Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Druga domaća zadaća iz predmeta "Uvod u teoriju računarstva"

Zadatak broj 2068

Zagreb, lipanj 2010.

Sadržaj

Zadatak

Zadatak broj 2068:

Konstruirati i programski simulirati Turingov stroj koji provjerava ispravnost XML dokumenta. Koristiti pojednostavljeni oblik dokumenta koji se sastoji od slobodnog niza znakova i tagova oblika <niz_slova> i </niz_slova>. Vrijede pravila:

1. Za svaki tag <tag> mora postojati odgovarajući zatvarajući tag oblika </tag>
2. Jedan tag, odgovarajući zatvarajući tag i sadržaj između njih čine jednu cjelinu
3. Unutar jedne cjeline može se nalaziti niz znakova ili niz novih cjelina

Turingov stroj staje ako je zadani XML dokument ispravan.

Uvod

Turingov stroj (u nastavku TS) je najopćenitiji poznati matematički model računanja pomoću kojeg je moguće opisati i rješiti vrlo komplicirane problemske zadatke. Osnovni model TS-a se sastoji od upravljačke jedinke, ulazne trake i glave za čitanje i pisanje. Upravljačka jedinka može poprimiti konačan broj stanja, ovisno o funkcijama prijelaza. Ulazna traka je omeđena s lijeve, a beskonačna s desne strane i podijeljena na ćelije u koje se zapisuje po jedan od konačnog broja znakova, opet ovisno o funkcijama prijelaza. Glava za čitanje i pisanje se miče od ćelije do ćelije, čita znakove u njima i, ovisno o odgovarajućoj funkciji prijelaza, zapisuje znak u pročitanu ćeliju i nakon toga se pomiče ulijevo ili udesno.

Opisani osnovni model može se, ovisno o potrebama konkretnog zadatka, proširiti na više načina. Moguće ga je proširiti tako da umjesto trake omeđene s jedne strane, a beskonačne s druge, imamo traku koja je beskonačna s obje strane. Moguće je koristiti i više ulaznih traka, naravno, svaka traka sa vlastitom glavom za čitanje i pisanje. Također je moguće imati više glava za čitanje i pisanje na jednoj traci, tako da glave rade praktički neovisno jedna o drugoj, oviseći samo o funkcijama prijelaza. Također je moguće ne micati glavu za čitanje i pisanje. Spomenuta proširenja TS-a je, naravno, moguće kombinirati.

Funkcije prijelaza su temelj svakog TS-a. Pomoću njih TS odlučuje u koje stanje upravljačka jedinka treba prijeći, koji znak treba zapisati na mjesto pročitanog znaka i u kojem se smjeru glava za čitanje i pisanje treba pomaknuti.

TS se formalno zadaje uređenom sedmorkom:

TS = (Q, Σ, Γ, δ, q0, B, F), gdje je:

- Q – konačan skup stanja
- Σ – konačan skup ulaznih znakova
- Γ – konačan skup znakova trake
- δ – funkcija prijelaza δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$, gdje L i R označavaju pomak glave u lijevo i desno
- q0 – početno stanje
- B – znak kojim se označava prazna ćelija
- F – skup prihvatljivih stanje

Osnovni model TS-a ima funkcije prijelaza oblika:

$\delta(q0, \alpha) = (q1, \beta, \lambda)$, gdje je:

- q0 - stanje u kojem se TS trenutno nalazi
- α - znak u ćeliji iznad koje se glava za čitanje i pisanje trenutno nalazi
- q1 - stanje u koje TS prelazi
- β - znak koji glava za čitanje i pisanje zapisuje na mjesto pročitanog znaka
- λ - smjer u kojem se glava za čitanje i pisanje pomiče, L za lijevo, R za desno

Kod definiranja TS-a, prije funkcija prijelaza potrebno je odrediti skup stanja, skup ulaznih znakova, skup znakova trake, početno stanje, skup prihvatljivih stanja i konačno, funkcije prijelaza.

Ovaj način definiranja TS-a je u praksi teško ostvariv, jer prilikom same izgradnje nerijetko je potrebno dodati neko novo stanje, neki novi znak trake i sl., tako da se gore spomenuti skupovi često proširuju.

Ostvarenje

TS prihvaća nizove oblika:

<abc>cba<ba>bc</ba></abc>

Da bi niz bio ispravan, prvi tag koji zatvaramo mora biti onaj kojeg smo posljednjeg otvorili. Nizovi znakova mogu se nalaziti samo unutar tagova.

TS radi tako da traži prvi zatvarajući tag (znak /) te se nakon toga vraća u lijevo po traci do prvog otvorenog tag-a (znak **<**) i zatim ih uspoređuje znak po znak. Ukoliko se tagovi poklapaju stroj nastavlja s radom, a u protivnom staje i niz se ne prihvaća. Pošto je potrebno uspoređivati znakove, za svaki znak koji se može naći unutar taga potrebno je posebno stanje. Zbog toga, ovaj model Turingovog stroja bit će ograničen na slova **a**, **b** i **c**, te znakove koji se koriste za tagove (**<, >**, /). Taj skup znakova je dovoljan da bi se demonstrirao rad stroja, dok je rješenje i dalje općenito. Dodavanje novih znakova samo povećava broj stanja i funkcija prijelaza koje su u suštini iste, osim što odgovaraju drugom znaku, dok princip rada stroja ostaje nepromijenjen.

Model Turingovog stroja:

TS *M* = TS = (Q, Σ, Γ, δ, q0, B, q13)
Q = { q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13 }
Σ = { a, b, c, <, >, ,/ }
Γ = { a, b, c, <, >, , /, $, #, _, *, B }

Na početku rada u krajnje lijevim ćelijama trake zapisan je niz znakova, a ostale ćelije su prazne. TS je stanju **q0**, a glava za čitanje i pisanje se nalazi na krajnjoj lijevoj ćeliji. Stroj u stanju **q0** pomiče glavu u desno dok ne dođe do znaka /, kojeg zamijeni sa znakom **$**, mijenja stanje u **q1** i miče glavu u lijevo ili dok ne dođe do prazne ćelije i promijeni stanje u **q 13**.

δ(q0,a)=(q0,a,R) δ(q0,b)=(q0,b,R) δ(q0,c)=(q0,c,R)
δ(q0,<)=(q0,<,R) δ (q0,>)=(q0,>,R) δ(q0,_)=(q0,_,R)
δ(q0,$)=(q0,$,R) δ(q0,/)=(q1,$,L) δ (q0,B)=(q13,B,L)

U stanju q1 glava se pomiče u lijevo te znak **<** mijenja u **$** i promijeni stanje u **q2**.

δ(q1,<)=(q2,$,L)

U stanju **q2** glava se nastavlja micati u lijevo do znaka **<** (prvog otvorenog tag-a) ili znaka # (prethodno promijenjenog znaka **a**, **b** ili **c**), znak **<** mijenja u ***** i prelazi u stanje **q3**.

δ(q2,a)=(q2,a,L) δ(q2,b)=(q2,b,L) δ(q2,c)=(q2,c,L)
δ(q2,_)=(q2,_,L) δ(q2,$)=(q2,$,L) δ(q2,#)=(q2,#,R)
δ(q2,>)=(q2,>,L) δ(q2,<)=(q3,*,R)

Glava se pomiče u desno i ovisno o sljedećem pročitanom znaku mijenja stanje. Taj pročitani znak mijenja u znak #. Svaki znak ima odgovarajuće stanje te tako uspoređujemo znakove u tagovima.

δ(q3,a)=(q4,#,R) **δ(q3,b)=(q5,#,R)** **δ(q3,c)=(q6,#,R)**
δ(q3,>)=(q7,#,R)

Stanja **q4**, **q5**, **q6** i **q7** pomiču glavu u desno do prvog znaka **$** i prelaze u stanja **q8**, **q9**, **q10** i **q11** koja služe za usporedbu. Stroj pamti prvi pročitani znak u tagu i uspoređuje ga sa prvim znakom u zatvarajućem tagu.

δ(q4,a)=(q4,a,R) **δ(q4,b)=(q4,b,R)** **δ(q4,c)=(q4,c,R)**
δ(q4,<)=(q4,<,R) **δ(q4,>)=(q4,>,R)** **δ(q4,_)=(q4,_,R)**
δ(q4,$)=(q8,$,R)

δ(q5,a)=(q5,a,R) **δ(q5,b)=(q5,b,R)** **δ(q5,c)=(q5,c,R)**
δ(q5,<)=(q5,<,R) **δ(q5,>)=(q5,>,R)** **δ(q5,_)=(q5,_,R)**
δ(q5,$)=(q9,$,R)

δ(q6,a)=(q6,a,R) **δ(q6,b)=(q6,b,R)** **δ(q6,c)=(q6,c,R)**
δ(q6,<)=(q6,<,R) **δ(q6,>)=(q6,>,R)** **δ(q6,_)=(q6,_,R)**
δ(q6,$)=(q10,$,R)

δ(q7,a)=(q7,a,R) **δ(q7,b)=(q7,b,R)** **δ(q7,c)=(q7,c,R)**
δ(q7,<)=(q7,<,R) **δ(q7,>)=(q7,>,R)** **δ(q7,_)=(q7,_,R)**
δ(q7,$)=(q11,$,R)

Svako od stanja **q8**, **q9**, **q10** i **q11** prelazi preko preostalih znakova **$** do prvog sljedećeg znaka. Ukoliko se taj znak poklapa sa znakom koji stanje „pamti“, mijenja ga u znak **$** te prelazi u stanje **q2** pri čemu se ponavlja proces usporedbe. Ako se znakovi ne poklapaju stroj staje i niz se ne prihvaća. Stanje **q11**, ako dođe do znaka **>** (znači da se tagovi poklapaju), mijenja ga u **_** i prelazi u stanje **q12**.

δ(q8,a)=(q2,$,L) **δ(q8,$)=(q8,$,R)**
δ(q9,b)=(q2,$,L) **δ(q9,$)=(q9,$,R)**
δ(q10,c)=(q2,$,L) **δ(q10,$)=(q10,$,R)**
δ(q11,>)=(q12,_,L) **δ(q11,$)=(q11,$,R)**

Stanje **q12** mijenja sve znakove u znak**_** dok ne dođe znaka *****, zamijeni ga sa znakom **_** i prelazi u stanje **q0** koje ponavlja cijeli proces.

δ(q12,a)=(q12,_,L) **δ(q12,b)=(q12,_,L)** **δ(q12,c)=(q12,_,L)**
δ(q12,<)=(q12,_,L) **δ(q12,>)=(q12,_,L)** **δ(q12,_)=(q12,_,L)**
δ(q12,$)=(q12,_,L) **δ(q12,#)=(q12,_,L)** **δ(q12,*)=(q0,_,R)**

Stanje **q13** je prihvatljivo stanje. U ovo stanje stroj prelazi kad dođe do prve prazne ćelije. Glava se pomiče u lijevo preko znakova **_** i ako dođe do krajnje lijeve stroj staje i niz se prihvaća. U slučaju da glava pročita bilo koji znak osim **_** , stroj staje i niz se ne prihvaća.

δ(q13,_)=(q13,_,L)

Programsko ostvarenje

Program simulator.py učitava dvije datoteke: definicija.txt, u kojoj su definirani prijelazi u obliku:

```
q0,b=q0,b,R
q0,c=q0,c,R
q0,<=q0,<,R
...
```

i datoteku ulaz.txt, u kojoj se nalazi niz kojeg treba obraditi:

```
<abc>cba<ba>bc</ba></abc>
```

Iz datoteke definicija.txt gradi rječnik na način da razdvoji jedan prijelaz na stranu desno od znaka "=" i na stranu lijevo od znaka "=". Ključ rječnika je desna strana, dok je lijeva vrijednost tog ključa.

Potom se učitava ulazni niz koji se sprema u listu te se ulazi u while petlju s uvjetom da je prijelaz za trenutno stanje i učitani znak definiran u rječniku. U petlji se pristupa rječniku pomoću ključa (trenutno stanje + "," + učitani znak) te se određuje novo stanje, znak koji se upisuje na traku i smjer kretanja glave. Glava se pomiče te se provjerava da li je nova kombinacija stanja i učitanog znaka definirana. Ukoliko nije, stroj staje s radom i niz se ne prihvaća. Ukoliko je, stroj nastavlja s radom.

Stroj prestaje s radom i niz se prihvaća kada u stanju **q13** dođe do krajnje lijeve ćelije a da pritom ne nađe znak različit od znaka _.

Program nudi tri načina prikaza rada stroja; ispis cijelog procesa odjednom, korak po korak (koji se ostvaruje pritiskom tipke ENTER) ili ispis korak po korak samo nakon promjene stanja.

Primjer izvođenja (neprihvatljiv niz)

```
<ab>bb</ac>B
^

Odaberite nacin izvodjenja:

1: Odjednom
2: Korak po korak
3: Nakon promjene stanja

==> 3
Novo stanje: q1 Glava: <
<ab>bb<$ac>B
      ^

Novo stanje: q2 Glava: b
<ab>bb$$ac>B
     ^

Novo stanje: q3 Glava: a
*ab>bb$$ac>B
 ^

Novo stanje: q4 Glava: b
*#b>bb$$ac>B
  ^

Novo stanje: q8 Glava: $
*#b>bb$$ac>B
       ^

Novo stanje: q2 Glava: $
*#b>bb$$$c>B
       ^

Novo stanje: q3 Glava: b
*#b>bb$$$c>B
  ^

Novo stanje: q5 Glava: >
*##>bb$$$c>B
   ^

Novo stanje: q9 Glava: $
*##>bb$$$c>B
       ^


Dokument nije ispravan
```

Zaključak

Ovaj Turingov stroj provjerava ispravnost XML dokumenta oblika <niz_slova>niz</niz_slova>. Iako je ulazna abeceda ograničena na znakove znakove **a**, **b** i **c**, to ne smanjuje općenitost rješenja. Princip rada TS bio bi jednak i da ulazna abeceda sadrži sva slova, ali bi se time povećao broj stanja i funkcija prijelaza koje se koriste za uspoređivanje znakova. U tim stanjima TS bi se ponašao jednako kao ovaj TS u stanjima **q4-q11** i zbog toga je jednostavnije pratiti rad ovog pojednostavljenog TS-a.