11. NEJEDNOZNAČNOST GRAMATIKE, JEZIKA I NIZA

11. 1. Nejednoznačnost niza

- Kontekstno neovisni jezika i gramatika
- Nejednoznačnost interpretacije niza
- Redoslijedi primjene produkcija i zamjene nezavršnih znakova

Jezik je **kontekstno neovisan** ako postoji **kontekstno neovisna gramatika** koja ga generira. Time je definirana istovjetnost kontekstno neovisnih jezika i gramatika. Klasa kontekstno neovisnih jezika sadrži kao podskup regularne jezike.

Interpretacija niza se zasniva na generativnom stablu. Za neke nizove je moguće izgraditi više stabala, pa jer **interpretacija niza nejednoznačna** jer ne možemo odrediti generativno stablo.

Npr. $G = (\{E\}, \{a,\otimes\}, \{E \rightarrow E\otimes E|a\}, E)$.
Za niz $a\otimes a\otimes a$ moguće je izgraditi dva generativna stabla.

Različito stablo dobijemo:

- Promjenom redoslijeda produkcija
- Promjenom redoslijeda nezavršnih znakova

Redoslijed zamjene nezavršnih znakova je značajan za potupak generiranja niza.
Uvodimo postupak zamjene krajnjeg desnog ili krajnjeg lijevog nezavršnog znaka.

- Kod **zamjene krajnjeg lijevog** nezavršnog znaka produkcije primjenjujemo samo na krajnji lijevi nezavršni znak u izrazu.
  → radi djelomično, dodaje višak stvari na desnu stranu

- Kod **zamjene krajnjeg desnog** nezavršnog znaka produkcije primjenjujemo samo na krajnji desni nezavršni znak u izrazu.
  → radi djelomično, dodaje višak stvari na početak

11. 2. Sistematizacija zamjene

- Sistematizacija zamjene nezavršnih znakova
- Obilazak stabla
- Definicija nejednoznačnosti gramatike, niza i jezika
- Razrješenje jednoznačnosti

Redoslijed zamjene nezavršnih znakova je značajan za postupak generiranja niza.
Uvodimo postupak zamjene krajnjeg desnog ili krajnjeg lijevog nezavršnog znaka.

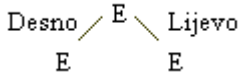
- Kod **zamjene krajnjeg lijevog** nezavršnog znaka produkcije primjenjujemo samo na krajnji lijevi nezavršni znak u izrazu.
  → radi djelomično, dodaje višak stvari na desnu stranu

- Kod **zamjene krajnjeg desnog** nezavršnog znaka produkcije primjenjujemo samo na krajnji desni nezavršni znak u izrazu.
  → radi djelomično, dodaje višak stvari na početak

Postupak obilaska stabla određuje redoslijed grana i čvorova. Koristimo **desni obilazak** ili **lijevi obilazak**. Obilazak započinje korijenom i

- kod desnog obilazi rekurzivno sve desne čvorove i grane
- kod lijevog obilazi rekurzivno sve lijeve čvorove i grane

„Desno“ i „lijevo“ se određuju pogledom sa korijena:



Postupak obilaska za neko stablo određuje redoslijed primjene produkcija i redoslijed zamjene znakova. Desni obilazak definira zamjenu krajnje lijevog znaka, dok lijevi obilazak definira zamjenu krajnje desnog znaka.

Gramatika G je **nejednoznačna**:

- ako je za niz $w \in L(G)$ moguće izgraditi više različitih generativnih stabala
- ako je niz $w \in L(G)$ moguće generirati primjenom više postupaka zamjene krajnjeg desnog ili krajnje lijevog znaka

Ako je za niz *w* moguće izgraditi više različitih generativnih stabala, onda je **niz nejednoznačan**.
Ako jezik L nije moguće generirati niti jednom jednoznačnom gramatikom, onda je **jezik L nejednoznačan.**

Nejednoznačnost se razrješava:

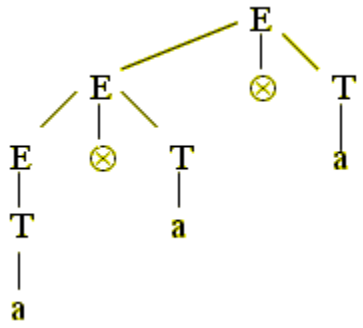
- promjenom gramatike: umjesto gramatike G se izgradi nova jednoznačna G'.
- promjenom jezika: umjesto jezika L izgradi se novi jezik L' kojeg je moguće generirati jednoznačnom gramatikom.

11. 3. Promjena gramatike

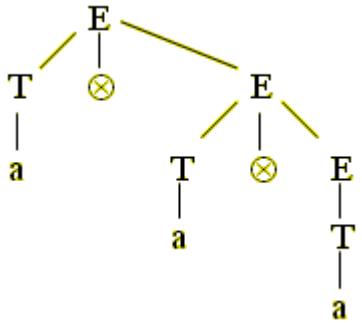
- Definicija promjene gramatike
- Primjer
- Svojstva promjene gramatike

Umjesto gramatike G se izgradi nova jednonzačna gramatika G'.
Jezik L kojeg generira G = ({E}, {a,⊗}, {E→E⊗E|a}, E) moguće je generirati više različitih jednoznačnih gramatika:

- Za lijevo asocijativni ⊗ uvodimo gramatiku:
  G_1 = ({E,T}, {a,⊗}, {E→E⊗T|T, T→a}, E)



- Za desno asocijativni ⊗ uvodimo gramatiku:
  G_2 = ({E,T}, {a,⊗}, {E→T⊗E|T, T→a}, E)



Ovisno o izboru jednoznačne gramatike, određuje se način gradnje generativnog stabla.
Promjenom gramatike ne mijenja se jezik i odbacuje se višestruko značenje niza.

11. 4. Promjena jezika

- Definicija promjene jezika
- Primjer
- Svojstva promjene jezika

Umjesto jezika L izgradi se novi 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 je uvođenje zagrada, zagrade su završni znakovi gramatike i dio su niza. Za gramatiku G = ({E}, {a,⊗}, {E→E⊗E|a}, E) moguće su dvije interpretacije niza a⊗a⊗a. Dodavanjem zagrada dobivamo gramatiku
G_3 = ({E}, {a,⊗,(,)}, {E→(E)⊗(E)|a}, E) zbog čega su nizovi (a)⊗((a)⊗(a)) i ((a)⊗(a))⊗(a) jednoznačni.

Promjenom jezika čuva se višestruko značenje niza i definira se zaseban niz za svako značenje.

12. POJEDNOSTAVLJENJE GRAMATIKE

12. 1. Definicija pojednostavljenja gramatike

- Svrha pojednostavljenja
- Željena svojstva gramatike
- Definicija beskorisnosti znaka

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

- Bilo koji znak koristi se u makar jednom nizu
- Ne koriste se jedinične produkcije A → B
- Ne koriste se ε-produkcije A→ε

Koristimo algoritme odbacivanja beskorisnih znakova, odbacivanja jediničnih produkcija i ε-produkcija, te algoritme postizanja normalnih oblika Chomskog i Greibacha.

Ako se znak X gramatike G = (V, T, P, S) koristi u postupku generiranja:

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

onda je X koristan, u suprotnom je **beskoristan.**
Dva su vida beskorisnosti:

- Ako iz znaka X nije moguće generirati niz završnih znakova onda je znak X **mrtav**, u suprotnome je živ.

- Ako znak X nije ni u jednom nizu koji se generira iz S, onda je znak X **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) koja generira neprazan jezik L(G) ≠ ∅ moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja nema **mrtvih znakova** tako da vrijedi:

$$A \overset{*}{\Rightarrow} w \;;\; w \in T^*$$

Ako su živi znakovi desne strane produkcija, živ je i nezavršni znak s lijeve strane produkcije. Iz $A \rightarrow X_1X_2\ldots X_n$ i $X_i \rightarrow w_i$ slijedi $A \rightarrow w_1w_2\ldots w_n = w$
Prvo se pronađu svi živi znakovi i njihovim odbacivanjem se dobije lista mrtvih znakova.
Algoritam se provodi u tri koraka:

- U listu živih znakova se stave lijeve strane produkcija koje na desnoj strani nemaju nezavršnih znakova.
- Ako su s desne strane produkcije isključivo živi znakovi, onda se u listu doda znak s lijeve strane
- Ako se lista živih znakova ne može proširiti, svi znakovi koji nisu na listi su mrtvi znakovi.

Za bilo koju CFG G = (V, T, P, S) koja generira neprazan jezik L(G) ≠ ∅ moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja nema **nedohvatljivih znakova** tako da vrijedi:

$$X \in V' \cup T' : \quad S \overset{*}{\Rightarrow} \alpha X \beta \;;\; \alpha, \beta \in (V' \cup T')^*$$

Ako je dohvatljiv nezavršni znak s lijeve strane produkcije, onda su dohvatljivi svi završni i nezavršni znakovi s desne strane produkcije.
Prvo se pronađu dohvatljivi znakovi i njihovim odbacivanjem se dobije lista nedohvatljivih znakova.
Algoritam se provodi u tri koraka:

- U listu dohvatljivih znakova stavi se početni nezavršni znak gramatike
- Ako je znak s lijeve strane produkcije dohvatljiv, u listu se dodaju svi znakovi s desne strane produkcije.
- Ako se lista dohvatljivih ne može proširiti, svi znakovi koji nisu na listi su nedohvatljivi znakovi

Nužno je prvo primijeniti algoritam odbacivanja mrtvih znakova, a zatim algoritam odbacivanja nedohvatljivih znakova jer neki znakovi postanu nedohvatljivi tek nakon odbacivanja mrtvih znakova.

12. 3. Odbacivanje ε-produkcija i jediničnih produkcija

- Odbacivanje ε-produkcija
- Odbacivanje jediničnih produkcija

Za bilo koju CFG G = (V, T, P, S) koja generira jezik L(G) \ {ε} moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja **nema ε-produkcija** $A \rightarrow \varepsilon$.

Algoritam se izvodi u dva osnovna koraka:

- Pronađu se svi nazavršni znakovi koji generiraju prazni niz $A \overset{*}{\Rightarrow} \varepsilon$
  - U listu praznih znakova stave se sve lijeve strane ε-produkcija
  - Ako su svi znakovi desne strane u listi, lista se nadopuni lijevom
  - Algoritam se nastavlja sve dok se lista može širiti
- Gradi se novi skup produkcija gramatike G'
  - Za produkciju iz G: $A \rightarrow X_1 X_2 ... X_n$ dodaju se u G' produkcije $A \rightarrow \xi_1 \xi_2 ... \xi_n$
  - Oznake ξ_i poprimaju vrijednosti X_i ako je X_i neprazan, i X_i ili ε ako je X_i prazan
  - Kada svi ξ_i poprime vrijednost ε nastaje ε-produkcija koja se NE dodaje u listu produkcija G'
  - Produkcije se grade na temelju svih mogućih kombinacija $\xi_1 \xi_2 ... \xi_n$

Za bilo koju CFG G = (V, T, P, S) koja generira neprazan jezik L(G) \ {ε} moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja **nema jediničnih produkcija** A→B.

Algoritam se provodi u dva koraka:

- U P' se stave sve produkcije iz P koje nisu jedinične
- Za sve postupke generiranja B iz A $A \overset{*}{\Rightarrow} B$ na osnovu $B \rightarrow \alpha$ stvore se nove produkcije $A \rightarrow \alpha$

13. NORMALNI OBLIK CHOMSKOG

13. 1. Definicija normalnog oblika Chomskog

- Definicija
- Postupak pojednostavljenja gramatike u CNF

Za bilo koju CFG G = (V, T, P, S) koja generira jezik L(G) \ {ε} moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja **ima sve produkcije** oblika:

A → BC ili A → a

Uz pretpostavku da G nema beskorisnih znakova, ε-produkcija niti jediničnih produkcija, algoritam pretvorbe u CNF se provodi u tri koraka:

1. U skup P' se stave sve produkcije koje su u CNF, tj.
   A → BC ili A → a
   a u skup V' upišu se svi nezavršni znakovi.

2. Za produkcije gramatike G oblika $A \rightarrow X_1X_2\ldots X_n$ X_i su završni ili nezavršni znakovi. Ako je X_i završni znak a, skup nezavršnih znakova se proširi sa C_a, a skup produkcija se proširi sa $C_a \rightarrow a$ koja je u CNF. Svi završni znakovi a se zamijene sa C_a. Postupak se nastavlja dok se ne zamijene svi završni znakovi i nastavlja se za sve produkcije.
3. Nakon drugog koraka, sve produkcije u P' su oblika $A \rightarrow a$ ili $A \rightarrow B_1B_2B_3\ldots B_m$, a produkcije oblika $A \rightarrow BC$ ili $A \rightarrow a$ su u CNF. Prudukcije koje s desna imaju 3 ili više znakova mijenjaju se novim produkcijama tako da se definiraju znakovi $D_1D_2D_3\ldots D_{m-2}$ pa se $A \rightarrow B_1B_2B_3\ldots B_m$ zamijeni skupom produkcija:
$\{A \rightarrow B_1D_1, D_1 \rightarrow B_2D_2, D_2 \rightarrow B_3D_3, \ldots D_{m-2} \rightarrow B_{m-1}B_m\}$

13. 2. Postupak pojednostavljenja gramatike u CNF

- Postupak pojednostavljenja gramatike u CNF
- Primjer

Uz pretpostavku da G nema beskorisnih znakova, ε-produkcija niti jediničnih produkcija, algoritam pretvorbe u CNF se provodi u tri koraka:

1. U skup P' se stave sve produkcije koje su u CNF, tj.
$A \rightarrow BC$ ili $A \rightarrow a$
a u skup V' upišu se svi nezavršni znakovi.
2. Za produkcije gramatike G oblika $A \rightarrow X_1X_2\ldots X_n$ X_i su završni ili nezavršni znakovi. Ako je X_i završni znak a, skup nezavršnih znakova se proširi sa C_a, a skup produkcija se proširi sa $C_a \rightarrow a$ koja je u CNF. Svi završni znakovi a se zamijene sa C_a. Postupak se nastavlja dok se ne zamijene svi završni znakovi i nastavlja se za sve produkcije.
3. Nakon drugog koraka, sve produkcije u P' su oblika $A \rightarrow a$ ili $A \rightarrow B_1B_2B_3\ldots B_m$, a produkcije oblika $A \rightarrow BC$ ili $A \rightarrow a$ su u CNF. Prudukcije koje s desna imaju 3 ili više znakova mijenjaju se novim produkcijama tako da se definiraju znakovi $D_1D_2D_3\ldots D_{m-2}$ pa se $A \rightarrow B_1B_2B_3\ldots B_m$ zamijeni skupom produkcija:

$\{A \rightarrow B_1D_1, D_1 \rightarrow B_2D_2, D_2 \rightarrow B_3D_3, \ldots D_{m-2} \rightarrow B_{m-1}B_m\}$

primjer: $G = (\{S, A\}, \{a,b\}, P, S)$

1) $A \rightarrow bAA$ 2) $A \rightarrow aS$ 3) $A \rightarrow a$

- produkcija 3 je u CNF
- definiraju se C_a i C_b, te dodaju produkcije $C_a \rightarrow a$ i $C_b \rightarrow b$:

1) $A \rightarrow C_bAA$ 2) $A \rightarrow C_aS$ 3) $A \rightarrow a$

- sada su 2 i 3 u CNF
- treba razriješiti produkciju 1

- definira se D_1, te doda produkcija $D_1 \rightarrow AA$
  1a) $A \rightarrow C_b D_1$ 1b) $D_1 \rightarrow AA$ 2) $A \rightarrow C_a S$ 3) $A \rightarrow a$

- sada su sve produkcije u CNF

14. NORMALNI OBLIK GREIBACHA

14. 1. Definicija normalnog oblika Greibacha

- Definicija
- Postupak pojednostavljenja gramatike u GNF

Za bilo koju CFG G = (V, T, P, S) koja generira jezik L(G) \ {ε} moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G) = L(G'), koja **ima sve produkcije** oblika:

$$A \rightarrow a\alpha; \quad \alpha \in V^*$$

Koriste se tri postupka:

- Algoritmom pretvorbe gramatike u normalni oblik Chomskog
- Algoritam zamjene krajnjeg lijevog nezavršnog znaka
- Algoritam razrješenja lijeve rekurzije

14. 2. Algoritam zamjene krajnje lijevog znaka

- Definicija algoritma
- Primjer

Neka je CFG G = (V, T, P, S). G ima r produkcija koje imaju D_j s lijeve strane:
$D_j \rightarrow \alpha_1$, $D_j \rightarrow \alpha_2$ … $D_j \rightarrow \alpha_r$
i G ima produkciju koja ima D_j na krajnjem lijevom mjestu desne strane: $D_i \rightarrow D_j\gamma$.
Prethodnih *r+1* produkcija zamijeni se sa *r* produkcija: $D_i \rightarrow \alpha_1\gamma$, $D_i \rightarrow \alpha_2\gamma$... $D_i \rightarrow \alpha_r\gamma$.
Produkcije generiraju isti jezik.

Primjer:

1) $D_1 \rightarrow aB$ 2) $D_1 \rightarrow bC$ 3) $D_1 \rightarrow cD$ 4) $D_2 \rightarrow D_1E$

Zamijenimo *r+1* (u ovome slučaju 4) produkcija sa *r* (3) produkcija:

1) $D_2 \rightarrow aBE$ 2) $D_2 \rightarrow bCE$ 3) $D_2 \rightarrow cDE$

14. 3. Algoritam razrješenja lijeve rekurzije

- Definicija algoritma
- Primjer

Produkcija je **lijevo rekurzivna** ako je isti nezavršni znak na lijevoj strani produkcije i krajnje lijevo na desnoj strani produkcije. Gramatika se preuređuje:

- Neka je za D_i zadano r lijevo rekurzivnih produkcija:
  $D_i \rightarrow D_i\alpha_1$, $D_i \rightarrow D_i\alpha_2$ ... $D_i \rightarrow D_i\alpha_r$
- Zadano je i s produkcija koje nisu lijevo rekurzivne:
  $D_i \rightarrow \beta_1$, $D_i \rightarrow \beta_2$ ... $D_i \rightarrow \beta_s$
- Prethodnih r+s produkcija zamijeni se sa 2(r+s) produkcija:
  $D_i \rightarrow \beta_t$, $D_i \rightarrow \beta_t C_i$, $C_i \rightarrow \alpha_k$, $C_i \rightarrow \alpha_k C_i$, $t = 1\ldots s, k = 1\ldots r$

C_i je novi nezavršni znak, dok su α_k i β su nizovi nezavršnih i završnih znakova. Dobivene produkcije generiraju isti jezik i nema lijevo rekurzivnih produkcija.

Primjer: 1) $D_1 \rightarrow D_2D_3$ 2) $D_2 \rightarrow D_3D_1$ 3) $D_2 \rightarrow b$ 4) $D_3 \rightarrow D_1D_2$ 5) $D_3 \rightarrow a$

Produkcije 3 i 5 su već u GNF-u, dok su produkcije 1 i 2 OK jer jer j>i. Produkciju 4 transformiramo zamjenom sa 1, pa 2 i 3:

$$D_3 \rightarrow D_1D_2 \Rightarrow \ldots \Rightarrow D_3 \rightarrow D_3D_1D_3D_2 \mid bD_3D_2.$$

Dobivena produkcija $D_3 \rightarrow bD_3D_2$ je OK, dok dobivenu produkciju $D_3 \rightarrow D_3D_1D_3D_2$ transformiramo razrješenjem lijeve rekurzije:

$$D_3 \rightarrow D_3D_1D_3D_2 \Rightarrow D_3 \rightarrow bD_3D_2C_3 \mid aC_3 \quad \text{i} \quad C_3 \rightarrow D_1D_3D_2 \mid D_1D_3D_2C_3$$

14. 4. Koraci postizanja GNF

- Definicija algoritma
- Primjer

Algoritam pretvorbe produkcija u GNF provodi se u četiri koraka:

1. Produkcije gramatike G preurede se u CNF produkcije oblika A → BC ili A→a. Preimenuje se m nezavršnih znakova u D_i , i = 1…m čime produkcije poprime oblik: $D_i \rightarrow D_jD_k$ ili $D_i \rightarrow a$. Produkcije $D_i \rightarrow a$ su u GNF.

2. Produkcije oblika $D_i \rightarrow D_jD_k$ za i=1…m preurede se u oblik $D_i \rightarrow D_j\,\beta$, $j \geq i$
   Postupak pretvorbe kreće od D_1. Za $D_i \rightarrow D_jD_k$, i>1 moguće je slijedeće:
   - Za j<i preuredimo zamjenom lijevog nezavršnog znaka; zamjenjujemo iterativno dok ne postignemo $j \geq i$
   - Za j=i preuredimo razrješenjem lijeve rekurzije, pojavi se C_i
   - Za j>i produkcija je u prihvatljivom obliku

   Dobijemo produkcije oblika $D_i \rightarrow D_j\beta$, $D_i \rightarrow a\beta$, $C_i \rightarrow D_j\xi$ i $D_m \rightarrow a\alpha$.

3. Produkcije oblika $D_i \rightarrow D_j\beta$ za i = m-1…1 preurede se u oblik $D_i \rightarrow a\alpha\beta$. Sve produkcije su oblika $D_i \rightarrow a\beta$, i $C_i \rightarrow D_j\xi$

4. Produkcije oblika $C_i \to D_j\xi$ preurede se u oblik $C_i \to a\beta$. Sve produkcije su oblika $D_i \to a\beta$, i $C_i \to a\beta$.

Primjer:

G = ({S, A, B}, {a,b}, P, S) zadana u CNF.

1) S→AB 2) A→ BS 3) A→ b 4) B → SA 5) B → a

Korak 1: Gramatika je već u CNF, zamijenimo S, A i B sa D_1, D_2 i D_3:
1) $D_1 \to D_2D_3$ 2) $D_2 \to D_3D_1$ 3) $D_2 \to b$ 4) $D_3 \to D_1D_2$ 5) $D_3 \to a$
Korak 2: Produkcije 3 i 5 su već u GNF, $D_2 \to b$ i $D_3 \to a$. Produkcija 1) $D_1 \to D_2D_3$ je OK, j>i i produkcija 2) $D_2 \to D_3D_1$ je OK, j>i. Produkciju 4 transformiramo zamjenom sa 1, pa 2 i 3: $D_3 \to D_1D_2 \Rightarrow D_3 \to D_2D_3D_2 \Rightarrow D_3 \to D_3D_1D_3D_2 \mid bD_3D_2$. Dobivena produkcija $D_3 \to bD_3D_2$ je OK, dok dobivenu produkciju $D_3 \to D_3D_1D_3D_2$ transformiramo razrješenjem lijeve rekurzije:
$D_3 \to D_3D_1D_3D_2 \Rightarrow D_3 \to bD_3D_2C_3 \mid aC_3$ i $C_3 \to D_1D_3D_2 \mid D_1D_3D_2C_3$
Korak 3: Počinjemo sa D_2, dobije se:
$D_2 \to D_3D_1 \Rightarrow D_2 \to aD_1 \mid bD_3D_2D_1 \mid bD_3D_2C_3D_1 \mid aC_3D_1$, a od ranije imamo $D_2 \to b$.
Nastavimo sa D_1, dobije se: $D_1 \to D_2D_3 \Rightarrow D_1 \to bD_3 \mid aD_1D_3 \mid bD_3D_2D_1D_3 \mid$
$\mid bD_3D_2C_3D_1D_3 \mid aC_3D_1D_3$
Od ranije imamo za D_3 i C_3.
Korak 4: dovodimo $C_3 \to D_1D_3D_2 \mid D_1D_3D_2C_3$ u GNF zamjenom:
$C_3 \to D_1D_3D_2 \Rightarrow C_3 \to bD_3D_3D_2 \mid aD_1D_3D_3D_2 \mid bD_3D_2D_1D_3D_3D_2 \mid bD_3D_2C_3D_1D_3D_3D_2 \mid$
$aC_3D_1D_3D_3D_2$
i
$C_3 \to D_1D_3D_2C_3 \Rightarrow C_3 \to bD_3D_3D_2C_3 \mid aD_1D_3D_3D_2C_3 \mid bD_3D_2D_1D_3D_3D_2C_3 \mid$
$bD_3D_2C_3D_1D_3D_3D_2C_3 \mid aC_3D_1D_3D_3D_2C_3$

Dobivena gramatika je u GNF.

15. RAZLAGANJE (PARSIRANJE) NIZA

15. 1. Definicija razlaganja niza

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

Određivanje pripadnosti niza w jeziku L(G) naziva se prepoznavanje niza. **Parsiranje** je postupak prepoznavanja niza i izgradnje generativnog stabla. To je stablo parsiranja. Razlikujemo metode:

- **Od vrha prema dnu** započinje korijenom, tj. od početnog nezavršnog znaka i ide prema listovima (završnim znakovima gramatike). Zadana je gramatika G = (V, T, P, S). Parsiranje započinje početnim nezavršnim znakom S. Produkcije

gramatike se primjenjuju sve dok se listovi stabla ne označe završnim znakovima traženog niza w. Ovo parsiranje se široko primjenjuje zbog učinkovitosti. Za višestruki izbor zamjene treba ispitati sve kombinacije.

- **Od dna prema vrhu** započinje listovima, odnosno od završnih znakova i ide prema korijenu.

15. 2. LL(1) gramatika i razlaganje

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

LL(1) gramatika i parsiranje od vrha prema dnu je:
L – ulazni niz čitamo s lijeva na desno
L – mijenjamo krajnje lijevi nezavršni znak
(1) – odluku donosimo na osnovu jednog pročitanog znaka.
Produkcija se primijeni na osnovu pročitanog ulaznog znaka i krajnje lijevog nezavršnog znaka u generiranom nizu.
Primijeni se ona produkcija koja ima na lijevoj strani nezavršni znak koji je krajnje lijevi u nizu i koja ima na desnoj strani na krajnje lijevom mjestu završni znak koji odgovara ulaznom nizu.

U realizaciji LL(1) parsera primjenjujemo tehniku **rekurzivnog spusta**. Ova tehnika služi za programsko ostvarenje LL(k) parsera (odluku donosi na osnovu k pročitanih znakova) koristeći rekurzivno pozivanje potprograma. Potprograme dodjeljujemo nezavršnim znakovima. Potprogram ispituje da li podniz ulaznog niza odgovara desnoj strani produkcije nezavršnog znaka kojem pripada. Za nezavršne znakove s desne strane poziva odgovarajuće potprograme. Ako je isti znak s lijeve i desne strane produkcije, potprogram se poziva rekurzivno.

Parser s rekurzivnim spustom:
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; ako je više produkcija, za svaku se gradi zasebni dio, a svaki dio ispituje podniz koji slijedi. Ako znak pročitan tijekom rada ne odgovara niti jednoj desnoj strani, niz se odbacuje.
U dijelu potprograma za svaki završni znak niza ispita se da li odgovara, ako ne odgovara, niz se odbacuje. Za svaki nezavršni znak na krajnjem lijevom mjestu niza poziva se potprogram (bez čitanja niza), a za svaki nezavršni znak koji nije na krajnjem lijevom mjestu niza prvo se pročita sljedeći znak, pa se poziva potprogram.

15. 3. Razlaganje od dna prema vrhu

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

Zadana je gramatika G = (V, T, P, S). Parsiranje započinje sa listovima, odnosno sa završnim znakovima gramatike. U nizu w, tj. u dobivenim međunizovima, nastoji se prepoznati jedna od desnih strana produkcija. Ako je dio međuniza jednak desnoj strani produkcije, zamjenjuje se s lijevom stranom. Tim zamjenama nastoji se doseći korijen stabla. Primjena obrnutih produkcija su **redukcije**. Postupak se koristi u generatorima parsera.

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

1) E → E + T 2) E → T 3) T → T * F 4) T → F 5) F → **(**E**)** 6) F → **var**

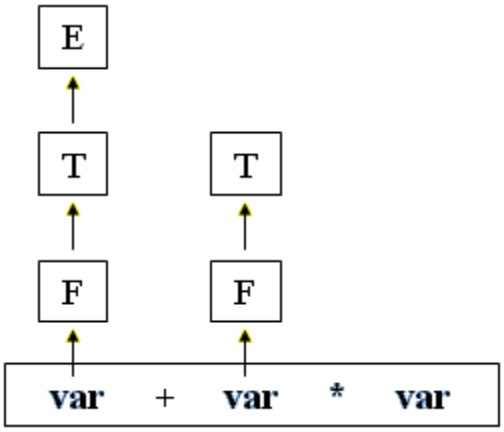
Parsiramo niz: **var+var*var**

1. Niz čitamo s lijeva na desno i primijenimo redukciju 6) :

var+var*var⇐ F+**var*var**

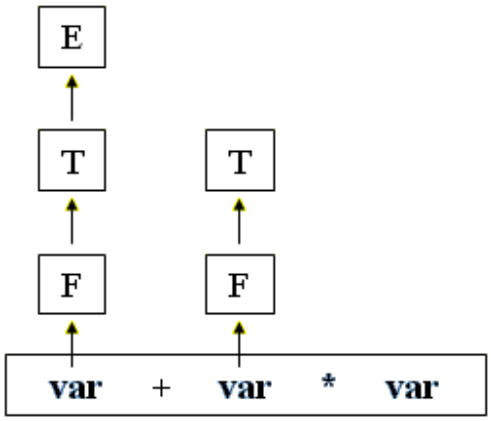
2. Jednoznačno primijenimo redukcije 4) 2) 6) 4) tj. :

F+**var*var**⇐ T+**var*var**
⇐ E+**var*var**
⇐ E+T***var**



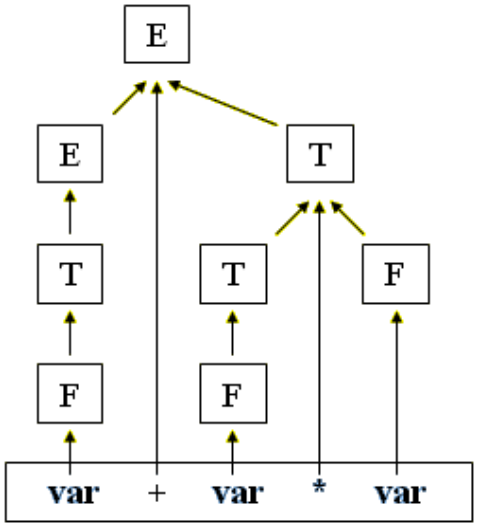
3. Dalji rad nije jednoznačan, moguće je primijeniti redukcije 1), 2) ili 6), tj. :

E+T***var*** ⇐ E***var***
⇐ E+E***var***
⇐ E+T*F



4. Napredak postižemo redukcijom 6), pa 3) i 1), tj. :

E+T***var*** ⇐ E+T*F
⇐ E+T
⇐ E



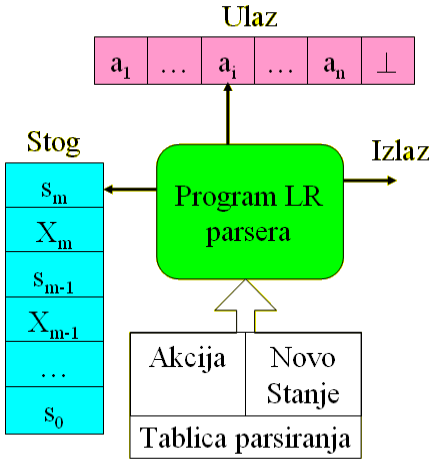
LR(k) parser koristi metodu od dna prema vrhu.
L – ulazni niz čitamo s lijeva na desno
R – mijenja krajnje desni nezavršni znak
(k) – za donošenje odluke potrebno je pročitati najviše k znakova ulaznog niza.
LR(k) je najopćenitiji postupak parsiranja od dna prema vrhu i provodi se bez unazadnog pretraživanja. Primjenjuje se na širok skup CFG, ali ne na sve. To su LR gramatike, jer je moguće izgraditi LR parser. LR(k) parser se gradi posebnim generatorom koji koristi skup produkcija, a postoje različite klase složenosti. Gradi se tablica parsiranja, program parsera je uvijek isti.

15. 4. LR(k) razlaganje

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

Model LR parsera ima:

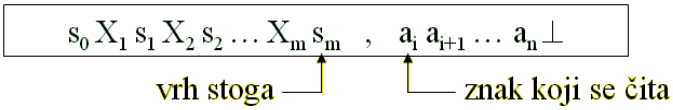
- ulazni spremnik
- potisni LIFO stog
- program LR parsera
- tablicu parsiranja
- izlaz.



U stog se sprema niz $s_0X_1s_1X_2s_2\ldots X_ms_m$; s_m je na vrhu stoga. X_i su završni ili nezavršni znakovi gramatike, s_i su stanja. Tijekom rada, parser koristi stanja s_i , znakovi X_i se pamte radi lakšeg praćenja rada.
Tablica parsiranja ima dva dijela: program čita znak a_i i stanje s_m i bira redak tablice parsiranja:

- Određuje akciju (Pomak s, Reduciraj $A \rightarrow \beta$, Prihvati, Odbaci)
- Određuje novo stanje

Spojimo stog i ulazni spremnik u jedinstvenu listu - to je **konfiguracija** LR parsera.



Algoritam LR parsera:
Ulaz: zadani niz w i tablica parsiranja
Izlaz: prihvaćanje ili neprihvaćanje niza
Postupak parsiranja: Ako je $w \in L(G)$ 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 je na stogu početno stanje s_0, a na ulazu niz $w\perp$. Program se izvodi sve dok se ne dobije izlaz prihvati ili odbaci.

Program LR parsera

```
ProgramLRParsera()
{ postavi kazaljku na početka niza
        dok(1)          //ponavljaj zauvijek
        {slučaj(Akcija[s,a])
                {Pomak s':    stavi a na stog
```

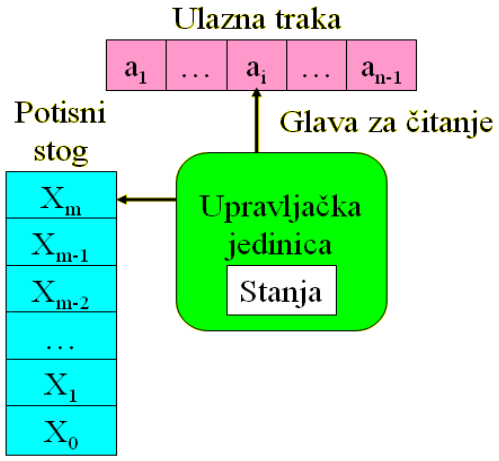
```
                    stavi s' na stog
                    pomakni kazaljku
            Reduciraj A → β:
                    uzmi sa stoga 2*|β|znakova
                    stavi A na stog
                    stavi NovoStanje[s', A] na stog
            Prihvati:   Ispiši("niz prihvaćen");kraj
            Ostalo:     Ispiši("niz nije prihvaćen");
                    kraj
            }
        }
}
```

16. POTISNI AUTOMAT (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, a automat čita ulazni znak i znak stoga.



Upravljačka jedinica donosi odluku o promjeni sadržaja stoga, pomaku glave za čitanje i promjeni stanja. Odluku donosi na osnovu stanja, znaka stoga i ulaznog znaka. Moguće su odluke na osnovu samo dva parametra: stanja i znaka stoga – tada se glava za čitanje ne pomiče u desno.
Na vrh stoga može se staviti:

- Prazni niz ε jednako je uzimanju znaka sa vrha stoga
- Niz duljine jednog znaka, isti kao prije ili neki drugi
- Niz duljine 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 nakon čitanja **svih** znakova niza PA donosi na jedan od načina:

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

Generalno vrijedi L(M) <> N(M)

16. 2. Formalna definicija potisnog automata

- Formalna definicija
- Funkcija prijelaza
- Konfiguracija PA

Potisni automat formalno zadajemo sedmorkom: PA = (Q, Σ, Γ, δ, q_0, Z_0, F) gdje su:

- **Q** je konačan skup stanja s početnim stanjem $\mathbf{q_0}$
- **Σ** je konačan skup ulaznih znakova (ulazna abeceda)
- **Γ** je konačan skup znakova stoga (abeceda stoga) s početnim znakom stoga $\mathbf{Z_0}$
- **δ** je funkcija prijelaza $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$
- $\mathbf{F \subseteq Q}$ podskup prihvatljivih stanja

Uobičajena značenja su: *a, b, c* ulazni znakovi, *v, w, x* nizovi ulaznih znakova, *A, B, C* znakovi stoga, α, β nizovi znakova stoga.

Funkcija prijelaza **δ** pridružuje trojci (q, a, Z) konačni skup parova (p, γ):

$$\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots (p_m, \gamma_m)\}$$

Ako je automat u stanju q, pročita ulazni znak a i na vrhu stoga je Z. Prijeći će u stanje p_i, zamijeniti Z nizom γ_i s desna na lijevo, te pomaknuti glavu na sljedeći znak ulaznog niza. Funkcija **δ** je nejednoznačna, a ako je zadano preslikavanje za **δ(q, ε, Z)** glava za čitanje se ne pomakne.

Konfiguracija PA je trojka (q, w, γ):

- Stanje
- Nepročitani dio ulaznog niza
- Sadržaj stoga (vrh je krajnje lijevi znak niza γ)

Uz zadani PA M = (Q, Σ, Γ, δ, q_0, Z_0, F), ako je δ(q, a, Z) = {(p,ß), …}, konfiguracija se mijenja iz {q, aw, Zα) u (p, w, ßα) i pišemo relacije $\succ$:

$$(\mathbf{q, aw, Z\alpha}) \underset{\mathbf{M}}{\succ} (\mathbf{p, w, ß\alpha})$$

16. 3. Deterministički i nederministički PA

- Prihvaćanje jezika
- Nedeterministički PA
- Deterministički PA

Prihvaćanje jezika za PA definira se na dva načina. PA M = (Q, Σ, Γ, δ, q_0, Z_0, F) :

- prihvaća **prihvatljivim stanjem** jezik:

$$L(M) = \left\{ w \middle| (\mathrm{q}_0, \mathrm{w}, Z_0) \overset{*}{\succ} (\mathrm{p}, \varepsilon, \gamma); \quad p \in F, \gamma \in \Gamma^* \right\}$$

- prihvaća **praznim stogom** jezik:

$$N(M) = \left\{ w \middle| (\mathrm{q}_0, \mathrm{w}, Z_0) \overset{*}{\succ} (\mathrm{p}, \varepsilon, \varepsilon); \quad p \in Q, \right\}$$

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

- 1) δ(q1, 0, N) = {(q1, NN), (q2, ε)}
- 2) δ(q1, 1, J) = {(q1, JJ), (q2, ε)}

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

- ako je δ(q, ε, Z) ≠ ∅ , tada δ(q, a, Z) = ∅
- u skupu δ(q, ε, Z) je najviše jedan element

Prvi uvjet sprječava izbor između prijelaza i ε prijelaza. Drugi uvjet garantira jednoznačnost prijelaza. PA prihvaća **širu klasu jezika** u odnosu na DPA (NKA prihvaća istu klasu kao i DKA). Pod PA podrazumijeva se nedeterministički PA. Deterministički PA označavamo s DPA.

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 prihvaćaju isti kontekstno neovisni jezik. PA mogu biti istovjetni bez obzira prihvaćaju li jezik prihvatljivim stanjem ili praznim stogom.
Neka PA $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ 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
- Koristimo posebni znak stoga X_0
- Isprazni li M_2 stog bez ulaska u prihvaljivo stanje, M_1 neće prihvatiti niz zbog znaka X_0

Konstruiramo PA $M_1 = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \varnothing)$. Funkcija prijelaza PA M_1 gradi se:

1. $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$ prelazi u početnu konfiguraciju kao M_2
2. u skup $\delta'(q, a, Z)$ stave se svi elementi od $\delta(q, a, Z)$, računa se za sve q iz Q, a iz $\Sigma \cup \{\varepsilon\}$ i za sve Z iz Γ.
3. u skup $\delta'(q, \varepsilon, Z)$, $q \in F$, doda se ε–prijelaz (q_e, ε)
4. u skup $\delta'(q_e, \varepsilon, Z)$, $q \in F$, doda se ε–prijelaz (q_e, ε), time se prazni stog

Primjer:
Zadan je ASPA $M_2 = (\{q_1, q_2\}, \{0,1\}, \{N, K\}, \delta, q_1, K, \{q_2\})$
1) $\delta(q_1, 0, K) = \{(q_1, NK)\}$ 2) $\delta(q_1, 0, N) = \{(q_1, NN)\}$
3) $\delta(q_1, 1, N) = \{(q_2, \varepsilon)\}$ 4) $\delta(q_2, 1, N) = \{(q_2, \varepsilon)\}$

M_2 prihvatljivim stanjem prihvaća jezik $L(M_2) = \{0^n1^m \mid n \geq 1, m \geq 1, m \leq n\}$. Konstruiramo istovjetni ESPA $M_1 = (\{q_1, q_2, q_0', q_e\}, \{0,1\}, \{N, K, X_0\}, \delta', q_0', X_0, \varnothing)$.

Korak (1) dopunjuje funkciju prijelaza: 0) $\delta'(q_0', \varepsilon, X_0) = \{(q_1, KX_0)\}$
Korak (2) preuzima sve prijelaze M_2:
1) $\delta'(q_1, 0, K) = \{(q_1, NK)\}$ 2) $\delta'(q_1, 0, N) = \{(q_1, NN)\}$
3) $\delta'(q_1, 1, N) = \{(q_2, \varepsilon)\}$ 4) $\delta'(q_2, 1, N) = \{(q_2, \varepsilon)\}$
Korak (3) dodaje ε–prijelaze u stanje q_e:
5) $\delta'(q_2, \varepsilon, N) = \{(q_e, \varepsilon)\}$
6) $\delta'(q_2, \varepsilon, K) = \{(q_e, \varepsilon)\}$
7) $\delta'(q_2, \varepsilon, X_0) = \{(q_e, \varepsilon)\}$

Korak (4) dodaje ε–prijelaze koji prazne stog
8) $\delta'(q_e, \varepsilon, N) = \{(q_e, \varepsilon)\}$
9) $\delta'(q_e, \varepsilon, K) = \{(q_e, \varepsilon)\}$
10) $\delta'(q_e, \varepsilon, X_0) = \{(q_e, \varepsilon)\}$

Kod ESPA M_1 niz je prihvaćen jer je pročitan, a stog je prazan. Kod ASPA M_2 niz je prihvaćen jer je pročitan, stanje je prihvatljivo.

17. 2. Dokaz istovjetnosti ESPA iz ASPA

- Priprema simulacije
- Simulacija rada ASPA
- Pražnjenje stoga

M_1 i M_2 su istovjetni:

– M_1 prihvaća niz ako ga prihvati M_2
– M_2 prihvaća niz ako ga prihvati M_1

M_2 prihvati niz prihvatljivim stanjem:

$$(q_0, x, Z_0) \underset{M_2}{\overset{*}{\succ}} (q, \varepsilon, A\gamma); \ q \in F$$

M_1 u prvom koraku prelazi u početnu konfiguraciju kao M_2:

$$(q_0', x, X_0) \underset{M_1}{\succ} (q_0, x, Z_0 X_0)$$

U drugom koraku, M_1 radi isto kao i M_2:

$$(q_0, x, Z_0 X_0) \underset{M_1}{\overset{*}{\succ}} (q, \varepsilon, A\gamma X_0)$$

U trećem koraku (doda se se ε–prijelaz (q_e, ε) u skup $\delta'(q, a, Z)$, $q \in F$) M_1 prijeđe u stanje q_e, a s vrha uzme jedan znak:

$$(q, \varepsilon, A\gamma X_0) \underset{M_1}{\succ} (q_e, \varepsilon, \gamma X_0)$$

U četvrtom koraku (doda se ε–prijelaz (q_e, ε) u skup $\delta'(q_e, \varepsilon, Z)$, $q \in F$) M_1 isprazni stog:

$$(q_e, \varepsilon, \gamma X_0) \underset{M_1}{\overset{*}{\succ}} (q, \varepsilon, \varepsilon)$$

Dakle, prihvati li M_2 niz prihvatljivim stanjem, M_1 će ga prihvatiti praznim stogom. M_1 prihvati niz praznim stogom ako je u koraku 1) dodao X_0, u koraku 2) simulirao rad M_2, a koraci 3) i 4) mogući su samo ako je bilo $q \in F$ (tj. stog se prazni ako i samo ako je **q** prihvatljivo stanje). Dakle, M_2 prihvaća niz prihvatljivim stanjem ako i samo ako ga je M_1 prihvatio praznim stogom.

17. 3. Konstrukcija ASPA iz ESPA

- Pristup konstrukciji ASPA iz ESPA
- Koraci konstrukcije ASPA iz ESPA
- Dokaz istovjetnosti
- Primjer

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.
Kontruiramo PA $M_2 = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \{q_f\})$.
Funkcija prijelaza PA M_2 gradi se:

1) $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$ prelazi u početnu konfiguraciju kao M_1, X_0 indicira dno stoga
2) U skup $\delta'(q, a, Z)$ stave se svi elementi od $\delta(q, a, Z)$
3) U skup $\delta'(q, \varepsilon, X_0)$ doda se ε–prijelaz (q_f, ε)

Pročita li se X_0, to je znak da je stog prazan i PA prelazi u prihvatljivo stanje q_f.
M_2 obavi prijelaze:

$$(q_0', x, X_0) \underset{M_2}{\succ} (q_0, x, Z_0X_0) \overset{*}{\underset{M_1}{\succ}} (q, \varepsilon, X_0) \underset{M_2}{\succ} (q_f, \varepsilon, \varepsilon)$$

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

Primjer:
PA $M_1 = (\{q_1, q_2\}, \{0, 1\}, \{N, J, K\}, \delta, q_1, K, \varnothing)$ prihvaća jezik praznim stogom.

1) $\delta(q_1, 0, K) = \{(q_1, NK)\}$	2) $\delta(q_1, 1, K) = \{(q_1, JK)\}$
3) $\delta(q_1, 0, N) = \{(q_1, NN), (q_2, \varepsilon)\}$	4) $\delta(q_1, 1, N) = \{(q_1, JN)\}$
5) $\delta(q_1, 0, J) = \{(q_1, NJ)\}$	6) $\delta(q_1, 1, J) = \{(q_1, JJ), (q_2, \varepsilon)\}$
7) $\delta(q_2, 0, N) = \{(q_2, \varepsilon)\}$	8) $\delta(q_2, 1, J) = \{(q_2, \varepsilon)\}$
9) $\delta(q_1, \varepsilon, K) = \{(q_2, \varepsilon)\}$	10) $\delta(q_2, \varepsilon, K) = \{(q_2, \varepsilon)\}$

M_1 prihvaća jezik ww^R, w je niz $(0+1)^*$.

Konstruiramo istovjetni ASPA $M_2 = (\{q_1, q_2, q_0', q_f\}, \{0,1\}, \{N, J, K, X_0\}, \delta', q_0', X_0, \{q_f\})$
Korak (1) dopunjuje funkciju prijelaza: 0) $\delta'(q_0', \varepsilon, X_0) = \{(q_1, KX_0)\}$.
Korak (2) preuzima sve prijelaze od M_1 (iste kao kod M_1, samo što se piše δ' umjesto δ).
Korak (3) dodaje ε-prijelaze u prihvatljivo stanje q_f:

11) $\delta'(q_1, \varepsilon, X_0) = \{(q_f, \varepsilon)\}$
12) $\delta'(q_2, \varepsilon, X_0) = \{(q_f, \varepsilon)\}$

Za ASPA M_2 niz je pročitan, stanje je prihvatljivo. Za 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 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:

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

Kontruira se PA M = ({q}, Σ, Γ, δ, q, S, ∅).

- PA M ima samo jedno stanje q koje je ujedno i početno stanje.
- Σ = T, skup ulaznih znakova jednak je skupu završnih znakova
- Γ = V, skup znakova stoga jednak je skupu nezavršnih znakova
- Početni znak stoga jednak je početnom znaku gramatike S
- Skup prihvatljivih stanja je prazan skup, F = ∅.
- PA M prihvaća praznim stogom.

F-ja prijelaza δ definira se: δ(q, a, A) sadrži (q, γ) samo ako postoji produkcija $A \rightarrow a\gamma$.

PA M simulira postupak generiranja niza zamjenom krajnjeg lijevog nezavršnog znaka. Bilo koji generirani međuniz je oblika xα. Nakon čitanja niza x, na stogu je niz α. PA prihvaća x praznim stogom samo ako G generira x.

Primjer: Zadana je gramatika G = ({S, A}, {a, b}, {$S \rightarrow aAA$, $A \rightarrow aS|bS|a$}, S).
Koristeći pravila izgradi se PA M = ({q}, {a, b}, {S, A}, δ, q, S, ∅):

- δ(q,a,S) = {(q,AA)}
- δ(q,a,A) = {(q,S), (q,ε)}
- δ(q,b,A) = {(q,S)}

Gramatika generira abaaaa:
$S \Rightarrow aAA \Rightarrow abSA \Rightarrow abaAAA \Rightarrow abaaAA \Rightarrow abaaaA \Rightarrow abaaaa$
PA M prihvaća abaaaa praznim stogom:
$(q, abaaaa, S) \succ (q, baaaa, AA) \succ (q, aaaa, SA) \succ (q, aaa, AAA) \succ (q, aa, AA) \succ (q, a, A) \succ (q, \varepsilon, \varepsilon)$

18. 2. Sinteza CFG iz ESPA

- Postupak gradnje gramatike
- Primjer

Koristimo samo konstrukciju iz ESPA, ASPA pretvorimo prvo u ESPA na osnovu istovjetnosti. Za zadani PA M = (Q, Σ, Γ, δ, q_0, Z_0, ∅) konstruira se gramatika

G = (V, T, P, S). Nezavršni znakovi gramatike označeni su [q, A, p] $\in$V, q,p $\in$Q, A$\in$ Γ. U skup nezavršnih znakova dodaje se S.
Gradi se skup produkcija u 2 koraka:

1. Za početno stanje q_0, Z_0 i sva stanja iz Q gradi se S $\rightarrow$ [q_0, Z_0, q] – n produkcija
2. Ako skup δ(q,a,A) sadrži (q_1, $B_1B_2...B_m$) gradi se:
   [q, A, q_{m+1}] $\rightarrow$ a[q_1, B_1, q_2][q_2, B_2, q_3]... [q_m, B_m, q_{m+1}] – za svaku funkciju prijelaza gradi se n^{m-1} produkcija

Dobivena gramatika G postupkom zamjene krajnjeg lijevog znaka simulira rad PA M. 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 promijeni stanje iz q u p.

Dokaz L(G) = N(M) slijedi iz: $[q, A, p] \underset{G}{\overset{*}{\Longrightarrow}} x$ ako i samo ako $(q, x, A) \underset{M}{\overset{*}{\succ}} (p, \varepsilon, \varepsilon)$.

Primjer: Zadan je PA M = ({q_1,q_2}, {0,1}, {N,K}, δ, q_1, K, {}):
1) δ(q_1, 0, K) = {(q_1, NK)} 2) δ(q_1, 0, N) = {(q_1, NN)}
3) δ(q_1, 1, N) = {(q_2, ε)} 4) δ(q_2, 1, N) = {(q_2, ε)}
5) δ(q_2, ε, N) = {(q_2, ε)} 6) δ(q_2, ε, K) = {(q_2, ε)}

PA M prihvaća jezik N(M) ={0^n1^m| n≥1, m≥1, m≤n}.
Konstruira se gramatika G= (V, {0,1}, P, S).
V = {S, [q_1,N,q_1], [q_1,N,q_2], [q_2,N,q_1], [q_2,N,q_2], [q_1,K,q_1], [q_1,K,q_2], [q_2,K,q_1], [q_2,K,q_2] }

Gradnja produkcija počinje iz S. Ostale produkcije grade se isključivo za dohvatljive znakove. Redoslijed gradnje zasniva se na traženju dohvatljivih znakova. Nakon dva koraka konstrukcije imamo:

1) [q_1, N, q_1]$\rightarrow$ 0 [q_1, N, q_1] [q_1, N, q_1] | 0 [q_1, N, q_2] [q_2, N, q_1]
2) [q_1, N, q_2]$\rightarrow$ 0 [q_1, N, q_1] [q_1, N, q_2] | 0 [q_1, N, q_2] [q_2, N, q_2]
3) δ(q_1,1,N) = {(q_2,ε)} [q_1, N, q_2]$\rightarrow$ 1
4) δ(q_2,1,N) = {(q_2,ε)} [q_2, N, q_2]$\rightarrow$ 1
5) δ(q_2,ε,N) = {(q_2,ε)} [q_2, N, q_2]$\rightarrow$ ε
6) δ(q_2,ε,K) = {(q_2,ε)} [q_2, K, q_2]$\rightarrow$ ε

Odbacivanjem mrtvih znakova (za koje nema prijelaza iz q_2 u q_1) i odbacivanjem gdje postoji samo ε - produkcija, ostane nam:

S $\rightarrow$ [q_1, K, q_2]
[q_1, K, q_2]$\rightarrow$ 0 [q_1, N, q_2]
[q_1, N, q_2]$\rightarrow$ 0 [q_1, N, q_2] [q_2, N, q_2]
[q_1, N, q_2]$\rightarrow$ 1
[q_2, N, q_2]$\rightarrow$ 1
[q_2, N, q_2]$\rightarrow$ ε

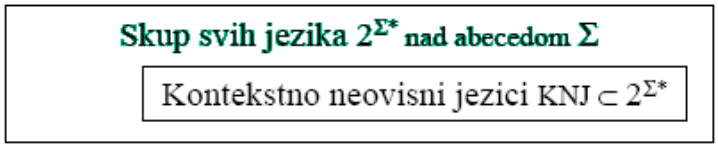
Gramatika generira niz nizom transformacija, a PA M prihvaća niz nizom konfiguracija.

19. SVOJSTVA KONTEKSTNO NEOVISNIH JEZIKA

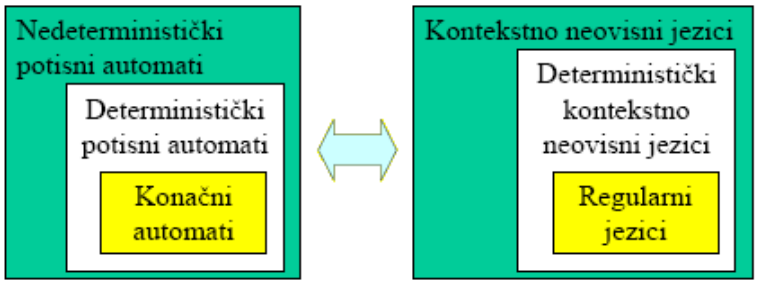
19. 1. Kontekstno neovisni jezik i PA

- Primjer jezika koji nije KNJ
- Položaj KNJ
- Istovjetnost KNJ i PA

Jezik L je kontekstno neovisan samo ako postoji PA koji ga prihvaća. $L = \{a^i b^i c^i \mid i \geq 1\}$ je primjer jezika koji nije KN jezik. Kontekstno neovisni jezici su pravi podskup svih jezika:



$L = \{ww^R\}$ je primjer jezika za koji je moguće izgraditi PA (NPA) i za koji **nije** moguće izgraditi DPA. Dakle postoje jezici koji su kontekstno neovisni, ali nisu deterministički kontekstno neovisni. Deterministički kontekstno neovisni jezici su **pravi podskup** skupa kontekstno neovisnih jezika.
DKA je poseban slučaj DPA koji ne koristi stog. Stoga su regularni jezici **pravi podskup** skupa deterministčkih 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 KNG, KNJ i PA koristi se za opis zatvorenosti KNJ.
KNG se koristi za dokazivanje da je KNJ zatvoren s obzirom na uniju, nadovezivanje, Kleene, supstituciju. PA i DKA se koriste za dokazivanje da presjek KNJ i RJ jest KNJ.

Unija: unija dvaju KNJ jest KNJ.
Neka $G_1 = (V_1, T_1, P_1, S_1)$ generira $L(G_1)$, a $G_2 = (V_2, T_2, P_2, S_2)$ generira $L(G_2)$, pri čemu je $V_1 \cap V_2 = \varnothing$. Gramatika $G_3 = (V_3, T_3, P_3, S_3)$ koja generira $L(G_3) = L(G_1) \cup L(G_2)$ konstruira se:

- $V_3 = V_1 \cup V_2 \cup \{S_3\}$; $S_3 \notin V_1 \cup V_2$
- $T_3 = T_1 \cup T_2$
- $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}$;

Dokaz:
Dokažimo prvo da je $L(G_1) \cup L(G_2) \subseteq L(G_3)$. Obzirom na $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 | S_2\}$ vrijedi:

- Pretpostavimo da je $w \in L(G_1)$: $S_3 \underset{G_3}{\Longrightarrow} S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} w$
- Slično za $w \in L(G_2)$: $S_3 \underset{G_3}{\Longrightarrow} S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} w$

Zaključujemo: $w \in L(G_1) \cup L(G_2) \Longrightarrow w \in L(G_3)$

Dokažimo drugo da je $L(G_3) \subseteq L(G_1) \cup L(G_2)$. Obzirom na $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 | S_2\}$ gramatika G_3 generira w na jedan od dva načina:

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

Obzirom da je $V_1 \cap V2 = \varnothing$, za S_1 koristimo P_1, za S_2 koristimo P_2.
Zaključujemo: $w \in L(G_3) \Longrightarrow w \in L(G_1) \cup L(G_2)$, i slijedi $L(G_1) \cup L(G_2) = L(G_3)$.

Nadovezivanje: nadovezivanje dvaju KNJ jest KNJ.
Neka $G_1 = (V_1, T_1, P_1, S_1)$ generira $L(G_1)$, a $G_2 = (V_2, T_2, P_2, S_2)$ generira $L(G_2)$, pri čemu je $V_1 \cap V_2 = \varnothing$. Gramatika $G_4 = (V_4, T_4, P_4, S_4)$ koja generira $L(G_4) = L(G_1)L(G_2)$ konstruira se:

- $V_4 = V_1 \cup V_2 \cup \{S_4\}$; $S_4 \notin V_1 \cup V_2$
- $T_4 = T_1 \cup T_2$
- $P_4 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 S_2\}$;

Dokaz:
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 $w_1w_2 \in L(G_4)$; $w_1 \in L(G_1)$ i $w_2 \in L(G_2)$

19. 3. Svojstva zatvorenosti KNJ na Kleene, supstituciju

- Dokaz zatvorenosti na Kleene
- Dokaz zatvorenosti na supstituciju, primjer

Kleene

KNJ zatvoreni su obzirom na Kleene.
Neka $G_1 = (V_1, T_1, P_1, S_1)$ generira $L(G_1)$. Gramatika $G_5 = (V_5, T_5, P_5, S_5)$ koja generira $L(G_5) = L(G_1)^*$ konstruira se:

- $V_5 = V_1 \cup \{S_5\}$; $S_5 \notin V_1$
- $T_5 = T_1$
- $P_5 = P_1 \cup \{S_5 \rightarrow S_1S_5 \mid \varepsilon\}$;

Dokaz:
Dokaz se zasniva na dva postupka generiranja niza:

1. $$S_5 \underset{G_5}{\Longrightarrow} S_1 S_5 \overset{*}{\underset{G_1}{\Longrightarrow}} w_1 S_5 \underset{G_5}{\Longrightarrow} w_1 S_1 S_5 \overset{*}{\underset{G_1}{\Longrightarrow}} w_1 w_1 S_5 \ldots \overset{*}{\underset{G_1}{\Longrightarrow}} w_1^+ S_5 \underset{G_5}{\Longrightarrow} w_1^+$$

   Gdje su $w_1^+ \in L(G_5)$ i $w_1 \in L(G_1)$

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

Supstitucija

KNJ zatvoreni su obzirom na supstituciju. Neka G = (V, T, P, S) generira L(G). Svi završni znakovi a_i jezika L(G) zamijene nizovima $L(G_i)$, $1 \leq i \leq k$, $k = |T|$.
Gramatika $G_i = (V_i, T_i, P_i, S_i)$ generira $L(G_i)$. Konstruira se gramatika G' koja generira nastali jezik L':

- $V' = V_1 \cup V_2 \cup .... V_k$, $V \cap V_i = \varnothing$, $V_i \cap V_j = \varnothing$, za sve i,j
- $T' = T_1 \cup T_2 \cup .... T_k$
- $S' = S$
- $P' = P_1 \cup P_2 \cup .... P_k$, a to su produkcije od G preuređene tako da se bilo koji a_i zamijeni početnim S_i.

Primjer supstitucije:
Zadan je jezik L u kojem nizovi imaju jednak broj znakova a i b.

- Jezik L generira gramatika G = ({S}, {a, b}, P, S), $S \rightarrow aSbS \mid bSaS \mid \varepsilon$
- Znak a zamijenimo nizovima jezika $L_1 = \{0^n1^n \mid n \geq 1\}$

- Jezik L_1 generira gramatika $G_1 = (\{S_1\}, \{0, 1\}, P, S_1)$, $S_1 \rightarrow 0S_1 1 \mid 01$
- Znak b zamijenimo nizovima jezika $L_2 = \{ww^R \mid w = (0+2)^*\}$
- Jezik L_2 generira gramatika $G_2 = (\{S_2\}, \{0, 2\}, P, S_2)$, $S_2 \rightarrow 0S_2 0 \mid 2S_2 2 \mid \varepsilon$
- Zamjenom znakova a i b nastaje jezik L'
- Jezik L' generira gramatika G'
    - $V' = \{S\} \cup \{S_1\} \cup \{S_2\} = \{S, S_1, S_2\}$
    - $T' = \{0, 1\} \cup \{0, 2\}$
    - $S' = S$
    - $P' = \{S_1 \rightarrow 0S_1 1 \mid 01\} \cup \{S_2 \rightarrow 0S_2 0 \mid 2S_2 2 \mid \varepsilon\} \cup \{S \rightarrow S_1 S S_2 S \mid S_2 S S_1 S \mid \varepsilon\}$

19. 4. Zatvorenost presjeka KNJ i RJ

- Dokaz zatvorenosti presjeka KNJ i RJ
- Primjer

Presjek KNJ i RJ **jest** KNJ. Pretpostavimo da:

- KNJ L_1 prihvaća PA $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_0, Z_1, F_1)$
- RJ L_2 prihvaća DKA $M_2 = (Q_2, \Sigma, \delta_2, p_0, F_2)$

Moguće je izgraditi PA $M' = (Q', \Sigma, \Gamma, \delta', q'_0, Z_0, F')$ koji prihvaća jezik $L = L_1 \cap L_2$:
$Q' = Q_2 \times Q_1$; $q'_0 = [p_0, q_0]$; $F' = F_2 \times F_1$
$\delta'([p,q], a, X)$ sadrži $([p',q'], \gamma)$ akko: za M_2 $\delta_2(p,a)=p'$ i za M_1 $(q', \gamma) \in \delta_1(q, a, X)$.
Ako a jest ε, onda je $p' = p$.
PA M' simulira rad M_1 i M_2.

Dokaz da je $L = L_1 \cap L_2$ izvodi se indukcijom.
Dokaže se da je:

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

ako i samo ako je:

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

gdje je $[p, q] \in F'$ ako i samo ako $p \in F_2$ i $q \in F_1$.

Primjer:

PA $M_1 = (\{q_1,q_2\}, \{0,1\}, \{N,K\}, \delta_1, q_1, K, \{q_2\})$ prihvaća jezik $L(M_1) = \{0^n 1^m \mid n \geq 1, m \geq 1, m \leq n\}$ prihvatljivim stanje q_2.
1) $\delta_1(q_1,0,K) = \{(q_1,NK)\}$ 2) $\delta_1(q_1,0,N) = \{(q_1,NN)\}$
3) $\delta_1(q_1,1,N) = \{(q_2,\varepsilon)\}$ 4) $\delta_1(q_2,1,N) = \{(q_2,\varepsilon)\}$
DKA $M_2 = (\{p_1, p_2, p_3\}, \{0,1\}, \delta_2, p_1, K, \{p_3\})$ prihvaća jezik $L(M_2)$ s barem dva znaka 1
$\delta_2(p_1,0) = p_1$ $\delta_2(p_1,1) = p_2$ $\delta_2(p_2,0) = p_2$ $\delta_2(p_2,1) = p_3$ $\delta_2(p_3,0) = p_3$ $\delta_2(p_3,1) = p_3$

Presjek $L(M_1)$ i $L(M_2)$ jest jezik $L_3 = L(M_1) \cap L(M_2) = \{0^n1^m \mid n\geq 2, m\geq 2, m\leq n\}$ i njega prihvaća $M_3 = (Q', \Sigma, \Gamma, \delta', q'_0, Z_0, F')$:

- $Q' = \{[p_1, q_1], [p_1, q_2], [p_2, q_1], [p_2, q_2], [p_3, q_1], [p_3, q_2],$
- $q'_0 = [p_1, q_1]$
- $F' = [p_3, q_2]$

1) $\delta'([p_1,q_1],0,K) = \{([p_1,q_1], NK)\}$
2) $\delta'([p_1,q_1],0,N) = \{([p_1,q_1], NN)\}$
3) $\delta'([p_1,q_1],1,N) = \{([p_2,q_2], \varepsilon)\}$
4) $\delta'([p_2,q_2],1,N) = \{([p_3,q_2], \varepsilon)\}$
5) $\delta'([p_3,q_2],1,N) = \{([p_3,q_2], \varepsilon)\}$

PA M_1 prihvaća niz prihvatljivim stanjem q_2. DKA M_2 također prihvaća niz. PA M' prihvaća niz prihvatljivim stanje $[p_3, q_2]$.

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. Primjena je dosta kompleksna. Zasniva se na **broju čvorova** generativnog stabla i na **broju nezavršnih članova** gramatike. Za dovoljno dugački niz broj unutrašnjih čvorova je veći 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. Sigurno postoji put stabla u kojem je jedan nezavršni znak na dva mjesta pri dnu stabla. Za takvo stablo postoji slijed generiranja niza:

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

Nezavršni znak A koristi se dva puta u postupku generiranja niza uvwxy.
Gramatika generira niz oblika:

$$S \underset{G}{\overset{*}{\Longrightarrow}} uAy \underset{G}{\overset{*}{\Longrightarrow}} uvAxy \underset{G}{\overset{*}{\Longrightarrow}} uvvAxxy \underset{G}{\overset{*}{\Longrightarrow}} uvvvAxxxy \underset{G}{\overset{*}{\Longrightarrow}} \ldots \underset{G}{\overset{*}{\Longrightarrow}} uv^iAx^iy \underset{G}{\overset{*}{\Longrightarrow}} uv^iwx^iy$$

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 $uv^iwx^iy \in L$