Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Mile Balenović

Druga domaća zadaća iz predmeta „Uvod u teoriju računarstva“

Zadatak broj 3017

Zagreb, lipanj, 2010.

Balenović Mile

Druga domaća zadaća iz predmeta „Uvod u teoriju računarstva“

Student: Mile Balenović

Matični broj studenta: 0036446666

Zadatak broj 3017: U nekom programskom jeziku implementirati algoritme za pojednostavljenje gramatike (izbacivanje mrtvih znakova, izbacivanje nedohvatljivih znakova, izbacivanje ε – produkcija i izbacivanje jediničnih produkcija). Format zapisa gramatike u ulaznoj datoteci je proizvoljan. Pokazati izvođenje na proizvoljnoj gramatici.

Balenović Mile

Sadržaj

Uvod 1

Ostvarenje 2

Odbacivanje mrtvih znakova 3

1.1 Postupak rješavanja. 3

1.2 Programsko ostvarenje odbacivanja mrtvih znakova 4

Odbacivanje nedohvatljivih znakova 5

1.3 Postupak rješavanja. 5

1.4 Programsko ostvarenje odbacivanja nedohvatljivih znakova 6

Odbacivanje ε – produkcija 7

1.5 Postupak rješavanja. 7

1.6 Programsko ostvarenje odbacivanja ε – produkcija 8

Odbacivanje jediničnih produkcija 9

1.7 Postupak rješavanja. 9

1.8 Programsko ostvarenje odbacivanja jediničnih produkcija 10

Upute za rukovanje programom 11

Zaključak 12

Literatura 13

Balenović Mile

Uvod

Postupci pojednostavljenja gramatike odbacuju beskorisne znakove (mrtve i nedohvatljive znakove) i produkcije (jedinične i ε – produkcije). To su sljedeći postupci i obavljaju se navedenim redoslijedom:

1. Odbacivanje mrtvih znakova
2. Odbacivanje nedohvatljivih znakova
3. Odbacivanje ε – produkcija
4. Odbacivanje jediničnih produkcija

Ostvario sam navedeno u C++ okruženju. Uz izvršni program i izvorni kod nalaze se i dvije .txt datoteke, ulaz.txt koja služi za definiranje gramatike i izlaz.txt u koju se pohranjuje pojednostavljena gramatika.

U sljedećih nekoliko stranica ću pobliže pojasniti način na koji se pojednostavljuje gramatika, prikazati to na svojoj gramatici, objasniti način ostvarenja programskog koda i upute za rukovanje.

Ostvarenje

Kako moram pokazati izvođenje na proizvoljnoj gramatici, uzeo sam kontekstno neovisnu gramatiku iz zadataka za vježbu:

$$G = (\{S, A, B, C, D, E\}, \{a, b, c, d, e, \varepsilon\}, P, S)$$

S produkcijama P:

$S \rightarrow bAbE$	$B \rightarrow DC$	$C \rightarrow cDAaB$
$S \rightarrow aABc$	$B \rightarrow ad$	$C \rightarrow bDaE$
$A \rightarrow beA$	$C \rightarrow eA$	$E \rightarrow ed$
$A \rightarrow \varepsilon$	$C \rightarrow \varepsilon$	$E \rightarrow ac$
$A \rightarrow B$		

Kratko ponavljanje kontekstno neovisne gramatike

Gramatika ima oblik

$$G = (V, T, P, S)$$

Pri čemu je:

1. V konačan skup nezavršnih znakova
2. T konačan skup završnih znakova,
3. P konačan skup produkcija
4. S početni nezavršni znak.

Nezavršni znakovi su označeni velikim slovima i ne smiju stajati na kraju obrađenog niza, završni znakovi malim slovima i morali bi to činiti. Lista produkcija nam govori što će se dogoditi za određeni ulazni znak iz niza dok je početni znak, kao što samo ime kaže, znak s kojim sve počinje.

Odbacivanje mrtvih znakova

Neka kontekstno neovisna gramatika G = (V, T, P, S) generira neprazni jezik, tj. L (G) ≠ Ø. Moguće je izraditi istovjetnu gramatiku[2] G' = (V',T',P',S) koja nema mrtvih znakova, odnosno za bilo koji A ∈ V' vrijedi A => w, w ∈ T*.

Iako odbacujemo mrtve znakove, nećemo tražiti njih nego ćemo učiniti suprotno, naći čemo sve žive znakove. Svi znakovi koji se neće nalaziti u listi živih znakova su mrtvi znakovi i njih ćemo odbaciti.

Algoritam za traženje živih znakova:

1. U listu živih znakova stave se lijeve strane produkcije koje na desnoj strani nemaju nezavršnih znakova.
2. Ako su na desnoj strani produkcije isključivo živi znakovi, onda se nezavršni znak lijeve strane produkcije doda u listu živih znakova.
3. Ako nije moguće proširiti listu živih znakova, onda se algoritam zaustavlja. Svi znakovi koji nisu u listi živih znakova su mrtvi znakovi.

1.1 Postupak rješavanja.

S → bAbE	B → DC	C → cDAaB
S → aABc	B → ad	C → bDaE
A → beA	C → eA	E → ed
A → ε	C → ε	E → ac
A → B		

Listu živih znakova za sad čine {a, b, c, d, e, ε }, tj. svi završni znakovi.

Prvim prolaskom kroz produkcije vidimo da imamo jos 4 živa znak jer s desne strane imaju samo žive znakove.

S → bAbE	B → DC	C → cDAaB
S → aABc	**B → ad**	C → bDaE
A → beA	C → eA	**E → ed**

A → ε	**C → ε**	**E → ac**
A → B		

Lista živih znakova {a, b, c, d, e, ε, A, B, C, E }. Sad kad se naša lista malo proširila, pretražujemo dalje.

S → bAbE	B → DC	C → cDAaB
S → aABc	**B → ad**	C → bDaE
A → beA	**C → eA**	**E → ed**
A → ε	**C → ε**	**E → ac**
A → B		

U listu živih znakova dodajemo još S {a, b, c, d, e, ε, A, B, C, E, S }. Vidimo da više nema isključivo živih znakova s desne strane i time smo gotovi s listom živih znakova. Lista mrtvih znakova je {D} i možemo odbaciti sve produkcije gdje se nalazi znak D neovisno o tome s koje strane produkcije se nalazi.

Lista produkcija nam nakon odbacivanja mrtvih znakova izgleda ovako:

S → bAbE	B → ad	E → ed
S → aABc	C → eA	E → ac
A → beA	C → ε	
A → ε		
A → B		

1.2 Programsko ostvarenje odbacivanja mrtvih znakova

U for petlji u polje živih znakova se pohranjuju završni znakovi. Nakon što se pohrane završni znakovi, dok je varijabla koja nam govori može li se lista živih znakova proširiti u 1, vrte se petlje koje prolaze kroz svaku produkciju posebno, provjeravaju nalaze li se desni znakovi u listi živih znakova. Ako su svi živi, onda dodaje lijevi znak u listu živih znakova. Kada se lista živih znakova više ne može proširiti, prekida se while petlja U sljedećoj petlji se u polje spremaju mrtvi znakovi (završni znakovi koji nisu u polju živih) te s dvije for petlje stvaram matricu u kojoj na produkcije koje sadrže mrtve znakove pohranjujem space. Na kraju se poziva funkcija u kojoj se nalazi petlja koja briše sve označene produkcije.

Odbacivanje nedohvatljivih znakova

Moguće je izraditi gramatiku G' = (V',T',P',S) koja je istovjetna kontekstno neovisnoj gramatici G=(V, T, P, S) i koja nema nedohvatljivih znakova, odnosno za bilo koji $X \in V' \cup T'$ vrijedi $S \Rightarrow \alpha X \beta$, gdje je $\alpha, \beta \in (V' \cup T')^*$.

Vodit ćemo se istom logikom kao i kod izbacivanja mrtvih znakova, naći ćemo dohvatljive znakove i oni koji se ne nalaze u toj listi su nedohvatljivi znakovi. Treba primjetiti da se ovo primjenjuje samo na nezavršnim znakovima.

Koraci algoritma za traženje nedohvatljivih znakova:

1. U listu dohvatljivih znakova stavi se početni nezavršni znak gramatike
2. Ako je znak s lijeve strane produkcije u listi dohvatljivih znakova, onda se svi znakovi desne strane produkcije dodaju u listu dohvatljivih znakova.
3. Ako listu dohvatljivih znakova nije moguće proširiti, onda se algoritam zaustavlja. Svi znakovi koji nisu u listi dohvatljivih znakova su nedohvatljivi.

1.3 Postupak rješavanja.

Lista produkcija nam nakon izbacivanja mrtvih znakova izgleda ovako

S → bAbE	B → ad	E → ed
S → aABc	C → eA	E → ac
A → beA	C → ε	
A → ε		
A → B		

Lista dohvatljivih znakova je {S}.

S → b**A**b**E**	B → ad	E → ed
S → a**AB**c	C → eA	E → ac
A → beA	C → ε	
A → ε		
A → B		

U listu dodajemo A, B i E pa nam lista sada izgleda {S, A, B, E}. Vidimo da više nema s desne strane A, B i E novih nezavršnih znakova i time je naša lista dohvatljivih znakova gotova. Listu nedohvatljivih znakova je {C}. Možemo odbaciti produkcije znaka C i sada lista produkcija izgleda ovako:

S → bAbE	A → ε	E → ed
S → aABc	A → B	E → ac
A → beA	B → ad	

1.4 Programsko ostvarenje odbacivanja nedohvatljivih znakova

U polje dohvatljivih se sprema početni znak S. Analogno prethodnom algoritmu, vrti se while petlja dok je varijabla koja nam govori može li se lista dohvatljivih znakova proširiti u 1, a u njoj se nalaze petlje koje prolaze kroz svaku produkciju i ako se s lijeve strane nalazi dohvatljivi znak, nezavršni znak/znakovi se dodaju u polje. Kada se više ne može proširiti lista dohvatljivih, prekida se while petlja i pokreće nova petlja koja stvara polje nedohvatljivih znakova na temelju polja dohvatljivih nezavršnih znakova. Kao i u prethodnom slučaju, u 2 petlje označavaju se produkcije s nedohvatljivim znakovima i opet se poziva funkcija koja odbacuje označene produkcije.

Odbacivanje ε – produkcija

Neka gramatika G = (V, T, P, S) generira kontekstno neovisni jezik L (G) - {ε}. Moguće je izraditi istovjetnu gramatiku G = (V', T', P', S) koja nema ε-produkcija.

Algoritam odbacivanja ε – produkcija se izvodi u 2 koraka:

1. Pronađu se svi nezavršni znakovi koji generiraju prazni niz. Prazni znakovi su oni znakovi za koje vrijedi A => ε. Prazni znakovi traže se slijedećim iterativnim postupkom: U listu praznih znakova stave se lijeve strane svih ε – produkcija. Ako su svi znakovi desne strane produkcije zapisani u listu praznih znakova, onda se lista praznih znakova nadopuni lijevom stranom te produkcije. Algoritam se nastavlja sve dok je moguće proširiti listu praznih znakova novim nezavršnim znakom.

2. Skup produkcija gramatike G' gradi se na sljedeći način. Ako je:
   $A \rightarrow X_1X_2 \ldots X_n$
   Produkcija gramatike G, onda se u skup produkcija gramatike G' dodaju produkcije oblika:
   $A \rightarrow \xi_1 \xi_2 \ldots \xi_n$
   Oznake ξ_i poprimaju slijedeće vrijednosti:

   a) Ako znak X_i nije prazni znak, onda je oznaka ξ_i jednaka X_i.
   b) Ako je znak X_i prazni znak, onda je oznaka ξ_i jednaka ε ili X_i.

   Produkcije se grade na temelju svih mogućih kombinacija oznaka $\xi_1 \xi_2 \ldots \xi_n$. Ako sve oznake ξ_i poprime vrijednost ε, onda nastaje ε -produkcija i ona se ne dodaje u skup produkcija gramatike G'. Time se odbace sve ε -produkcije iz zadane gramatike G.

1.5 Postupak rješavanja.

S → bAbE	A → ε	E → ed
S → aABc	A → B	E → ac
A → beA	B → ad	

Imamo jednu epsilon produkciju A → ε. Označiti ćemo ju s A_1. Nakon toga ćemo u svim produkcijama koje s desne strane imaju znak A razdijeliti produkcije stvarajući nove tako da prikažemo sve moguće kombinacije starih i novih znakova pa imamo:

$S \rightarrow bA_1bE$ $A_1 \rightarrow \varepsilon$ $E \rightarrow ed$

$S \rightarrow bAbE$ $A \rightarrow B$ $E \rightarrow ac$

$S \rightarrow aA_1Bc$ $B \rightarrow ad$

$S \rightarrow aABc$

$A \rightarrow beA_1$

$A \rightarrow beA$

Sad možemo odbaciti ε-produkcije i sve znakove koji predstavljaju ε-produkcije možemo zamijeniti praznim nizom (tj. pobrisati).

$S \rightarrow bbE$ $A \rightarrow be$ $E \rightarrow ed$

$S \rightarrow bAbE$ $A \rightarrow B$ $E \rightarrow ac$

$S \rightarrow aBc$ $B \rightarrow ad$

$S \rightarrow aABc$ $A \rightarrow beA$

Vidimo da nema novonastalih ε-produkcija pa smo time gotovi s izbacivanjem ε-produkcija. U slučaju da smo dobili nove ε-produkcije, proveli bi algoritam onoliko puta dok se ne riješimo svih ε-produkcija.

1.6 Programsko ostvarenje odbacivanja ε – produkcija

Programsko ostvarenje mi se malo razlikuje od ručnog rješavanja. Petlja se vrti dok postoji ε – produkcija (iz razloga što nakon odbacivanja ε – produkcije može nastati neka nova ε – produkcija). Unutar petlje se vrte sve produkcije i pomoću funkcije provjerava je li to ε – produkcija. Produkcije koje s desne strane imaju nezavršni znak koji ide u ε se kopiraju, ali za razliku od ručnog rješavanja, odmah se briše znak koji vodi u ε i time dobijemo dva para produkcija(jedna ostaje ista, a u drugoj je pobrisan znak s lijeve strane ε – produkcije. Nakon toga se označavaju ε-produkcije i brišu pomoću funkcije .

Odbacivanje jediničnih produkcija

Neka gramatika G = (V, T, P, S) generira kontekstno neovisni jezik L(G)\{ ε}. Moguće je izgraditi istovjetnu gramatiku G = (V', T', P', S) koja nema jediničnih produkcija oblika A → B.

Istovjetna gramatika G' koja nema jediničnih produkcija gradi se na sljedeći način:

1. U skup produkcija gramatike G' stave se sve produkcije gramatike G koje nisu jedinične.

2. Neka se postupkom generiranja iz nezavršnog znaka A dobije nezavršni znak B, tj. A => B, gdje su A i B nezavršni znakovi gramatike G. Za sve produkcije B → α koje nisu jedinične grade se nove produkcije A → α.

1.7 Postupak rješavanja.

S → bbE	A → be	E → ed
S → bAbE	A → B	E → ac
S → aBc	B → ad	
S → aABc	A → beA	

Imamo samo jednu jediničnu produkciju, A → B. Moramo staviti sve produkcije znaka B pod znak A.

S → bbE	A → be	E → ed
S → bAbE	B → ad	E → ac
S → aBc	A → beA	**A → ad**
S → aABc		

Nakon izbacivanja jediničnih produkcija potrebno je pregledati listu produkcija sadrži li nedohvatljive znakove jer se često dogodi da su znakovi kojima smo prebacili jediničnu produkciju postali nedohvatljivi (nije slučaj u našem primjeru).

1.8 Programsko ostvarenje odbacivanja jediničnih produkcija

Kao i kod izbacivanja ε – produkcija, imamo petlju koja se vrti sve dok postoji jedinična projekcija što se provjerava funkcijom. Opet se u petljama vrte sve produkcije i pronalaze one oblike A → B (jedinične produkcije). Nakon što je pronađena jedinična produkcija, u petlji se traže sve produkcije s lijevom stranom jednakom desnoj strani jedinične produkcije kako bi ih se moglo zamijenit. U petlji se spremaju nove, produkcije kojima je 'naljepljena' desna strana na lijevi dio jedinične produkcije, a jedinične označavaju i pozivom funkcije brišu. Moguće je da su nastale produkcije gdje je s obje strane isti nezavršni znak pa se pozivom funkcije brišu takve produkcije. Na kraju koda se nalazi poziv na funkciju za odbacivanje nedohvatljivih znakova koji su mogli naknadno nastati.

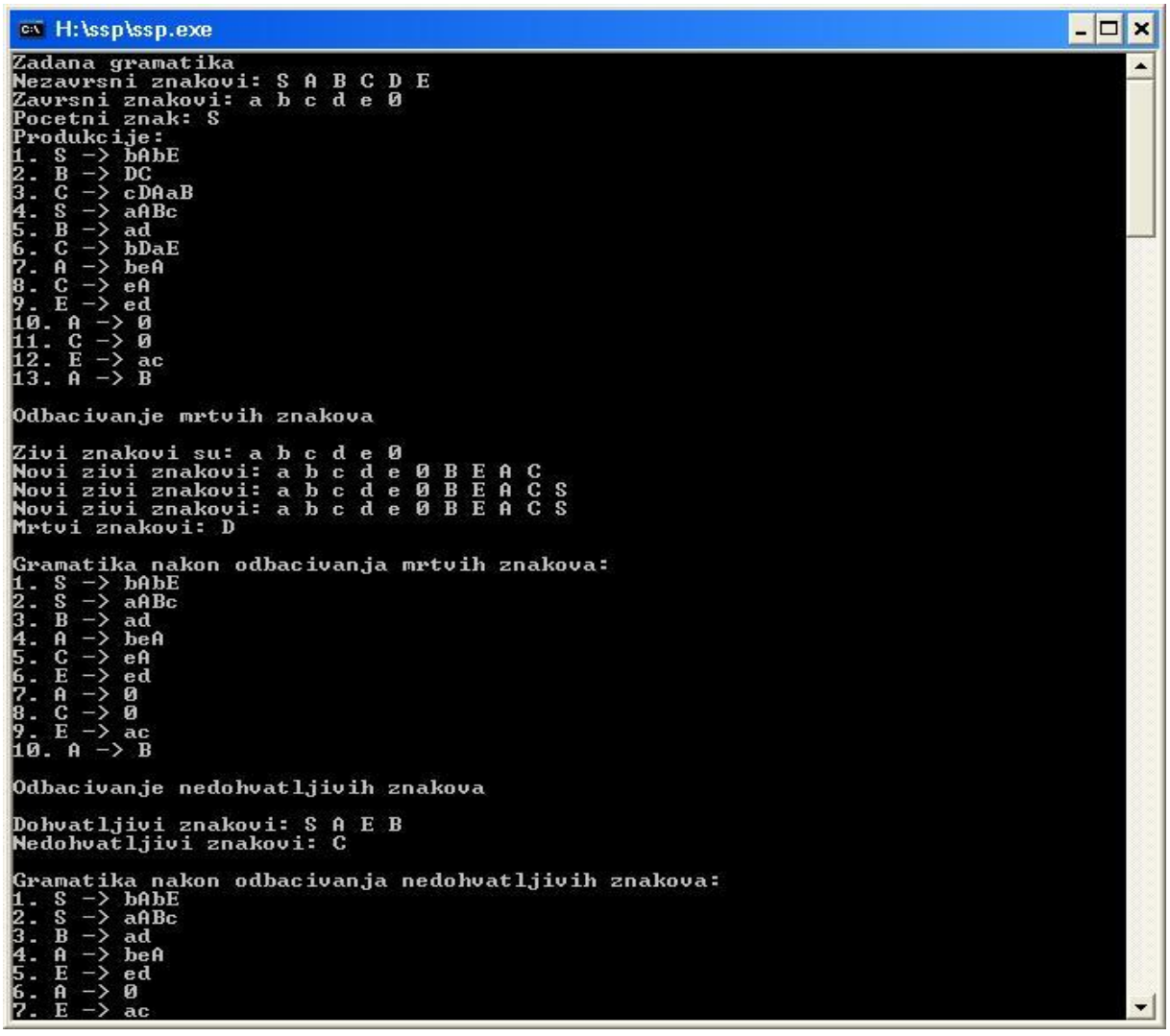
Upute za rukovanje programom

Prije pokretanja ssp.exe, potrebno je unijeti gramatiku u ulaz.txt u obliku(primjer je uploadan s izvršnim programom):

{Nezavršni znakovi međusobno odijeljeni zarezom}
{Završni znakovi međusobno odijeljeni zarezom}
{Početni znak}
Produkcije (svaka u drugome redu)

Važno je napomenuti da se znak ε unosi kao 0 zbog nemogućnosti korištenja znaka ε u C++.

Ovako bi trebao izgledati pokrenuti program ako je ispravno unijeta definicija gramatike:



Zaključak

U ovom seminarskom radu opisan je način na koji se pojednostavljuje gramatika. Razumijevanje pojednostavljenja gramatike i samo ručno rješavanje je bilo lagano budući da sam dobro savladao taj dio gradiva na predavanjima i nadam se da sam dovoljno dobro objasnio te kada bi netko tko ovo prvi put vidi znao riješavati slične zadatke uz pomoć mog seminarskog rada. Prebacivanje tog znanja u C++ je bilo malo kompleksnije, no uvelike je olakšala posao sličnost algoritama za izbacivanje mrtvih i nedohvatljivih znakova te algoritama za izbacivanje ε i jediničnih produkcija. Jedini pravi problem je bilo ubaciti ε u program što nisam uspio, pa je ε zamijenjen 0.

Literatura

1. Siniša Srbljić: Jezični procesori 1, 2. izdanje, Zagreb 2002.
2. Tutorialza2mi.pdf
3. ZadacizavježbuC2.pdf