# MODELI RAČUNARSTVA - JEZIČNI PROCESORI 1

Siniša Srbljić, Sveučilište u Zagrebu

1. UVOD
2. REGULARNI JEZICI
3. KONTEKSTNO NEOVISNI JEZICI
4. REKURZIVNO PREBROJIVI JEZICI
5. KONTEKSTNO OVISNI JEZICI
6. RAZREDBA (TAKSONOMIJA) JEZIKA, AUTOMATA I GRAMATIKA

# 3. KONTEKSTNO NEOVISNI JEZICI

- 3.1. KONTEKSTNO NEOVISNA GRAMATIKA
- 3.2. POTISNI AUTOMAT
- 3.3. SVOJSTVA KONTEKSTNO NEOVISNIH JEZIKA

# 3. KONTEKSTNO NEOVISNI JEZICI

- KONTEKSTNO NEOVISNI JEZIK I GRAMATIKA
  - 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
  - jezik N = $\{wcw^R\}$ nije regularan, ali ga je moguće generirati kontekstno neovisnom gramatikom

# 3.1. Kontekstno neovisna gramatika (Context Free Grammar, CFG)

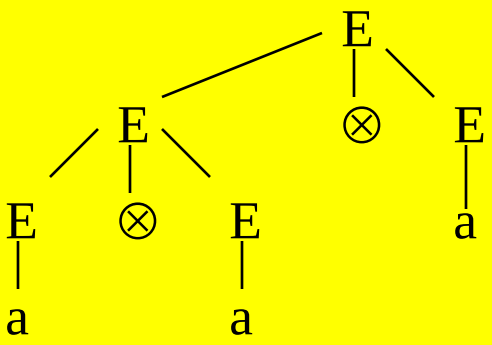
- 3.1.1. Nejednoznačnost gramatike, jezika i niza
- 3.1.2. Pojednostavljenje gramatike
- 3.1.3. Parsiranje niza

# 3.1.1. Nejednoznačnost gramatike, jezika i niza

- INTERPRETACIJA NIZA
  - zasniva se na generativnom stablu
  - za neke nizove moguće je izgraditi više stabala
  - npr. G = ({E}, {a,⊗}, {E→E⊗E|a}, E)
  - za niz a⊗a⊗a moguće je izgraditi dva generativna stabla
  - interpretacija niza je nejednoznačna jer ne možemo odrediti generativno stablo
  - različito stablo dobijemo
    - promjenom redoslijeda produkcija
    - promjenom redoslijeda nezavršnih znakova

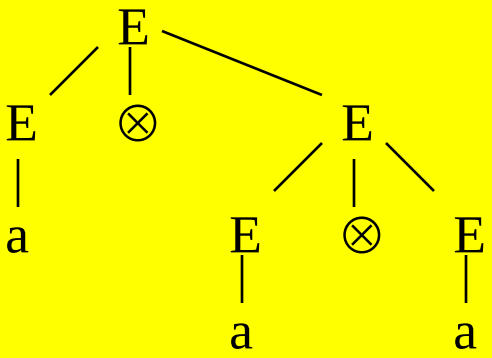
# Nejednoznačnost gramatike, jezika i niza

- NEJEDNOZNAČNOST NIZA




(1) $\underline{E} \Rightarrow \underline{E} \otimes E \Rightarrow \underline{E} \otimes E \otimes E \Rightarrow a \otimes \underline{E} \otimes E$
$\Rightarrow a \otimes a \otimes \underline{E} \Rightarrow a \otimes a \otimes a$

(3) $\underline{E} \Rightarrow E \otimes \underline{E} \Rightarrow \underline{E} \otimes a \Rightarrow E \otimes \underline{E} \otimes a$
$\Rightarrow \underline{E} \otimes a \otimes a \Rightarrow a \otimes a \otimes a$




(2) $\underline{E} \Rightarrow \underline{E} \otimes E \Rightarrow a \otimes \underline{E} \Rightarrow a \otimes \underline{E} \otimes E$
$\Rightarrow a \otimes a \otimes \underline{E} \Rightarrow a \otimes a \otimes a$

(4) $\underline{E} \Rightarrow E \otimes \underline{E} \Rightarrow E \otimes E \otimes \underline{E} \Rightarrow E \otimes \underline{E} \otimes a$
$\Rightarrow \underline{E} \otimes a \otimes a \Rightarrow a \otimes a \otimes a$

# Nejednoznačnost gramatike, jezika i niza

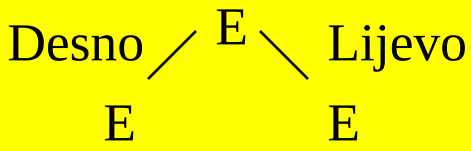
- NEJEDNOZNAČNOST NIZA
  - generativno stablo određuje **grupiranje**:
    (a⊗a)⊗a  a⊗(a⊗a)
  - u slučaju aritmetičkog izraza grupiranje određuje redoslijed operacija
  - za neke operacije to je bitno, npr. oduzimanje
  - nastojimo konstruirati gramatiku koja za zadani niz gradi samo jedno stablo
  - utvrdili smo:
    - za bilo koji niz CFG moguće je izgraditi jedno ili više stabala
    - bilo koje stablo moguće je izgraditi jednim ili više postupaka

# Nejednoznačnost gramatike, jezika i niza

- ZAMJENA KRAJNJIH NEZAVRŠNIH ZNAKOVA
  - redoslijed zamjene nezavršnih znakova je značajan
  - uvedimo postupak zamjene krajnjeg desnog ili krajnjeg lijevog nezavršnog znaka
  - kod zamjene **krajnjeg lijevog** produkcije primjenjujemo samo na krajnji lijevi nezavršni znak u izrazu, (1) i (2)
  - kod zamjene **krajnjeg desnog** produkcije primjenjujemo samo na krajnji desni nezavršni znak u izrazu, (3) i (4)

# Nejednoznačnost gramatike, jezika i niza

- OBILAZAK STABLA
  - 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:



# Nejednoznačnost gramatike, jezika i niza

- OBILAZAK STABLA
  - postupak obilaska za neko stablo jednoznačno određuje
    - redoslijed primjene produkcija i
    - redoslijed zamjene znakova
  - **desni** obilazak definira zamjenu krajnje **lijevog** znaka
  - **lijevi** obilazak definira zamjenu krajnje **desnog** znaka
  - za stablo (a⊗a)⊗a to su postupci (1) i (3)
  - to su ujedno jedini postupci koji generiraju to stablo
  - bilo koje stablo moguće je izgraditi primjenom jednog i samo jednog postupka zamjene krajnje lijevog ili desnog znaka

# Nejednoznačnost gramatike, jezika i niza

- DEFINIRANJE NEJEDNOZNAČNOSTI
  - **Nejednoznačnost CFG** G definiramo
    - ako je za niz $w \in L(G)$ moguće izgraditi više različitih generativnih stabala, gramatika je nejednoznačna
    - ako je niz $w \in L(G)$ moguće generirati primjenom više postupaka zamjene krajnjeg desnog ili krajnjeg lijevog znaka, onda je gramatika nejednoznačna
  - gramatika

    $$G = (\{E\}, \{a, \otimes\}, \{E \rightarrow E \otimes E | a\}, E)$$

    je nejednoznačna

# Nejednoznačnost gramatike, jezika i niza

- DEFINIRANJE NEJEDNOZNAČNOSTI
  - **Nejednoznačnost niza** definiramo
    - ako je za niz w moguće izgraditi više različitih generativnih stabala, on je nejednoznačan
  - **Nejednoznačnost jezika** definiramo
    - ako jezik L nije moguće generirati niti jednom jednoznačnom gramatikom, onda je jezik L nejednoznačan
  - primjer nejednoznačnog jezika je:

$$L_n = L_1 \cup L_2 = \left\{ a^n b^n c^m d^m \middle| n \geq 1, m \geq 1 \right\} \cup \left\{ a^n b^m c^m d^n \middle| n \geq 1, m \geq 1 \right\}$$

# Nejednoznačnost gramatike, jezika i niza

- RAZRJEŠAVANJE NEJEDNOZNAČNOSTI
  - Nejednoznačnost razrješujemo:
  - Promjenom gramatike
    - umjesto gramatike G izgradi se nova jednoznačna gramatika G’
  - Promjenom jezika
    - umjesto gramatike L izgradi se novi jezik L’ koji je moguće generirati jednoznačnom gramatikom

# Nejednoznačnost gramatike, jezika i niza

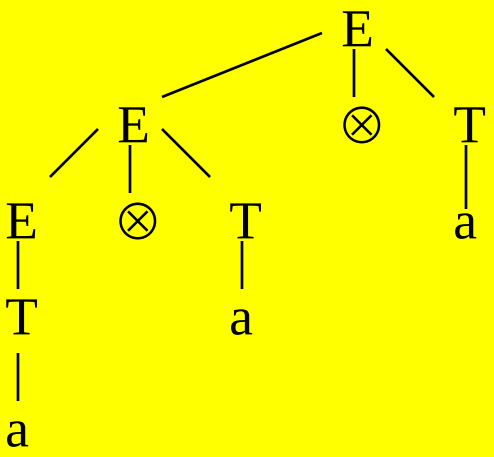
- PROMJENA GRAMATIKE
  - jezik L kojeg generira G = ({E}, {a,⊗}, {E→E⊗E|a}, E) moguće je generirati više različitih jednoznačnih gramatika:
  - Za lijevo asocijativni ⊗ uvodimo gramatiku:
    $G_1$ = ({E,T}, {a,⊗}, {E→E⊗T|T, T→a}, E)
  - Za desno asocijativni ⊗ uvodimo gramatiku:
    $G_2$ = ({E,T}, {a,⊗}, {E→T⊗E|T, T→a}, E)
  - izbor jednoznačne gramatike određuje način gradnje generativnog stabla

# Nejednoznačnost gramatike, jezika i niza

- PROMJENA GRAMATIKE
  - lijevo asocijativna gramatika

    $G_1 = (\{E,T\}, \{a,\otimes\}, \{E \to E\otimes T \mid T,\ T \to a\}, E)$

    daje niz:




(5) $\underline{E} \Rightarrow \underline{E}\otimes T \Rightarrow \underline{E}\otimes T\otimes T \Rightarrow \underline{T}\otimes T\otimes T$

$\Rightarrow a\otimes \underline{T}\otimes T \Rightarrow a\otimes a\otimes \underline{T} \Rightarrow a\otimes a\otimes a$

(6) $\underline{E} \Rightarrow E\otimes \underline{T} \Rightarrow \underline{E}\otimes a \Rightarrow E\otimes \underline{T}\otimes a$

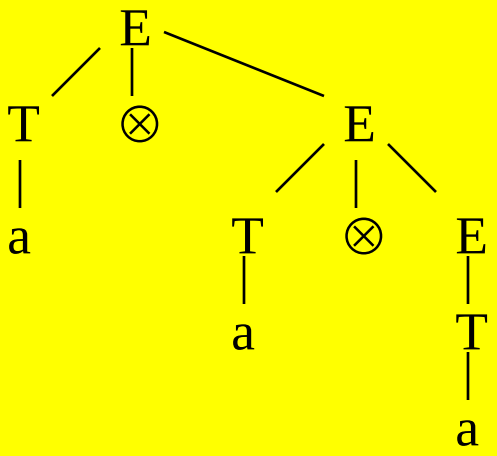
$\Rightarrow \underline{E}\otimes a\otimes a \Rightarrow \underline{T}\otimes a\otimes a \Rightarrow a\otimes a\otimes a$

# Nejednoznačnost gramatike, jezika i niza

- PROMJENA GRAMATIKE
  - desno asocijativna gramatika

$$G_2 = (\{E,T\}, \{a,\otimes\}, \{E \to T\otimes E \mid T,\ T \to a\}, E)$$

daje niz:




(7) $\underline{E} \Rightarrow \underline{T}\otimes E \Rightarrow a\otimes\underline{E} \Rightarrow a\otimes\underline{T}\otimes E$

$\Rightarrow a\otimes a\otimes\underline{E} \Rightarrow a\otimes a\otimes\underline{T} \Rightarrow a\otimes a\otimes a$

(8) $\underline{E} \Rightarrow T\otimes\underline{E} \Rightarrow T\otimes T\otimes\underline{E} \Rightarrow T\otimes T\otimes\underline{T}$

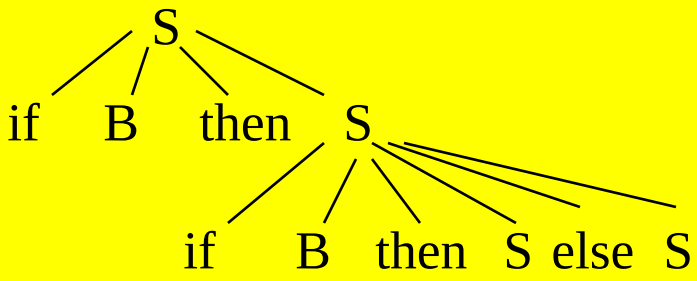
$\Rightarrow T\otimes\underline{T}\otimes a \Rightarrow \underline{T}\otimes a\otimes a \Rightarrow a\otimes a\otimes a$

# Nejednoznačnost gramatike, jezika i niza

- PRIMJER IF-THEN-ELSE
  - gramatika
    G = ({S,B}, {**if**, **then**, **else**, **true**, **false**},
    {S→**if** B **then** S **else** S|**if** B **then** S, B→**true**|**false**}, S)
    generira složene naredbe programskog jezika
  - nezavršni znak S generira naredbu, a B logički izraz
  - gramatika nije jednoznačna, što je vidljivo na primjeru
    **if** B **then if** B **then** S **else** S
  - moguće je izgraditi dva generativna stabla
  - algol je bio nejednoznačan

# Nejednoznačnost gramatike, jezika i niza

- PRIMJER IF-THEN-ELSE
  - **if true then**
    **if false then** print X
    **else** print Y

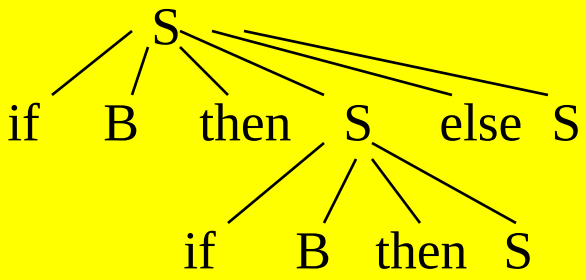



$\underline{S} \Rightarrow$ if B then $\underline{S} \Rightarrow$

$\Rightarrow$ if B then if B then S else S

# Nejednoznačnost gramatike, jezika i niza

- PRIMJER IF-THEN-ELSE
  - **if true then**
    **if false then** print X
    **else** print Y




$\underline{S} \Rightarrow$ if B then $\underline{S}$ else S $\Rightarrow$

$\Rightarrow$ if B then if B then S else S

# Nejednoznačnost gramatike, jezika i niza

- PRIMJER IF-THEN-ELSE
  - problem rješavamo gramatikom

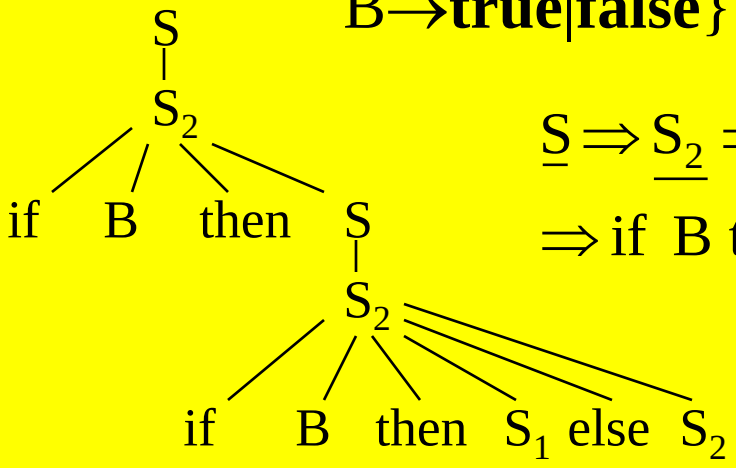
    $G = (\{S, S_1, S_2, B\}, \{\textbf{if}, \textbf{then}, \textbf{else}, \textbf{true}, \textbf{false}\}, P, S)$

    $P = \{ \quad S \rightarrow S_1 | S_2,$

    $S_1 \rightarrow \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ ,$

    $S_2 \rightarrow \textbf{if}\ B\ \textbf{then}\ S\ |\textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2,$

    $B \rightarrow \textbf{true}|\textbf{false}\}$




$$\underline{S} \Rightarrow \underline{S_2} \Rightarrow \text{if } B \text{ then } \underline{S} \Rightarrow \text{if } B \text{ then } \underline{S_2} \Rightarrow$$

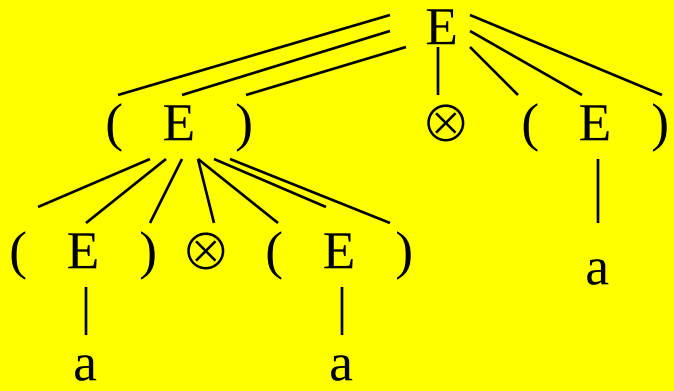
$$\Rightarrow \text{if } B \text{ then if } B \text{ then } S_1 \text{ else } S_2$$

# Nejednoznačnost gramatike, jezika i niza

- PROMJENA JEZIKA
  - 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
  - npr. pogledajmo gramatiku
    $G_3 = (\{E\}, \{a,\otimes,(,)\}, \{E\rightarrow(E)\otimes(E)|a\}, E)$
  - sada su nizovi $(a)\otimes((a)\otimes(a))$ i $((a)\otimes(a))\otimes(a)$ jednoznačni

# Nejednoznačnost gramatike, jezika i niza

- PROMJENA JEZIKA

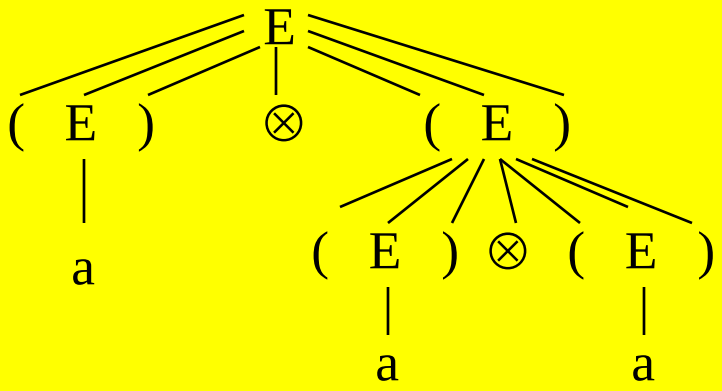



$$\underline{E} \Rightarrow (\underline{E}) \otimes (E) \Rightarrow ((E) \otimes (E)) \otimes (E)$$

$$\overset{*}{\Rightarrow} ((a) \otimes (a)) \otimes (a)$$

# Nejednoznačnost gramatike, jezika i niza

- PROMJENA JEZIKA




$$\underline{E} \Rightarrow (\underline{E}) \otimes (E) \Rightarrow (a) \otimes (\underline{E})$$
$$\Rightarrow (a) \otimes ((E) \otimes (E))$$
$$\overset{*}{\Rightarrow} (a) \otimes ((a) \otimes (a))$$

# Nejednoznačnost gramatike, jezika i niza

- RAZLIKA PROMJENA GRAMATIKE I JEZIKA
  - promjenom gramatike
    - ne mijenja se jezik
    - odbacuje se višestruko značenje niza
  - promjenom jezika
    - čuva se višestruko značenje niza
    - definira se zaseban niz za svako značenje
  - primjer:
    - lijevo asocijativna gramatika $G_1$
    - desno asocijativna gramatika $G_2$
    - gramatika promijenjenog jezika sa zagradama $G_3$
  - u programskim jezicima koristimo {} za složene naredbe

# 3.1.2. Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA
  - odbacuju beskorisne znakove i produkcije
  - generiramo gramatiku sa tri svojstva:
    (i) bilo koji znak koristi se u makar jednom nizu
    (ii) ne koriste se jedinične produkcije $A \rightarrow B$
    (iii) ne koriste se $\varepsilon$-produkcije
  - koristimo algoritme
    - odbacivanja beskorisnih znakova
    - odbacivanja jediničnih produkcija i $\varepsilon$-produkcija
    - postizanja normalnih oblika Chomskog i Greibacha

# Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA

  **(i)** bilo koji znak gramatike G koristi se za generiranje makar jednog niza jezika L

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

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

  onda je X koristan, u suprotnom je **beskoristan**

  – dva su vida beskorisnosti:
    - znak je mrtav
    - znak je nedohvatljiv

# Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA
  - ako iz znaka X nije moguće generirati niz završnih znakova, tj. ako ne postoji postupak:

$$X \not\overset{*}{\Rightarrow} w_X \qquad ; \qquad w_X \in T^*$$

    onda je znak X **mrtav**, u suprotnom je živ
  - ako znak X nije ni u jednom nizu koji se generira iz S, tj. ako ne postoji postupak:

$$S \not\overset{*}{\Rightarrow} \alpha X \beta$$

    onda je znak X **nedohvatljiv**

# Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA
  - neka je X dohvatljiv i živ tj. neka vrijedi:

    $S \overset{*}{\Rightarrow} \alpha X \beta$ $\quad X \overset{*}{\Rightarrow} w_X$ ; $\quad w_X \in T^*$

    znak X nije nužno koristan
  - moguće je da jedan od podnizova $\alpha$ ili $\beta$ sadrži mrtvi znak
  - ako u bilo kojem postupku $S \overset{*}{\Rightarrow} \alpha X \beta$ barem jedan podniz sadrži mrtvi znak, X je beskoristan

# Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA

  **(ii)** gramatika G nema jediničnih produkcija tipa $A \rightarrow B$

  – $A \rightarrow B$ je jedinična produkcija

  – sve ostale produkcije uključujući $A \rightarrow a$ i $A \rightarrow \varepsilon$ nazivaju se nejedinične produkcije

# Pojednostavljenje gramatike

- POSTUPCI POJEDNOSTAVLJENJA

  **(iii)** ako prazni niz $\varepsilon$ nije element jezika L, moguće je izbjeći korištenje produkcija tipa $A \rightarrow \varepsilon$

  – produkcija $A \rightarrow \varepsilon$ naziva se $\varepsilon$-produkcija

  – gramatiku G preuredimo tako da sve produkcije budu oblika $A \rightarrow BC$ i $A \rightarrow a$,
  te su u normalnom obliku Chomskog

  – gramatiku G preuredimo tako da sve produkcije budu oblika $A \rightarrow a\alpha$, $\alpha$ može biti prazan
  te su u normalnom obliku Greibacha

# Pojednostavljenje gramatike

- ODBACIVANJE MRTVIH ZNAKOVA
  - neka CFG G = (V, T, P, S)
    generira neprazan jezik $L(G) \neq \emptyset$
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **nema mrtvih znakova** tako da je

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

  - npr. G = ({S, A, B}, {a,b,c,d,e,f}, P, S)
    P = {S $\rightarrow$ aSa, S $\rightarrow$ bAd, S $\rightarrow$ c,
    A $\rightarrow$ cBd, A $\rightarrow$ aAd, B $\rightarrow$ dAf}
  - nezavršni znakovi A i B su mrtvi znakovi

# Pojednostavljenje gramatike

- ALGORITAM TRAŽENJA ŽIVIH ZNAKOVA
  - ako su živi svi znakovi desne strane produkcije

$$A \to X_1 X_2 \cdots X_n$$

    živ je i nezavršni znak s lijeve strane produkcije
  - budući da s desna nema mrtvih znakova, vrijedi:

$$X_i \to w_i;\ w_i \in T^*$$

  - stoga vrijedi:

$$A \to w_1 w_2 \cdots w_n = w$$

# Pojednostavljenje gramatike

- ALGORITAM TRAŽENJA ŽIVIH ZNAKOVA
  - algoritam se provodi u tri koraka:
    1. U listu živih znakova stave se lijeve strane produkcija koje na desnoj nemaju nezavršnih znakova
    2. Ako su s desne strane produkcije isključivo živi znakovi, onda se u listu doda znak s lijeve strane
    3. Ako se lista živih ne može proširiti, svi znakovi koji nisu na listi su mrtvi znakovi

# Pojednostavljenje gramatike

- ALGORITAM TRAŽENJA ŽIVIH ZNAKOVA
  - primjer: G = ({S, A, B, C}, {a,b,c,d,}, P, S)

    1) $S \rightarrow aABS$ 4) $A \rightarrow cSA$ 7) $B \rightarrow cSB$

    2) $S \rightarrow bCACd$ 5) $A \rightarrow cCC$ 8) $C \rightarrow cS$

    3) $A \rightarrow bAB$ 6) $B \rightarrow bAB$ 9) $C \rightarrow c$
  - u listu stavljamo žive znakove:

    C zbog 9), A zbog 5) i S zbog 2)
  - B je mrtav, odbacujemo produkcije i dobijemo:

    G = ({S, A, C}, {a,b,c,d,}, P, S)

    2) $S \rightarrow bCACd$ 4) $A \rightarrow cSA$ 8) $C \rightarrow cS$

    5) $A \rightarrow cCC$ 9) $C \rightarrow c$

# Pojednostavljenje gramatike

- ODBACIVANJE NEDOHVATLJIVIH ZNAKOVA
  - neka CFG G = (V, T, P, S) generira jezik $L(G) \neq \emptyset$
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **nema nedohvatljivih znakova**

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

  - npr. G = ({S, A}, {a,b,c}, P, S)
    P = {S → aSb, S → c, A → bS, A → a}
  - produkcije sa S na lijevoj strani nemaju A na desnoj, pa je A nedohvatljiv, ostaju samo produkcije iz S

# Pojednostavljenje gramatike

- TRAŽENJE DOHVATLJIVIH ZNAKOVA
  - ako je dohvatljiv nezavršni znak s lijeve strane produkcije

    $$A \to \alpha_1 | \alpha_2 | \alpha_3 \cdots | \alpha_n$$

    onda su dohvatljivi svi završni i nezavršni znakovi s desne strane produkcije
  - neka je S početni nezavršni znak, i neka je A dohvatljiv vrijedi:

    $$S \overset{*}{\Rightarrow} \beta A \gamma \Rightarrow \beta \alpha_i \gamma; \; i = 1 \cdots n$$

  - pa su svi znakovi $\alpha_i$ dohvatljivi

# Pojednostavljenje gramatike

- TRAŽENJE DOHVATLJIVIH ZNAKOVA
  - algoritam se provodi u tri koraka:
    1. U listu dohvatljivih znakova stavi se početni nezavršni znak gramatike
    2. Ako je znak s lijeve strane produkcije dohvatljiv, u listu se dodaju svi znakovi s desne strane produkcije
    3. Ako se lista dohvatljivih ne može proširiti, svi znakovi koji nisu na listi su nedohvatljivi znakovi

# Pojednostavljenje gramatike

- TRAŽENJE DOHVATLJIVIH ZNAKOVA
  - primjer: G = ({S, A, B, C, D, E}, {a,b,c,d,e,f,g}, P, S)

    1) $S \rightarrow aAB$ 5) $B \rightarrow bE$ 9) $C \rightarrow a$

    2) $S \rightarrow E$ 6) $B \rightarrow f$ 10) $D \rightarrow eA$

    3) $A \rightarrow dDA$ 7) $C \rightarrow cAB$ 11) $E \rightarrow fA$

    4) $A \rightarrow e$ 8) $C \rightarrow dSD$ 12) $E \rightarrow g$
  - u listu stavljamo dohvatljive znakove:
    S; A, B i a zbog 1), E zbog 2) d i D zbog 3), e zbog 4) b zbog 5), f zbog 6), e zbog 10) i g zbog 12)
  - C i c su nedohvatljivi, odbacujemo produkcije pa je:
    G = ({S, A, B, D, E}, {a,b,d,e,f}, P, S)
    odbacimo produkcije 7, 8 i 9

# Pojednostavljenje gramatike

- ODBACIVANJE BESKORISNIH ZNAKOVA
  - primjenom algoritma
    - odbacivanja mrtvih znakova
    - pa odbacivanja nedohvatljivih znakova
  - iz gramatike se izbace svi beskorisni znakovi
  - nužno je ići tim redoslijedom (odbaciti prvo mrtve)
  - primjer: G = ({S, A, B}, {a}, P, S)
    $S \rightarrow AB|a$ $A \rightarrow a$
  - B je mrtav, ostaje: $S \rightarrow a$ $A \rightarrow a$
  - A je nedohvatljiv, ostaje $S \rightarrow a$
  - obrnuto: A bi bio dohvatljiv, a samo B bi bio mrtav

# Pojednostavljenje gramatike

- ODBACIVANJE BESKORISNIH ZNAKOVA
  - neka CFG G = (V, T, P, S) generira jezik $L(G)\neq\varnothing$
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **nema beskorisnih znakova**
  - neka je $G_1$ nastala odbacivanjem mrtvih znakova iz G
  - neka je $G_2$ nastala odbacivanjem nedohvatljivih iz $G_1$
  - $G_2$ nema nedohvatljivih znakova i vrijedi $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha X \beta$
  - kako $G_1$ i $G_2$ imaju iste znakove, a $G_1$ nema mrtvih, onda ni $G_2$ nema mrtvih znakova pa su svi znakovi u nizu $\alpha X \beta$ živi $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha X \beta \overset{*}{\underset{G_2}{\Rightarrow}} w, \quad w \in T^*$

# Pojednostavljenje gramatike

- ODBACIVANJE BESKORISNIH ZNAKOVA
  - primjer: G = ({S, A, B, C}, {a,b,c,d}, P, S)
    1) $S \rightarrow ac$ 3) $A \rightarrow cBC$ 5) $C \rightarrow bC$
    2) $S \rightarrow bA$ 4) $B \rightarrow aSA$ 6) $C \rightarrow d$
  - u listu stavljamo žive znakove: C zbog 6) i S zbog 1)
  - A i B su mrtvi, dobijemo: $G_1$ = ({S, C}, {a,b,c,d,}, $P_1$, S)
    1) $S \rightarrow ac$ 5) $C \rightarrow bC$ 6) $C \rightarrow d$
  - u listu stavljamo dohvatljive znakove: S
  - C, b i d su nedohvatljivi, dobijemo:
    $G_2$ = ({S}, {a,c,}, $P_2$, S) 1) $S \rightarrow ac$

# Pojednostavljenje gramatike

- ODBACIVANJE ε-PRODUKCIJA
  - neka CFG G = (V, T, P, S) generira jezik L(G)\{ε}
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S) L(G)=L(G'), koja **nema ε-produkcija** $A \rightarrow \varepsilon$
  - primjer: G = ({S, A}, {a,b,c}, P, S)
    1) $S \rightarrow aASA$ 2) $S \rightarrow b$ 3) $A \rightarrow cS$ 4) $A \rightarrow \varepsilon$
    - umjesto nezavršnog znaka A definiramo dva: $A_{DA}$ i $A_{NE}$
    - $A_{DA}$ koristimo u produkciji 4, $A_{NE}$ u produkciji 3, zamijenimo:
      1a) $S \rightarrow aA_{NE}SA_{NE}$ 1b) $S \rightarrow aA_{NE}SA_{DA}$ 1c) $S \rightarrow aA_{DA}SA_{NE}$
      1d) $S \rightarrow aA_{DA}SA_{DA}$ 2) $S \rightarrow b$ 3) $A_{NE} \rightarrow cS$ 4) $A_{DA} \rightarrow \varepsilon$
    - zamijenimo $A_{DA}$ s ε, odbacimo 4), $A_{NE}$ zamijenimo s A:
      1a) $S \rightarrow aASA$ 1b) $S \rightarrow aAS$ 1c) $S \rightarrow aSA$ 1d) $S \rightarrow aS$
      2) $S \rightarrow b$ 3) $A \rightarrow cS$

# Pojednostavljenje gramatike

- ODBACIVANJE ε-PRODUKCIJA
  - algoritam se izvodi u dva osnovna koraka:
    1) pronađu se svi nezavršni 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

# Pojednostavljenje gramatike

- ODBACIVANJE ε-PRODUKCIJA

  2) gradi se novi skup produkcija gramatike G'

  - za produkciju iz G: $A \rightarrow X_1 X_2 \ldots X_n$
    dodaju se u G' produkcije $A \rightarrow \xi_1 \xi_2 \ldots \xi_n$
  - oznake $\xi_i$ poprimaju vrijednosti:
    - $X_i$ ako je $X_i$ neprazan,
    - $X_i$ ili ε ako je $X_i$ prazan
  - kada svi $\xi_i$ poprime vrijednost ε nastaje ε-produkcija koja se NE dodaje u listu produkcija G'
  - ako produkcija na desnoj strani ima k praznih znakova,
    - potrebno je izgraditi $2^k$ novih produkcija
    - ako je s desne strane neprazni znak, svih $2^k$ produkcija ostaje
    - ako s desne strane nije neprazni znak, ostaje $2^k$-1 produkcija

# Pojednostavljenje gramatike

- ODBACIVANJE JEDINIČNIH PRODUKCIJA
  - neka CFG G = (V, T, P, S) generira jezik L(G)\{ε}
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **nema jediničnih produkcija** $A \rightarrow B$
  - algoritam se provodi u dva koraka:
    1) u P' stave se sve produkcije iz P koje nisu jedinične
    2) za sve postupke generiranja B iz A

$$A \overset{*}{\Rightarrow} B$$

na osnovu $B \rightarrow \alpha$ stvore se nove produkcije $A \rightarrow \alpha$

# Pojednostavljenje gramatike

- CHOMSKYJEV NORMALNI OBLIK (Chomsky Normal Form, CNF)
  - neka CFG G = (V, T, P, S) generira jezik L(G)\{ε}
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **ima sve produkcije** oblika:

    $A \rightarrow BC$ ili $A \rightarrow a$
  - pretpostavi se da G nema beskorisnih znakova, ε-produkcija niti jediničnih produkcija
  - algoritam pretvorbe u CNF se provodi u tri koraka:

# Pojednostavljenje gramatike

- CHOMSKYJEV NORMALNI OBLIK

  1) u skup P' stave se sve produkcije koje su u CNF, tj.

  $A \rightarrow BC$ ili $A \rightarrow a$

  a u skup V' upišu se svi nezavršni znakovi

  2) neka je produkcija gramatike G oblika:

  $A \rightarrow X_1 X_2 \ldots X_n$ ; $A \in V$, $X_i \in V \cup T$

  ako je $X_i$ završni znak $a \in T$,

  - skup nezavršnih znakova proširi se sa $C_a \in V'$
  - skup produkcija proširi se sa $C_a \rightarrow a$ koja je u CNF
  - svi završni znakovi a zamijene se sa $C_a$
  - postupak se nastavlja dok se ne zamijene svi završni znakovi
  - postupak se nastavlja za sve produkcije

# Pojednostavljenje gramatike

- CHOMSKYJEV NORMALNI OBLIK

  3) nakon koraka 2

  - sve su produkcije u P' oblika $A \rightarrow a$ ili $A \rightarrow B_1 B_2 B_3 ... B_m$ ,
  - a one oblika $A \rightarrow BC$ ili $A \rightarrow a$ su u CNF

  produkcije koje s desna imaju 3 ili više znakova

  - mijenjaju se n ovim produkcijama
  - definiraju se novi znakovi $D_1 D_2 D_3 ... D_{m-2}$
  - pa se $A \rightarrow B_1 B_2 B_3 ... B_m$ zamijeni skupom produkcija:
    $\{A \rightarrow B_1 D_1, D_1 \rightarrow B_2 D_2, D_2 \rightarrow B_3 D_3, ... D_{m-2} \rightarrow B_{m-1} B_m\}$

# Pojednostavljenje gramatike

- CHOMSKYJEV NORMALNI OBLIK
  - primjer: G = ({S, A, B}, {a,b}, P, S)

    1) $S \rightarrow bA$ 3) $A \rightarrow bAA$ 6) $B \rightarrow aBB$

    2) $S \rightarrow aB$ 4) $A \rightarrow aS$ 7) $B \rightarrow bS$

    5) $A \rightarrow a$ 8) $B \rightarrow b$

    - produkcije 5 i 8 su u CNF
    - definira se $C_a$ i $C_b$, te dodaju produkcije $C_a \rightarrow a$ i $C_b \rightarrow b$:

      1) $S \rightarrow C_bA$ 3) $A \rightarrow C_bAA$ 6) $B \rightarrow C_aBB$ 9) $C_a \rightarrow a$

      2) $S \rightarrow C_aB$ 4) $A \rightarrow C_aS$ 7) $B \rightarrow C_bS$ 10) $C_b \rightarrow b$

      5) $A \rightarrow a$ 8) $B \rightarrow b$
    - sada su 1, 2, 4, 5, 7, 8, 9 i 10 u CNF

# Pojednostavljenje gramatike

- CHOMSKYJEV NORMALNI OBLIK
  - primjer:
    - treba razriješiti produkcije 3 i 6
    - definira se $D_1$ i $E_1$, te dodaju produkcije $D_1 \rightarrow AA$ i $E_1 \rightarrow BB$:

      1) $S \rightarrow C_bA$ 3a) $A \rightarrow C_b D_1$ 6a) $B \rightarrow C_aE_1$ 9) $C_a \rightarrow a$

      2) $S \rightarrow C_aB$ 3b) $D_1 \rightarrow AA$ 6a) $E_1 \rightarrow BB$ 10) $C_b \rightarrow b$

      4) $A \rightarrow C_aS$ 7) $B \rightarrow C_bS$

      5) $A \rightarrow a$ 8) $B \rightarrow b$
    - sada su sve produkcije u CNF

# Pojednostavljenje gramatike

- GREIBACHOV NORMALNI OBLIK (Greibach Normal Form, GNF)
  - neka CFG G = (V, T, P, S) generira jezik L(G)\{ε}
  - moguće je izgraditi istovjetnu CFG G' = (V', T', P', S), L(G)=L(G'), koja **ima sve produkcije** oblika:

    $A \rightarrow a\alpha; \qquad \alpha \in V^*$
  - Koriste se tri postupka
    - algoritam pretvorbe gramatike u normalni oblik Chomskog
    - algoritam zamjene krajnjeg lijevog nezavršnog znaka
    - algoritam razrješenja lijeve rekurzije
  - algoritam pretvorbe u CNF je opisan ranije

# Pojednostavljenje gramatike

- ZAMJENA KRAJNJE LIJEVOG ZNAKA
  - neka je u CFG G = (V, T, P, S)
    - r produkcija koje imaju $D_j$ s lijeve strane

      $D_j \rightarrow \alpha_1$ , $D_j \rightarrow \alpha_2$ ... $D_j \rightarrow \alpha_r$
    - produkcija 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

# Pojednostavljenje gramatike

- RAZRJEŠENJE LIJEVE REKURZIJE
  - ako je isti nezavršni znak na lijevoj strani produkcije i krajnje lijevo na desnoj strani produkcije
  - produkcija je **lijevo rekurzivna**
  - gramatika se preuređuje:
    - neka je za $D_i$ zadano r lijevo rekurzivnih produkcija

      $D_i \rightarrow D_i\alpha_1$ , $D_j \rightarrow D_i\alpha_2$ ... $D_j \rightarrow D_i\alpha_r$
    - i s produkcija koje nisu lijevo rekurzivne

      $D_i \rightarrow \beta_1$ , $D_i \rightarrow \beta_2$ ... $D_i \rightarrow \beta_r$
  - prethodnih r+s produkcija zamijeni se sa:

    $D_i \rightarrow \beta_l$ , $D_i \rightarrow \beta_l C_i$ , $C_i \rightarrow \alpha_k$ , $C_i \rightarrow \alpha_k C_i$
  - produkcije generiraju isti jezik

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF
  - provodi se u četiri koraka:

  **1) korak**:

  - produkcije gramatike G preurediti u CNF
    produkcije su oblika $A \rightarrow BC$ ili $A \rightarrow a$
  - preimenovati m nezavršnih znakova u $D_i$ , i=1...m
    produkcije su oblika $D_i \rightarrow D_jD_k$ ili $D_i \rightarrow a$
  - produkcije $D_i \rightarrow a$ su u GNF

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF

  **2) korak**:

  – produkcije oblika $D_i \rightarrow D_j D_k$ redom od i=1 do i=m
  preurede se u oblik $D_i \rightarrow D_j \beta$, $j>i$

  – za $D_1$ , produkcije su oblika $D_1 \rightarrow D_j D_k$ , $j \geq 1$
    - za j>1 produkcija je u prihvatljivom obliku
    - za j=1 preuredimo razrješenjem lijeve rekurzije, pojavi se $C_1$

  – za $D_i \rightarrow D_j D_k$, i>1moguće je slijedeće:
    - za j<i preuredimo zamjenom lijevog nezavršnog znaka; zamjenjujemo iterativno sve dok ne postignemo j≥i
    - za j=i preuredimo razrješenjem lijeve rekurzije, pojavi se $C_i$
    - za j>i produkcija je u prihvatljivom obliku

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF

  **2) korak -** nastavak:

  – sve produkcije su oblika:

  - $D_i \rightarrow D_j \beta$, $j>i$
  - $D_i \rightarrow a \beta$, $a \in T$ produkcija je u GNF
  - $C_i \rightarrow D_j \xi$, $\xi$ niz znakova iz $\{D_i\} \cup \{C_i\}$
  - $D_m \rightarrow a\alpha$, produkcija je u GNF

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF

  **3) korak**:

  – produkcije oblika $D_i \rightarrow D_j \beta$ redom od i=m-1 do i=1
  preurede se u oblik $D_i \rightarrow a\alpha\beta$

  – za $D_{m-1}$ , produkcije su oblika $D_{m-1} \rightarrow D_m\alpha \mid a\alpha$
    - $D_{m-1} \rightarrow a\alpha$ produkcija je u prihvatljivom obliku
    - $D_{m-1} \rightarrow D_m\alpha$ koristimo sigurno $D_m \rightarrow a\alpha$ , dobijemo GNF

  – za $D_{m-k}$ , k>1, imamo produkcije $D_{m-k} \rightarrow D_j\alpha \mid a\alpha$, j>m-k
    - sigurno postoji produkcija $D_j \rightarrow a\alpha$:
    - preuredimo u oblik $D_{m-k} \rightarrow a\alpha\beta$, produkcija je u GNF

  – sve produkcije su oblika:
    - $D_i \rightarrow a\,\beta$, i $C_i \rightarrow D_j\,\xi$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF

  **4) korak**:

  – produkcije oblika $C_i \rightarrow D_j \xi$
  preurede se u oblik $C_i \rightarrow a\beta$

  – u koraku 3 je postignuto $D_i \rightarrow a\,\beta$

    - primijenimo algoritam zamjene krajnje lijevog nezavršnog znaka
    - dobije se $C_i \rightarrow a\,\alpha\,\xi = C_i \rightarrow a\beta$, produkcija je u GNF

  – sve produkcije su oblika:

    - $D_i \rightarrow a\,\beta$, i $C_i \rightarrow a\,\beta$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF - PRIMJER
  - primjer: G = ({S, A, B}, {a,b}, P, S) zadana u CNF

    1) $S \rightarrow AB$ 2) $A \rightarrow BS$ 4) $B \rightarrow SA$

    3) $A \rightarrow b$ 5) $B \rightarrow a$
  - korak 1:
    - gramatika je već u CNF
    - zamijenimo S, A i B sa $D_1$ , $D_2$ i $D_3$

      1) $D_1 \rightarrow D_2\ D_3$ 2) $D_2 \rightarrow D_3\ D_1$ 4) $D_3 \rightarrow D_1\ D_2$

      3) $D_2 \rightarrow b$ 5) $D_3 \rightarrow a$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF - PRIMJER
  - korak 2:
    - produkcije 3 i 5 su već u GNF, $D_2 \rightarrow b$ i $D_3 \rightarrow a$
    - produkcija 1 $D_1 \rightarrow D_2\ D_3$ je OK, j>i
    - produkcija 2 $D_2 \rightarrow D_3\ D_1$ je OK, j>i
    - produkciju 4 transformiramo zamjenom sa 1 pa 2 i 3:
      $D_3 \rightarrow D_1\ D_2 \Rightarrow D_3 \rightarrow D_2\ D_3\ D_2 \Rightarrow D_3 \rightarrow D_3\ D_1\ D_3\ D_2 \mid b\ D_3\ D_2$
    - dobivena produkcija $D_3 \rightarrow b\ D_3\ D_2$ je OK
    - dobivenu produkciju $D_3 \rightarrow D_3\ D_1\ D_3\ D_2$ transformiramo razrješenjem lijeve rekurzije:
      $D_3 \rightarrow D_3\ D_1\ D_3\ D_2 \Rightarrow$
      $D_3 \rightarrow b\ D_3\ D_2\ C_3 \mid a\ C_3$ i $C_3 \rightarrow D_1\ D_3\ D_2 \mid D_1\ D_3\ D_2\ C_3$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF - PRIMJER
  - korak 3:
    - počinjemo sa $D_2$, dobije se:
      $D_2 \rightarrow D_3\ D_1 \Rightarrow D_2 \rightarrow a\ D_1 \mid b\ D_3\ D_2\ D_1 \mid b\ D_3\ D_2\ C_3\ D_1 \mid a\ C_3\ D_1$
      od ranije imamo $D_2 \rightarrow b$
    - nastavimo sa $D_1$, dobije se:
      $D_1 \rightarrow D_2\ D_3 \Rightarrow D_1 \rightarrow b\ D_3 \mid a\ D_1\ D_3 \mid b\ D_3\ D_2\ D_1 D_3 \mid$
      $\mid b\ D_3\ D_2\ C_3\ D_1\ D_3 \mid a\ C_3\ D_1\ D_3$
    - od ranije imamo
      $D_3 \rightarrow b\ D_3\ D_2\ C_3 \mid a\ C_3 \mid b\ D_3\ D_2 \mid a$
      $C_3 \rightarrow D_1\ D_3\ D_2 \mid D_1\ D_3\ D_2\ C_3$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF - PRIMJER
  - korak 4:
    - dovodimo $C_3 \rightarrow D_1\ D_3\ D_2 \mid D_1\ D_3\ D_2\ C_3$ u GNF zamjenom:
      $C_3 \rightarrow D_1\ D_3\ D_2 \Rightarrow C_3 \rightarrow b\ D_3\ D_3\ D_2 \mid a\ D_1\ D_3\ D_3\ D_2 \mid$
      $\mid b\ D_3\ D_2\ D_1 D_3\ D_3\ D_2 \mid b\ D_3\ D_2\ C_3\ D_1\ D_3\ D_3\ D_2 \mid$
      $\mid a\ C_3\ D_1\ D_3\ D_3\ D_2$
    - i
      $C_3 \rightarrow D_1\ D_3\ D_2\ C_3 \Rightarrow C_3 \rightarrow b\ D_3\ D_3\ D_2\ C_3 \mid a\ D_1\ D_3\ D_3\ D_2\ C_3 \mid$
      $\mid b\ D_3\ D_2\ D_1 D_3\ D_3\ D_2\ C_3 \mid b\ D_3\ D_2\ C_3\ D_1\ D_3\ D_3\ D_2\ C_3 \mid$
      $\mid a\ C_3\ D_1\ D_3\ D_3\ D_2\ C_3$

# Pojednostavljenje gramatike

- PRETVORBA PRODUKCIJA U GNF - PRIMJER
  - dobivena gramatika je u GNF:
    - $D_1 \rightarrow b\ D_3 \mid a\ D_1\ D_3 \mid b\ D_3\ D_2\ D_1 D_3 \mid$
      $\mid b\ D_3\ D_2\ C_3\ D_1\ D_3 \mid a\ C_3\ D_1\ D_3$
    - $D_2 \rightarrow a\ D_1 \mid b\ D_3\ D_2\ D_1 \mid b\ D_3\ D_2\ C_3\ D_1 \mid a\ C_3\ D_1 \mid b$
    - $D_3 \rightarrow b\ D_3\ D_2\ C_3 \mid a\ C_3 \mid b\ D_3\ D_2 \mid a$
    - $C_3 \rightarrow b\ D_3\ D_3\ D_2 \mid a\ D_1\ D_3\ D_3\ D_2 \mid b\ D_3\ D_2\ D_1 D_3\ D_3\ D_2 \mid$
      $\mid b\ D_3\ D_2\ C_3\ D_1\ D_3\ D_3\ D_2 \mid a\ C_3\ D_1\ D_3\ D_3\ D_2$
    - $C_3 \rightarrow b\ D_3\ D_3\ D_2\ C_3 \mid a\ D_1\ D_3\ D_3\ D_2\ C_3 \mid b\ D_3\ D_2\ D_1 D_3\ D_3\ D_2\ C_3$
      $\mid$
      $\mid b\ D_3\ D_2\ C_3\ D_1\ D_3\ D_3\ D_2\ C_3 \mid a\ C_3\ D_1\ D_3\ D_3\ D_2\ C_3$

# 3.1.3. Parsiranje niza

- PARSIRANJE JE POSTUPAK
  - prepoznavanja niza
  - izgradnje generativnog stabla
  - to je stablo parsiranja
  - razlikujemo metode:
    - od vrha prema dnu, započinje korijenom, odnosno od početnog nezavršnog znaka
    - od dna prema vrhu, započinje listovima, odnosno od završnih znakova

# Parsiranje niza

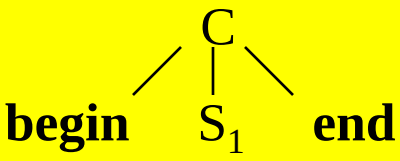
- PARSIRANJE OD VRHA PREMA DNU
  - zadana je gramatika G = (V, T, P, S)
  - parsiranje započinje sa početnim nezavršnim znakom S
  - produkcije gramatike se primjenjuju sve dok se listovi stabla ne označe završnim znakovima traženog niza w
  - ovo parsiranje se široko primjenjuje zbog učinkovitosti
  - za višestruki izbor zamjene, treba ispitati sve kombinacije

# Parsiranje niza

- PARSIRANJE OD VRHA PREMA DNU PRIMJER
  - gramatika G generira **begin-end** blokove
  - generiraju se naredbe pridruživanja i **while** petlje
  - zadano je: V = {C, S, $S_1$, $S_2$}; C je početni
    T = {**begin, end, while, do, ;, :=, <>, var** }
    P = { C $\rightarrow$ **begin** $S_1$ **end**, $S_1 \rightarrow SS_2$, $S_2 \rightarrow ;S_1 \mid \varepsilon$
    S $\rightarrow$ **var := var | while var <> var do** S | **begin** $S_1$ **end** }
  - **begin, end, while, do** su ključne riječi, **var** je varijabla, **:=** je pridruživanje, <> je nejednakost, a **;** ulančavanje

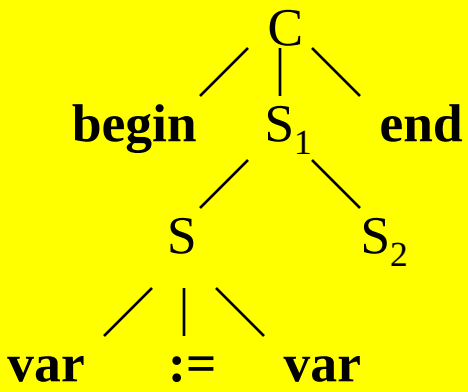
# Parsiranje niza

- TD PARSIRANJE PRIMJER
  - parsirati niz: **begin var := var ;**
    **while var <> var do**
    **var := var**
    **end**
  - započnimo korijenom i jedinom mogućom produkcijom:
    - C $\rightarrow$ **begin** $S_1$ **end**

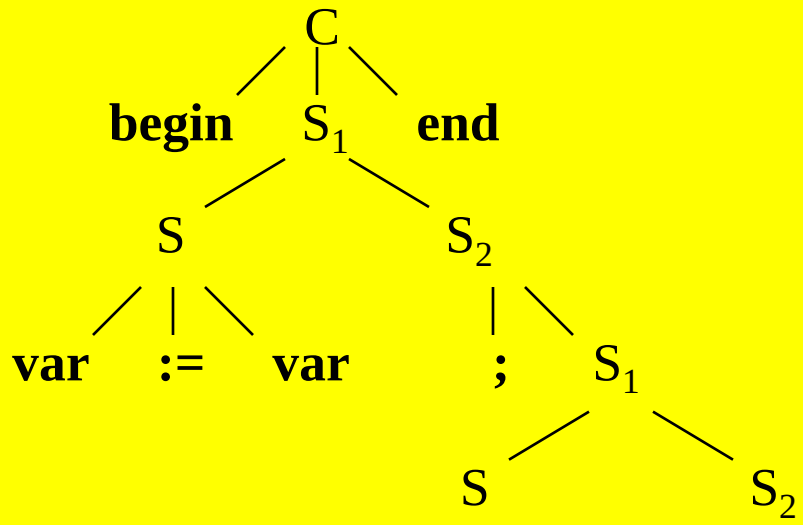


# Parsiranje niza

- TD PARSIRANJE PRIMJER
  - nadalje koristimo:
    - $S_1 \rightarrow SS_2$ i
      $S \rightarrow$ **var := var**

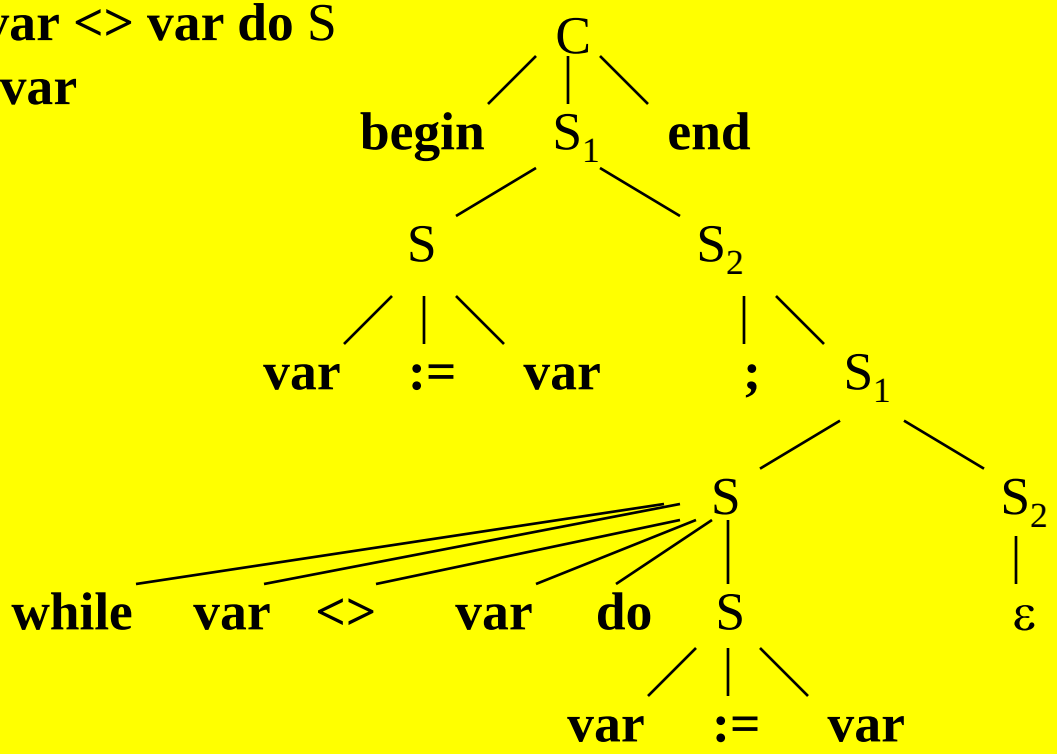



- $S_2 \rightarrow ; S_1$i
- $S_1 \rightarrow SS_2$



# Parsiranje niza

- TD PARSIRANJE PRIMJER
  - nadalje koristimo:
    - S → **while var <> var do** S
      S → **var := var**
      $S_2$ → ε



# Parsiranje niza

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

# Parsiranje niza

- LL(k) PARSIRANJE
  - LL(k) gramatika i parsiranje od vrha prema dnu je:
    - L: ulazni niz čitamo s lijeva na desno (left to right scanning)
    - L: mijenjamo krajnje lijevi nezavršni znak (leftmost derivation)
    - (k): odluku donosimo na osnovu k pročitanih znakova ulaznog niza
  - gramatika zadana Greibacnovom normalnom formom pogodna je za učinkovito ostvarenje LL(1) parsera
  - u realizaciji LL(1) parsera primjenjujemo tehniku **rekurzivnog spusta**

# Parsiranje niza

- TEHNIKA REKURZIVNOG SPUSTA
  - služi za programsko ostvarenje LL(k) parsera
  - koristi 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
    - moguća je i indirektna rekurzija, posredstvom drugih potprograma

# Parsiranje niza

- TEHNIKA REKURZIVNOG SPUSTA PRIMJER
  - za gramatiku G = ({C, S, $S_1$, $S_2$},
    {**begin, end, while, do, ;, :=, <>, var** }, P, C)
    s produkcijama:
    1) C $\rightarrow$ **begin** $S_1$ **end**, 3) $S_2 \rightarrow ;S_1$ 5) S $\rightarrow$ **var := var**
    2) $S_1 \rightarrow SS_2$, 4) $S_2 \rightarrow \varepsilon$ 6) S $\rightarrow$ **while var <> var do** S
    7) S $\rightarrow$ **begin** $S_1$ **end**
  - tehnikom rekurzivnog spusta programski ostvariti parser
  - za 4 nezavršna znaka imamo 4 potprograma
  - neka je $\perp$ oznaka kraja ulaznog niza w

# Parsiranje niza

- TEHNIKA REKURZIVNOG SPUSTA PRIMJER

```
GlavniProgram()
{
  ulaz = čitaj(znak);
  C();
  ako(ulaz = ⊥) piši (“w ∈ L(G)”);
   inače piši (“w ∉ L(G)”);
}


C()       //produkcija 1)
{
  ako(ulaz <> begin) piši (“w ∉ L(G)”);
  ulaz = čitaj(znak)
  S1();
  ako(ulaz <> end) piši (“w ∉ L(G)”);
  ulaz = čitaj(znak);
}
```

# Parsiranje niza

- TEHNIKA REKURZIVNOG SPUSTA PRIMJER

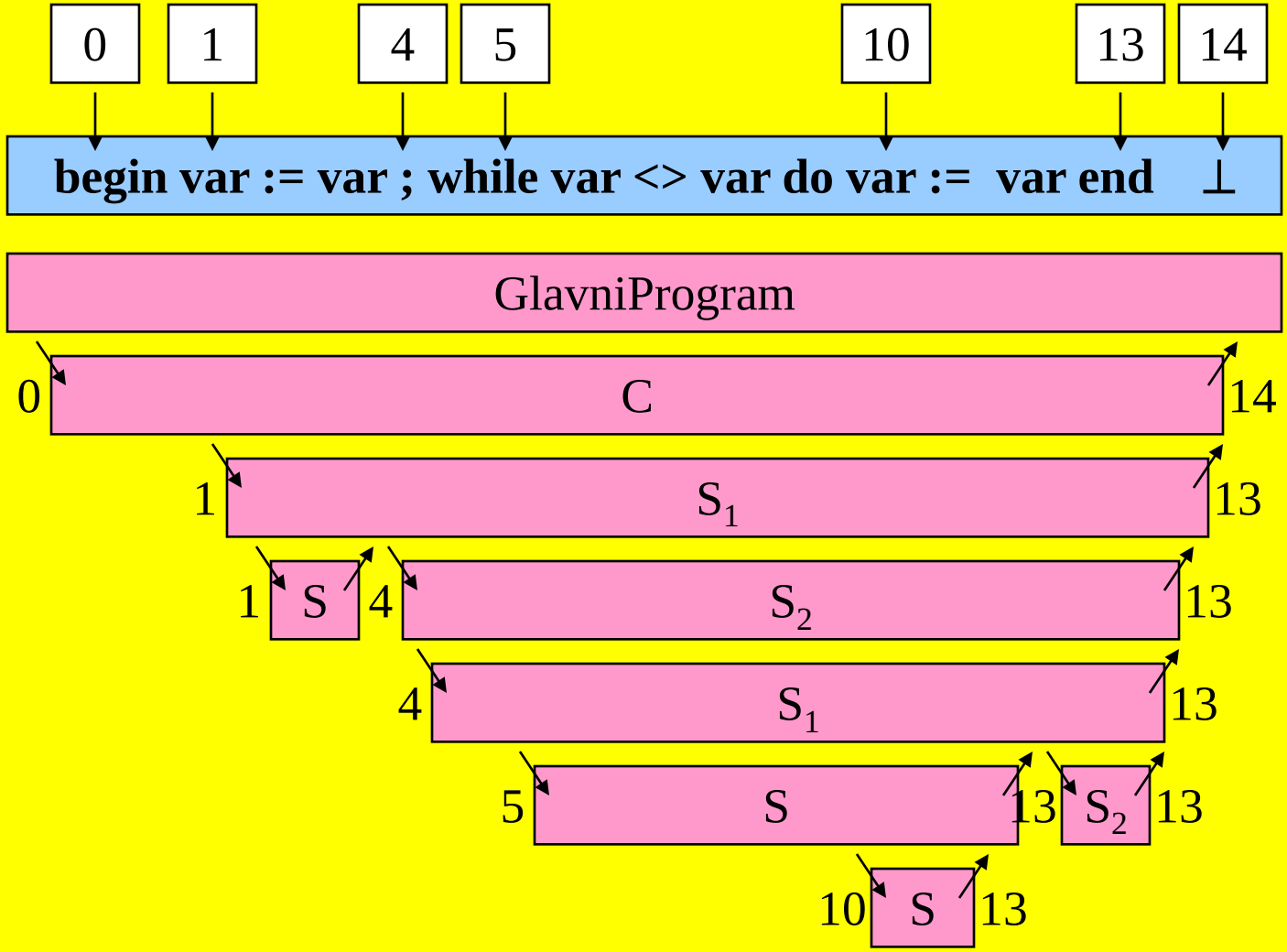
```
S1()        //produkcija 2)
{
  S();
  S2();
}


S2()        //produkcija 3), i 4) implicitno
{
  ako(ulaz == ;)
    {
          ulaz = čitaj(znak)
          S1()
    }
}
```

- TEHNIKA REKURZIVNOG SPUSTA PRIMJER

```
S()          //produkcije 5), 6) i 7)
{ slučaj(ulaz)
  {var:  ulaz = čitaj(znak)
         ako(ulaz <> :=) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
         ako(ulaz <> var) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
   while: ulaz = čitaj(znak)
         ako(ulaz <> var) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
         ako(ulaz <> <>) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
         ako(ulaz <> var) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
         ako(ulaz <> do) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
         S()
   begin:ulaz = čitaj(znak)
         S1()
         ako(ulaz <> end) piši ("w ∉ L(G)");
         ulaz = čitaj(znak)
    ostalo: piši ("w ∉ L(G)");
  }
}
```

- TEHNIKA REKURZIVNOG SPUSTA PRIMJER
  - dijagram pozivanja potprograma za zadani niz



# Parsiranje niza

- 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
    - svaki dio ispituje podniz koji slijedi
    - ako znak pročitan tijekom rada ne odgovara niti jednoj desnoj strani, niz se odbacuje

# Parsiranje niza

- PARSER S REKURZIVNIM SPUSTOM
  - 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)
    - za svaki nezavršni znak koji nije na krajnjem lijevom mjestu niza prvo se pročita sljedeći znak, pa se poziva potprogram

# Parsiranje niza

- PARSIRANJE OD DNA PREMA VRHU
  - zadana je gramatika G = (V, T, P, S)
  - 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 obrnutih produkcija su **redukcije**
  - postupak se koristi u generatorima parsera

# Parsiranje niza

- PARSIRANJE OD DNA KA VRHU PRIMJER
  - zadana je gramatika G = ({E,T,F}, {**var**,+,*,**(**,**)**}, P, E) s produkcijama:

    1) E → E + T  3) T → T * F  5) F → **(**E**)**

    2) E → T  4) T → F  6) F → **var**
  - parsirati niz **var+var*var**
  - niz čitamo s lijeva na desno i primijenimo redukciju 6)


**var+var*var**⇐ F+**var*var**

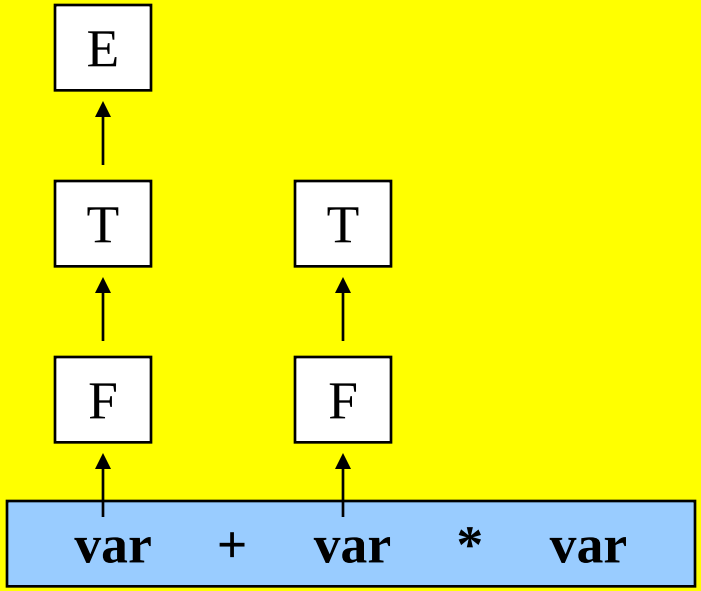
# Parsiranje niza

- PARSIRANJE OD DNA KA VRHU PRIMJER
  - nadalje jednoznačno primijenimo redukcije 4) 2) 6) 4)

F+**var*var**⇐ T+**var*var**

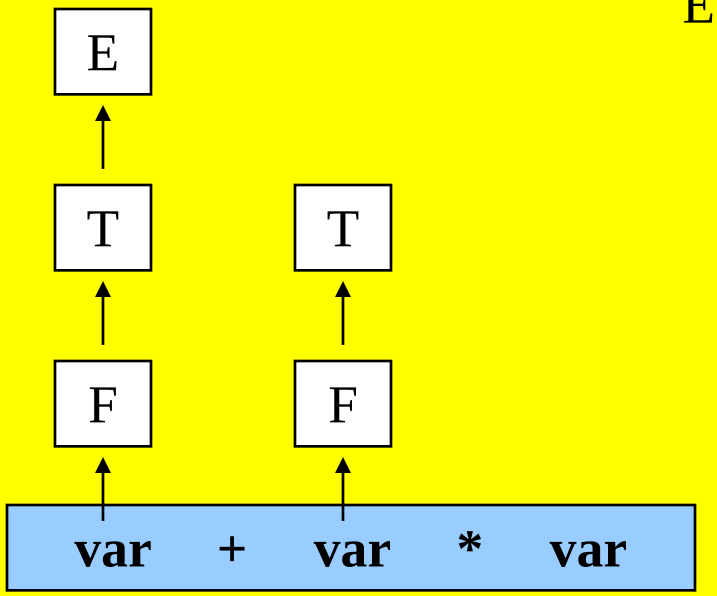
⇐ E+**var*var**

⇐ E+T***var**



# Parsiranje niza

- PARSIRANJE OD DNA KA VRHU PRIMJER
  - dalji rad nije jednoznačan,
    moguće je primijeniti redukcije 1), 2) ili 6):



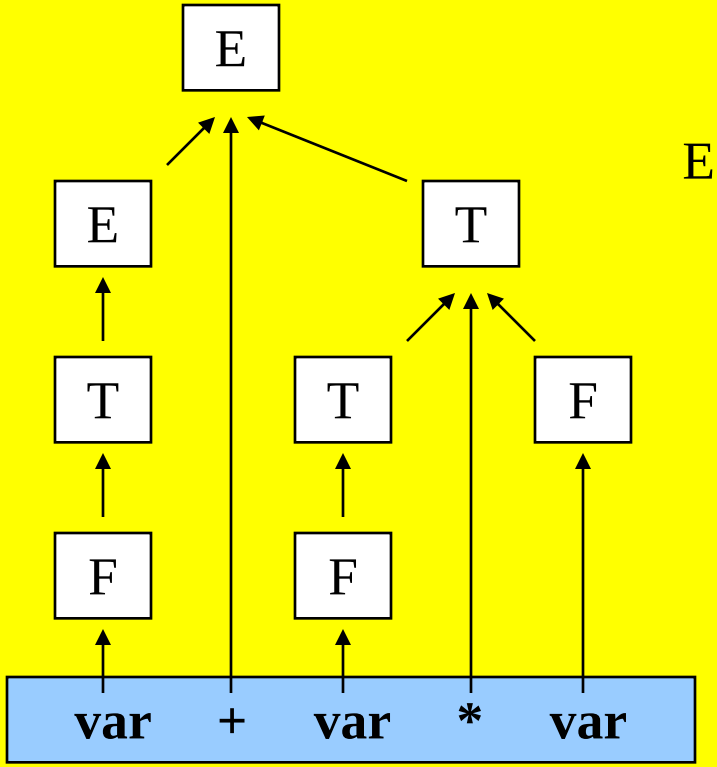

E+T**var** ⇐ E***var**

⇐ E+E***var**

⇐ E+T*F

# Parsiranje niza

- PARSIRANJE OD DNA KA VRHU PRIMJER
  - napredak postižemo redukcijom 6), pa 3) i 1):




E+T**var** ⇐ E+T*F

⇐ E+T

⇐ E

# Parsiranje niza

- PARSIRANJE OD DNA PREMA VRHU
  - BU parsiranje nije jednoznačno
  - te redukcije su **višeznačne**
  - izaberemo jednu od višetnačnih redukcija
    i nastojimo izgraditi generativno stablo
  - ako ne uspijemo, ide se unatrag
    i pokuša slijedećom od višeznačnih redukcija
  - postupak **unazadnog pretraživanja** ponavlja se
    dok se ne ispitaju sve višeznačne mogućnosti
  - uspijemo li izgraditi stablo, niz w je dio jezika L(G)
  - učinkovitost ovisi ovišeznačnosti

# Parsiranje niza

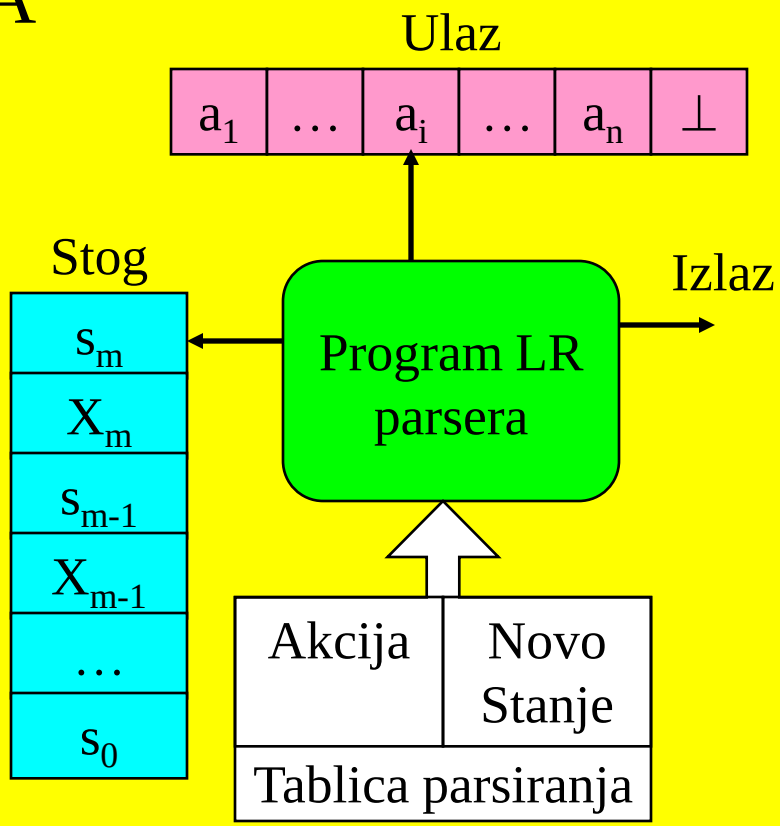
- LR PARSER
  - LR(k) parser koristi metodu od dna prema vrhu
    - L: ulazni niz čitamo s lijeva na desno (left to right scanning)
    - R: mijenja  krajnje desni nezavršni znak (rightmost derivation)
    - (k): za donošenje odluke potrebno je pročitati najviše k znakova ulaznog niza
  - LR(k) je najopćenitiji postupak parsiranja od dna premavrhu
  - provodi se bez unazadnog pretraživanja
  - primjenjuje se na široj skup CFG, ali ne na sve
  - to su LR gramatike, jer je moguće izgraditi LR parser
  - ostale nisu LR gramatike

# Parsiranje niza

- LR PARSER
  - konstrukcija LR(k) parsera je složena
  - gradi se posebnim generatorom
    - generator koristi skup produkcija
    - postoje različite klase složenosti
    - gradi se tablica parsiranja
    - program parsera je uvijek isti

# Parsiranje niza

- MODEL LR PARSERA
  - model LR parsera ima
    - ulazni spremnik
    - potisni LIFO stog
    - program LR parsera
    - tablicu parsiranja
    - izlaz



# Parsiranje niza

- MODEL LR PARSERA
  - u stog se sprema niz $s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m$
    - $s_m$ je na vrhu stoga
    - $X_i$ su završni ili nezavršni znakovi gramatike
    - $s_i$ su stanja
    - stanje na vrhu stoga $s_m$ jednoznačno određuje sadržaj stoga
    - 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

# Parsiranje niza

- MODEL LR PARSERA
  - spojimo stog i ulazni spremnik u jedinstvenu listu:

$$s_0\, X_1\, s_1\, X_2\, s_2 \ldots X_m\, s_m \quad , \quad a_i\, a_{i+1} \ldots a_n \perp$$

vrh stoga — znak koji se čita

  - to je **konfiguracija** LR parsera
  - ako izuzmemo oznake stanja dobijemo niz:

$$X_1\, X_2 \ldots X_m \quad , \quad a_i\, a_{i+1} \ldots a_n \perp$$

  - to je jedan od međunizova dobiven zamjenom krajnjeg desnog nezavršnog znaka

# Parsiranje niza

- MODEL LR PARSERA
  - neka je LR parser u konfiguraciji:

$$s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m \quad , \quad a_i a_{i+1} \ldots a_n \perp$$

  - iz tablice se čita Akcija koja mijenja konfiguraciju:
    - **Pomak s**: spremi na stog $a_i$ i s, čitaj novi znak:

$$s_0 X_1 s_1 X_2 s_2 \ldots X_m s_m a_i s \quad , \quad a_{i+1} \ldots a_n \perp$$

    - **Reduciraj A $\rightarrow$ β**: uzmi 2r=2|β| znakova sa stoga i zamj. A

$$s_0 X_1 s_1 X_2 s_2 \ldots X_{m-r} s_{m-r} A s \quad , a_i a_{i+1} \ldots a_n \perp$$

    - **Prihvati**: prihvaća niz
    - **Odbaci**: odbacuje niz

# Parsiranje niza

- ALGORITAM LR PARSERA
  - Ulaz: zadani niz w i tablica parsiranja
  - Izlaz:prihvaćanje ili neprihvaćanje niza
  - Postupak parsiranja:
    - na početku na stogu je početno stanje $s_0$ a na ulazu niz w⊥
    - 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
    }
  }
}
```

# Parsiranje niza

- PROGRAM LR PARSERA PRIMJER
  - zadana je gramatika G = ({E,T,F}, {**var**,+,*,**(**,**)**}, P, E) s produkcijama:

    1) $E \rightarrow E + T$ 3) $T \rightarrow T * F$ 5) $F \rightarrow$ **(**E**)**

    2) $E \rightarrow T$ 4) $T \rightarrow F$ 6) $F \rightarrow$ **var**
  - parsirati niz **var+var*var**

# Parsiranje niza

- PROGRAM LR PARSERA PRIMJER
  - tablica parsera je:

| | Akcija | | | | | | NovoStanje | | |
|---|---|---|---|---|---|---|---|---|---|
| Stanje | var | + | * | ( | ) | ⊥ | E | T | F |
| 0 | s5 | | | s4 | | | 1 | 2 | 3 |
| 1 | | s6 | | | | prihvati | | | |
| 2 | | r2 | s7 | | r2 | r2 | | | |
| 3 | | r4 | r4 | | r4 | r4 | | | |
| 4 | s5 | | | s4 | | | 8 | 2 | 3 |
| 5 | | r6 | r6 | | r6 | r6 | | | |
| 6 | s5 | | | s4 | | | | 9 | 3 |
| 7 | s5 | | | s4 | | | | | 10 |
| 8 | | s6 | | | s11 | | | | |
| 9 | | r1 | s7 | | r1 | r1 | | | |
| 10 | | r3 | r3 | | r3 | r3 | | | |
| 11 | | r5 | r5 | | r3 | r3 | | | |

si - stavi znak i stanje i na stog

rj - reduciraj produkcijom j

prihvati- niz je OK

praznini -odbaci

početno stanje je0

- PROGRAM LR PARSERA PRIMJER
  - za niz **var+var*var** imamo transformacije

| Korak | Stog | Ulaz | Akcija |
|---|---|---|---|
| 1. | 0 | var+var*var ⊥ | pomak 5 |
| 2. | 0 var 5 | +var*var ⊥ | r F → **var** |
| 3. | 0 F 3 | +var*var ⊥ | r T → F |
| 4. | 0 T 2 | +var*var ⊥ | r E → T |
| 5. | 0 E 1 | +var*var ⊥ | pomak 6 |
| 6. | 0 E 1 + 6 | var*var ⊥ | pomak 5 |
| 7. | 0 E 1 + 6 var 5 | *var ⊥ | r F → **var** |
| 8. | 0 E 1 + 6 F 3 | *var ⊥ | r T → F |
| 9. | 0 E 1 + 6 T 9 | *var ⊥ | pomak 7 |
| 10. | 0 E 1 + 6 T 9 * 7 | var ⊥ | pomak 5 |
| 11. | 0 E 1 + 6 T 9 * 7 var 5 | ⊥ | r F → **var** |
| 12. | 0 E 1 + 6 T 9 * 7 F 10 | ⊥ | r T →T* F |
| 13. | 0 E 1 + 6 T 9 | ⊥ | r E → E+T |
| 14. | 0 E 1 | ⊥ | prihvati |