

Ekvivalencija determinističkog i nedeterminističkog Turingovog odlučivača

Nora Berdalović

8. kolovoza 2024.

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

Dokaz. Za dani nedeterministički Turingov odlučivač \mathcal{N} , simuliramo njegov rad na determinističkom. 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 $(q_0, 0, w_{\sqcup} \dots)$, 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 $\delta_1, \dots, \delta_b$. 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 obidene 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 $\delta_1\#\delta_2\#\dots\delta_b\#\delta_1\delta_1X\#\dots\delta_1\delta_b\#\delta_2\delta_1\dots$

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 prijedene 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 pamtitu u varijabli `odbijeno`, 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 $\delta_1, \delta_2, \dots, \delta_b$, 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 `odbijeno` 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 `odbijeno` = 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 $i = 1, \dots, b$ 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 `odbijeno` 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 $\delta_1 \delta_1, \delta_1 \delta_2, \dots, \delta_1 \delta_b$. Ako nije naišao ni na $\delta_1 \delta_1 X$, provjerava $\delta_1 \delta_1 \delta_1, \dots, \delta_1 \delta_1 \delta_b$ itd.

Ako je poslije ovog računa ostalo odbijeno = 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( $t_1, R, w$ );
if ( $q_0 == \mathcal{N}.q_{\checkmark}$ ) prihvati;
else if ( $q_0 == \mathcal{N}.q_X$ ) odbij;
write ( $\delta_1, R, t_4$ );
adrLength=1;
while True:
    adresa=read( $t_4, R$ );
    odbijeno=1;
    gotovo=0;
    dio="";
    while True:
        getDio();
        if (dio==adresa) break;
        if (findAddress(dio)==1):
            gotovo=1;
            break;
    if (not gotovo):
         $s = \mathcal{N}.q_0$ ;
        while(read( $t_4, R$ ) == ( $p, \alpha, q, \beta, d$ ))  $\neq \perp$ :
            if (read( $t_2, R$ ) =  $\alpha$  and  $s = p$ ):
                 $s = q$ ;
                write( $t_2, \beta, d$ );
    zapisi(adresa, s);
    if( $s == \mathcal{N}.q_{\checkmark}$ ) prihvati;
    tempAdrLength=adrLength;
    isprazni  $t_4$ ; write( $t_4, dio, R$ );
    write(sljedeca(),  $R, t_4$ );
    for(i=glava( $t_4$ ), i<adrLength, i++):
        write( $\delta_1, R, t_4$ );
    if(tempAdrLength==adrLength):
        continue;
    for (i=1, i<=b, i++):

```

```

// provjera za svaku početnu granu je li odbijena
if (not kraj( $\delta_i$ )):
    odbijeno=0;
    break;
if (odbijeno==1) odbij; //znači da su sve grane odbijene

```

Korištene funkcije:

```

getDio():
     $\delta_j$ =read( $t_4$ );
    if( $\delta_j$ == $\perp$ ) return;
    dio+= $\delta_j$ ;
    return;

```

```

zapisi(adresa,s):
    do (read( $t_3, R$ ))
        until (read( $t_3, R$ )== $\perp$ );
    move( $t_3, L$ );
    write(adresa, $t_3, R$ );
    if ( $s==\mathcal{N}.q_X$ ):
        write( $X, t_3, R$ );
    write( $\#, t_3, R$ );
    return;

```

```

sljedeca():
    headTemp=1;
    while True:
        while (read( $t_4, L$ )== $\perp$ ):
            headTemp=glava( $t_4$ );
             $\delta_{zadnji}$ =read( $t_4, L$ );
        if(headTemp!=0): read( $t_4, R$ );
        if ( $\delta_{zadnji} \neq \delta_b$ ):
             $\delta_{zadnji}=\delta_{zadnji+1}$ ;
            write( $\delta_{zadnji+1}, t_4, R$ );
            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( $t_3$ ,  $R$ ))  
    until (read( $t_3$ )==#);  
isprazni readT3;  
help=0;
```

□