PITANJA ZA USMENO ODGOVARANJE IZ PPJ

Što su lijevi i desni kontekst?

Kod desnog konteksta pretraživanje se radi na način **r/r'**, tj. neki niz je leksička jedinka samo ako je iza niza definiranog regularnim izrazom **r** niz koji je definiran regularnim izrazom **r'.**

Pretraživanje kod lijevog konteksta se zadaje na način **<ImeStanja> r**, što označava da je niz definiran regularnim izrazom **r** ako i samo ako je simulator u dodatnom stanju **ImeStanja**

Kako rade analizatori?

LEKSIČKI ANALIZATOR – slijedno čita tekst izvornog programa znak po znak, odbacuje znakove koji se ne koriste u danjim koracima rada JP-a, grupira znakove u leksičke jedinke, određuje im klasu, pronalazi pogreške i mjesto pogreške, zapisuje parametre leksičkih jedinki u tablicu znakova i čuva tekstualnu strukturu izvornog programa

SINTAKSNI ANALIZATOR – slijedno čita uniformne znakove leksičkih jedinki, grupira ih u sintaksne cjeline, stvara hijerarhiju sintaksnih cjelina, opisuje i određuje mjesto sintaksnih pogrešaka, pokreće postupak oporavka od pogreške i gradi sintaksno stablo

Ukratko, leksički analizator čita znak po znak izvornog programa, prepravlja ga i radi tablicu znakova i šalje niz leksičkih jedinki sintaksnom analizatoru, a sintaksni analizator tad čita te uniformne znakove i stvara sintaksno stablo.

Kako rade generatori analizatora (dakle, što su ulazi i izlazi)?

GENERATOR LEKSIČKOG ANALIZATORA – ulaz mu je opis procesa leksičkog analizatora zadanog tekstualnom datotekom, a izlaz je izvorni kod leksičkog analizatora napisan u jeziku izgradnje

GENERATOR SINTAKSNOG ANALIZATORA – ulaz je opis procesa sintaksnog analizatora zadan tekstualnom datotekom, a izlaz je izvorni kod sintaksnog analizatora

Kako određujemo klasu leksičke jedinke? (pitao je i što bi bilo ako ne bismo tražili najdulji prefiks).

1) Za sve leksičke jedinke koje je potrebno razlikovati tijekom sintaksne analize definira se zasebni regularni izraz
2) Ako je niz znakova x definiran primjenom dva regularna izraza r_k i r_l koji označavaju dvije različite klase leksičkih jedinki k i l, onda je niz x u klasi k, ako i samo ako je regularni izraz r_k zapisan u listi regularnih izraza prije od izraza r_l

Tijekom grupiranja traži se najdulji prefiks niza w koji je definiran barem jednim regularnim izrazom

Kako radi LR parser?

Koristi tehniku *Pomakni-Reduciraj*. Koriste se 2 tablice: tablica *Akcija* i tablica *NovoStanje. Akcija* je proširenje tablice *Pomakni/Reduciraj*. Akcije *Pomakni()* proširuju se podacima iz tablice *Stavi*. Proširena akcija *Pomakni(Stanje)* stavlja 2 znaka na stog: pročitani znak ulaznog niza i stanje LR parsera. Tablica *NovoStanje* koristi se isključivo tijekom izvođenja akcije *Reduciraj.*

Koje su moguće akcije? Što koja od njih znači?
Prihvati, Reduciraj, Odbaci, Pomakni.

PRIHVATI – ako su pročitani svi znakovi ulaznog niza, a na stogu je početni nezavršni znak <S> i oznaka dna stoga, parser prihvati niz w
- niz w je u jeziku koji generira zadana gramatika samo ako ispod početnog nezavršnog znaka nema nijednog drugog znaka osim oznake dna stoga

REDUCIRAJ – ako je na vrhu stoga uzorak za zamjenu, onda se ta akcija primjenjuje
- s vrha stoga se uzmu znakovi desne strane produkcije, a na vrh stoga stavi se nezavršni znak lijeve strane produkcije

ODBACI – ako nije moguća daljnja gradnja stabla, primjenjuje se ta akcija i niz w se ne prihvaća
- akcija se primjenjuje u 2 slučaja:
  - ako nije zadana nijedna akcija za pročitani znak ulaznog niza i znak vrha stoga
  - kad kod akcije *Reduciraj* znakovi na vrhu stoga nisu jednaki znakovima nijedne desne strane

POMAKNI – pročitani znak stavlja se na vrh stoga, a kazaljka se miče za jedan znak u desno

Koja je razlika između SLR i kanonskog parsera?
SLR parsiranje je najjednostavniji postupak, a nedostatak mu je nemogućnost primjene na veliki skup jezika. Obuhvaća nešto širi skup jezika od klase LR(0) jezika.
Kanonski LR algoritam je nešto složeniji i obuhvaća LR(1) klasu jezika.

Čemu služe sinkronizacijski znakovi?
Služe za nadziranje pogrešaka LR parsera.

Kako smo ostvarili generativno stablo?
Pomoću sintaksnog analizatora na standardni izlaz smo ispisali konstruirano generativno stablo, a ispis smo obavili dubinskim obilaskom generativnog stabla, pri čemu smo prvo ispisivali oznaku čvora roditelja, a nakon toga podstabla čiji korijeni su djeca čvora roditelja, s lijeva na desno. Na svakoj sljedećoj razini stabla, ispis treba prefiksirati jednim dodatnim razmakom.

Što je izlaz leksičkog analizatora?
Niz leksičkih jedinki i tablica znakova

Čemu služi stablo?
Da znamo način na koji smo obradili program, tj. da znamo redoslijed izvođenja naredbi itd. što ćemo poslije koristiti u semantičkoj analizi.

Čemu služe relacije ZAPOČINJE()?
ZAPOČINJE(α) je skup svih završnih znakova gramatike koji su na krajnje lijevom mjestu barem jednog međuniza generiranog iz niza α

Kako rješavamo nejednoznačnost? (oba labosa)

Kod leksičkog analizatora u 1. labosu, nejednoznačnost smo riješili pomoću 2 pravila:

1) Ako je niz znakova x definiran primjenom dva regularna izraza r_k i r_l koji označavaju dvije različite klase leksičkih jedinki k i l, onda je niz x u klasi k, ako i samo ako je regularni izraz r_k zapisan u listi regularnih izraza prije od izraza r_l
2) Tijekom grupiranja traži se najdulji prefiks niza w koji je definiran barem jednim regularnim izrazom. Neka je w niz znakova izvornog programa, a x i y su nizovi znakova definirani zadanim regularnim izrazima niz znakova x, koji je prefiks niza w, jest leksička jedinka ako i samo ako bilo koji drugi prefiks y niza w jest ujedno i prefiks niza x.

Kod sintaksnog analizatora u 2. labosu, nejednoznačnosti u gramatici u gradnji LR(1) parsera očituju se kroz Pomakni/Reduciraj i Reduciraj/Reduciraj proturječja. Pomakni/Reduciraj proturječje izgrađeni generator treba razriješiti u korist akcije Pomakni. Reduciraj/Reduciraj proturječje potrebno je razriješiti u korist one akcije koja reducira produkciju zadanu ranije u Ulaznoj Datoteci. Poželjno je da generator korisniku ispiše gdje je došlo do proturječja i kako je proturječje razriješeno.

Kako se odvija oporavak od pogreške? (oba labosa)

1. Labos: Odbacivanjem krajnje lijevog znaka niza koji nema nijedan prefiks koji je definiran barem jednim od regularnih izraza. Postupak se ponavlja sve dok ne ostane niz koji ima prefiks definiran barem jednim od regularnih izraza.
2. Labos: U simulator LR parsera se ugradi postupak oporavka od pogreške traženjem sinkronizacijskog znaka. Kad naiđe na pogrešku, simulator ispisuje poruku o pogrešci s navedenim brojem retka u kojem se dogodila pogreška, očekivani uniformni znakovi, te pročitani uniformni znak iz niza uniformnih znakova i odgovarajući znakovni prikaz iz izvornog koda programa. Nakon ispisa pogreške, simulator preskoči sve znakove u nizu uniformnih znakova do prvog sljedećeg sinkronizacijskog znaka. Kad je simulator pronašao prvi sljedeći sinkronizacijski znak, sa stoga odbacuje stanja dok ne dođe do stanja u kojem je definirana *Akcija[s,sinkronizacijski_znak]*. Nakon toga simulator nastavlja s normalnim radom.

Kako izgrađujemo ε-NKA? (oba labosa)

1. Labos

Sve regularne definicije smo zamijenili regularnim izrazima, te za svaki regularni izraz izgradili ε-NKA. Generator leksičkog analizatora gradi konačni automat M koji prihvaća jezik zadan regularnim izrazom $r_1+r_2+r_3+\ldots+r_n$. Konačni automat za koji vrijedi da je $L(M)=L(r_1+r_2+r_3+\ldots+r_n)$ gradi se na način:

1) Generator leksičkog analizatora konstruira ε-NKA M_i za sve regularne izraze r_i tako da vrijedi $L(r_i)=L(M_i)$
2) Nakon što se izgrade ε-NKA M_i za sve regularne izraze r_i, generator doda novo početno stanje p_0 i ε-prijelaze iz stanja p_0 u početna stanja svih ε-NKA M_i

2. Labos

Stanja automata su označena LR(1) stavkama. Zbog uniformnosti oznaka stanja, nije potrebno ugraditi stanje q_0, nego se za početno stanje može koristiti LR(1) stavka koja proizlazi iz dodane produkcije koja iz novog početnog nezavršnog znaka generira originalni početni nezavršni znak gramatike zadan u Ulaznoj Datoteci. U ovom slučaju, automat sa i bez stanja q_0 potpuno su ekvivalentni, tj. prihvaćaju isti jezik. Prilikom izgradnja automata, nužno je računati ZAPOČINJE skupove za nizove znakova, te ih proširiti s desnih strana produkcije na bilo koji sufiks desne strane produkcije. Dobiveni ε-NKA pretvorimo u istovjetni DKA.