to r_3^{(8)} = r_3 \cdot 2 + 1 \to pc = 10,$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (1 + 0) \bmod 2 = r_3 \cdot 2 + (r_1 \bmod 2 + r_2 \bmod 2) \bmod 2,$$
$$r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2.$$

- $r_1 \bmod 2 = 0 \wedge r_2 \bmod 2 = 1$. Praćenjem instrukcija u binmakrou i po lemi 3.1 vidimo da je

$$r_1 \bmod 2 = 0 \wedge r_2 \bmod 2 = 1 \to pc = 1 \to r_3^{(4)} = r_3 \cdot 2 \to pc = 10,$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (0 + 1) \bmod 2 = r_3 \cdot 2 + (r_1 \bmod 2 + r_2 \bmod 2) \bmod 2,$$
$$r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2.$$

- $r_1 \bmod 2 = 0 \wedge r_2 \bmod 2 = 0$. Praćenjem instrukcija u binmakrou i po lemi 3.1 vidimo da je

$$r_1 \bmod 2 = 1 \wedge r_2 \bmod 2 = 1 \to pc = 1 \to pc = 2 \to r_3^{(3)} = r_3 \cdot 2 \to pc = 10,$$
$$r_3' = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (0 + 0) \bmod 2 = r_3 \cdot 2 + (r_1 \bmod 2 + r_2 \bmod 2) \bmod 2,$$
$$r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2.$$

Slijedi da binmakro (BITADD) za početnu konfiguraciju s proizvoljnim ulazom ima semantiku $r_3' = r_3 \cdot 2 + (r_1 + r_2) \bmod 2$.
Također, za svaki od slučaja broj koraka je konstantan u binmakrou pa slijedi da je binmakro (BITADD2) vremenske složenosti $O(1)$. □

**Definicija 3.4:** Definirajmo binmakro

$$(\textsc{storedBitAdd}) \coloneqq \left[\begin{array}{l} 0.\ \textsc{bitAdd} \\ 1.\ \textsc{odd}\ \mathcal{R}_4, 8 \\ 2.\ \textsc{odd}\ \mathcal{R}_1, 4 \\ 3.\ \textsc{go to}\ 17 \\ 4.\ \textsc{odd}\ \mathcal{R}_2, 6 \\ 5.\ \textsc{go to}\ 17 \\ 6.\ \textsc{shs}\ \mathcal{R}_4 \\ 7.\ \textsc{go to}\ 17 \\ 8.\ \textsc{odd}\ \mathcal{R}_3, 15 \\ 9.\ \textsc{shr}\ \mathcal{R}_3 \\ 10.\ \textsc{shs}\ \mathcal{R}_3 \\ 11.\ \textsc{odd}\ \mathcal{R}_1, 17 \\ 12.\ \textsc{odd}\ \mathcal{R}_2, 17 \\ 13.\ \textsc{shr}\ \mathcal{R}_4 \\ 14.\ \textsc{go to}\ 17 \\ 15.\ \textsc{shr}\ \mathcal{R}_3 \\ 16.\ \textsc{shl}\ \mathcal{R}_3 \end{array}\right]^{\flat *} . \tag{3.2}$$

◁

**Propozicija 3.5:** Semantika binmakroa (storedBitAdd) za proizvoljnu početnu konfiguraciju je $r_3' = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2 \wedge r_4' \bmod 2 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor$.
Također, binmakro-program (storedBitAdd) je polinomne vremenske složenosti.

*Dokaz.* Promotrimo semantiku binmakroa pri vanjskom (storedBitAdd)-izračunavanju za proizvoljnu početnu konfiguraciju $c$. Podijelimo vanjsko (storedBitAdd)-izračunavanje s $c$ na slučajeve po vrijednostima $r_1, r_2, r_3$. Imamo

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 0)$

$$\begin{gathered} (r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0,0,0) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2, \\ r_3^{(1)} = r_3 \cdot 2 \to pc = 2 \to pc = 3 \to pc = 17; \\ r_3' = r_3 \cdot 2 = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2, \\ r_4' \bmod 2 = r_4 \bmod 2 = 0 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor. \end{gathered}$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 \to pc = 8 \to pc = 9 \to r_3^{(10)} = \lfloor \frac{r_3^{(9)}}{2} \rfloor = r_3,$$
$$r_3^{(10)} = r_3 \to r_3^{(11)} = r_3^{(10)} \cdot 2 + 1 = r_3 \cdot 2 + 1 \to pc = 12 \to pc = 13,$$
$$pc = 13 \to r_4^{(14)} = \lfloor \frac{r_4^{(13)}}{2} \rfloor = \lfloor \frac{r_4}{2} \rfloor \to pc = 17;$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = \lfloor \frac{r_4}{2} \rfloor \bmod 2 = 0 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 1, 1)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 + 1 \to pc = 8 \to pc = 15 \to r_3^{(16)} = \lfloor \frac{r_3^{(15)}}{2} \rfloor = \lfloor \frac{r_3 \cdot 2 + 1}{2} \rfloor = r_3,$$
$$r_3^{(16)} = r_3 \to r_3^{(17)} = r_3^{(16)} \cdot 2 = r_3 \cdot 2;$$
$$r_3' = r_3 \cdot 2 = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = r_4 \bmod 2 = 1 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 1, 0)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 1, 0) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 + 1 \to pc = 2 \to pc = 3 \to pc = 17;$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = r_4 \bmod 2 = 0 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (1, 0, 0)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (1, 0, 0) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 + 1 \to pc = 2 \to pc = 4 \to pc = 5 \to pc = 17;$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = r_4 \bmod 2 = 0 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (1, 1, 0)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 \to pc = 2 \to pc = 4 \to pc = 6 \to r_4^{(7)} = r_4^{(6)} \cdot 2 + 1 = r_4 \cdot 2 + 1,$$
$$r_4^{(7)} = r_4 \cdot 2 + 1 \to pc = 17;$$
$$r_3' = r_3 \cdot 2 = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = (r_4 \cdot 2 + 1) \bmod 2 = 1 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (1, 0, 1)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 + 1 \to pc = 8 \to pc = 15 \to r_3^{(16)} = \lfloor \frac{r_3^{(15)}}{2} \rfloor = \lfloor \frac{r_3 \cdot 2 + 1}{2} \rfloor = r_3,$$
$$r_3^{(16)} = r_3 \to r_3^{(17)} = r_3^{(16)} \cdot 2 = r_3 \cdot 2;$$
$$r_3' = r_3 \cdot 2 = r_3 \cdot 2 + 0 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = r_4 \bmod 2 = 1 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

- $(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (1, 1, 1)$

$$(r_1 \bmod 2, r_2 \bmod 2, r_4 \bmod 4) = (0, 0, 1) \to r_3^{(1)} = r_3 \cdot 2 + (r_1 + r_2) \bmod 2,$$
$$r_3^{(1)} = r_3 \cdot 2 \to pc = 8 \to pc = 9 \to r_3^{(10)} = \lfloor \frac{r_3^{(9)}}{2} \rfloor = \lfloor \frac{r_3 \cdot 2}{2} \rfloor = r_3,$$
$$r_3^{(10)} = r_3 \to r_3^{(11)} = r_3^{(10)} \cdot 2 + 1 = r_3 \cdot 2 + 1 \to pc = 17;$$
$$r_3' = r_3 \cdot 2 + 1 = r_3 \cdot 2 + (r_1 + r_2 + r_4) \bmod 2,$$
$$r_4' \bmod 2 = r_4 \bmod 2 = 1 = \lfloor \frac{r_1 \bmod 2 + r_2 \bmod 2 + r_4 \bmod 2}{2} \rfloor.$$

Slijedi tvrdnja za semantiku iz propozicije. Što se tiče vremenske složenosti binmakroa (STOREDBITADD), vidimo da je vanjska vremenska složenost $O(1)$ po vanjskom izračunavanju koje smo koristili za određivanje semantike binmakro-programa (STOREDBITADD). Zatim, budući da je binmakro (BITADD) vremenske složenosti $O(1)$ i da je veličina registra tijekom vanjskog izračunavanja uvijek $O(n)$, jer je broj vanjskih koraka konstantan, po teoremu 2.37 slijedi da je vremenska složenost binmakroa (STOREDBITADD) polinomna. □

**Napomena 3.6:** U binmakro-algoritmu za zbrajanje dva broja simulirat ćemo proceduru pisanog zbrajanja u binarnom sustavu za brojeve $a, b \in \mathbb{N}$. Neka su $a = (a_l \cdots a_1 a_0)_2$ i $b = (b_m \cdots b_1 b_0)_2$. Za $k = \max\{l, m\} + 1$, definiramo konačni niz $d = (d_k, \cdots, d_1, 0)$ gdje je $d_{i+1} = \lfloor \frac{a_i + b_i + d_i}{2} \rfloor$ za $i < k$, uz napomenu da su $a$ i $b$ prošireni nulama do $a_k$ i $b_k$. Konačni niz $d$ nam predstavlja pamćene znamenke prilikom procedure pisanog zbrajanja te zbroj $a$ i $b$ predstavlja $c = (c_k \cdots c_1 c_0)_2$ gdje je $c_i = (a_i + b_i + d_i) \bmod 2$, uz napomenu da je $a_k = b_k = 0$, jer je $k > l \wedge k > m$ pa je posebno $a_k = d_k \bmod 2$, odnosno $a_k = d_k$. ◁

**Definicija 3.7:** Definirajmo binmakro-algoritam

$$(\textsc{bAdd}_2) \coloneqq \left[\begin{array}{l} 0.\ \textsc{bz}\ \mathcal{R}_1, 2 \\ 1.\ \textsc{go to}\ 3 \\ 2.\ \textsc{bz}\ \mathcal{R}_2, 15 \\ 3.\ \textsc{shs}\ \mathcal{R}_3 \\ 4.\ (\textsc{storedBitAdd}) \\ 5.\ \textsc{shr}\ \mathcal{R}_1 \\ 6.\ \textsc{shr}\ \mathcal{R}_2 \\ 7.\ \textsc{bz}\ \mathcal{R}_1, 9 \\ 8.\ \textsc{go to}\ 4 \\ 9.\ \textsc{bz}\ \mathcal{R}_2, 11 \\ 10.\ \textsc{go to}\ 4 \\ 11.\ \textsc{odd}\ \mathcal{R}_4, 13 \\ 12.\ \textsc{go to}\ 14 \\ 13.\ \textsc{shs}\ \mathcal{R}_3 \\ 14.\ \textsc{specreverse}\ \mathcal{R}_3\ \textsc{to}\ \mathcal{R}_0 \end{array}\right]^2 . \tag{3.3}$$

◁

**Propozicija 3.8:** Semantika totalnog binmakro-algoritma $\textsc{bAdd}_2$ za početnu konfiguraciju s proizvoljnim ulazom $(a, b) \in \mathbb{N}^2$ je $r'_0 = a + b$. Također, vremenska složenost binmakro-algoritma ($\textsc{bAdd}_2$) je polinomna.

*Dokaz.* Neka je $(a, b) \in \mathbb{N}^2$ proizvoljan. Promotrimo vanjsko ($\textsc{bAdd}_2$)-izračunavanje s početnom konfiguracijom $c$ s ulazom $(a, b)$. Neka je $a = (a_l \cdots a_1 a_0)_2$ i $b = (b_m \cdots b_1 b_0)$, $d = (d_k, \cdots, d_1, d_0)$ neka predstavlja niz pamćenih znamenaka u proceduri pisanog zbrajanja, a $c = (c_k \cdots c_1 c_0)_2$ rezultat procedure pisanog zbrajanja za $a$ i $b$, odnosno $c = a + b$, uz napomenu da ćemo smatrati da su $a$ i $b$ prošireni nulom do $a_k$ i $b_k$ za $k = \max\{l, m\} + 1$. Promotrimo vanjsko izračunavanje po slučajevima za vrijednosti $a$ i $b$.

- $a = 0 \wedge b = 0$

$$r_1 = 0 \wedge r_2 = 0 \to pc = 2 \to pc = 15;$$
$$r'_0 = 0 = a + b$$

- $a \neq 0 \vee b \neq 0$ Budući da je $a \neq 0 \vee b \neq 0$, vrijedi

$$r_1 \neq 0 \vee r_2 \neq 0 \rightarrow pc = 1 \vee pc = 2 \rightarrow pc = 3.$$

Zatim imamo

$$r_1^{(3)} = a = (a_k \dots a_1 a_0)_2 \wedge r_2^{(3)} = b = (b_k \dots b_1 b_0)_2 \rightarrow r_3^{(4)} = 1,$$
$$r_3^{(4)} = 1 \rightarrow r_3^{(5)} = r_3^{(3)} \cdot 2 + (a + b + 0) \bmod 2 \wedge r_4^{(5)} \bmod 2 = \lfloor \frac{a_0 + b_0 + d_0}{2} \rfloor,$$
$$r_3^{(5)} = 1 \cdot 2 + (a_0 + b_0 + d_0) \bmod 2 = 1 \cdot 2 + c_0 = (1c_0)_2 \wedge r_4^{(5)} \bmod 2 = d_1,$$
$$r_3^{(5)} = (1c_0)_2 \wedge r_4^{(5)} = d_1 \rightarrow r_1^{(6)} = (a_k \dots a_2 a_1)_2 \rightarrow r_2^{(7)} = (b_k \dots b_2 b_1)_2$$

Sad dolazimo do grananja gdje ako je $r_2^{(7)} \neq 0 \vee r_1^{(7)} \neq 0$, onda je

$$r_1^{(7)} = (a_k \dots a_2 a_1)_2 \neq 0 \vee r_2^{(7)} = (b_k \dots b_2 b_1)_2 \neq 0 \rightarrow pc = 8 \vee pc = 9,$$
$$pc = 8 \vee pc = 9 \rightarrow pc = 4 \vee pc = 11 \rightarrow pc = 4 \rightarrow r_3^{(5)} = (1c_0c_1)_2 \wedge r_4^{(5)} = d_2$$

Primijetimo da smo ušli u izvršavanje petlje od instrukcija 4. do 10. za koju se lako induktivno pokaže da će se petlja izvršiti nakon $k$ iteracija (jer je $a_k = 0 \wedge b_k = 0 \wedge (a_{k-1} \neq 0 \vee b_{k-1} \neq 0)$), odnosno kad se registri $\mathcal{R}_1$ i $\mathcal{R}_2$ isprazne pa će zadnja iteracija petlje izgledati

$$r_3^{(5)} = (1c_0c_1 \cdots c_{k-1})_2 \wedge r_4^{(5)} \bmod 2 = d_k \rightarrow r_1^{(6)} = 0 \rightarrow r_2^{(7)} = 0,$$
$$r_2^{(7)} = 0 \rightarrow pc = 9 \rightarrow pc = 11.$$

Sad dolazimo do grananja ovisno o parnosti $r_4^{(11)}$, odnosno o vrijednosti $d_k$. Za $d_k = 1$ vrijedi

$$r_4^{(11)} \bmod 2 = d_k = 1 \rightarrow pc = 13,$$
$$pc = 13 \rightarrow r_3^{(14)} = (1c_0c_1 \cdots c_{k-1}1)_2 = (1c_0c_1 \cdots c_{k-1}d_k)_2 = (1c_0c_1 \cdots c_{k-1}c_k)_2,$$
$$r_3^{(14)} = (1c_0 \cdots c_{k-1}c_k)_2 \rightarrow r_0^{(15)} = (c_kc_{k-1} \cdots c_0)_2 = c \wedge r_3^{(15)} = 0;$$
$$r_0' = c = a + b.$$

Za $d_k = 0$ vrijedi

$$r_4^{(11)} \bmod 2 = d_k = 0 \rightarrow pc = 12,$$
$$pc = 14 \rightarrow r_0^{(15)} = (c_{k-1} \cdots c_1c_0)_2 \wedge r_3^{(15)} = 0;$$
$$r_0' = (c_{k-1} \cdots c_1c_0)_2 = c = a + b \text{ (jer je } c_k = d_k = 0).$$

Slijedi tvrdnja za semantiku u propoziciji, posebno, binmakro-algoritam ($\text{bADD}_2$) je totalan jer iz prikazanog vidimo da stane za sve ulaze $x \in \mathbb{N}^2$.
Što se tiče vremenske složenosti binmakro-algoritma ($\text{bADD}_2$), vidimo da je njegova vanjska složenost $O(n)$ za ulaz veličine $n$, budući da se u vanjskom izračunavanju petlja iterira $k$ puta i vrijedi $n + 1 \geq k$, i da je broj vanjskih koraka van petlje konstantan. Veličina svakog registra tijekom vanjskog izračunavanja uvijek $O(n)$, što je testirano u programskom dijelu diplomskog i može se provjeriti pomnim praćenjem izvršavanja vanjskih naredbi u $\mathcal{R}_4$ gdje su vrijednosti 0 ili 1, a za ostale registre je praćena vrijednost u svakom vanjskom koraku pri opisu semantike binmakro-algoritma ($\text{bADD}_2$). Zatim, budući da su binmakroi $(\text{STOREDBITADD})^{\flat *}$ i SPECREVERSE $\mathcal{R}_3$ TO $\mathcal{R}_0$ polinomne vremenske složenosti, po teoremu 2.37 slijedi da je binmakro-algoritam ($\text{bADD}_2$) polinomne vremenske složenosti. □

**Napomena 3.9:** Potvrđeno testovima u programskom dijelu diplomskog rada te pomnim promatranjem izvršavanja koraka u binmakro-algoritmu ($\text{bADD}_2$), može se vidjeti da je ($\text{bADD}_2$) vremenske složenosti $O(n)$. ◁

**Korolar 3.10:** Funkcija zbrajanja $add^2 : \mathbb{N}^2 \to \mathbb{N}$, definirana sa $add^2(x_1, x_2) = x_1 + x_2$, je polinomno BRAM-izračunljiva.

*Dokaz.* Iz propozicije 3.8 slijedi da binmakro-algoritam ($\text{bADD}_2$) računa funkciju $add^2$, iz čega slijedi da BRAM-algoritam mjesnosti 2 spljoštenja binmakro-programa od ($\text{bADD}_2$) također računa $add^2$. Sad imamo zbog korolara 2.44 da je $add^2$ BRAM-izračunljiva. □

**Definicija 3.11:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$ te neka je $t = 5$ ako je $k < 5$, inače neka je $t = k + 1$. Definirajmo binmakro-algoritam

$$(\text{bADD}_k) \coloneqq \begin{bmatrix} 0.\ (\text{bADD2}(\mathcal{R}_0, \mathcal{R}_1) \to \mathcal{R}_0 \text{ USING } \mathcal{R}_t..) \\ 1.\ (\text{bADD2}(\mathcal{R}_0, \mathcal{R}_2) \to \mathcal{R}_0 \text{ USING } \mathcal{R}_t..) \\ \vdots \\ (k-1).\ (\text{bADD2}(\mathcal{R}_0, \mathcal{R}_k) \to \mathcal{R}_0 \text{ USING } \mathcal{R}_t..) \end{bmatrix}^k . \tag{3.4}$$

◁

**Propozicija 3.12:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Semantika totalnog binmakro-algoritma ($\text{bADD}_k$) za početnu konfiguraciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$ je $r'_0 = \sum_{i=1}^{k} x_i$. Također, binmakro-algoritam ($\text{bADD}_k$) je polinomne vremenske složenosti.

*Dokaz.* Promotrimo semantiku kroz vanjsko ($\text{bADD}_k$)-izračunavanje za početnu konfigura-

ciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$. Vrijedi

$$(r_1, r_2, \cdots, r_k) = \vec{x} \to r_0^{(1)} = r_1 = x_1,$$
$$r_0^{(1)} = x_1 \to r_0^{(2)} = r_0^{(1)} + x_2 = x_1 + 2,$$
$$\vdots$$
$$r_0^{(k-1)} = \sum_{i=1}^{k-1} x_i \to r_0^{(k)} = r_0^{(k-1)} + x_k = \sum_{i=1}^{k-1} x_i + x_k = \sum_{i=1}^{k} x_i;$$
$$r_0' = \sum_{i=1}^{k} x_i.$$

Vrijedi tvrdnja za semantiku iz propozicije, a posebno je binmakro-algoritam ($\textsc{bAdd}_k$) totalan, jer iz priloženog vidimo da ($\textsc{bAdd}_k$)-izračunavanje stane za početnu konfiguraciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$. Što se tiče vremenske složenosti, vidimo da je binmakro-algoritam ($\textsc{bAdd}_k$) vanjske vremenske složenosti $O(1)$, da je ($\textsc{bAdd2}(\mathcal{R}_0, \mathcal{R}_i) \to \mathcal{R}_0$ $\textsc{using}$ $\mathcal{R}_5$..) polinomne složenosti iz propozicija 2.73 i 3.8 za svaki $i \in [1..k]$, te po prikazanim stanjima registara u vidimo da su svi registri veličine $O(n)$ u svakoj konfiguraciji vanjskog ($\textsc{bAdd}_k$)-izračunavanja, iz čega po teoremu 2.37 slijedi da je binmakro-program ($\textsc{bAdd}_k$) polinomne vremenske složenosti, a budući da je $k \in \mathbb{N}$ takav da je $k > 2$ bio proizvoljan, to vrijedi i za svaki takav $k$. □

**Napomena 3.13:** Potvrđeno testovima u programskom dijelu diplomskog rada te pomnim promatranjem izvršavanja koraka, u binmakro-algoritmu ($\textsc{bAdd}_k$), može se vidjeti da je ($\textsc{bAdd}_k$) vremenske složenosti $O(n^2)$. ◁

**Korolar 3.14:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Funkcija zbrajanja $add^k : \mathbb{N}^k \to \mathbb{N}$, definirana sa $add^k(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} x_i$, je polinomno BRAM-izračunljiva.

*Dokaz.* Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Iz propozicije 3.12 slijedi da binmakro-algoritam ($\textsc{bAdd}_k$) računa funkciju $add^k$, iz čega analognim razmišljanjem kao u korolaru 3.10 dolazimo do zaključka da je funkcija $add^k$ polinomno BRAM-izračunljiva i budući da je $k \in \mathbb{N}$ takav da je $k > 2$ bio proizvoljan, to vrijedi za svaki takav $k$. □

# Poglavlje 4

# Množenje

**Napomena 4.1:** U binmakro-algoritmu za množenje dva broja simulirat ćemo proceduru pisanog množenja u binarnom sustavu za brojeve $a, b \in \mathbb{N}$. Neka je $b = (b_m \cdots b_1 b_0)_2$. Procedura i rezultat pisanog množenja je $a \cdot b = \sum_{i=0}^{m} a \cdot 2^i \cdot b_i$. ◁

**Definicija 4.2:** Definirajmo binmakro-algoritam

$$(\text{BMUL}_2) \coloneqq \left[\begin{array}{l} 0.\ \text{BZ}\ \mathcal{R}_2, 9 \\ 1.\ \text{MOVE}\ \mathcal{R}_1\ \text{TO}\ \mathcal{R}_3\ \text{USING}\ \mathcal{R}_4 \\ 2.\ \text{ODD}\ \mathcal{R}_2, 4 \\ 3.\ \text{GO TO}\ 5 \\ 4.\ (\text{BADD2}(\mathcal{R}_0, \mathcal{R}_3) \to \mathcal{R}_0\ \text{USING}\ \mathcal{R}_5..) \\ 5.\ \text{SHR}\ \mathcal{R}_2 \\ 6.\ \text{SHL}\ \mathcal{R}_3 \\ 7.\ \text{BZ}\ \mathcal{R}_2, 9 \\ 8.\ \text{GO TO}\ 2 \end{array}\right]^2 . \tag{4.1}$$

◁

**Propozicija 4.3:** Binmakro-algoritam $\text{BMUL}_2$ za početnu konfiguraciju s proizvoljnim ulazom $(a, b) \in \mathbb{N}^2$ ima semantiku $r_0' = a \cdot b$. Također, vremenska složenost binmakro-algoritma $\text{BMUL}_2$ je polinomna.

*Dokaz.* Neka je $(a, b) \in \mathbb{N}^2$ proizvoljan. Neka je $b = (b_m \cdots b_1 b_0)_2$. Promotrimo vanjsko $(\text{BMUL}_2)$-izračunavanje s početnom konfiguracijom $c$ s ulazom $(a, b)$ po slučajevima za vrijednost $b$.

- $b = 0$

$$r_2 = b = 0 \to pc = 9;$$
$$r_0' = 0 = a \cdot 0 = a \cdot b.$$

- $b \neq 0$

$$r_1 = a \wedge r_2 = b = (b_m \cdots b_1 b_0)_2 \rightarrow pc = 1 \rightarrow r_3^{(2)} = r_1 = a.$$

Sad imamo grananje po parnosti $r_2^{(2)} = (b_m \cdots b_1 b_0)_2$, odnosno grananje po vrijedosti $b_0$. U slučaju da je $b_0 = 0$ vrijedi

$$r_2^{(2)} \bmod 2 = (b_m \cdots b_1 b_0)_2 \bmod 2 = b_0 = 0 \rightarrow pc = 3 \rightarrow pc = 5,$$
$$pc = 5 \rightarrow r_2^{(6)} = (b_m \cdots b_2 b_1)_2 \rightarrow r_1^{(7)} = a \cdot 2.$$

U slučaju da je $b_0 = 1$ vrijedi

$$r_2^{(2)} \bmod 2 = (b_m \cdots b_1 b_0)_2 \bmod 2 = b_0 = 1 \rightarrow pc = 4 \rightarrow r_0^{(5)} = r_0^{(4)} + r_3^{(4)} = a,$$
$$r_0^{(5)} = a \rightarrow r_2^{(6)} = (b_m \cdots b_2 b_1)_2 \rightarrow r_1^{(7)} = a \cdot 2.$$

U nastavku izračunavanja, vidimo da se izvršava petlja od 2. do 8. instrukcije dok se registar $\mathcal{R}_2$ ne isprazni, iz čega slijedi da petlja ima točno $m + 1$ iteracija, za svaku znamenku od $(b_m \cdots b_1 b_0)_2$. Induktivno se pokaže da će zadnja iteracija petlje izgledati

$$r_2^{(2)} \bmod 2 = (b_m)_2 \bmod 2 = b_m = 1 \rightarrow pc = 4,$$
$$pc = 4 \rightarrow r_0^{(5)} = r_0^{(4)} + r_3^{(4)} = \sum_{i=0}^{m-1} a \cdot 2^i \cdot b_i + a \cdot 2^m = \sum_{i=0}^{m} a \cdot 2^i \cdot b_i,$$
$$r_0^{(5)} = \sum_{i=0}^{m} a \cdot 2^i \cdot b_i \rightarrow r_2^{(6)} = 0 \rightarrow r_1^{(7)} = a \cdot 2^{m+1} \rightarrow pc = 9;$$
$$r_0' = \sum_{i=0}^{m} a \cdot 2^i \cdot b_i = a \cdot b$$

Slijedi tvrdnja za semantiku u propoziciji, posebno, binmakro-algoritam ($\text{BMUL}_2$) je totalan jer iz prikazanog vidimo da stane za sve ulaze $x \in \mathbb{N}^2$. Što se tiče vremenske složenosti binmakro-algoritma ($\text{BMUL}_2$), vidimo da je njegova vanjska složenost $O(n)$ za ulaz veličine $n$, budući da se u vanjskom izračunavanju petlja iterira $m$ puta i vrijedi $n \geq m$, te je broj vanjskih koraka van petlje konstantan. Veličina svakog registra tijekom vanjskog izračunavanja je uvijek $O(n)$ što se lako vidi iz prikazanih vanjskih prijelaza gore. Zatim, budući da su binmakroi MOVE $\mathcal{R}_1$ TO $\mathcal{R}_3$ USING $\mathcal{R}_4$ i ($\text{BADD2}(\mathcal{R}_0, \mathcal{R}_3) \rightarrow \mathcal{R}_0$ USING $\mathcal{R}_t$..) polinomne vremenske složenosti, po teoremu 2.37 slijedi da je binmakro-algoritam ($\text{BMUL}_2$) polinomne vremenske složenosti. □

**Napomena 4.4:** Potvrđeno testovima u programskom dijelu diplomskog rada te pomnim promatranjem izvršavanja koraka, u binmakro-algoritmu ($\text{BMUL}_2$), može se vidjeti da je ($\text{BMUL}_2$) vremenske složenosti $O(n^2)$. ◁

**Korolar 4.5:** Funkcija množenja $mul^2 : \mathbb{N}^2 \to \mathbb{N}$, definirana sa $mul^k(x_1, x_2) = x_1 \cdot x_2$, je polinomno BRAM-izračunljiva.

*Dokaz.* Iz propozicije 4.3 slijedi da binmakro-algoritam ($\textsc{bMul}_2$) računa funkciju $mul^2$, iz čega analognim razmišljanjem kao u korolaru 3.10 dolazimo do zaključka da je funkcija $mul^2$ polinomno BRAM-izračunljiva. □

**Definicija 4.6:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$ te neka je $t = 9$ ako je $k < 9$, inače neka je $t = k + 1$. Definirajmo binmakro-algoritam

$$(\textsc{bMul}_k) := \begin{bmatrix} 0.\ \textsc{shs}\ \mathcal{R}_0 \\ 1.\ (\textsc{bMul}_2(\mathcal{R}_0, \mathcal{R}_1) \to \mathcal{R}_0\ \textsc{using}\ \mathcal{R}_t..) \\ 2.\ (\textsc{bMul}_2(\mathcal{R}_0, \mathcal{R}_2) \to \mathcal{R}_0\ \textsc{using}\ \mathcal{R}_t..) \\ \vdots \\ k.\ (\textsc{bMul}_2(\mathcal{R}_0, \mathcal{R}_k) \to \mathcal{R}_0\ \textsc{using}\ \mathcal{R}_t..) \end{bmatrix}^k . \tag{4.2}$$

◁

**Propozicija 4.7:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Semantika binmakro-algoritma ($\textsc{bMul}_k$) za početnu konfiguraciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$ je $r'_0 = \prod_{i=1}^{k} r_i$.
Također, binmakro-algoritam ($\textsc{bMul}_k$) je polinomne vremenske složenosti.

*Dokaz.* Promotrimo semantiku kroz vanjsko ($\textsc{bMul}_k$)-izračunavanje za početnu konfiguraciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$. Vrijedi

$$\begin{gathered} (r_1, r_2, \cdots, r_k) = \vec{x} \to r_0^{(1)} = 1 \to r_0^{(2)} = r_0^{(1)} \cdot r_1^{(1)} = 1 \cdot x_1 = x_1, \\ r_0^{(2)} = x_1 \to r_0^{(3)} = r_0^{(2)} \cdot r_2^{(2)} = x_1 \cdot x_2, \\ \vdots \\ r_0^{(k)} = \prod_{i=1}^{k-1} x_i \to r_0^{(k+1)} = r_0^{(k)} \cdot x_k = \prod_{i=1}^{k-1} x_i \cdot x_k = \prod_{i=1}^{k} x_i; \\ r'_0 = \prod_{i=1}^{k} x_i. \end{gathered}$$

Vrijedi tvrdnja za semantiku iz propozicije, a posebno je da je binmakro-algoritam ($\textsc{bMul}_k$) totalan, jer iz priloženog vidimo da ($\textsc{bMul}_k$)-izračunavanje stane za početnu konfiguraciju s proizvoljnim ulazom $\vec{x} \in \mathbb{N}^k$. Što se tiče vremenske složenosti, vidimo da je binmakro-algoritam ($\textsc{bMul}_k$) vanjske vremenske složenosti $O(1)$, da je $(\textsc{bMul2}(\mathcal{R}_0, \mathcal{R}_i) \to \mathcal{R}_0\ \textsc{using}\ \mathcal{R}_t..)$ polinomne složenosti iz propozicija 2.73 i 4.3 za svaki $i \in [1..k]$, te po prikazanim stanjima registara vidimo da su svi registri veličine $O(n)$ u svakoj konfiguraciji vanjskog ($\textsc{bMul}_k$)-izračunavanja, iz čega po teoremu 2.37 slijedi da je binmakro-program ($\textsc{bMul}_k$) polinomne vremenske složenosti, a budući da je $k \in \mathbb{N}$ takav da je $k > 2$ bio proizvoljan, to vrijedi i za svaki takav $k$. □

**Napomena 4.8:** Potvrđeno testovima u programskom dijelu diplomskog rada te pomnim promatranjem izvršavanja koraka, u binmakro-algoritmu ($\textsc{bMul}_k$), može se vidjeti da je ($\textsc{bMul}_k$) vremenske složenosti $O(n^3)$. ◁

**Korolar 4.9:** Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Funkcija množenja $mul^k : \mathbb{N}^k \to \mathbb{N}$ definirana sa $mul^k(x_1, x_2, \ldots, x_k) = \prod_{i=1}^{k} x_i$, je polinomno BRAM-izračunljiva.

*Dokaz.* Neka je $k \in \mathbb{N}$ takav da je $k > 2$. Iz propozicije 4.7 slijedi da binmakro-algoritam ($\textsc{bMul}_k$) računa funkciju $mul^k$, iz čega analognim razmišljanjem kao u korolaru 3.10 dolazimo do zaključka da je funkcija $mul^k$ polinomno BRAM-izračunljiva i budući da je $k \in \mathbb{N}$ takav da je $k > 2$ bio proizvoljan, to vrijedi za svaki takav $k$. □

# Bibliografija

[1] V. Čačić, *Komputonomikon*, PMF-MO, Zagreb, 2020.

# Sažetak

U početku rada smo definirali binarni RAM-stroj i svojstvo binarne BRAM-izračunljivosti za totalne brojevne funkcije. Zatim smo definirali pojam vremenske složenosti za totalne BRAM-algoritme i za funkcije koje oni računaju. Zatim smo radi pojednostavljenja računanja definirali pojmove binmakro-stroj i vanjsku vremensku složenost kako bismo lakše dokazali polinomnu BRAM-izračunljivost totalnih brojevnih funkcija. Zatim smo pomoću postupka spljoštenja pokazali vezu između izračunavanja s binmakro-programima i BRAM-programima. Definiranjem makroa izgradili dovoljno složene programske cjeline kako bismo na kraju izgradili makro-algoritme polinomne vremenske složenosti čija spljoštenja računaju totalne brojevne funkcije zbrajanja i množenja za mjesnosti $k \geq 2$ iz čega slijedi zbog dokazanih tvrdnji da su one polinomno BRAM-izračunljive.

# Summary

In the beginning of this thesis we defined binary RAM-machine and the property of binary RAM-computability for total natural number functions. Then we defined the term time complexity for total BRAM-algorithms and also for the functions they compute. Then, for the sake of simplifying computation, we defined terms binmacro-machine and outer time complexity so we could more easily prove polynomial BRAM-computability of total natural number functions. Then with the help of flattening procedure we showed the connection between the computation with binmacro-programs and BRAM-programs. By defining macros, we built fairly complex programming units so we could build macro-algorithms of polynomial time complexity whose flattenings compute total number functions of addition and multiplication of arities $k \geq 2$ from which follows, with proven statements, that they are polynomially BRAM-computable.

# Životopis

Rođen sam 16.2.1995. u Zagrebu. Pohađao sam osnovnu školu Krune Krstića te gimnaziju Jurja Barakovića, opći smjer, u Zadru. 2013. godine upisujem preddiplomski sveučilišni studij Matematika na PMF-u koji uspješno završavam u 7. mjesecu 2017. godine. Zatim upisujem diplomski studij Računarstvo i matematika na matematičkom odsjeku PMF-a koji završavam s napisanim diplomskim radom o binarnim RAM-strojevima.
U slobodno vrijeme bavim se sportom i programiram honorarno.