Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

rookie

Samostalni studentski projekt iz predmeta "Prevođenje programskih jezika"

Zadatak broj 3 (teža inačica)

Zagreb, prosinac 2008.

Samostalni studentski projekt iz predmeta " Prevođenje programskih jezika"

Student: rookie

Matični broj studenta:

Zadatak broj 3 (teža inačica):

Programski ostvariti pretvorbu produkcija u produkcije *LL*(1)-gramatike (udžbenik "Jezični procesori 2", str. 108-111). Ulaz u program je datoteka s produkcijama gramatike u proizvoljnom formatu. Program ne treba provjeravati da li je ulazna gramatika LL(1)-gramatika. Prikazati rad programa na nekoliko primjera.

Napomena:

Potrebno je ostvariti algoritme za lijevo izlučivanje, zamjenu nezavršnih znakova i uklanjanje (izravne) lijeve rekurzije. S obzirom da je potrebu za zamjenom nezavršnih znakova praktički nemoguće otkriti bez računanja cjelovitih *PRIMJENI* skupova, ulaznu datoteku treba proširiti s oznakama koje program upućuju da u označenim produkcijama treba ostvariti zamjenu nezavršnih znakova.

Uvod

Kontekstno neovisna gramatika jest *LL*(1)-gramatika ako i samo ako vrijedi sljedeće:

1) Ako više produkcija ima isti nezavršni znak na lijevoj strani, onda njihovi skupovi *PRIMJENI* nemaju zajedničkih elemenata.

Primjer *LL*(1)-gramatike:

1) <*S*> → <*A*> *b* <*B*>
2) <*S*> → *d*
3) <*A*> → <*C*> <*A*> *b*
4) <*A*> → <*B*>
5) <*B*> → *c* <*S*> *d*
6) <*B*> → *ε*
7) <*C*> → *a*
8) <*C*> → *e* *d*

Tri postupka koji omogućuju pretvorbu produkcija zadane gramatike koje ne zadovoljavaju uvjete *LL*(1)-gramatike u produkcije koje zadovoljavaju uvjete *LL*(1)-gramatike su: *lijevo izlučivanje*, *zamjena nezavršnih znakova* i *razrješavanje lijeve rekurzije*.

Lijevo izlučivanje je postupak preuređivanja produkcija na sljedeći način:

1) Neka gramatika ima *n* produkcija oblika:

   <*A*> → α β_1
   <*A*> → α β_2

   <*A*> → α β_n

   gdje je <*A*> nezavršni znak gramatike, α i β_i su nizovi nezavršnih i završnih znakova gramatike. Skupovi *PRIMJENI* različitih nizova β_i nemaju zajedničkih završnih znakova. Budući da se produkcije započinju istim nizom znakova α, svih *n* produkcija ima iste završne znakove *ZAPOČINJE*(<α>) u skupovima *PRIMJENI*, odnosno gramatika nije *LL*(1)-gramatika.

1) Prethodno zadanih *n* produkcija zamijeni se sljedećim *n*+1 produkcijama koje zadovoljavaju uvjete *LL*(1)-gramatike:

   <*A*> → α <*Nastavak*>

$$
\begin{aligned}
&\langle Nastavak\rangle \quad \rightarrow \quad \beta_1\\
&\langle Nastavak\rangle \quad \rightarrow \quad \beta_2\\
&\qquad\text{---}\\
&\langle Nastavak\rangle \quad \rightarrow \quad \beta_n
\end{aligned}
$$

gdje je <*Nastavak*> novi nezavršni znak.

Zamjena nezavršnih znakova se provodi na sljedeći način:

1) Neka gramatika ima *n* produkcija oblika:

$$
\begin{aligned}
&\langle A\rangle \quad \rightarrow \quad \alpha_1\\
&\langle A\rangle \quad \rightarrow \quad \alpha_2\\
&\qquad\text{---}\\
&\langle A\rangle \quad \rightarrow \quad \alpha_n
\end{aligned}
$$

gdje bilo koja od *n* produkcija ima nezavršni znak <*A*> na lijevoj strani. Nadalje, neka je u gramatici zadana produkcija:

$$\langle B\rangle \quad \rightarrow \quad \beta \ \langle A\rangle \ \gamma$$

gdje su β i γ nizovi završnih i nezavršnih znakova gramatike.

2) Prethodno zadanih *n*+1 produkcija zamijeni se sljedećim *n* produkcijama:

$$
\begin{aligned}
&\langle B\rangle \quad \rightarrow \quad \beta \ \alpha_1 \ \gamma\\
&\langle B\rangle \quad \rightarrow \quad \beta \ \alpha_2 \ \gamma\\
&\qquad\text{---}\\
&\langle B\rangle \quad \rightarrow \quad \beta \ \alpha_n \ \gamma
\end{aligned}
$$

gdje je u produkciji $\langle B\rangle\rightarrow\beta\langle A\rangle\gamma$ nezavršni znak <*A*> zamijenjen desnim stranama *n* produkcija $\langle A\rangle\rightarrow\alpha_i$.

Razrješavanje lijeve rekurzije za opći slučaj je složeno, no za razrješavanje izravne lijeve rekurzije moguće je primjeniti sljedeći jednostavni algoritam:

2) Neka je za nezavršni znak <*A*> zadano *m* izravno lijevo rekurzivnih produkcija:

$$\langle A\rangle \quad \rightarrow \quad \langle A\rangle \ \alpha_i, \qquad 1\leq i\leq m,$$

i *n* produkcija koje nisu izravno lijevo rekurzivne:

$$<A> \quad \rightarrow \quad \beta_j, \qquad 1 \leq j \leq n,$$

gdje su α_i i β_j nizovi završnih i nezavršnih znakova. Lijevo rekurzivne produkcije zamijene se sljedećim produkcijama:

$$\begin{aligned} <A> &\rightarrow \beta_j \; <Ponovi> \qquad 1 \leq j \leq n, \\ <Ponovi> &\rightarrow \alpha_i \; <Ponovi> \qquad 1 \leq i \leq m, \\ <Ponovi> &\rightarrow \varepsilon \end{aligned}$$

gdje je <*Ponovi*> novi nezavršni znak.

Ostvarenje

Zadatak je programski ostvaren u jeziku C# korištenjem alata Microsoft Visual Studio 2008. Ideja zadatka je da program prihvaća datoteku s produkcijama gramatike, te ih primjenom različitih algoritama pretvara u produkcije *LL*(1)-gramatike. Iz samog teksta zadatka je očito da se zadatak sastoji od tri glavna dijela, a to su algoritmi za lijevo izlučivanje, zamjenu nezavršnih znakova i razrješavanje izravne lijeve rekurzije.

Kao što je i u udžbeniku "Jezični procesori 2" navedeno, razrješavanje opće lijeve rekurzije je komplicirano, te u udžbeniku nije naveden način na koji je to moguće ostvariti. Zbog toga naš program može razrješiti samo izravnu lijevu rekurziju u produkcijama zadane gramatike (nezavršni znak lijeve strane produkcije se istodobno nalazi i na krajnje lijevom mjestu desne strane produkcije), poput produkcije:

$$\underline{<S>} \rightarrow \underline{<S>}a.$$

Program u svom radu koristi navedene algoritme određenim redoslijedom, što je potrebno zbog same prirode pretvorbe produkcija u produkcije *LL*(1)-gramatike. Prvo se koristi algoritam za razrješavanje izravne lijeve rekurzije, potom algoritam za zamjenu nezavršnih znakova, te na kraju algoritam za lijevo izlučivanje. Pošto je kod programa temeljito komentiran, lako je shvatiti načine na koji spomenuti algoritmi rade.

Programu se gramatika zadaje pomoću tekstualne datoteke (.txt) u kojoj produkcije gramatike trebaju biti napisane u formatu:

<S>->α ,

gdje je <S> nezavršni znak, a α niz završnih i nezavršnih znakova, te se svaka produkcija mora nalaziti u novom redu, i to bez praznih redova između njih. Znakovi '<' i '>' se ne smiju koristiti kao završni znakovi gramatike, zato jer se koriste za označavanje neazvršnih znakova gramatike. Također se ε-produkcije zbog nemogućnosti prikaza slova ε u text-only formatu tekstualne datoteke zadaju i ispisuju na zaslon kao:

<S>->epsilon .

Kako je u tekstu zadatka i navedeno, s obzirom da je potrebu za zamjenom nezavršnih znakova praktički nemoguće otkriti bez računanja cjelovitih *PRIMJENI* skupova, ulaznu datoteku treba proširiti s oznakama koje program upućuju da u označenim produkcijama treba ostvariti zamjenu nezavršnih znakova. Prilagođujemo format ulaznih produkcija:

[*]<S>->α ,

gdje znak '*' ukoliko se nalazi na početku produkcije, ispred nezavršnog znaka, označava da se u toj produkciji treba obaviti zamjena nezavršnog znaka. Uglatim zagradama se naznačuje neobavezni izbor znaka '*' .

Primjer pravilno zapisane gramatike u ulaznoj .txt datoteci:

```
<Sentence>-><Sentence>a
<Sentence>-><Att>
<Sentence>->b
<Att>->a
*<Att>-><B>c
<B>->acd<Att>deabc
<B>->b<B>
```

Primjer nepravilno zapisane gramatike u ulaznoj .txt datoteci:

```
<Sentence>->Sentence>a
<Sentence>--><Att>

<Sentence>=>b
Att>->a
*<Att-><B>c
**<B>->acd<Att>dea->bc
b->b<B>
```

Napomena: U gore navedenoj nepravilno zapisanoj gramatici, u svakoj se produkciji nalazi barem jedna greška.

Pokretanjem programa se učitava tekstualna datoteka, koja se potom ispisuje na zaslon, svaka produkcija se stavlja u posebnu string varijablu, prebrojava se broj produkcija u ulaznoj gramatici, te se poruka o broju ispisuje na zaslon. Nakon toga se izvršavaju algoritmi za pretvorbu produkcija u produkcije *LL*(1)-gramatike, te se naposljetku ispisuje dobivena *LL*(1)-gramatika, kao i sadašnji broj produkcija u njoj.

Korištenjem spomenutog alata Microsoft Visual Studio 2008 napravljeno je i rudimentarno grafičko sučelje koje korisniku olakšava služenje programom. Sastoji se od trake s menijem, trake s alatima, dvije tekstualne kućice za ispis ulazne gramatike i dobivene *LL*(1)-gramatike, te gumba za pokretanje pretvorbe izmedju njih. Funkcionalna statusna traka pruža korisne informacije tokom izvođenja programa.

File→Open:	Otvori ulazni tekstualnu datoteku
File→Convert:	Pretvori ulaznu gramatiku u LL(1)-gramatiku
File→Exit:	Prekini s radom programa

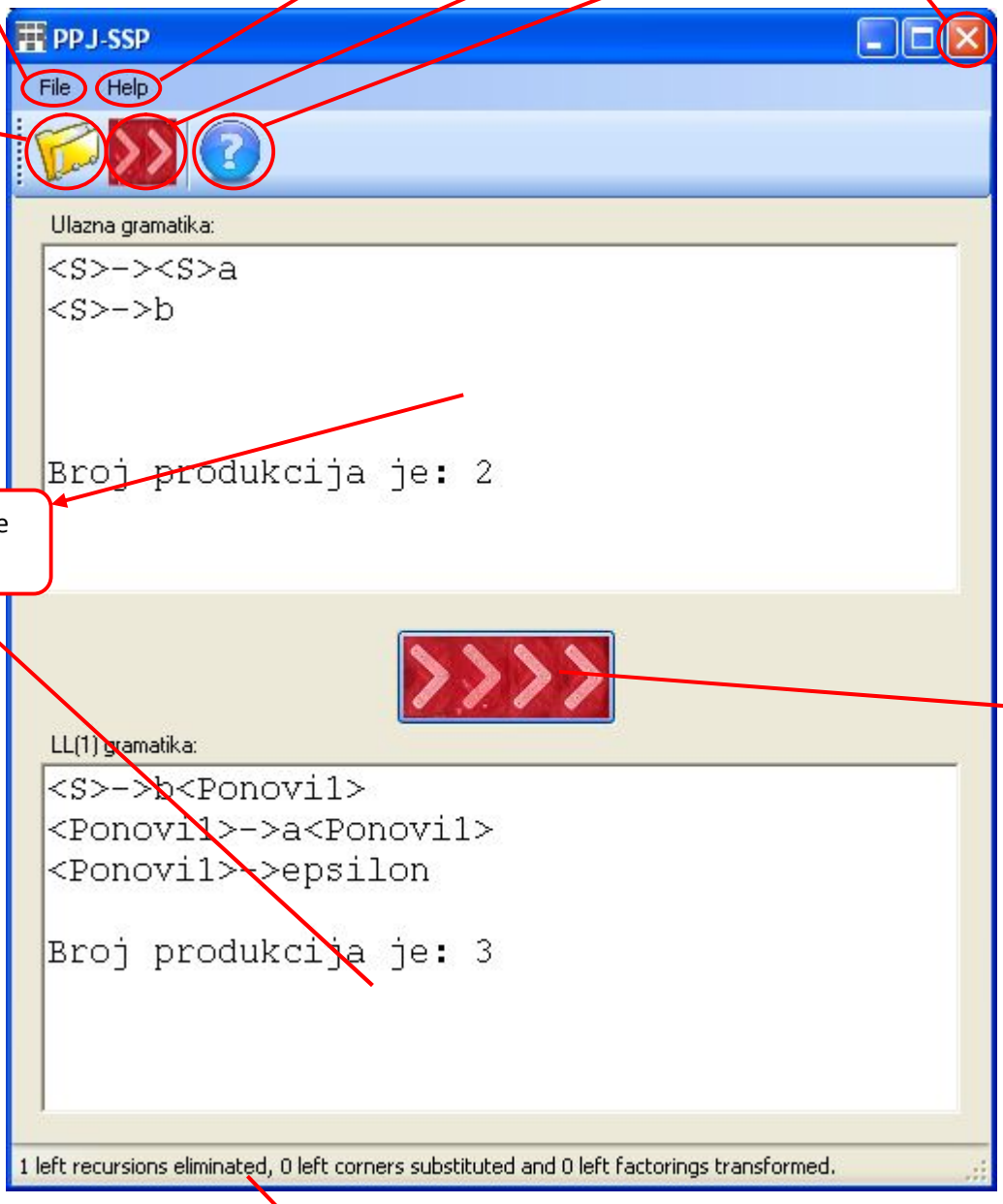
Help→About:	Informacije o programu
Help→Help:	Otvori ovu datoteku

Convert

Help

Exit

Open



Tekstualne kućice za ispis

Convert

Statusna traka

Na gornjoj slici programa je vidljiv i rad programa na jednostavnom primjeru ulazne gramatike. Na sljedećem primjeru je korištena gramatika pri čijoj su pretvorbi korištena sva tri algoritma.

Ulazna gramatika:

```
<S>-><S>a
<S>->c
<A>->a
*<A>-><B>c
<B>->a<A>
<B>->b<B>

Broj produkcija je: 6
```

LL(1)-gramatika:

```
<S>->c<Ponovi1>
<A>->a<Nastavak1>
<A>->b<B>c
<B>->a<A>
<B>->b<B>
<Ponovi1>->a<Ponovi1>
<Ponovi1>->epsilon
<Nastavak1>-><A>c
<Nastavak1>->epsilon

Broj produkcija je: 9
```

Zaključak

U ovom samostalnom studentskom projektu bilo je potrebno ostvariti funkcionalno programsko rješenje koje će provoditi pretvorbu produkcija ulazne gramatike u produkcije *LL*(1)-gramatike. To je i učinjeno korištenjem alata Microsoft Visual Studio 2008, te je rješenje napisano u programskom jeziku C#. Rezultat je program PPJ-SSP koji uspješno rješava zadani problem pretvorbe u *LL*(1)-gramatiku.

Glavnina zadanog problema je bila implementacija tri algoritma za pretvorbu produkcija u produkcije *LL*(1)-gramatike. Algoritmi su uspješno implementirani, sukladno objašnjenjima iz udžbenika "Jezični procesori 2" uz manje preinake. Programsko rješenje može razrješiti samo izravnu lijevu rekurziju, te također da bi se uspješno zamijenio nezavršni znak, produkcija ulazne gramatike u ulaznoj tekstualnoj datoteci mora na početku, ispred nezavršnog znaka, imati znak '*'.

Ispis ulazne i preuređene gramatike, kao i sve ostale potrebne informacije o toku izvođenja programa se prikazuju u grafičkom sučelju programskog rješenja, koje svojom jednostavnošću i intuitivnim dizajnom olakšava služenje programom.