Ekvivalencija determinističkog i nedeterminističkog Turingovog odlučivača

Nora Berdalović

21. kolovoza 2024.

1 Dokaz teorema

Teorem 1. Nedeterministički Turingov odlučivač sa svojstvom da svako njegovo izračunavanje stane u $q_\checkmark$ ili q_X ekvivalentan je deterističkom.

Dokaz. Za dani nedeterministički Turingov odlučivač $\mathcal{N}$, simuliramo njegov rad na deterističkom. Koristimo četiri glavne trake i eventualne pomoćne: kao što znamo, svaki TO s proizvoljnim konačnim brojem traka je ekvivalentan jednotračnom.

Rad TO ilustriramo sa stablom, gdje svaki čvor predstavlja jednu konfiguraciju. Djeca nekog čvora one su konfiguracije u koje je iz njega moguće prijeći. Korijen je početna konfiguracija $(q_0, 0, w\textvisiblespace\textvisiblespace...)$, za ulaznu riječ w.

Zbog toga što je broj stanja te broj znakova u abecedi konačan, znamo da svaki čvor ima gornju ogradu na broj djece. Taj indeks grananja označavamo s b. Dakle, imamo konačan broj prijelaza između konfiguracija nad kojima napravimo neki uređaj (npr. leksikografski) te ih označavamo redom s δ_1, ... δ_b. Adresa nekog čvora bit će niz prijelaza koje je potrebno izvršiti iz korijena nadalje kako bi se došlo do čvora.

Za razliku od DTO, koji će u svom radu prolaziti samo jednom granom stabla, NTO može prolazi svim granama simultano. Kako bi ga simulirali, DTO će također morati proći cijelim stablom.

Iz pretpostavke teorema znamo da će $\mathcal{N}$ završiti izračun u jednom od dva stanja $q_\checkmark$ ili q_X. Ako u simulaciji njegovog rada dođemo do $\mathcal{N}.q_\checkmark$, prihvaćamo riječ. Međutim, ako dođemo do $\mathcal{N}.q_X$, nemamo dovoljno informacija da znamo znači li to konačno odbijanje riječi, ili samo trenutne grane – moguće je da postoji druga grana gdje ćemo doći do $\mathcal{N}.q_\checkmark$. Zbog toga je

potrebno pratiti dosad obiđene čvorove i njihova stanja, jer tek kad dođemo do stanja $\mathcal{N}.q_X$ u svim granama stabla, znamo da DTO mora odbiti riječ.

Na prvu traku DTO zapisujemo ulaznu riječ i ostavljamo ju nepromjenjenu.

Drugu traku koristimo za simulaciju rada NTO u određenoj grani.

Na trećoj traci, koristeći adrese, pratimo čvorove koje smo dosad obišli te jesu li u stanju $\mathcal{N}.q_X$, tj. jesu li odbijeni ili ne. Odbijene adrese označavamo sa znakom X poslije njihovog zapisa. Između različitih adresa pišemo delimiter #. Stablo obilazimo sa breadth first search, pa će zapis na traci biti oblika $\delta_1\#\delta_2\#.....\delta_b\#\delta_1\delta_1X\#....\delta_1\delta_b\#\delta_2\delta_1...$

Na četvrtoj traci nalazi se trenutna adresa, koja je na početku δ_1 (prvo dijete korijena).

Algoritam, zapisan niže u pseudokodu, funkcionira na sljedeći način:

Nakon zapisivanja ulazne riječi na traku t_1, stroj provjerava poseban slučaj kad je izračun gotov već u korijenu stabla, odnosno početno stanje jednako je nekom završnom. Ako je to $\mathcal{N}.q_{\checkmark}$, prihvaća riječ, a ako je $\mathcal{N}.q_X$, odbija ju – znamo da stroj završava rad odmah na početku, pa nema daljnjih konfiguracija za provjeriti.

Ako to nije slučaj, stroj ulazi u petlju. Za trenutnu adresu, provjerava nalazi li se u grani koja je već odbijena. To postiže čitanjem zapisa na t_3, gdje se nalaze sve dosad prijeđene adrese. TO čita prvi prijelaz u trenutnoj adresi sa t_4 i sprema ga u varijablu `dio`. Nju prosljeđuje pomoćnoj funkciji findAddress(), koja čita traku t_3 i traži `dio` na njoj. Ako je on zapisan na traci, i idući pročitani simbol nakon njega je X, znači da je stroj granu u kojoj se adresa nalazi već odbio, kod nekog pretka čvora na trenutnoj adresi, pa ju nije potrebno dalje analizirati. Tad postavlja varijablu `gotovo` na 1, a inače nastavlja s radom, za sve početne komade adrese (početna dva prijelaza u adresi, početna tri itd.). Ako nijedan komad na t_3 ne slijedi X, znamo da grana u kojoj je trenutna adresa nije ranije bila odbijena.

Npr. za adresu $\delta_3\delta_1\delta_2\delta_1$, čitat će t_3 dok ne naiđe na adresu δ_3. Ako nije praćena sa X, nastavlja čitanje i isto radi za adresu $\delta_3\delta_1$, i zatim $\delta_3\delta_1\delta_2$. Cijelu adresu (sva četiri prijelaza) nije potrebno provjeravati, jer se sigurno ne nalazi na traci: svaki čvor stabla posjećujemo samo jednom.

Ako trenutna adresa nije odbijena, DTO na t_2 simulira rad $\mathcal{N}$ od početne konfiguracije do one koja je na toj adresi. To znači da će, s inicijalnom pozicijom glave na početku t_2, redom izvršavati prijelaze koji čine adresu, a koje može čitati s t_4. Adresu zatim zapisuje na t_3, tako što čita simbole na traci dok ne dođe do prve praznine, zapiše adresu i znak # koji će ju odvajati

od iduće, u slučaju da na kraju simulacije nismo u stanju $\mathcal{N}.q_X$. Ako jesmo, tad zapisuje $X\#$, označavajući da je ta grana odbijena. Naravno, ako čvor na toj adresi ne postoji (čvorovi mogu imati manje od b djece, ali stroj uvijek provjerava svih b mogućnosti), također uzima da je grana odbijena. S druge strane, ako smo na kraju u stanju $\mathcal{N}.q_{\checkmark}$, tad stroj odmah prihvaća riječ i završava s radom.

Nakon toga, stroj miče trenutnu adresu sa t_4, zapisuje adresu koja se nalazi u varijabli `dio`, te ju pretvara u njenog sljedbenika putem funkcije sljedeca(), uz eventualni dodatak znakova δ_1 na kraj kako bi ostali na istoj dubini stabla (koju pamtimo u varijabli `adrLength`). Unutar `dio` će biti zapisana ili cijela adresa - u tom slučaju na t_4 će sad stajati njezin sljedbenik - ili neki čvor prethodnik adrese koji je već odbijen, pri čemu ovim računom osiguravamo da se ne provjeravaju i sva ostala njegova djeca, već nastavljamo provjeru za iduću neodbijenu granu.

Funkcija sljedeca() vrti beskonačnu petlju. Unutar nje, prvo se pomiče do zadnjeg nepraznog znaka na t_4, tj. zadnjeg prijelaza δ_{zadnji} u adresi. Ako je on različit od δ_b, umjesto njega zapisuje $\delta_{zadnji+1}$ i vraća se iz funkcije. Inače zapisuje δ_1 i ulazi u sljedeću iteraciju petlje, sve dok ne dođe do prvog prijelaza u adresi. Ako je i on jednak δ_b, nakon što ga promijeni u δ_1 zapisuje još δ_1 na sam kraj adrese, te povećava `adrLength` - ušli smo na novu dubinu stabla.

Ako se duljina adrese povećala, stroj provjerava je li u situaciji da je svaka grana odbijena, pa zna da mora odbiti riječ. To može pamtiti u varijabli `kontrola`, koju na početku petlje stavlja na 1, te mijenja na 0 čim naiđe na granu koja nije još odbijena. Provjera se obavlja počevši od čvorova koji su djeca korijena, tj. na adresama $\delta_1, \delta_2, ...\delta_b$, te s glavom čitača na početnom mjestu trake t_3. Stroj čita traku dok ne naiđe na danu adresu uz pomoć findAddress(). U slučaju da ne naiđe na nju prije prvog praznog znaka, postavlja varijablu `kontrola` na 0 – znači da granu nije još ni obišao pa ju sigurno nije ni odbio. Ako je adresa na traci i idući pročitani znak iza nje je X, odmah zna da je grana odbijena, pa opet postavlja `kontrola` $= 0$ i završava provjeru. Ako nije, potrebno je provjeriti jesu li sva djeca čvora na danoj adresi bila odbijena, što bi značilo da je i on sam, tj. njegova cijela grana, odbijena. Stroj dakle u petlji za $i = 1, ...b$ rekurzivno provjerava je li čvor na adresi oblika `trenutna_adresa`δ_i odbijen. Čim nađe jedno dijete za koje se nije ušlo u stanje $\mathcal{N}.q_X$, zna da nije cijela grana odbijena, pa postavlja varijablu `kontrola` na 0 i završava provjeru.

Npr. za provjeru je li odbijena grana δ_1, ako na traci nije zapisano $\delta_1 X$,

stroj provjerava grane $\delta_1\delta_1$, $\delta_1\delta_2$, ... $\delta_1\delta_b$. Ako nije naišao ni na $\delta_1\delta_1 X$, provjerava $\delta_1\delta_1\delta_1$, ... $\delta_1\delta_1\delta_b$ itd.

Ako je poslije ovog računa ostalo `kontrola` $= 1$, znači da stroj nije naišao ni na jednu granu NTO u kojoj riječ nije odbijena. Došao je do stanja $\mathcal{N}.q_X$ u svim granama – može odbiti riječ.

Pseudokod:

```
write(t1, R, w);
if (q0==N.q✓) prihvati;
else if (q0==N.qX) odbij;
write (δ1, R, t4);
adrLength=1;
while True:
   adresa=read(t4, R);
   kontrola=1;
   gotovo=0;
   dio="";
   while True:
      getDio();
      if (dio==adresa) break;
      if(findAddress(dio)==1):
         gotovo=1;
         break;
   if (not gotovo):
      s=N.q0;
      while(read(t4, R)==(p, α, q, β, d))≠␣:
         if (read(t2, R)=α and s=p):
            s=q;
            write(t2, β, d);
   zapisi(adresa,s);
   if(s==N.q✓) prihvati;
   tempAdrLength=adrLength;
   isprazni t4; write(t4,dio,R);
   write(sljedeca(),R, t4);
   for(i=glava(t4), i<adrLength, i++):
      write(δ1, R, t4);
   if(tempAdrLength==adrLength):
```

```
      continue;
   for (i=1, i<=b, i++):
      // provjera za svaku početnu granu je li odbijena
      if (not kraj(δ_i)):
         kontrola=0;
         break;
   if (kontrola==1) odbij; //znači da su sve grane odbijene
```

Korištene funkcije:

```
getDio():
   δ_j=read(t_4);
   if(δ_j==␣) return;
   dio+=δ_j;
   return;


zapisi(adresa,s):
   do (read(t_3, R))
      until (read(t_3, R)==␣);
   move(t_3, L);
   write(adresa,t_3, R);
   if (s==𝒩.q_X):
      write(X, t_3, R);
   write(#, t_3, R);
   return;


sljedeca():
   headTemp=1;
   while True:
      while (read(t_4, L)==␣):
         headTemp=glava(t_4);
         δ_zadnji=read(t_4, L);
      if(headTemp!=0):  read(t_4, R);
      if (δ_zadnji ≠ δ_b):
         δ_zadnji=δ_zadnji+1;
         write(δ_zadnji+1, t_4, R);
```

```
         return;
      else:
         if(glava($t_4$>=0):
            write($\delta_1, t_4, L$);
         else:
            write ($\delta_1, t_4, L$);
            premotaj na kraj;
            write ($\delta_1, t_4, R$);
            adrLength++;
            return;


kraj(adresa):
   help=0;
   vrati $t_3$ na početak;
   find=findAddress(adresa);
   if (find==1):  return 1;
   if (find==0):
      for (i=1, i<b, i++):
         help=kraj(adresa$\delta_i$);
         if (!help):  break;
   return help;

findAddress(adresa):
   help=0;
   while True:
      temp=read($t_3, R$);
      if (help==0):
         if(temp==␣):  return 2;
         readT3+=temp;
         if (readT3==pocetni dio adrese):
            if(readT3==adresa):
               isprazni readT3;
               if(read($t_3, R$)==$X$):  return 1;
               else:  return 0;
         else if (temp==$X$):
            isprazni readT3;
            return 1;
```

```
   else:  help=1;
if(help==1):
   do (read(t3, R))
      until (read(t3)==#);
   isprazni readT3;
   help=0;
```
□

2 Implementacija

Implementacija pseudokoda iz dokaza rađena je u Processingu i dostupna na GitHubu. Rad TO isproban je na sljedećim jezicima:

1. "Parni palindromi" - riječi oblika ww^R, pri čemu w^R označava reverz od w, a w je riječ nad abecedom $\Sigma = \{a, b\}$. Primjeri riječi koje se nalaze u jeziku su "aa" (riječ će biti prihvaćena na adresi {7,9,0,2,4,6}) i "abba" (na adresi {7,9,8,10,1,3,5,2,4,6}). Riječ "ab" nije u jeziku te neće biti prihvaćena, no zbog toga što DTO prolazi cijelim stablom koje predstavlja rad NTO, ova provjera je dosta dugotrajna - riječ će biti odbijena tek na adrLength=6.

2. "Neparni palindromi" - riječi oblika wsw^R, pri čemu je s slovo iz abecede, nad abecedom $\Sigma = \{a, b\}$. Primjeri riječi koje se nalaze u jeziku su "a" (na adresi {0,6}) i "aba" (adresa {7,9,1,2,4,6}), dok se u jeziku ne nalaze "aa" i "ab" (obje riječi će biti odbijene na adrLength=6, opet uz dugotrajan rad).

3. Riječi oblika 0^n1^n, nad abecedom $\Sigma = \{0, 1\}$. Primjeri riječi u jeziku su "01" (adresa {1,2,3}) i "0011" (adresa {0,1,2,2,3}), dok u jeziku nisu "00" i "011" (odbijene na adrLength=4).

Odabir primjera vrši se unutar koda, promjenom varijable `primjer`. U kodu se također može promjeniti ulazna riječ, predstavljena nizom stringova u varijabli `input`. Prilikom pokretanja koda, iscrtava se četverotračni Turingov stroj iz dokaza. Njime se upravlja sljedećim tipkama:

1. ENTER: pokretanje stroja

2. TAB: pauziranje stroja

3. 1: postavlja frameRate na 1 (iscrtava se jedna slika po sekundi). Preporučeno za početak uporabe, za bolje praćenje događaja.

4. 3, 5, 6: postavlja frameRate na 30, 50 ili 60 (defaultni). Preporučeno za ubrzanje stroja.

5. RIGHT i LEFT: pauziraju stroj i omogućavaju kretanje desno/lijevo po trećoj traci.

Osim crtanja stroja, ispisuju se i vrijednosti šest pomoćnih varijabli iz koda:

1. `korak`: varijabla koje nema u pseudokodu, koristi se kao pomoć pri iscrtavanju promjena na stroju, tj. kako bi se promjene na ekranu javljale postupno umjesto sve odjednom. Poznavanjem njezine vrijednosti lakše se može pratiti trenutno mjesto u kodu.

2. `kontrola`: varijabla koja prati jesu li sve grane odbijene, u provjeri koja se događa u 14. koraku. Na početku glavne petlje ona se uvijek postavlja na 1, te se mijenja na 0 čim u navedenoj provjeri stroj naiđe na granu koja još nije odbijena.

3. `gotovo`: varijabla koja prati je li trenutna grana odbijena, u provjeri koja se događa u 8. koraku. Na početku glavne petlje postavlja se na 0 te se mijenja na 1 čim vidi da je grana odbijena, tj. da je zapis trenutne adrese ili nekog njezinog pretka na 3. traci praćen sa X.

4. `adrLength`: dubina na kojoj se nalazimo u stablu NTO, iliti duljina zadnje adrese zapisane na 4. traci. Pri svakom povećavanju ove varijable, tj. prelasku na iduću dubinu stabla, vrši se provjera jesu li sve grane odbijene u koraku 14.

5. `adresa`: trenutna adresa s kojom radimo, a koju stroj čita s 4. trake. Predstavljen je nizom stringova, a ispisuje se kao niz integera koji predstavljaju indekse elemenata u nizu deltaFunction, koji sadrži sve moguće prijelaze iz jedne konfiguracije stroja u drugu. Resetira se na prazan niz (*null*) na kraju svake iteracije glavne petlje.

6. `dio`: početni dio adrese sa 4. trake, za kojeg se u 8. koraku provjerava je li već odbijen (je li ga na 3. traci slijedi X), dok u 13. koraku služi za računanje sljedeće adrese s kojom stroj radi. Na kraju glavne petlje se resetira na prazan niz.

3 Izvori

1. M. Sipser, Introduction to the Theory of Computation, Third Edition, Cengage Learning, 2012.

2. M. Horvat, slajdovi s predavanja Interpretacije programa, dostupni na stranici kolegija.

3. Repozitorij na GitHubu: github.com/ring-bearer/ts