11. NEJEDNOZNAČNOST GRAMATIKE, JEZIKA I NIZA

11.1.Nejednoznačnost niza

- kontekstno neovisni jezik i gramatika
- nejednoznačnost interpretacije niza
- redoslijedi primjene produkcija i zamjene nezavršnih znakova

Jezik je **kontekstno neovisan** ako i samo ako postoji kontekstno neovisna gramatika koja ga generira. Time je definirana istovjetnost kontekstno neovisne gramatike i kontekstno neovisnog jezika.

Interpretacija niza se zasniva na generativnom stablu koje se gradi tijekom generiranja niza. Mogućnost gradnje više generativnih stabala uzrokuje **nejednoznačnost u interpretaciji niza**.

Redoslijed nezavršnih znakova na koje se primjenjuju produkcije je značajan za postupak generiranja niza. Koriste se dva postupka:

- Generiranje niza zamjenom krajnje lijevog nezavršnog znaka → produkcije se primjenjuju isključivo na krajnje lijeve nezavršne znakove.
- Generiranje niza zamjenom krajnje desnog nezavršnog znaka → produkcije se primjenjuju isključivo na krajnje desne nezavršne znakove.

11.2. Sistematizacija zamjene

- sistematizacija zamjene nezavršnih znakova
- obilazak stabla
- definicija nejednoznačnosti gramatike, niza i jezika
- razrješenje nejednoznačnosti

Redoslijed nezavršnih znakova na koje se primjenjuju produkcije je značajan za postupak generiranja niza. Koriste se dva postupka:

- Generiranje niza zamjenom krajnje lijevog nezavršnog znaka → produkcije se primjenjuju isključivo na krajnje lijeve nezavršne znakove.
- Generiranje niza zamjenom krajnje desnog nezavršnog znaka → produkcije se primjenjuju isključivo na krajnje desne nezavršne znakove.

Postupak **obilaska** generativnog stabla određuje redoslijed kojim se obilaze grane i čvorovi. Stablo se može obilaziti :

- Desnim obilaskom → rekurzivno obilazi sve neobiđene desne grane i čvorove
- Lijevim obilaskom → rekurzivno obilazi sve neobiđene lijeve grane i čvorove

Obilazak stabla započinje korijenom stabla. Postupci obilaska stabla jednoznačno definiraju redoslijed primjene produkcija i redoslijed nezavršnih znakova na koje se te produkcije primjenjuju.

Ako je za neki niz $w \in L(G)$ (niz gramatike G) moguće izgraditi više različitih generativnih stabala, tada je **kontekstno neovisna gramatika G nejednoznačna**. Ako je moguće za neki niz $w \in L(G)$ izgraditi više različitih generativnih stabala, onda je **niz nejednoznačan** za zadanu gramatiku G. Ako ne postoji jednoznačna gramatika koja generira jezik L tada je on **nejednoznačan.** Nejednoznačnost se rješava **promjenom gramatike** ili **promjenom jezika**. Razlika između ove dvije promjene je da se promjenom gramatike ne mijenja jezik i odbacuje se višestruko značenje niza dok se promjenom jezika čuva višestruko značenje niza.

11.3. Promjena gramatike

- definicija promjene gramatike
- primjer
- svojstva promjene gramatike

Umjesto nejednoznačne gramatike G gradi se jednoznačna gramatika G'. **Na primjer:**

Jezik L, koji generira nejednoznačna gramatika G=({E}, {a, ®}, { E → E®E | a }, E), moguće je generirati primjenom više različitih jednoznačnih gramatika.

Za lijevo asocijativni operator ® gradi se jednoznačna gramatika: G_1=({ E, T }, {a, ®}, { E → E®T | T, T → a }, E). Za desno asocijativni operator ® gradi se jednoznačna gramatika: G_2=({ E, T }, {a, ®}, { E → T®E | T, T → a }, E).

Izbor jednoznačne gramatike određuje način gradnje generativnog stabla. Promjenom gramatike se ne mijenja jezik i odbacuje se višestruko značenje niza.

11.4. Promjena jezika

- definicija promjene jezika
- primjer
- svojstva promjene jezika

Umjesto nejednoznačnog jezika L izgradi se jednoznačan jezik L' kojeg je moguće generirati jednoznačnom gramatikom. Promjenu jezika primjenjujemo:

- kada je jezik inherentno nejednoznačan
- kada je jednoznačna gramatika previše složena
- kada se žele sačuvati sve interpretacije nizova

Primjer promjene jezika su zagrade. Za gramatiku G=({E}, {a, ®}, { E → E®E | a }, E) moguće su dvije interpretacije niza a®a®a. Dodavanjem zagrada moguće je definirati redoslijed primjene operatora ® i izbjeći nejednoznačnost → G_3=({E}, {a, ®, (,)}, { E → (E)®(E) | a }, E).

Promjenom jezika se zadržava višestruko značenje definiranjem zasebnog niza za svaku interpretaciju.

12. POJEDNOSTAVLJENJE GRAMATIKE

12.1. Definicija pojednostavljenja gramatike

- svrha pojednostavljenja
- željena svojstva gramatike
- definicija beskorisnosti znaka

Postupci pojednostavljenja gramatike odbacuju beskorisne znakove i produkcije. Za bilo koji neprazni kontekstno neovisni jezik L moguće je izgraditi kontekstno neovisnu gramatiku G sa svojstvima:

- Bilo koji znak gramatike G se koristi u postupku generiranja barem jednog niza jezika L
- Gramatika G nema produkcije oblika A→B (A i B nezavršni znakovi)
- Ako prazni niz ε nije element jezika L, tada je moguće izbjeći produkcije oblika A→ε

Znak je **beskoristan** ako se **ne koristi** u postupku generiranja:

$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w \; ; \qquad \alpha, \beta \in (V \cup T)^*, \;\; w \in T^*$

Dva su vida beskorisnosti: znak je **mrtav** (ne koristi se u drugom dijelu X→w) ili **nedohvatljiv.**

12.2. Odbacivanje beskorisnih znakova

- Definicija i algoritam odbacivanja mrtvih znakova
- Definicija i algoritam odbacivanja nedohvatljivih znakova
- Odbacivanje beskorisnih znakova

Za bilo koju CFG G=(V, T, P, S) je moguće izgraditi istovjetnu gramatiku G'=(V', T', P', S') koja nema mrtvih znakova (Za bilo koji X_i vrijedi $X_i \rightarrow^* w_i$). Prvo se pronađu svi živi znakovi i njihovim odbacivanjem se dobije lista mrtvih znakova. Algoritam traženja živih znakova se odvija u tri koraka:

- U listu živih znakova se stave lijeve strane produkcija koje s desne strane nemaju nezavršne znakove
- U listu živih znakova se stave lijeve strane produkcija koje s desne strane imaju isključivo žive znakove
- Algoritam se zaustavlja kada više nema živih znakova.

Za bilo koju CFG G=(V, T, P, S) je moguće izgraditi istovjetnu gramatiku G'=(V', T', P', S') koja nema nedohvatljivih znakova (Za bilo koji $X \in V' \cup T'$ vrijedi $S \rightarrow^* \alpha X \beta$). Prvo se pronađu svi dohvatljivi znakovi i njihovim odbacivanjem se dobije lista nedohvatljivih znakova. Algoritam traženja dohvatljivih znakova se odvija u tri koraka:

- U listu dohvatljivih znakova se stavi početni nezavršni znak
- Ako je znak s lijeve strane produkcije u listi dohvatljivih znakova, tada su svi znakovi s desne strane također dohvatljivi znakovi
- Algoritam se zaustavlja kada listu više nije moguće proširiti.

Nužno je prvo primijeniti algoritam odbacivanja mrtvih znakova, a zatim algoritam odbacivanja nedohvatljivih znakova.

12.3. Odbacivanje ε i jediničnih produkcija

- Odbacivanje ε produkcija
- Odbacivanje jediničnih produkcija

Iz gramatike G se gradi istovjetna gramatika G' koja nema ε produkcija. Algoritam odbacivanje ε produkcija se odvija u dva koraka:

- Pronađu se svi znakovi za koje vrijedi $A \rightarrow^* \varepsilon$. U listu praznih znakova se stave lijeve strane ε produkcija. Ako su svi znakovi s desne strane prazni tada se u listu dodaje i znak s lijeve strane produkcije.
- Skup produkcija gramatike G' se gradi na sljedeći način. Ako je produkcija $A \rightarrow X_1X_2X_3\ldots X_n$ produkcija gramatike G, onda u skup produkcija gramatike G' se dodaju produkcije oblika $A \rightarrow \xi_1\xi_2\ldots\xi_n$. ξ_i primaju sljedeće vrijednosti:
  - Ako je X_i prazan znak tada je ξ_i ili ε ili X_i
  - Ako je X_i nije prazan znak tada je $\xi_i = X_i$

  Produkcije se grade na temelju svih mogućih kombinacija oznaka $\xi_1\xi_2\ldots\xi_n$.

Iz gramatike G, koja generira kontekstno neovisni jezik L(G)\{ε} je moguće izgraditi istovjetnu gramatiku G' koja nema jediničnih produkcija oblika A→B. Gramatika G' se gradi na način:

- U skup produkcija G' se stave sve produkcije iz G koje nisu jedinične.
- Ako vrijedi A→* B tada se za sve produkcije B→ a koje nisu jedinične se grade nove produkcije A→a

13. NORMALNI OBLIK CHOMSKOG

13.1. Definicija normalnog oblika Chomskog

- Definicija
- Postupak pojednostavljenja gramatike u CNF

Neka gramatika G = (V, T, P, S) generira kontekstno neovisni jezik L(G)\{ε}. Moguće je izgraditi istovjetnu gramatiku G' koja ima sve produkcije oblika A→BC ili A→a.

Postupak pretvorbe se odvija u tri koraka:

- U skup produkcija gramatike G' se stave sve produkcije koje su već oblika A→BC ili A→a.
- Za produkcije oblika A→ $X_1X_2…X_m$ (X_i su nezavršni ili završni znakovi). Ako je X_i završni znak a tada se skup nezavršnih znakova proširi sa novim znakom C_a , a skup produkcija P' se proširi s produkcijom C_a→ a (te produkcije su u CNF). Taj postupak se ponavlja za sve završne znakove na desnoj strani produkcije.
- Nakon što završi drugi korak sve produkcije su oblika A→a ili A→$B_1B_2..B_m$ (m>=2). Produkcije oblika A→a i A→B_1B_2 su u CNF. Produkcije oblika A→$B_1B_2..B_m$ m>=3 se zamijene s novim produkcijama tako da se definiraju novi nezavršni znakovi D tako da sve produkcije budu u CNF-u (npr za A→ $B_1B_2B_3$; A→ B_1 D_1; D_1→ B_2B_3)

13.2. Postupak pojednostavljenja gramatike u CNF

- Postupak pojednostavljenja gramatike u CNF
- Primjer

Primjer:

S→bA	S→aB	A→bAA
A→aS	A→a	B→aBB
B→bS	B→b	

Produkcije A→a i B→b su u CNF, te se izravno stavljaju u skup produkcija G'. Produkcija S→bA se zamijeni produkcijom S→C_bA i doda se produkcija C_b→b. Slično se preurede i ostale produkcije, pa imamo:

S→C_bA	A→C_aS	B→C_bS	S→C_aB
A→a	B→b	A→C_bAA	B→C_aBB

Produkcije A→C_bAA i B→C_aBB nisu u CNF i obavlja se zamjena: A→C_bD_1 ; D_1→AA i B→C_aD_2 ; D_2→BB. Nakon ovakve zamjene sve produkcije su u CNF-u.

14. NORMALNI OBLIK GREIBACHA

14.1. Definicija normalnog oblika Greibacha

- Definicija
- Postupak pojednostavljenja gramatike u GNF

Neka gramatika G generira kontekstno neovisni jezik L(G)\{ε}. Moguće je izgraditi istovjetnu gramatiku G' koja ima sve produkcije oblika $A \rightarrow a\alpha$. $a \in T$, $\alpha \in V^*$.

Tijekom pretvorbe produkcija u Greibachov normalni oblik koriste se tri algoritma: algoritam pretvorbe produkcija u CNF, algoritam zamjene krajnjeg lijevog nezavršnog znaka i algoritam rješavanja lijeve rekurzije.

14.2. Algoritam zamjene krajnjeg lijevog znaka

- Definicija
- Primjer

Neka u gramatici G r produkcija ima nezavršni znak D_j na lijevoj strani: $D_j \rightarrow a_1$; $D_j \rightarrow a_2$;…; $D_j \rightarrow a_r$ i neka je u gramatici produkcija koja ima D_j na krajnjem lijevom mjesto desne strane: $D_i \rightarrow D_j\gamma$. Prethodno zadanih **r+1** produkcija se zamijeni s **r** produkcija. $D_i \rightarrow a_1\gamma$; $D_i \rightarrow a_2\gamma$;…; $D_i \rightarrow a_r\gamma$. (U produkciji $D_i \rightarrow D_j\gamma$ nezavršni znak D_j se zamjeni desnim stranama svih **r** produkcija koje su oblika $D_j \rightarrow a_i$)

14.3. Algoritam razrješavanja lijeve rekurzije

- Definicija
- Primjer

Produkcija je **lijevo rekurzivna** ako je isti nezavršni znak na lijevoj strani produkcije i na krajnje lijevom mjestu desne strane produkcije.

Algoritam razrješavanja lijeve rekurzije:
Neka je za nezavršni znak D_i zadano **r** lijevo rekurzivnih produkcija: $D_i \rightarrow D_i a_k$ $(1 \le k \le r)$ i neka je zadano **s** produkcija koje nisu lijevo rekurzivne: $D_i \rightarrow \beta_l$ $(1 \le l \le s)$; a_k i β_l su nizovi nezavršnih i završnih znakova. Te produkcije se zamjene sa slijedećim produkcijama:

$D_i \rightarrow \beta_l$; $1 \le l \le s$
$D_i \rightarrow \beta_l C_i$; $1 \le l \le s$
$C_i \rightarrow a_k$; $1 \le k \le r$
$C_i \rightarrow a_k C_i$; $1 \le k \le r$

C_i je novi nezavršni znak. Preuređena gramatika generira isti jezik i nema lijevo rekurzivnih produkcija.

14.4. Koraci postizanja GNF

- Definicija algoritma
- Primjer

Algoritam pretvorbe produkcija u GNF se odvija u četiri koraka:

1) Produkcije gramatike G se preurede u CNF ($A \rightarrow BC$ i $A \rightarrow a$). Skup nezavršnih znakova gramatike G se zamijeni sa skupom nezavršnih znakova $\{D_1, D_2, \ldots, D_m\}$. Nakon

pretvorbe sve produkcije su oblika $D_i \rightarrow D_j D_k$ ili $D_i \rightarrow a$. Produkcije $D_i \rightarrow a$ su u GNF i u daljnjem postupku se ne preuređuju

2) Produkcije oblika $D_i \rightarrow D_j D_k$ se preurede u oblik $D_i \rightarrow D_j \beta$ (j>i). β je niz nezavršnih znakova. Postupak pretvorbe kreće od D_1. Tijekom postupka preuređivanja može se dogoditi da je j=i, j<i ili j>i. Ako je j=i tada se koristi algoritam razrješenja lijeve rekurzije (skup nezavršnih znakova se proširi sa novim nezavršnim znakom C_i), a ako je j<i onda se primjenjuje algoritam zamjene krajnjeg lijevog nezavršnog znaka.
3) U trećem koraku produkcije oblika $D_i \rightarrow D_j \beta$ se preurede u oblik $D_i \rightarrow a a \beta$ ($a \in T$, a,β su nizovi nezavršnih znakova). Postupak pretvorbe započinje s nezavršnim znakom D_{m-1} i ide prema D_1.
4) Preuređuju se produkcije koje imaju C_i na lijevoj strani. Takve produkcije s desne strane počinju isključivo nezavršnim znakovima iz skupa $\{D_1, D_2, \ldots, D_m\}$. Korištenjem zamjene krajnje lijevog nezavršnog znaka takve produkcije se preurede u GNF.

Nakon završetka ova četiri koraka sve produkcije su oblika $D_i \rightarrow a\beta$ ili $C_i \rightarrow a\beta$ ($a \in T$, β je niz nezavršnih znakova)

PRIMJER:

Neka je zadana gramatika G=({S,A,B}, {a,b}, P, S) i neka su produkcije gramatike preuređene u CNF.

1)$S \rightarrow AB$ 2)$A \rightarrow BS$ 3)$A \rightarrow b$ 4)$B \rightarrow SA$ 5)$B \rightarrow a$

U prvom koraku se odredi novi skup nezavršnih znakova { D_1, D_2, D_3}. Na način da se S zamijeni sa D_1, A sa D_2, B sa D_3. Produkcije se promjene na način:

1)$D_1 \rightarrow D_2 D_3$ 2)$D_2 \rightarrow D_3 D_1$ 3)$D_2 \rightarrow b$ 4)$D_3 \rightarrow D_1 D_2$ 5)$D_3 \rightarrow a$

Produkcije 3) i 5) su već u GNF. **U drugom koraku** se ostale produkcije preurede na oblik $D_i \rightarrow D_j \beta$ (j>i). Postupak započinje sa D_1. Produkcije 1) i 2) se ne preuređuju jer su već u tom obliku. Produkcija 4) se obrađuje na način da se krajnje lijevi nezavršni znak D_1 zamijeni s desnom stranom produkcije 1): $D_3 \rightarrow D_1 D_2$; $D_3 \rightarrow D_2 D_3 D_2$. Za novo dobivenu produkciju još uvijek vrijedi i < j pa se i ona obrađuje. Krajnje lijevi znak D_2 se može zamijeniti sa dvije produkcije 2) i 3), pa se produkcija $D_3 \rightarrow D_2 D_3 D_2$ mijenja sa skupom produkcija $D_3 \rightarrow D_3 D_1 D_3 D_2$ i $D_3 \rightarrow b D_3 D_2$.

Produkcija $D_3 \rightarrow D_3 D_1 D_3 D_2$ je lijevo rekurzivna pa se koristi algoritam rješavanja lijeve rekurzije. Definira se novi nezavršni znak C_3 i produkcija $D_3 \rightarrow D_3 D_1 D_3 D_2$ se zamijeni sa skupom produkcija $D_3 \rightarrow b D_3 D_2 C_3$, $D_3 \rightarrow a C_3$, $C_3 \rightarrow D_1 D_3 D_2$, $C_3 \rightarrow D_1 D_3 D_2 C_3$.

Dobiju se produkcije:

1)$D_1 \rightarrow D_2 D_3$ 2)$D_2 \rightarrow D_3 D_1$ 3)$D_2 \rightarrow b$ 4aaa)$D_3 \rightarrow b D_3 D_2 C_3$ 4aab)$D_3 \rightarrow a C_3$

4ab)$D_3 \rightarrow b D_3 D_2$ 4aac)$C_3 \rightarrow D_1 D_3 C_2$ 4aad)$C_3 \rightarrow D_1 D_3 D_2 C_3$ 5)$D_3 \rightarrow a$

U **trećem koraku** produkcije oblika $D_i \rightarrow D_j \beta$ se preurede u oblik $D_i \rightarrow a a \beta$. Postupak pretvorbe započinje s nezavršnim znakom D_2 (D_{m-1}) i ide prema D_1. U produkciji 2) nezavršni znak D_3 se mijenja desnim stranama produkcija 4aaa), 4aab), 4ab) i 5). Dobiva se skup produkcija:

1) $D_1 \rightarrow D_2 D_3$ 2a) $D_2 \rightarrow b D_3 D_2 C_3 D_1$ 2b) $D_2 \rightarrow a C_3 D_1$ 2c) $D_2 \rightarrow b D_3 D_2 D_1$ 2d) $D_2 \rightarrow a D_1$

3)$D_2 \rightarrow b$ 4aaa)$D_3 \rightarrow b D_3 D_2 C_3$ 4aab)$D_3 \rightarrow a C_3$ 4ab)$D_3 \rightarrow b D_3 D_2$ 4aac)$C_3 \rightarrow D_1 D_3 C_2$

4aad)$C_3 \rightarrow D_1 D_3 D_2 C_3$ 5)$D_3 \rightarrow a$

Postupak se nastavlja obradom znaka D_1 na isti način…

U **četvrtom koraku** produkcije koje na lijevim stranama imaju nezavršne znakove D_1,D_2,D_3 su u GNF. Preuređuju se produkcije koje imaju C_3 na lijevoj strani. Takve produkcije s desne strane počinju isključivo nezavršnim znakovima iz skupa {D_1, D_2, D_3}. Korištenjem zamjene krajnje lijevog nezavršnog znaka takve produkcije se preurede u GNF. (Produkcije 4aac)$C_3 \rightarrow D_1D_3\ C_2$ i 4aad)$C_3 \rightarrow D_1D_3\ D_2\ C_3$ se obrade na taj način).

15. RAZLAGANJE (PARSIRANJE NIZA)

15.1. Definicija razlaganja niza

- Postupak razlaganja niza
- Vrste razlaganja
- Razlaganje od vrha prema dnu

Određivanje pripadnosti niza w jeziku L(G) naziva se prepoznavanje niza. Objedinjeno prepoznavanje i gradnja generativnog stabla se naziva **parsiranje (razlaganje) niza.**

Metode parsiranja se razlikuju po načinu gradnje generativnog stabla:

- **Metoda parsiranja od vrha prema dnu** → započinje gradnju stabla s korijenom (početnim nezavršnim znakom) i ide prema listovima (završnim znakovima gramatike). Za gramatiku G = (V, T, P, S). Parsiranje započinje sa početnim nezavršnim znakom S, produkcije gramatike se primjenjuju sve dok se listovi stabla ne označe završnim znakovima traženog niza w, za višestruki izbor zamjene treba ispitati sve kombinacije.
- **Metoda parsiranje od dna prema vrhu** → počinje gradnju stabla od listova i ide prema korijenu.

15.2. LL(1) gramatika i razlaganje

- Definicija LL(1) gramatike i razlaganja
- Tehnika rekurzivnog spusta
- Parser s rekurzivnim spustom

Prvi „L“ u nazivu LL označava da se ulazni niz čita s lijeva na desno (Left-to-right scanning), a drugi „L“ označava da se produkcije primjenjuju na krajnji lijevi nezavršni znak (leftmost derivation). (1) označava da se čita jedan po jedan ulazni znak. Na temelju pročitanog znaka i krajnje lijevog nezavršnog znaka se primjeni produkcija gramatike.

Tehnika rekurzivnog spusta služi za programsko ostvarenje LL(k) parsera, koristi rekurzivno pozivanje potprograma, s time da potprograme dodjeljujemo nezavršnim znakovima.

U glavnom programu pročita se prvi znak niza w, pozove se potprogram pridružen početnom nezavršnom znaku, provjerava se posljednji očitani znak; ako je to oznaka kraja niza, niz se prihvaća. Za nezavršni znak gramatike koristi se potprogram; za svaku produkciju se gradi zasebni dio, svaki dio ispituje podniz koji slijedi, ako znak pročitan tijekom rada ne odgovara niti jednoj desnoj strani, niz se odbacuje.

15.3. Razlaganje od dna prema vrhu

- Definicija razlaganja od dna prema vrhu
- Primjena i primjer
- LR(k) razlaganje

Za gramatiku G = (V, T, P, S). Kod parsiranja od dna prema vrhu parsiranje započinje sa listovima, odnosno sa završnim znakovima gramatike. U nizu w odnosno u dobivenim međunizovima nastoji se prepoznati jedna od desnih strana produkcija. Ako je dio međuniza jednak desnoj strani produkcije, zamjenjuje se sa lijevom stranom. Tim zamjenama nastoji se doseći korijen stabla. Primjena: Postupak se koristi u generatorima parsera.

Primjer:

Zadana je gramatika G = ({E,T,F}, {var,+,*,(,)}, P, E) s produkcijama:

1) $E \rightarrow E + T$; 2) $E \rightarrow T$; 3) $T \rightarrow T * F$; 4) $T \rightarrow F$; 5) $F \rightarrow (E)$; 6) $F \rightarrow var$

Parsiramo niz var+var*var; niz čitamo s lijeva na desno i primijenimo redukciju 6): var+var*var $\Leftarrow$ F+var*var. Nadalje jednoznačno primijenimo redukcije 4) 2) 6) 4): F+var*var $\Leftarrow$ T+var*var$\Leftarrow$ E+var*var$\Leftarrow$ E+T*var. Dalji rad nije jednoznačan, jer je moguće primijeniti redukcije 1), 2) ili 6): E+T*var $\Leftarrow$ E*var $\Leftarrow$ E+E*var $\Leftarrow$ E+T*F. Napredak postižemo redukcijom 6), pa 3) i 1): E+T*var $\Leftarrow$ E+T*F $\Leftarrow$ E+T$\Leftarrow$ E

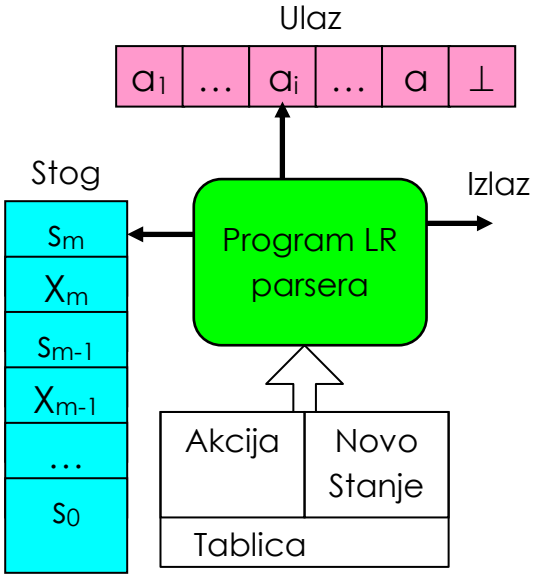
LR(k) parser koristi metodu parsiranja od dna prema vrhu (L → ulazni niz čitamo s lijeva na desno; R→ mijenja krajnje desni nezavršni znak, (k)→ potrebno je pročitati najviše k znakova unaprijed za donošenje odluke o primjeni redukcije). Parsiranje se provodi bez unazadnog pretraživanja; primjenjuje se na širok skup CFG, ali ne na sve.

15.4. LR(k) razlaganje

- Model LR parsera

- Stog i tablica razlaganja
- Konfiguracija LR parsera
- Algoritmi i program LR parsera

Dijelovi LR parsera su: ulazni spremnik, potisni stog(LIFO stog), program LR parsera, tablica parsiranja i izlaz.



Na **potisni stog** se sprema niz oblika $s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m$; s_m je na vrhu stoga. Znakovi X_i su završni ili nezavršni znakovi gramatike, dok su znakovi s_i stanja. Stanje na vrhu stoga jednoznačno određuje sadržaj stoga. Znakove X_i nije nužno stavljati na stog. **Tablica parsiranja** se sastoji od dva dijela: tablice Akcija i tablice NovoStanje. Na osnovu ulaznog znaka a_i i trenutnog stanja s_m iz tablice akcija odredi se akcija koja se izvodi nad ulaznim nizom i stogom. (Akcije: Pomak s; Reduciraj A→β; Prihvati, Odbaci)

Konfiguracija LE parsera je lista koja se dobije spajanjem trenutnog niza znakova na stogu i niza znakova u ulaznom spremniku koji su desno od a_i, s time da se te dvije liste međusobno odvoje ',' .

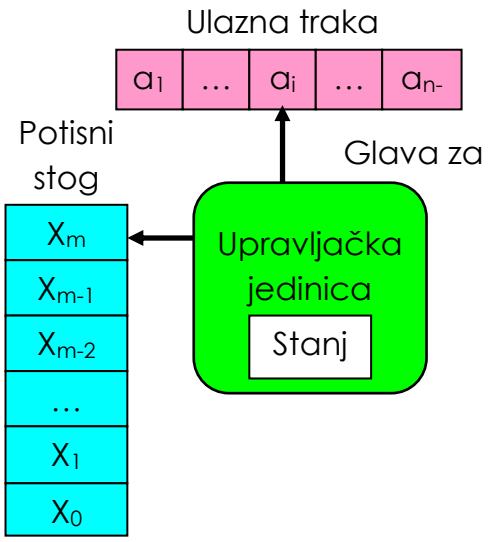
Ulaz **algoritma LR parsera** je zadani niz w i tablica LR parsera koja se dobije na osnovu zadane CFG G. Ako je $w \in L(G)$ tada LR parser ispiše da prihvaća niz, u suprotnom ga odbacuje. Program LR parsera čita znak po znak ulaznog spremnika i koristi postupak generiranja niza zamjenom krajnjeg desnog nezavršnog znaka. Na početku na stogu je početno stanje s0 a na ulazu niz w⊥. Program se izvodi sve dok se ne akcija Prihvati ili Odbaci.

16. POTISNI AUTOMA (PA)

16.1. Model i rad potisnog automata

- Model potisnog automata
- Rad potisnog automata
- Odluka o prihvaćanju niza

Potisni automat je proširenje konačnog automata za potrebe prihvata kontekstno neovisnih jezika. Konačnom automatu dodaje se potisni stog, automat čita ulazni znak i znak stoga.



Upravljačka jedinica donosi odluku o promjeni sadržaja stoga, pomaku glave za čitanje i promjeni stanja na osnovu stanja, znaka stoga i ulaznog znaka. Na vrh stoga može se staviti prazni niz ε, niz duljine jednog znaka ili više znakova. Upravljačka jedinica obavlja dvije vrste prijelaza:

- na temelju trojke (q, a, Z) mijenja stanje u p, pomakne glavu i zamijeni Z sa nizom γ
- na temelju trojke (q, ε, Z) mijenja stanje u p, ostavi glavu na istom mjestu i zamijeni Z sa nizom γ

Potisni automat odluku o prihvaćanju niza donosi nakon čitanja svih znakova niza, i to na jedan od dva načina:

- PA M prihvaća niz prihvatljivim stanjem (ASPA) - ako uđe u prihvatljivo stanje niz se prihvaća, PA M prihvaća jezik L(M)
- PA M prihvaća niz praznim stogom (ESPA) - ako je stog prazan niz se prihvaća, PA M prihvaća jezik N(M)

16.2. Formalna definicija potisnog automata

- Formalna definicija
- Funkcija prijelaza
- Konfiguracija PA

Potisni automat formalno zadajemo sedmorkom PA = $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ gdje su:

- Q konačan skup stanja s početnim stanjem q_0
- Σ konačan skup ulaznih znakova (ulazna abeceda)
- Γ konačan skup znakova stoga (abeceda stoga) s početnim znakom stoga Z_0
- **δ funkcija prijelaza** $Q \times (\Sigma \cup \{ \varepsilon \}) \times \Gamma \rightarrow Q \times \Gamma^*$ pridružuje trojci (q, a, Z) konačni skup parova (p, γ): $\delta(q, a, Z) = \{ (p_1, \gamma_1), (p_2, \gamma_2), \ldots (p_m, \gamma_m) \}$
- $F \subseteq Q$ podskup prihvatljivih stanja

Ako je automat u stanju q, pročita ulazni znak a i na vrhu stoga je Z. Preći će u stanje p_i, zamijeniti Z nizom γ_i s desna na lijevo, te pomaknuti glavu na slijedeći znak ulaznog niza.

Konfiguracija PA se zadaje stanjem, nepročitanim dijelom ulaznog niza i znakovima koji su na stogu. Konfiguracija PA je uređena trojka (q, w, γ) gdje je q stanje, w nepročitani dio ulaznog niza, a γ sadržaj stoga.

16.3. Deterministički i nedeterministički PA

- Prihvaćanje jezika
- Nedeterministički PA
- Deterministički PA

PA M = (Q, Σ, Γ, δ, q_0, Z_0, F) prihvaća jezik:

1)prihvatljivim stanjem ako vrijedi:

$$L(M)=\left\{w \middle| (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \gamma); \quad p \in F, \gamma \in \Gamma^*\right\}$$

2)praznim stogom ako vrijedi:

$$N(M)=\left\{w \middle| (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon); \quad p \in Q,\right\}$$

Prva vrsta PA prihvaća jezik ako vrijedi p$\in$F, dok druga vrsta prihvaća jezika ako se stog u potpunosti isprazni nakon pročitanih svih znakova niza w.

Nedeterminizam PA sličan je nedeterminizmu NKA: postoji li mogućnost izbora više prijelaza, započinje istodobno s radom više determinističkih PA. Uspije li barem jedan deterministički PA isprazniti stog čitanjem svih znakova ulaznog niza, niz se prihvaća.

PA M = (Q, Σ, Γ, δ, q_0, Z_0, F) jest deterministički ako su ispunjena OBA uvjeta:

- ako je $\delta(q, \varepsilon, Z) \neq \varnothing$, tada $\delta(q, a, Z) = \varnothing$ (sprječava izbor između prijelaza i ε prijelaza)
- u skupu $\delta(q, \varepsilon, Z)$ je najviše jedan element (garantira jednoznačnost prijelaza)

Nedeterministički PA prihvaća širu klasu jezika od determinističkog PA.

17. TRANSFORMACIJE POTISNOG AUTOMATA

17.1. Konstrukcija ESPA iz ASPA

- istovjetnost PA
- pristup konstrukciji ESPA iz ASPA
- koraci konstrukcije ESPA iz ASPA
- primjer

Dva PA su **istovjetna** ako i samo ako prihvaćaju isti kontekstno neovisni jezik. Dva PA mogu biti istovjetna bez obzira prihvaćaju li jezik prihvatljivim stanjem ili praznim stogom.

Za zadani PA M_2 = (Q, Σ, Γ, δ, q_0, Z_0, F) koji prihvaća jezik prihvatljivim stanjem konstruiramo istovjetni PA M_1 koji prihvaća jezik praznim stogom. Konstrukcija se zasniva na simulaciji. Uđe li M_2 u prihvatljivo stanje, M_1 isprazni stog. Koristi se posebni znak stoga **X_0**. Isprazni li M_2 stog bez da uđe u prihvatljivo stanje,M_1 neće prihvatiti niz zbog znaka X_0. Konstruiramo PA $M_1 = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \varnothing)$. Prihvati li M_2 niz prihvatljivim stanjem,M_1 će ga prihvatiti praznim stogom.

Primjer:

Ako je zadan neki ASPA M_2 koji prihvaća jezik prihvatljivim stanjem. Konstruiramo istovjetni ESPA M_1,tako što:1- nadopunimo funkciju prijelaza;2-preuzmemo sve prijelaze M_2;3-dodamo ε-prijelaze u stanje q_e ;4- dodamoε-prijelaze koji prazne stog. Kod ESPA M_1 niz je prihvaćen jer je pročitan a stog prazan. Kod ASPA M_2 niz je prihvaćen jer je pročitan,stanje je prihvatljivo.

17.2. Dokaz istovjetnosti ESPA i ASPA

- priprema simulacije
- simulacija rada ASPA
- pražnjenje stoga

Dokaz istovjetnosti PA M_1 i PA M_2 se provodi u dva dijela: u prvom dijelu se pokazuje da PA M_1 prihvaća niz x ako ga prihvaća PA M_2,a u drugom obrnuto. Za početnu konfiguraciju PA M_1 vrijedi:

$\left(q_0', x, X_0\right) \underset{M_1}{\succ} \left(q_0, x, Z_0 X_0\right)$ (korak 1). Svi prijelazi PA M_2 su ujedno i prijelazi PA M_1

$\left(q_0, x, Z_0 X_0\right) \overset{*}{\underset{M_1}{\succ}} \left(q, \varepsilon, A\gamma X_0\right)$ (korak 2). Budući da je $q \in F$ korak (3) omogućuje prijelaz u stanje q_e ,a s vrha stoga se uzima jedan znak. Korak (4) omogućuje pražnjenje stoga:

$\left(q_e, \varepsilon, \gamma X_0\right) \overset{*}{\underset{M_1}{\succ}} \left(q_e, \varepsilon, \varepsilon\right)$ (korak 4). Slijed prijelaza PA M_1 tijekom prihvaćanja niza x čine prijelazi zadani u koraku(1),zatim oni u koraku(2) koji simuliraju rad PA M_2 te slijed prijelaza koji prazni stog.

Na osnovu koraka (3) i (4) stog se prazni ako i samo ako je u konstrukciji $(q,\varepsilon,\gamma,X_o)$ stanje **q** je prihvatljivo stanje. Prema tome PA M_2 prihvaća niz prihvatljivim stanjem ako i samo ako ga PA M_1 prihvaća praznim stogom.

17.3.Konstrukcija ASPA iz ESPA

- koraci
- dokaz istovjetnosti
- primjer

Konstrukcija: Za zadani PA $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ koji prihvaća jezik praznim stogom,konstruiramo istovjetni PA M_2 koji prihvaća jezik prihvatljivim stanjem. Konstrukcija se zasniva na simulaciji. Isprazni li M_1 stog, M_2 uđe u prihvatljivo stanje. Konstruiramo PA $M_2 = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \{q_f\})$. Pročita li se X_o to je znak da je stog prazan i PA prelazi u prihvatljivo stanje q_f. PA M_2 obavlja prijelaze:

$$\left(q_0', x, X_0\right) \underset{M_2}{\succ} \left(q_0, x, Z_0 X_0\right) \overset{*}{\underset{M_2}{\succ}} \left(q, \varepsilon, X_0\right) \underset{M_2}{\succ} \left(q_f, \varepsilon, \varepsilon\right)$$

Prvi prijelaz osigurava početne uvjete. Dalji niz prijelaza simulira rad zadanog ESPA i na kraju prijeđe u prihvatljivo stanje. M_2 prihvaća niz **x** prihvatljivim stanjem ako i samo ako M_1 prihvaća niz praznim stogom.

Primjer:

ESPA M_1 prihvaća jezik praznim stogom. Konstruiramo istovjetni ASPA M_2 . Nadopunjujemo funkciju prijelaza tako da preuzimamo sve prijelaze od M_1.Dodajemo ε-prijelaze u prihvatljivo stanje q_f. Kod ASPA M_2 niz je prihvaćen jer je pročitan,stanje je prihvatljivo. Kod ESPA M_1 niz je prihvaćen jer je pročitan,a stog je prazan.

18. POTISNI AUTOMAT I CFG

18.1. Konstrukcija ESPA iz CFG

- pristup sintezi
- postupak sinteze
- primjer

-koristimo samo konstrukciju ESPA

- ASPA dobijemo lako na osnovu istovjetnosti

- Nedeterministički PA, NPA, prepozna klasu kontekstno neovisnih jezika.

-Za bilo koji CFL L postoji NPA M koji ga prihvaća praznim stogom: N(M)=L

-pretpostavljamo da prazni niz ε nije element jezika L.

-jezik je zadan gramatikom G = (V, T, P, S) s produkcijama u standardnom obliku Greibacha oblika:

$A \rightarrow a\alpha$; $A \in V, a \in T, \in \alpha V^*$

-Konstruiramo **PA M = ({q}, Σ, Γ, δ, q, S, ∅)**

-q je jedino stanje, ujedno i početno

-skup ulaznih znakova jednak je skupu završnih znakova,Σ = T

-skup znakova stoga jednak je skupu nezavršnih znakova, Γ = V

-početni znak stoga jednak je početnom znaku gramatike S

-skup prihvatljivih stanja je prazan skup F = ∅

-PA M prihvaća praznim stogom

Funkcija prijelaza δ se definira:

$\delta(q, a, A)$ sadrži (q, γ) samo ako postoji produkcija $\mathbf{A \rightarrow a\gamma}$.

-PA M simulira postupak generiranja niza zamjenom krajnjeg lijevog nezavršnog znaka.

-PA prihvaća niz x praznim stogom samo ako G generira x.

Primjer:

Ako imamo zadanu gramatiku koristeći pravila izgradimo PA M. Gramatika generira neki niz a PA M ga prihvaća praznim stogom.

18.2. Sinteza CFG iz ESPA

- postupak gradnje gramatike
- primjer

Izgradnja:

-Koristimo samo konstrukciju iz ESPA

-ASPA pretvorimo prvo u ESPA na osnovu istovjetnosti

-Za zadani PA M = (Q, Σ, Γ, δ, q0, Z0, ∅), konstruira se gramatika

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

- Nezavršni znakovi gramatike označeni su [q,A,p] ∈ V ; q,p∈Q, A∈Γ
- U skup nezavršnih znakova dodaje se S

-Gradi se skup produkcija

-Dobivena gramatika G:

- Simulira rad PA M postupkom zamjene krajnjeg lijevog znaka.
- Niz nezavršnih znakova u međunizu jednak je nizu znakova stoga PA M.
- Gramatika počinje generirati **x** iz [q,A,p] ako i samo ako PA M čitanjem **x** izbriše **A** i promjeni stanje iz **q** u **p**.

-dokaz L(G) = N(M) slijedi iz:

$$[q, A, p] \underset{G}{\overset{*}{\Longrightarrow}} x \text{ ako i samo ako } (q, x, A) \underset{M}{\overset{*}{\succ}} (p, \varepsilon, \varepsilon)$$

Primjer:

1. Za zadani PA M koji prihvaća jezik,konstruira se gramatika.
2. Gradnja produkcija: p
    - počinje iz S;
    - ostale produkcije grade se isključivo za dohvatljive znakove;
    - redoslijed gradnje zasniva se na traženju dohvatljivih znakova.

-Gramatika generira niz nizom transformacija a PA M ga prihvaća nizom konfiguracija.

19.SVOJSTVA KONTEKSNO NEOVISNIH JEZIKA

19.1. Kontekstno neovisni jezik i PA

- primjer jezika koji nije KNJ
- položaj KNJ
- istovjetnost KNJ i PA

Jezik L je kontekstno neovisan ako i samo ako postoji PA koji ga prihvaća.

Jezik L = $\{a^i b^i c^i \mid i \geq 1\}$ je ***primjer jezika koji nije KN jezik***, jer za taj jezik nije moguće izgraditi PA.

-Položaj kontekstno neovisnih jezika:

Kontekstno neovisni jezici su pravi podskup svih jezika:

Skup svih jezika $2^{\Sigma*}$ nad abecedom Σ

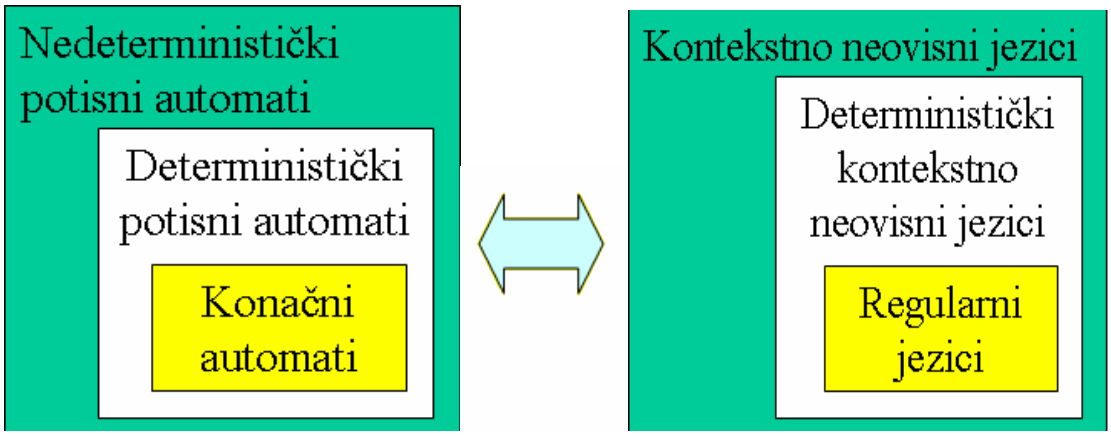
L = $\{w w^R\}$ je primjer jezika koji :

- za koji je moguće izgraditi PA (NPA)
- za koji **nije** moguće izgraditi DPA

-Deterministički kontekstno neovisni jezici su **pravi podskup** skupa kontekstno neovisnih jezika.

-Istovjetnost KNJ i PA:

- DKA je poseban slučaj DPA koji ne koristi stog
- stoga su regularni jezici **pravi podskup** skupa deterističkih kontekstno neovisnih jezika
- definiramo hijerarhiju automata i jezika:



19.2.Svojstva zatvorenosti KNJ na uniju, nadovezivanje

- dokaz zatvorenosti na uniju
- dokaz zatvorenosti na nadovezivanje

Istovjetnost kontekstno neovisne gramatike, kontekstno neovisne jezika i potisnih automata koristi se za opis svojstva zatvorenosti KNJ.

-Dokaz zatvorenosti na uniju:

- dokažimo **prvo** da je $L(G1) \cup L(G2) \subseteq L(G3)$
- obzirom na $P3 = P1 \cup P2 \cup \{S3 \rightarrow S1 \mid S2\}$ vrijedi:
  - pretpostavimo da je $w \in L(G1)$

$$S_3 \underset{G_3}{\Longrightarrow} S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} w$$

  - slično za $w \in L(G2)$

$$S_3 \underset{G_3}{\Longrightarrow} S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} w$$

- zaključujemo: $w \in L(G1) \cup L(G2) \Rightarrow w \in L(G3)$
- dokažimo **drugo** da je $L(G3) \subseteq L(G1) \cup L(G2)$
- obzirom na $P3 = P1 \cup P2 \cup \{S3 \rightarrow S1 \mid S2\}$
  gramatika G3 generira w na jedan od dva načina:

$$S_3 \underset{G_3}{\Longrightarrow} S_1 \overset{*}{\underset{G_3}{\Longrightarrow}} w \quad \text{ili} \quad S_3 \underset{G_3}{\Longrightarrow} S_2 \overset{*}{\underset{G_3}{\Longrightarrow}} w$$

- obzirom da je $V1 \cap V2 = \varnothing$
  - za S1 koristimo P1, za S2 koristimo P2
- zaključujemo: $w \in L(G3) \Rightarrow w \in L(G1) \cup L(G2)$
- slijedi: $L(G1) \cup L(G2) = L(G3)$

-Dokaz zatvorenosti na nadovezivanje:

- dokaz je sličan kao za uniju
- niz generiramo postupkom:

$$S_4 \underset{G_4}{\Longrightarrow} S_1 S_2 \overset{*}{\underset{G_1}{\Longrightarrow}} w_1 S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} w_1 w_2$$

- gdje su $w1w2 \in L(G4)$; $w1 \in L(G1)$ i $w2 \in L(G2)$

19.3.Svojstva zatvorenosti KNJ na Kleene, supstituciju

- dokaz zatvorenosti na Kleene
- dokaz zatvorenosti na supstituciju, primjer

-Dokaz zatvorenosti na Kleene:

- dokaz se zasniva na dva postupka generiranja niza
- 1. postupak:

$$S_5 \underset{G_5}{\Longrightarrow} S_1S_5 \overset{*}{\underset{G_1}{\Longrightarrow}} w_1S_5 \underset{G_5}{\Longrightarrow} w_1S_1S_5 \overset{*}{\underset{G_1}{\Longrightarrow}} w_1w_1S_5\Lambda \overset{*}{\underset{G_1}{\Longrightarrow}} w_1^+S_5 \underset{G_5}{\Longrightarrow} w_1^+$$

gdje su w1+ ∈ L(G5) i w1 ∈ L(G1)

- 2. postupak:

$$S_5 \underset{G_5}{\Longrightarrow} \varepsilon$$

-Dokaz zatvorenosti na supstituciju:

-Neka

- G = (V, T, P, S) generira L(G)
- svi završni znakovi $\mathbf{a_i}$ jezika L(G) zamijene nizovima L(Gi)
  $\mathbf{1 \le i \le k}$, k= | T |

- gramatika Gi = (Vi, Ti, Pi, Si) generira L(Gi)
- konstruira se gramatika G' koja generira nastali jezik L':
  - V' = V1∪ V2∪ .... Vk, V∩Vi = ∅, Vi∩Vj = ∅, za sve i,j
  - T' = T1∪ T2∪ .... Tk
  - S' = S
  - P' = P1∪ P2∪ .... Pk ,

-Primjer :

- neka je zadan jezik **L** u kojem nizovi imaju jednak broj znakova **a** i **b**
- neka jezik **L** generira gramatika G = ({S}, {a, b}, P, S) ;znak a zamijenimo nizovima jezika L1
- neka jezik **L1** generira gramatika G1 = ({S1}, {0, 1}, P, S1) ; znak b zamijenimo nizovima L2
- jezik L2 generira gramatika G2 = ({S2}, {0, 2}, P, S2)
- zatim zamjenom znakova a i b nastaje jezik L'
- slijedi da jezik L' generira gramatika G'

19.4. Zatvorenost presjeka KNJ i RJ

- dokaz zatvorenosti presjeka KNJ i RJ
- primjer

-Dokaz zatvorenosti presjeka KNJ i RJ:

- presjek KNJ i RJ **jest** KNJ
- pretpostavimo
  - da KNJ L1 prihvaća PA M1 = (Q_1, Σ, Γ, δ_1, q_0, Z_1, F_1)
  - da RJ L2 prihvaća DKA M2 = (Q_2, Σ, δ_2, p_0, F_2)
- moguće je izgraditi PA M' = (Q', Σ, Γ, δ', q'$_0$, Z_0, F')
  koji prihvaća jezik L = L1 $\cap$ L2
- PA M' simulira rad M1 i M2
- dokaz da je L = L1 $\cap$ L2 izvodi se indukcijom
- dokaže se da je:

$$([p_0,q_0],w,Z_0) \overset{i}{\underset{M'}{\vdash}} ([p,q],\varepsilon,\gamma)$$

ako i samo ako je:

$$(q_0,w,Z_0) \overset{i}{\underset{M_1}{\vdash}} (q,\varepsilon,\gamma) \quad \text{i} \quad \delta_2(p_0,w)=p$$

gdje je [p, q] $\in$ F' ako i samo ako p $\in$ F2 i q $\in$ F1

PRIMJER

- PA M1 = ({q1,q2}, {0,1}, {N,K}, δ1, q1, K, {q2})
  prihvaća jezik L(M1) prihvatljivim stanjem q2
- DKA M2 = ({p1, p2, p3}, {0,1}, δ2, p1, K, {p3})
  prihvaća jezik L(M2) s barem dva znaka 1
- presjek jezika L(M1) i L(M2) jest jezik L3 = L(M1) $\cap$ L(M2)
- prihvaća ga M3 = (Q', Σ, Γ, δ', q'0, Z0, F')
- PA M1 prihvaća neki niz prihvatljivim stanjem q2
- DKA M2 također prihvaća taj niz

PA M' prihvaća taj niz prihvatljivim stanjem [p3,q2]

20. SVOJSTVO NAPUHAVANJA KNJ

20.1. Svojstvo napuhavanja KNJ

- pristup preko gramatike
- analiza generiranja niza
- oblici generiranja niza

– Svojstvo napuhavanja KNJ koristi se za dokazivanje kontekstne neovisnosti jezika
- zasniva se

  - na broju čvorova generativnog stabla
  - na broju nezavršnih članova gramatike

– za dovoljno dugački niz
  - broj unutrašnjih čvorova veći je od kardinalnog broja skupa V
  - znači da je više čvorova označeno istim nezavršnim znakom

-Neka gramatika g = (V, T, P, S) generira stablo koje ima više čvorova od kardinalnog broja skupa V. Tada sigurno postoji put stabla u kojem je jedan nezavršni znak barem na dva mjesta pri dnu stabla. Za takvo stablo postoji slijed generiranja niza:

$$S \underset{G}{\overset{*}{\Rightarrow}} uAy \underset{G}{\overset{*}{\Rightarrow}} uvAxy \underset{G}{\overset{*}{\Rightarrow}} uvwxy$$

-Nezavršni znak A koristi se dva puta u postupku generiranja niza uvwxy.

-Gramatika generira niz oblika:

$$S \underset{G}{\overset{*}{\Rightarrow}} uAy \underset{G}{\overset{*}{\Rightarrow}} uvAxy \underset{G}{\overset{*}{\Rightarrow}} uvvAxxy \underset{G}{\overset{*}{\Rightarrow}} uvvvAxxxy \underset{G}{\overset{*}{\Rightarrow}} A$$

$$\underset{G}{\overset{*}{\Rightarrow}} uv^iAx^iy \underset{G}{\overset{*}{\Rightarrow}} uv^iwx^iy$$

-Da je jezik KNJ dokazuje se korištenjem svojstva:

– neka je L KNJ, postoji konstanta n ovisna o L:
  - ako je za niz z $|z| \geq n$, z pišemo kao uvwxy
  - $|vx| \geq 1$; $|vwx| \leq n$; za $i \geq 1$ niz $uviwxiy \in L$