4 Граматики и Push-down автомати

Во овој дел ќе се запознаеме со друг метод за претставување на јазици над дадена азбука Σ, т.е. ќе воведеме поим за граматика и јазик генериран од граматика. Потоа ќе се запознаеме со различни видови граматики и со поделба на јазиците во зависност од видот на граматиката од која е генериран. Посебно ќе се задржиме на така наречените контекстно слободни граматики или КС-граматики и регуларните граматики. Ќе дефинираме push down автомати, ќе покажеме дека регуларните граматики ги генерираат регуларните јазици и ќе дадеме врска помеѓу push down автоматите и контекстно слободните граматики.

4.1 Јазици генерирани од граматики

4.1.1 Граматики

Поимот "формална граматика" најопшто може да се интерпретира како алгоритам кој овозможува да се определи јазик над дадена азбука. Притоа, возможно е:

- Да означува можност за даден збор од јазикот да постои таков начин на работа на алгоритмот што на крајот од работата на алгоритмот да се добие ("генерира") збор од зададениот јазик. Ако некој збор не е од јазикот, тогаш со оваа постапка тој не може да биде генериран.

- Да се даде алгоритам, кој на крајот од својата работа ќе даде одговор на прашањето дали даден збор е од јазикот или не.

  или

- Алгоритмот да го "преброи" јазикот, т.е. да се даде таков алгоритам, што на крајот се добива подредена низа зборови во која може да се најде секој збор од зададениот јазик, но не и збор што не припаѓа на јазикот.

Во овој дел ние ќе го избереме првиот начин на определување формални граматики, така наречени генеративни граматики. Имено, *формална граматика* или само *граматика* G е подредена четворка $G = (V, \Sigma, S, R)$, каде што

- $V \cap \Sigma = \emptyset$ и V и Σ се конечни множества;

- $S \in V$ е почетен нетерминал;

- R е конечно множество зборови од облик $v \to w$, каде што $v, w \in (V \cup \Sigma)^*$, $\to$ е симбол што не е во $V \cup \Sigma$.

За Σ велиме дека е *основна азбука* или азбука од терминални симболи, а за V дека е *помошна азбука* на граматиката G или азбука од нетерминални симболи, S е *почетен симбол* од G, R е множество правила за генерирање зборови и се вика *шема* или *програма* на граматиката G. Секој член од R се вика *правило*, v се вика *лева страна*, а w *десна страна* на правилото $v \to w$. Множеството $V \cup \Sigma$ се вика *потполна азбука* на граматиката G.

Нека $r = v \to w$ е правило и нека $\alpha \star v \star \beta$ е настапување на v во зборот $s = \alpha v \beta$ од азбуката $V \cup \Sigma$. Во тој случај за зборот $t = \alpha \star w \star \beta$ велиме дека е *добиен од s со примена на правилото r*.

Ако t се добива од s со примена на едно правило од G, пишуваме $s \Rightarrow_G t$ или само $s \Rightarrow t$.

За низата зборови $s_0, s_1, \cdots s_n$ велиме дека е *извод на s_n од s_0* во граматиката G ако за секој $1 \leq i \leq n$ важи $s_{i-1} \Rightarrow s_i$. Бројот n се вика *должина* на изводот на s_n од s_0 во G, и притоа пишуваме $s_0 \Rightarrow_G{}^* s_n$ или само $s_0 \Rightarrow^* s_n$. За изводот $s_0, s_1, \cdots, s_n$ велиме дека е *потполн* ако $s_0 = S$ е почетниот симбол на G, а $s_n \in \Sigma^*$.

Јазикот $L(G)$ генериран од граматиката G се состои од сите зборови од Σ^* кои имаат потполн извод во G, т.е

$L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$.

Пример 1.

Нека е зададена граматиката

$$G = (\{S, B, C, D\}, \{a, b, c\}, S,$$

$$\{S \to BCS, BCB \to D, BC \to bc, DC \to a, cS \to c, aS \to a\}).$$

Пример на потполн извод во G е низата:

$S, BCS, BCBCS, BCBCBCS, BCDCS, BCDCBCS, BCDCbcS, BCabcS$, $bcabcS, bcabc$.

Значи, $bcabc \in L(G)$.

За граматиката G велиме дека е *контекстно осетлива* или *контекстно зависна* ако секое нејзино правило е од облик $\alpha A \beta \to \alpha w \beta$, каде што $\alpha, \beta \in (V \cup \Sigma)^*, A \in V, w \in (V \cup \Sigma)^*$.

Во суштина овие правила се такви што замената на симбол A од V зависи од "контекстот" во кој се наоѓа симболот A, па оттука и името на овој тип граматики.

Ако во правилото $r = \alpha A \beta \to \alpha w \beta$, $\alpha, \beta = \lambda$, тогаш велиме дека тоа е *безконтекстно* правило. За граматиката G велиме дека е *контекстно слободна* ако сите нејзини правила се безконтекстни. Контекстно слободните граматики пократко ќе ги викаме KS-граматики.

За KS-граматиката G велиме дека е *регуларна* или R-граматика ако секое нејзино правило е од облик $A \to aB, A \to a$ каде што $A, B \in V$ а $a \in \Sigma$.

Забелешка: Во литературата се сретнува и следната дефиниција на регуларна граматика: G е регуларна граматика ако секое нејзино правило е од облик $A \to wB$, $A \to w$, каде што $A, B \in V$ а $w \in \Sigma^*$. Се покажува дека овие дефиниции се еквивалентни.

Во зависност од типот на граматиката со која е генериран некој јазик, тие се делат на *контекстно зависни јазици, контекстно слободни јазици и R–јазици*, притоа еден јазик е контекстно осетлив (контекстно слободен, R–јазик) ако постои контекстно зависна (KS, R) граматика што го генерира јазикот, содветно.

Да забележиме дека постои врска помеѓу регуларните јазици и јазиците генерирани од R-граматики. Имено, ќе покажеме дека класта јазици генерирани од R-граматики е точно класата регуларни јазици. Затоа не е во противречност називот "регуларен јазик" за јазик генериран од R-граматика, па натаму и R–јазиците ќе ги викаме регуларни јазици.

Примери:

2 Секој непразен конечен јазик

$$L = \{s_1, s_2, \cdots, s_k\},$$

каде што $s_i \in \Sigma^*$, е генериран од R-граматиката

$G = (\{S\}, \Sigma, S, \{S \to s_1, \cdots, S \to s_k\})$, каде што $s_i \in \Sigma^*$.

3 Празниот јазик е генериран од R-граматиката

$$G = (\{S\}, \Sigma, S, \{S \to aS\}),$$

каде што $a \in \Sigma$.

4 Нека Σ е произволна азбука. Тогаш јазикот $L = \Sigma^+$ е генериран од R-граматиката

$$G = (\{S\}, \Sigma, S, \{S \to aS, S \to a | a \in \Sigma\}).$$

5 Секоја аритметичка прогресија во N, каде што природните броеви се во унарен запис, е генерирана од R-граматика. Имено, ако $k \in N^+, l = 0, 1, \cdots, k-1$, тогаш $\{kn + l | n \in N^+\}$ е аритметичка прогресија. Граматиката

$$G = (\{A_0 = S, A_1, \cdots, A_{k-1}, B_1, \cdots B_{l-1}\}, \{|\}, S, \{A_0 \to |A_1, \cdots$$

$$\cdots, A_{k-2} \to |A_{k-1}, A_{k-1} \to |A_0, A_0 \to |B_1, \cdots, B_{l-2} \to |B_{l-1}, B_{l-1} \to |\}).$$

Ако $l = 1$, тогаш последните l-правила се заменуваат со $A_0 \to |$, а ако $l = 0$, со $A_{k-1} \to |$.

6 Јазикот a^+b^+ е генериран од R-граматика

$$G = (\{S, B\}, \{a, b\}, S, \{S \to aS, S \to aB, B \to bB, B \to b\}).$$

7 Јазикот $\{a^n b^n | n \in N^+\}$ е генериран од KS-граматика

$$G = (\{S\}, \{a, b\}, S, \{S \to aSb, S \to ab\}).$$

8 Јазикот

(а) $\{a^n b^n a^m | m, n \in N^+\}$ е генериран од KS- граматиката

$$G = (\{S, A\}, \{a, b\}, S, \{S \to Sa, S \to Aa, A \to aAb, A \to ab\}).$$

(б) $\{a^m b^n a^n | m, n \in N^+\}$ е генериран од KS- граматиката

$$G = (\{S, A\}, \{a, b\}, S, \{S \to aS, S \to aA, A \to bAa, A \to ba\}).$$

9 Множеството од сите можни правилно поставеи загради), (е генериран од KS-граматиката

$$G = (\{S\}, \{), (\}, S, \{S \to SS, S \to (S), S \to (\ )\}).$$

10 Јазикот L во азбуката $\Sigma = \{a, b\}$, што ги содржи сите непразни зборови од Σ кои содржат еднаков број на појавувања на буквите a и b е генериран од KS-граматиката

$$G = (\{S\}, \{a, b\}, S, \{S \to SS, S \to aSb, S \to bSa, S \to ab, S \to ba\}).$$

Јасно е дека секој збор генериран од граматиката G е од бараниот облик. Со индукција по должина на зборовите од L ќе го докажеме обратното тврдење.

Нека $|w| = 2$. Тогаш $w = ab$ или $w = ba$, па w е генериран од граматиката G.

Нека тврдењето е точно за секој збор t од L таков што $|t| = 2k$ и $|t| < n$ и нека $|w| = 2s = n$. Тогаш, $w \in \{a\alpha b, b\beta a, \gamma\delta\}$, каде што $\alpha, \beta, \gamma, \delta \in L$ и $|\alpha|, |\beta|, |\gamma|, |\delta| < 2s$, па според индуктивната претпоставка постои извод на $\alpha, \beta, \gamma, \delta$ во G, при што S се појавува во секој од овие зборови. Со примена на уште едно правило (соодветно) се добива зборот w.

11 Јазикот $L = \{xx^R | x \in \Sigma^+\}$, каде што x^R е инверзијата на x во Σ е генериран од KS-граматиката

$$G = (\{S\}, \Sigma, S, \{S \to aSa, S \to aa | a \in \Sigma\}).$$

12 Јазикот $L = \{a^n b^n a^n | n \in N\}$ е генериран од граматиката

$$G = (\{S, B\}, \{a, b\}, S, \{S \to aSBa, S \to aba, aB \to Ba, bB \to bb\}).$$

Забелешка: Оваа граматика не е KS–граматика бидејќи $bB \to bb$ не е контекстно слободно правило, но не е ни контекстно осетлива, бидејќи правилото $aB \to Ba$ не е контекстно осетливо.

Во секој извод во G само на едно место може да се примени правилото $S \to aba$. Тогаш, се добива збор од облик $a^n bw$, каде што w содржи n настапувања на a и $n-1$ на B. За да се добие потполн извод потребно е сите симболи B да се најдат заедно по симболот b, што се постигнува со последователна примена на правилото $aB \to Ba$. Дури потота може да се примени правилото $bB \to bb$, и тоа додека не се заменат сите симболи B со терминалниот симбол b.

Теорема 4.1.1 *Класата јазици генерирани од граматики се совпаѓа со класата рекурзивно пребројви јазици (т.е. јазици чија карактеристична функција е рекурзивна).* □

Забелешка: Од својствата дадени во втората глава, како и од горнава теорема, непосредно следува дека класата препознаена од Тјурингови машини, како и класата јазици препознаена со нормални алгоритми, се совпаѓа со класата јазици генерирани од граматики.

4.1.2 Регуларни и контекстно слободни јазици

Во овој дел ќе го оправдаме терминот "регуларна граматика", т.е. ќе покажеме дека класата јазици генерирани од R-граматики е точно класата регуларни јазици (т.е. јазици препознаени од конечни автомати).

Да забележиме дека класата јазици генерирани од R-граматики е поткласа од класата KS-јазици. Земајќи го предвид својството што следува, како и примерот 7, ќе следува дека класата R-јазици е вистинска поткласа од класта KS-јазици.

Теорема 4.1.2 *Еден јазик е регуларен ако и само ако е генериран од R-граматика.*

Доказ: Нека L е регуларен јазик и нека $M = (K, \Sigma, \delta, s, F)$ е детерминистички конечен автомат што го препознава јазикот L, т.е. $L(M) = L$. Дефинираме регуларна граматика $G = (V, \Sigma, S, R)$, каде што

- $V = K$,
- $S = s$,
- $R = \{q \to ap | \delta(q, a) = p\} \cup \{q \to \lambda | q \in F\}$.

Значи правилата на граматиката ја имитираат работата на машината.

Треба да докажеме дека $L(M) = L(G)$.

Нека $w \in L(M)$. Тогаш

$$(s, w) \models_M (p, \lambda),$$

за некој $p \in F$, па

$$S \Rightarrow^* wp \Rightarrow w,$$

т.е. $w \in L(G)$.

Ако, пак, $w \in L(G)$, тогаш $S \Rightarrow^* w$, т.е. $s \Rightarrow^* w$ и, притоа, последното употребено правило мора да биде од облик $p \to \lambda$, за некој $p \in F$. Но, тогаш $(s, w) \models (p, \lambda)$ и $w \in L(M)$.

Обратно, нека $G = (V, \Sigma, S, R)$ е регуларна граматика. Ќе дефинираме недетерминистички конечен автомат M, таков што $L(M) = L(G)$. Нека $M = (K, \Sigma, \Delta, s, F)$, каде што:

- $K = V \cup \{f\}$, при што f е нов симбол,
- $s = S$,
- $F = \{f\}$,
- $\Delta = \{(A, w, B) | A \to wB \in R, A, B \in V, w \in \Sigma^*\}$
  $\cup \{(A, w, f) | A \to w \in R, A \in V, w \in \Sigma^*\}$.

Повторно, машината е конструирана така што го имитира изводот на зборовите генерирани од граматиката.

Како и погоре, и во овој случај лесно се проверува точноста на равенството $L(M) = L(G)$ $\square$.

Значи класата јазици генерирани од R-граматики е точно класата регуларни јазици. Бидејќи јазикот од примерот 7 е генериран од KS-граматика и не е регуларен (што покажавме порано со користење на теоремата за пумпање за регуларни јазици), следува дека класата R-јазици е вистинска поткласа од класата KS-јазици.

4.2 Push-down автомати

Видовме порано дека класата регуларни јазици се совпаѓа со класата јазици препознаени од конечни автомати, како и со класата R-јазици, но дека е вистинска поткласа од класата KS-јазици. Сакаме да конструираме специјални автомати кои ќе ги препознаваат точно KS-јазиците и ќе бидат обопштување на конечните автомати. Ваквите автомати се викаат *push-down автомати*. Овие автомати во општ случај ќе бидат недетерминистички и, за разлика од случајот кај конечни автомати, ќе постојат KS-јазици кои нема да може да бидат препознаени од "детерминистички" push-down автомати.

4.2.1 Push-down автомати

Во овој оддел ќе дадеме обопштување на поимот конечен автомат. Идејата е да се воведе начин на сместување на симболи во "склад" (stack) и со помош на контрола да се обезбеди следење на редоследот на складираните симболи. Притоа, последниот симбол сместен во складот е најгоре и може прв да се извади од складот.

Прецизно, *push-down автомат* е подредена шесторка $M = (K, \Sigma, \Gamma, \Delta, s, F)$, каде што:

- K е конечно множество состојби,
- Σ е азбука од влезни симболи,
- Γ е множество симболи од складот (stack symbols),
- $s \in K$ е почетна состојба,
- $F \subseteq K$ е множество завршни состојби,
- Δ е релација на премин, т.е. конечно подмножество од

$$(K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*).$$

Интуитивно гледано, ако $((p, u, \beta), (q, \gamma)) \in \Delta$, тогаш секогаш кога M е во состојба p, го гледа зборот u на лентата и β е на врвот од складот, машината го чита зборот u, преминува во состојба q, оди едно место на десно и во складот го заменува β со γ.

Парот $((p, u, \beta), (q, \gamma))$ се вика *премин на* M.

Вака дефинираните автомати овозможуваат симбол да се додаде или да се извади од складот. Имено, преминот $((p, u, \lambda), (q, a))$ додава симбол a, додека преминот $((p, u, a), (q, \lambda))$ го вади симболот a од складот.

Како и кај конечните автомати, push-down автоматите можат само да препознаваат збор од даден јазик и почетниот, веќе прочитан, подзбор не влијае на натамошната работа на автоматот. Во согласност со ова *конфигурација на push-down автомат* е елемент од $K \times \Sigma^* \times \Gamma^*$; притоа, првиот елемент е состојбата во која се наоѓа автоматот во дадениот момент, вториот е зборот кој допрва треба да го прочита, а третиот е содржината на складот.

На пример, (q, w, abc) е конфигурација на M таква што M е во состојба q, го гледа најлевиот симбол од зборот w, a е на врвот, а c на дното од складот.

Дефинираме релација $\vdash_M$ во $K \times \Sigma^* \times \Gamma^*$ на следниов начин:

$$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha),$$

за секој $x \in \Sigma^*, \alpha \in \Gamma^*$, $((p, u, \beta), (q, \gamma)) \in \Delta$.

Рефлексивното и транзитивно затворање на $\vdash_M$ го означуваме со $\models_M$.

Велиме дека M го препознава зборот $w \in \Sigma^*$ ако и само ако $(s, w, \lambda) \models (p, \lambda, \lambda)$, каде што $p \in F$.

Значи јазикот $L(M)$ препознаен од автоматот M е дефиниран со:

$$L(M) = \{w \in \Sigma^* | (s, w, \lambda) \models (p, \lambda, \lambda), p \in F\}.$$

Примери:

1 Да конструираме push-down автомат којшто го препознава јазикот $L = \{wcw^R | w \in \{a, b\}^*\}$. Автоматот $M = (K, \Sigma, \Gamma, \Delta, s, F)$, каде што

- $K = \{s, f\}$,
- $\Sigma = \{a, b, c\}$,
- $\Gamma = \{a, b\}$,
- $F = \{f\}$,
- $\Delta = \{((s, a, \lambda), (s, a)), ((s, b, \lambda), (s, b)), ((s, c, \lambda), (f, \lambda)),$
  $((f, a, a), (f, \lambda)), ((f, b, b), (f, \lambda))\}$.

Како работи овој автомат?

Ако влезниот збор на лентата е wcw^R, тогаш се додека не стигне до симболот c, автоматот е во почетната состојба и ги сместува по редослед на читање симболите a и b во складот. Кога ќе стигне до симболот c, преминува во завршната состојба f и ниту внесува ниту изнесува симбол од складот. Натаму, го чита секој нареден симбол, го споредува со најгорниот симбол во складот и ако тие се совпаѓаат го вади соодветниот симбол од складот и постапката продолжува со читање на наредниот симбол од лентата. Бидејќи влезниот симбол е од бараниот облик, на крај ќе го испразни складот и ќе остане во завршната состојба f. Ако зборот не е од дадениот облик, автоматот или никогаш нема да стигне до завршната состојба f или, пак, нема да го испразни складот.

На пример, ако влезниот збор е $abbcbba$, тогаш: $(s, abbcbba, \lambda) \vdash (s, bbcbba, a) \vdash (s, bcbba, ba) \vdash (s, cbba, bba) \vdash$
$\vdash (f, bba, bba) \vdash (f, ba, ba) \vdash (f, a, a) \vdash (f, \lambda, \lambda)$.

2 Да конструираме push-down автомат којшто ќе го препознава јазикот $L = \{ww^R | w \in \{a, b\}^*\}$. Бараниот автомат не се разликува многу од оној во примерот 1. Имено

$$M = (K, \Sigma, \Gamma, \Delta, s, F),$$

каде што:

- $K = \{s, f\}$,
- $\Sigma = \{a, b\}$,
- $\Gamma = \{a, b\}$,
- $F = \{f\}$,
- $\Delta = \{((s, a, \lambda), (s, a)), ((s, b, \lambda), (s, b)), ((s, \lambda, \lambda), (f, \lambda)),$
  $((f, a, a), (f, \lambda)), ((f, b, b), (f, \lambda))\}$.

Во овој случај преминот $((s, \lambda, \lambda), (f, \lambda))$ може да се примени во секој момент, па затоа овој автомат е недетерминистички. Битно е дека ако влезниот збор е од бараниот облик постои низа премини која од почетната состојба s и празен склад, по читање на зборот го доведува автоматот до завршната состојба и празен склад. Меѓутоа, постојат низи премини и при збор од бараниот облик кои нема да доведат до завршна состојба или до завршна состојба и празен склад. Значи, за автоматот да препознае некој збор од дадената азбука потребно е да постои низа премини кои од почетната состојба и празен склад, по читање на зборот ќе ја доведат машината до завршна состојба и празен склад.

4.2.2 Push-down автомати и KS-граматики

Во овој оддел ќе ја покажеме врската помеѓу KS-јазици и push-down автомати. Пред да го искажеме и докажеме основното својство ќе воведеме некои помошни поими и леми.

Нека $G = (V, \Sigma, S, R)$ е KS-граматика. Тогаш, како што дефиниравме порано $w \in L(G)$ ако и само ако $w \in \Sigma^*$ и постои извод

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_{n-1} \Rightarrow w,$$

за некои $w_i \in (V \cup \Sigma)^*, i = 1, 2, \cdots, n-1, n > 0$.

За еден извод во G велиме дека е *лев извод* ако во секој чекор замена се изведува на првиот од лево нетерминален симбол во зборот.

На пример, ако $G = (\{S\}, \{), (\}, S, \{S \to \lambda, S \to SS, S \to (S)\})$, тогаш

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (\ )S \Rightarrow (\ )(S) \Rightarrow (\ )(\ )$$

е лев извод на $(\ )(\ )$, додека

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (\ )(S) \Rightarrow (\ )(\ )$$

не е лев извод во G на истиот збор $(\ )(\ )$.

Формално ќе пишуваме $v \Rightarrow^L w$, каде $v, w \in (V \cup \Sigma)^*$ акко во овој чекор е применето правило од G на најлевиот нетерминал, а $w_0 \Rightarrow^{*L} w_n$ ако е даден лев извод на w_n од w_0 во G.

Лема 4.2.1 *За секоја KS-граматика G и збор $w \in \Sigma^*$, $S \Rightarrow^* w$ акко $S \Rightarrow^{*L} w$.*

Доказ: Во едната насока, т.е. акко постои лев извод на w во G, тогаш постои и извод на w во G тврдењето е очигледно точно. Затоа да ја докажеме втората насока.

Нека

$$w_0 \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n$$

е извод во G на w_n од w_0, при што $w_n \in \Sigma^*$. Ако овој извод е лев извод, тогаш тврдењето е точно. Затоа, нека k е најмалиот природен број таков што преминот $w_k \Rightarrow w_{k+1}$ не е извршен на најлевиот нетерминал во w_k. Ќе дефинираме нов извод во G на w_n од w_0:

$$w'_0 \Rightarrow w'_1 \Rightarrow \cdots \Rightarrow w'_n$$

таков што $w'_0 = w_0$ и $w'_n = w_n$, во кој ако преминот $w'_{k'} \Rightarrow w'_{k'+1}$ не е лев премин, тогаш $k' > k$. Од тука тврдењето ќе следува со индукција.

Нека $w_k \Rightarrow w_{k+1}$ не е лев премин. Тогаш w_k е од облик $\alpha A\beta B\gamma$, каде што $\alpha \in \Sigma^*, \beta, \gamma \in (V \cup \Sigma)^*, A, B \in V$ и $w_{k+1} = \alpha A\beta\delta\gamma$, каде $B \to \delta$ е правилото применето во преминот. Бидејќи $w_n \in \Sigma^*, k+1 < n$, тогаш постои правило применето по k-тиот чекор со кое појавувањето на A е заменето. Нека l е најмалиот природен број $l > k$ такво што $w_l = \alpha A\varepsilon$ за некој $\varepsilon \in (V \cup \Sigma)^*$ и $w_{l+1} = \alpha\zeta\varepsilon$, каде што е применето правилото $A \to \zeta$. Тогаш $\beta B\gamma \Rightarrow^* \varepsilon$ во $l-k$ чекори и можеме да ги замениме примените на правилата $B \to \delta$ и $A \to \zeta$ на следниов начин:

- $w_0 \Rightarrow^{*L} w_k = \alpha A\beta B\gamma$ во k чекори,
- $\alpha A\beta B\gamma \Rightarrow^L \alpha\zeta\beta B\gamma$ во 1 чекор,
- $\alpha\zeta\beta B\gamma \Rightarrow^* \alpha\zeta\varepsilon$ во $l-k$ чекори,
- $\alpha\zeta\varepsilon \Rightarrow^* w_n$ во $n-l-1$ чекор.

Новиот извод е повторно со должина n и започнува со најмалку $k+1$ леви премини. □

Оваа лема овозможува полесно да се спроведат доказите на останатите својства, така што наместо со општ извод на збор во KS-граматика G, ќе работиме само со леви изводи, при што нема да се изгуби од општоста на тврдењето.

Исто така, полезно е да се комбинираат два премини во push down автомат M во еден, поради што потребна ни е наредната лема.

Лема 4.2.2 *Ако M е push-down автомат и $(q_1, w_1, \alpha_1) \models (q_2, \lambda, \alpha_2)$ и $(q_2, w_2, \alpha_2\alpha_3) \models (q_3, \lambda, \alpha_4)$, тогаш $(q_1, w_1w_2, \alpha_1\alpha_3) \models (q_3, \lambda, \alpha_4)$.*

Специјално, ако $(q_1, w_1, \alpha_1) \models (q_2, \lambda, \lambda)$ и $(q_2, w_2, \alpha_3) \models (q_3, \lambda, \lambda)$, тогаш $(q_1, w_1w_2, \alpha_1\alpha_3) \models (q_3, \lambda, \lambda)$. □

Доказот на ова својство е едноставен и е оставен како задача.

Наредното својство е основната теорема од овој дел која ја дава врската помеѓу KS-јазиците и push-down автоматите.

Теорема 4.2.1 *Класата јазици препознаена од push-down автомати е точно класта контекстно слободни јазици.*

Доказот на оваа теорема ќе го поделиме на два дела преку наредните две леми.

Лема 4.2.3 *Секој контекстно слободен јазик е препознаен од некој push-down автомат.*

Доказ: Нека $G = (V, \Sigma, S, R)$ е контекстно слободна граматика. Ќе конструираме push-down автомат M, таков што $L(M) = L(G)$. Автоматот што ќе го конструираме ќе има само две состојби p и q и ќе останува перманенто во состојбата q после првиот чекор. Имено

$$M = (\{p, q\}, \Sigma, V \cup \Sigma, \Delta, p, \{q\}),$$

каде што Δ ги содржи следниве релации:

1 $((p, \lambda, \lambda), (q, S))$,

2 $((q, \lambda, A), (q, x))$, за секое правило $A \to x$ во R,

3 $((q, a, a), (q, \lambda))$, за секој $a \in \Sigma$.

Овој автомат започнува со сместување на S во складот којшто е празен и преминува во состојбата q. Во секој нареден чекор, или го заменува најгорниот симбол A од складот со десната страна на правилото $A \to x$ од R или го вади најгорниот симбол од складот доколку тој е терминален симбол и е еднаков со следниот влезен симбол. Автоматот е конструиран така што при својата работа имитира лев извод на граматиката G.

За да докажеме дека $L(M) = L(G)$, треба да ги докажеме следниве две тврдења:

Тврдење 1. *Ако* $S \Rightarrow^{*L} \alpha_1\alpha_2$, *каде што* $\alpha_1 \in \Sigma^*$ *и* $\alpha_2 \in V(V \cup \Sigma)^* \cup \{\lambda\}$, *тогаш* $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$.

Тврдење 2. *Ако* $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$, *каде што* $\alpha_1 \in \Sigma^*$ *и* $\alpha_2 \in (V \cup \Sigma)^*$, *тогаш* $S \Rightarrow^{*L} \alpha_1\alpha_2$.

Лемата е директна последица од овие две тврдења во случај $\alpha_2 = \lambda$.

Доказ на Тврдењето 1. Нека $S \Rightarrow^{*L} \alpha$ каде што $\alpha = \alpha_1\alpha_2, \alpha_1 \in \Sigma^*$ и $\alpha_2 \in V(V \cup \Sigma)^* \cup \{\lambda\}$. Ќе го докажеме тврдењето 1 со индукција по должината на изводот на α од S.

Ако изводот е со должина 0, Тогаш $S = \alpha$, па $\alpha_1 = \lambda$ и $\alpha_2 = S$; но тогаш јасно е дека $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$.

Да претпоставиме дека тврдењето е точно за секој лев извод со должина помала или еднаква на n, каде што $n \geq 0$.

Нека

$$S = u_0 \Rightarrow^L u_1 \Rightarrow^L \cdots \Rightarrow^L u_{n+1} = \alpha$$

е лев извод на α од S во G со должина $n+1$, и нека α_1 и α_2 се како што е наведено погоре. Тогаш е јасно дека u_n има барем еден нетерминал и притоа $u_n = \beta_1 A \beta_2$, а $u_{n+1} = \beta_1 \gamma \beta_2$, каде што $\beta_1 \in \Sigma^*, A \in V$ и $A \to \gamma$. Според индуктивната претпоставка

$$(q, \beta_1, S) \models (q, \lambda, A\beta_2), \tag{10}$$

но поради $A \to \gamma$, $((q, \lambda, A), (q, \gamma))$ е премин во M од видот 2, па

$$(q, \lambda, A\beta_2) \models (q, \lambda, \gamma\beta_2). \tag{11}$$

Но, од $\alpha = \beta_1\gamma\beta_2$, каде што $\beta_1 \in \Sigma^*$ и, исто така, $\alpha = \alpha_1\alpha_2$, следува дека α_2 е или λ или започнува со нетерминал. Значи $|\alpha_1| \geq |\beta_1|$ и $|\alpha_2| \leq |\gamma\beta_2|$, и можеме α_1 да го запишеме како $\beta_1\delta$ за некој $\delta \in \Sigma^*$ таков што $\delta\alpha_2 = \gamma\beta_2$. Значи,

$$(q, \delta, \gamma\beta_2) \models (q, \lambda, \alpha_2) \tag{12}$$

според преминот од тип 3. Од (10), (11) и (12), според Лема 4.2.2 добиваме

$$\begin{aligned}(q, \alpha_1, S) &= (q, \beta_1\delta, S) \\ &\models (q, \delta A\beta_2) \\ &\vdash (q, \delta, \gamma\beta_2) \\ &\models (q, \lambda, \alpha_2)\end{aligned}$$

и со тоа е завршен доказот на чекорот на индукција.

Доказ на тврдењето 2. Да претпоставиме, сега, дека $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$, при што $\alpha_1 \in \Sigma^*$ и $\alpha_2 \in (V \cup \Sigma)^*$. Ќе покажеме дека $S \Rightarrow^{*L} \alpha_1\alpha_2$. Повторно доказот ќе го спроведеме со индукција, но овој пат по бројот на чекори на автоматот M.

Ако $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$ во нула чекори, тогаш е јасно дека $(q, \alpha_1, S) = (q, \lambda, \alpha_2)$, па $\alpha_1 = \lambda$, а $\alpha_2 = S$, што значи дека $S \Rightarrow^{*L} \alpha_1\alpha_2$.

Да претпоставиме дека ако $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$ е добиено со n или помалку од n чекори, $n \geq 0$, тогаш $S \Rightarrow^{*L} \alpha_1\alpha_2$.

Да го докажеме тврдењето ако $(q, \alpha_1, S) \models (q, \lambda, \alpha_2)$ е добиено со $n+1$ чекори. Имено, во овој случај за некој $\beta \in \Sigma^*, \gamma \in \Gamma^*$, $(q, \alpha_1, S) \models (q, \beta, \gamma)$ е добиено со n чекори и $(q, \beta, \gamma) \vdash (q, \lambda, \alpha_2)$. Притоа, овој последен чекор е резултат на премин од типот 2 или од типот 3.

Ако последниот премин бил од типот 2, тогаш $\beta = \lambda, \gamma = A\gamma_1$ и $\alpha_2 = \delta\gamma_1$, за некој $A \in V, \gamma_1 \in (V \cup \Sigma)^*$ и некое правило од облик $A \to \delta$.

Бидејќи од индуктивната претпоставка имаме $S \Rightarrow^{*L} \alpha_1 A\gamma_1$, добиваме $S \Rightarrow^{*L} \alpha_1\delta\gamma_1 = \alpha_1\alpha_2$.

Ако, пак, последниот премин бил од типот 3, тогаш $\beta = a$ е терминален симбол, а $\gamma = a\alpha_2$. Тогаш $\alpha_1 = \delta a$, за некој $\delta \in \Sigma^*$, и $(q, \delta, S) \models (q, \lambda, a\alpha_2)$ во n чекори, па според индуктивната претпоставка $S \Rightarrow^{*L} \delta a\alpha_2 = \alpha_1\alpha_2$.

Со ова е докажано и тврдењето 2., што значи и лемата 4.2.3. Останува да се докаже обратната насока на Теоремата. □

Лема 4.2.4 *Ако еден јазик е препознаен од push-down автомат, тогаш тој е контекстно слободен јазик.*

Доказ: Погодно е да се задржиме на поедноставен вид push-down автомат. Имено, за push-down автоматот $M = (K, \Sigma, \Gamma, \Delta, s, F)$ велиме дека е *едноставен* ако се исполнети следниве два услова:

(1) ако $((q, u, \beta), (p, \gamma))$ е премин во Δ, тогаш $|\beta| \leq 1$;

(2) ако $((q, u, \lambda), (p, \gamma)) \in \Delta$, тогаш и $((q, u, A), (p, \gamma A)) \in \Delta$, за секој $A \in \Gamma$.

Со други зборови (1) значи дека ако машината го проверува складот, тогаш го проверува и евентуално го извлекува само најгорниот симбол; додека (2) значи дека ако автоматот може да се придвижи без проверка на складот, тој тоа може да го направи и со истовремено извлекување и внесување на истиот најгорен симбол во него. Тврдиме дека не се губи од општоста на доказот со ова ограничување на разгледуваниот автомат.

Нека $M = (K, \Sigma, \Gamma, \Delta, s, F)$ е произволен push-down автомат. Ќе конструираме едноставен push-down автомат кој ќе го препознава истиот јазик $L(M)$.

Прво ги отстрануваме од Δ сите премини $((q, u, \beta), (p, \gamma))$ кај кои $|\beta| > 1$. Ова се обезбедува со модификација на M да извлекува само поединечни симболи од β, наместо да ги отстрани сите во еден чекор. Попрецизно, ако $\beta = B_1B_2 \cdots B_n$, каде што $n > 1$ и $B_1, B_2, \cdots, B_n \in \Gamma$, тогаш на K се додаваат нови состојби $t_1, t_2, \cdots, t_{n-1}$, а единствениот премин $((q, u, \beta), (p, \gamma))$ од Δ се заменува со преминитe:

$$((q, \lambda, B_1), (t_1, \lambda))$$

$$((t_1, \lambda, B_2), (t_2, \lambda))$$

$$\vdots$$

$$((t_{n-1}, u, B_n), (p, \gamma))$$

Слична модификација на преминитe во Δ се прави доколку тие го нарушуваат условот (1) за едноставност на автоматот. За да се

обезбеди условот (2), на Δ само и ги додаваме премините $((q,u,A),(p,\gamma A))$ за секој $A \in \Gamma$, секогаш кога $((q,u,\lambda),(p,\gamma))$ е премин во Δ. Јасно е дека добиениот автомат е едноставен и го препознава истиот јазик како и M.

Останува да покажеме дека ако M е произволен едноставен push-down автомат, тогаш постои KS-граматика G што го генерира јазикот $L(M)$.

Конструираме контекстно слободна граматика $G = (V, \Sigma, S, R)$, каде што V содржи симбол S и нови симболи $\langle q, A, p\rangle$, за секои $q, p \in K$ и секој $A \in \Gamma \cup \{\lambda\}$. За да се разбере улогата на новите нетерминали, да се потсетиме дека G треба да ги генерира точно оние зборови што ги препознава M.

Правилата од R се од следниве четири типа:

1. За секој $f \in F$, правило од облик $S \to \langle s, \lambda, f\rangle$.

2. За секој премин $((q,u,A),(r,B_1B_2\cdots B_n)) \in \Delta$, каде што $q, r \in K, u \in \Sigma^*, n > 0, B_1, B_2, \cdots, B_n \in \Gamma$, и $A \in \Gamma \cup \{\lambda\}$, и за сите $p, q_1, \cdots, q_{n-1} \in K$, правило од облик

$$\langle q, A, p\rangle \to u\langle r, B_1, q_1\rangle\langle q_1, B_2, q_2\rangle \cdots \langle q_{n-1}, B_n, p\rangle.$$

3. За секој премин $((q,u,A),(r,\lambda)) \in \Delta$, каде што $A \in \Gamma \cup \{\lambda\}$, и за сите $p \in K$, правило од облик

$$\langle q, A, p\rangle \to u\langle r, \lambda, p\rangle.$$

4. За секој $q \in K$, правило од облик

$$\langle q, \lambda, q\rangle \to \lambda.$$

Да забележиме дека од фактот што M е едноставна, само едно од правилата од облик 2. или 3. се применуваат за секој премин во M.

Во суштина, правилото од тип 1. има за цел да премине од почетната состојба во завршна состојба, притоа оставајќи го складот во иста состојба на крајот, во која бил и на почетокот. Правилото од тип 4. вели дека е потребен премин од една состојба во себе. Правило од тип 2. опишува долг премин со ефект на премин од состојбата q во состојбата p, при што од складот го извлекува нетерминалот A (или евентуално λ. Десната страна на правило од типот 2. претставува $n+1 \geq 2$ пократки премини на M, од кои првиот е единствено преминување во состојбата r додека го чита влезниот симбол u, а останатите n се меѓу премини кои имаат за цел да ги извлечат $B_1, \cdots, B_n$ од складот. Правилото од тип 3. е по аналогија на правилото од тип 2. за $n = 0$,

т.е симбол (или λ) е одстранет од складот и во него не е внесен нов симбол.

За да го докажеме тврдењето од лемата, всушност треба да го докажеме следното тврдење:

Тврдење. *За секои* $q, p \in K, A \in \Gamma \cup \{\lambda\}$ *и* $x \in \Sigma^*$, *точно е*

$$\langle q, A, p\rangle \Rightarrow^* x \text{ акко } (q, x, A) \models (p, \lambda, \lambda).$$

Во едната насока доказот се спроведува со индукција по должината на изводот во G, а во другата со индукција по должина на бројот на премини во автоматот M. Доказот ќе му го оставиме на читателот како задача. □

4.3 Својства на контекстно слободните јазици

Во овој последен дел ќе разгледаме неколку својства на контекстно слободните јазици и ќе покажеме дека постои јазик генериран од граматика што не е контекстно слободен.

4.3.1 Својства на затвореност на KS-јазици во однос на некои операции со множества

Теорема 4.3.1 *Контекстно слободните јазици се затворени во однос на унија, конкатенација и Клиниева ѕвезда.*

Доказ: Нека $G_1 = (V_1, \Sigma_1, S_1, R_1)$ и $G_2 = (V_2, \Sigma_2, S_2, R_2)$ се контекстно слободни граматики и нека $V_1 \cap V_2 = \emptyset$.

Унија. Нека S е нов симбол и нека $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\})$. Тогаш $L(G_1) \cup L(G_2) = L(G)$. Имено, единствени правила кои го вклучуваат новиот почетен симбол S се $S \to S_1$ и $S \to S_2$, па $S \Rightarrow^* w$, каде што $w \in (\Sigma_1 \cup \Sigma_2)$ ако и само ако $S_1 \Rightarrow^* w$ или $S_2 \Rightarrow^* w$. Приота, $S_1 \Rightarrow^*_G w$ ако и само ако $S_1 \Rightarrow^*_{G_1} w$, а $S_2 \Rightarrow^*_G w$ ако и само ако $S_2 \Rightarrow^*_{G_2} w$.

Конкатенација. Конструкцијата на граматиката G таква што $L(G) = L(G_1)L(G_2)$ е слична како и во случајот за унија. Имено:

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R_1 \cup R_2 \cup \{S \to S_1 S_2\}).$$

Клиниева ѕвезда. $L(G_1)^*$ е генериран од граматиката

$$G = (V_1, \Sigma_1, S_1, R_1 \cup \{S_1 \to \lambda, S_1 \to S_1 S_1\}). \square$$

Како што ќе видиме подолу, пресек од контекстно слободни јазици не мора да биде контекстно слободен јазик. Меѓутоа точна е следнава теорема.

Теорема 4.3.2 *Пресек од контекстно слободен и регуларен јазик е контекстно слободен јазик.*

Доказ: Во овој случај поедноставен е доказ со помош на конечни автомати. Нека L е контекстно слободен, а R е регуларен јазик. Тогаш, $L = L(M_1)$ за некој push-down автомат $M_1 = (K_1, \Sigma_1, \Gamma_1, \Delta_1, s_1, F_1)$, а $R = L(M_2)$, за некој детерминистички конечен автомат $M_2 = (K_2, \Sigma_2, \delta_2, s_2, F_2)$. Идејата е да се комбинираат овие два автомата во еден push-down автомат M чии премини зависат и од премините во M_1 и од оние во M_2 и ги препознава само оние зборови кои ги препознаваат и двата автомата. Имено, нека $M = (K, \Sigma, \Gamma, \Delta, s, F)$, каде што

- $K = K_1 \times K_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $\Gamma = \Gamma_1$;
- $s = (s_1, s_2)$;
- $F = F_1 \times F_2$, и
- Δ, релацијата на премин, е дефинирана со

$$(((q_1, q_2), u, \beta), ((p_1, p_2), \gamma)) \in \Delta$$

акко

$$((q_1, u, \beta), (p_1, \gamma)) \in \Delta_1$$

и

$$(q_2, u) \models_{M_2} (p_2, \lambda).$$

Автоматот M преминува од состојба (q_1, q_2) во состојба (p_1, p_2) на ист начин на кој M_1 преминува од состојба q_1 во состојба p_1, но притоа води контрола врз промените во состојбите на M_2 создадени од истиот влезен збор. Ако се има ова предвид, лесно се покажува дека M е push-down автомат со бараното својство. □

4.3.2 Својства на периодичност

Бесконечните контекстно слободни јазици имаат својство на периодичност што се разликува од она кај регуларните јазици. За да го испитаме ова својство, ќе воведеме графички приказ на претставување на извод кај контекстно слободните јазици, кое ќе биде полезно и во натамошни испитувања на овој вид јазици.

Нека G е контекстно слободна граматика. Еден збор $w \in L(G)$ може да има повеќе изводи во G. На пример, ако $G = (\{S\}, \{a\}, S, \{S \rightarrow$

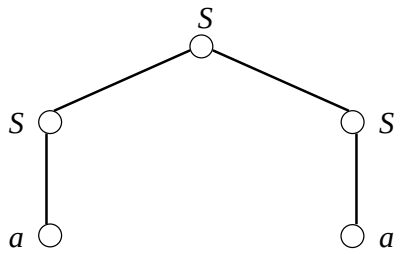
$SS, S \to a\}$), тогаш зборот aa може да биде генериран од S на следниве два различни начини:

$$S \Rightarrow SS \Rightarrow aS \Rightarrow aa$$

и

$$S \Rightarrow SS \Rightarrow Sa \Rightarrow aa.$$

Но, овие два извода се во суштина еднакви бидејќи се употребени истите правила на истите места во средниот дел од изводот. Единствената разлика е во тоа на кое појавување на нетерминалниот симбол S е применето соодветното правило. И двата извода можат да се претстават како на сликата 4.1

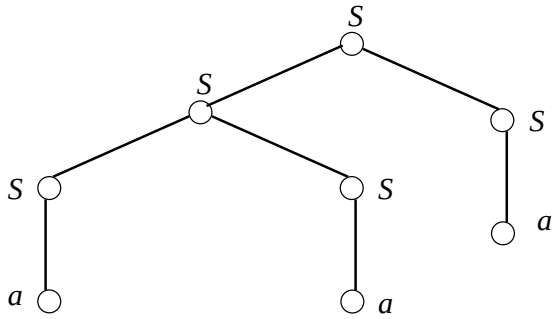


Слика 4.1:

Слично и изводот:

$$S \Rightarrow SS \Rightarrow Sa \Rightarrow SSa \Rightarrow aSa \Rightarrow aaa$$

може да се претстави како на сликата 4.2:



Слика 4.2:

Ваквото графичко претставување го викаме *парсирачко дрво* . Точките се викаат *темиња* , најгорното теме *корен*, додека најдолните темиња *листови* на парсирачкото дрво. Со конкатенација на ознаките на листовите од лево на десно, се добива генерираниот збор, кој се вика *резултат* од парсирачкото дрво.

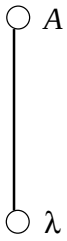
Попрецизно, за произволна контекстно слободна граматика $G = (V, \Sigma, S, R)$, парсирачкото дрво, неговиот корен, листови и производ ги дефинираме на следниов начин:

1.

Слика 4.3:

е парсирачко дрво за секој $A \in V \cup \Sigma$. Единственото теме на ова парсирачко дрво е неговиот корен и неговиот лист. Резултат е зборот A.

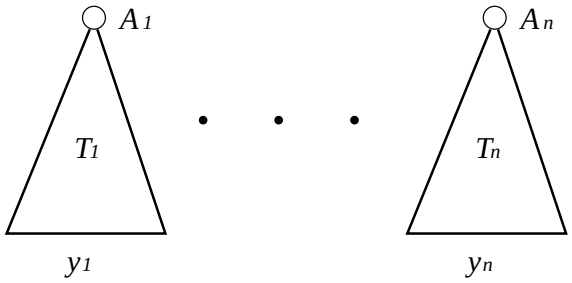
2. Ако $A \to \lambda$ е правило од R, тогаш



Слика 4.4:

е парсирачко дрво со корен означен со A, листот е означен со λ, а резултат е празниот збор λ.

3. Ако

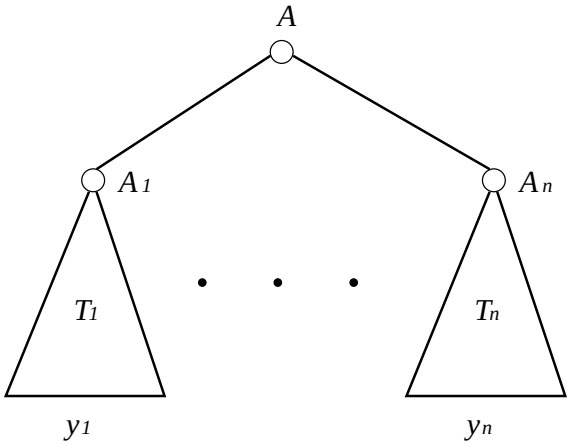


Слика 4.5:

се парсирачки дрва ($n \geq 1$) со корени означени со $A_1, \cdots, A_n$ и со резултати $y_1, \cdots, y_n$, соодветно, а притоа правило во G е

$$A \to A_1 A_2 \cdots A_n,$$

тогаш



Слика 4.6:

е парсирачко дрво со нов корен означен со A, листови се листовите на $T_1, \cdots, T_n$, а резултат е $y_1 \cdots y_n$.

4. Парсирачко дрво може да се добие единствено со помош на 1, 2 и 3.

Пат во парсирачко дрво е низа од различни темиња, секое поврзано со претходно со отсечка. Притоа првото теме е коренот, а последното лист на парсирачкото дрво. *Должина на патот* е бројот на отсечки, којшто е за еден помалку од бројот на темиња. *Висина* на парсирачко дрво е должината на најдолгиот пат во парсирачкото дрво.

Теорема 4.3.3 (*Теорема за пумпање) Нека G е контекстно слободна граматика. Тогаш постои број K што зависи од G, таков што секој збор w од $L(G)$ со должина поголема од K може да се запише во облик $w = uvxyz$ на таков начин што или v или y се непразни зборови и за секој $n \geq 0$, uv^nxy^nz е истотака во $L(G)$.*

Доказ: Нека $G = (V, \Sigma, S, R)$. За да ја докажеме теоремата доволно е да покажеме дека постои број K таков што секој терминален збор од $L(G)$ со должина поголема од K има извод од облик

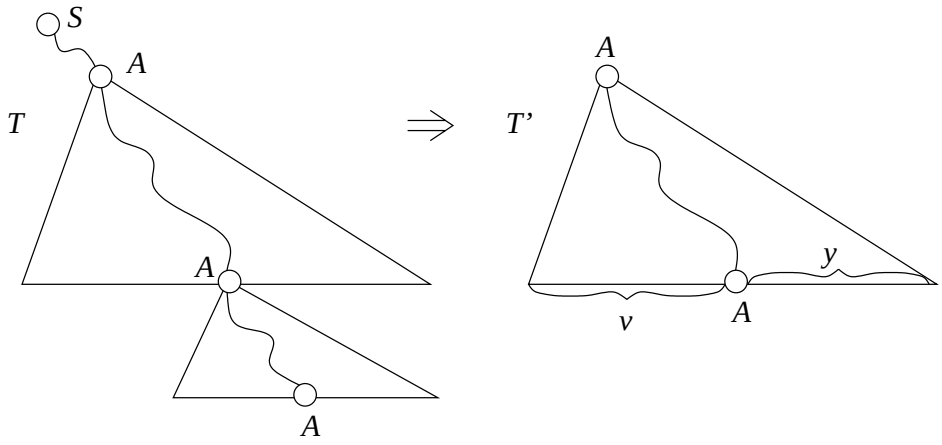
$$S \Rightarrow^* uAz \Rightarrow^* uvAyz \Rightarrow^* uvxyz,$$

каде што $u, v, x, y, z \in \Sigma^*$, $A \in V$ и или v или y се непразни. Тогаш изводот $A \Rightarrow^* vAy$ може да се повтори произволен број пати и да се добијат зборови uv^nxy^nz за различни n.

Нека p е најголемиот број симболи од десната страна на правило од R, т.е.

$$p = max\{|\alpha| : A \to \alpha \text{ е правило од } G\}.$$

За секој $m \geq 1$, парсирачко дрво со висина m може да има најмногу p^m листови, што лесно може да се покаже со индукција по m. Според тоа, парсирачко дрво со висина m може да има резултат со должина најмногу p^m. Со други зборови, ако T е парсирачко дрво со резултат со должина поголема од p^m, тогаш T има пат со должина поголема од m. Нека $m = |V|$, $K = p^m$, и нека w е збор од $L(G)$ со должина поголема од K. Нека T е парсирачко дрво со корен означен со S и резултат w. Тогаш T има барем еден пат со повеќе од $|V|+1$ темиња, па според тоа барем еден пат кој вклучува две темиња означени со истиот елемент $A \in V$. Да го претставиме тој пат графички:



Слика 4.7:

и да го разгледаме парсирачкото дрво T' чиј корен е означен со A, а листовите се листови од T, освен темето означено со A. Значи T' има теме означено со A и резултат од облик vAy, за некои $v, y \in \Sigma^*$. Возможно е $v = y = \lambda$, но тоа не е можно при сите можни избори на пат и две темиња со иста ознака. За $v = y = \lambda$, парсирачкото дрво T' може да се одстрани од T без да го промени резултатот на T, со додавање на парсирачкото дрво со темето означено со A што е пониско како корен, прикачено за погорното теме означено со A. Ако секој пат со должина што ја надминува m може да се скрати на овој начин, без да го промени резултатот на парсирачкото дрво T, ќе се добие парсирачко дрво со резултат w и висина помала од m, што не е

можно. Но, тогаш A, u, v, x, y и z можат да се определат од T. □

Оваа теорема е полезна за докажување дека некој јазик не е контекстно слободен.

Теорема 4.3.4 *$L = \{a^n b^n c^n | n \geq 0\}$ не е контекстно слободен.*

Доказ: Теоремата ќе ја докажеме со контрадикција и примена на теоремата за пумпање. Нека $L = L(G)$ за некоја контекстно слободна граматика G. Нека K е константа за G според теоремата за пумпање и нека $n = \frac{K}{3}$. Тогаш $w = a^n b^n c^n$ е во $L(G)$ и има претставување $w = uvxyz$ такво што или v или y се непразни и $uv^i xy^i z$ е во $L(G)$ за секој $i = 1, 2, \cdots$. Но ова не е можно бидејќи ако v или y содржат два симбола од $\{a, b, c\}$, тогаш uv^2xy^2z содржи b пред a или c пред b. Ако пак или v или y содржат само еден симбол од $\{a, b, c\}$, тогаш uv^2xy^2z не може да содржи еднаков број на симболи a, b, и c. □

Теорема 4.3.5 *Контекстно слободните јазици не се затворени во однос на операциите пресек или комплемент.*

Доказ: Јасно е дека $\{a^n b^n c^m | m, n \geq 0\}$ и $\{a^m b^n c^n | m, n \geq 0\}$ се контекстно слободни јазици, но нивниот пресек $\{a^n b^n c^n | n \geq 0\}$ не е.

Кога контекстно слободен јазик би бил затворен во однос на комплемент, тогаш би бил затворен и во однос на пресек. Имено:

$$L_1 \cap L_2 = \Sigma^* \setminus ((\Sigma^* \setminus L_1) \cup (\Sigma^* \setminus L_2)).$$ □

4.3.3 Задачи

1 Да се конструира R-граматика што ги генерира:

   (а) сите зборови во $\Sigma = \{a_1, \cdots, a_n\}$ кои содржат појавување на даден подзбор $x = a_{i_1}, \cdots, a_{i_s}$;

   (б) сите зборови во $\Sigma = \{a_1, \cdots, a_n\}$ кои содржат барем n појавувања на дадениот подзбор x;

   (в) сите зборови во $\Sigma = \{a_1, \cdots, a_n\}$ кои не содржат појавување на дадениот подзбор x;

   (г) сите зборови во $\Sigma = \{a_1, \cdots, a_n\}$ кои содржат точно n појавувања на дадениот подзбор x;

2 Да се конструира R-граматика што го препознава јазикот:

   (а) $\Sigma_n^+ = \{x | x \in \Sigma^+, |x| \geq n\}$, каде што $\Sigma = \{a_1, \cdots, a_n\}$;

   (б) $\Sigma_{sl}^+ = \{x | x \in \Sigma^+, |x| = si + l, i \in N\}$, каде што $s, l \geq 0$;

   (в) $w_1^+ \cdots w_n^+$, каде што w_i се произволни непразни зборови во Σ.

3 Да се конструира KS-граматика што го генерира множеството:

(а) правилно формирани формули од исказното сметање во "обичен" запис со фиксиран избор на променливи;

(б) $\{a^n b^m | n, m \in N, n < m\}$;

(в) $\{x_1 b x_2 | x_1, x_2 \in \{a_1, \cdots a_n\}^* | x_1| < |x_2|\}$;

(г) $\{a^{p+m} b^{m+n} c^{n+p} | n, m, p \in N\}$;

(д) $\{uawb | u, w \in \{a, b\}^*, |u| = |w|\}$;

(ѓ) $\{a^m b^n c^p d^q | m + n = p + q\}$;

(е) $\{a^m b^n | 2n \geq m \geq n\}$.

(ж) $\{wcw^R | w \in \{a, b\}^*, c \notin \{a, b\}\}$;

(з) $\{ww^R | w \in \{a, b\}^*\}$;

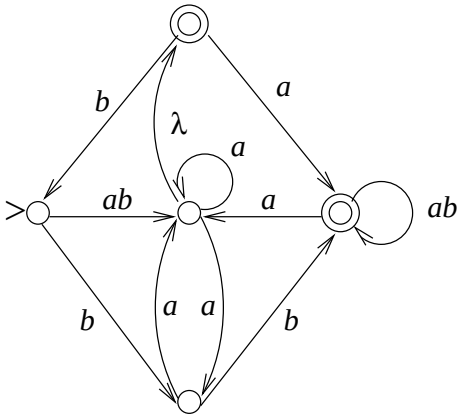
(ѕ) $\{w \in \{a, b\}^* | w = w^R\}$;

(и) $\{ucw | u, w \in \{a, b\}^*, c \notin \{a, b\}, \#a(u) = \#b(u)\}$;

(ј) $\{w \in \{a, b\}^* | w = xu^r uvv^r x^r, x, u, v \in \{a, b\}^*, r \in N\}$;

(к) $\{|^{2n} * |^m = |^{n+m+2} | n, m \in N\}$.

4 Конструирај R-граматика која го генерира јазикот препознаен од следниов автомат:



Слика 4.8:

5 Конструирај недетерминистички конечен автомат кој го препознава јазикот генериран од граматиката

$$G = (V, \Sigma, S, R),\ V = \{S, A, B\},\ \Sigma = \{a, b\}$$

$$R = \{S \to abA, S \to B, S \to baB, S \to \lambda, A \to bS, A \to b, B \to aS\}.$$

6 Да се конструира контекстно слободна граматика која го генерира јазикот на добро формирани загради во азбуката $\Sigma = \{), (\}$.

7 Дефинирај детерминистички push-down автомат (ДПДА) кој го прифаќа јазикот:

(а) $L = \{a^m b^n c^k | m + k \neq n\}$;
(б) $L = \{wcucw^R | w, u \in \{a, b\}^*, c \notin \{a, b\}^*\}$;
(в) $L = \{w \in \{a, b, c, d\}^* | \#a(w) + \#c(w) = \#b(w) + \#d(w)\}$.

8 Конструирај ДПДА кој го препознава јазикот

$$L = \{w \in \{a, b\}^* | \#aa(w) \neq \#ab(w)\}.$$

9 Конструирај ДПДА кој го препознава јазикот

$$L = \{a_1 a_2 \cdots a_n c b_1 b_2 \cdots b_n | a_i, b_i \in \{a, b\},\ n \in N^+, a_1 = b_1, a_n = b + n, c \notin \{a, b\}^*\}.$$

10 Конструирај push-down автомат (ПДА) кој го препознава јазикот

$$L = \{uwu^R | u, w \in \{a, b\}^*, \#a(w) = \#b(w)\}.$$

11 Конструирај ДПДА кој го препознава јазикот

$$L = \{a_m b_n a^{n+3m} | n > 0,\ m \geq 0\}.$$

12 Конструирај ПДА кој го препознава јазикот

$$L = \{a_n b_m | n, m \in N,\ n \leq m \leq 3n\}.$$

13 Да се конструира ДПДА кој го препознава јазикот L што се состои од сите зборови $w \in \{a, b\}^*$ кои започнуваат со aaa или имаат парен број појавувања на b. Каков јазик е L? Дали е потребен стекот?

14 Да се конструира ДПДА кој го препознава јазикот

$$L = \{wcu | u, w \in \{a, b\}^*, c \notin \{a, b\}^*, |w| + |u| \text{ е парен број}, \#a(w) = \#b(w)\}.$$

15 Да се конструира ДПДА кој го препознава јазикот L кој се состои од сите зборови $w \in \{a, b, c\}^*$ такви што последната буква во w е единствена таква буква во w. Каков јазик е L? Дали е потребен стекот?

16 Да се конструира ПДА кој го препознава јазикот:

$$L = \{a_1 a_2 \cdots a_n b_n \cdots b_2 b_1 | a_i, b_i \in \{a, b\}, (\exists i \in \{1, 2, \cdots, n\}) a_i = b_i\}.$$

17 Да се конструира ДПДА кој го препознава јазикот:

$$L = \{w \in \{a, b\}^* | |w| \text{ е парен број},\ |\#a(w) - \#b(w)| \geq 4\}.$$

Индекс

Литература

[1] А. И. Мальцев: *"Алгоритми и рекурзивние функции"*. Наука, Москва 1965

[2] Б. А. Кушнер: *"Лекции по конструктивному математическому анализу"*. Наука, Москва, 1973

[3] J. E. Hopcroft, J. D. Ullman: *"Formal languages and their relation to automata"*. Addison-Wesley Publishing Company, 1969

[4] E. Mendelson *"Introduction to mathematical logic"*. D. Van Nostrand Company, 1979 (second edition)

[5] H. R Lewis, C. H. Papadimitriou *"Elements of the theory of computation"*. Prentice-Hall International, Inc, 1981

[6] Д. Бошначки, А. Соколова, З. Шуниќ "Алгоритми и сложеност", Математичка Школа, Струга,'96,