Programski prevodioci: Vežbe 2

Sadržaj

1. Uvod

Na ovim vežbama biće prikazana BNF notacija, kao i njena upotreba u implementaciji parsera.

2. Parser

Bison je program koji generiše sintaksni analizator, odnosno parser. Bison čita ulazna pravila i kao izlaz vraća kod koji implementira parser u programskom jeziku C. Nastao je 1985. godine kao slobodna verzija programa Yacc. Dostupan je kao deo GNU projekta, pod GNU opštom javnom licencom, verzija 3.

Sintaksnom analizom se odgovara na pitanje "Da li je raspored tokena u skladu sa gramatikom?". Gramatika se zadaje u BNF notaciji (Bakus-Naurova forma).

Bison se često koristi u kombinaciji sa Flex-om za izradu kompajlera. Leksičkom analizom se prepoznaju šabloni u tekstu i tekst se deli na tokene, a zatim se sintaksnom analizom proverava da li se tokeni nalaze u ispravnom redosledu.

Primer jednostavnog Bison programa:

```
 /* Deo za definicije */
%% ①
 /* Deo za pravila */
izraz ②
        : BROJ PLUS BROJ
        | BROJ MINUS BROJ
        ;
%% ①
 /* Deo za proizvoljan C kod */
```

① Kao i kod Flex programa, Bison programi se sastoje iz 3 sekcije. Razlika je što pravila u sekciji 2 predstavljaju gramatiku jezika i definišu se u BNF notaciji.

② Pravila se sastoje iz pojma sa leve strane znaka `:` koji može biti zamenjen sekvencom pojmova i/ili simbola sa desne strane znaka `:`.

Prvi pojam koji je naveden u datoteci se uzima kao polazni pojam. Ako kompletan niz ulaznih tokena može da se svede na polazni pojam, tada se može reći da je ulaz u skladu sa gramatikom.

Bison generiše LALR(1) parser, odnosno LR parser sa jednim "*lookahead*" tokenom. LR parseri spadaju u grupu *bottom-up* parsera, što znači da grade stablo parsiranja od listova ka korenu stabla. Parser preuzima tokene sa ulaza i pokušava da prepozna desnu stranu nekog pravila. Ako prepozna desnu stranu nekog pravila, zamenjuje je levom stranom, odnosno radi redukciju. Cilj je da se stigne do polaznog pojma gramatike, odnosno do korena stabla.

U nazivu LR parsera, "**L**" ("***l**eft*") označava da se ulaz čita sa **l**eva na desno, a "**R**" ("***r**ight*") označava da se koristi **d**esno izvođenje na gore.

3. BNF notacija

Primer	**Značenje**
`p1: T1 T2 p2 T3;`	Pojam `p1` može biti zamenjen sekvencom tokena/pojmova sa desne strane znaka `:`.
`p: T1 T2 | T3 T4;`	Pojam `p` može biti zamenjen sekvencom tokena `T1 T2` *ili* `T3 T4`.
`p: T | p T;`	BNF notacija koju Bison koristi ne podržava kvantifikatore, već se ponavljanje specificira rekurzivnim pravilima. Rekurzivno pravilo predstavlja pravilo u kojem se pojam `p` sa leve strane takođe nalazi i sa desne strane znaka `:`. Dakle, navedeni primer označava da pojam `p` može biti zamenjen sekvencom od jednog ili više ponavljanja tokena `T`.

4. Kompajliranje, pokretanje i testiranje

4.1. "Ručno" kompajliranje

1. `.l` datoteku je potrebno proslediti programu Flex:

```
flex *.l
```

2. `.y` datoteku je potrebno proslediti programu Bison:

```
bison -d *.y
```

3. Dobijeni C kod je potrebno kompajlirati standardnim C kompajlerom:

```
gcc *.c -o parser
```

4. Dobijeni izvršni program pokrenuti na sledeći način:

```
./parser
```

4.2. Kompajliranje upotrebom `make` alata

Kao što se može videti, proces kompajliranja zahteva ručno pozivanje čak 3 različita alata. Kako bi se proces pojednostavio, od ovih vežbi pa do kraja semestra, biće data datoteka `Makefile`. Ova datoteka predstavlja "recept" kako se dolazi do izvršnog programa.

Prema tome, sve što treba uraditi je pozvati sledeću komandu:

```
make
```

5. Bison upozorenja o postojanju konflikata

Prilikom kompajliranja rešenja, može se dogoditi da Bison prikaže neku od sledećih poruka:

```
syntax.y: warning: 1 shift/reduce conflict
```

```
syntax.y: warning: 1 reduce/reduce conflict
```

Postojanje konflikta znači da je gramatika napisana na dvosmislen način.

`reduce/reduce` **konflikt**

U jednom trenutku postoje 2 ili više različitih pravila sa desne strane po kojima parser može da uradi redukciju.

`shift/reduce` **konflikt**

U jednom trenutku parser može da uradi redukciju ali isto tako može i da nastavi sa preuzimanjem tokena.

Iako se radi samo o upozorenju, a ne o grešci, ovo ne znači da ovakve poruke treba ignorisati. Potrebno je izmeniti gramatiku tako da ona više ne bude dvosmislena.

6. Zadaci

6.1. Zadatak 1

Proširiti tekst gramatiku upitnim i uzvičnim rečenicama. Na kraju programa ispisati koliko ima upitnih, koliko uzvičnih, a koliko izjavnih rečenica.

6.2. Zadatak 2

Proširiti tekst gramatiku tako da se bilo koje dve reči u rečenici mogu odvojiti jednim zarezom. Zarez ne sme da se pojavi iza poslednje reči.

6.3. Zadatak 3

Proširiti tekst gramatiku pasusima:

- Tekst je niz od jednog ili više pasusa.
- Pasus je niz od jedne ili više rečenica.
- Pasusi su odvojeni bar jednim znakom NEWLINE.

Na kraju programa ispisati ukupan broj pasusa.

7. Projekat

U dokumentu [https://docs.google.com/document\d\e\2PACX-1vTh-HFm3Ujw2s6Oa_bWZGeYQM87NHUpS2L90rq72DQd0QigfpDDax0-Rh01PLNKKiwx7iGQUV3Ouecu\pub] se nalaze osnovne informacije vezane za projekat i zajednički zadaci koje je potrebno da svi implementiraju. Svake nedelje dokument će biti dopunjavan sa obaveznim delovima koje je potrebno implementirati.