Prevodjenje programskih jezika – beleške sa predavanja Jezici i gramatike

Milan Banković

*Matematički fakultet,
Univerzitet u Beogradu

Jesenji semestar 2024/25.

Pregled

Azbuka i reč

Definicija 1

Azbuka Σ *je konačan, neprazan skup elemenata koje nazivamo slovima ili simbolima azbuke. Reč* $w = a_1 a_2 \dots a_n$ *nad* Σ *je bilo koji konačan niz simbola* $a_i \in \Sigma$. *Specijalno, postoji i prazna reč koju obično označavamo sa* ε. *Skup svih reči nad* Σ *označavamo sa* Σ^*. *Broj simbola od kojih se sastoji reč* w *nazivamo dužinom reči* w *i označavamo sa* $|w|$. *Specijalno, prazna reč* ε *je dužine nula:* $|\varepsilon| = 0$.

Primedba

Primetimo da reči nad azbukom Σ ima beskonačno mnogo, iako su sve reči konačne dužine. Ovo je zato što konačnih dužina ima beskonačno mnogo (skup prirodnih brojeva $\mathbb{N}_0$)

Azbuka i reč

Primer

Neka je $\Sigma = \{0, 1\}$. Reči nad ovom azbukom su sve konačne niske nula i jedinica (npr. 0, 010, 11110, i sl.) Ovu azbuku nazivaćemo i binarnom azbukom, jer se u pomoću nje mogu zapisivati binarni brojevi. Dužine navedenih reči su: $|0| = 1$, $|010| = 3$, $|11110| = 5$.

Primer

Azbuka koju ćemo često koristiti u našim primerima je $\Sigma = \{a, b\}$. Primeri reči nad ovom azbukom su baba, abba, abab, aaaa, bbaaaabb ...

Primer

Većina modernih programskih jezika kao azbuku koriste ASCII skup karaktera. Ova azbuka ima 128 simbola. Svi programi su zapravo ASCII tekstualni fajlovi, te se mogu razumeti kao nizovi ASCII karaktera, tj. kao reči nad ovom azbukom.

Primer

S obzirom da sintaksni analizator na ulazu ima niz tokena, možemo razumeti da su nizovi tokena zapravo reči nad azbukom čiji su simboli tokeni.

Dopisivanje reči

Definicija 2

Neka su date dve reči $u = a_1 a_2 \dots a_n$ *i* $v = b_1 b_2 \dots b_m$ *nad azbukom* Σ*. Tada definišemo:* $u \cdot v := a_1 a_2 \dots a_n b_1 b_2 \dots b_m$*. Operaciju* $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$ *nazivamo operacijom dopisivanja (ili konkatenacije) reči. Specijalno, definišemo:* $u \cdot \varepsilon = \varepsilon \cdot u = u$ *za svaku reč* $u \in \Sigma^*$*.*

Primedba

Može se dokazati da važi: $|u \cdot v| = |u| + |v|$.

Primedba

Lako se vidi da je $(\Sigma^*, \cdot)$ nekomutativni monoid. Operacija $\cdot$ je asocijativna, ali ne i komutativna. Jedini invertibilni element ovog monoida je njegov neutralni element ε.

Primedba

Umesto $u \cdot v$ često pišemo samo uv.

Primer

Dopisivanjem reči baba i abba dobijamo reč babaabba.

Stepen reči

Definicija 3

Operacija n-tog stepena reči w^n definiše se rekurzivno na sledeći način:

- $w^0 = \varepsilon$
- $w^{n+1} = w^n \cdot w$

Primedbe

- n-ti stepen reči nastaje dopisivanjem reči same na sebe n puta
- Operacija stepena ima uobičajene osobine: $w^n \cdot w^m = w^{n+m}$, $(w^n)^m = w^{n \cdot m}$

Primer

$(ab)^3 = ababab$

Jezik

Definicija 4

Jezik nad azbukom Σ je bilo koji podskup od Σ^*.

Primedba

Jezik može biti konačan ili prebrojivo beskonačan.

Primer

Neka je data azbuka $\Sigma = \{a, b\}$. Neki primeri jezika nad ovom azbukom su:

- *Skup $L_1 = \{\varepsilon, a, b, aa, bb, ab, ba\}$ je jezik svih reči dužine najviše dva. Ovaj jezik je konačan.*
- *Skup $L_2 = \{a, aa, aaa, aaaa, \dots\}$ je jezik svih reči koje se sastoje samo iz slova a. Ovaj jezik je beskonačan.*
- *Neka je L_3 skup svih reči nad Σ takvih da sadrže jednak broj pojavljivanja slova a i b. Ovo je jedan (beskonačan) jezik nad Σ.*
- *Neka je L_4 skup svih reči nad Σ takvih da sadrže paran broj pojavljivanja slova a. Ovaj skup je jedan jezik nad Σ.*
- *...*

Jezik

Važna primedba

Jezik $\varnothing = \{\}$ je prazan jezik. Ovaj jezik postoji nad svakom azbukom. Takodje, nad svakom azbukom postoji i jezik $L_\varepsilon = \{\varepsilon\}$. Ova dva jezika nisu jednaka (ovaj drugi ima jednu reč u sebi, dok je prvi prazan).

Operacije nad jezicima

Primedba

Kako su jezici skupovi, nad njima su definisane sve uobičajene skupovne operacije:

- $L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$
- $L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$
- $L_1 \setminus L_2 = \{w \mid w \in L_1 \wedge w \notin L_2\}$
- $\complement L = \Sigma^* \setminus L$

Dodatno, definišemo i neke specifične operacije za jezike:

Definicija 5

- *Dopisivanje jezika:* $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \wedge v \in L_2\}$
- *Stepen jezika:* $L^0 = L_\varepsilon = \{\varepsilon\}$, $L^{n+1} = L^n \cdot L$
- *Klinijevo zatvorenje (iteracija):* $L^* = \bigcup_{k=0}^{\infty} L^k = L^0 \cup L^1 \cup L^2 \cup \ldots$
- *Pozitivno zatvorenje:* $L^+ = \bigcup_{k=1}^{\infty} L^k = L^1 \cup L^2 \cup \ldots$
- *Opcioni operator:* $L^? = L \cup \{\varepsilon\}$

Operacije nad jezicima

Osobine operacija nad jezicima

- Važi $L^* = L^+ \cup \{\varepsilon\}$, ali ne mora da važi $L^+ = L^* \setminus \{\varepsilon\}$
  - Ako $\varepsilon \in L$, tada $\varepsilon \in L^+$, dok $\varepsilon \notin L^* \setminus \{\varepsilon\}$
- Ukoliko jezik L sadrži bar jednu reč različitu od prazne reči, jezici L^* i L^+ su beskonačni
  - Ako je $w \in L$ i $w \neq \varepsilon$, tada su $w^1, w^2, w^3, \ldots$ različite reči koje sve pripadaju ovim jezicima
- Važi $L^? = L$ akko $\varepsilon \in L$
- Operacija dopisivanja jezika je asocijativna, ali nije komutativna
  - Ove osobine slede iz osobina operacije dopisivanja za reči
- Neutralni element za operaciju dopisivanja je $L_\varepsilon = \{\varepsilon\}$ $(L_\epsilon \cdot L = L \cdot L_\epsilon = L)$
  - $L_\epsilon \cdot L = \{\varepsilon \cdot u \mid u \in L\} = \{u \mid u \in L\} = L$
- Važi $L \cdot \varnothing = \varnothing$
  - Ovo je zato što je $\varnothing$ prazan skup
- Važi $L \cdot (L_1 \cup L_2) = L \cdot L_1 \cup L \cdot L_2$
  - $x \in L \cdot (L_1 \cup L_2) \Leftrightarrow \exists uv.\ x = uv \wedge u \in L \wedge v \in L_1 \cup L_2 \Leftrightarrow \exists uv.\ x = uv \wedge u \in L \wedge (v \in L_1 \vee v \in L_2) \Leftrightarrow \exists uv.(x = uv \wedge u \in L \wedge v \in L_1) \vee (x = uv \wedge u \in L \wedge v \in L_2) \Leftrightarrow (\exists uv.\ x = uv \wedge u \in L \wedge v \in L_1) \vee (\exists uv.\ x = uv \wedge u \in L \wedge v \in L_2) \Leftrightarrow x \in L \cdot L_1 \vee x \in L \cdot L_2 \Leftrightarrow x \in L \cdot L_1 \cup L \cdot L_2$

Operacije nad jezicima

Da li važi ova osobina?

$L \cdot (L_1 \cap L_2) = L \cdot L_1 \cap L \cdot L_2$

U jednom smeru važi

$x \in L \cdot (L_1 \cap L_2) \Leftrightarrow \exists uv.\ x = uv \wedge u \in L \wedge v \in L_1 \cap L_2 \Leftrightarrow \exists uv.\ x = uv \wedge u \in L \wedge v \in L_1 \wedge v \in L_2 \Leftrightarrow \exists uv.(x = uv \wedge u \in L \wedge v \in L_1) \wedge (x = uv \wedge u \in L \wedge v \in L_2) \Rightarrow (\exists uv.\ x = uv \wedge u \in L \wedge v \in L_1) \wedge (\exists uv.\ x = uv \wedge u \in L \wedge v \in L_2) \Leftrightarrow x \in L \cdot L_1 \wedge x \in L \cdot L_2 \Leftrightarrow x \in L \cdot L_1 \cap L \cdot L_2$

U drugom smeru ne važi!!

Ovo je zato što u logici prvog reda implikacija
$\exists z.\ p(z) \wedge q(z) \Rightarrow (\exists z.\ p(z)) \wedge (\exists z.\ q(z))$ važi samo u jednom smeru.

Kontraprimer

$L = \{a, aa\}$, $L_1 = \{a\}$, $L_2 = \{aa\}$. Jezici L_1 i L_2 su disjunktni, pa je $L \cdot (L_1 \cap L_2)$ prazan. Sa druge strane, reč aaa pripada i jeziku $L \cdot L_1$ i jeziku $L \cdot L_2$, pa presek ovih jezika nije prazan.

Pregled

Regularni jezici

Definicija 6

Klasa regularnih jezika nad azbukom Σ (u oznaci $\mathcal{R}(\Sigma)$) je najmanji skup jezika nad Σ koji zadovoljava sledeće osobine:

- *Prazan jezik $\varnothing$ je regularan*
- *Jezik $L_\varepsilon = \{\varepsilon\}$ je regularan*
- *Za svaki simbol $a \in \Sigma$, jezik $\{a\}$ je regularan*
- *Ako su L_1 i L_2 regularni, tada su i jezici $L_1 \cup L_2$ i $L_1 \cdot L_2$ regularni*
- *Ako je L regularan, tada je i jezik L^* regularan*

Primedba

Regularni jezici su oni i samo oni jezici koji se mogu dobiti polazeći od jezika $\varnothing$, L_ε i $\{a\}$ ($a \in \Sigma$), konačnom primenom operacija unije, nadovezivanja i Klinijevog zatvorenja.

Primeri

Primer

- *Neka je $w \in \Sigma^*$ proizvoljna reč nad Σ. Tada je jezik $\{w\}$ regularan. Zaista, ako je $w = a_1 a_2 \dots a_n$, tada je $\{w\} = \{a_1\} \cdot \{a_2\} \cdot \dots \cdot \{a_n\}$, pa je regularan.*
- *Svaki konačan jezik $L = \{w_1, w_2, \dots, w_n\}$ je regularan. Zaista, on se može predstaviti kao $\{w_1\} \cup \{w_2\} \cup \dots \cup \{w_n\}$, pa je regularan.*
- *Postoje i beskonačni regularni jezici: oni se dobijaju primenom operacije Klinijevog zatvorenja*
- *Jezik $\{a^n \mid n \geq 0\} = \{\varepsilon, a, aa, aaa, \dots\}$ je regularan, jer se može predstaviti kao $\{a\}^*$*
- *Jezik $\{a^n b^m \mid n, m \geq 0\}$ je regularan, jer se može predstaviti kao $\{a\}^*\{b\}^*$*
- *Jezik Σ je regularan, jer se sastoji iz svih jednoslovnih reči, pa je konačan*
- *Jezik Σ^* (skup svih reči nad Σ) je regularan, jer se dobija primenom Klinijevog zatvorenja na jezik Σ*

Regularni jezici i operacije nad jezicima

U odnosu na koje operacije je klasa regularnih izraza zatvorena?

- Iz same definicije sledi da je klasa regularnih jezika zatvorena za uniju
- Može se pokazati da je klasa regularnih jezika zatvorena i za ostale skupovne operacije:
  - Presek dva regularna jezika je regularan
  - Komplement regularnog jezika je regularan
  - Razlika dva regularna jezika je regularan
- Dokaz ove činjenice nije trivijalan: ostavljamo ga za kasnije!!
- Takodje, iz definicije sledi da je klasa regularnih jezika zatvorena za operaciju dopisivanja jezika:
  - Posledica: ako je L regularan jezik, tada je i L^i regularan jezik za svako $i \in \mathbb{N}_0$
- Slično, iz definicije sledi i da su jezici $L^? = L \cup \{\varepsilon\}$ i $L^+ = L \cdot L^*$ takodje regularni, ako je L regularan

Regularni izrazi

Šta su regularni izrazi?

Regularni izrazi predstavljaju notaciju za kompaktno opisivanje regularnih jezika. U osnovi, iz skupovnog opisa regularnih izraza izbacujemo vitičaste zagrade, a operator unije $\cup$ zamenjujemo operatorom $|$ („ili" operator).

Definicija 7

Svakom regularnom jeziku pridružujemo regularni izraz na sledeći način:

- *Regularnom jeziku* $\varnothing$ *pridružujemo regularni izraz* $\varnothing$
- *Regularnom jeziku* $\{\varepsilon\}$ *pridružujemo regularni izraz* ε
- *Za svako* $a \in \Sigma$, *regularnom jeziku* $\{a\}$ *pridružujemo regularni izraz* a
- *Ako su regularnim jezicima* L_1 *i* L_2 *redom pridruženi regularni izrazi* r_1 *i* r_2, *tada jezicima* $L_1 \cdot L_2$ *i* $L_1 \cup L_2$ *pridružujemo regularne izraze* $r_1 r_2$ *i* $r_1|r_2$ *respektivno*
- *Ako je regularnom jeziku* L *pridružen regularni izraz* r, *tada regularnom jeziku* L^* *pridružujemo regularni izraz* r^*

Primeri

Primer

- *Za* $w \in \Sigma^*$, *jeziku* $\{w\}$ *pridružujemo regularni izraz* w
- *Jeziku* $\{w_1, w_2, \ldots, w_n\}$ *pridružujemo regularni izraz* $w_1|w_2|\ldots|w_n$
- *Jeziku* $\{a^n \mid n \geq 0\}$ *pridružujemo regularni izraz* a^*
- *Jeziku* $\{a^n b^m \mid n, m \geq 0\}$ *pridružujemo regularni izraz* $a^* b^*$

Regularni izrazi – napomene

Ekvivalentnost izraza

- Dva regularna izraza su ekvivalentna ako definišu isti regularni jezik
- Jasno je, s obzirom na definiciju regularnih operatora i osobina odgovarajućih operacija nad jezicima, da će npr. izraz $p|q$ biti ekvivalentan izrazu $q|p$, kao i da će izraz $(pq)r$ biti ekvivalentan izrazu $p(qr)$
- Medjutim, postoje i regularni izrazi koji uopšte nisu ,,slični", a ekvivalentni su
- Pitanje ekvivalentnosti regularnih izraza nije uopšte trivijalno; odgovor na ovo pitanje daćemo kasnije
- Posledično: regularni izraz koji predstavlja neki regularni jezik L i opštem slučaju nije jedinstven

Prioritet operatora

- Prilikom zapisivanja regularnih izraza, najveći prioritet ima operator zatvorenja, zatim operator dopisivanja, i na kraju ,,ili" operator
- Prioritet se može promeniti zagradama
- Na primer, izraz $a|bc^*$ je ekvivalentan izrazu $a|(b(c^*))$

Primeri

Primer

- *Neka je* $\Sigma = \{0, 1\}$. *Jezik svih reči nad ovom azbukom koje sadrže dve uzastopne nule je regularan i može se opisati izrazom* $(0|1)^*00(0|1)^*$
- *Jezik svih reči nad azbukom* $\Sigma = \{0, 1\}$ *koje sadrže paran broj nula je takodje regularan i može se opisati izrazom* $(1^*01^*0)^*1^*$
- *Neka je* Σ *skup svih ASCII simbola. Jezik koji sadrži sve identifikatore programskog jezika C se može opisati izrazom* $(a|b|\ldots|z|A|B|\ldots|Z|\_)(a|b|\ldots|z|A|B|\ldots|Z|\_|0|1|\ldots|9)^*$

Prošireni regularni izrazi

Klase simbola

Neka su $a_1, a_2, \ldots, a_n \in \Sigma$. Izraz $[a_1 a_2 \ldots a_n]$ predstavlja jezik $\{a_1, a_2 \ldots, a_n\}$. Slično, izraz $[\hat{}\, a_1 a_2 \ldots a_n]$ predstavlja jezik $\Sigma \setminus \{a_1, a_2, \ldots, a_n\}$.

Primer

Na primer, izraz `[abc]` *označava isto što i izraz* $a|b|c$.

Intervali

Pretpostavimo da je azbuka Σ uređen skup, tj. da je definisana relacija potpunog poretka $\prec$ nad azbukom Σ. Pretpostavimo da su a i b simboli iz Σ takvi da je $a \prec b$. Tada `[a-b]` označava jezik $\{c \mid a \preceq c \preceq b\}$. Slično, `[^a-b]` označava jezik $\Sigma \setminus \{c \mid a \preceq c \preceq b\}$.

Primer

Klase i intervali se mogu i kombinovati. Na primer, izraz `[abc0-9]` *predstavlja isto što i izraz* $a|b|c|0|1|2|3|4|5|6|7|8|9$.

Napomena

Klase i intervali predstavljaju samo kraći zapis regularnih izraza, tj. ne utiču na izražajnost regularnih izraza.

Prošireni regularni izrazi

Još neka proširenja

- Ako regularni izraz r označava regularni jezik L, tada izrazi r^+ i $r^?$ označavaju redom jezike L^+ i $L^?$
- Ako regularni izraz r označava regularni jezik L, tada izrazi $r^{\{m\}}$, $r^{\{-m\}}$, $r^{\{m-\}}$ i $r^{\{m-n\}}$ označavaju, respektivno, jezike L^m, $\bigcup_{i=0}^{m} L^i$, $\bigcup_{i=m}^{\infty} L^i$ i $\bigcup_{i=m}^{n} L^i$

Primer

Izraz $a(ab)^{\{1-3\}}$ *opisuje jezik* $\{aab, aabab, aababab\}$, *a izraz* $a^? ba^+$ *opisuje jezik* $\{aba, abaa, abaaa, \ldots, ba, baa, baaa, \ldots\}$.

Napomena

Znamo od ranije da su jezici L^m, L^+ i $L^?$ regularni ako je L regularan. Slično jezici $\bigcup_{i=0}^{m} L^i$ i $\bigcup_{i=m}^{n} L^i$ su regularni kao konačne unije regularnih jezika. Najzad, jezik $\bigcup_{i=m}^{\infty} L^i$ se može predstaviti kao $L^m \cdot L^*$, pa je regularan kao proizvod dva regularna jezika. Odavde sledi da se navedenim proširenjima regularnih izraza ne proširuje klasa jezika koji se mogu predstaviti, već se samo pojednostavljuje zapis.

Regularni izrazi i operacije nad jezicima

Šta sa ostalim skupovnim operacijama?

- Ranije je konstatovano da je klasa regularnih jezika zatvorena za operacije preseka, razlike i komplementa skupa
- Ipak, ne postoje operatori kojima se mogu direktno opisati ovakvi regularni jezici (kao što je slučaj sa unijom)
- Otuda, nije tako očigledno koji bi regularni izraz odgovarao npr. preseku dva regularna jezika zadata nekim regularnim izrazima r_1 i r_2
- Ipak, znamo da takav regularni izraz postoji
- Sistematski postupak za određivanje regularnih izraza u takvim slučajevima pokazaćemo kasnije

Regularni izrazi i operacije nad jezicima

Primer

Znamo da je jezik svih binarnih niski koje sadže dve uzastopne nule predstavljen npr. izrazom $(0|1)^*00(0|1)^*$*. Komplement ovog jezika je takođe regularan, ali nije očigledno koji bi mu izraz odgovarao. Možemo primeniti ad-hoc pristup i konstruisati izraz u zavisnosti od konkretnog slučaja: ovde tražimo jezik svih reči koje ne sadrže dve uzastopne nule, a jedan izraz koji opisuje ovaj jezik bi mogao da bude:* $1^*(01^+)^*0^?$*.*

Regularni izrazi u leksičkoj analizi

Koji su jezici regularni u praksi?

- U terminima regularnih jezika mogu se opisati različiti jezici koji se sreću u praksi:
  - Celi i realni brojevi
  - Datumi
  - Različiti formati za lozinke, korisnička imena naloga i sl.
  - Razni obrasci u tekstu
  - ...
- Zbog toga su regularni izrazi veoma korisni u pretraživanju i obradi teksta:
  - Konzolni alat `grep` je primer alata za pretragu teksta zasnovan na regularnim izrazima
  - Većina programskih jezika imaju ili ugradjenu podršku za regularne izraze (npr. `Perl`) ili ih podržavaju kroz odgovarajuće biblioteke

Leksička analiza

- Svaka klasa leksema (kojoj odgovara jedan token) predstavlja jedan jezik
- Ovi jezici su po pravilu regularni i mogu se opisati regularnim izrazima
- Otuda je proučavanje regularnih jezika veoma značajno sa stanovišta konstrukcije leksičkih analizatora

Primeri

Primer

- *Jezik identifikatora u C-u se može kraće opisati regularnim izrazom* $[a-zA-Z\_][a-zA-Z\_0-9]^*$.
- *Jezik celobrojnih (dekadnih) konstanti u C-u se može opisati regularnim izrazom* $[1-9][0-9]^*[uUlL]^{\{-2\}}$
- *Jezik označenih realnih konstanti se može opisati regularnim izrazom* $([0-9]|[1-9][0-9]^*).[0-9]^+$

NAPOMENA: poslednja dva izraza predstavljaju samo aproksimacije stvarnih jezika i navedeni su kao ilustracija. Student može za vežbu da pokuša da precizno definiše izraze koji opisuju odgovarajuće leksičke kategorije u jeziku C.

Ograničenja regularnih jezika

Sledeća lema se u literaturi obično zove lema o razrastanju (engl. pumping lemma):

Lema 1

Neka je L regularan jezik. Tada postoji neko $p \in \mathbb{N}$ (koje zavisi samo od jezika L), takvo da za svaku reč $w \in L$ za koju je $|w| > p$ važi da se w može predstaviti u obliku $w = xzy$, gde je $|z| \geq 1$ i $xz^k y \in L$ za svako $k \in \mathbb{N}_0$.

Ovu lemu dokazaćemo kasnije.

Posledica

Da bismo dokazali da jezik L nije regularan, dovoljno je da pokažemo da možemo pronaći proizvoljno dugu reč $w \in L$ takvu da je nije moguće predstaviti u opisanom obliku.

Lema o razrastanju se obično koristi da se dokaže da jezik nije regularan.

Ograničenja regularnih jezika

Primer

Jezik $L = \{a^n b^n \mid n \geq 0\}$ nije regularan. Zaista, pretpostavimo suprotno, da jeste regularan – prema prethodnoj lemi postojalo bi neko p takvo da zadovoljava uslove iz leme. Mi uvek možemo uzeti dovoljno veliko $n \in \mathbb{N}_0$ takvo da je $|a^n b^n| > p$. Tada bi za ovu reč važilo da se može predstaviti u obliku $a^n b^n = xzy$, takvo da je $|z| > 0$ i reč $xz^k y \in L$ za svako $k \in \mathbb{N}_0$. Medjutim, lako se vidi da kako god da podelimo reč $a^n b^n$ na tri dela (pri čemu je srednji deo neprazan), stepenovanjem srednjeg dela dobijamo reči koje više ne pripadaju jeziku, što je kontradikcija.

VAŽNA NAPOMENA

U jeziku iz prethodnog primera zahtevali smo da reči sadrže jednak broj a-ova i b-ova (isto n je u oba stepena). To nije isto kao da smo imali jezik $L' = \{a^n b^m \mid m, n \geq 0\}$, gde broj a-ova i b-ova u reči može biti različit. Jezik L' jeste regularan i može se opisati izrazom $a^* b^*$.

Ograničenja regularnih jezika

Posledice

- Jezik $L = \{a^n b^n \mid n \geq 0\}$ je apstraktna varijanta jezika ,,uparenih zagrada"
- Ovakve konstrukcije se često javljaju u sintaksi programskih jezika:
  - Svaka otvorena zagrada (u izrazima mora da ima odgovarajuću zatvorenu zagradu)
  - Svaki početak bloka { u C-u mora da ima odgovarajući kraj bloka }
  - Svaki početak bloka *begin* u Pascal-u mora da ima odgovarajući kraj bloka *end*
  - Svaka otvorena zagrada [za indeksiranje nizova mora da ima odgovarajuću zatvorenu zagradu]
  - Svaki otvoreni tag u HTML-u mora da ima odgovarajući zatvoreni tag
  - ...
- Odavde sledi da glavne sintaksne kategorije koje postoje u svim programskim jezicima – izrazi i naredbe – ne mogu da se opišu regularnim izrazima
- Dakle, regularni izrazi će nam biti korisni za leksičku, ali ne i za sintaksnu analizu

Pregled

Kontekstno slobodna gramatika

Definicija 8

Kontekstno slobodna gramatika (KSG) je uređena četvorka oblika $G = (\Sigma, N, S, P)$, *gde je:*

- Σ – *azbuka nad kojom se gradi gramatika (skup terminala)*
- N – *konačni skup nezavršnih simbola (ili neterminala)*
- $S \in N$ – *početni neterminal (ili aksioma)*
- $P \subseteq N \times (N \cup \Sigma)^*$ – *konačni skup pravila izvodjenja*

Pravilo $(A, \alpha) \in P$ *zapisujemo kao* $A \longrightarrow \alpha$ *i čitamo ,,A izvodi* α”

Kontekstno slobodna gramatika

Primer

Neka je $G = (\Sigma, N, S, P)$, *gde je:*

- $\Sigma = \{a, b\}$
- $N = \{S\}$ *(tj.* S *je jedini neterminal)*
- $P = \{S \longrightarrow aSb, S \longrightarrow \varepsilon\}$

Gramatike ćemo često zapisivati neformalno, zadavanjem samo skupa pravila. Na primer, za gornju gramatiku, mogli smo napisati samo:

$$\begin{aligned} S &\longrightarrow aSb \\ S &\longrightarrow \varepsilon \end{aligned}$$

Po konvenciji, neterminale ćemo označavati velikim slovima, dok ćemo terminale označavati malim slovima. Otuda, nije neophodno eksplicitno navoditi skupove Σ *i* N. *Takodje, po konvenciji će početni neterminal biti onaj za koji se prvo navode pravila, te ni njega nije neophodno eksplicitno naglašavati.*
Gornju gramatiku ćemo još kraće zapisivati i ovako:

$$\begin{aligned} S &\longrightarrow aSb \\ &\mid \varepsilon \end{aligned}$$

Uopšte, pravila koja odgovaraju istom neterminalu ćemo grupisati korišćenjem ,,ili" operatora. Napominjemo da je ovo samo kraći zapis, a ne bilo kakva suštinska izmena u odnosu na prethodnu definiciju gramatike.

Kontekstno slobodni jezici

Definicija 9

- *Relacija izvodjenja indukovana gramatikom* $G = (\Sigma, N, S, P)$ *je binarna relacija* $\Longrightarrow$ *nad skupom* $(\Sigma \cup N)^*$ *takva da* $u \Longrightarrow v$ *akko je* $u = \alpha X \beta$, *a* $v = \alpha\gamma\beta$, *za neke* α, β, X *i* γ, *pri čemu je* $X \longrightarrow \gamma \in P$.
- *Izvođenje u gramatici* G *je bilo koji lanac* $\alpha_1 \Longrightarrow \alpha_2 \Longrightarrow \ldots \Longrightarrow \alpha_m$.
- *Tranzitivno zatvorenje relacije* $\Longrightarrow$ *označavamo sa* $\Longrightarrow^+$, *a tranzitivno i refleksivno zatvorenje sa* $\Longrightarrow^*$.
- *Rečenična forma gramatike* $G = (\Sigma, N, S, P)$ *je bilo koja reč* $\alpha \in (\Sigma \cup N)^*$ *takva da* $S \Longrightarrow^* \alpha$, *tj. takva da postoji izvođenje* $S \Longrightarrow \ldots \Longrightarrow \alpha$ *u gramatici* G.
- *Jezik generisan gramatikom* G *(u oznaci* $L(G)$*) je skup svih završnih rečeničnih formi gramatike* G, *tj. skup svih reči* $w \in \Sigma^*$ *takvih da* $S \Longrightarrow^* w$, *odnosno takvih da postoji izvođenje* $S \Longrightarrow \ldots \Longrightarrow w$ *u gramatici* G.
- *Za jezik* L *nad azbukom* Σ *kažemo da je kontekstno slobodan ako postoji kontekstno slobodna gramatika koja ga generiše.*

Kontekstno slobodni jezici

Primer

Vratimo se na gramatiku:

$$\begin{aligned} S &\longrightarrow aSb \\ &\mid \varepsilon \end{aligned}$$

Primer jednog izvođenja u ovoj gramatici je:

$$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow aaaSbbb \Longrightarrow aaabbb$$

Sve reči u ovom izvođenju predstavljaju rečenične forme gramatike. Reč aaabbb predstavlja završnu rečeničnu formu, jer se sastoji samo iz terminala. Ova reč pripada jeziku ove gramatike.
Jezik generisan ovom gramatikom je $L(G) = \{a^n b^n \mid n \geq 0\}$.

Regularni i kontekstno slobodni jezici

Primedba

- Gramatika iz prethodnog primera zapravo generiše jezik $L(G) = \{a^n b^n \mid n \geq 0\}$.
- Za ovaj jezik smo ranije dokazali da nije regularan.
- Ovo znači da kontekstno slobodni jezici ne moraju biti regularni.
- Specijalno, ,,uparivanje zagrada" nije problem za kontekstno slobodne jezike.
- Obratno pitanje: da li regularni jezici moraju biti kontekstno slobodni?

Regularni i kontekstno slobodni jezici

Teorema 1

Svaki regularan jezik je kontekstno slobodan.

Dokaz

Potrebno je dokazati da se svaki regularan jezik može generisati gramatikom:

- *Jezik* $\varnothing$ *se može generisati bilo kojom gramatikom sa praznim skupom pravila*
- *Jezik* $\{\varepsilon\}$ *se može generisati gramatikom* $S \longrightarrow \varepsilon$
- *Jezik* $\{a\}$ *za proizvoljno* $a \in \Sigma$ *se može generisati gramatikom* $S \longrightarrow a$
- *Ako su jezici* L_1 *i* L_2 *generisani redom gramatikama* $G_1 = (\Sigma, N_1, S_1, P_1)$ *i* $G_2 = (\Sigma, N_2, S_2, P_2)$*, tada:*
  - *Jezik* $L_1 \cup L_2$ *će biti generisan gramatikom* $G = (\Sigma, N_1 \cup N_2 \cup \{S\}, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1, S \longrightarrow S_2\})$
  - *Jezik* $L_1 \cdot L_2$ *će biti generisan gramatikom* $G = (\Sigma, N_1 \cup N_2 \cup \{S\}, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1 S_2\})$
- *Ako je jezik* L *generisan gramatikom* $G = (\Sigma, N, S, P)$*, tada će jezik* L^* *biti generisan gramatikom* $G' = (\Sigma, N \cup \{S'\}, S', P \cup \{S' \longrightarrow SS', S' \longrightarrow \varepsilon\})$

Odavde skup svih kontekstno slobodnih jezika mora sadržati sve regularne jezike.

Rekurzija u gramatikama

Definicija 10

Za pravilo gramatike kažemo da je rekurzivno ako je oblika $A \longrightarrow \alpha A \beta$ *(*$\alpha, \beta \in (\Sigma \cup N)^*$*), tj. ako se simbol sa leve strane pojavljuje i u desnoj strani pravila. Specijalno, pravilo je levo rekurzivno ako je oblika* $A \longrightarrow A\alpha$*, a desno rekurzivno ako je oblika* $A \longrightarrow \alpha A$*.*

Primer

U gramatici $S \longrightarrow aSb \mid \varepsilon$*, pravilo* $S \longrightarrow aSb$ *je rekurzivno pravilo. Ovo pravilo nije ni levo ni desno rekurzivno, već je rekurzivno po sredini.*

Primer

U gramatici $S \longrightarrow SAB$*,* $A \longrightarrow aA \mid B$*,* $B \longrightarrow ab \mid Ba$*, pravila* $S \longrightarrow SAB$ *i* $B \longrightarrow Ba$ *su levo rekurzivna, dok je pravilo* $A \longrightarrow aA$ *desno rekurzivno.*

Rekurzija u gramatikama

Primedba

- Rekurzija može biti i posredna – na primer: $S \longrightarrow aAb \mid \varepsilon$, $A \longrightarrow bSa \mid a$
- Može se pokazati da gramatika u kojoj nema rekurzije (ni posredne ni neposredne) može generisati samo konačne jezike
- Otuda, gramatike koje ne sadrže rekurziju nisu naročito interesantne

Izvođenje nalevo i nadesno

Primer

Neka je data gramatika G:

$$\begin{array}{rcl} S & \longrightarrow & (L) \\ & | & a \\ L & \longrightarrow & S\,L \\ & | & S \end{array}$$

Reč $(a\,(a\,a))$ pripada jeziku $L(G)$. Jedno izvođenje ove reči bi moglo da izgleda ovako: $S \Longrightarrow (L) \Longrightarrow (S\,L) \Longrightarrow (a\,L) \Longrightarrow (a\,S) \Longrightarrow (a\,(L)) \Longrightarrow (a\,(S\,L)) \Longrightarrow (a\,(a\,L)) \Longrightarrow (a\,(a\,S)) \Longrightarrow (a\,(a\,a))$

Ovakvo izvođenje se naziva i izvođenje nalevo (ili najlevlje izvođenje), jer se u svakom koraku pravilo primenjuje na najlevlji neterminal. Analogno, postoji i izvođenje nadesno (ili najdešnje izvođenje): $S \Longrightarrow (L) \Longrightarrow (S\,L) \Longrightarrow (S\,S) \Longrightarrow (S\,(L)) \Longrightarrow (S\,(S\,L)) \Longrightarrow (S\,(S\,S)) \Longrightarrow (S\,(S\,a)) \Longrightarrow (S\,(a\,a)) \Longrightarrow (a\,(a\,a))$

Napomene

- Za svaku reč jezika $L(G)$ postoji bar jedno najlevlje i jedno najdešnje izvođenje
- Mogu postojati izvođenja koja nisu ni najlevlja ni najdešnja
- Pitanje: da li je najlevlje (najdešnje) izvođenje proizvoljne reči jezika jedinstveno?

Stablo izvođenja

Definicija 11

Neka je dato izvođenje $S \Longrightarrow \alpha_1 \Longrightarrow \alpha_2 \Longrightarrow \ldots \Longrightarrow w$ *reči* $w \in \Sigma^*$ *jezika* $L(G)$. *Stablo izvođenja koje odgovara datom izvođenju se formira na sledeći način:*

- *U korenu stabla se nalazi početni simbol* S
- *Ako u nekom koraku imamo list stabla u kome se nalazi neterminal* A, *pri čemu je na taj neterminal* A *u datom izvođenju primenjeno pravilo* $A \longrightarrow X_1 X_2 \ldots X_k$, *tada se u stablo kao potomci ovog čvora dodaju čvorovi u kojima se nalaze simboli* $X_1, X_2, \ldots, X_k$
- *Postupak se završava kada u listovima imamo samo terminale*

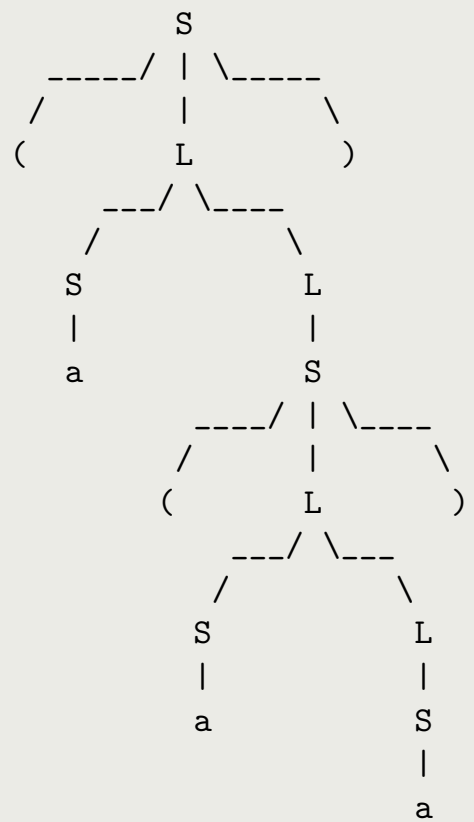
Primedba

- Obilaskom listova stabla izvođenja sa leva na desno dobija se reč w
- U unutrašnjim čvorovima stabla nalaze se neterminali koji učestvuju u izvođenju

Stablo izvođenja

Primer

Najlevljem izvođenju iz prethodnog primera odgovara sledeće stablo izvođenja:



Za vežbu: uveriti se da najdešnjem izvođenju iz prethodnog primera odgovara to isto stablo izvođenja.

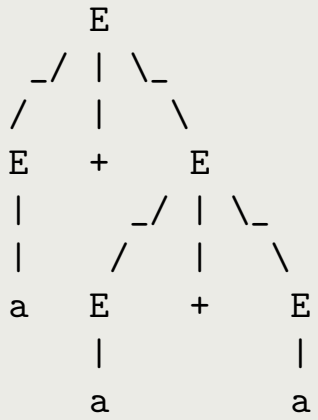
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo sledeću gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & E + E \\ & | & a \end{array}$$

kao i reč $a + a + a$. Ova reč pripada jeziku gramatike, jer npr. imamo izvođenje: $E \Longrightarrow E + E \Longrightarrow a + E \Longrightarrow a + E + E \Longrightarrow a + a + E \Longrightarrow a + a + a$. *Ovo izvođenje je najlevlje i odgovara mu sledeće stablo izvođenja:*

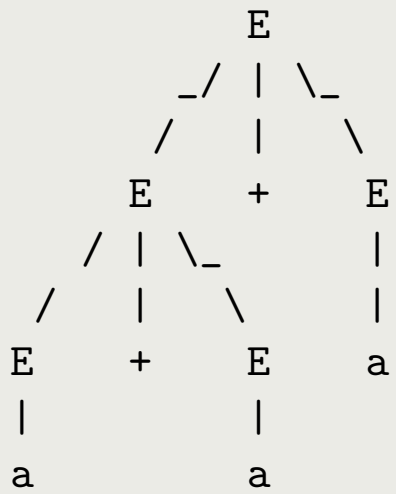


Lako se vidi da najdešnjem izvođenju: $E \Longrightarrow E + E \Longrightarrow E + \underbrace{E + E} \Longrightarrow E + E + a \Longrightarrow E + a + a \Longrightarrow a + a + a$ *odgovara isto stablo izvođenja.*

Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo ponovo gramatiku i reč iz prethodnog primera. Izvođenje: $E \Longrightarrow E + E \Longrightarrow \underbrace{E + E}_{} + E \Longrightarrow a + E + E \Longrightarrow a + a + E \Longrightarrow a + a + a$ *je takođe jedno najlevlje izvođenje reči* $a + a + a$*. Ovom izvođenju odgovara stablo:*



Kao i ranije, postoji i najdešnje izvođenje koje odgovara istom ovom stablu: $E \Longrightarrow E + E \Longrightarrow E + a \Longrightarrow E + E + a \Longrightarrow E + a + a \Longrightarrow a + a + a.$

Jednoznačne i višeznačne gramatike

Definicija 12

- *Za dva izvođenja iste reči jezika $L(G)$ kažemo da su ekvivalentna ukoliko im odgovara isto stablo izvođenja*
- *Za gramatiku G kažemo da je jednoznačna ako za svaku reč $w \in L(G)$ postoji jedinstveno stablo izvođenja (tj. ako su sva izvođenja reči w međusobno ekvivalentna)*
- *Gramatika je višeznačna ako nije jednoznačna*

Primedba

- Za svako stablo izvođenja reči w u gramatici G postoji jedinstveno najlevlje (najdešnje) izvođenje koje mu odgovara
- Otuda je gramatika jednoznačna akko za svaku reč $w \in L(G)$ postoji jedinstveno najlevlje (najdešnje) izvođenje

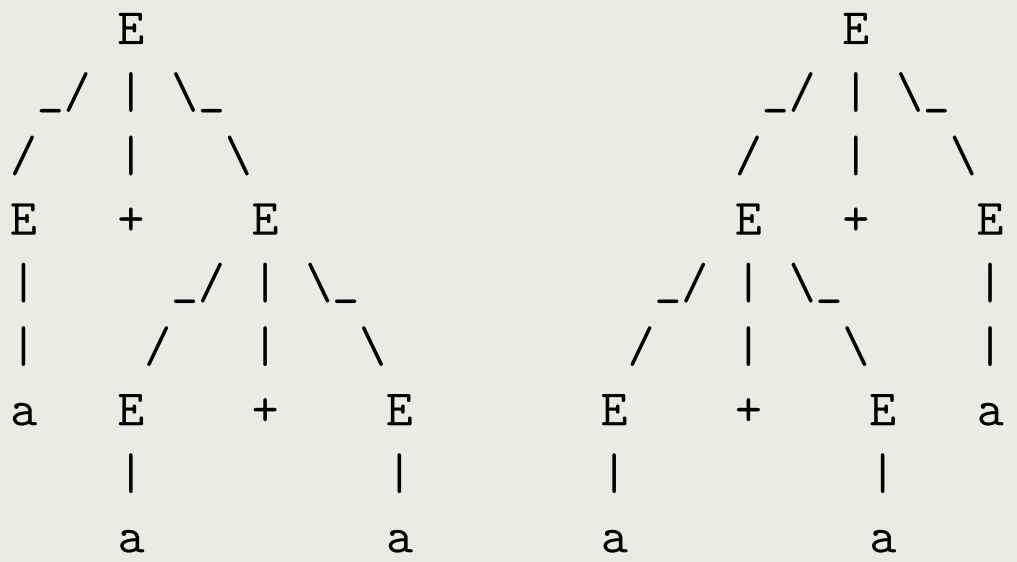
Napomene

- Višeznačnost je svojstvo gramatike, a ne jezika
- Isti jezik može biti generisan različitim gramatikama, pri čemu neke mogu biti jednoznačne, a neke višeznačne
- Jezik je inherentno višeznačan ako su sve gramatike koje ga generišu višeznačne

Jednoznačne i višeznačne gramatike

Primer

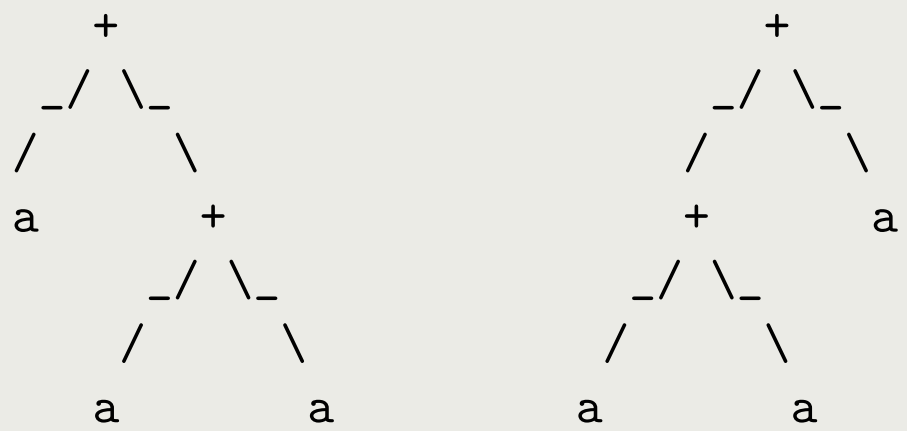
Gramatika iz prethodnog primera je višeznačna, jer smo videli da reč $a + a + a$ ima dva različita stabla izvođenja:



Jednoznačne i višeznačne gramatike

Primer

Stablima izvođenja sa prethodnog slajda odgovaraju sledeća stabla apstraktne sintakse:



U prvom slučaju imamo desnu asocijativnost operatora $+$, a u drugom slučaju levu asocijativnost. Dakle, semantika izraza $a + a + a$ će zavisiti od toga koje izvođenje izaberemo.

Jednoznačne i višeznačne gramatike

Višeznačnost: uzroci i posledice

- Višeznačnost gramatike, po pravilu, ima za posledicu višeznačnost semantike jezika
- Zbog toga je neophodno izbeći višeznačnost kada god je to moguće
- U ovom primeru, uzrok višeznačnosti je nedefinisana asocijativnost operatora $+$
- Na koji način možemo pravilima gramatike definisati asocijativnost operatora?
  - Problem je u dvostrukoj rekurziji u pravilu $E \longrightarrow E + E$
  - Levo E omogućava levu asocijativnost, jer je moguće izvesti $E \Longrightarrow \underbrace{E + E}\,+E$
  - Desno E omogućava desnu asocijativnost, jer je moguće izvesti $E \Longrightarrow E + \underbrace{E + E}$
- Osnovni princip: leva asocijativnost se postiže levom rekurzijom, a desna desnom

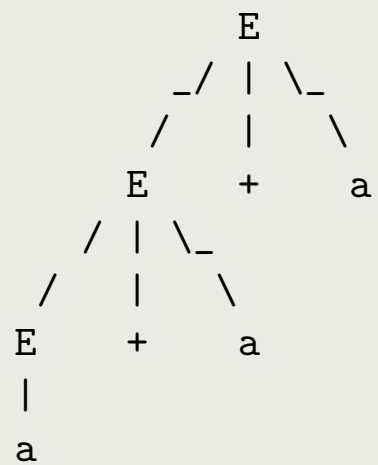
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo sada gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & E + a \\ & | & a \end{array}$$

Lako se može videti da je ova gramatika ekvivalentna prethodnoj, u smislu da generiše isti jezik. Međutim, ova gramatika je jednoznačna. Jedino najlevlje izvođenje reči $a + a + a$ u ovoj gramatici je: $E \Longrightarrow E + a \Longrightarrow E + a + a \Longrightarrow a + a + a$ (ovo je ujedno i najdešnje izvođenje). Jedinstveno stablo izvođenja ove reči je:



Ovoga puta, leva asocijativnost je garantovana.

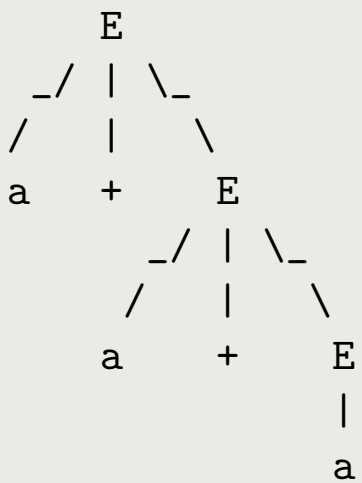
Jednoznačne i višeznačne gramatike

Primer

Ako želimo desnu asocijativnost operatora $+$, možemo da koristimo gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & a + E \\ & | & a \end{array}$$

Jedino najlevlje izvođenje reči $a + a + a$ u ovoj gramatici je:
$E \Longrightarrow a + E \Longrightarrow a + a + E \Longrightarrow a + a + a$. Jedinstveno stablo izvođenja je:



Ovoga puta imamo desnu asocijativnost.

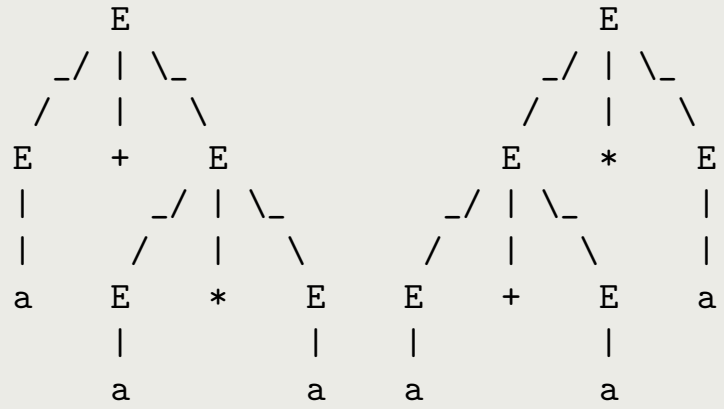
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & E + E \\ & | & E * E \\ & | & a \end{array}$$

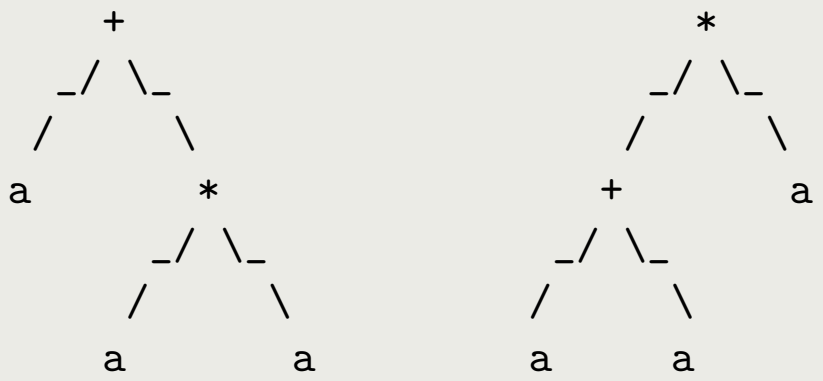
U ovoj gramatici opet imamo višeznačnost, ovog puta iz više razloga: pored nedefinisane asocijativnosti, imamo nedefinisan i prioritet operatora. Na primer, ako imamo reč jezika $a + a * a$*, možemo imati sledeće najlevlje izvođenje:*
$E \Longrightarrow E + E \Longrightarrow a + E \Longrightarrow a + E * E \Longrightarrow a + a * E \Longrightarrow a + a * a$*. Sa druge strane, možemo imati i sledeće najlevlje izvođenje:*
$E \Longrightarrow E * E \Longrightarrow \underbrace{E + E} * E \Longrightarrow a + E * E \Longrightarrow a + a * E \Longrightarrow a + a * a$*. Ovim izvođenjima odgovaraju sledeća dva stabla:*



Jednoznačne i višeznačne gramatike

Primer

Stablima izvođenja sa prethonog slajda odgovaraju sledeća apstraktna sintaksna stabla:



Dakle, u prvom slučaju se izraz izračunava tako što se prvo primeni operacija množenja, dok se u drugom slučaju prvo primenjuje operacija sabiranja. Opet imamo različita značenja istog izraza!

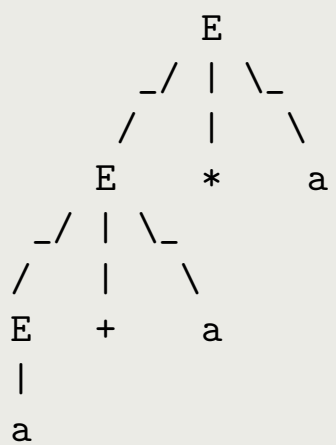
Jednoznačne i višeznačne gramatike

Primer

Problem možemo da pokušamo da rešimo na stari način, fiksiranjem leve (ili desne) asocijativnosti: posmatrajmo gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & E + a \\ & | & E * a \\ & | & a \end{array}$$

Ova gramatika je sada jednoznačna: jedino najlevlje izvođenje reči $a + a * a$ je sada $E \Longrightarrow E * a \Longrightarrow E + a * a \Longrightarrow a + a * a$. Međutim, ovo verovatno nije ono što bismo želeli:



Mi bismo želeli da množenje ima viši prioritet, ali to ovde nije tako.

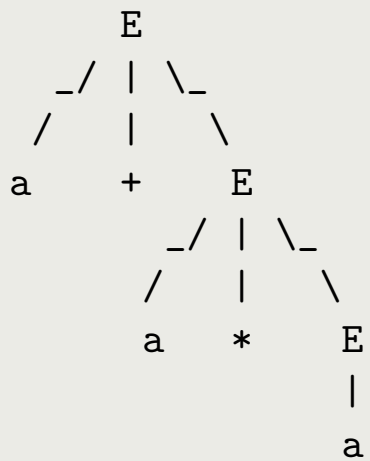
Jednoznačne i višeznačne gramatike

Primer

Prva ideja koja nam pada na pamet je da probamo desnu asocijativnost:

$$\begin{array}{rcl} E & \longrightarrow & a + E \\ & | & a * E \\ & | & a \end{array}$$

Jedino najlevlje izvođenje reči $a + a * a$ *je sada* $E \Longrightarrow a + E \Longrightarrow a + a * E \Longrightarrow a + a * a$.
Stablo izvođenja je sada:



Sada deluje u redu. Izgleda da smo uspeli. Ili nismo?

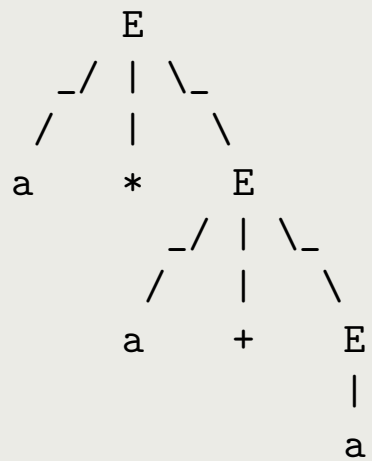
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo ponovo desno rekurzivnu gramatiku sa prethodnog slajda:

$$\begin{array}{rcl} E & \longrightarrow & a + E \\ & | & a * E \\ & | & a \end{array}$$

i reč $a * a + a$. Ovoga puta imamo najlevlje izvođenje:
$E \Longrightarrow a * E \Longrightarrow a * a + E \Longrightarrow a * a + a$, kao i stablo:



Opet imamo pogrešan prioritet. Kako sad to?

Jednoznačne i višeznačne gramatike

Uzroci višeznačnosti ponovo

- U prethodnom primeru imali smo višeznačnost usled nedefinisane asocijativnosti i prioriteta
- Fiksiranjem leve (ili desne) rekurzije rešili smo problem asocijativnosti, ali šta je sa prioritetima?
- Zapravo, u gornjim rešenjima oba operatora imala su jednak prioritet
- Redosled izvršavanja operatora u slučaju istih prioriteta određen je asocijativnošću:
  - Ako su operatori levo asocijativni, tada se izraz izračunava sa leva na desno
  - Ako su operatori desno asocijativni, tada se izraz izračunava sa desna na levo
- Otuda se kod desno asocijativne verzije gramatike u izrazu $a + a * a$ prvo izračunavalo množenje (kao što i želimo), a u izrazu $a * a + a$ se prvo izračunavalo sabiranje (što nije ono što želimo).

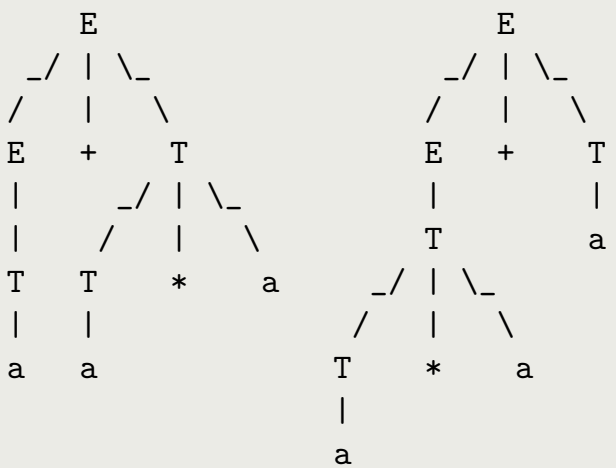
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo sledeću gramatiku:

$$\begin{aligned} E &\longrightarrow E + T \\ &\mid T \\ T &\longrightarrow T * a \\ &\mid a \end{aligned}$$

Ova gramatika je ekvivalentna sa prethodnim gramatikama. Međutim, sada za reč $a + a * a$ imamo sledeće jedinstveno najlevlje izvođenje:
$E \Longrightarrow E + T \Longrightarrow T + T \Longrightarrow a + T \Longrightarrow a + T * a \Longrightarrow a + a * a$. *Slično, za reč $a * a + a$ imamo najlevlje izvođenje:* $E \Longrightarrow E + T \Longrightarrow T + T \Longrightarrow T * a + T \Longrightarrow a * a + T \Longrightarrow a * a + a$. *Ovim izvođenjima odgovaraju sledeća stabla:*



Sada je u oba slučaja prioritet u redu.

Jednoznačne i višeznačne gramatike

Rešenje za prioritet operatora

- Prioriteti se rešavaju uvođenjem gramatičkih kategorija na više nivoa
- U prethodnom primeru, E je izraz, a T je term
  - Izraz je zbir termova
  - Term je proizvod atoma
- Operatori nižeg prioriteta se navode na višem nivou u gramatici
- Operatori višeg prioriteta se navode niže u gramatici
- Operatori na istom nivou imaju isti prioritet
- Šta ako želimo da možemo da menjamo prioritet po potrebi?
  - Tome u izrazima služe zagrade: treba ih uvesti u gramatiku!

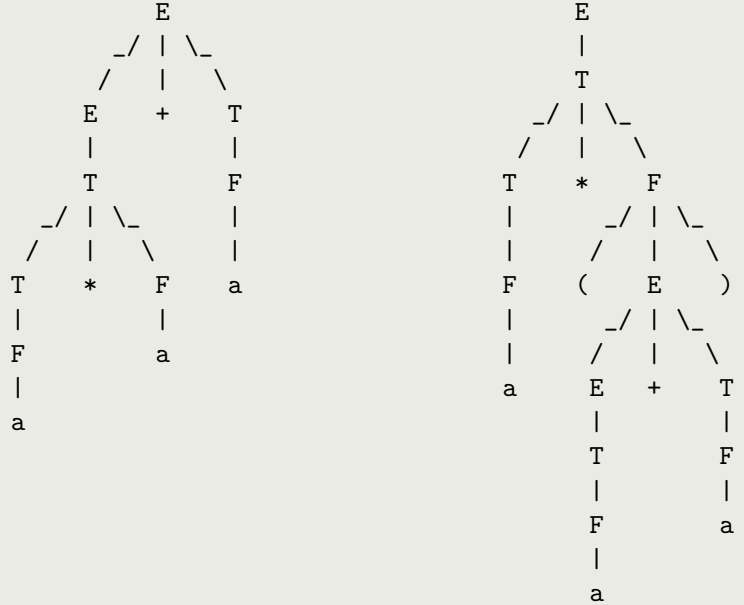
Jednoznačne i višeznačne gramatike

Primer

Posmatrajmo gramatiku:

$$\begin{aligned} E &\longrightarrow E + T \\ &\mid T \\ T &\longrightarrow T * F \\ &\mid F \\ F &\longrightarrow (E) \\ &\mid a \end{aligned}$$

Reč $a * a + a$ pripada jeziku ove gramatike. Jedinstveno najlevlje izvođenje ove reči je: $E \Longrightarrow E + T \Longrightarrow T + T \Longrightarrow T * F + T \Longrightarrow F * F + T \Longrightarrow a * F + T \Longrightarrow a * a + T \Longrightarrow a * a + F \Longrightarrow a * a + a$. *Sa druge strane, reč $a * (a + a)$, koja takođe pripada jeziku ove gramatike ima najlevlje izvođenje:* $E \Longrightarrow T \Longrightarrow T * F \Longrightarrow F * F \Longrightarrow a * F \Longrightarrow a * (E) \Longrightarrow a * (E + T) \Longrightarrow a * (T + T) \Longrightarrow a * (F + T) \Longrightarrow a * (a + T) \Longrightarrow a * (a + F) \Longrightarrow a * (a + a)$. *Stabla ova dva izvođenja su:*



Jednoznačne i višeznačne gramatike

Asocijativnost i prioritet

- Operatori istog prioriteta moraju imati istu asocijativnost
- U suprotnom, gramatika će biti višežnačna

Primer

Posmatrajmo gramatiku:

$$\begin{array}{rcl} E & \longrightarrow & a + E \\ & | & E - a \\ & | & a \end{array}$$

Reč $a + a - a$ sada ima dva najlevlja izvođenja. Jedno je:
$E \Longrightarrow a + E \Longrightarrow a + E - a \Longrightarrow a + a - a$, *a drugo je*
$E \Longrightarrow E - a \Longrightarrow a + E - a \Longrightarrow a + a - a$. *Za vežbu nacrtati odgovarajuća stabla izvođenja i uveriti se da su različita.*

Napomena

Ovo ne važi za operatore različitog prioriteta koji ne moraju da imaju istu asocijativnost.

Jednoznačne i višeznačne gramatike

Primer

Kompletna gramatika aritmetičkih izraza sa četiri osnovne računske operacije bi mogla da izgleda ovako:

$$\begin{aligned} E &\longrightarrow E + T \\ &\mid E - T \\ &\mid T \\ T &\longrightarrow T * F \\ &\mid T/F \\ &\mid F \\ F &\longrightarrow (E) \\ &\mid a \end{aligned}$$

Najniži prioritet imaju sabiranje i oduzimanje, koji su levo asocijativni. Množenje i deljenje imaju viši prioritet i takođe su levo asocijativni. Zagradama se prioritet može promeniti.

Primer

Ako želimo da u gramatiku uključimo i unarne operatore $+$ i $-$ (koji se zapisuju prefiksno), imali bismo gramatiku:

$$\begin{aligned} E &\longrightarrow E + T \\ &\mid E - T \\ &\mid T \\ T &\longrightarrow T * F \\ &\mid T/F \\ &\mid F \\ F &\longrightarrow +F \\ &\mid -F \\ &\mid A \\ A &\longrightarrow (E) \\ &\mid a \end{aligned}$$

Unarni operatori imaju najviši prioritet i zato su pri dnu gramatike.

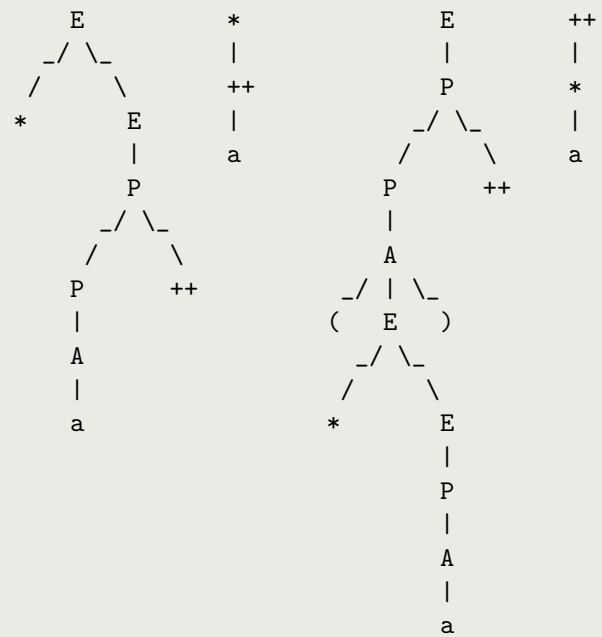
Jednoznačne i višeznačne gramatike

Primer

Ovaj primer uključuje fragment gramatike jezika C sa unarnim operatorom $*$ i postfiksnim operatorom $++$:

$$\begin{aligned} E &\longrightarrow *E \\ &\mid ++E \\ &\mid P \\ P &\longrightarrow P++ \\ &\mid A \\ A &\longrightarrow (E) \\ &\mid a \end{aligned}$$

Na primer, niske $*a++$ i $(*a)++$ bi imale, respektivno, sledeća stabla izvođenja (kao i odgovarajuća apstraktna stabla):



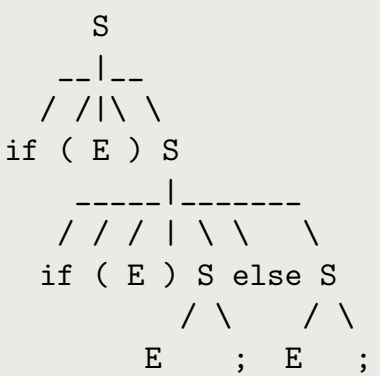
Jednoznačne i višeznačne gramatike

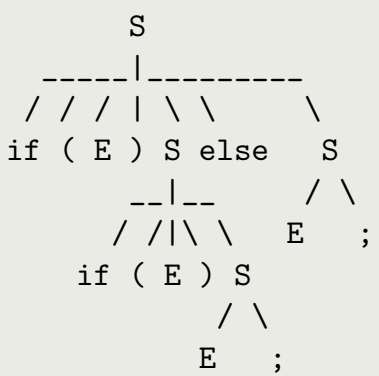
Primer

Posmatrajmo sledeći primer fragmenta gramatike jezika C koji opisuje naredbe:

$$
\begin{array}{rcl}
S & \longrightarrow & E; \\
 & | & \mathit{while}(E)\ S \\
 & | & \mathit{do}\ S\ \mathit{while}(E); \\
 & | & \mathit{for}(E; E; E)\ S \\
 & | & \ldots \\
 & | & \mathit{if}(E)\ S \\
 & | & \mathit{if}(E)\ S\ \mathit{else}\ S \\
 & | & \{L\} \\
L & \longrightarrow & L\ S \\
 & | & \varepsilon
\end{array}
$$

Ova gramatika je višeznačna. Problem je u if naredbi koja može, ali ne mora da ima else granu. Tako, na primer, naredba $\mathit{if}(E)\mathit{if}(E)E; \mathit{else}\ E;$ *može da ima dva moguća stabla izvođenja:*

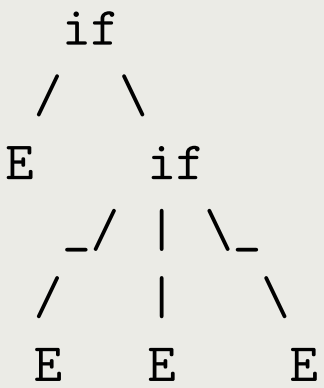


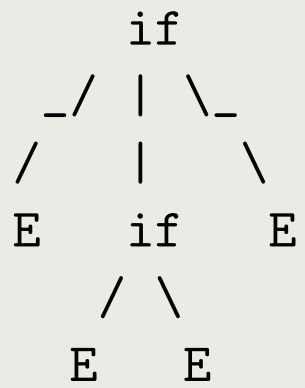


Jednoznačne i višeznačne gramatike

Primer

Stablima izvođenja sa prethodnog slajda odgovaraju sledeća stabla apstraktne sintakse:





Otuda semantika ove naredbe nije jednoznačno određena.

Jednoznačne i višeznačne gramatike

Primer

Ovaj problem se obično rešava na sledeći način:

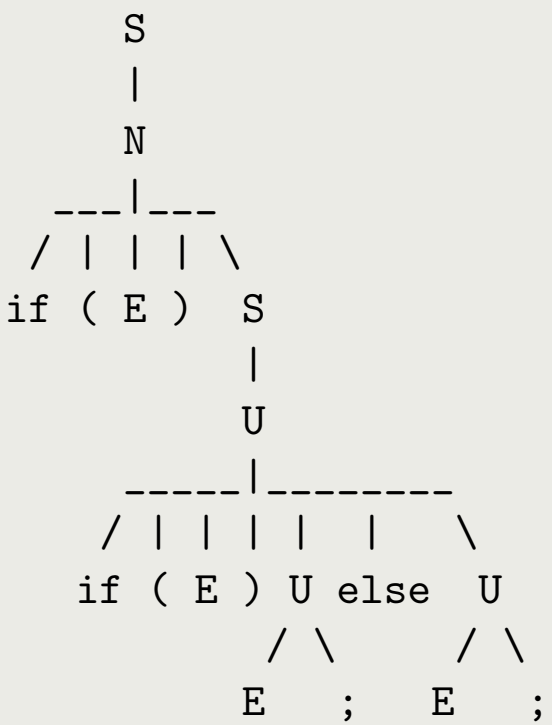
$$
\begin{array}{lcl}
S & \longrightarrow & U \\
 & | & N \\
U & \longrightarrow & E; \\
 & | & \mathit{while}(E)\ U \\
 & | & \mathit{do}\ S\ \mathit{while}(E); \\
 & | & \mathit{for}(E; E; E)\ U \\
 & | & \ldots \\
 & | & \{L\} \\
 & | & \mathit{if}(E)\ U\ \mathit{else}\ U \\
L & \longrightarrow & L\ S \\
 & | & \varepsilon \\
N & \longrightarrow & \mathit{if}(E)\ U\ \mathit{else}\ N \\
 & | & \mathit{while}(E)\ N \\
 & | & \mathit{for}(E; E; E)\ N \\
 & | & \mathit{if}(E)\ S
\end{array}
$$

Intuitivno, naredba je ili uparena naredba (U) ili neuparena naredba (N). Uparene naredbe su sve ne-if naredbe, kao i uparena if naredba (koja ima else granu koja sadrži uparenu naredbu). Neuparene naredbe nastaju upotrebom if naredbe bez else grane. Sada je uslov da naredba u okviru direktne grane if naredbe mora biti uparena, ako postoji i else grana.

Jednoznačne i višeznačne gramatike

Primer

Sada naredba $if(E)if(E)E;else\ E;$ *ima samo jedno stablo izvođenja:*



Jednoznačne i višeznačne gramatike

Primer

Sledeći primer prikazuje pojednostavljeni fragment gramatike jezika C koji opisuje deklaracije:

$$
\begin{array}{lcl}
D & \longrightarrow & T\ L; \\
L & \longrightarrow & L, K \\
 & | & K \\
T & \longrightarrow & int \\
 & | & double \\
 & | & \ldots \\
K & \longrightarrow & *K \\
 & | & P \\
P & \longrightarrow & P[c] \\
 & | & A \\
A & \longrightarrow & (K) \\
 & | & id
\end{array}
$$

gde je c token koji označava celobrojnu konstantu, a id token koji označava identifikator. Sada se deklaracija $int * a[3], (**b[3])[2];$ *može izvesti na sledeći način:*
$D \Longrightarrow T\ L; \Longrightarrow int\ L; \Longrightarrow int\ L, K; \Longrightarrow int\ K, K; \Longrightarrow int\ *K, K; \Longrightarrow int\ *P, K; \Longrightarrow int\ *P[c], K; \Longrightarrow$
$int\ *A[c], K; \Longrightarrow int\ *id[c], K; \Longrightarrow int\ *id[c], P; \Longrightarrow int\ *id[c], P[c]; \Longrightarrow int\ *id[c], A[c]; \Longrightarrow$
$int\ *id[c], (K)[c]; \Longrightarrow int\ *id[c], (*K)[c]; \Longrightarrow int\ *id[c], (**K)[c]; \Longrightarrow int\ *id[c], (**P)[c]; \Longrightarrow$
$int\ *id[c], (**P[c])[c]; \Longrightarrow int\ *id[c], (**A[c])[c]; \Longrightarrow int\ *id[c], (**id[c])[c];$ *(lekseme a, b,* 2 *i* 3 *su apstrahovane odgovarajućim tokenima).*

Jednoznačne i višeznačne gramatike

Napomena

Iako višeznačne gramatike imaju očigledne nedostatke imaju i jednu prednost: obično su znatno jednostavnije

Primer

Višeznačna gramatika aritmetičkih izraza bi mogla da izgleda ovako:

$$\begin{aligned} E &\longrightarrow E + E \\ &\mid E - E \\ &\mid E * E \\ &\mid E / E \\ &\mid (E) \\ &\mid a \end{aligned}$$

Napomena

Kasnije u toku semestra ćemo videti da pojedini alati mogu da prihvataju i ovakve gramatike, pod uslovom da se prioriteti i asocijativnost eksplicitno definišu.

Jednoznačne i višeznačne gramatike

Domaći zadatak

Proučiti gramatiku jezika C koja se nalazi u dodatku knjige „Programski jezik C” Brajana Kernigena i Denisa Ričija.

Transformacije gramatika

Zašto transformisati gramatiku?

- Videli smo da gramatika koja generiše dati kontekstno slobodni jezik nije jednoznačna
- Često je moguće gramatiku transformisati u ekvivalentnu gramatiku koja ima formu koja je za nas u nekom smislu pogodnija
  - Neke forme gramatike su pogodne u teorijskom smislu, jer se na njima mogu lakše dokazati neka svojstva
  - Druge su korisne u praksi, jer se na njih mogu primeniti neke efikasne metode prepoznavanja jezika
- Zbog toga u nastavku proučavamo neke najčešće transformacije gramatika koje se u literaturi javljaju

Eliminacija nekorisnih simbola

Definicija 13

Nezavršni simbol gramatike A je nekorisan, ako ne postoji ni jedno izvođenje oblika: $S \Longrightarrow^* \alpha A \beta \Longrightarrow^* w$, gde je $w \in \Sigma^*$.
Gramatika je čista ako ne sadrži nekorisne simbole.

Dakle, simbol je nekorisan ako ne učestvuje ni u jednom izvođenju neke reči jezika.

Neproduktivni i nedostižni simboli

Simbol A je nekorisan ako je ili neproduktivan ili nedostižan:

- Simbol A je produktivan ako $A \Longrightarrow^* w$ ($w \in \Sigma^*$). Simbol je neproduktivan ako nije produktivan.
- Simbol A je dostižan ako $S \Longrightarrow^* \alpha A \beta$. Simbol je nedostižan ako nije dostižan.

Eliminacija nekorisnih simbola

Eliminacija neproduktivnih simbola

- Najpre određujemo skup produktivnih simbola
  - Ako postoji pravilo $A \longrightarrow w$, gde je $w \in \Sigma^*$, tada se A dodaje u skup produktivnih simbola
  - Ako postoji pravilo $A \longrightarrow X_1 X_2 \ldots X_k$, tako da je svako X_i ili terminal ili je već u skupu produktivnih, tada se A dodaje u skup produktivnih simbola
  - Postupak se ponavlja do dostizanja fiksne tačke
- Kada odredimo skup produktivnih simbola, one koji nisu u tom skupu eliminišemo kao neproduktivne
  - Eliminišu se sva pravila koja sadrže neproduktivne simbole, bilo sa leve, bilo sa desne strane

Eliminacija nekorisnih simbola

Primer

Posmatrajmo gramatiku:

$$\begin{aligned} S &\longrightarrow AB \\ &\mid CA \\ A &\longrightarrow a \\ B &\longrightarrow ABD \\ &\mid EA \\ C &\longrightarrow aB \\ &\mid b \\ D &\longrightarrow aC \\ E &\longrightarrow BA \end{aligned}$$

U skup produktivnih simbola prvo dodajemo A (zbog pravila $A \longrightarrow a$), kao i C (zbog pravila $C \longrightarrow b$). Nakon toga dodajemo i simbol S (zbog pravila $S \longrightarrow CA$), kao i D (zbog pravila $D \longrightarrow aC$). Dalje proširivanje skupa produktivnih simbola nije moguće. Otuda su simboli E i B neproduktivni. Njihovom eliminacijom dobijamo gramatiku:

$$\begin{aligned} S &\longrightarrow CA \\ A &\longrightarrow a \\ C &\longrightarrow b \\ D &\longrightarrow aC \end{aligned}$$

Eliminacija nekorisnih simbola

Eliminacija nedostižnih simbola

- Najpre formiramo skup dostižnih simbola:
  - Simbol S dodajemo u skup dostižnih simbola
  - Ako je simbol A u skupu dostižnih simbola i postoji pravilo $A \longrightarrow \alpha B \beta$, tada se B dodaje u skup dostižnih simbola
  - Postupak se ponavlja do dostizanja fiksne tačke
- Kada odredimo skup dostižnih simbola, one koji nisu u tom skupu eliminišemo kao nedostižne
  - Eliminišemo sva pravila koja sadrže nedostižne simbole, bilo sa leve, bilo sa desne strane

Eliminacija nekorisnih simbola

Primer

Vratimo se na gramatiku dobijenu eliminacijom neproduktivnih simbola u prethodnom primeru:

$$\begin{aligned} S &\longrightarrow CA \\ A &\longrightarrow a \\ C &\longrightarrow b \\ D &\longrightarrow aC \end{aligned}$$

Simbol S je dostižan. Otuda su dostižni i simboli C i A. Dalje proširivanje skupa dostižnih simbola nije moguće, pa je simbol D nedostižan. Njegovom eliminacijom dobijamo gramatiku:

$$\begin{aligned} S &\longrightarrow CA \\ A &\longrightarrow a \\ C &\longrightarrow b \end{aligned}$$

Ova gramatika je čista (ne sadrži ni neproduktivne ni nedostižne simbole).

Eliminacija nekorisnih simbola

Napomena

- Postavlja se pitanje da li prvo uklanjati neproduktivne ili nedostižne simbole?
  - Eliminacijom neproduktivnih simbola mogu nastati novi nedostižni simboli
  - Eliminacijom nedostižnih simbola se ne kreiraju novi neproduktivni
- Odavde sledi da je prvo potrebno ukloniti neproduktivne, pa zatim nedostižne (i stare i novonastale).

Eliminacija nekorisnih simbola

Primer

Vratimo se ponovo na gramatiku:

$$\begin{aligned} S &\longrightarrow AB \\ &\mid CA \\ A &\longrightarrow a \\ B &\longrightarrow ABD \\ &\mid EA \\ C &\longrightarrow aB \\ &\mid b \\ D &\longrightarrow aC \\ E &\longrightarrow BA \end{aligned}$$

Da smo prvo eliminisali nedostižne simbole, ne bismo eliminisali ništa, jer su u ovoj gramatici svi simboli dostižni. Nakon toga bismo, kao i ranije, eliminacijom neproduktivnih simbola dobili gramatiku:

$$\begin{aligned} S &\longrightarrow CA \\ A &\longrightarrow a \\ C &\longrightarrow b \\ D &\longrightarrow aC \end{aligned}$$

koja nije čista, jer u njoj postoji nedostižni simbol D (koji nije bio nedostižan u početnoj gramatici, već je to postao eliminacijom neproduktivnih simbola).

Eliminacija ε-pravila

ε-slobodne gramatike

- ε-pravila su veoma česta u gramatikama
- Ipak, u nekim slučajevima nije pogodno da postoje ovakva pravila
  - Postojanje ε-pravila u gramatici omogućava skraćivanje rečeničnih formi tokom izvođenja
  - Ovo svojstvo ponekad može da smeta, kako u teorijskim, tako i u praktičnim razmatranjima
- Ipak, za metode parsiranja koja ćemo mi izučavati, ε-pravila obično neće predstavljati problem

Da li je moguće u potpunosti se osloboditi ε-pravila?

Na žalost, ε-pravila nije moguće eliminisati u potpunosti. Naime, jasno je da gramatika bez ε-pravila ne može generisati praznu reč (a ona može pripadati kontekstno slobodnim jezicima).

Eliminacija ε-pravila

Definicija 14

Za gramatiku kažemo da je ε-slobodna ako:

- *ne sadrži ni jedno ε-pravilo, ili*
- *sadrži samo jedno ε-pravilo $S \longrightarrow \varepsilon$, pri čemu je S početni simbol gramatike i on se ne pojavljuje na desnoj strani ni jednog od pravila gramatike.*

Napomena

Ovom ,,relaksiranom" definicijom smo ostavili mogućnost da generišemo praznu reč ε izvođenjem $S \Longrightarrow \varepsilon$, dok u svim ostalim izvođenjima garantovano nema primene ε-pravila, tj. nema skraćivanja rečeničnih formi tokom izvođenja.

Teorema 2

Za svaku konteksno slobodnu gramatiku G postoji ekvivalentna kontekstno slobodna gramatika G' koja je ε-slobodna.

Eliminacija ε-pravila

Napomena

Dokaz prethodne teoreme daćemo tako što ćemo konstruisati algoritam koji transformiše proizvoljnu gramatiku G u njoj ekvivalentnu ε-slobodnu gramatiku G'.

Definicija 15

Za neterminal A kažemo da je anulirajući ako $A \Longrightarrow^* \varepsilon$. Skup svih anulirajućih simbola gramatike G označavamo sa $N_\varepsilon(G)$ (ili samo N_ε, ako je jasno o kojoj gramatici je reč).

Algoritam eliminacije ε-pravila

- Formiramo skup svih anulirajućih simbola gramatike N_ε
- Svako pravilo gramatike $A \longrightarrow \alpha$ zamenimo skupom pravila koja nastaju tako što na sve moguće načine eliminišemo anulirajuće simbole iz niske α
- ε-pravila eliminišemo iz skupa pravila
- Ako je $S \in N_\varepsilon$, tada se u gramatiku dodaje novi početni simbol S', kao i pravila $S' \longrightarrow S \mid \varepsilon$

Eliminacija ε-pravila

Algoritam određivanja skupa anulirajućih simbola

- Iniicijalno, $N_\varepsilon = \varnothing$
- Ako u gramatici G postoji pravilo $A \longrightarrow \varepsilon$, tada simbol A dodajemo u skup N_ε
- Ako u gramatici G postoji pravilo $A \longrightarrow \alpha$, pri čemu svi simboli iz α pripadaju skupu N_ε, tada se i simbola A dodaje u skup N_ε
- Postupak se nastavlja do dostizanja fiksne tačke

Eliminacija ε-pravila

Primer

Neka je data gramatika:

$$\begin{aligned} S &\longrightarrow aSb \\ &\mid \varepsilon \end{aligned}$$

Ova gramatika nije ε-slobodna: jeste da ima samo jedno ε-pravilo $S \longrightarrow \varepsilon$, gde je S početni simbol, ali se ovaj simbol nalazi i na desnoj strani nekog od pravila gramatike, što nije dozvoljeno po definiciji.
Da bismo transformisali ovu gramatiku, odredimo najpre skup anulirajućih simbola N_ε. Kod nas je to skup $N_\varepsilon = \{S\}$. Sada se pravilo $S \longrightarrow aSb$ zamenjuje skupom pravila $S \longrightarrow aSb, S \longrightarrow ab$, dok se ε-pravilo izbacuje. Kako je $S \in N_\varepsilon$, dodajemo novi početni simbol S', čime dobijamo gramatiku:

$$\begin{aligned} S' &\longrightarrow S \\ &\mid \varepsilon \\ S &\longrightarrow aSb \\ &\mid ab \end{aligned}$$

Ova gramatika je ε-slobodna i ekvivalentna je polaznoj.

Eliminacija ε-pravila

Primer

Neka je data gramatika:

$$\begin{aligned} S &\longrightarrow ABC \\ A &\longrightarrow CAC \\ &\mid C \\ C &\longrightarrow aC \\ &\mid \varepsilon \\ B &\longrightarrow bC \end{aligned}$$

U skup anulirajućih simbola najpre dodajemo C (zbog pravila $C \longrightarrow \varepsilon$), a zatim A (zbog pravila $A \longrightarrow C$). Otuda je $N_\varepsilon = \{A, C\}$. Sada se npr. pravilo $S \longrightarrow ABC$ može zameniti skupom pravila $S \longrightarrow ABC \mid BC \mid AB \mid B$ (ili ne eliminišemo ništa, ili samo A, ili samo C, ili oba). Slično postupimo i sa ostalim pravilima, dok ε-pravila izbrišemo. Konačni rezultat je sledeća gramatika:

$$\begin{aligned} S &\longrightarrow ABC \\ &\mid BC \\ &\mid AB \\ &\mid B \end{aligned} \qquad \begin{aligned} A &\longrightarrow CAC \\ &\mid AC \\ &\mid CC \\ &\mid CA \\ &\mid C \end{aligned} \qquad \begin{aligned} C &\longrightarrow aC \\ &\mid a \end{aligned} \qquad \begin{aligned} B &\longrightarrow bC \\ &\mid b \end{aligned}$$

Eliminacija jednostrukih pravila

Definicija 16

Pravilo gramatike je jednostruko pravilo ako je oblika $A \longrightarrow B$, gde su A i B neterminali.

Zbog čega želimo da eliminišemo ovakva pravila?

- Postojanje jednostrukih pravila može produžavati izvođenja (npr. $\alpha A\beta \Longrightarrow \alpha B\beta \Longrightarrow \alpha C\beta \Longrightarrow \ldots$)
  - Ovo utiče samo na efikasnost i nije toliko kritično
- Ukoliko postoje ciklusi jednostrukih pravila (npr. $A \longrightarrow B, B \longrightarrow C, C \longrightarrow A$), ovo može dovesti do beskonačnih petlji tokom izvođenja ($\alpha A\beta \Longrightarrow \alpha B\beta \Longrightarrow \alpha C\beta \Longrightarrow \alpha A\beta \Longrightarrow \alpha B\beta \Longrightarrow \ldots$)
  - POSLEDICA: Gramatika koja sadrži ovakve cikluse je višeznačna!!
- Otuda je eliminacija jednostrukih pravila neophodna jedino ako postoje ciklusi

Eliminacija jednostrukih pravila

Algoritam eliminacije jednostrukih pravila

Pretpostavimo da je data gramatika ε-slobodna (ako nije, transformišemo je najpre u ekvivalentnu ε-slobodnu gramatiku).

- Za svaki neterminal A odredimo skup $J_A = \{B \in N \mid A \Longrightarrow^* B\}$
  - Ovaj skup možemo odrediti obilaskom grafa u kome su čvorovi neterminali, a grane su jednostruka pravila
  - Skup svih čvorova dostižnih iz A čine skup J_A
  - Primetimo da je uvek $A \in J_A$
- Sada za svaki neterminal A, skup njegovih pravila zamenjujemo skupom pravila koji nastaje tako što za svako $B \in J_A$ uzmemo desne strane svih ne-jednostukih pravila simbola B iz originalne gramatike

Teorema 3

Gramatika dobijena opisanim algoritmom ne sadrži jednostruka pravila i ekvivalentna je sa polaznom.

Eliminacija jednostrukih pravila

Primer

Neka je data gramatika iz prethodnog primera:

$$\begin{array}{lcl} S & \longrightarrow & ABC \\ & | & BC \\ & | & AB \\ & | & B \end{array} \qquad \begin{array}{lcl} A & \longrightarrow & CAC \\ & | & AC \\ & | & CC \\ & | & CA \\ & | & C \end{array} \qquad \begin{array}{lcl} C & \longrightarrow & aC \\ & | & a \end{array} \qquad \begin{array}{lcl} B & \longrightarrow & bC \\ & | & b \end{array}$$

Sada imamo $J_S = \{S, B\}$, $J_A = \{A, C\}$, $J_B = \{B\}$, $J_C = \{C\}$. Skup pravila za simbol S dobijamo tako što objedinimo sva ne-jednostruka S-pravila i sva ne-jednostruka B-pravila. Slično uradimo i za sve ostale simbole. Dobijamo gramatiku:

$$\begin{array}{lcl} S & \longrightarrow & ABC \\ & | & BC \\ & | & AB \\ & | & bC \\ & | & b \end{array} \qquad \begin{array}{lcl} A & \longrightarrow & CAC \\ & | & AC \\ & | & CC \\ & | & CA \\ & | & aC \\ & | & a \end{array} \qquad \begin{array}{lcl} C & \longrightarrow & aC \\ & | & a \end{array} \qquad \begin{array}{lcl} B & \longrightarrow & bC \\ & | & b \end{array}$$

Eliminacija jednostrukih pravila

Primer

Posmatrajmo gramatiku izraza:

$$\begin{array}{rcl} E & \longrightarrow & E + T \\ & | & T \\ T & \longrightarrow & T * F \\ & | & F \\ F & \longrightarrow & (E) \\ & | & a \end{array}$$

U ovoj gramatici imamo: $J_F = \{F\}$, $J_T = \{T, F\}$, $J_E = \{E, T, F\}$. *Sada eliminacijom jednostrukih pravila dobijamo gramatiku:*

$$\begin{array}{rcl} E & \longrightarrow & E + T \\ & | & T * F \\ & | & (E) \\ & | & a \\ T & \longrightarrow & T * F \\ & | & (E) \\ & | & a \\ F & \longrightarrow & (E) \\ & | & a \end{array}$$

Eliminacija jednostrukih pravila

Napomene

- Eliminacijom jednostrukih pravila skraćuju se izvođenja, ali zato gramatika postaje znatno složenija
- U praksi se ne primenjuje, osim u slučaju da postoje ciklusi

Definicija 17

Gramatika je svojstvena ako je ε-slobodna i ne sadrži cikluse (tj. izvođenja oblika $A \Longrightarrow^* A$).

Eliminacija leve rekurzije

Podsetnik

Levo rekurzivno pravilo je pravilo oblika $A \longrightarrow A\alpha$ (gde je $\alpha \in (\Sigma \cup N)^+$).

Zbog čega je eliminišemo ovakva pravila?

- Za neke metode parsiranja koje ćemo raditi, leva rekurzija nije dozvoljena
- Sa druge strane, kao što znamo, rekurziju nije moguće u potpunosti eliminisati
- Ono što možemo je da, u slučaju potrebe, levu rekurziju zamenimo desnom

Eliminacija leve rekurzije

Algoritam eliminacije leve rekurzije

Neka su data pravila za simbola A:

$$\begin{aligned} A &\longrightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \\ &\mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \end{aligned}$$

pri čemu su u prvom redu navedena levo-rekurzivna pravila, a u drugom redu pravila koja nisu levo-rekurzivna. Ovaj skup pravila zamenjujemo skupom pravila:

$$\begin{aligned} A &\longrightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_m A' \\ A' &\longrightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A' \mid \varepsilon \end{aligned}$$

gde je A' novouvedeni neterminal.

Primedba

Ovim postupkom se uvodi ε-pravilo, pa gramatika više nije ε-slobodna (ako je prethodno bila).

Eliminacija leve rekurzije

Varijanta algoritma bez uvođenja ε-pravila

Pravila za simbol A:

$$\begin{aligned} A &\longrightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \\ &\mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \end{aligned}$$

zamenjujemo skupom pravila:

$$\begin{aligned} A &\longrightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_m A' \\ &\mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \\ A' &\longrightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A' \\ &\mid \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \end{aligned}$$

Eliminacija leve rekurzije

Primer

Neka je data gramatika izraza:

$$\begin{aligned} E &\longrightarrow E + T \\ &\mid T \\ T &\longrightarrow T * F \\ &\mid F \\ F &\longrightarrow (E) \\ &\mid a \end{aligned}$$

Ovde je leva rekurzija prisutna kod simbola E i T. Eliminacijom leve rekurzije dobijamo gramatiku:

$$\begin{aligned} E &\longrightarrow TE' \\ E' &\longrightarrow +TE' \mid \varepsilon \\ T &\longrightarrow FT' \\ T' &\longrightarrow *FT' \mid \varepsilon \\ F &\longrightarrow (E) \\ &\mid a \end{aligned}$$

gde su E' i T' novouvedeni neterminali.

Eliminacija leve rekurzije

Primer

(nastavak) U slučaju da smo želeli da izbegnemo ε-pravila, dobili bismo gramatiku:

$$\begin{array}{lcl} E & \longrightarrow & TE' \mid T \\ E' & \longrightarrow & +TE' \mid +T \\ T & \longrightarrow & FT' \mid F \\ T' & \longrightarrow & *FT' \mid *F \\ F & \longrightarrow & (E) \\ & \mid & a \end{array}$$

Ovo je isto kao da smo na prethodni rezultat primenili algoritam eliminacije ε-pravila.

Eliminacija leve rekurzije

A šta ako imamo posrednu levu rekurziju?

Posredna leva rekurzija postoji u gramatici ako postoji izvođenje oblika $A \Longrightarrow^{+} A\alpha$, gde $\alpha \in (\Sigma \cup N)^{+}$.

Da li ovo smeta?

Posredna leva rekurzija je jednako nepovoljna po neke metode parsiranja kao i neposredna. Zbog toga je i nju neophodno eliminisati pre primene takvih metoda.

Kako je prepoznati?

Pretpostavimo da je gramatika ε-slobodna:

- Formiramo graf u kome su čvorovi neterminali, a grana od A do B postoji akko postoji pravilo oblika $A \longrightarrow B\alpha$ $(\alpha \in (\Sigma \cup N)^{*})$
- Ako u tom grafu postoje ciklusi (petlje) tada u gramatici postoji posredna (neposredna) leva rekurzija

Eliminacija leve rekurzije

Algoritam eliminacije posredne leve rekurzije

- Poređamo sve neterminale u niz $A_1, A_2, \ldots, A_n$:
  - Trudimo se da, kad god je to moguće, poredak bude takav da simbol X bude ispred Y kad god postoji pravilo $X \longrightarrow Y\alpha$.
- Formiramo grane od simbola A_i do simbola A_j gde god postoji pravilo oblika $A_i \longrightarrow A_j\alpha$
  - ako nema petlji ni ,,povratnih" grana (tj. grana koje idu od A_i do A_j, gde je $i \geq j$), tada u gramatici nema leve rekurzije (ni posredne ni neposredne)
- Povratne grane i petlje eliminišemo sa leva u desno:
  - za svaki simbol A_i (za $i = 1, 2, \ldots, n$, tim redom) ispitujemo da li postoji povratna grana od A_i do A_j, tj. pravilo oblika $A_i \longrightarrow A_j\alpha$ (za $j = 1, 2, \ldots, i$, tim redom).
  - Ako postoji pravilo $A_i \longrightarrow A_j\alpha$, gde je $i > j$, tada se ovo pravilo zamenjuje skupom pravila oblika $A_i \longrightarrow \alpha_j^k\alpha$, gde su $A_j \longrightarrow \alpha_j^k$ pravila za simbol A_j u datoj gramatici
  - Ako postoji pravilo $A_i \longrightarrow A_i\alpha$, tada se ovo pravilo uklanja eliminacijom neposredne leve rekurzije

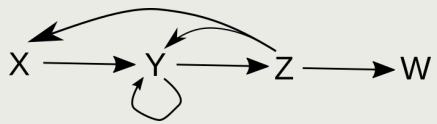
Eliminacija leve rekurzije

Primer

Neka je data gramatika:

$$\begin{aligned} X &\longrightarrow Yb \mid aZ \\ Y &\longrightarrow Ya \mid Zb \\ Z &\longrightarrow Wa \mid Xb \mid Yc \mid c \\ W &\longrightarrow aW \mid b \end{aligned}$$

Pretpostavimo da smo odabrali poredak X, Y, Z, W. Sada imamo sledeće grane u grafu:



Najpre se oslobađamo petlje na Y (neposredna leva rekurzija), gde se Y-pravila zamenjuju sledećim pravilima:

$$\begin{aligned} Y &\longrightarrow ZbY' \\ Y' &\longrightarrow aY' \mid \varepsilon \end{aligned}$$

Zatim se oslobađamo povratne grane od Z do X, tako što Z-pravila zamenjujemo sledećim pravilima:

$$Z \longrightarrow Wa \mid Ybb \mid aZb \mid Yc \mid c$$

Zatim se oslobađamo grane od Z do Y tako što Z-pravila zamenjujemo sledećim pravilima:

$$Z \longrightarrow Wa \mid ZbY'bb \mid aZb \mid ZbY'c \mid c$$

Sada nam se pojavila petlja na Z koje se oslobađamo eliminacijom neposredne leve rekurzije:

$$\begin{aligned} Z &\longrightarrow WaZ' \mid aZbZ' \mid cZ' \\ Z' &\longrightarrow bY'bbZ' \mid bY'cZ' \mid \varepsilon \end{aligned}$$

Sada više nema povratnih grana ni petlji u gornjem grafu, te je dobijena gramatika bez leve rekurzije.

Eliminacija leve rekurzije

Primedbe

- U prethodnom postupku se moglo dogoditi da se eliminacijom jedne povratne grane dobije nova povratna grana (ili petlja)
- Zbog toga je bitno da se grane uklanjaju sa leva na desno, kako bi se novostvorene grane ,,pokupile" u daljem postupku
- Simboli koji se uvode prilikom eliminacije neposredne leve rekurzije nikada nisu na početku desne strane nekog pravila (pod pretpostavkom da je polazna gramatika bila ε-slobodna)
- Otuda se oni uvek mogu staviti na početak niza (neće biti uvedene nove povratne grane)

Eliminacija leve faktorisanosti

Definicija 18

Gramatika je levo faktorisana ako postoje pravila oblika $A \longrightarrow \alpha\gamma_1 \mid \alpha\gamma_2 \mid \ldots \mid \alpha\gamma_k$, *gde je* $\alpha \in (\Sigma \cup N)^+$, *a* $\gamma_1, \ldots, \gamma_k \in (\Sigma \cup N)^*$.

Napomena

Gramatika je, dakle, levo faktorisana, ako postoje dva ili više pravila sa istom levom stranom čije desne strane imaju zajednički neprazan prefiks (levi faktor).

Zbog čega nam to smeta?

Kao i kod leve rekurzije, postojanje leve faktorisanosti onemogućava primenu nekih metoda parsiranja, pa ju je u tim slučajevima potrebno ukloniti.

Eliminacija leve faktorisanosti

Algoritam uklanjanja leve faktorisanosti

Pravila $A \longrightarrow \alpha\gamma_1 \mid \alpha\gamma_2 \mid \ldots \mid \alpha\gamma_k$ zamenjujemo pravilima:

$$\begin{aligned} A &\longrightarrow \alpha A' \\ A' &\longrightarrow \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k \end{aligned}$$

pri čemu pretpostavljamo da je α najduži mogući zajednički prefiks za navedena pravila.

Eliminacija leve faktorisanosti

Primer

Neka je data gramatika:

$$\begin{aligned} S &\longrightarrow aBS \mid acB \mid aBB \mid a \\ B &\longrightarrow aBc \mid acB \mid b \end{aligned}$$

Gramatika je levo faktorisana i za S-pravila i za B-pravila. U slučaju B pravila situacija je jednostavna – prva dva pravila imaju zajednički prefiks a, pa uvodimo novi simbol B' i pravila:

$$\begin{aligned} B &\longrightarrow aB' \mid b \\ B' &\longrightarrow Bc \mid cB \end{aligned}$$

pri čemu pravilo $B \longrightarrow b$ ostaje, jer ono nije bilo predmet eliminacije leve faktorisanosti. U slučaju S pravila, situacija je složenija. Najpre imamo zajednički prefiks a za sva četiri pravila. Eliminacijom ovog zajedničkog prefiksa dobijamo pravila:

$$\begin{aligned} S &\longrightarrow aS' \\ S' &\longrightarrow BS \mid cB \mid BB \mid \varepsilon \end{aligned}$$

Sada prvo i treće S' pravilo imaju zajednički prefiks B, pa se na njih dalje primenjuje isti postupak (dok drugo i četvrto pravilo ostaju):

$$\begin{aligned} S &\longrightarrow aS' \\ S' &\longrightarrow BS'' \mid cB \mid \varepsilon \\ S'' &\longrightarrow S \mid B \end{aligned}$$