Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Shefinho

Seminarski rad iz predmeta Prevođenje programskih jezika

Zadatak broj 62

Zagreb, siječanj 2009.

Seminarski rad iz predmeta Prevođenje programskih jezika

Student: Shefinho

Matični broj studenta: 0036xxxxxx

Zadatak broj 62 (teža inačica, 14 bodova):

Izgraditi gramatiku koja generira jezik u kojem su svi pravilno napisani logički izrazi izgrađeni pomoću operatora, (,), ~ (NOT), * (AND) i + (OR). Na temelju izgrađene gramatike programski ostvariti jezični procesor koji pretvara pravilno napisane izraze u troadresne naredbe sa simboličkim registrima. Primjer ulaznog logičkog izraza je:

~(T+F*T)+(~F*T)

(značenje: NOT(*true* OR (*false* AND *true*)) OR (NOT(*false*) AND *true*))

Napomena: operandi mogu biti isključivo znakovi F (*false*) i T (*true*). Pretpostaviti standardne prioritete operatora. Logičke konstante ostvariti pomoću troadresnih naredbi za oduzimanje i množenje cijelih brojeva.

Sadržaj:

1. Uvod 3

2. Ostvarenje 4

2.1. Opis 4

2.2. Formalna definicija 6

3. Programsko ostvarenje 8

4. Kratke upute 9

5. Zaključak..... 10

1. Uvod

Logički izrazi pojavljuju se u gotovo svim programskim jezicima. Njihova evaluacija temelji se na principima matematičke logike. Osnovne logičke operacije su negacija, konjunkcija i disjunkcija, koje se i pojavljuju u ovom primjeru. Sve složenije operacije grade se pomoću tih triju osnovnih.

Generiranje troadresnog međukoda je jedna od zadnjih faza rada jezičnog procesora prije one konačne, generiranja ciljnog programa. Ime dolazi od činjenice da se radi sa 3 adrese: adresom lijevog operanda, adresom desnog operanda i adresom rezultata. Registri su simbolički i prilikom generiranja koda koristi ih se proizvoljno mnogo. Svaki međurezultat sprema se u novi simbolički registar. Izradom grafa zavisnosti i primjenom odgovarajućih algoritama moguće je svesti broj potrebnih registara na minimum i/ili svesti veličinu samog ciljnog programa na minimum, te time generirati učinkovit ciljni program. No to nije zadatak ovog samostalnog studentskog projekta.

U ovom zadatku pojavljuju se, nazovimo ih tako, i "jednoadresne" i "dvoadresne" naredbe. "Jednoadresne" su kad se registru pridružuje konstanta. "Dvoadresne" su kad se obavlja operacija oduzimanja ili množenja s lijevim operandom kao registrom, a desnim operandom kao konstantom.

2. Ostvarenje

2.1. Opis

Kao i kod svakog jezičnog procesora, i u ovom primjeru pojavljuju se tri faze analize izvornog programa: leksička, sintaksna i semantička analiza. Na temelju semantičke analize radi se prva faza sinteze - generiranje troadresnog međukoda.

Leksička analiza u ovom slučaju je trivijalna. Provjerava se jesu li svi znakovi ulaznog niza elementi skupa završnih znakova T = {'T', 'F', '(', ')', '~', '*', '+'}. Ukoliko jesu, ulazni niz je leksički ispravan, u protivnom nije.

Sintaksna analiza radi se tehnikom rekurzivnog spusta na temelju produkcija kontekstno neovisne LL(1) gramatike. Pritom se gradi sintaksno stablo.

Semantička analiza temelji se na atributnoj prijevodnoj gramatici, koja je proširenje izgrađene gramatike izlaznim završnim znakovima. Svaki izlazni završni znak odgovara jednoj semantičkoj akciji. Semantičke akcije su: pridruživanje konstante 1 registru, pridruživanje konstante 0 registru, disjunkcija, konjunkcija i negacija.

Svaki izlazni završni znak ima 3 svojstva: adresa lijevog operanda, adresa desnog operanda i adresa rezultata. U slučaju da se radi pridruživanje, prva dva svojstva imaju vrijednost 0, budući da se radi o "jednoadresnoj" naredbi. U slučaju da se radi negacija, vrijednost drugog svojstva je 0, budući da se radi o "dvoadresnoj" naredbi.

Formalni opis gramatike i princip programskom ostvarenja nalaze se u daljnjem tekstu.

U troadresnom međukodu generira se 5 vrsta naredbi: pridruživanje konstante 1 registru, pridruživanje konstante 0 registru, množenje dva registra, množenje registra i konstante -1 i oduzimanje registra i konstante 1.

Evidentno je da su prve dvije semantičke akcije i prve dvije vrste troadresnih naredbi identične. Stoga se dotične dvije semantičke akcije ostvaruju sljedećim oblicima naredbi troadresnog međukoda:

Pridruzi0(0,0,i): $R_i := 0$

Pridruzi1(0,0,i): $R_i := 1$

Slijedi opis načina ostvarenja preostalih semantičkih akcija: konjunkcija, negacija i disjunkcija, navedenim redoslijedom.

Budući da se radi isključivo s konstantama 0 i 1, operacija konjunkcije ekvivalentna je operaciji množenja: 0 AND 0 = 0 = 0*0, 0 AND 1 = 0 = 0*1, 1 AND 1 = 1 = 1 * 1. Stoga se svaka pojava semantičke akcije Konjunkcija(i,j,k) ostvaruje sljedećim oblikom naredbe troadresnog međukoda:

$R_k := R_i * R_j$

Kod negacije, potrebno je pronaći takvu funkciju f, za koju vrijedi da je f(0) = 1 i f(1) = 0, a da se pritom koriste isključivo operacije oduzimanja i množenja. Takva funkcija je:

f(x) = (x-1) * (-1).

Vrijedi:

f(0) = (0-1) * (-1) = -1 * (-1) = 1

f(1) = (1-1) * (-1) = 0 * (-1) = 0

Stoga se svaka pojava semantičke akcije Negacija(i,0,j) ostvaruje sljedećim oblikom naredbi troadresnog međukoda:

$R_j := R_i - 1$

$R_{j+1} := R_j * (-1)$

Ostvarenje preostale operacije, disjunkcije, temelji se na ostvarenju konjunkcije i disjunkcije i primjeni de Morganovog teorema. Naime, prema de Morganovom teoremu vrijedi:

$$A + B = \overline{\overline{A} * \overline{B}}$$

Stoga se svaka pojava semantičke akcije Disjunkcija(i,j,k) ostvaruje sljedećim oblikom naredbi troadresnog međukoda:

$R_k := R_i - 1$

$R_{k+1} := R_k * (-1)$ (računanje NOT(A))

$R_{k+2} := R_j - 1$

$R_{k+3} := R_{k+2} * (-1)$ (računanje NOT(B))

$R_{k+4} := R_{k+1} * R_{k+3}$ (računanje NOT(A)*NOT(B))

$R_{k+5} := R_{k+4} - 1$

$R_{k+6} := R_{k+5} * (-1)$ (računanje NOT(NOT(A)*NOT(B)))

2.2. Formalna definicija

Prvi korak jest izgradnja gramatike koja generira zadane izraze:

<izraz1> →

<izraz1>+<izraz2>

<izraz2>

<izraz2> →

<izraz2>*<izraz3>

<izraz3>

<izraz3> →

<izraz4>

~<izraz4>

<izraz4> →

T

F

(<izraz1>)

Drugi korak je pretvorba tih produkcija u produkcije LL(1) gramatike standardnim postupkom eliminacije lijeve rekurzije:

<izraz1> →

<izraz2><ponovi1>

<ponovi1> →

+<izraz2><ponovi1>

ε

<izraz2> →

<izraz3><ponovi2>

<ponovi2> →

*<izraz3><ponovi2>

ε

<izraz3> →

<izraz4>

~<izraz4>

<izraz4> →

T

F

(<izraz1>)

Treći korak je izgradnja atributne prijevodne gramatike proširenjem prethodno izgrađene gramatike izlaznim završnim znakovima i pravilima računanja njihovih svojstava. Pritom se koristi varijabla globalniIndeks, čija vrijednost označava adresu sljedećeg slobodnog mjesta za pohranu međurezultata. Varijable a i b poprimaju vrijednost varijable globalniIndeks u trenutku završetka parsiranja odgovarajućih nezavršnih znakova. Kad se generira izlazni završni znak, varijabla globalniIndeks se poveća za 1.

<izraz1> →

<izraz2><ponovi1>

<ponovi1>$_a$ →

+<izraz2>{Disjunkcija}$_{a,globalniIndeks-1,globalniIndeks}$<ponovi1>

ε

<izraz2> →

<izraz3><ponovi2>

<ponovi2>$_a$ →

*<izraz3>{Konjunkcija}$_{a,globalniIndeks-1,globalniIndeks}$<ponovi2>

ε

<izraz3> →

<izraz4>

~<izraz4>{Negacija}$_{globalniIndeks-1,0,globalniIndeks}$

<izraz4> →

T {Pridruzi1}$_{0,0,globalniIndeks}$

F {Pridruzi0}$_{0,0,globalniIndeks}$

(<izraz1>)

3. Programsko ostvarenje

Programsko ostvarenje može se formalno opisati sljedećom oznakom:

$$JP_{C\#}^{LogIzrazi->TroAdrMedjukod}$$

Izvorni kod sastoji se od 4 datoteke: frmGlavna.cs, LeksickiAnalizator.cs, SintaksniAnalizator.cs i GeneratorTroadrMedjukoda.cs.

frmGlavna.cs

Datoteka koja sadrži kod za grafičko sučelje. Funckija KursorUpdate() ispisuje trenutni položaj kursora u polju za unos teksta. Klikom na gumb "Kreni" pokreće se rad jezičnog procesora, koji se sastoji od pozivanja pojedinih modula jednog po jednog, uobičajenim redoslijedom: leksički analizator, sintaksni analizator i generator međukoda, koji u sebi sadrži i semantički analizator.

LeksickiAnalizator.cs

Zadatak leksičkog analizatora je trivijalan: provjeriti jesu li svi znakovi ulaznog niza i skupa završnih znakova; ako nisu, signalizirati leksičku grešku.

SintaksniAnalizator.cs

Središnji dio programskog ostvarenja. Prilikom parsiranja, nalaskom na svaki nezavršni, završni i izlazni završni znak stvara se nova instanca razreda Cvor. Čvorovi se spremaju u listu. Po završetku parsiranja, ukoliko nije detektirana greška, funkcija Povezi() obavlja dodavanje čvorova djece čvorovima roditeljima.

Funkcija Analiziraj() pokreće rad sintaksnog analizatora. Početni nezavršni znak je izraz1. Dalje se pozivaju funkcije za preostale nezavršne znakove i generiraju se novi čvorovi u skladu s produkcijama atributne prijevodne gramatike.

GeneratorTroadrMedjukoda.cs

Prvo se pokreće semantički analizator (razred SemantickiAnalizator) koji iz sintaksnog stabla izvlači izlazne završne znakove i stvara listu semantičkih akcija.

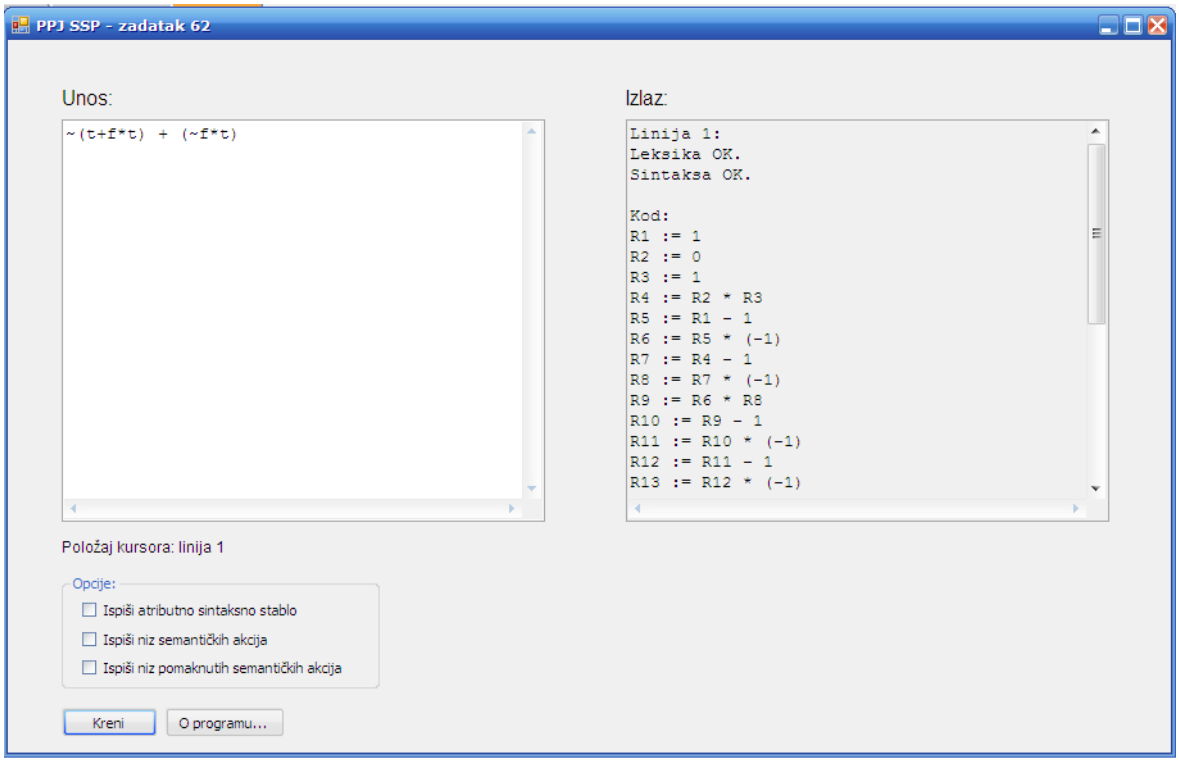
Pri pokretanju generatora troadresnog međukoda prvo se napravi kopija te liste. U kopiji se po potrebi vrši pomak, budući da se za semantičke akcije negacije i disjunkcije generiraju 2, odnosno 7 naredbi troadresnog međukoda.

Generiranje koda radi se na principu dodavanja tekstualnog zapisa naredbi u varijablu Kod tipa string za svaku semantičku akciju iz liste.

Ostatak dokumentacije nalazi se u samom izvornom kodu u obliku xml notacije.

Datoteke frmGlavna.Designer.cs i Program.cs automatski je generirao Visual Studio.

4. Kratke upute



U grafičkom sučelju nalaze se 2 TextBox-a. Lijevi služi za unos, desni za ispis rezultata. Rezultat se ispisuje klikom na gumb "Kreni". Klikom na gumb "O programu..." prikazuje se MessageBox s imenom i prezimenom autora i još par informacija o programu. Dozvoljava se unos proizvoljnog broja sintaksnih cjelina ako se svaka nalazi u svom retku. Prije leksičke analize uklanjaju se sve bjeline unutar retka te se mala slova t i f zamjenjuju velikim T i F. Ponuđene su i dodatne opcije za ispis, kao što se vidi iz priloženog.

5. Zaključak

Ostvaren je jezični procesor koji pretvara logičke izraze u troadresni međukod primjenom programskog jezika C#. Teži dio zadatka bilo je proučavanje i primjena potrebnih formalnih postupaka sintaksne i semantičke analize, te osmišljavanje mehanizama generiranja troadresnih naredbi na temelju utvrđenih semantičkih akcija. Samo programiranje bilo je lakši dio zadatka, iako se znao pojaviti pokoji bug. Izrađeno je funkcionalno i pregledno grafičko sučelje. To također ne predstavlja problem, budući da korišteno razvojno okružje Visual Studio 2008 Express Edition sadrži alat Form Designer.