Sveučilište u Splitu

Fakultet elektrotehnike, strojarstva i brodogradnje

MODELI RAČUNARSTVA

ODGOVORI NA TEORIJSKA PITANJA – NESLUŽBENA STUDENTSKA SKRIPTA

Niste vi ništa gori od studeata FER-a, dapače...

(Julije Ožegović, 10.10.2016.)

AUTORI:

GLIGORIJE ČUPKOVIĆ

KARLO MARDEŠIĆ

ISPRAVCI I SUGESTIJE:

ANTE VULETIĆ ANTIĆ

JOSIP KRNJIĆ

SPLIT, AK. GODINA 2016/17

SADRŽAJ

1 UVODNA RAZMATRANJA

1.1 JEZIČNI PROCESOR

- Uloga jezičnog procesora
- Jezici u postupku prevođenja
- Definicija jezika

Osnovna uloga jezičnog procesora je prevođenje zapisa algoritma iz izvornog jezika L_i u zapis algoritma u ciljnom jeziku L_c koji je moguće izvesti na zadanom računalu.

Na temelju dane definicije **jezični procesor** prikazujemo kao:

$$JP_{L_g}^{L_i \to L_c}$$

JP – Jezični procesor
L_i – Izvorni jezik
L_c – Ciljni jezik
L_g – Jezik izgradnje (najčešće $L_c = L_g$)

L_{svi} označava skup svih nizova koje je moguće napisati primjenom zadanih znakova abecede, dekadskih znamenaka, operatora, posebnih znakova i pravopisnih znakova.

Jezik se definira skupom nizova nad nekom abecedom.

1.2 PROCESI JEZIČNOG PROCESORA

- Uloga formalnog automata i formalne gramatike
- Faze i razine rada jezičnog procesora
- Sintaksa i semantika

Formalni automat M je matematički model automata koji odlučuje pripada li ulazni niz zadanom jeziku.

Formalna gramatika M je matematički model koji primjenom skupa produkcija generira nizove znakova.

Različite **razine rada** prevođenja podijeljene su u dvije **osnovne faze rada** jezičnog procesora:

1) Faza analize izvornog programa
   Razina 1. Leksička analiza
   Razina 2. Sintaktička i semantička analiza – generiranje višeg međukoda
2) Faza sinteze ciljnog programa
   Razina 3. Prevođenje višeg u srednji međukod
   Razina 4. Prevođenje srednjeg u niži međukod
   Razina 5. Prevođenje nižeg međukoda u ciljni program

Sintaksa definira jezik kao skup svih dozvoljenih nizova leksičkih jedinki. Sintaktička pravila definiraju strukturu programa i način gradnje izraza od leksičkih jedinki, naredbi i blokova naredbi.

Semantika jezika određuje skup dozvoljenih značenja (npr. provjera tipa varijable). Semantička pravila povezuju ponašanje računala s izvođenjem programa (npr. dodjela tipa varijable).

1.3 ZNAKOVLJE I OZNAKE

- Znak, niz, nadovezivanje, dijelovi niza
- Jezik i operacije nad jezicima
- Skupovi, kardinalni broj
- Graf, usmjereni graf i stablo

Znak je elementarni simbol od kojeg se grade riječi.

Abeceda je skup znakova (npr. $B = \{0, 1\}$).

Niz je konačan slijed znakova abecede postavljenih jedan do drugog. Prazni niz označava se znakom ε.

Nadovezivanje nizova označava se potencijama. Potencije definiraju sljedeće izraze:

$$w^0 = \varepsilon \qquad w^i = w^{i-1}w \text{ za } i > 0 \qquad w^1 = w^0w = \varepsilon w = w$$

Razlikujemo **dijelove niza**: prefiks, sufiks, podniz i podslijed.

Jezik se definira skupom nizova nad nekom abecedom. Definirane su sljedeće **operacije nad jezicima**:

Unija jezika L i N	$L \cup N$	$= \{w \mid w \in L \lor w \in N\}$
Presjek jezika L i N	$L \cap N$	$= \{w \mid w \in L \land w \in N\}$
Razlika jezika L i N	$L - N$	$= \{w \mid w \in L \land w \notin N\}$
Nadovezivanje L i N	LN	$= \{xy \mid x \in L \land y \in N\}$
Kartezijev produkt	$L \times N$	$= \{(x, y) \mid x \in L \land y \in N\}$
Partitivni skup	2^L	$= \{X \mid X \subseteq L\}$
Kleeneov operator *	L^*	$= \bigcup_{i=0}^{\infty} L^i$
Kleeneov operator +	L^+	$= \bigcup_{i=1}^{\infty} L^i$
Komplement jezika	L^c	$= \{w \mid w \notin L\}$

Kardinalni broj je broj članova nekog skupa.

Skupove dijelimo na:

- Prebrojivo beskonačne skupove kod kojih postoji bijekcija na skup prirodnih brojeva.
- Neprebrojivo beskonačne skupove kod kojih ne postoji bijekcija na skup prirodnih brojeva.

Graf $G = (V, E)$ čini konačni skup čvorova V i skup parova čvorova E. Parovi čvorova čine grane grafa.

Grane **usmjerenog grafa** su uređeni parovi koje nazivamo usmjerenim granama. Usmjerena grana od čvora v do čvora w označava se izrazom $v \rightarrow w$.

Stablo je usmjereni graf sljedećih svojstava:

1) Korijen je čvor bez prethodnika i od njega vodi put do svih ostalih čvorova.
2) Bilo koji čvor, osim korijena stabla, ima točno jednog neposrednog prethodnika.

2 DETERMINISTIČKI KONAČNI AUTOMAT

2.1 REGULARNI JEZIK I KONAČNI AUTOMAT

- Regularni jezik i konačni automati
- Definicija determinističkog konačnog automata

Jezik je **regularan** ako i samo ako postoji **konačni automat** koji ga prihvaća. Time je definirana istovjetnost konačnih automata i regularnih jezika – za bilo koji regularni jezik moguće je izgraditi konačni automat koji ga prihvaća i obrnuto.

Vrste konačnih automata:

- Deterministički konačni automat (DKA)
- Nedeterministički konačni automat (NKA)
- Epsilon-nedeterministički konačni automat (ε-NKA)

DKA se formalno definira uređenom petorkom:

$$dka = (Q, \Sigma, \delta, q_0, F) \tag{1}$$

gdje je:

Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
$\delta: Q \times \Sigma \to Q$	funkcija prijelaza
$q_0 \in Q$	početno stanje
$F \subseteq Q$	skup prihvatljivih stanja

2.2 DETERMINISTIČKI KONAČNI AUTOMAT I NIZ

- Proširena funkcija prijelaza DKA
- Prihvaćanje ulaznog niza DKA

Funkcija prijelaza jednoznačno određuje prijelaz u iduće stanje što se zapisuje na sljedeći način:

$$NovoStanje = \delta(StaroStanje, UlazniZnak)$$

Proširena funkcija prijelaza $\hat{\delta}: Q \times \Sigma^* \to Q$ definira stanje automata nakon čitanja ulaznog niza. Oznaka Σ^* označava skup svih mogućih nizova ulaznih znakova, uključujući i prazni niz ε. Funkcija $\hat{\delta}$ definira se na sljedeći način:

- $\hat{\delta}(q, \varepsilon) = q$
- $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$, gdje je $w \in \Sigma^*$, $a \in \Sigma$

Deterministički konačni automat $dka = (Q, \Sigma, \delta, q_0, F)$ **prihvaća niz** $x \in \Sigma^*$ ako vrijedi:

$$\delta(q_0, x) = p, p \in F$$

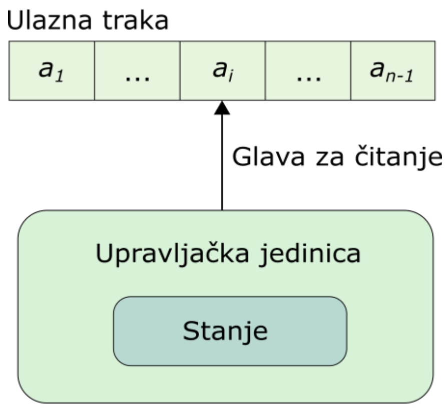
DKA prihvaća skup $L(dka) = \{x \mid \delta(q_0, x) \in F\}$ koji je podskup svih mogućih nizova ($L(dka) \subseteq \Sigma^*$). Za sve ostale nizove koji nisu u skupu $L(dka)$ kaže se da ih DKA ne prihvaća.

2.3 MODEL DKA I PROGRAMSKO OSTVARENJE

- Model DKA
- Programsko ostvarenje DKA
- Pretvorba ulaznog niza
- Načini ostvarenja funkcija prijelaza

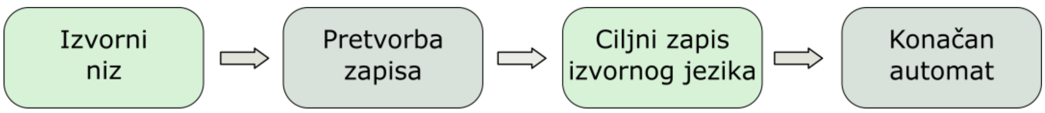
Rad konačnog automata opisan je modelom na slici 2.1. Za **model DKA** vrijedi:

- Ulazna traka je konačna
- Glava samo čita znakove (ne može pisati)
- Glava se pomiče u desno



Slika 2.1 Model konačnog automata

Konačni automat je matematički model koji je moguće **ostvariti programskim jezikom** (slika 2.2). Za potrebe učinkovitog programskog ostvarenja razmatraju se načini zapisa ulaznih znakova, stanja i funkcije prijelaza.



Slika 2.2 Jednostavni model pretvorbe znakova

Dva su **osnovna načina zapisa stanja**:

- Izravan način – zapis stanja u varijablu
- Posredan način – stanje se određuje na temelju dijela programa koji se izvodi

Načini ostvarenja funkcija prijelaza:

1) Vektorski pristup
    - Za svako stanje automata definira se po jedan vektor
    - Vektori sadrže onoliko elemenata koliko je različitih ulaznih znakova
    - Kod izravnog načina zapisa element vektora je stanje
    - Kod posrednog načina zapisa element vektora je adresa potprograma
2) Korištenje liste
    - Učinkovito korištenje memorije
    - Unosi se lista parova *ulazni znak – stanje*
    - Dugo vrijeme pretraživanja liste

2.4 DEFINICIJE EKVIVALENTNOSTI

- Ekvivalentnost stanja i automata
- Svojstva ekvivalentnosti
- Strategija redukcije broja stanja
- Uvjeti istovjetnosti stanja

Stanje p DKA M **istovjetno je** stanju p' DKA M' ako i samo ako DKA M u stanju p prihvaća isti niz znakova kao i DKA M' u stanju p':

$$(\delta(p,w) \in F \ \wedge \ \delta'(p',w) \in F') \ \vee \ (\delta(p,w) \notin F \ \wedge \ \delta'(p',w) \notin F'), \ \ \forall\, w \in \Sigma^*$$

Iz definicije istovjetnosti stanja slijedi – **automati** M i N **su istovjetni** ako i samo ako su istovjetna njihova početna stanja.

Za relaciju istovjetnosti vrijedi **svojstvo tranzitivnosti** – ako su stanja p i q istovjetna, te ako su stanja q i r istovjetna, tada su istovjetna i stanja p i r.

Strategija redukcije broja stanja temelji se na zamjeni istovjetnih stanja jednim jedinstvenim stanjem:

1) Sva istovjetna stanja označe se zajedničkim imenom.
2) Sve oznake istovjetnih stanja u funkciji prijelaza označe se izabranim zajedničkim imenom.
3) U skupu Q ostavi se samo jedno od istovjetnih stanja.

Ispitivanje istovjetnosti stanja p i q svodi se na ispitivanje dva uvjeta:

- **Uvjet podudarnosti** – Stanja p i q moraju biti ili oba prihvatljiva ili oba neprihvatljiva:
$$(p \in F \wedge q \in F) \vee (p \notin F \wedge q \notin F)$$
- **Uvjet napredovanja** – Za bilo koji ulazni znak a vrijedi da su stanja $\delta(p,a)$ i $\delta(q,a)$ istovjetna.

Za smanjivanje broja stanja koriste se tri algoritma:

- Algoritam primitivne tablice
- Huffman-Mealy algoritam
- Algoritam tablice implikanata

3 MINIMALIZACIJA KONAČNOG AUTOMATA

3.1 ALGORITAM PRIMITIVNE TABLICE

- Koraci algoritma
- Primjer

Koraci algoritma primitivne tablice:

1) Za svaki ulazni znak gradi se zasebni stupac tablice. Par stanja čija se istovjetnost ispituje upisuje se u prvi redak tablice.
2) Ispituje se uvjet podudarnosti stanja. Ako par stanja u retku nije podudaran, onda ta stanja sigurno nisu istovjetna i algoritam se zaustavlja.
3) Provjerava se uvjet napredovanja. Ako stanja u paru nisu ista i ako nisu već zapisana u jednom od prethodnih redaka, stvara se novi redak za taj par stanja.
4) Ako u prethodnom koraku nije zapisan novi redak u tablicu, onda je par stanja u prvom retku istovjetan, kao i svi parovi u ostalim recima tablice. Ako je zapisan novi redak, algoritam se ponavlja od drugog koraka.

Za **primjer** minimizirajmo DKA zadan tablicom 3.1.

	0	**1**	**⊥**
$\boldsymbol{q_0}$	q_1	q_2	0
$\boldsymbol{q_1}$	q_0	q_3	0
$\boldsymbol{q_2}$	q_4	q_5	1
$\boldsymbol{q_3}$	q_3	q_5	1
$\boldsymbol{q_4}$	q_2	q_5	1
$\boldsymbol{q_5}$	q_5	q_5	0

Tablica 3.1 Primjer DKA s ekvivalentnim stanjima

Želimo ispitati istovjetnost stanja q_0 i q_1. Stoga prvo zapišemo taj par stanja u prvi redak tablice. Stanja su podudarna (oba su neprihvatljiva) pa nastavljamo s ispitivanjem uvjeta napredovanja koji za prvi par stanja ovisi o istovjetnosti stanja q_0 i q_1 za ulazni znak 0 i o istovjetnosti q_2 i q_3 za ulazni znak 1. Par q_0q_1 već je obrađen pa novi redak stvaramo samo za par q_2q_3 i ponavljamo postupak. Algoritam završava parom q_3q_4 za koji ne možemo stvarati nove retke. Na osnovu minimizacije prikazane tablicom 3.2 zaključujemo da su stanja q_0 i q_1 istovjetna. Također su istovjetna stanja q_2, q_3 i q_4 zbog uvjeta napredovanja pa možemo ispisati tablicu minimiziranog automata (tablica 3.3).

	0	**1**
$\boldsymbol{q_0q_1}$	q_0q_1	q_2q_3
$\boldsymbol{q_2q_3}$	q_3q_4	q_5
$\boldsymbol{q_3q_4}$	q_2q_3	q_5

Tablica 3.2 Postupak minimizacije primitivnom tablicom

	0	**1**	**⊥**
$\boldsymbol{q_0}$	q_0	q_2	0
$\boldsymbol{q_2}$	q_2	q_5	1
$\boldsymbol{q_5}$	q_5	q_5	0

Tablica 3.3 DKA dobiven minimizacijom

3.2 Algoritam Huffman-Mealy

- Koraci algoritma
- Primjer

Koraci algoritma:

1) Skup stanja podijeli se u dvije grupe. U prvoj grupi su sva prihvatljiva, a u drugoj sva neprihvatljiva stanja.
2) Ispituje se zatvorenost podskupova, tj. svaka grupa G_j u trenutnoj podjeli dijeli se na nove podskupove tako da su dva stanja p i q u istom podskupu ako i samo ako za bilo koji ulazni znak a vrijedi:
$$\delta(p,a) \in G_i \land \delta(q,a) \in G_i$$
gdje je G_i jedna od grupa trenutne podjele.
3) Ako je podjela na grupe ostala ista, tj. ako u prethodnom koraku nisu nastale nove grupe, algoritam se zaustavlja i stanja u istim grupama su istovjetna. U suprotnom algoritam se nastavlja drugim korakom za novonastalu podjelu.

Primjer Huffman-Mealy minimizacije automata zadanog tablicom 3.1 prikazan je tablicama 3.4, 3.5, 3.6 i 3.7.

G_1	0	1
q_0	G_1	G_2
q_1	G_1	G_2
q_5	G_1	G_1

Tablica 3.4 Grupa neprihvatljivih stanja zadanog DKA

G_2	0	1
q_2	G_2	G_1
q_3	G_2	G_1
q_4	G_2	G_1

Tablica 3.5 Grupa prihvatljivih stanja zadanog DKA

G_{11}	0	1
q_0	G_{11}	G_2
q_1	G_{11}	G_2

Tablica 3.6 Prvi podskup grupe G_1

G_{12}	0	1
q_5	G_{12}	G_{12}

Tablica 3.7 Drugi podskup grupe G_1

Stanja su prvo podijeljena na dvije grupe: prihvatljiva i neprihvatljiva. Grupa neprihvatljivih stanja (G_1) ima prijelaze u G_1 i G_2 iz stanja q_0 i q_1, dok iz stanja q_5 uvijek ostaje u grupi G_1. Stoga je dijelimo na dva podskupa, od kojih su u prvom stanja q_0 i q_1, a u drugom samo q_5. Novonastale grupe G_{11} i G_{12} ne mogu se dalje dijeliti, kao ni grupa G_2, jer iz svih stanja prelaze u istu grupu za isti ulazni znak.

Dakle, minimizacijom se dobije da su stanja q_0 i q_1 istovjetna, kao i stanja q_2, q_3 i q_4. Preostalo je samo ispisati tablicu minimalnog DKA (tablica 3.3).

3.3 ALGORITAM TABLICE IMPLIKANATA

- Koraci algoritma
- Primjer

Koraci algoritma:

1) Svi neistovjetni parovi stanja označe se na osnovu uvjeta podudarnosti.
2) Za bilo koji par različitih stanja koja su ili oba neoznačena ili oba označena:
   Ako za neki znak a par $(\delta(p,a),\delta(q,a))$ jest označen, označi se i par (p,q) i rekurzivno se označe svi parovi u listi pridruženoj paru (p,q) kao i svi parovi u listama pridruženima parovima koji su označeni u ovom koraku.
   Inače za svaki ulazni znak a ako je $\delta(p,a) \neq \delta(q,a)$, par (p,q) stavlja se u listu pridruženu paru $(\delta(p,a),\delta(q,a))$.

Primjer minimizacije automata zadanog tablicom 3.1 tablicom implikanata prikazan je u tablici 3.8 i tablici 3.9.

q_1					
q_2	×	×			
q_3	×	×			
q_4	×	×			
q_5			×	×	×
	q_0	q_1	q_2	q_3	q_4

Tablica 3.8 Tablica implikanata nakon prvog koraka algoritma

q_1					
q_2	×	×			
q_3	×	×			
q_4	×	×			
q_5	×	×	×	×	×
	q_0	q_1	q_2	q_3	q_4

Tablica 3.9 Tablica implikanata nakon drugog koraka algoritma

Prvo su znakom „×" označeni parovi prihvatljivih stanja. Počevši od para (q_0,q_1) provjerava se jesu li parovi stanja u koja q_0 i q_1 prelaze označena. To su parovi (q_0,q_1) za ulazni znak 0 i (q_2,q_3) za ulazni znak 1. S obzirom da nijedan od tih parova nije označen, q_0 i q_1 stavljaju se u liste pridružene tim parovima. U nastavku drugog koraka dobiva se tablica 3.9 iz koje se očitaju istovjetna stanja.

Iz dobivene tablice implikanata jednostavno se očita da su ekvivalentna stanja q_0 i q_1 te stanja q_2, q_3 i q_4. Na kraju se ispiše tablica minimalnog DKA (tablica 3.3).

3.4 NEDOHVATLJIVA STANJA

- Definicija nedohvatljivih stanja
- Algoritam eliminacije
- Algoritam postizanja DKA s minimalnim brojem stanja

Stanje p DKA $M = (Q,\Sigma,\delta,q_0,F)$ je **nedohvatljivo** ako ne postoji niti jedan niz $w \in \Sigma^*$ za koji vrijedi da je $p = \delta(q_0,w)$.

Eliminacija nedohvatljivih stanja obavlja se algoritmom traženja dohvatljivih stanja:

1) U listu dohvatljivih stanja DS upiše se početno stanje q_0.

2) Lista DS proširi se skupom stanja $\{p \mid p = \delta(q_0, a),\ \forall a \in \Sigma\}$.
3) Za svako stanje $q_i \in DS$ lista se proširi skupom stanja $\{p \mid p = \delta(q_i, a),\ p \notin DS,\ \forall a \in \Sigma\}$. Ovaj se korak ponavlja sve dok se u listu DS mogu dodavati nova stanja.

Kad se u listu dohvatljivih stanja više ne mogu dodati nova stanja, sva stanja koja nisu u listi su nedohvatljiva.

DKA s minimalnim brojem stanja dobiva se odbacivanjem istovjetnih i nedohvatljivih stanja. Ne postoji nijedan drugi DKA koji prihvaća isti jezik, a ima manji broj stanja. S obzirom da je postupak eliminacije nedohvatljivih stanja jednostavniji od minimizacije, učinkovitije je prvo obaviti odbacivanje nedohvatljivih stanja pa zatim odbacivanje istovjetnih stanja.

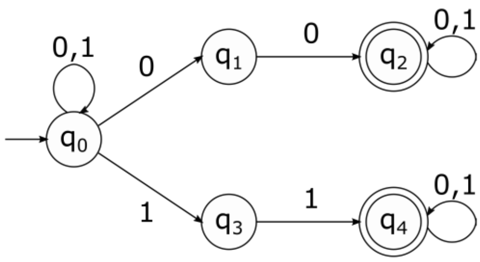
4 NEDETERMINISTIČKI KONAČNI AUTOMAT

4.1 PRISTUP I RAD NKA

- Opis NKA
- Primjer i rad NKA
- Formalna definicija NKA

Funkcija prijelaza **NKA** određuje prijelaz u skup stanja $\delta(q, a) = \{p_1, p_2 \dots p_n\}$. Skup stanja može biti i prazan. Za bilo koji NKA N moguće je izgraditi DKA D koji prihvaća isti jezik $L(N) = L(D)$.

Na slici 4.1 prikazan je **primjer** NKA koji prihvaća nizove koji sadrže barem dvije uzastopne nule ili jedinice.



Slika 4.1 Primjer NKA

Rad NKA opisuje se na sljedeći način: svaki put kada postoji mogućnost prijelaza u više različitih stanja za jedan ulazni znak, stvori se upravo toliko DKA koliko ima prijelaza u različita stanja. Svi nastali DKA paralelno obrađuju ostatak ulaznog niza. Niz se prihvaća ako barem jedan DKA ima slijed prijelaza koji završava u prihvatljivom stanju.

NKA se formalno definira uređenom petorkom:

$$nka = (Q, \Sigma, \delta, q_0, F) \qquad (2)$$

gdje je:

Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
$\delta: Q \times \Sigma \to 2^Q$	funkcija prijelaza
$q_0 \in Q$	početno stanje
$F \subseteq Q$	skup prihvatljivih stanja

4.2 DEFINICIJA NKA

- Formalna definicija NKA
- Proširena funkcija prijelaza NKA
- Prihvaćanje ulaznog niza
- Funkcija prijelaza nad skupom stanja

Formalna definicija NKA nalazi se u prethodnom pitanju (jednadžba (2)).

Proširena funkcija prijelaza $\hat{\delta}: Q \times \Sigma^* \rightarrow 2^Q$ koja određuje stanje automata nakon pročitanog ulaznog niza, definira se na sljedeći način:

$$\hat{\delta}(q, \varepsilon) = \{q\} \tag{3}$$

$$\hat{\delta}(q, wa) = P = \{p \mid p \in \delta(r, a), \forall r \in \hat{\delta}(q, w)\} \tag{4}$$

gdje je $w \in \Sigma^*$, $a \in \Sigma$ i $P \subseteq Q$.

U (3) je zadano da automat ne može promijeniti stanje, dok se ne pročita barem jedan ulazni znak. Uvrštavanjem $w = \varepsilon$ u (4) dobije se:

$$\hat{\delta}(q, \varepsilon a) = \hat{\delta}(q, a) = \{p \mid p \in \delta(q, a)\} = \delta(q, a)$$

Time je dokazano da su obična i proširena funkcija prijelaza istovjetne.

NKA prihvaća ulazni niz ako se u skupu $P = \delta(q_0, w)$ nalazi barem jedno prihvatljivo stanje ($P \cap F \neq \emptyset, w \in \Sigma^*$).

NKA prihvaća skup $L(nka) \subseteq \Sigma^*$, $L(nka) = \{w \mid \delta(q_0, w) \cap F \neq \emptyset\}$. Za sve nizove koji nisu u $L(nka)$ kažemo da ih NKA ne prihvaća.

Funkcija prijelaza iz skupa stanja određuje stanje automata nakon čitanja ulaznog niza polazeći iz nekog od podskupova stanja $P \in 2^Q$:

$$\delta(P, w) = \bigcup_{q \in P} \delta(q, w) \tag{5}$$

4.3 IZGRADNJA DKA IZ ZADANOG NKA

- Definicija algoritma
- Problem nedostupnih stanja
- Primjer izgradnje

Za bilo koji NKA $M = (Q, \Sigma, \delta, q_0, F)$ moguće je izgraditi istovjetni DKA $M' = (Q', \Sigma, \delta', q_0', F')$. NKA i DKA su istovjetni ako prihvaćaju isti jezik.

Izgradnja DKA M' obavlja se na sljedeći način:

1) $Q' = 2^Q$, tj. $Q' = \{[\emptyset], [q_0], \dots, [q_i], [q_0, q_1], \dots, [q_0, q_i], \dots, [q_0, q_1, \dots, q_i]\}$, $q_0, \dots, q_i \in Q$.
2) $F' = \{[p_0, \dots, p_j] \mid \exists p_k \in F,\ 0 \leq k \leq j\}$, tj. F' je skup svih stanja $[p_0, \dots, p_j]$ gdje je barem jedan $p_k \in F$.
3) $q_0' = [q_0]$.
4) $\delta'([p_0, \dots, p_j], a) = [r_0, \dots, r_j]$ ako i samo ako je $\delta(\{p_0, \dots, p_j\}, a) = \{r_0, \dots, r_j\}$.

Mnoga stanja ovako dobivenog DKA su **nedostupna** te ih je potrebno eliminirati i minimizirati automat.

Za **primjer** uzmimo NKA $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$ zadan tablicom 4.1.

	0	**1**	**⊥**
q_0	$\{q_0, q_1\}$	$\{q_1\}$	1
q_1	$\emptyset$	$\{q_0\}$	0

Tablica 4.1 Primjer NKA zadan tablicom

Konstruirajmo ekvivalentni DKA $M' = (Q', \Sigma, \delta', q_0', F')$:

1) $Q' = \{[\emptyset], [q_0], [q_1], [q_0, q_1]\}$
2) $F' = \{[q_0], [q_0, q_1]\}$
3) $q_0' = [q_0]$
4) δ':

$$\delta'([q_0], 0) = [q_0, q_1] \qquad \delta'([q_0, q_1], 0) = [q_0, q_1]$$
$$\delta'([q_0], 1) = [q_1] \qquad \delta'([q_0, q_1], 1) = [q_0, q_1]$$
$$\delta'([q_1], 0) = [\emptyset] \qquad \delta'([\emptyset], 0) = [\emptyset]$$
$$\delta'([q_1], 1) = [q_0] \qquad \delta'([\emptyset], 1) = [\emptyset]$$

Izgrađeni DKA prikazan je tablicom 4.2[1].

	0	**1**	**⊥**
$[\boldsymbol{q_0}]$	$[q_0, q_1]$	$[q_1]$	1
$[\boldsymbol{q_1}]$	$[\emptyset]$	$[q_0]$	0
$[\boldsymbol{q_0}, \boldsymbol{q_1}]$	$[q_0, q_1]$	$[q_0, q_1]$	1
$[\emptyset]$	$[\emptyset]$	$[\emptyset]$	0

Tablica 4.2 Tablica DKA izgrađenog na temelju NKA danog u tablici 4.1

4.4 ISTOVJETNOST NKA I DKA

- Dokaz istovjetnosti

Neka je DKA $M' = (Q', \Sigma, \delta', q_0', F')$ konstruiran na temelju NKA $M = (Q, \Sigma, \delta, q_0, F)$. Potrebno je dokazati da M i M' prihvaćaju isti jezik $L(M) = L(M')$, odnosno da za bilo koji niz $w \in \Sigma^*$ vrijedi:

$$\delta'([q_0], w) = [r_0, r_1, \dots, r_j] \text{ ako i samo ako } \delta(q_0, w) = \{r_0, r_1, \dots, r_j\}$$

1) U prvom koraku dokazuje se da prethodna tvrdnja vrijedi za $|w| = 0$, tj. $w = \varepsilon$. S obzirom da DKA i NKA ne mogu promijeniti stanje ako se ne pročita barem jedan ulazni znak, očigledno je tvrdnja istinita za $w = \varepsilon$ jer je $q_0' = [q_0]$.
2) Pretpostavimo da vrijedi $\delta'([q_0], x) = [p_0, p_1, \dots, p_j]$ ako i samo ako $\delta(q_0, x) = \{p_0, p_1, \dots, p_j\}$ za neki niz $w = x$, $x \in \Sigma^*$. Potrebno je dokazati da tvrdnja vrijedi i za niz xa, gdje je $a \in \Sigma$.
3) Na temelju pretpostavke i definicije konstrukcije funkcije δ':

$$\delta'([p_0, p_1, \dots, p_j], a) = [r_0, r_1, \dots, r_j] \text{ ako i samo ako je } \delta(\{p_0, p_1, \dots, p_j\}, a) = \{r_0, r_1, \dots, r_j\}$$

Može se zaključiti da vrijedi:

$$\delta'([q_0], xa) = [r_0, r_1, \dots, r_j] \text{ ako i samo ako } \delta(q_0, xa) = \{r_0, r_1, \dots, r_j\}$$

Uzevši u obzir da je $\delta'([q_0], w) \in F'$ ako i samo ako $\delta(q_0, w) \cap F \neq \emptyset$, zaključujemo da M i M' prihvaćaju isti jezik.

[1] S obzirom da je ovaj primjer vrlo jednostavan, dobiveni DKA nema nedostupnih stanja.

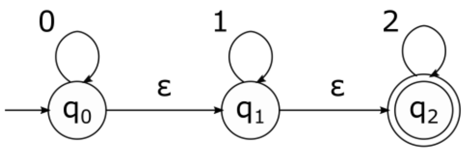
5 EPSILON-NEDETERMINISTIČKI KONAČNI AUTOMAT (E-NKA)

5.1 PRISTUP I RAD E-NKA

- Opis ε-NKA
- Primjer i rad ε-NKA
- Formalna definicija ε-NKA

Epsilon-nedeterministički konačni automat (ε-NKA) proširuje funkciju prijelaza NKA tako da se konačnom automatu omogućuje da promijeni stanje a da ne pročita niti jedan ulazni znak. Takva promjena stanja naziva se ε-prijelaz.

Primjer ε-NKA nalazi se na slici 5.1. Prikazani automat prihvaća nizove koji započinju proizvoljnim brojem nula, nastavljaju se proizvoljnim brojem jedinica i završavaju proizvoljnim brojem dvojki (što uključuje i prazan niz).



Slika 5.1 Primjer dijagrama stanja ε-NKA

E-NKA prihvaća niz ako postoji barem jedan slijed prijelaza iz početnog stanja u jedno od prihvatljivih stanja, uz uvjet da se pročitaju svi znakovi niza. U taj slijed prijelaza uključeni su i ε-prijelazi.

E-NKA formalno se definira uređenom petorkom:

$$\varepsilon\text{-}nka = (Q, \Sigma, \delta, q_0, F) \qquad (6)$$

gdje je:

Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$	funkcija prijelaza
$q_0 \in Q$	početno stanje
$F \subseteq Q$	skup prihvatljivih stanja

5.2 DEFINICIJA E-NKA

- Formalna definicija ε-NKA
- Definicija ε-okruženja
- Definicija proširene funkcije prijelaza

Formalna definicija ε-NKA nalazi se u prethodnom pitanju (jednadžba (6)).

E-okruženje je funkcija koja stanju $q \in Q$ pridružuje skup stanja $R = \{\, r \mid r = q \ \vee\ r = \delta(q, \varepsilon), r \in Q \,\}$. Dakle, ε-okruženje stanju q pridružuje skup stanja koji se sastoji od samoga stanja q i svih stanja u koja q prelazi isključivo ε-prijelazom. Za skup stanja $P \subseteq Q$ ε-okruženje računa se na sljedeći način:

$$\varepsilon\text{-}okruženje(P) = \bigcup_{q \epsilon P} \varepsilon\text{-}okruženje(q) \tag{7}$$

Proširena funkcija prijelaza $\hat{\delta}: Q \times \Sigma^* \to 2^Q$ određuje skup stanja za koje postoji slijed prijelaza (uključujući i ε-prijelaze) iz stanja q za zadani niz znakova w. Funkcija $\hat{\delta}$ definira se pomoću funkcije $\varepsilon\text{-}okruženje$:

$$\hat{\delta}(q, \varepsilon) = \varepsilon\text{-}okruženje(q) \tag{8}$$

$$\hat{\delta}(q, wa) = \varepsilon\text{-}okruženje(P) \tag{9}$$

gdje je $P = \{p \mid r \in \hat{\delta}(q, w) \Rightarrow p \in \delta(r, a), w \in \Sigma^*, a \in \Sigma\}$.

Za skupove stanja, funkcije δ i $\hat{\delta}$ računaju se na sljedeći način:

$$\delta(R, a) = \cup_{q \in R}\, \delta(q, a),\ R \subseteq Q, a \in \Sigma \tag{10}$$

$$\hat{\delta}(R, w) = \cup_{q \in R}\, \hat{\delta}(q, w),\ R \subseteq Q, w \in \Sigma^* \tag{11}$$

5.3 IZGRADNJA NKA IZ ZADANOG E-NKA

- Definicija algoritma izgradnje
- Primjer izgradnje

Za bilo koji ε-NKA $M = (Q, \Sigma, \delta, q_0, F)$ moguće je **izgraditi** ekvivalentni NKA $M' = (Q', \Sigma, \delta', q_0', F')$ kroz sljedeće korake:

1) $Q' = Q$
2) $q_0' = q_0$
3) $F' = F \cup \{q_0\}$ ako $\varepsilon\text{-}okruženje(q_0)$ sadrži barem jedno stanje skupa F ($F \cap \varepsilon\text{-}okruženje(q_0) \neq \emptyset$), inače $F' = F$
4) $\delta'(q, a) = \hat{\delta}(q, a),\ \forall a \in \Sigma\ i\ \forall q \in Q$

Primjer izgradnje za ε-NKA zadan dijagramom na slici 5.1:

1) $Q' = Q = \{q_0, q_1, q_2\}$
2) $q_0' = q_0$
3) $F' = F \cup \{q_0\} = \{q_0, q_2\}$, jer je $F \cap \varepsilon\text{-}okruženje(q_0) = \{q_2\} \cap \{q_0, q_1, q_2\} = \{q_2\}$
4) Funkcija δ' određuje se na sljedeći način za svaku kombinaciju stanja i ulaznog znaka:

$$\delta'(q_0, 0) = \hat{\delta}(q_0, 0)$$

(9) $\Rightarrow$ $\delta'(q_0, 0) = \hat{\delta}(q_0, \varepsilon 0) = \varepsilon\text{-}okruženje\left(\delta\left(\hat{\delta}(q_0, \varepsilon), 0\right)\right)$

(8) $\Rightarrow$ $\delta'(q_0, 0) = \varepsilon\text{-}okruženje\big(\delta(\varepsilon\text{-}okruženje(q_0), 0)\big)$

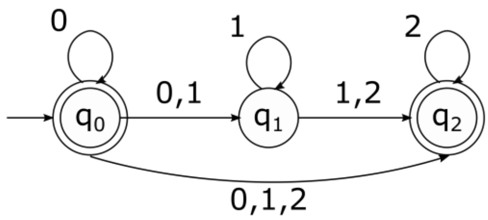
$= \varepsilon\text{-}okruženje\big(\delta(\{q_0, q_1, q_2\}, 0)\big)$

(2) $\Rightarrow$ $\delta'(q_0, 0) = \varepsilon\text{-}okruženje(\{q_0\} \cup \emptyset \cup \emptyset) = \{q_0, q_1, q_2\}$

Nastavimo li postupak, dobivamo NKA definiran tablicom 5.1.

	0	1	2	⊥
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$	1
q_1	$\emptyset$	$\{q_1, q_2\}$	$\{q_2\}$	0
q_2	$\emptyset$	$\emptyset$	$\{q_2\}$	1

Tablica 5.1 Tablica prijelaza NKA konstruiranog na temelju ε-NKA zadanog na slici 5.1



Slika 5.2 Dijagram stanja NKA konstruiranog na temelju ε-NKA zadanog na slici 5.1

5.4 ISTOVJETNOST E-NKA I NKA

- Dokaz istovjetnosti

Istovjetnost se dokazuje u dva koraka indukcijom s obzirom na duljinu niza x:

1) Dokazuje se da vrijedi $\delta'(q_0, x) = \hat{\delta}(q_0, x)$:

Za duljinu niza $|x| = 1$, koji se sastoji isključivo od znaka a, vrijedi da je $\delta'(q_0, x) = \hat{\delta}(q_0, x)$ na temelju definicije algoritma izgradnje.

Neka je niz $x = wa$, $a \in \Sigma$ i $w \in \Sigma^*$. Potrebno je dokazati da je $\delta'(q_0, wa) = \hat{\delta}(q_0, wa)$.

Induktivna hipoteza:

$$P = \delta'(q_0, w) = \hat{\delta}(q_0, w)$$

U definiciju funkcije $\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a)$ uvrstimo jednakost hipoteze:

$$\delta'(\delta'(q_0, w), a) = \delta'(P, a)$$

Na temelju definicije funkcije δ' NKA (jednadžba (5)) vrijedi $\delta'(P, a) = \bigcup_{q \in P} \delta'(q, a)$, a na temelju četvrtog koraka definicije algoritma izgradnje vrijedi $\bigcup_{q \in P} \delta'(q, a) = \bigcup_{q \in P} \hat{\delta}(q, a)$.

Prema definiciji (9) vrijedi jednakost $\bigcup_{q \in P} \hat{\delta}(q, a) = \hat{\delta}(P, a) = \hat{\delta}\big(\hat{\delta}(q_0, w), a\big) = \hat{\delta}(q_0, wa)$ čime je dokazano da vrijedi $\delta'(q_0, wa) = \hat{\delta}(q_0, wa)$.

2) Dokazuje se da $\delta'(q_0, x)$ sadrži stanje iz skupa F' ako i samo ako $\hat{\delta}(q_0, x)$ sadrži stanje iz skupa F:

Po definiciji je $F' = F$, odnosno $F' = F \cup \{q_0\}$ ako $\varepsilon\text{-}okruženje(q_0)$ sadrži barem jedno stanje iz F. Zato je potrebno dokazati da za bilo koji niz x za koji je $q_0 \in \delta'(q_0, x)$ vrijedi da je $\varepsilon\text{-}okruženje(q_0) \subseteq \hat{\delta}(q_0, x)$.

Ova tvrdnja je istinita za prazan niz $x = \varepsilon$ jer vrijedi da je $\hat{\delta}(q_0, \varepsilon) = \varepsilon\text{-}okruženje(q_0)$.

Za niz $x = wa$, $|x| > 0$, $a \in \Sigma$ i $w \in \Sigma^*$, pretpostavimo da je $q_0 \in \delta'(q_0, x)$. Potrebno je dokazati da $\hat{\delta}(q_0, x)$ sadrži sva stanja iz $\varepsilon\text{-}okruženje(q_0)$.

Prethodno je dokazano da je $\delta'(q_0, x) = \hat{\delta}(q_0, x)$. Ako je $q_0 \in \delta'(q_0, x)$, onda je sigurno $q_0 \in \hat{\delta}(q_0, x)$. Budući da je $q_0 \in \hat{\delta}(q_0, x)$, a po definiciji $\hat{\delta}(q_0, x) = \varepsilon\text{-}okruženje\big(\delta(\hat{\delta}(q_0, w), a)\big)$ za svako stanje p iz $\hat{\delta}(q_0, x)$ uključuje i $\varepsilon\text{-}okruženje(p)$, onda je i $\varepsilon\text{-}okruženje(q_0) \subseteq \hat{\delta}(q_0, x)$.

6 KONAČNI AUTOMATI S IZLAZOM

6.1 MOORE AUTOMAT

- Ideja izlaza
- Formalna definicija Moore automata
- Primjer

Izlaz konačnog automata ograničen je binarnom funkcijom koja označava prihvaća li se niz ili ne. Funkcija izlaza proširuje se na dva osnovna načina:

- Izlazna se funkcija pridružuje stanju automata – Mooreov automat
- Izlazna se funkcija pridružuje stanju i ulaznom znaku – Mealyjev automat

Za bilo koji Mooreov automat moguće je izgraditi ekvivalentni Mealyjev automat i obrnuto. Dva su automata ekvivalentna ako za bilo koji ulazni niz daju jednake izlazne nizove.

Mooreov automat formalno se definira uređenom šestorkom:

$$MoDka = (Q, \Sigma, \Delta, \delta, \lambda, q_0) \tag{12}$$

gdje je:

Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
Δ	konačan skup izlaznih znakova
$\delta: Q \times \Sigma \to Q$	funkcija prijelaza
$\lambda: Q \to \Delta$	funkcija izlaza
$q_0 \in Q$	početno stanje

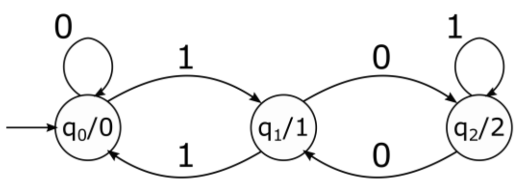
Funkcija izlaza $\lambda(q)$ stanju q pridružuje jedan od izlaznih znakova iz skupa Δ. Za prazan niz Mooreov automat daje izlaz $\lambda(q_0)$.

DKA $M' = (Q', \Sigma', \delta', q_0', F')$ je samo poseban slučaj Mooreovog automata, koji ima skup izlaznih znakova $\Delta = \{0, 1\}$.

Primjer Mooreovog automata prikazan u tablici 6.1 i na slici 6.1 na izlazu daje ostatak dijeljenja cijelog broja s brojem 3. Cijeli broj dolazi na ulaz u obliku binarnog niza.

	0	**1**	$\boldsymbol{\lambda}$
$\boldsymbol{q_0}$	q_0	q_1	0
$\boldsymbol{q_1}$	q_2	q_0	1
$\boldsymbol{q_2}$	q_1	q_2	2

Tablica 6.1 Tablica prijelaza i izlaza za Mooreov automat koji na izlazu daje ostatak dijeljenja ulaznog niza s tri



Slika 6.1 Dijagram stanja Mooreovog automata zadanog u tablici 6.1

6.2 MEALY AUTOMAT

- Formalna definicija Mealy automata
- Primjer

Mealyjev automat formalno se definira uređenom šestorkom

$$MeDka = (Q, \Sigma, \Delta, \delta, \lambda, q_0) \tag{13}$$

gdje je:

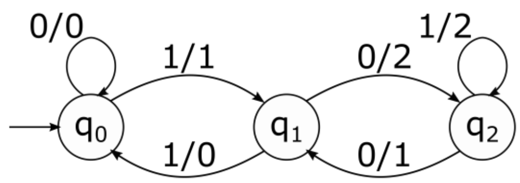
Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
Δ	konačan skup izlaznih znakova
$\delta: Q \times \Sigma \rightarrow Q$	funkcija prijelaza
$\lambda: Q \times \Sigma \rightarrow \Delta$	funkcija izlaza
$q_0 \in Q$	početno stanje

Izlazna funkcija Mealyjevog automata $\lambda(q, a)$ prijelazu $\delta(q, a)$ pridružuje jedan od izlaznih znakova iz skupa Δ. Za prazan niz Mealyjev automat ne daje nikakav izlaz.

Primjer Mealyjevog automata prikazan u tablici 6.2 i na slici 6.2 na izlazu daje ostatak dijeljenja cijelog broja s brojem 3. Cijeli broj dolazi na ulaz u obliku binarnog niza.

	$\boldsymbol{\delta}$		$\boldsymbol{\lambda}$	
	0	**1**	**0**	**1**
$\boldsymbol{q_0}$	q_0	q_1	0	1
$\boldsymbol{q_1}$	q_2	q_0	2	0
$\boldsymbol{q_2}$	q_1	q_2	1	2

Tablica 6.2 Tablica prijelaza i izlaza za Mealyjev automat koji na izlazu daje ostatak dijeljenja ulaznog niza s tri



Slika 6.2 Dijagram stanja Mealyjevog automata zadanog u tablici 6.2

6.3 KONSTRUKCIJA MEALY IZ ZADANOG MOORE DKA

- Definicija istovjetnosti
- Izgradnja funkcije izlaza
- Primjer

Istovjetnost Mealyjevog automata M' i Mooreovog automata M **definira se** na sljedeći način:

$$b\, T_{M'}(w) = T_M(w) \tag{14}$$

gdje je:

w	niz ulaznih znakova
b	izlaz Mooreovog automata za prazni niz $b = \lambda(q_0)$
$T_{M'}(w)$	izlazni niz Mealyjevog automata
$T_M(w)$	izlazni niz Mooreovog automata

Neka je zadan Mooreov automat $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$. Istovjetni Mealyjev automat $M' = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ gradi se **promjenom funkcije izlaza**:

$$\lambda'(q, a) = \lambda\big(\delta(q, a)\big), \forall q \in Q, \forall a \in \Sigma \tag{15}$$

Primjer izgradnje za Mooreov automat zadan dijagramom prijelaza na slici 6.1:

$$\begin{aligned}
\lambda'(q_0, 0) &= \lambda\big(\delta(q_0, 0)\big) = \lambda(q_0) = 0\\
\lambda'(q_0, 1) &= \lambda\big(\delta(q_0, 1)\big) = \lambda(q_1) = 1\\
\lambda'(q_1, 0) &= \lambda\big(\delta(q_1, 0)\big) = \lambda(q_2) = 2\\
\lambda'(q_1, 1) &= \lambda\big(\delta(q_1, 1)\big) = \lambda(q_0) = 0\\
\lambda'(q_2, 0) &= \lambda\big(\delta(q_2, 0)\big) = \lambda(q_1) = 1\\
\lambda'(q_2, 1) &= \lambda\big(\delta(q_2, 1)\big) = \lambda(q_2) = 2
\end{aligned}$$

Dobiveni Mealyjev automat prikazan je dijagramom prijelaza na slici 6.2.

6.4 KONSTRUKCIJA MOORE IZ ZADANOG MEALY DKA

- Definicija istovjetnosti
- Izgradnja Moore automata
- Primjer

Definicija istovjetnosti dana je u pitanju 6.3.

Neka je zadan Mealyjev automat $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$. Istovjetni Mooreov automat $M' = (Q', \Sigma, \Delta, \delta', \lambda', q_0')$ **gradi se** na sljedeći način:

1) $Q' = Q \times \Delta = \{[q, b] \mid q \in Q,\ b \in \Delta\}$
2) $q_0' = [q_0, b_0]$, $b_0 \in \Delta$ je proizvoljan
3) $\delta'([q, b], a) = [\delta(q, a),\ \lambda(q, a)]$, $q \in Q, b \in \Delta, a \in \Sigma$
4) $\lambda'([q, b]) = b$, $q \in Q, b \in \Delta$

Primjer izgradnje za Mealyjev automat zadan dijagramom prijelaza na slici 6.2 iz kojega se dobije ekvivalentni Mooreov automat prikazan na slici 6.1:

1) $Q' = \{[q_0, 0], [q_0, 1], [q_0, 2], [q_1, 0], [q_1, 1], [q_1, 2], [q_2, 0], [q_2, 1], [q_2, 2]\}$
2) $q_0' = [q_0, 0]$
3) Funkcija prijelaza δ' je:

$$\begin{aligned}
\delta'([q_0, 0], 0) &= [\delta(q_0, 0), \lambda(q_0, 0)] = [q_0, 0]\\
\delta'([q_0, 0], 1) &= [\delta(q_0, 1), \lambda(q_0, 1)] = [q_1, 1]\\
&\vdots\\
\delta'([q_2, 2], 1) &= [\delta(q_2, 1), \lambda(q_2, 1)] = [q_2, 2]
\end{aligned}$$

4) Funkcija izlaza λ' je:

$$\begin{array}{lll}
\lambda'([q_0, 0]) = 0 & \lambda'([q_1, 0]) = 0 & \lambda'([q_2, 0]) = 0\\
\lambda'([q_0, 1]) = 1 & \lambda'([q_1, 1]) = 1 & \lambda'([q_2, 1]) = 1\\
\lambda'([q_0, 2]) = 2 & \lambda'([q_1, 2]) = 2 & \lambda'([q_2, 2]) = 2
\end{array}$$

7 REGULARNI JEZICI I IZRAZI

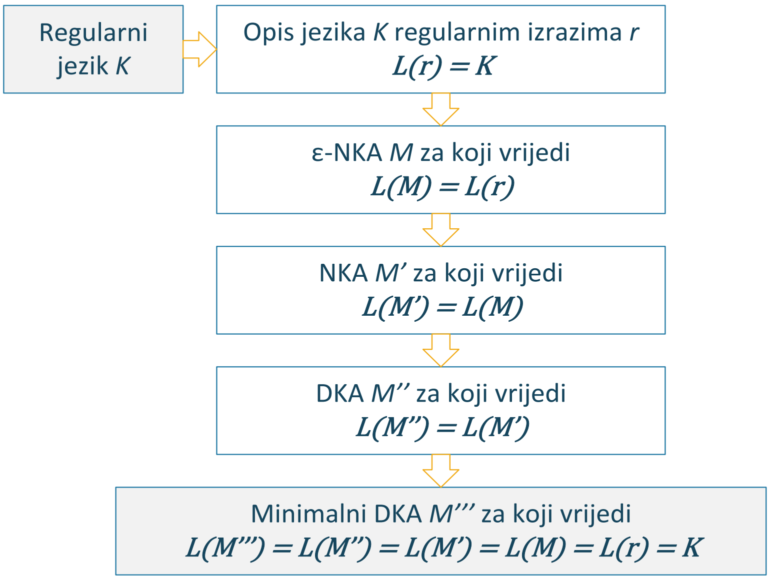
7.1 DEFINICIJA REGULARNIH IZRAZA

- Regularni jezik i izraz
- Algoritam sinteze DKA

Regularni izrazi koriste se za opisivanje regularnih jezika.

Jezik je regularan ako ga je moguće opisati regularnim izrazima, odnosno ako postoji konačni automat koji ga prihvaća. Za bilo koji jezik $L(r)$ zadan regularnim izrazima r moguće je izgraditi DKA M za koji vrijedi $L(M) = L(r)$.

Algoritam izgradnje DKA na temelju zadanih regularnih izraza prikazan je na slici 7.1.



Slika 7.1 Algoritam izgradnje DKA na temelju regularnih izraza

7.2 REKURZIVNA PRAVILA ZA RI

- Navesti rekurzivna pravila
- Pravila asocijativnosti i prednosti
- Istovjetnost i svojstva RI

Rekurzivna pravila regularnih izraza nad abecedom Σ su:

1) $\emptyset$ je RI i označava jezik $L(\emptyset) = \{\}$
2) ε je RI i označava jezik $L(\varepsilon) = \{\varepsilon\}$
3) $\forall a \in \Sigma$, a je RI i označava jezik $L(a) = \{a\}$
4) Ako su r i s regularni izrazi koji označavaju jezike $L(r)$ i $L(s)$, onda vrijedi:
   a) $(r) \vee (s)$ je regularni izraz koji označava jezik $L\big((r) \vee (s)\big) = L(r) \cup L(s)$.
   b) $(r)(s)$ je regularni izraz koji označava jezik $L\big((r)(s)\big) = L(r)L(s)$.
   c) $(r)^*$ je regularni izraz koji označava jezik $L((r)^*) = L(r)^*$

Kleene operator (*), nadovezivanje i operator ∨ su lijevo **asocijativni**.

Najveću **prednost** ima *, zatim nadovezivanje i na kraju ∨.

Dva regularna izraza r i s su **istovjetna** ako označavaju iste jezike: $L(r) = L(s)$. Istovjetnost RI r i s označava se s $r = s$.

Svojstva RI su:

Komutativnost ∨	$r \vee s = s \vee r$
Asocijativnost ∨	$r \vee (s \vee t) = (r \vee s) \vee t$
Asocijativnost nadovezivanja	$(rs)t = r(st)$
Distributivnost nadovezivanja nad ∨	$r(s \vee t) = rs \vee rt$ ili $(s \vee t)r = sr \vee t$
Neutralni element ε	$\varepsilon r = r\varepsilon = r$
Relacija između ∨ i *	$r^* = (r \vee \varepsilon)^*$
Idempotentnost	$r^{**} = r^*$

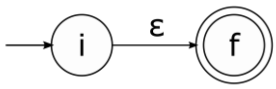
7.3 KONSTRUKCIJA ε-NKA IZ RI

- Pravila elementarnih automata

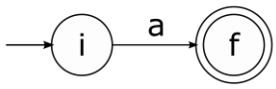
Za bilo koji regularni izraz r moguće je izgraditi ε-NKA M tako da vrijedi $L(M) = L(r)$:

1) **Za RI** ∅ koji definira jezik $L(\emptyset) = \{\}$ konstruira se ε-NKA $M = (\{i, f\}, \Sigma, \{\}, i, \{f\})$ prikazan na slici 7.2. Ovaj automat ne prihvaća niti jedan niz, čak ni prazni niz ε.
2) **Za RI $\boldsymbol{\varepsilon}$** koji definira jezik $L(\varepsilon) = \{\varepsilon\}$ konstruira se ε-NKA $M = (\{i, f\}, \Sigma, \{\delta(i, \varepsilon) = f\}, i, \{f\})$ prikazan na slici 7.3. Ovaj automat prihvaća samo prazni niz ε.
3) **Za RI $\boldsymbol{a}$** koji definira jezik $L(a) = \{a\}$ konstruira se ε-NKA $M = (\{i, f\}, \Sigma, \{\delta(i, a) = f\}, i, \{f\})$ prikazan na slici 7.4. Ovaj automat prihvaća samo niz a.

Slika 7.2 Elementarni automat za RI ∅

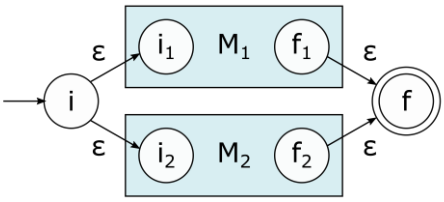


Slika 7.3 Elementarni automat za RI ε

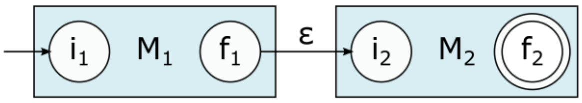


Slika 7.4 Elementarni automat za RI a

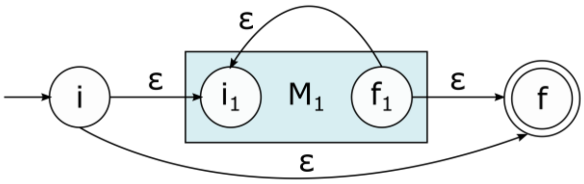
4) **Za RI $\boldsymbol{r_1 \vee r_2}$** koji definira jezik $L(r_1 \vee r_2) = L(r_1) \cup L(r_2)$ prvo pretpostavimo da su prethodno izgrađeni ε-NKA M_1 i M_2 takvi da vrijedi $L(M_1) = L(r_1)$, odnosno $L(M_2) = L(r_2)$. Automat M za kojeg vrijedi $L(M) = L(r_1 \vee r_2)$ konstruira se kao što je prikazano na slici 7.5.
5) **Za RI $\boldsymbol{r_1 r_2}$** koji definira jezik $L(r_1 r_2) = L(r_1)L(r_2)$ također prvo pretpostavimo da postoje M_1 i M_2 takvi da vrijedi $L(M_1) = L(r_1)$, odnosno $L(M_2) = L(r_2)$. Automat M za kojeg vrijedi $L(M) = L(r_1 r_2)$ konstruira se kao što je prikazano na slici 7.6.
6) **Za RI $\boldsymbol{r_1^*}$** koji definira jezik $L(r_1^*) = L(r_1)^*$ pretpostavimo da je prethodno izgrađen automat M_1 za koji vrijedi $L(M_1) = L(r_1)$. Automat M za koji vrijedi $L(M) = L(r_1^*)$ gradi se kao što je prikazano na slici 7.7.
7) **Za RI $(\boldsymbol{r})$** uzima se ε-NKA M regularnog izraza r jer vrijedi da je $L\big((r)\big) = L(r)$.



Slika 7.5 Elementarni automat za RI $r_1 \vee r_2$



Slika 7.6 Elementarni automat za RI $r_1 r_2$



Slika 7.7 Elementarni automat za RI r_1^*

7.4 GENERATOR KONAČNOG AUTOMATA

- Generator i simulator
- Struktura ostvarenja

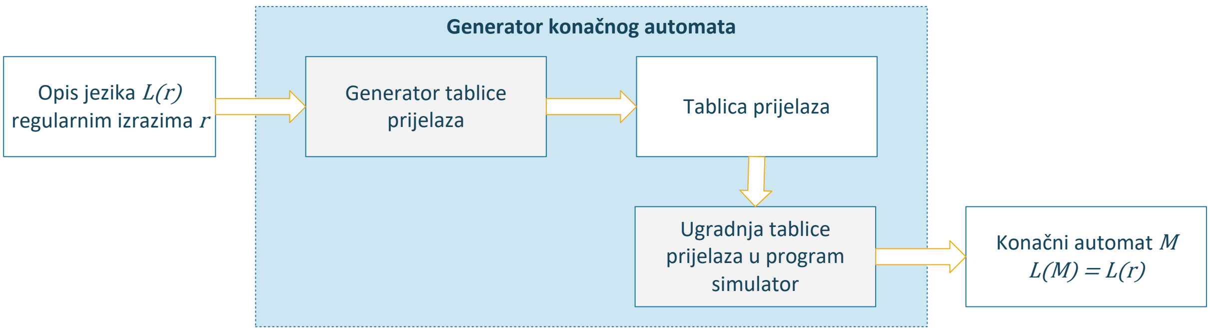
Generator konačnog automata za jezik zadan regularnim izrazima gradi konačni automat (slika 7.8). Generator ostvaruje cjelokupnu ili dio pretvorbe regularnih izraza u DKA.

Slika 7.8 Generator konačnog automata

Simulator konačnog automata (slika 7.9) je program koji slijedno čita ulazne znakove i na temelju njih i tablice prijelaza računa prijelaz u novo stanje.

Slika 7.9 Program simulator konačnog automata

Generator konačnog automata gradi tablicu prijelaza na temelju zadanih regularnih izraza. Tablica prijelaza ugradi se u program simulator te se na temelju nje generira konačni automat (**slika 7.10**).



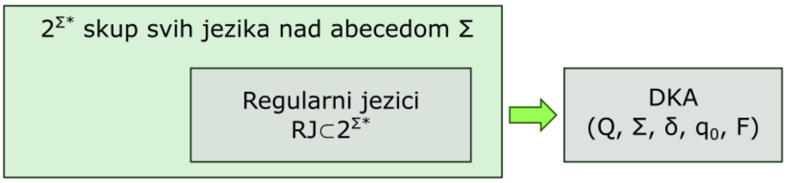
Slika 7.10 Struktura generatora konačnog automata

8 SVOJSTVA REGULARNIH JEZIKA

8.1 KLASE JEZIKA

- Skup svih jezika
- Položaj regularnih jezika
- Zatvorenost klase jezika
- Unija, nadovezivanje, Kleene

Skup svih jezika L definiranih nad abecedom Σ označavamo kao 2^{Σ^*}.



Slika 8.1 Regularni jezici i DKA

Klasa regularnih jezika nad abecedom Σ pravi je podskup svih jezika 2^{Σ^*} (slika 8.1).

Klasa jezika zatvorena je s obzirom na neku operaciju ako primjenom te operacije na bilo koji jezik iz te klase dobijemo jezik koji je u toj istoj klasi.

Regularni jezici zatvoreni su s obzirom na operacije:

- **Unija** – $L \cup N = \{a \mid a \in L \ \vee a \in N\}$
- **Nadovezivanje** – $LN = \{ab \mid a \in L \wedge b \in N\}$
- **Kleene operator** – $L^* = \bigcup_{i=0}^{\infty} L^i$

Svojstva zatvorenosti slijede neposredno iz definicije regularnih izraza.

8.2 ZATVORENOST REGULARNIH JEZIKA

- Zatvorenost s obzirom na komplement
- Zatvorenost s obzirom na presjek

Regularni jezici **zatvoreni su s obzirom na operaciju komplementa**. Neka DKA $M = (Q, \Sigma, \delta, q_0, F)$ prihvaća regularni jezik $L(M)$. Za komplement jezika $L(M)^c$ moguće je izgraditi DKA $M' = (Q, \Sigma, \delta, q_0, Q \backslash F)$ koji prihvaća jezik:

$$L(M') = \{w \mid \delta(q_0, w) \in (Q \backslash F)\} = \{w | \delta(q_0, w) \notin F\} = \Sigma^* \backslash \{w | \delta(q_0, w) \in F\} = \Sigma^* \backslash L(M) = L(M)^c$$

Regularni jezici **zatvoreni su s obzirom na operaciju presjeka**. Ovo svojstvo proizlazi izravno iz svojstva zatvorenosti s obzirom na uniju i komplement, što se dokazuje De Morganovim pravilom:

$$L \cap N = ((L \cap N)^c)^c = (L^c \cup N^c)^c$$

8.3 REGULARNE DEFINICIJE

- Zatvorenost s obzirom na supstituciju
- Regularne definicije

Regularni jezici **zatvoreni su s obzirom na operaciju supstitucije**. Neka je $R \subseteq \Sigma^*$ regularni jezik i neka se znaku a iz skupa Σ pridruži regularni jezik $R_a \subseteq \Delta^*$, gdje je Δ određena abeceda. Zamijenimo niz $a_1 a_2 \dots a_n$, ($a_i \in \Sigma$) regularnog jezika R nizom $w_1 w_2 \dots w_n$, ($w_i \in R_{a_i}$). Dobiveni jezik $f(R)$ je regularan. Svojstvo zatvorenosti s obzirom na supstituciju dokazuje se opisivanjem jezika R i R_{a_i} regularnim izrazima.

Svojstvo supstitucije omogućava pojednostavljeno zapisivanje regularnih definicija.

Regularne definicije su oblika:

$$\begin{gathered} d_1 \to r_1 \\ d_2 \to r_2 \\ \vdots \\ d_n \to r_n \end{gathered}$$

gdje su d_i znakovi označeni različitim nazivima, a r_i su regularni izrazi nad abecedom $\Sigma \cup \{d_1, d_2, \dots, d_{i-1}\}$.

9 SVOJSTVO NAPUHAVANJA

9.1 DEFINICIJA SVOJSTVA NAPUHAVANJA

- Problem dugačkih nizova
- Ponavljanje stanja
- Prihvaćanje dugačkog niza

Svojstvo napuhavanja pogodno je za dokazivanje neregularnosti jezika, ispravnosti algoritama kojima se utvrđuje nepraznost ili beskonačnost regularnog jezika itd.

Ako je jezik regularan, onda postoji DKA M koji ga prihvaća. Neka M ima n stanja. Promotrimo ulazni niz $a_1 a_2 \dots a_m$ koji je duži od broja stanja ($m > n$). Budući da je ulaznih znakova više nego stanja, **ponavljat** će se barem jedno stanje u slijedu prijelaza za niz $a_1 a_2 \dots a_m$.

Bilo koji niz $z \in L(M)$ može se rastaviti na podnizove $z = uvw$. S obzirom da za niz uw automat dolazi u isto stanje kao za niz z, **prihvaća se** i niz uw. Isto vrijedi i za podniz $uvvw$. Podniz v može se ponoviti proizvoljan broj puta jer vrijedi $uv^i w \in L(M)$, $i \geq 0$.

Ako regularni jezik sadrži dovoljno dugačak niz $z = uvw$, taj jezik sadrži i beskonačni skup nizova oblika $uv^i w$.

9.2 DOKAZ NEREGULARNOSTI

- Pristup dokazu neregularnosti
- Primjer
- Nepraznost i beskonačnost regularnog jezika

Za **dokazivanje neregularnosti** koristi se pravilo napuhavanja (*pumping lemma*).

Ako je L regularan jezik, postoji cijeli broj n takav da je bilo koji niz z iz jezika L za koji vrijedi $|z| > n$ moguće rastaviti na podnizove $z = uvw$ tako da vrijedi:

1) $|v| \geq 1$
2) $|uv| \leq n$
3) $uv^i w \in L$ za bilo koji $i \geq 0$

Za **primjer** pokažimo da jezik $K = \{0^{l^2} \mid l \in \mathbb{N}\}$ nije regularan. Jezik K sadrži nizove znakova 0 čija je duljina kvadrat nekog broja.

- Pretpostavimo da je K regularan.
- Neka je n proizvoljan cijeli broj i neka je $z = 0^{n^2}$ niz jezika K za koji vrijedi $|z| = n^2$ i $|z| > n$.
- Prema pravilu napuhavanja niz z rastavlja se na podnizove uvw gdje je $1 \leq |v| \leq |uv| \leq n$. Potrebno je utvrditi je li niz $uv^i w$ element jezika K za bilo koji i.
- Uzmimo da je $i = 2$. Tada je $|uvw| = |z| = n^2 < |uv^2 w| = (n^2 + |v|) \leq (n^2 + n)$.
- Budući da je $(n^2 + n) < (n + 1)^2$, vrijedi $n^2 < |uv^2 w| < (n + 1)^2$, tj. duljina niza $uv^2 w$ nije kvadrat niti jednog cijelog broja.
- Time je dokazano da $uv^2 w$ nije element jezika K te stoga taj jezik nije regularan.

Regularni jezik $L(M)$ kojeg prihvaća DKA M s n stanja je **neprazan** ako i samo ako M prihvaća niz z duljine manje od n. Dakle, $L(M)$ je neprazan ako je u skupu dohvatljivih stanja barem jedno prihvatljivo.

Regularni jezik $L(M)$ je **beskonačan** ako i samo ako odgovarajući DKA M s n stanja prihvaća niz duljine l, $n \leq l \leq 2n$. Drugim riječima, $L(M)$ je beskonačan ako njegov dijagram stanja ima barem jednu zatvorenu petlju kad se izuzmu sva neprihvatljiva stanja koja nemaju nijedan slijed prijelaza u prihvatljivo stanje.

10 REGULARNA GRAMATIKA

10.1 KONTEKSTNO NEOVISNA GRAMATIKA

- Formalna definicija gramatike
- Oznake u formalnoj gramatici
- Kontekstno neovisna gramatika
- Relacija primjene produkcija

Formalna gramatika je skup pravila za generiranje i analizu nizova znakova formalnog jezika zamjenom nezavršnih znakova završnim primjenom produkcija. Primjena pravila gramatike označava se znakom $\Rightarrow$, dok se višestruka primjena pravila označava s $\overset{*}{\Rightarrow}$.

Oznake koje se koriste u formalnoj gramatici:

- A, B, C, D, E, S – nezavršni znakovi
- $a, b, c, \dots, 0, 1, 2, \dots$ – završni znakovi
- X, Y, Z – završni ili nezavršni znakovi
- u, v, w, x, y, z – nizovi završnih znakova
- α, β, γ – nizovi završnih i nezavršnih znakova

Kontekstno neovisna gramatika definira se uređenom četvorkom

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

gdje je:

V	konačan skup nezavršnih znakova
T	konačan skup završnih znakova, $V \cap T = \emptyset$
P	konačan skup produkcija oblika $A \to \alpha$, $A \in V$, $\alpha \in (V \cup T \cup \{\varepsilon\})^*$
S	početni nezavršni znak

Za zadanu gramatiku G definira se **relacija** $\underset{G}{\Rightarrow}$ nad nizovima iz skupa $(V \cup T)^*$. Ako je $A \to \beta$ produkcija skupa P i ako su α i γ iz $(V \cup T)^*$, onda vrijedi relacija:

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

Niz $\alpha\beta\gamma$ generira se neposredno iz niza $\alpha A \gamma$ primjenom produkcije $A \to \beta$, dok oznaka G određuje kojoj gramatici pripada primijenjena produkcija.

10.2 FORMALNA GRAMATIKA I JEZICI

- Generiranje jezika
- Kontekstno neovisni jezici
- Generativno stablo

Gramatika $G = (V, T, P, S)$ **generira jezik** $L(G) = \{w \mid w \in T^* \wedge S \overset{*}{\underset{G}{\Rightarrow}} w\}$.

Niz w je u jeziku $L(G)$ koji generira gramatika G ako su u tom nizu isključivo završni znakovi gramatike i ako ga je moguće generirati iz početnog nezavršnog znaka S.

Gramatike G_1 i G_2 su istovjetne ako generiraju iste jezike: $L(G_1) = L(G_2)$.

Kontekstno neovisna gramatika generira klasu **kontekstno neovisnih jezika** koja obuhvaća i regularne jezike ($RJ \subset KNJ$).

Stablo je generativno za gramatiku $G = (V, T, P, S)$ ako vrijedi:

1) Čvorovi stabla označeni su znakovima iz $V \cup T \cup \{\varepsilon\}$.
2) Korijen stabla označen je početnim nezavršnim znakom S.
3) Unutrašnji čvorovi označeni su nezavršnim znakovima $A \in V$.
4) Za čvor A i njegovu djecu $X_1, X_2, \dots, X_n$ vrijedi produkcija $A \to X_1 X_2 \dots X_n$.
5) Znakom ε označeni su isključivo listovi stabla. List označen s ε jedino je dijete svog roditelja.
6) Listovi stabla označeni su znakovima iz skupa $T \cup \{\varepsilon\}$ i čitani s lijeva na desno čine generirani niz jezika $L(G)$.

Neka gramatika G generira niz završnih znakova w. Relacija $S \overset{*}{\underset{G}{\Rightarrow}} w$ vrijedi ako i samo ako je moguće za gramatiku G izgraditi generativno stablo čiji su listovi isključivo označeni članovima niza w i znakom ε.

10.3 REGULARNA GRAMATIKA

- Definicija
- Konstrukcija regularne gramatike iz DKA
- Odnos regularne gramatike i DKA

Regularna gramatika generira regularne jezike te je ujedno i kontekstno neovisna. Konstrukcijom gramatike za regularni jezik zadan DKA-om, dokazujemo da je gramatika regularna.

Konstrukcija gramatike $G = (V, T, P, S)$ za regularni jezik kojeg prihvaća DKA $M = (Q, \Sigma, \delta, q_0, F)$, za koju vrijedi $L(G) = L(M)$ provodi se na sljedeći način:

1) $T = \Sigma$, tj. skup završnih znakova jednak je skupu ulaznih znakova DKA.
2) $V = Q$, tj. skup nezavršnih znakova jednak je skupu stanja DKA.
3) $S = q_0$, tj. početni nezavršni znak jednak je početnom stanju DKA.
4) Na temelju prijelaza DKA $\delta(A, a) = B$ gradi se produkcija $A \to aB$, $A, B \in V$, $a \in T$
5) Za sva prihvatljiva stanja $A \in F$ grade se produkcije $A \to \varepsilon$, $A \in V$ i ε označava prazan niz.

Ako DKA prihvaća isti jezik koji generira gramatika, DKA i gramatika su **istovjetni**:

$$A \overset{*}{\Rightarrow} wB \iff \delta(A, w) = B \qquad (16)$$

Neka je $C = \delta(q_0, v) \in F$, odnosno neka DKA prihvaća niz v. Na temelju (16) vrijedi da je $q_0 \overset{*}{\Rightarrow} vC$. Budući da je $C \in F$, postoji produkcija $C \to \varepsilon$ i vrijedi relacija $q_0 \overset{*}{\Rightarrow} vC \Rightarrow v$. Dakle, ako DKA prihvaća niz v, onda ga izgrađena gramatika generira.

Neka gramatika generira niz v i neka je $q_0 \overset{*}{\Rightarrow} vC \Rightarrow v$ za neki nezavršni znak/stanje C. U generiranju niza v primjenjuje se produkcija $C \to \varepsilon$. Na temelju (16) vrijedi da je $\delta(q_0, v) = C$. Produkcija $C \to \varepsilon$

gradi se ako i samo ako je stanje C prihvatljivo. Budući da je C prihvatljivo, DKA prihvaća niz v. Dakle, ako izgrađena gramatika generira niz v, onda ga DKA prihvaća.

Time je dokazana istovjetnost zadanog DKA i konstruirane gramatike.

10.4 SINTEZA NKA IZ REGULARNE GRAMATIKE

- Istovjetnost regularne gramatike i DKA
- Konstrukcija NKA iz jednostavne gramatike
- Izvor nedeterminiranosti

Istovjetnost regularne gramatike i DKA objašnjena je u prethodnom pitanju.

Konstrukcija NKA $M = (Q, \Sigma, \delta, q_0, F)$ na temelju gramatike $G = (V, T, P, S)$ s produkcijama oblika $A \to aB$ i $A \to \varepsilon$ tako da vrijedi $L(M) = L(G)$ provodi se na sljedeći način:

1) $\Sigma = T$, tj. skup ulaznih znakova NKA jednak je skupu završnih znakova gramatike.
2) $Q = V$, tj. skup stanja NKA jednak je skupu nezavršnih znakova gramatike.
3) $q_0 = S$, tj. početno stanje NKA jednako je početnom nezavršnom znaku gramatike.
4) Na temelju produkcije $A \to aB$ gradi se prijelaz $\delta(A, a) = \delta(A, a) \cup \{B\}$ jer je moguće da više produkcija ima isti nezavršni znak na lijevoj i isti završni znak na desnoj, ali različite nezavršne znakove na desnoj strani (**razlog nedeterminiranosti**).
5) Ako je u gramatici produkcija $A \to \varepsilon$, onda je A prihvatljivo stanje.

10.5 DESNO LINEARNA GRAMATIKA

- Definicija desno linearne gramatike
- Konstrukcija NKA iz DLG

Gramatika je desno linearna ako svaka produkcija gramatike ima najviše jedan nezavršni znak na krajnje desnom mjestu desne strane: $A \to wB$ ili $A \to w$, $A, B \in V$, $w \in T^*$.

Konstrukcija NKA iz DLG obavlja se tako da se složene produkcije preuređuju dok ne dobiju oblik $A \to aB$ ili $A \to \varepsilon$:

1) Produkcije oblika $A \to w$ ($w \in T^*, w \neq \varepsilon$) zamijene se novim produkcijama oblika $A \to w[\varepsilon]$ i doda se produkcija $[\varepsilon] \to \varepsilon$, gdje je $[\varepsilon]$ novi nezavršni znak.
2) Produkcije oblika $A \to a_1 \dots a_n B$ zamijene se produkcijama oblika:

$$\begin{gathered} A \to a_1[a_2 \dots a_n B] \\ [a_2 \dots a_n B] \to a_2[a_3 \dots a_n B] \\ \vdots \\ [a_{n-1} a_n B] \to a_{n-1}[a_n B] \\ [a_n B] \to a_n B \end{gathered}$$

3) Ako je nezavršni znak B jedini znak desne strane produkcije ($A \to B$), onda se izuzmu sve produkcije koje imaju istu lijevu i desnu stranu ($B \to B$). Ako ostane produkcija koje imaju različite lijeve i desne strane $A \to B$, one se zamijene produkcijama $A \to y$ na temelju produkcija $B \to y$.

10.6 LIJEVO LINEARNA GRAMATIKA

- Definicija lijevo linearne gramatike
- Konstrukcija ε-NKA iz LLG

Gramatika je lijevo linearna ako svaka njena produkcija na krajnjem lijevom mjestu desne strane ima najviše jedan nezavršni znak: $A \to Bw$ ili $A \to w$, $A, B \in V$, $w \in T^*$.

Konstrukcija ε-NKA M' koji prihvaća jezik $L(M') = L(G)$, gdje je $G = (V, T, P, S)$ zadana LLG, obavlja se na sljedeći način:

1) Iz G se konstruira desno linearna gramatika $G' = (V, T, P', S)$ tako što se skup produkcija P preuredi obrtanjem redoslijeda desnih strana produkcija:

$$P' = \{ A \to \alpha^R \mid (A \to \alpha) \in P\} \quad (17)$$

   Tada vrijedi $L(G') = L(G)^R$.
2) Na temelju desno linearne gramatike G' konstruira se NKA M koji prihvaća jezik:

$$L(M) = L(G') = L(G)^R$$

3) Na temelju NKA M izgradi se ε-NKA M' koji prihvaća jezik $L(M') = L(M)^R = L(G')^R = L(G)$.
   a) M' se preuredi tako da ima samo jedno prihvatljivo stanje definiranjem ε-prijelaza iz trenutnih prihvatljivih stanja u novo, jedinstveno prihvatljivo stanje.
   b) Početno stanje od M postaje prihvatljivo stanje od M', a prihvatljivo stanje od M postaje početno od M'.
   c) Funkcije prijelaza ε-NKA M' grade se obrtanjem smjera usmjerenih grana u dijagramu stanja.

11 JEDNOZNAČNOST GRAMATIKE, JEZIKA I NIZA

11.1 NEJEDNOZNAČNOST NIZA

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

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

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

Promjenom redoslijeda **primjene produkcija** na nezavršne znakove dobivamo različita stabla koja određuju različita grupiranja završnih znakova. Stoga nastojimo konstruirati gramatiku koja za zadani niz gradi samo jedno stablo.

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

1) Zamjena krajnje lijevog nezavršnog znaka – produkcije se primjenjuju isključivo na krajnje lijeve nezavršne znakove.
2) Zamjena krajnje desnog nezavršnog znaka – produkcije se primjenjuju isključivo na krajnje desne nezavršne znakove.

11.2 SISTEMATIZACIJA ZAMJENE

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

Redoslijed **nezavršnih znakova** na koje se primjenjuju produkcije značajan je za postupak generiranja niza pa se koriste dva postupka:

1) Zamjena krajnje lijevog nezavršnog znaka – produkcije se primjenjuju isključivo na krajnje lijeve nezavršne znakove.
2) Zamjena krajnje desnog nezavršnog znaka – produkcije se primjenjuju isključivo na krajnje desne nezavršne znakove.

Postupak **obilaska stabla** određuje redoslijed kojim se obilaze grane i čvorovi. Stablo se može obilaziti (počevši od korijena):

- Desnim obilaskom – rekurzivno se obilaze sve neobiđene desne grane i čvorovi.
- Lijevim obilaskom – rekurzivno se obilaze sve neobiđene lijeve grane i čvorovi.

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

Kontekstno neovisna **gramatika $\boldsymbol{G}$ je nejednoznačna** ako je moguće za neki niz $w \in L(G)$ izgraditi više različitih generativnih stabala.

Niz $\boldsymbol{w} \in \boldsymbol{L}(\boldsymbol{G})$ je nejednoznačan za zadanu gramatiku G ako je za njega moguće izgraditi više različitih generativnih stabala.

Jezik $\boldsymbol{L}$ je nejednoznačan ako ga nije moguće generirati niti jednom jednoznačnom gramatikom G.

Nejednoznačnost jezika $\boldsymbol{L}$ razrješava se zamjenom nejednoznačne gramatike G jednoznačnom gramatikom G' ili promjenom jezika L u novi jezik L' koji se generira jednoznačnom gramatikom G'. Razlika je u tome što se promjenom gramatike ne mijenja jezik i odbacuje se višestruko značenje niza, dok se promjenom jezika definira zaseban niz za svako značenje i čuva se višestruko značenje niza.

11.3 PROMJENA GRAMATIKE

- Definicija promjene gramatike
- Primjer
- Svojstva promjene gramatike

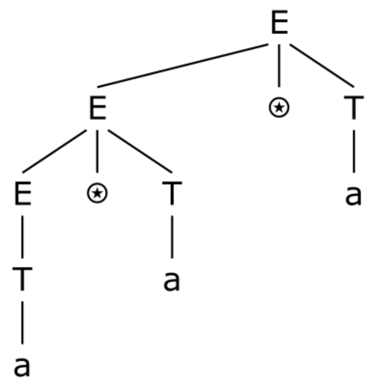
Jezik L, koji generira nejednoznačna gramatika, moguće je generirati primjenom više različitih jednoznačnih gramatika.

Za primjer uzmimo jezik L koji generira gramatika $G = (\{E\}, \{a, \circledast\}, \{E \to E \circledast E \mid a\}, E)$. Gramatika G je nejednoznačna.

Za lijevo asocijativni operator $\circledast$ gradi se jednoznačna gramatika $G' = (\{E, T\}, \{a, \circledast\}, \{E \to E \circledast T \mid T, T \to a\}, E)$. Za ovu gramatiku, niz $a \circledast a \circledast a$ je jednoznačan (slika 11.1).

Za desno asocijativni operator $\circledast$ postupak je sličan. Jednoznačna gramatika je $G' = (\{E, T\}, \{a, \circledast\}, \{E \to T \circledast E \mid T, T \to a\}, E)$.

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



Slika 11.1 Generativno stablo za lijevo asocijativni niz $a \circledast a \circledast a$ i gramatiku G'

11.4 PROMJENA JEZIKA

- Definicija promjene jezika
- Primjer
- Svojstva promjene jezika

Umjesto nejednoznačnog jezika L, gradi se jednoznačan jezik L' kojeg je moguće generirati jednoznačnom gramatikom. Promjena jezika provodi se iz tri razloga:

1) Kada je jezik inherentno nejednoznačan.
2) Kada je jednoznačna gramatika previše složena.
3) Kada se žele sačuvati sve interpretacije nizova.

Uzmimo **primjer** niza $a \circledast a \circledast a$. Za gramatiku $G = (\{E\}, \{a, \circledast\}, \{E \to E \circledast E \mid a\}, E)$ moguće su dvije interpretacije tog niza – lijevo i desno asocijativni. Dodavanjem zagrada u niz moguće je definirati redoslijed primjena operatora ⍟ i izbjeći jednoznačnost. U novom su jeziku umjesto jednog niza dva nova niza: $((a)\circledast(a))\circledast(a)$ i $(a)\circledast((a)\circledast(a))$.

Neka gramatika $G' = (\{E\},\ \{a,\ \circledast,\ (,\ )\},\ \{E \rightarrow (E)\ \circledast(E) \mid a\},\ E)$ generira novi jezik. Za zadanu gramatiku G' oba niza $((a)\circledast(a))\circledast(a)$ i $(a)\circledast((a)\circledast(a))$ su jednoznačna.

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

12 POJEDNOSTAVLJENJE GRAMATIKE

12.1 DEFINICIJA POJEDNOSTAVLJENJA GRAMATIKE

- Svrha pojednostavljenja
- Željena svojstva gramatike
- Definicija beskorisnosti znaka

Svrha pojednostavljenja gramatike je odbacivanje beskorisnih znakova i produkcija.

Za bilo koji neprazni kontekstno neovisni jezik L moguće je izgraditi kontekstno neovisnu gramatiku G sa **svojstvima**:

1) Bilo koji znak gramatike G koristi se u postupku generiranja barem jednog niza jezika L.
2) Gramatika G nema produkcija oblika $A \rightarrow B$ (jedinične produkcije), gdje su A i B nezavršni znakovi.
3) Ako prazni niz ε nije element jezika L, onda je moguće izbjeći korištenje produkcija oblika $A \rightarrow \varepsilon$. Takve produkcije nazivaju se ε-produkcije.

Znak X gramatike $G = (V, T, P, S)$ je **beskoristan** ako se ne koristi u postupku generiranja:

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

gdje su $\alpha, \beta \in (V \cup T \cup \{\varepsilon\})^*$, a $w \in T^*$.

Dva su vida beskorisnosti:

1) Znak X je **mrtav** – iz njega nije moguće generirati niz završnih znakova.
2) Znak X je **nedohvatljiv** – nije niti u jednom nizu koji se generira iz početnog nezavršnog znaka.

12.2 ODBACIVANJE BESKORISNIH ZNAKOVA

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

Za bilo koju kontekstno neovisnu gramatiku $G = (V, T, P, S)$ koja generira neprazni jezik moguće je izgraditi istovjetnu gramatiku $G' = (V', T', P', S)$ koja nema **mrtvih** znakova:

$$A \overset{*}{\Rightarrow} w, \forall A \in V', w \in T'^*$$

Algoritam odbacivanja mrtvih znakova temelji se na pronalaženju živih znakova i provodi se u tri koraka:

1) U listu živih znakova stave se lijeve strane produkcija koje na desnoj strani nemaju nezavršnih znakova.
2) Ako su na desnoj strani produkcije samo živi i završni znakovi, nezavršni znak s lijeve strane stavi se u listu živih znakova.
3) Ako nije moguće proširiti listu živih znakova, algoritam se zaustavlja i svi znakovi koji nisu na listi živih znakova su mrtvi.

Za bilo koju kontekstno neovisnu gramatiku $G = (V, T, P, S)$ koja generira neprazni jezik moguće je izgraditi istovjetnu gramatiku $G' = (V', T', P', S)$ koja nema **nedohvatljivih** znakova

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

Algoritam odbacivanja nedohvatljivih znakova temelji se na pronalaženju dohvatljivih znakova i provodi se u tri koraka:

1) U listu dohvatljivih znakova stavi se početni nezavršni znak S.
2) Ako je znak s lijeve strane produkcije u listi dohvatljivih znakova, onda se svi znakovi s desne strane dodaju u listu dohvatljivih znakova.
3) Ako listu dohvatljivih znakova nije moguće proširiti, algoritam se zaustavlja. Svi znakovi koji nisu u listi dohvatljivih znakova su nedohvatljivi.

Primjenom algoritma odbacivanja mrtvih, a zatim algoritma odbacivanja nedohvatljivih znakova, iz gramatike se **odbacuju svi beskorisni znakovi**. Primjena algoritama odbacivanja obrnutim redoslijedom neće nužno odbaciti sve beskorisne znakove.

12.3 ODBACIVANJE E I JEDINIČNIH PRODUKCIJA

- Odbacivanje ε-produkcija
- Odbacivanje jediničnih produkcija

Za svaku kontekstno neovisnu gramatiku G koja generira jezik $L(G) \backslash \{\varepsilon\}$ moguće je izgraditi istovjetnu gramatiku G' koja nema ε-produkcija.

Algoritam odbacivanja ε-produkcija odvija se u dva koraka:

1) Pronađu se svi nezavršni znakovi koji generiraju prazni niz ($A \overset{*}{\Rightarrow} \varepsilon$) tako što se u listu praznih znakova stave lijeve strane svih ε-produkcija. Ako su svi znakovi desne strane produkcije prazni, onda se lista nadopuni lijevom stranom te produkcije. Algoritam se ponavlja sve dok je moguće proširiti listu praznih znakova.
2) Skup produkcija gramatike G' gradi se na sljedeći način:

   Ako je $A \to X_1 X_2 \dots X_n$ produkcija gramatike G, u skup produkcija gramatike G' dodaju se produkcije oblika $A \to \xi_1 \xi_2 \dots \xi_n$, gdje oznake ξ_i poprimaju sljedeće vrijednosti:

   a) Ako X_i nije prazni znak, onda je $\xi_i = X_i$.
   b) Ako je X_i prazni znak, onda je $\xi_i = \varepsilon$ ili $\xi_i = X_i$.

   Produkcije se grade na temelju svih mogućih kombinacija oznaka ξ_1, ξ_2, … , ξ_n. Ako sve oznake ξ_i poprime vrijednost ε, nastaje ε-produkcija i ona se ne dodaje u skup produkcija gramatike G'.

Za svaku kontekstno neovisnu gramatiku G koja generira jezik $L(G) \backslash \{\varepsilon\}$ moguće je izgraditi istovjetnu gramatiku G' koja nema jediničnih produkcija oblika $A \to B$.

Algoritam odbacivanja jediničnih produkcija odvija se u dva koraka:

1) U skup produkcija gramatike G' stave se sve produkcije gramatike G koje nisu jedinične.
2) Ako za nezavršne znakove A i B gramatike G vrijedi $A \overset{*}{\Rightarrow} B$, za sve produkcije $B \to \alpha$ koje nisu jedinične grade se nove produkcije $A \to \alpha$.

13 NORMALNI OBLIK CHOMSKOG

13.1 DEFINICIJA NORMALNOG OBLIKA CHOMSKOG

- Definicija
- Postupak pojednostavljenja gramatike u CNF

Neka gramatika $G = (V, T, P, S)$ generira kontekstno neovisni jezik $L(G)\backslash\{\varepsilon\}$. Moguće je izgraditi istovjetnu gramatiku $G' = (V', T', P', S)$ koja ima sve produkcije oblika $A \to BC$ ili $A \to a$, gdje su $A, B, C \in V$ i $a \in T$.

Algoritam pretvorbe u Chomskyjev normalni oblik (CNF) odvija se u tri koraka:

1) U skup produkcija P' stave se sve produkcije koje su u CNF, tj. oblika $A \to BC$ ili $A \to a$. U skup V' upišu se svi nezavršni znakovi.
2) Neka je produkcija gramatike G oblika $A \to X_1X_2 \dots X_m$, $A \in V$, $X_i \in T$ ili $X_i \in V$. Ako je X_i završni znak a, onda se skup V' proširi s novim znakom C_a, a skup produkcija P' proširi se produkcijom $C_a \to a$ koja je u CNF. Svi završni znakovi a u $A \to X_1X_2 \dots X_m$ zamijene se znakom C_a. Postupak se ponavlja za sve završne znakove na desnoj strani produkcije i ponavlja se za sve produkcije oblika $A \to X_1X_2 \dots X_m$.
3) Nakon završetka drugog koraka sve produkcije su oblika $A \to a$ ili $A \to B_1B_2 \dots B_m$ $(m \geq 2)$. Produkcije oblika $A \to a$ i $A \to B_1B_2$ već su u CNF, dok se ostale $(m > 2)$ zamijene s novim produkcijama tako da se definiraju novi nezavršni znakovi $D_1D_2 \dots D_{m-2}$, a zatim se produkcija $A \to B_1B_2 \dots B_m$ zamijeni skupom produkcija:

$$\{A \to B_1D_1, D_1 \to B_2D_2, \dots, D_{m-3} \to B_{m-2}D_{m-2}, D_{m-2} \to B_{m-1}B_m\}$$

13.2 POSTUPAK POJEDNOSTAVLJENJA GRAMATIKE U CNF

- Postupak pojednostavljenja gramatike u CNF
- Primjer

Postupak pojednostavljenja gramatike u CNF objašnjen je u prethodnom pitanju.

Za primjer neka je zadana gramatika $G = (\{S, A\},\ \{a, b\},\ P,\ S)$ sa sljedećim produkcijama:

1) $S \to bA$	2) $A \to bAA$
	3) $A \to aS$
	4) $A \to a$

Produkcija (4) već je u CNF pa se izravno stavlja u skup novih produkcija bez preuređivanja. Produkcija $S \to bA$ zamijeni se produkcijom $S \to C_bA$ i doda se produkcija $C_b \to b$. Na sličan način preurede se ostale produkcije te se dobije sljedeći skup produkcija:

1) $S \to C_bA$	2) $A \to C_bAA$	5) $C_a \to a$
	3) $A \to C_aS$	6) $C_b \to b$
	4) $A \to a$	

Preostala je samo produkcija (2) koja ima tri nezavršna znaka na desnoj strani. Stoga se produkcija $A \to C_bAA$ zamijeni produkcijama $A \to C_bD_1$ i $D_1 \to AA$:

1) $S \to C_b A$

2) $A \to C_b D_1$
3) $D_1 \to AA$
4) $A \to C_a S$
5) $A \to a$

6) $C_a \to a$
7) $C_b \to b$

14 NORMALNI OBLIK GREIBACHA

14.1 DEFINICIJA NORMALNOG OBLIKA GREIBACHA

- Definicija
- Postupak pojednostavljenja gramatike u GNF

Neka gramatika $G = (V, T, P, S)$ generira kontekstno neovisni jezik $L(G)\backslash\{\varepsilon\}$. Moguće je izgraditi istovjetnu gramatiku G' koja ima sve produkcije oblika $A \to a\gamma$, gdje je $a \in T$ i $\gamma \in V^*$.

Pojednostavljenje gramatike u GNF obavlja se korištenjem tri algoritma:

- Algoritam pretvorbe produkcija u CNF
- Algoritam zamjene krajnje lijevog nezavršnog znaka
- Algoritam razrješavanja lijeve rekurzije

14.2 ALGORITAM ZAMJENE KRAJNJE LIJEVOG ZNAKA

- Definicija algoritma
- Primjer

Neka je u gramatici $G = (V, T, P, S)$ r produkcija koje imaju nezavršni znak D_j na lijevoj strani:

$$\begin{gathered} D_j \to \alpha_1 \\ D_j \to \alpha_2 \\ \vdots \\ D_j \to \alpha_r \end{gathered}$$

i neka je u gramatici G produkcija koja ima nezavršni znak D_j na krajnje lijevom mjestu desne strane:

$$D_i \to D_j\gamma$$

gdje su $\alpha, \gamma \in (V \cup T)^*$.

Prethodno zadanih $r + 1$ produkcija zamijeni se sa sljedećih r produkcija:

$$\begin{gathered} D_i \to \alpha_1\gamma \\ D_i \to \alpha_2\gamma \\ \vdots \\ D_i \to \alpha_r\gamma \end{gathered}$$

gdje se u produkciji $D_i \to D_j\gamma$ nezavršni znak D_j zamijeni desnim stranama svih r produkcija oblika $D_j \to \alpha_i$.

Uzmimo za **primjer** gramatiku $G = (\{A, B, S\},\ \{a, b\},\ \{S \to aAa,\ A \to Bb,\ A \to a,\ B \to a,\ B \to b,\ B \to bb\},\ S)$.

Promotrimo produkcije $B \to a,\ B \to b,\ B \to bb$ i $A \to Bb$. Njih je moguće zamijeniti produkcijama $A \to \boldsymbol{a}b,\ A \to \boldsymbol{b}b,\ A \to \boldsymbol{bb}b$.

14.3 ALGORITAM RAZRJEŠAVANJA LIJEVE REKURZIJE

- Definicija algoritma
- Primjer

Produkcija je **lijevo rekurzivna** ako je isti nezavršni znak na lijevoj strani i na krajnje lijevom mjestu desne strane produkcije. Gramatika koja ima lijevo rekurzivne produkcije preuređuje se na sljedeći način:

Neka je za nezavršni znak D_i zadano r lijevo rekurzivnih produkcija:

$$D_i \to D_i\alpha_k, \qquad 1 \le k \le r$$

i s produkcija koje nisu lijevo rekurzivne:

$$D_i \to \beta_l, \qquad 1 \le l \le s$$

gdje su α_k i β_l nizovi završnih i nezavršnih znakova. Dane produkcije zamijene se sljedećim produkcijama:

$$\begin{gathered} D_i \to \beta_l \\ D_i \to \beta_l C_i \\ C_i \to \alpha_k \\ C_i \to \alpha_k C_i \end{gathered}$$

gdje je C_i novi nezavršni znak, $1 \le l \le s$ i $1 \le k \le r$.

Uzmimo za **primjer** gramatiku $G = (\{A, B, S\},\ \{a, b, c, d\},\ \{S \to Ac,\ A \to Aab,\ A \to Adc,\ A \to Bd,$ $B \to ad\},\ S)$.

Produkcije $A \to Aab,\ A \to Adc$ i $A \to Bd$ možemo zamijeniti produkcijama $A \to Bd$, $A \to Bd\boldsymbol{C}$, $C \to$ $\boldsymbol{ab}$, $C \to \boldsymbol{dc}$, $C \to \boldsymbol{ab}C$, $C \to \boldsymbol{dc}C$.

14.4 KORACI POSTIZANJA GNF

- Definicija algoritma
- Primjer

Algoritam pretvorbe produkcija u GNF provodi se u četiri koraka:

1) Produkcije gramatike $G = (V, T, P, S)$ preurede se u CNF. Skup nezavršnih znakova V zamijeni se skupom nezavršnih znakova $\{D_1, D_2, \dots, D_m\}$, gdje je m broj elemenata skupa V. Nezavršni znakovi u produkcijama zamijene se nezavršnim znakovima iz dobivenog skupa. Nakon pretvorbe, sve su produkcije oblika $D_i \to D_j D_k$ ili $D_i \to a$.
2) Produkcije oblika $D_i \to D_j D_k$ preurede se u oblik $D_i \to D_j\beta$, gdje je $j > i$, a β je niz nezavršnih znakova. Pretvorba se obavlja počevši od D_1. Produkcije kod kojih je $j = i$ preuređuju se algoritmom razrješavanja lijeve rekurzije, a produkcije kod kojih je $j < i$ algoritmom zamjene krajnje lijevog nezavršnog znaka.
3) Produkcije oblika $D_i \to D_j\beta$ preurede se u oblik $D_i \to a\alpha\beta$, gdje je $a \in T$, dok su α i β nizovi nezavršnih znakova. Pretvorba se obavlja počevši od D_{m-1}, pa se nastavlja redom za $D_{m-2}, \dots, D_1$. Pretvorba se temelji na algoritmu zamjene krajnje lijevog nezavršnog znaka te se produkcija $D_i \to D_j\beta$ zamjenjuje produkcijom $D_i \to a\alpha\beta$ na temelju produkcije $D_j \to a\alpha$.

4) Produkcije nastale razrješavanjem lijeve rekurzije, koje na lijevoj strani imaju nezavršne znakove C_i , a njihove desne strane počinju jednim od nezavršnih znakova iz skupa $\{D_1, D_2, \dots, D_m\}$, preurede se primjenom zamjene krajnje lijevog nezavršnog znaka.

Za **primjer** neka je zadana gramatika $G = (\{S, A, B\}, \ \{a, b\}, \ P, \ S)$ i neka su produkcije već preuređene u CNF:

1) $S \to AB$ 2) $A \to SB$ 3) $A \to b$ 4) $B \to a$

1) Uvodi se novi skup nezavršnih znakova $\{D_1, D_2, D_3\}$ kojim se zamijeni skup $\{S, A, B\}$ te se preurede produkcije:

1) $D_1 \to D_2D_3$ 2) $D_2 \to D_1D_3$ 3) $D_2 \to b$ 4) $D_3 \to a$

2) Produkcije (3) i (4) već su u GNF. Produkcija (1) se ne preuređuje jer je indeks krajnje lijevog znaka desne strane veći od indeksa znaka lijeve strane. Preuređujemo samo $D_2 \to D_1D_3$:

1) $D_1 \to D_2D_3$ 2) $\boldsymbol{D_2 \to D_2D_3D_3}$ 3) $D_2 \to b$ 4) $D_3 \to a$

Indeksi znaka lijeve strane i krajnjeg lijevog znaka desne strane su jednaki pa primjenjujemo razrješavanje lijeve rekurzije:

1) $D_1 \to D_2D_3$ 2) $\boldsymbol{D_2 \to b}$ 3) $\boldsymbol{D_2 \to bC_2}$ 4) $\boldsymbol{C_2 \to D_3D_3}$ 5) $\boldsymbol{C_2 \to D_3D_3C_2}$ 6) $D_3 \to a$

3) Treći korak započinje znakom D_2, no sve njegove produkcije su već u GNF, pa prelazimo na D_1:

1) $\boldsymbol{D_1 \to bD_3}$ 2) $\boldsymbol{D_1 \to bC_2D_3}$ 3) $D_2 \to b$ 4) $D_2 \to bC_2$ 5) $C_2 \to D_3D_3$ 6) $C_2 \to D_3D_3C_2$ 7) $D_3 \to a$

4) Preostalo je razriješiti samo znak C_2 zamjenom krajnje lijevog znaka:

1) $D_1 \to bD_3$ 2) $D_1 \to bC_2D_3$ 3) $D_2 \to b$ 4) $D_2 \to bC_2$ 5) $\boldsymbol{C_2 \to aD_3}$ 6) $\boldsymbol{C_2 \to aD_3C_2}$ 7) $D_3 \to a$

15 RAZLAGANJE (PARSIRANJE) NIZA

15.1 DEFINICIJA RAZLAGANJA NIZA

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

Određivanje pripadnosti niza w jeziku $L(G)$ naziva se prepoznavanje niza.

Parsiranje niza objedinjuje prepoznavanje niza i gradnju generativnog stabla.

U **postupku parsiranja** nastoji se izgraditi generativno stablo za zadani niz završnih znakova w i zadanu gramatiku G. Uspije li se izgraditi generativno stablo kojem su svi listovi označeni završnim znakovima niza w, taj niz pripada jeziku $L(G)$.

Vrste parsiranja dijele se prema načinu gradnje generativnog stabla:

- Parsiranje od vrha prema dnu – stablo se gradi od korijena prema listovima.
- Parsiranje od dna prema vrhu – stablo se gradi od listova prema korijenu.

Parsiranje od vrha prema dnu započinje gradnju stabla početnim nezavršnim znakom S. Ostali čvorovi grade se primjenom produkcija iz skupa P. Produkcije se primjenjuju sve dok se listovi stabla ne označe isključivo završnim znakovima zadanog niza w. Tijekom gradnje stabla završni znakovi niza w određuju koja se produkcija primjenjuje.

15.2 LL(1) GRAMATIKA I RAZLAGANJE

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

LL(1) gramatika i LL(1) parsiranje od vrha prema dnu su gramatika, odnosno parsiranje sa svojstvima:

- Ulazni se niz čita s lijeva na desno (*Left-to-right scanning*).
- Produkcije se primjenjuju na krajnje lijevi nezavršni znak u generiranom međunizu (*Leftmost derivation*).
- Odluka o primjeni produkcije donosi se na temelju samo jednog pročitanog znaka (zato je broj 1 u nazivu).

Tehnika rekurzivnog spusta služi za programsko ostvarenje parsiranja od vrha prema dnu. Ostvaruje se programskim jezikom koji ima svojstvo rekurzivnog poziva potprograma koji se pridružuju nezavršnim znakovima.

Parser s rekurzivnim spustom ostvaruje se prema sljedećim načelima:

1) U glavnom programu pročita se krajnje lijevi znak niza w i pozove se potprogram pridružen početnom nezavršnom znaku gramatike. Nakon što završi izvođenje potprograma, provjerava se je li pročitana oznaka kraja niza. Ako jest, niz se prihvaća, a inače se odbacuje.

2) Za svaki nezavršni znak izgradi se potprogram. Ako znak pročitan tijekom poziva potprograma nije jednak niti jednom od znakova $a_1, a_2, \dots, a_n$ iz produkcije $A \to a_1\beta_1 \mid a_2\beta_2 \mid \dots \mid a_n\beta_n$, niz se ne prihvaća.
3) Dijelovi potprograma koji ispituju znakove prema desnoj strani produkcije grade se na sljedeći način:
    a) Za bilo koji završni znak b na desnoj strani produkcije $A \to \alpha\alpha b\gamma$ koji nije na krajnje lijevom mjestu, niz se ne prihvaća ako pročitani znak nije b.
    b) Ako nezavršni znak B na desnoj strani produkcije $A \to \alpha\alpha B\gamma$ nije na krajnje lijevom mjestu, pročita se sljedeći znak niza i pozove se potprogram pridružen znaku B. Ako B jest na krajnje lijevom mjestu, onda se samo pozove potprogram.

15.3 RAZLAGANJE OD DNA PREMA VRHU

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

Parsiranje od dna prema vrhu započinje gradnju generativnog stabla listovima, odnosno završnim znakovima gramatike. U nizu završnih znakova w, odnosno u dobivenim međunizovima završnih i nezavršnih znakova, nastoji se prepoznati jedna od desnih strana produkcija. Ako je dio međuniza jednak desnoj strani produkcije, taj dio se zamijeni lijevom stranom te produkcije. Tako se nastoje izgraditi svi čvorovi stabla uključujući i korijen. Ova metoda **primjenjuje se** u generatorima parsera.

Za **primjer** uzmimo gramatiku $G = (\{E, T, F\},\ \{var, +, *, (,)\},\ P,\ E)$ s produkcijama:

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

- Provjerimo pripada li niz $var + var * var$ jeziku $L(G)$. Niz se čita s lijeva na desno.
- Prvo se primijeni produkcija (6) na osnovu koje se krajnje lijevi znak var zamijeni nezavršnim znakom F. Tako nastaje međuniz $F + var * var$:

$$var + var * var \Leftarrow F + var * var$$

- Daljnja gradnja stabla je jednoznačna te se redom primijene produkcije (4), (2), (6) i opet (4):

$$F + var * var \Leftarrow T + var * var \Leftarrow E + var * var \Leftarrow E + F * var \Leftarrow E + T * var$$

- Nastavak gradnje je nejednoznačan jer se mogu primijeniti tri različite produkcije:

Produkcija (1): $E + T * var \Leftarrow E * var$
Produkcija (2): $E + T * var \Leftarrow E + E * var$
Produkcija (6): $E + T * var \Leftarrow E + T * F$

- Prve dvije produkcije ne omogućavaju završetak gradnje generativnog stabla, pa gradnju dovršavamo primjenom redukcije (6), zatim (3) i na kraju (1):

$$E + T * F \Leftarrow E + T \Leftarrow E$$

LR(k) parser koristi metodu parsiranja od dna prema vrhu te ima sljedeća svojstva:

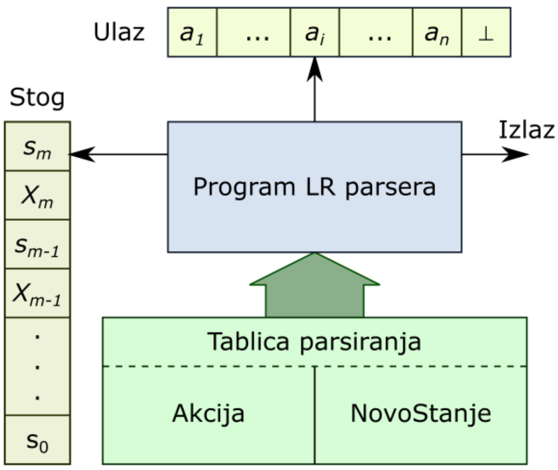
- Niz se čita s lijeva na desno (*Left-to-right scanning*).

- Stablo se gradi obrnutim postupkom generiranja niza zamjenom krajnje desnog nezavršnog znaka (*Rightmost derivation*).
- Broj k određuje da je potrebno pročitati najviše k znakova unaprijed kako bi se donijela odluka o primjeni produkcije.

15.4 LR(K) RAZLAGANJE

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

Model LR parsera prikazan je na slici 15.1. Dijelovi LR parsera su: ulazni spremnik, potisni stog, program LR parsera, tablica parsiranja i izlaz.



Slika 15.1 Model LR parsera

Potisni stog služi za spremanje niza oblika $s_0X_1s_1X_2s_2 \dots X_ms_m$. Znakovi X_i su znakovi gramatike, a s_i su stanja. Stanje na vrhu stoga (s_m) jednoznačno određuje njegov sadržaj. Znakove X_i nije nužno dodavati na stog.

Tablica parsiranja ima dva dijela: tablicu *Akcija* i tablicu *NovoStanje*. Na osnovu stanja na vrhu stoga s_m i ulaznog znaka a_i iz tablice *Akcija* odredi se akcija koja se izvodi nad ulaznim nizom i stogom i koja mijenja konfiguraciju LR parsera. To može biti pomak stanja, redukcija, prihvaćanje ili odbacivanje.

Konfiguracija LR parsera je lista koja se dobije spajanjem i stavljanjem zareza između trenutnog niza znakova na stogu i niza znakova u ulaznom spremniku koji su desno od a_i.

Algoritam LR parsera izvodi se na sljedeći način:

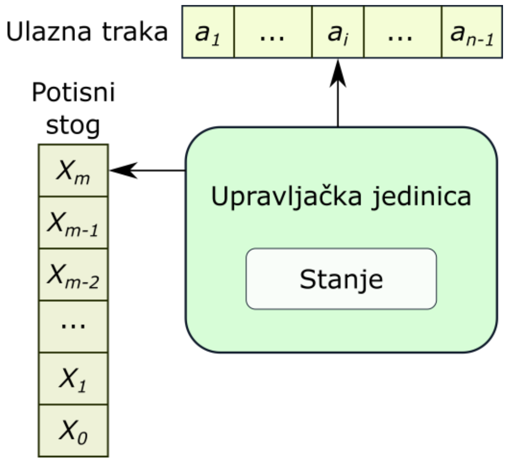
1) Ulaz algoritma je zadani niz w i tablica LR parsera koja se generira na osnovu zadane gramatike G. Na početku rada na stogu je početno stanje s_0, a u ulaznom spremniku niz $w\perp$, gdje je $\perp$ oznaka kraja niza.
2) **Program LR parsera** čita znak po znak ulaznog spremnika i koristi postupak generiranja stabla zamjenom krajnje desnog nezavršnog znaka. Ako je niz u jeziku $L(G)$, ispisuje da se niz prihvaća, a inače ispisuje da se odbacuje.
3) Parser izvodi program sve dok se ne izvede akcija prihvaćanja ili akcija odbacivanja

16 POTISNI AUTOMAT (PA)

16.1 MODEL I RAD POTISNOG AUTOMATA

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

Model potisnog automata prikazan na slici 16.1 proširuje model konačnog automata te se gradi za potrebe prihvaćanja kontekstno neovisnih jezika. Uz postojeću upravljačku jedinicu, glavu za čitanje i ulaznu traku dodaje se potisni stog (LIFO stog).



Slika 16.1 Model potisnog automata

Upravljačka jedinica donosi odluku o promjeni sadržaja vrha stoga, pomaku glave za čitanje i promjeni stanja na temelju tri podatka:

- Stanja upravljačke jedinice
- Znaka na vrhu stoga
- Znaka na ulaznoj traci

Upravljačka jedinica odlučuje koji se niz stavlja na vrh stoga. Na vrh stoga moguće je staviti:

- Prazni niz ε
- Niz duljine jednog znaka
- Niz duljine više znakova

Na osnovu stanja q upravljačke jedinice, znaka a ulazne trake te znaka Z na vrhu stoga, upravljačka jedinica obavlja jedan od prijelaza:

- Na temelju trojke (q, a, Z) upravljačka jedinica mijenja stanje u novo stanje p, pomakne glavu za čitanje jedno mjesto u desno i zamijeni znak na vrhu stoga nizom znakova γ.
- Na temelju trojke (q, ε, Z) upravljačka jedinica mijenja stanje u novo stanje p, ostavi glavu za čitanje na istom mjestu i zamijeni znak na vrhu stoga nizom znakova γ.

Odluka o prihvaćanju niza donosi se isključivo na jedan od dva moguća načina:

- PA M prihvaća prihvatljivim stanjem – uđe li upravljačka jedinica u prihvatljivo stanje nakon što M pročita sve znakove ulazne trake, niz se prihvaća.
- PA M prihvaća praznim stogom – isprazni li se stog nakon što M pročita sve znakove ulazne trake, niz se prihvaća.

16.2 FORMALNA DEFINICIJA POTISNOG AUTOMATA

- Formalna definicija
- Funkcija prijelaza
- Konfiguracija PA

Potisni automat (PA) formalno se definira uređenom sedmorkom:

$$pa = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \tag{18}$$

gdje je:

Q	konačan skup stanja
Σ	konačan skup ulaznih znakova
Γ	konačan skup znakova stoga
$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$	funkcija prijelaza
$q_0 \in Q$	početno stanje
$Z_0 \in \Gamma$	početni znak stoga
$F \subseteq Q$	skup prihvatljivih stanja

Funkcija prijelaza potisnog automata $\boldsymbol{\delta}$ pridružuje trojci (q, a, Z) konačni skup parova (p_i, γ_i):

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

gdje je $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, $p_i \in Q$, $\gamma_i \in \Gamma^*$, $1 \leq i \leq m$. Ako PA u stanju q pročita ulazni znak a, a na vrhu stoga je znak Z, tada PA prelazi u jedno od stanja p_i, vrh stoga Z zamijeni se odgovarajućim nizom γ_i, a glava za čitanje pomakne se na sljedeći ulazni znak. Prijelaz $\delta(q, \varepsilon, Z)$ definiran je isključivo stanjem PA i znakom na vrhu stoga te se naziva ε-prijelaz.

Konfiguracija potisnog automata definira se uređenom trojkom (q, w, γ), gdje je q stanje, w je nepročitani dio ulaznog niza, a γ je sadržaj stoga.

Konfiguracija PA M mijenja se iz $(q, aw, Z\alpha)$ u $(p, w, \beta\alpha)$ ako i samo ako skup $\delta(q, a, Z)$ sadrži (p, β). Primjenom relacije $\succ$ formalno se zapisuje promjena konfiguracije:

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

16.3 DETERMINISTIČKI I NEDETERMINISTIČKI PA

- Prihvaćanje jezika
- Nedeterministički PA
- Deterministički PA

Prihvaćanje jezika potisnim automatom $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ definira se na dva načina:

- PA M prihvatljivim stanjem prihvaća jezik:

$$L(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\succ} (p, \varepsilon, \gamma), p \in F, \gamma \in \Gamma^*\}$$

- PA M praznim stogom prihvaća jezik:

$$N(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\succ} (p, \varepsilon, \varepsilon), p \in Q\}$$

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

PA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je **deterministički PA** (DPA) ako i samo ako su ispunjena sljedeća dva uvjeta:

1) Ako je $\delta(q, \varepsilon, Z) \neq \emptyset$, tada je $\delta(q, a, Z) = \emptyset$, $\forall a \in \Sigma$. Ovaj uvjet sprječava izbor između običnog prijelaza i ε-prijelaza.
2) Skup $\delta(q, a, Z)$ sadrži najviše jedan element. Time je zagarantirana jednoznačnost prijelaza.

17 TRANSFORMACIJE POTISNOG AUTOMATA

17.1 KONSTRUKCIJA ESPA[2] IZ ASPA[3]

- Istovjetnost PA
- Pristup konstrukciji ESPA iz ASPA
- Koraci konstrukcije ESPA iz ASPA
- Primjer

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

Neka je zadan ASPA $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ koji prihvaća jezik $L(M_2)$. **Konstrukcija** istovjetnog ESPA M_1 zasniva se na simulaciji ASPA M_2. Uđe li M_2 tijekom simulacije u prihvatljivo stanje, M_1 isprazni svoj stog. Kako bi se to omogućilo, koristi se dodatni znak stoga X_0 koji se stavi na dno stoga M_1. Ako M_2 isprazni stog bez da uđe u prihvatljivo stanje, M_1 neće prihvatiti niz jer je X_0 još uvijek na stogu.

Na temelju M_2 konstruira se $M_1 = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \emptyset)$ kroz sljedeće **korake**:

1) M_1 na početku rada prelazi u početnu konfiguraciju M_2: $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$.
2) U skup $\delta'(q, a, Z)$ stave se svi elementi skupa $\delta(q, a, Z)$, $\forall q \in Q$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\forall Z \in \Gamma$.
3) U skup $\delta'(q, \varepsilon, Z)$ dodaje se ε-prijelaz (q_e, ε), $\forall q \in F$, $\forall Z \in \Gamma \cup \{X_0\}$.
4) U skup $\delta'(q_e, \varepsilon, Z)$ dodaje se ε-prijelaz (q_e, ε), $\forall Z \in \Gamma \cup \{X_0\}$.

Za **primjer** neka je zadan PA $M_2 = (\{q_1, q_2\}, \{0,1\}, \{N, K\}, \delta, q_1, K, \{q_2\})$ s prijelazima:

$$\delta(q_1, 0, K) = \{(q_1, NK)\} \qquad \delta(q_1, 0, N) = \{(q_1, NN)\}$$
$$\delta(q_1, 1, N) = \{(q_2, \varepsilon)\} \qquad \delta(q_2, 1, N) = \{(q_2, \varepsilon)\}$$

Istovjetni PA $M_1 = (\{q_1, q_2, q_0', q_e\}, \{0,1\}, \{N, K, X_0\}, \delta', q_0', X_0, \emptyset)$ gradi se na sljedeći način:

1) Definira se prijelaz u početnu konfiguraciju M_2:

$$\delta'(q_0', \varepsilon, X_0) = \{(q_1, KX_0)\}$$

2) Preuzimaju se svi prijelazi M_2:

$$\delta'(q_1, 0, K) = \{(q_1, NK)\} \qquad \delta'(q_1, 0, N) = \{(q_1, NN)\}$$
$$\delta'(q_1, 1, N) = \{(q_2, \varepsilon)\} \qquad \delta'(q_2, 1, N) = \{(q_2, \varepsilon)\}$$

3) Dodaju se ε-prijelazi u stanje q_e:

$$\delta'(q_2, \varepsilon, N) = \{(q_e, \varepsilon)\} \qquad \delta'(q_2, \varepsilon, K) = \{(q_e, \varepsilon)\} \qquad \delta'(q_2, \varepsilon, X_0) = \{(q_e, \varepsilon)\}$$

4) Dodaju se ε-prijelazi koji prazne stog:

$$\delta'(q_e, \varepsilon, N) = \{(q_e, \varepsilon)\} \qquad \delta'(q_e, \varepsilon, K) = \{(q_e, \varepsilon)\} \qquad \delta'(q_e, \varepsilon, X_0) = \{(q_e, \varepsilon)\}$$

[2] Empty Stack Pushdown Automata
[3] Acceptable State Pushdown Automata

17.2 DOKAZ ISTOVJETNOSTI ESPA I ASPA

- Priprema simulacije
- Simulacija rada ASPA
- Pražnjenje stoga

Dokaz istovjetnosti PA $M_1 = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \emptyset)$ i PA $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ izvodi se u dva dijela:

1) U prvom dijelu dokazuje se da PA M_1 prihvaća niz x ako ga prihvaća PA M_2:

   Pretpostavimo da M_2 prihvaća niz x prihvatljivim stanjem:

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

   gdje je $q \in F$, $A \in \Gamma$ i $\gamma \in \Gamma^*$.

   Na temelju prvog koraka konstrukcije ESPA iz ASPA vrijedi:

   $$(q_0', x, X_0) \underset{M_1}{\succ} (q_0, x, Z_0X_0)$$

   Na temelju drugog koraka svi prijelazi M_2 ujedno su i prijelazi M_1:

   $$(q_0, x, Z_0X_0) \underset{M_1}{\overset{*}{\succ}} (q, \varepsilon, A\gamma X_0)$$

   Budući da je $q \in F$, treći korak omogućuje prijelaz u q_e, a s vrha stoga skida se jedan znak:

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

   Četvrti korak konstrukcije omogućuje pražnjenje stoga:

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

   Dakle, prihvaća li M_2 niz x prihvatljivim stanjem, M_1 prelazi u konfiguraciju:

   $$(q_0', \varepsilon, X_0) \underset{M_1}{\overset{*}{\succ}} (q_e, \varepsilon, \varepsilon)$$

   i prihvaća niz x praznim stogom.

2) U drugom dijelu dokazuje se da PA M_2 prihvaća niz x prihvatljivim stanjem ako ga PA M_1 prihvaća praznim stogom. Slijed prijelaza M_1 tijekom prihvaćanja niza x čine prijelazi zadani u prvom koraku, zatim slijed prijelaza zadanih u drugom koraku konstrukcije i na kraju slijed prijelaza koji prazni stog. Na temelju zadnja dva koraka, stog se prazni ako i samo ako je u konfiguraciji $(q, \varepsilon, \gamma X_0)$ stanje q prihvatljivo. Prema tome, M_2 prihvaća niz x prihvatljivim stanjem ako i samo ako ga M_1 prihvaća praznim stogom.

17.3 KONSTRUKCIJA ASPA IZ ESPA

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

Neka PA $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$ prihvaća jezik praznim stogom. Potrebno je **konstruirati** PA M_2 koji isti jezik prihvaća prihvatljivim stanjem. M_2 simulira rad M_1 i prelazi u prihvatljivo stanje ako i samo ako M_1 isprazni svoj stog.

Koraci konstrukcije $M_2 = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \{q_f\})$:

1) M_2 na početku rada prelazi u početnu konfiguraciju M_1: $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$.
2) U skup $\delta'(q, a, Z)$ stave se svi elementi skupa $\delta(q, a, Z)$, $\forall q \in Q$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\forall Z \in \Gamma$.
3) U skup $\delta'(q, \varepsilon, X_0)$ dodaje se ε-prijelaz (q_f, ε), $\forall q \in Q$.

Na temelju provedenih koraka dobije se slijed prijelaza:

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

Prvi prijelaz osigurava početne uvjete, a daljnji slijed prijelaza simulira prijelaze PA M_1. Posljednji korak definira prihvaćanje niza prijelazom u prihvatljivo stanje ako je M_1 ispraznio svoj stog. Prikazani slijed prijelaza pokazuje da M_2 prihvaća niz x prihvatljivim stanjem ako i samo ako M_1 prihvaća niz x praznim stogom.

Za **primjer** neka je zadan PA $M_1 = (\{q_1\},\ \{0,1\},\ \{N, K\},\ \delta,\ q_1,\ K,\ \emptyset)$ koji prihvaća jezik $L(M_1) = \{0^n1^n \mid n \geq 0\}$ praznim stogom:

$$\delta(q_1, 0, K) = \{q_1, NK\} \qquad \delta(q_1, 0, N) = \{q_1, NN\}$$
$$\delta(q_1, 1, N) = \{q_1, \varepsilon\} \qquad \delta(q_1, \varepsilon, K) = \{q_1, \varepsilon\}$$

Na temelju navedenih koraka može se konstruirati PA $M_2 = (\{q_0', q_1, q_f\},\ \{0,1\},\ \{N, K, X_0\},\ \delta',\ q_0',\ X_0,\ \{q_f\})$ koji isti jezik prihvaća prihvatljivim stanjem:

1) Uvede se početni prijelaz:

$$\delta'(q_0', \varepsilon, X_0) = \{q_1, KX_0\}$$

2) Preuzmu se svi prijelazi PA M_1:

$$\delta'(q_1, 0, K) = \{q_1, NK\} \qquad \delta'(q_1, 0, N) = \{q_1, NN\}$$
$$\delta'(q_1, 1, N) = \{q_1, \varepsilon\} \qquad \delta'(q_1, \varepsilon, K) = \{q_1, \varepsilon\}$$

3) Doda se prijelaz u prihvatljivo stanje:

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

18 POTISNI AUTOMAT I CFG

18.1 KONSTRUKCIJA ESPA IZ CFG

- Pristup sintezi
- Postupak sinteze
- Primjer

Zbog istovjetnosti ESPA i ASPA dovoljno je na temelju gramatike konstruirati samo ESPA. Za bilo koji kontekstno neovisni jezik L postoji nedeterministički PA M koji ga prihvaća praznim stogom ($N(M) = L$) uz pretpostavku da prazni niz ε nije element jezika L. Jezik L zadan je gramatikom $G = (V, T, P, S)$ s produkcijama u GNF.

PA $M = (\{q\}, \Sigma, \Gamma, \delta, q, S, \emptyset)$ **konstruira se** na sljedeći način:

1) M ima samo jedno stanje q koje je ujedno i početno stanje.
2) $\Sigma = T$ (skup ulaznih znakova jednak je skupu završnih znakova gramatike).
3) $\Gamma = V$ (skup znakova stoga jednak je skupu nezavršnih znakova gramatike).
4) Početni znak stoga je početni nezavršni znak gramatike S.
5) $F = \emptyset$ (skup prihvatljivih stanja je prazan).
6) PA M prihvaća praznim stogom.
7) Funkcija prijelaza definira se na sljedeći način:

$$\delta(q, a, A) \text{ sadrži } (q, \gamma) \text{ ako i samo ako je zadana produkcija } A \to a\gamma$$

PA M simulira postupak generiranja niza zamjenom krajnje lijevog nezavršnog znaka. M prihvaća niz x praznim stogom ako i samo ako gramatika G generira niz x.

Za **primjer** neka je zadana gramatika $G = (\{S, A\},\ \{a, b\},\ \{S \to aAA,\ A \to aS \mid bS \mid a\},\ S)$. Na temelju danih pravila gradi se PA $M = (\{q\},\ \{a, b\},\ \{S, A\},\ \delta,\ q,\ S,\ \emptyset)$ s funkcijom prijelaza:

$\delta(q, a, S) = \{(q, AA)\}$ na temelju produkcije $S \to aAA$,

$\delta(q, a, A) = \{(q, S), (q, \varepsilon)\}$ na temelju produkcija $A \to aS$ i $A \to a$,

$\delta(q, b, A) = \{(q, S)\}$ na temelju produkcije $A \to bS$.

18.2 SINTEZA CFG IZ ESPA

- Postupak gradnje gramatike
- Primjer

Za zadani PA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$ kontekstno neovisna gramatika $G = (V, T, P, S)$ **konstruira se** na sljedeći način:

1) Nezavršni znakovi označe se zagradama oblika $[q, A, p] \in V$, $q, p \in Q$, $A \in \Gamma$ te se u skup V doda početni nezavršni znak S.
2) Za početno stanje q_0, početni znak stoga Z_0 i sva stanja $q \in Q$ grade se produkcije:

$$S \to [q_0, Z_0, q]$$

3) Ako skup $\delta(q, a, A)$ sadrži $(q_1, B_1 B_2 \dots B_m)$, onda se grade produkcije (za sve moguće kombinacije stanja $q_2 \dots q_{m+1}$ iz skupa Q):

$$[q, A, q_{m+1}] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$$

Dobivena gramatika G simulira rad PA M postupkom zamjene krajnje lijevog znaka. Niz nezavršnih znakova u međunizu jednak je nizu znakova stoga PA M. Gramatika počinje generirati niz x iz $[q, A, p] \in V$ ako i samo ako PA M čitanjem znaka niza x izbriše A sa stoga i promjeni stanje iz q u p.

Neka je kao **primjer** zadan ESPA $M = (\{q_1\}, \{0,1\}, \{N, K\}, \delta, q_1, K, \emptyset)$ koji prihvaća jezik $L(M) = \{0^n 1^n \mid n \geq 0\}$ s funkcijom prijelaza:

$$\delta(q_1, 0, K) = \{(q_1, NK)\} \qquad \delta(q_1, 0, N) = \{(q_1, NN)\}$$
$$\delta(q_1, 1, N) = \{(q_1, \varepsilon)\} \qquad \delta(q_1, \varepsilon, K) = \{(q_1, \varepsilon)\}$$

Na temelju definiranih koraka gradi se gramatika $G = (V, \{0,1\}, P, S)$:

- U skup nezavršnih znakova V unose se sljedeći znakovi:

$$V = \{S, [q_1, K, q_1], [q_1, N, q_1]\}$$

- Na temelju prvog koraka stvara se produkcija za početni nezavršni znak:

$$S \to [q_1, K, q_1]$$

- Na temelju prijelaza PA M grade se sljedeće produkcije[4]:

$$[q_1, K, q_1] \to 0[q_1, N, q_1][q_1, K, q_1]$$

$$[q_1, N, q_1] \to 0[q_1, N, q_1][q_1, N, q_1]$$

$$[q_1, N, q_1] \to 1$$

$$[q_1, K, q_1] \to \varepsilon$$

[4] Kod složenijih primjera potrebno je odbaciti produkcije s mrtvim i nedohvatljivim znakovima.

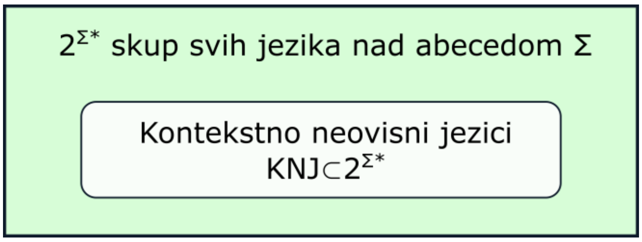
19 Svojstva kontekstno neovisnih jezika

19.1 Kontekstno neovisni jezik i PA

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

Jezik L je kontekstno neovisan ako i samo ako postoji PA koji ga prihvaća. Jezik $L = \{a^i b^i c^i \mid i \geq 1\}$ **primjer** je jezika koji nije kontekstno neovisan jer ne postoji PA koji ga prihvaća.

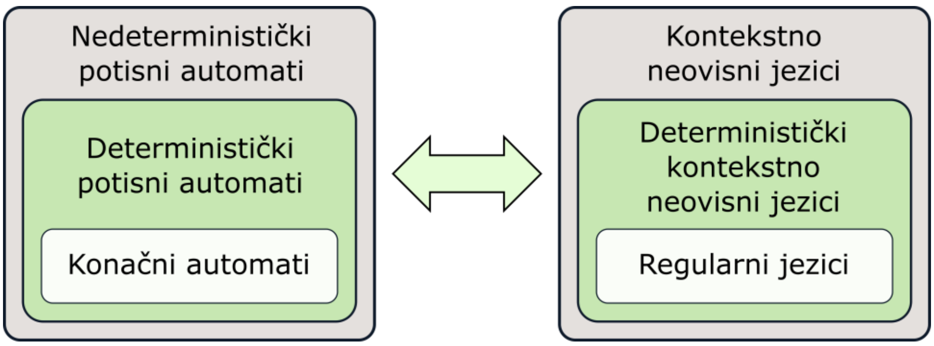
Klasa kontekstno neovisnih jezika pravi je podskup skupa svih jezika (slika 19.1).



Slika 19.1 Skup kontekstno neovisnih jezika

Jezici koje prihvaća deterministički PA su **deterministički** kontekstno neovisni jezici, a jezici koje prihvaća nedeterministički PA su **nedeterministički** kontekstno neovisni jezici. Za jezik $L = \{ww^R \mid R \geq 1\}$ nije moguće izgraditi DPA, ali je moguće izgraditi NPA. Dakle, klasa determinističkih kontekstno neovisnih jezika je pravi podskup klase kontekstno neovisnih jezika (slika 19.2).

DKA je poseban slučaj DPA koji ne koristi stog što znači da je regularne jezike moguće prihvatiti sa DPA, odnosno klasa regularnih jezika je pravi podskup klase determinističkih kontekstno neovisnih jezika (slika 19.2).



Slika 19.2 Hijerarhija automata i jezika

19.2 SVOJSTVA ZATVORENOSTI KNJ NA UNIJU, NADOVEZIVANJE

- Dokaz zatvorenosti na uniju
- Dokaz zatvorenosti na nadovezivanje

Istovjetnost kontekstno neovisne gramatike, kontekstno neovisnih jezika i potisnih automata koristi se za opis svojstava kontekstno neovisnih jezika.

Neka gramatika $G_1 = (V_1, T_1, P_1, S_1)$ generira jezik $L(G_1)$, a gramatika $G_2 = (V_2, T_2, P_2, S_2)$ neka generira jezik $L(G_2)$, pri čemu vrijedi $V_1 \cap V_2 = \emptyset$.

Unija kontekstno neovisnih jezika je kontekstno neovisni jezik. Gramatika $G_3 = (V_3, T_3, P_3, S_3)$ koja generira jezik $L(G_3) = L(G_1) \cup L(G_2)$ gradi se na sljedeći način:

1) $V_3 = V_1 \cup V_2 \cup \{S_3\},\ S_3 \notin V_1 \cup V_2$
2) $T_3 = T_1 \cup T_2$
3) $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 | S_2\}$

Prvo se dokazuje $L(G_1) \cup L(G_2) \subseteq L(G_3)$. Uzmimo neki $w \in L(G_1)$. S obzirom da su sve produkcije gramatike G_1 ujedno i produkcije G_3, koristeći novu produkciju $S_3 \rightarrow S_1$ gramatike G_3, vrijedi:

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

Istim postupkom gramatika G_3 generira neki $w \in L(G_2)$:

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

Dakle, bilo koji niz w iz unije jezika $L(G_1) \cup L(G_2)$ pripada jeziku $L(G_3)$, odnosno vrijedi $L(G_1) \cup L(G_2) \subseteq L(G_3)$. Sada dokazujemo da vrijedi $L(G_3) \subseteq L(G_1) \cup L(G_2)$. S obzirom da je $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 | S_2\}$, gramatika G_3 generira neki niz w na sljedeći način:

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

Budući da je $V_1 \cap V_2 = \emptyset$, iz S_1 koristimo produkcije P_1, a iz S_2 koristimo produkcije P_2, pa se može zaključiti da bilo koji niz $w \in L(G_3)$ ujedno pripada i uniji jezika $L(G_1) \cup L(G_2)$, odnosno da je $L(G_3) \subseteq L(G_1) \cup L(G_2)$.

Na temelju prethodno dokazanih tvrdnji dokazano je da unija kontekstno neovisnih jezika jest kontekstno neovisni jezik: $L(G_1) \cup L(G_2) = L(G_3)$.

Nadovezivanje kontekstno neovisnih jezika jest kontekstno neovisni jezik. Gramatika $G_3 = (V_3, T_3, P_3, S_3)$ koja generira jezik $L(G_3) = L(G_1)L(G_2)$ gradi se na sljedeći način:

1) $V_3 = V_1 \cup V_2 \cup \{S_3\},\ S_3 \notin V_1 \cup V_2$
2) $T_3 = T_1 \cup T_2$
3) $P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 S_2\}$

Dokazuje se na sličan način kao i za operaciju unije. Dokaz se zasniva na sljedećem postupku generiranja niza:

$$S_3 \underset{G_3}{\Rightarrow} S_1 S_2 \overset{*}{\underset{G_1}{\Rightarrow}} w_1 S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w_1 w_2$$

gdje je $w_1 w_2 \in L(G_3),\ w_1 \in L(G_1), w_2 \in L(G_2)$.

19.3 SVOJSTVA ZATVORENOSTI KNJ NA KLEENE, SUPSTITUCIJU

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

Kontekstno neovisni jezici zatvoreni su **s obzirom na Kleeneov operator**.

Neka gramatika $G_1 = (V_1, T_1, P_1, S_1)$ generira jezik $L(G_1)$. Gramatika $G_2 = (V_2, T_2, P_2, S_2)$ koja generira jezik $L(G_2) = L(G_1)^*$ konstruira se na sljedeći način:

1) $V_2 = V_1 \cup \{S_2\}, S_2 \notin V_1$
2) $T_2 = T_1$
3) U skup produkcija $P_2 = P_1$ dodaju se produkcije:

$$S_2 \to S_1 S_2 \mid \varepsilon$$

Dokaz zatvorenosti zasniva se na postupcima generiranja niza:

1) $S_2 \underset{G_2}{\Rightarrow} S_1 S_2 \overset{*}{\underset{G_1}{\Rightarrow}} w_1 S_2 \underset{G_2}{\Rightarrow} w_1 S_1 S_2 \overset{*}{\underset{G_1}{\Rightarrow}} w_1 w_1 S_2 \underset{G_2}{\Rightarrow} \dots \overset{*}{\underset{G_1}{\Rightarrow}} w_1^+ S_2 \underset{G_2}{\Rightarrow} w_1^+$, gdje su $w_1^+ \in L(G_2)$ i $w_1 \in L(G_1)$
2) $S_2 \underset{G_2}{\Rightarrow} \varepsilon$

Kontekstno neovisni jezici zatvoreni su **s obzirom na supstituciju**.

Neka gramatika $G = (V, T, P, S)$ generira kontekstno neovisni jezik $L(G)$. Zamijenimo sve završne znakove $a_i \in T$ nizovima kontekstno neovisnog jezika $L(G_i)$ kojeg generira gramatika $G_i = (V_i, T_i, P_i, S_i)$, $1 \le i \le k$, $k = |T|$. Nastali jezik L' je kontekstno neovisan i za njega se gramatika $G' = (V', T', P', S')$ konstruira na sljedeći način:

1) $V' = V \cup V_1 \cup V_2 \cup \dots \cup V_k, V \cap V_i = \emptyset, V_i \cap V_j = \emptyset, \forall i, j \in [1, k], i \neq j$
2) $T' = T_1 \cup T_2 \cup \dots \cup T_k$
3) $S' = S$
4) U skup produkcija $P' = P_1 \cup P_2 \cup \dots \cup P_k$ dodaju se produkcije gramatike G kojima se svaki završni znak a_i zamijeni početnim nezavršnim znakom S_i gramatike G_i.

Za **primjer** neka je zadan jezik $L(G)$ kojeg generira gramatika $G = (\{S\}, \{a, b\}, P, S)$ s produkcijama:

$$S \to aSbS \mid bSaS \mid \varepsilon$$

Znak a zamijenimo nizovima jezika $L_1 = \{0^n 1^n \mid n \ge 1\}$ kojeg generira gramatika $G_1 = (\{S_1\}, \{0,1\}, \{S_1 \to 0S_1 1 \mid 01\}, S_1)$, a znak b nizovima jezika $L_2 = \{ww^R \mid w \in (0 \vee 2)^*\}$ kojeg generira gramatika $G_2 = (\{S_2\}, \{0,2\}, \{S_2 \to 0S_2 0 \mid 2S_2 2 \mid \varepsilon\}, S_2)$. Ovim zamjenama nastaje jezik L' koji generira gramatika $G' = (V', T', P', S')$:

1) $V' = \{S, S_1, S_2\}$
2) $T' = \{0,1,2\}$
3) $S' = S$
4) P':

$$S \to S_1 S S_2 S \mid S_2 S S_1 S \mid \varepsilon \qquad S_1 \to 0S_1 1 \mid 01 \qquad S_2 \to 0S_2 0 \mid 2S_2 2 \mid \varepsilon$$

19.4 ZATVORENOST PRESJEKA KNJ I RJ

- Dokaz zatvorenosti presjeka KNJ i RJ
- Primjer

Presjek kontekstno neovisnog jezika i regularnog jezika je kontekstno neovisni jezik.

Pretpostavimo da KNJ L_1 prihvaća PA $M_1 = (Q_1,\ \Sigma,\ \Gamma,\ \delta_1,\ q_0,\ Z_1,\ F_1)$, a da RJ L_2 prihvaća DKA $M_2 = (Q_2,\ \Sigma,\ \delta_2,\ p_0,\ F_2)$. Moguće je izgraditi PA $M' = (Q',\ \Sigma,\ \Gamma,\ \delta',\ q_0',\ Z_0,\ F')$ koji prihvatljivim stanjem prihvaća jezik $L = L_1 \cap L_2$:

1) $Q' = Q_2 \times Q_1$
2) $q_0' = [p_0, q_0]$
3) $F' = F_2 \times F_1$
4) Skup $\delta'([p,q],\ a,\ X)$ sadrži $([p',q'],\ \gamma)$ ako i samo ako je $\delta_2(p,a) = p'$ i $(q',\gamma) \in \delta_1(q,a,X)$. Ako je $a = \varepsilon$, onda je $p' = p$.

Dokaz da M' prihvaća $L = L_1 \cap L_2$ izvodi se indukcijom s obzirom na i, dokazivanjem da za M' vrijedi:

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

ako i samo ako je:

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

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

Uzmimo za **primjer** PA $M_1 = (\{q_1,q_2\},\ \{0,1\},\ \{N,K\},\ \delta_1,\ q_1,\ K,\ \{q_2\})$ koji prihvaća jezik $L(M_1) = \{0^n1^m \mid m,n \geq 1,\ m \leq n\}$ i DKA $M_2 = (\{p_1,p_2,p_3\},\ \{0,1\},\ \delta_2,\ p_1,\ \{p_3\})$ koji prihvaća jezik $L(M_2)$ čiji nizovi imaju barem dva znaka 1. Funkcije prijelaza su:

$$\delta_1(q_1,0,K) = \{(q_1,NK)\} \qquad \delta_1(q_1,0,N) = \{(q_1,NN)\}$$
$$\delta_1(q_1,1,N) = \{(q_2,\varepsilon)\} \qquad \delta_1(q_2,1,N) = \{(q_2,\varepsilon)\}$$

$$\delta_2(p_1,0) = p_1 \qquad \delta_2(p_2,0) = p_2 \qquad \delta_2(p_3,0) = p_3$$
$$\delta_2(p_1,1) = p_2 \qquad \delta_2(p_2,1) = p_3 \qquad \delta_2(p_3,1) = p_3$$

Jezik $L = L(M_1) \cap L(M_2) = \{0^n1^m \mid m,n \geq 2,\ m \leq n\}$ prihvaća PA $M' = (Q',\Sigma,\Gamma,\delta',q_0',Z_0,F')$:

1) $Q' = \{[p_1,q_1],[p_1,q_2],[p_2,q_1],[p_2,q_2],[p_3,q_1],[p_3,q_2]\}$
2) $q_0' = [p_1,q_1]$
3) $F' = \{[p_3,q_2]\}$
4) δ' (za dohvatljiva stanja):

$$\delta_1([p_1,q_1],0,K) = \{([p_1,q_1],NK)\} \qquad \delta_1([p_1,q_1],0,N) = \{([p_1,q_1],NN)\}$$
$$\delta_1([p_1,q_1],1,N) = \{([p_2,q_2],\varepsilon)\} \qquad \delta_1([p_2,q_2],1,N) = \{([p_3,q_2],\varepsilon)\}$$
$$\delta_1([p_3,q_2],1,N) = \{([p_3,q_2],\varepsilon)\}$$

20 SVOJSTVO NAPUHAVANJA KNJ

20.1 SVOJSTVO NAPUHAVANJA KNJ

- Pristup preko gramatike
- Analiza generiranja niza
- Oblici generiranja niza

Svojstvo napuhavanja KNJ koristi se za dokazivanje kontekstne neovisnosti jezika te se zasniva na:

- Broju čvorova generativnog stabla
- Broju nezavršnih znakova gramatike

Za dovoljno dugački niz broj unutrašnjih čvorova stabla veći je od kardinalnog broja skupa nezavršnih znakova gramatike, što znači da je više čvorova označeno istim nezavršnim znakom.

Neka **gramatika** $G = (V, T, P, S)$ generira stablo s brojem čvorova većim od $|V|$. Tada sigurno postoji put stabla u kojem je jedan nezavršni znak barem na dva mjesta na istom putu, i to pri dnu stabla. Za takvo stablo postoji sljedeći postupak generiranja niza:

$$S \overset{*}{\underset{G}{\Rightarrow}} u\underline{A}y \overset{*}{\underset{G}{\Rightarrow}} u\underline{vAx}y \overset{*}{\underset{G}{\Rightarrow}} uvwxy$$

Nezavršni znak A koristi se dva puta u postupku generiranja niza $uvwxy$ $(u, v, w, x, y \in T^*)$.

Budući da je **postupak generiranja niza** $A \overset{*}{\underset{G}{\Rightarrow}} vAx$ moguće ponavljati proizvoljan broj puta, gramatika G generira i niz sljedećeg oblika:

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

gdje je $i \geq 0$.

Svojstvo napuhavanja KNJ glasi:

Neka je L kontekstno neovisni jezik. Postoji konstanta n koja ovisi isključivo o jeziku L takva da je niz $z \in L$, $|z| \geq n$, moguće napisati kao niz $uvwxy$ za koji vrijedi:

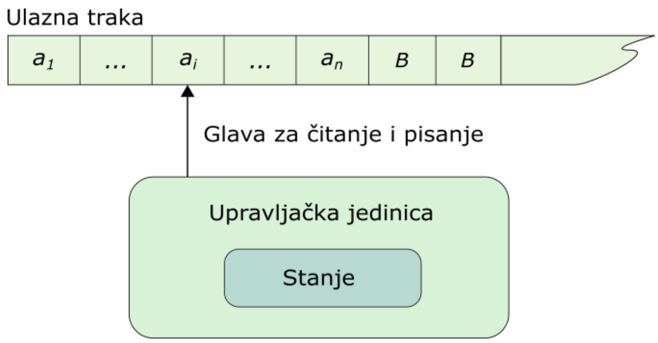
1) $|vx| \geq 1$
2) $|vwx| \leq n$
3) $uv^iwx^iy \in L, \forall i \geq 0$

21 TURINGOV STROJ

21.1 FORMALNA DEFINICIJA TURINGOVOG STROJA

- Rekurzivno prebrojivi jezici i Turingov stroj
- Osnovni model Turingovog stroja
- Donošenje odluka Turingovog stroja
- Formalna definicija Turingovog stroja

Jezik je **rekurzivno prebrojiv** ako i samo ako postoji **Turingov stroj** koji ga prihvaća. Time je definirana istovjetnost Turingovog stroja i rekurzivno prebrojivih jezika. Na slici 21.1 prikazan je osnovni model Turingovog stroja.



Slika 21.1 Model Turingovog stroja

Na temelju pročitanog znaka i stanja upravljačke jedinice **Turingov stroj donosi sljedeće odluke**:

- Koji znak se zapiše na traku umjesto pročitanog znaka
- U koje novo stanje prelazi upravljačka jedinica
- U koju stranu se miče glava za čitanje i pisanje

Turingov stroj (TS) formalno se definira uređenom sedmorkom:

$$ts = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \qquad (19)$$

gdje je:

Q	konačan skup stanja
$\Sigma \subseteq (\Gamma \backslash \{B\})$	konačan skup ulaznih znakova
Γ	konačan skup znakova trake
$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$	funkcija prijelaza, L i R označavaju smjer pomaka glave
$q_0 \in Q$	početno stanje
$B \in \Gamma$	znak kojim se označava prazna ćelija
$F \subseteq Q$	skup prihvatljivih stanja

21.2 PRIHVAĆANJE JEZIKA TURINGOVIM STROJEM

- Konfiguracija Turingovog stroja
- Prihvaćanje niza
- Prihvaćanje jezika
- Rekurzivni i rekurzivno prebrojivi jezici

Konfiguracija TS zadaje se sadržajem ćelija lijevo od glave, stanjem upravljačke jedinice i sadržajem ćelija koje su desno od glave: $\alpha_1 q \alpha_2$, gdje je $q \in Q$, α_1 je zapis na traci lijevo od glave za čitanje, a α_2 je zapis na traci desno od glave za čitanje ($\alpha_1, \alpha_2 \in \Gamma^*$). Izraz $J \overset{m}{\succ} K$ označava da se iz konfiguracije J prelazi u konfiguraciju K primjenom m prijelaza.
Niz w zapisuje se u krajnje lijevim ćelijama ulazne trake. TS je u početnom stanju q_0, a glava je postavljena na krajnje lijevi znak niza w. **Niz w se prihvaća** ako TS uđe u jedno od prihvatljivih stanja.

TS $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ **prihvaća jezik**:

$$L(M) = \{w \mid w \in \Sigma^* \wedge q_0 w \overset{*}{\succ} \alpha_1 p \alpha_2, p \in F, \alpha_1, \alpha_2 \in \Gamma^*\}$$

TS prihvaća **klasu rekurzivno prebrojivih jezika** (RPJ). Za bilo koji rekurzivno prebrojivi jezik moguće je izgraditi TS koji ispisuje sve nizove tog jezika.

Klasa rekurzivnih jezika (RekJ) je klasa jezika za koju je moguće izgraditi TS koji uvijek stane za bilo koji ulazni niz. Klasa rekurzivnih jezika je pravi podskup klase rekurzivno prebrojivih jezika.

21.3 CJELOBROJNA ARITMETIKA TURINGOVIM STROJEM

- Osnovna sintaksa zapisa
- Rekurzivne funkcije i jezici
- Primjer

Turingov stroj koristi se za računanje vrijednosti cjelobrojnih funkcija. Za cjelobrojnu funkciju s k argumenata $i_1, i_2, \dots, i_k$ na ulaznoj traci koristi se **notacija** $0^{i_1} 1 0^{i_2} 1 \dots 1 0^{i_k}$:

- Zapis koristi bazu $s = 1$ (rimska notacija).
- Broj znakova 0 označava vrijednost cijelog broja. Cijeli broj $i \geq 1$ zapiše se nizom 0^i.
- Nizovi nula odijeljeni su znakom 1.

Funkcije koje je moguće izračunati Turingovim strojem nazivamo **parcijalno rekurzivnim funkcijama**. Rekurzivno prebrojivi jezici i parcijalno rekurzivne funkcije su analogni u smislu da se prihvaćaju, odnosno da se računaju TS koji ne mora uvijek stati za bilo koji ulazni niz.

Ako je funkcija $f(i_1, i_2, \dots, i_k)$ definirana za sve argumente $i_1, i_2, \dots, i_k$, onda je ta funkcija **potpuno rekurzivna funkcija**. Rekurzivni jezici i potpuno rekurzivne funkcije su analogni u smislu da se prihvaćaju, odnosno da se računaju TS koji uvijek stane za bilo koji ulazni niz.

Primjer računanja funkcije TS je razlika brojeva m i n. Broj m se predstavi kao 0^m, a broj n kao 0^n. Brojevi su na ulaznoj traci zapisani u obliku niza $0^m 1 0^n$. TS čita znakove niza s lijeva i prvu nulu zamijeni praznom ćelijom B, prelazi na drugi dio niza (broj n nakon jedinice) gdje pronalazi prvu nulu te nju zamjenjuje jedinicom. Vraća se ponovno na prvu nulu u nizu, zamijenjuje je praznom ćelijom i nastavlja postupak sve dok u desnom podnizu postoje nule. Kada postupak završi, sve jedinice na desnoj strani niza TS zamijeni praznim ćelijama, a preostali broj nula u nizu predstavlja rezultat. Za slučaj $m \leq n$ rezultat je 0 i TS nule oba broja zamjenjuje praznim ćelijama.

22 SVOJSTVA TURINGOVOG STROJA

22.1 VIŠEKOMPONENTNA STANJA I ZNAKOVI TRAKE

- Višekomponentne oznake stanja
- Višekomponentni znakovi trake
- Primjeri

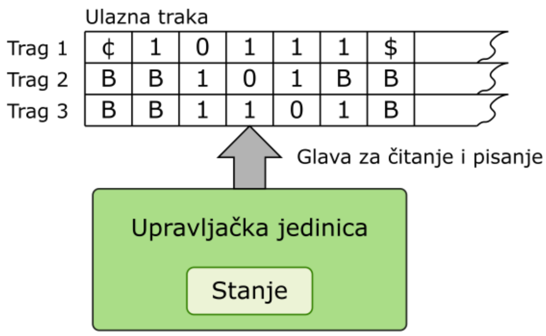
Oznake stanja i znakova TS razlažu se na više komponenata. Umjesto jedinstvene oznake stanja, koristi se složena oznaka koju čini više komponenata. Komponente složenih oznaka stanja pišu se u uglatim zagradama $[q_1, q_2, \dots, q_n]$.

Komponente stanja dijele se na:

- Upravljačke komponente – upravljaju radom TS i smije biti samo jedna takva komponenta u stanju.
- Radne komponente – koriste se za spremanje podataka.

Za **primjer** promotrimo jezik L koji sadrži nizove čiji se krajnje lijevi znak ne javlja nigdje u ostatku niza. TS $M = (Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], B, \{[q_1, B]\})$ koji prihvaća jezik L ima složena stanja $[q, a]$, gdje je q upravljačka, a a radna komponenta. Komponenta q može poprimiti vrijednost q_0 ili q_1. U stanju q_0, M pročita krajnje lijevi znak i spremi ga u radnu komponentu a, dok u stanju q_1 čita ostatak niza i uspoređuje svaki pročitani znak sa spremljenim znakom u komponenti a.

Znakovi trake i ulazni znakovi mogu imati više komponenata: $A_j = [a_1, a_2, \dots, a_n]$. Rad TS lakše je pratiti zapisivanjem pojedinih komponenti složenih znakova trake u zasebne tragove (slika 22.1).



Slika 22.1 Model TS s tri traga ulazni trake

Za **primjer** uzmimo model TS na slici 22.1 koji prihvaća jezik u kojem su prosti brojevi. Prvi trag je ulazni trag i sadrži binarni broj. Druga dva (pomoćna) traga koriste se za ispitivanje binarnog broja. Brojevi s ulaznog traga prepisuju se na drugi pomoćni trag i dijele s brojem na prvom pomoćnom tragu postupkom uzastopnog oduzimanja kako bi se ispitalo je li ulazni broj prost.

22.2 Proširenja i pojednostavljenja Turingovog stroja

- Proširenja trake, neizravni TS
- Stogovni stroj i stroj s brojilima
- Ograničenja stanja i trake
- Univerzalni TS

Šest osnovnih načina **proširenja** osnovnog modela TS su:

- TS s dvostranom beskonačnom trakom
- TS s višestrukim trakama
- Nedeterministički TS
- TS s višedimenzionalnim ulaznim poljem
- TS s više glava
- Neizravni TS

Neizravni TS koristi se u istraživanju prostorne složenosti prihvaćanja jezika i prostorne složenosti računanja cjelobrojnih funkcija. Ima više radnih traka i jednu ulaznu traku koju je moguće samo čitati.

Stogovni stroj je deterministički TS s jednom ulaznom trakom i više stogova. Ulaznu traku je moguće samo čitati. Stog je posebna radna traka pojednostavljenih funkcija prijelaza.

Stroj s brojilima je pojednostavljeni stogovni stroj s dva stoga koji umjesto ta dva stoga koristi četiri brojila. Pomak glave ograničava se uvođenjem oznaka dna stoga X i prazne ćelije B. Pomicanjem glave mijenja se vrijednost brojila.

Istodobnim **ograničenjem** broja stanja, broja traka i broja znakova trake, ograničava se broj različitih TS koje je moguće izgraditi. Zato takav TS ne prihvaća isti skup jezika kao osnovni model TS. Ako se ne ograniči broj znakova trake, dovoljna je jedna traka i tri stanja (od kojih jedno prihvatljivo) za prihvaćanje bilo kojeg rekurzivno prebrojivog jezika. Također, ako se ne ograniči broj stanja, za prihvaćanje bilo kojeg RPJ dovoljni su znakovi trake $\{0,1,B\}$.

Univerzalni TS omogućuje simuliranje rada bilo kojeg TS s jednom trakom. Univerzalni TS ima tri trake:

- Na prvu se zapišu kodirane funkcije prijelaza TS-a M i niz w.
- Sadržaj druge trake simulira sadržaj trake proizvoljnog TS-a.
- Na treću se zapiše stanje simuliranog TS-a.

22.3 Generiranje jezika Turingovim strojem

- Struktura TS za generiranje jezika
- Prihvaćanje jezika generiranog TS
- Generiranje jezika prihvaćenog TS
- Jednostavni i složeni TS

Za generiranje jezika koristi se TS s višestrukim trakama od kojih je jedna izlazna. Znak se na izlaznu traku zapiše trajno i nije ga moguće mijenjati. Glava se miče isključivo desno. Na traku se nizovi jezika $G(M)$ kojeg generira TS M ispisuju odvojeni graničnicima #.

Za bilo koji TS M_1 koji generira jezik $G(M_1)$ može se izgraditi TS M_2 koji **prihvaća** jezik $L(M_2) = G(M_1)$. M_2 ima jednu traku više od M_1. Dodatna traka je ulazna i na nju se zapiše niz koji se ispituje. M_2 simulira rad M_1 i uspoređuje generirani niz s nizom na ulaznoj traci. Ako su nizovi jednaki, M_2 stane i prihvati niz, inače generira sljedeći niz.

Za bilo koji TS M_2 koji prihvaća jezik $L(M_2)$ može se izgraditi TS M_1 koji **generira** jezik $G(M_1) = L(M_2)$.

Jednostavni TS koristi se za generiranje rekurzivnih jezika. Svi znakovi niza ispisuju se na radnu traku. Nakon ispisa niza w_i, TS M_1 simulira rad M_2 i provjeri ispisani niz. Ako je niz prihvatljiv, M_1 ga kopira na izlaznu traku.

Ako je niz rekurzivno prebrojiv, moguće je da za neki niz M_2 nikad ne stane. Zato se ne smije dozvoliti neograničen broj prijelaza tijekom simulacije.

Složeni TS koristi se za generiranje rekurzivno prebrojivih jezika. Složeni TS M_1 generira par cijelih brojeva (i, j), a zatim simulira rad M_2 za i-ti niz w_i primjenjujući najviše j prijelaza.

22.4 ISTOVJETNOST REKURZIVNOG JEZIKA I KANONSKOG SLIJEDA

- Definicija kanonskog slijeda
- Istovjetnost RekJ i kanonskog slijeda

U **kanonskom slijedu** kraći nizovi su ispred duljih nizova. Redoslijed nizova jednake duljine određuje se na temelju njihove numeričke vrijednosti. Baza za računanje određuje se na temelju $|\Sigma|$.

Rekurzivne jezike moguće je generirati kanonskim slijedom. Ako je jezik $L(M)$ rekurzivan, onda je za generiranje jezika moguće koristiti jednostavni TS koji nizove tog jezika generira na izlaznu traku onim redoslijedom kojim se ti nizovi generiraju na radnu traku.

Jezik $G(M)$ koji je moguće generirati kanonskim slijedom je rekurzivan.

Nije moguće izgraditi TS za opći slučaj konačnog jezika, ali je za bilo koji točno određeni konačni jezik moguće izgraditi zasebni TS koji ga prihvaća i uvijek stane za bilo koji ulazni niz.

23 Gramatika neograničenih produkcija

23.1 Formalna specifikacija gramatike neograničenih produkcija

- Oblik produkcije i vrsta gramatike
- Definiranje jezika generiranog gramatikom

Neograničene produkcije su oblika:

$$\alpha \to \beta$$

gdje su α i β nizovi nezavršnih i završnih znakova gramatike i $\alpha \neq \varepsilon$.

Gramatika neograničenih produkcija (gramatika tipa 0) je uređena četvorka $G = (V, T, P, S)$. Za produkciju $\alpha \to \beta$ gramatike G definirana je relacija $\Rightarrow$ na sljedeći način:

$$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$$

Relacija $\overset{*}{\Rightarrow}$ je refleksivno i tranzitivno okruženje relacije $\Rightarrow$.

Gramatika $G = (V, T, P, S)$ generira jezik koji pripada klasi rekurzivno prebrojivih jezika:

$$L(G) = \{w \mid w \in T^* \land S \overset{*}{\Rightarrow} w\}$$

23.2 Konstrukcija TS iz GNP

- Gramatika, jezik i TS
- Postupak rada automata
- Simulacija gramatike G

Ako **gramatika** neograničenih produkcija G generira **jezik** $L(G)$, onda je $L(G)$ rekurzivno prebrojiv jezik. Jezik $L(G)$ kojeg generira gramatika G je rekurzivno prebrojiv ako i samo ako postoji **TS** M koji prihvaća jezik $L(M) = L(G)$.

Nedeterministički TS M s dvije trake koji simulira rad gramatike $G = (V, T, P, S)$ gradi se na sljedeći način:

- Na prvu traku se zapiše niz znakova w.
- Na drugu traku zapiše se početni nezavršni znak gramatike S.
- Tijekom simulacije M na drugu traku ispisuje međunizove α koje generira gramatika G.
- M uspoređuje nizove α s nizom w. Ako je $\alpha = w$, prelazi u prihvatljivo stanje i prihvaća niz w.

Simulacija rada gramatike G izvodi se na sljedeći način:

1) TS nedeterministički izabere mjesto i u nizu α zapisanom na drugoj traci, gdje je $1 \leq i \leq |\alpha|$. Tako se istodobno izvode simulacije za sve moguće vrijednosti mjesta i.
2) TS nedeterministički izabere produkciju $\beta \to \gamma$ gramatike G. Tako se istodobno izvode simulacije za sve produkcije gramatike G.
3) Ako je na mjestu i niz β, onda se β zamijeni nizom γ.

4) Niz generiran na drugoj traci uspoređuje se s nizom w zapisanom na prvoj traci. Ako su nizovi jednaki, TS prihvaća w i zaustavlja rad, a u suprotnom nastavlja korakom 1.

23.3 KONSTRUKCIJA GNP IZ TS

- Pristup definiranju gramatike
- Veza TS i gramatike

Ako TS M prihvaća rekurzivno prebrojiv jezik $L(M)$, onda postoji GNP G koja generira jezik $L(G) = L(M)$.

Gramatika $G = (V, T, P, S)$ koja **simulira rad** TS $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ gradi se na sljedeći način:

- G generira redom međunizove znakova koji predstavljaju konfiguracije TS M.
- Nezavršni znak q u međunizu generiranom gramatikom G predstavlja oznaku stanja TS.
- Početna konfiguracija TS simulira se međunizom s nezavršnim znakovima oblika $[a_i, a_i]$, gdje je $a_i \in \Sigma$. Međuniz ima oblik $q_0[a_1, a_1][a_2, a_2] \dots [a_n, a_n]$.
- Prva komponenta nezavršnog znaka čuva znakove niza tijekom simulacije, dok druga komponenta predstavlja znakove trake TS.
- Na temelju znakova sačuvanih u prvoj komponenti gramatika generira niz ako i samo ako TS prihvaća isti taj niz.

24 SVOJSTVA REKURZIVNIH JEZIKA

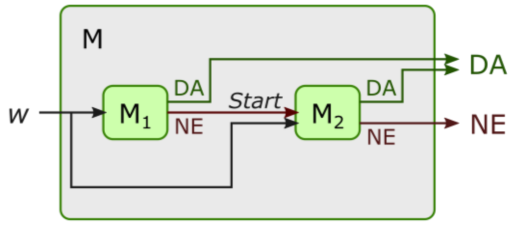
24.1 ZATVORENOST REKJ I RPJ S OBZIROM NA UNIJU

- Unija rekurzivnih jezika
- Unija rekurzivno prebrojivih jezika

Unija rekurzivnih jezika je rekurzivni jezik.

Neka TS M_1 prihvaća RekJ L_1 i neka TS M_2 prihvaća RekJ L_2. Tada je moguće simulirati rad TS M koji prihvaća uniju ta dva jezika $L_1 \cup L_2$ i to serijskim spojem automata (slika 24.1):

- Ako M_1 stane i prihvati niz, M stane i prihvati niz.
- Ako M_1 stane i ne prihvati niz, M pokrene simulaciju rada M_2.
- Ako M_2 stane i prihvati niz, M stane i prihvati niz.
- Ako M_2 stane i ne prihvati niz, M stane i ne prihvati niz.

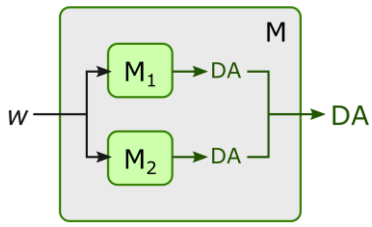


Slika 24.1 Struktura TS M koji prihvaća uniju RekJ $L_1 \cup L(M_2)$

Unija rekurzivno prebrojivih jezika je rekurzivno prebrojiv jezik.

Neka TS M_1 prihvaća RPJ L_1 i neka TS M_2 prihvaća RPJ L_2. Zbog mogućnosti da M_1 nikad ne stane za neki niz, kod RPJ koristi se paralelna simulacija tako da se gradi TS M koji na zasebnim trakama istodobno simulira rad M_1 i rad M_2 (slika 24.2):

- Ako M_1 stane i prihvati niz, M stane i prihvati niz.
- Ako M_2 stane i prihvati niz, M stane i prihvati niz.



Slika 24.2 Struktura TS M koji prihvaća uniju dva RPJ L_1 i L_2

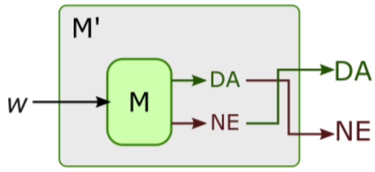
24.2 ZATVORENOST REKJ I RPJ S OBZIROM NA KOMPLEMENT

- Komplement rekurzivnih jezika
- Komplement rekurzivno prebrojivih jezika

Komplement rekurzivnog jezika je rekurzivni jezik.

Ako je L rekurzivan jezik, onda postoji TS M koji ga prihvaća i koji uvijek stane za bilo koji ulazni niz w. Za prihvaćanje komplementa RekJ $L' = L^c$ gradi se TS M' koji simulira rad TS M (slika 24.3):

- Ako M stane i prihvati niz, M' stane i ne prihvati niz.
- Ako M stane i ne prihvati niz, M' stane i prihvati niz.



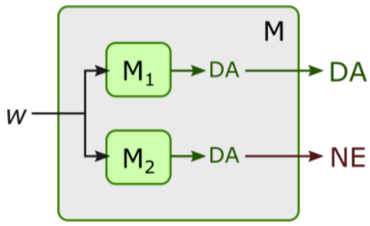
Slika 24.3 Struktura TS M' koji prihvaća komplement rekurzivnog jezika $L(M)$

Ako su jezik L i njegov **komplement** L^c **rekurzivno prebrojivi**, onda su oba jezika rekurzivna.

Neka TS M_1 prihvaća RPJ L_1 i neka TS M_2 prihvaća RPJ $L_2 = {L_1}^c$. TS M gradi se kao paralelni spoj M_1 i M_2 (slika 24.4):

- M prihvaća niz w ako i samo ako M_1 prihvaća niz w.
- M ne prihvaća niz w ako i samo ako M_2 prihvaća niz w.

Budući da je niz w ili u L_1 ili u L_2, TS M uvijek stane i jednoznačno odluči prihvaća li se niz. Dakle, L_1 i L_2 su rekurzivni jezici.



Slika 24.4 Shema dokaza da je jezik $L(M)$ rekurzivan ako su jezici $L = L_1$ i $L^c = L_2$ rekurzivno prebrojivi

25 IZRAČUNLJIVOST I ODLUČIVOST

25.1 IZRAČUNLJIVOST

- Definicija izračunljivosti
- Church-Turingova hipoteza
- Problem kodiranja: RAM, dijagonalni jezik

Problem je **izračunljiv** ako postoji automat koji mehaničkim postupkom korak po korak rješava zadani problem. Ova **definicija izračunljivosti** je intuitivna, ne daje nikakva ograničenja, dana je u najširem obliku i jedino što zahtijeva je mogućnost raščlambe problema na korake.

Church-Turingova hipoteza tvrdi da su izračunljive i parcijalno rekurzivne funkcije istovjetne:

> *Parcijalno rekurzivne funkcije su izračunljive jer je za njihovo računanje moguće izgraditi TS koji ih mehaničkim putem računa korak po korak primjenom zadanih funkcija prijelaza.*

Rad apstraktnog računala **RAM** moguće je simulirati primjenom TS s višestrukim trakama. Prva traka koristi se za spremanje adresa i sadržaja memorijskih ćelija u obliku:

$$\#0^*v_0\#1^*v_1\ \#10^*v_2\#\ldots\#k^*v_k$$

gdje je $i \in [0, k]$ adresa memorijske ćelije, a v_i sadržaj memorijske ćelije. Još se tri trake koriste za spremanje sadržaja radnih registara, programskog brojila i adresnog registra.

Dijagonalni jezik[5] L_d je neizračunljivi jezik za koji nije moguće izgraditi TS M koji ga prihvaća. Gradnja dijagonalnog jezika L_d započinje kodiranjem funkcija prijelaza proizvoljnog TS-a. Za kodiranje se koristi model TS-a $M = (Q,\ \{0,1\},\ \{0,1,B\},\ \delta,\ q_1,\ B,\ \{q_2\})$. Funkcija prijelaza $\delta\left(q_i, X_j\right) = (q_k, X_l, D_m)$ kodira se u oblik $0^i10^j10^k10^l10^m$. Binarni kod za TS M je:

$$111k\hat{o}d_1 11k\hat{o}d_2 11\ldots 11k\hat{o}d_r 111$$

gdje su $k\hat{o}d_p$ kodovi funkcije prijelaza.

25.2 ODLUČIVOST

- Definicija odlučivosti
- Univerzalni TS i jezik

Rekurzivni jezici su **odlučivi** jer ih prihvaćaju TS koji uvijek stanu i odluče o prihvaćanju ili neprihvaćanju niza. Za rekurzivno prebrojive jezike ne postoji TS koji uvijek stane, pa oni nisu odlučivi.

	Izračunljiv	Odlučiv
RekJ	DA	DA
RPJ	DA	NE

Tablica 25.1 Izračunljivost i odlučivost RekJ i RPJ

[5] Za one koji žele znati više ili barem razumjeti problem dijagonalnog jezika, a s obzirom da je objašnjenje u udžbeniku prof. Srbljića katastrofa svih katastrofa, preporučujemo ovu lekciju profesora Dana Gusfielda.

Univerzalni TS M_u ima tri trake:

- Prva traka je ulazna i sadrži niz $< M, w >$ kodiran u obliku

$$111k\hat{o}d_1 11k\hat{o}d_2 11 \dots 11k\hat{o}d_r 111w$$

  gdje su $k\hat{o}d_p$ kodovi funkcije prijelaza TS M, a w je ulazni niz.

- Sadržaj druge trake simulira sadržaj trake TS M.
- Na treću traku zapisuje se stanje q_i TS M nizom 0^i.

TS M_u prihvaća niz $< M, w >$ ako i samo ako TS M prihvaća niz w. Univerzalni TS prihvaća **univerzalni jezik** L_u:

$$L_u = \{< M, w > \mid M\ prihva\acute{c}a\ w\}$$

26 KONTEKSTNO OVISNI JEZICI

26.1 DEFINICIJA KONTEKSTNO OVISNOG JEZIKA I GRAMATIKE

- Kontekstno ovisni jezik i gramatika
- Oblik produkcija
- Primjer

Jezik je **kontekstno ovisan** (KOJ) ako i samo ako postoji **kontekstno ovisna gramatika** (KOG) koja ga generira. Time je definirana istovjetnost kontekstno ovisne gramatike i kontekstno ovisnih jezika. Klasa kontekstno ovisnih jezika pravi je podskup rekurzivnih jezika ($KOJ \subset RekJ$).

Oblik produkcija kod KOG $G = (V, T, P, S)$ je sljedeći:

$$\alpha \to \beta$$

gdje su α i β nizovi nezavršnih i završnih znakova gramatike te vrijedi $|\alpha| \leq |\beta|$ i $\alpha \neq \varepsilon$. Naziv kontekstno ovisna gramatika dolazi od normalnog oblika produkcija:

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

gdje su $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$, $A \in V$ i $\beta \neq \varepsilon$.

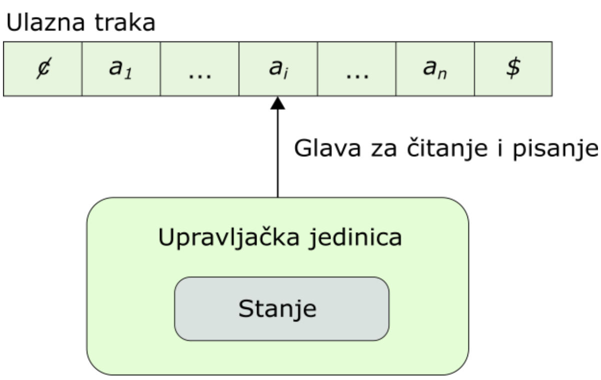
Primjer KOG je gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$ koja generira KOJ $L(G) = \{a^n b^n c^n \mid n \geq 1\}$. Skup P sadrži sljedeće produkcije:

1) $S \to aSBC$
2) $S \to aBC$
3) $CB \to BC$
4) $aB \to ab$
5) $bB \to bb$
6) $bC \to bc$
7) $cC \to cc$

26.2 LINEARNO OGRANIČEN AUTOMAT

- Model LOA
- Formalna definicija LOA
- Naziv LOA

Linearno ograničen automat (LOA) je nedeterminstički TS koji koristi samo onaj dio trake na kojem je zapisan ulazni niz w (slika 26.1). Središnje ćelije koriste se za spremanje niza w, a krajnje dvije ćelije trake sadrže lijevi graničnik ¢ i desni graničnik \$. Zabranjuje se pomak glave izvan označenog dijela.



Slika 26.1 Model linearno ograničenog automata

Linearno ograničen automat (LOA) **formalno se definira** uređenom osmorkom:

$$loa = (Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F) \tag{20}$$

gdje se $Q, \Sigma, \Gamma, \delta, q_0$ i F zadaju na isti način kao i za nedeterministički TS, dok su graničnici ¢ i \$ znakovi u skupu ulaznih znakova Σ.

LOA $M = (Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F)$ prihvaća jezik:

$$L(M) = \{w \mid w \in (\Sigma \backslash \{¢, \$\})^* \wedge q_0 ¢ w \$ \overset{*}{\succ} \alpha q \beta, q \in F\}$$

LOA mora stati u slučaju prihvaćanja niza. Graničnici ¢ i \$ nalaze se u skupu Σ, ali nisu dio ulaznog niza w. Ako je za prihvaćanje jezika L moguće izgraditi deterministički LOA (DLOA), onda je jezik L deterministički kontekstno ovisan jezik. Oznaka LOA podrazumijeva nedeterministički LOA.

Naziv LOA nastao je na temelju svojstva:

> Ako je duljina radne trake TS M ograničena za bilo koji niz w linearnom funkcijom $\mathrm{f(w)}$, onda je moguće izgraditi istovjetni TS $\mathrm{M'}$ koji koristi samo onaj dio trake na koji je zapisan ulazni niz w.

26.3 KONSTRUKCIJA LINEARNO OGRANIČENOG AUTOMATA IZ KOG

- Postupak izgradnje
- Konstrukcija LOA iz KOG

Ako KOG $G = (V, T, P, S)$ generira KOJ $L(G) \backslash \{\varepsilon\}$, tada je moguće izgraditi LOA M koji prihvaća KOJ:

$$L(M) = L(G)$$

Postupak izgradnje LOA za jezik zadan KOG-om sličan je postupku gradnje TS zadanog gramatikom neograničenih produkcija. LOA M koristi dva traga ulazne trake – u gornji trag zapiše niz $¢w\$$, a na početak donjega traga M zapiše početni nezavršni znak S gramatike G.

Za prazni niz ε LOA odmah stane i ne prihvati niz. Za neprazni niz w LOA simulira gramatiku G tako da na donjem tragu generira nizove primjenom produkcija gramatike G koje uspoređuje sa zadanim nizom w. LOA M prihvaća niz w ako i samo ako gramatika G generira niz w.

Tijekom rada LOA M ne koristi veći broj ćelija od duljine niza w. Ako se u postupku izgradnje generira međuniz α, $|\alpha| > |w|$, prekida se rad LOA jer se iz α sigurno ne može generirati w s obzirom da su kod KOG-a lijeve strane produkcija kraće od desnih.

27 SVOJSTVA KONTEKSTNO OVISNIH JEZIKA

27.1 UNIJA, NADOVEZIVANJE I KLEENEE

- Zatvorenost KOJ po uniji
- Zatvorenost KOJ po nadovezivanju
- Zatvorenost KOJ po Kleenee

Neka KOG $G_1 = (V_1, T_1, P_1, S_1)$ i $G_2 = (V_2, T_2, P_2, S_2)$ generiraju KOJ $L(G_1)$ i $L(G_2)$ pri čemu vrijedi $V_1 \cap V_2 = \emptyset$.

Unija kontekstno ovisnih jezika je kontekstno ovisni jezik.

KOG $G_3 = (V_3, T_3, P_3, S_3)$ koja generira jezik $L(G_3) = L(G_1) \cup L(G_2)$ konstruira se na sljedeći način:

1) $V_3 = V_1 \cup V_2 \cup \{S_3\},\ S_3 \notin V_1 \cup V_2$
2) $T_3 = T_1 \cup T_2$
3) $P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 | S_2\}$

Nakon prijelaza iz S_3 za generiranje niza w koriste se produkcije samo jedne gramatike G_1 ili G_2.

Nadovezivanje kontekstno ovisnih jezika je kontekstno ovisni jezik.

Kod nadovezivanja gramatika G_3 generira jezik $L(G_3) = L(G_1)L(G_2)$. V_3, T_3 konstruiraju se na isti način kao kod unije, dok je $P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 S_2\}$. Nakon prijelaza u S_1 i S_2 za generiranje niza w koriste se po dijelovima produkcije gramatika G_1 i G_2. G_3 generira niz γδ na sljedeći način:

$$S_3 \underset{G_3}{\Rightarrow} S_1 S_2 \overset{*}{\underset{G_1}{\Rightarrow}} \gamma S_2 \overset{*}{\underset{G_2}{\Rightarrow}} \gamma\delta$$

S obzirom da se ne može osigurati $T_1 \cap T_2 = \emptyset$, uvodi se ograničenje da na lijevoj strani produkcija mogu biti samo nezavršni znakovi gramatike.

Kontekstno ovisni jezici zatvoreni su s ozbirom na **Kleeneov operator L^+**.

Neka gramatika $G = (V, T, P, S)$ generira KOJ $L(G)$ i neka su na lijevim stranama isključivo nezavršni znakovi. Za izgradnju gramatike $G_1 = (V_1, T_1, P_1, S_1)$ koja generira jezik $L(G_1) = L(G)^+$ prvo konstruiramo pomoćnu gramatiku $G' = (V', T, P', S')$ tako da se nezavršni znakovi A gramatike G zamijene novim nezavršnim znakovima A'. Zamjena znakova osigurava $V \cap V' = \emptyset$. Gramatika G_1 konstruira se na sljedeći način:

1) $V_1 = V \cup V' \cup \{S_1, S_1'\},\ S_1, S_1' \notin V \cup V'$
2) $T_1 = T$
3) $P_1 = P \cup P' \cup \{S_1 \to S S_1' | S,\ S_1' \to S' S_1 | S'\}$

27.2 PRESJEK I KOMPLEMENT

- Zatvorenost KOJ po presjeku
- Zatvorenost KOJ po komplementu

Presjek kontekstno ovisnih jezika je kontekstno ovisni jezik.

Neka LOA M_1 prihvaća jezik $L(M_1)$, a neka LOA M_2 prihvaća jezik $L(M_2)$. LOA M_3 koji prihvaća jezik $L(M_3) = L(M_1) \cap L(M_2)$ ima dva traga ulazne trake na koje zapisuje isti niz završnih znakova w. LOA M_3 simulira rad M_1 na gornjem tragu ulazne trake, dok se rad M_2 simulira na donjem tragu.

Ako M_1 stane i prihvati niz, M_3 tada pokrene simulaciju M_2. Ako M_2 stane i prihvati niz, simulacija se zaustavlja i M_3 prihvati niz. Da M_1 ili M_2 stanu i ne prihvate niz, M_3 također stane i ne prihvati niz.

Komplement determinističkog kontekstno ovisnog jezika je deterministički kontekstno ovisni jezik.

Neka DLOA M prihvaća deterministički KOJ $L(M)$. Istovjetni DLOA M' koji uvijek stane za bilo koji ulazni niz w konstruira se na sljedeći način. Tijekom simulacije M' broji pomake glave M.

1) Ako M stane i prihvati niz, M' stane i prihvati niz.
2) Ako M stane i ne prihvati niz, M' stane i ne prihvati niz.
3) Ako M' odbroji više od $s(n+2)t^n$ pomaka glave M, M' stane i ne prihvati niz ($s = |Q|$, $n = |w|$, $t = |\Gamma|$).

DLOA M^c koji prihvaća jezik $L(M^c)$ koji je komplement jezika $L(M')$, gradi se zamjenom odluka o prihvaćanju i ne prihvaćanju niza u prethodnim pravilima.

27.3 ODLUČIVOST KONTEKSTNO OVISNIH JEZIKA

- Svojstvo odlučivosti KOJ

Svaki kontekstno ovisni jezik je rekurzivni jezik.

Neka je KOJ zadan pomoću KOG $G = (V, T, P, S)$ i neka je w ulazni niz. Algoritam prihvaćanja jezika zasniva se na gradnji usmjerenog grafa:

- Čvorovi grafa su nizovi završnih i nezavršnih znakova α za koje vrijedi $|\alpha| \leq |w|$.
- Ako vrijedi relacija $\alpha \Rightarrow \beta$, čvorovi grafa α i β povezuju se usmjerenom granom.
- Put usmjerenog grafa predstavlja postupak generiranja niza primjenom produkcija G.
- Put od čvora S do čvora w postoji ako i samo ako gramatika G generira niz w.

Budući da je konačan broj nizova α koji su jednako dugački ili kraći od ulaznog niza w, TS u konačnom broju koraka izgradi usmjereni graf i pretraži sve putove usmjerenog grafa. TS se uvijek zaustavi za bilo koji ulazni niz i odluči o prihvaćanju niza, što dokazuje da su kontekstno ovisni jezici rekurzivni, odnosno **odlučivi**.

28 STRUKTURNA SLOŽENOST JEZIKA

28.1 KLASE I HIJERARHIJA JEZIKA

- Klase jezika
- Hijerarhija Chomskog
- Definicije istovjetnosti

Neka su A i B dvije **klase jezika**. Ako je klasa A pravi podskup klase B, onda za te dvije klase vrijedi:

- Automat koji prihvaća jezike klase A jednostavnije je strukture od automata koji prihvaća jezike klase B.
- Produkcije gramatike koja generira jezike klase A jednostavnije su od produkcija gramatike koja generira jezike klase B.

Hijerarhija Chomskog (slika 28.1) je hijerarhija klasa jezika nad zadanom abecedom.

Slika 28.1 Hijerarhija Chomskog

Postoje sljedeće **istovjetnosti**:

- Regularna gramatika → regularni jezik → konačni automat
- Kontekstno neovisna gramatika → kontekstno neovisni jezik → potisni automat
- Kontekstno ovisna gramatika → kontekstno ovisni jezik → linearno ograničeni automat
- Gramatika neograničenih produkcija → Rekurzivno prebrojivi jezik → Turingov stroj

28.2 HIJERARHIJA GRAMATIKA I AUTOMATA

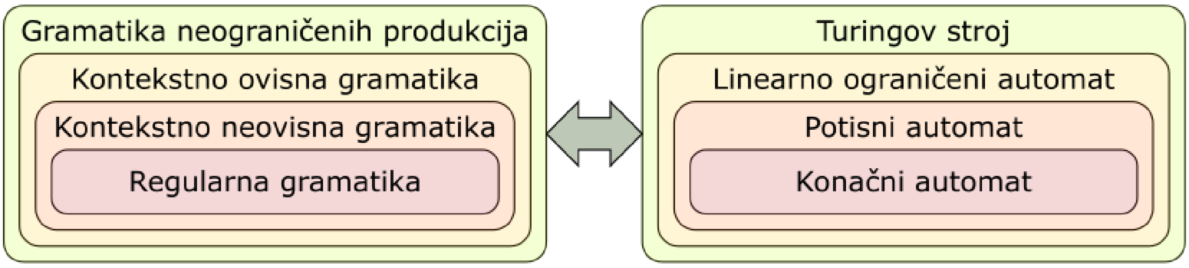
- Jezici i istovjetnosti
- Hijerarhija gramatika i automata
- Oblici produkcija i definicije automata

Hijerarhija gramatika i automata temelji se na hijerarhiji **jezika** i **istovjetnostima**:

- Regularnog jezika, konačnog automata i regularne gramatike
- Kontekstno neovisnog jezika, kontekstno neovisne gramatike i potisnog automata
- Kontekstno ovisnog jezika, kontekstno ovisne gramatike i linearno ograničenog automata

- Rekurzivno prebrojivog jezika, gramatike neograničenih produkcija i Turingovog stroja

Hijerarhija gramatika i automata prikazana je na slici 28.2.



Slika 28.2 Hijerarhija gramatika i automata

Oblici produkcija i definicije automata:

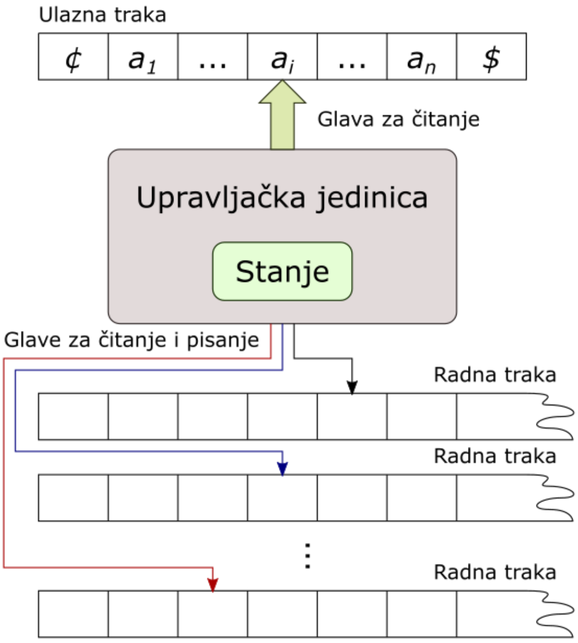
- GNP G_0 s produkcijama oblika $\alpha \rightarrow \beta, \alpha \neq \varepsilon$ odgovara TS $M_0 = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.
- KOG G_1 s produkcijama oblika $\alpha \rightarrow \beta$, $\alpha \neq \varepsilon$, $|\alpha| \leq |\beta|$ odgovara LOA $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F)$.
- KNG G_2 s produkcijama oblika $A \rightarrow \beta$ odgovara PA $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.
- Regularnoj gramatici G_3 s produkcijama oblika $A \rightarrow wB$ i $A \rightarrow w$ ili $A \rightarrow Bw$ i $A \rightarrow w$ odgovara KA $M_3 = (Q, \Sigma, \delta, q_0, F)$.

29 SLOŽENOST PRIHVAĆANJA JEZIKA

29.1 PROSTORNA SLOŽENOST

- Korištenje modela TS
- Model za prostornu složenost
- Definicija prostorne složenosti

Za ocjenu prostorne složenosti prihvaćanja **koristi se** neizravni deterministički TS s k polubeskonačnih traka (**slika 29.1**). Ulazna traka sadrži niz koji se ispituje i može se samo čitati.



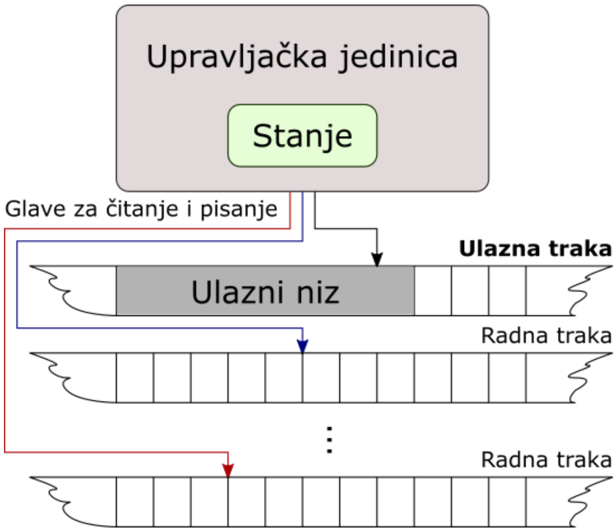
Slika 29.1 Model neizravnog TS za ocjenu prostorne složenosti računanja

Prostorna složenost određuje se na temelju samo jedne radne trake, i to one na kojoj TS koristi najviše ćelija. Koristi li TS M najviše $S(n)$ ćelija na jednoj od radnih traka tijekom prihvaćanja bilo kojeg niza duljine n, TS M je prostorne složenosti $S(n)$. Za bilo koji TS s k radnih traka moguće je izgraditi istovjetni TS koji koristi samo jednu traku i iste je prostorne složenosti. Jezik $L(M)$ koji prihvaća TS M je prostorne složenosti $S(n)$.

29.2 VREMENSKA SLOŽENOST

- Model za vremensku složenost
- Definicija vremenske složenosti

Model za ocjenu vremenske složenosti računanja prikazan je na slici 29.2. TS ima k dvostrano beskonačnih traka. Jedna radna traka je ulazna i na njoj je zapisan ulazni niz duljine n. Sve je trake moguće čitati i po svim je trakama moguće pisati.



Slika 29.2 Model TS s višestrukim dvostranim beskonačnim trakama za ocjenu vremenske složenosti računanja

Vremenska složenost računa se na temelju broja pomaka glave za čitanje i pisanje. Izvede li TS M najviše $T(n)$ pomaka glave tijekom prihvaćanja bilo kojeg niza duljine n, TS M je vremenske složenosti $T(n)$. Jezik $L(M)$ koji prihvaća TS M je vremenske složenosti $T(n)$.

Budući da se čitaju svi znakovi niza vremenska složenost je $max(n+1, \lceil T(n) \rceil)$, što se kraće zapisuje $T(n)$.

29.3 BROJ TRAKA I SLOŽENOST

- Svojstva složenosti
- Broj traka i prostorna složenost
- Broj traka i vremenska složenost

Broj traka nema utjecaja na prostornu složenost, već samo na vremensku složenost. Vremenska složenost se povećava smanjivanjem broja traka. Konstantne faktore u obje funkcije složenosti moguće je zanemariti.

Ako TS M_1 s k radnih traka **prostorne složenosti** $S(n)$ prihvaća jezik $L(M_1)$, onda postoji TS M_2 s jednom radnom trakom koji je jednake prostorne složenosti $S(n)$ i prihvaća jezik $L(M_2) = L(M_1)$.

Neka M_1 na jednoj od radnih traka koristi najviše $S(n)$ ćelija. Istovjetni TS M_2 sa samo jednom trakom ima $2k$ tragova – prvih k tragova je za spremanje sadržaja traka M_1, a ostali za bilježenje položaja glava M_1.

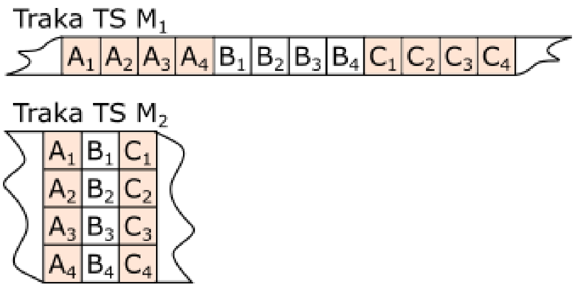
Ako TS M_1 s k radnih traka **vremenske složenosti** $T(n)$ prihvaća jezik $L(M_1)$, onda postoji TS M_2 s jednom radnom trakom koji je vremenske složenosti $T^2(n)$ i prihvaća jezik $L(M_2) = L(M_1)$.

29.4 SAŽIMANJE PROSTORA I UBRZANJE VREMENA

- Sažimanje prostora
- Ubrzanje vremena

Primjer **sažimanja** prostora prikazan je na slici 29.3.

Ako TS M_1 s k radnih traka prostorne složenosti $S(n)$ prihvaća jezik $L(M_1)$, onda za bilo koju konstantu $c > 0$ postoji TS M_2 prostorne složenosti $cS(n)$ koji prihvaća jezik $L(M_2) = L(M_1)$.



Slika 29.3 Sažimanje prostora za $c = 4$

Ako TS M_1 s k radnih traka **vremenske složenosti** $T(n)$ prihvaća jezik $L(M_1)$, onda za bilo koju konstantu $c > 0$ postoji TS M_2 s k traka vremenske složenosti $cT(n)$ koji prihvaća jezik $L(M_2) = L(M_1)$, gdje je $k > 1$ i $inf_{n\to\infty} \frac{T(n)}{n} = \infty$. Funkcija $inf_{n\to\infty} f(n)$ je najveća donja granica funkcije $f(n)$ kada n teži u beskonačnost.

30 KLASE JEZIKA PO SLOŽENOSTI

30.1 MODEL SLOŽENOSTI I ODNOSI MEĐU KLASAMA

- Model složenosti i TS
- Osnovne klase složenosti
- Odnosi među klasama složenosti

Nedeterministički **TS** je **prostorne složenosti** $S(n)$ ako za niti jedan niz duljine n niti jedan mogući slijed prijelaza na niti jednoj od radnih traka ne koristi više od $S(n)$ ćelija.

Nedeterministički **TS** je **vremenske složenosti** $T(n)$ ako za niti jedan niz duljine n niti jedan mogući slijed prijelaza ne pomakne glavu više od $T(n)$ puta.

Četiri su **osnovne klase složenosti**:

- $DTIME(T(n))$ – jezici determinističke vremenske složenosti $T(n)$
- $DSPACE\big(S(n)\big)$ – jezici determinističke prostorne složenosti $S(n)$
- $NTIME(T(n))$ – jezici nedeterminističke vremenske složenosti $T(n)$
- $NSPACE(S(n))$ – jezici nedeterminističke prostorne složenosti $S(n)$

Odnosi među klasama složenosti su sljedeći:

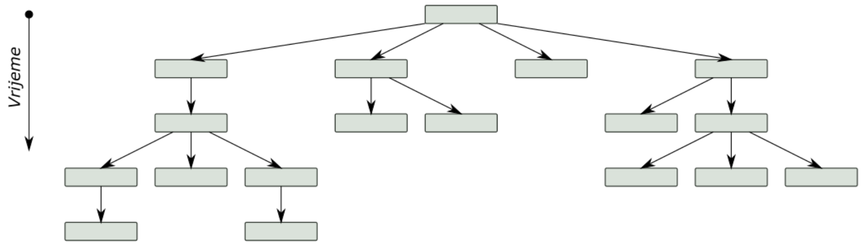
- Ako je jezik L u klasi $DTIME(f(n))$, onda je L i u klasi $DSPACE(f(n))$.
- Ako je jezik L u klasi $DSPACE(f(n))$ i ako je $f(n) \geq \log_2 n$, onda je L i u klasi $DTIME(c^{f(n)})$. Vrijednost konstante c ovisi o jeziku L.
- Ako je jezik L u klasi $NTIME(f(n))$ i ako je $f(n) \geq \log_2 n$, onda je L i u klasi $DTIME(c^{f(n)})$. Vrijednost konstante c ovisi o jeziku L.
- Ako je jezik L u klasi $NSPACE(f(n))$ i ako je funkcija $f(n) \geq \log_2 n$ potpuno prostorno izgradiva, onda je L i u klasi $DSPACE(f^2(n))$.

30.2 VREMENSKA SLOŽENOST DTS I NTS I PROSTORNA IZGRADIVOST

- Opis NTS stablom
- Vremenska složenost i stablo
- Prostorna izgradivost $S(n)$
- Svojstva hijerarhije

Primjer **stabla** vremenskog modela rada nedeterminističkog TS prikazan je na slici 30.1. Čvorovi grafa su konfiguracije TS. Korijen je početna konfiguracija, a grane određuju prijelaze u skup konfiguracija definiranih funkcijom prijelaza.

Vremenska složenost NTS proporcionalna je vremenu slijednog obilaženja stabla u cilju pronalaženja barem jedne konfiguracije u listovima stabla koja prihvaća zadani niz.



Slika 30.1 Stablo vremenskog modela rada NTS

Neka je TS M prostorne složenosti $S(n)$. Funkcija $S(n)$ je **prostorno izgradiva** ako za bilo koji n postoji barem jedan niz duljine n za koji TS M koristi svih $S(n)$ ćelija.

Svojstva hijerarhije jezika:

- Hijerarhija je beskonačna.
- Za potpuno prostorno i vremenski izgradive funkcije hijerarhija je neprekinuta.
- Za neke jezike ne postoji TS koji ih prihvaća u minimalnom vremenu ili prostoru.
- Za uniju klasa jezika moguće je odrediti funkciju složenosti tako da obuhvati sve jezike iz unije.

30.3 Klasa jezika polinomne složenosti

- Odlučivost i jezici polinomne složenosti
- Označivanje
- Odnosi determinističke i nedeterminističke polinomne složenosti

Ne postoji učinkovit način prihvaćanja jezika koji su izračunljivi i **odlučivi**. Značajna je klasa jezika determinističke **polinomne složenosti**. Klasa jezika polinomne vremenske složenosti **označava** se znakom P.

Klasa jezika P definira se na sljedeći način:

$$P = \bigcup_{i \geq 1} DTIME(n^i)$$

Na temelju svojstva unije klasa jezika, u klasi jezika P su svi oni jezici za koje postoji deterministički TS polinomne vremenske složenosti n, n^2, n^3, ...

Mnogi značajni jezici nisu u klasi P, ali za njih postoji **nedeterministički TS** polinomne vremenske složenosti. Klasa jezika nedeterminističke polinomne složenosti **označava** se oznakom NP:

$$NP = \bigcup_{i \geq 1} NTIME(n^i)$$

INDEKS KRATICA

ASPA	Accept State Pushdown Automata
CFG	Context Free Grammar
CNF	Chomsky Normal Form
DKA	Deterministički Konačni Automat
DLG	Desno Linearna Gramatika
DLOA	Deterministički Linearno Ograničeni Automat
DPA	Deterministički Potisni Automat
ESPA	Empty Stack Pushdown Automata
GNF	Greibach Normal Form
GNP	Gramatika Neograničenih Produkcija
JP	Jezični Procesor
KNG	Kontekstno Neovisna Gramatika
KNJ	Kontekstno Neovisni Jezik
KOG	Kontekstno Ovisna Gramatika
KOJ	Kontekstno Ovisni Jezik
LIFO	Last In First Out
LLG	Lijevo Linearna Gramatika
LOA	Linearno Ograničeni Automat
NKA	Nedeterministički Konačni Automat
NTS	Nedeterministički Turingov Stroj
PA	Potisni Automat (Pushdown Automata)
RAM	Random Access Machine
RekJ	Rekurzivni Jezik
RI	Regularni Izraz
RJ	Regularni Jezik
RPJ	Rekurzivno Prebrojivi Jezik
TS	Turingov Stroj
ε-NKA	Epsilon Nedeterministički Konačni Automat

Za sve sugestije i ispravke javite se na glisoc@gmail.com.

Zadnji put ažurirano: 30. kol. 17.