1. UVODNA RAZMATRANJA

1.1. Jezični procesor

- uloga jezičnog procesora
- jezici u postupku prevođenja
- definicija jezika

Osnovna uloga: jezični procesor prevodi zapis algoritma iz izvornog jezika $\mathbf{L_i}$ u zapis algoritma u ciljnom jeziku $\mathbf{L_c}$ koji je moguće izvesti na zadanom računalu.

Definicija jezičnog procesora zasnovana je na tri **jezika**: izvornom, ciljnom i jeziku izgradnje.

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

JP - jezični procesor
L_i - izvorni jezik
L_c - ciljni jezik
L_g - jezik izgradnje (najčešće $L_c = L_g$)

L_{svi} – 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 – matematički model automata koji odlučuje da li niz na ulazu pripada zadanom jeziku.
Formalna gramatika G – je matematička struktura koja koristeći skupove produkcija generura nizove znakova koji pripadaju jeziku.

Razine rada podijeljene su u dvije **osnovne faze** rada jezičnog procesora:
1) Faza analize:
    1. razina: **leksička analiza**
    2. razina: **sintaktička i semantička analiza -** generiranje **višeg međukoda**
2) Faza sinteze:
    3. razina: prevođenje višeg u **srednji međukod**
    4. razina: prevođenje srednjeg u **niži međukod**
    5. razina: prevođenje nižeg međukoda u **ciljni program**

Sintaksa: definira jezik kao skup svih dozvoljenih nizova leksičkih jedinki; način gradnje izraza od leksema, naredbi...

Semantika: određuje skup dozvoljenih značenja (semantička pravila povezuju ponašanje računala s izvođenjem programa).

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.
Niz: je konačni slijed znakova abecede postavljenih jedan iza drugog; ε prazni niz.
Nadovezivanje nizova: označava se potencijom te vrijedi:

$$w^0 = \varepsilon \; ; \; w^i = w^{i-1} w \; ; \; w^1 = w^0 w = \varepsilon w = w.$$

Dijelovi niza: prefiks, sufiks, podniz, podslijed…
Formalni jezik: skup nizova nad abecedom.
Operacije nad jezicima:

1. Unija jezika L i N — $L \cup N$
2. Presjek jezika L i N — $L \cap N$
3. Razlika jezika L i N — $L - N$
4. Nadovezivanje jezika L i N — LN
5. Kartezijev produkt — $L \times N$
6. Partitivni skup jezika — 2^L
7. Kleeneov operator * — L^*
8. Kleeneov operator + — L^+
9. Komplement jezika — L^C

Skupovi mogu biti:
- prebrojivo beskonačni (postoji bijekcija na skup prirodnih brojeva)
- neprebrojivo beskonačni (ne postoji bijekcija na skup prirodnih brojeva)

Kardinalni broj: broj elemenata skupa.

Graf: G = (V, E) čini konačni skup čvorova V i skup parova čvorova E. Parovi čvorova su grane grafa.
Usmjereni graf: Grane usmjerenog grafa su uređeni parovi čvorova koje nazivamo usmjerenim granama.
Stablo: Usmjereni graf sljedećih svojstava:

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

2. DETERMINISTIČKI KONAČNI AUTOMAT (DKA)

2.1. Regularni jezik i konačni automat

- regularni jezik i konačni automati
- definicija deterministčkog konačnog automata

Jezik je **regularan** ako i samo ako postoji **konačni automat** koji ga prihvaća. Time je definirana **istovjetnost** konačnih automata i regularnih jezika.
Konačni automati:

- deterministički konačni automat (**DKA**, DFA)
- nedeterministički konačni automat (**NKA**, NFA)
- NKA s ε prijelazima (**ε-NKA**, ε-NFA)

Definicija DKA:

dka = (Q, ∑, δ, q_0, F)

Q - konačan skup stanja
∑ - konačan skup ulaznih znakova
δ - funkcija prijelaza δ: Q x ∑ → Q
$q_0 \in Q$ - početno stanje
$F \in Q$ - skup prihvatljivih stanja

DKA se prikazuje **dijagramom stanja** ili **tablicom prijelaza.**

2.1. Deterministički konačni automat i niz

- proširena fja prijelaza DKA
- prihvaćanje ulaznog niza DKA

Fja prijelaza**:** jednoznačno određuje prijelaz u iduće stanje što se zapiše na sljedeći način:
NovoStanje = δ (StaroStanje, UlazniZnak).
Za bilo koji par starog stanja i ulaznog znaka jednoznačno je određeno u koje stanje prelazi DKA.

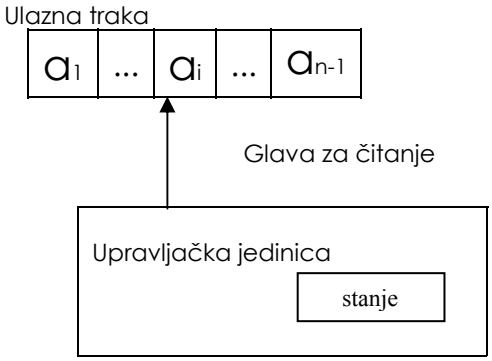
Definira se **fja**: $\hat{\delta} : Q \times \Sigma^* \to Q$, koja definira stanje automata nakon čitanja ulaznog niza.
Σ* – skup svih mogućih nizova ulaznih znakova, uključujući i prazni niz;

DKA prihvaća niz ako je: $\delta(q_0, x) = p, \quad p \in F$
DKA prihvaća skup L(dka) $\subseteq \Sigma^*$ ako je: $L(dka) = \{x | \delta(q_0, x) \in F\}$
Za nizove koji nisu u skupu L(DKA) kaže se da ih DKA ne prihvaća.

2.3. Model DKA i programsko ostvarenje

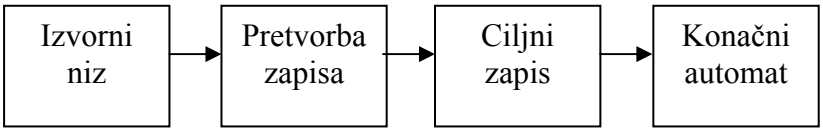
- model DKA
- programsko ostvarenje DKA
- pretvorba ulaznog niza
- načini ostvarenja fja prijelaza

Model DKA:



- Traka je konačna
- Glava samo čita i ne može pisati

Programsko ostvarenje DKA:
Konačni automat je **matematički model** koji je moguće ostvariti **programskim jezikom**. Za potrebe učinkovitog programskog ostvarenja razmatraju se načini zapisa ulaznih znakova, stanja i fje prijelaza.



Načini zapisa stanja:
- **izravan:** zapisom u varijablu
- **posredan**: zapis dijelom programa

Načini zapisa fje prijelaza:
- **vektorski:**
  - za svako stanje jedan vektor
  - za svaki znak jedan element vektora
  - kod izravnog zapisa stanja element vektora je stanje
  - kod neizravnog zapisa stanja element vektora je adresa
- **listom:**
  - postiže se učinkovito korištenje memorije
  - unosi se lista parova znak,stanje
  - troši se više vremena

2.4. Definicije ekvivalentnosti

- ekvivalentnost stanja i automata
- svojstva ekvivalentnosti
- strategija redukcije broja stanja
- uvjeti istovjetnosti stanja

Istovjetnost stanja (ekvivalentnost): dva stanja su istovjetna ako prihvaćaju ili odbijaju isti skup nizova.
Ekvivalentnost automata M i N: M i N su istovjetni ako su istovjetna njihova početna stanja.
Uvjeti istovjetnosti: npr. stanja p i q
- uvjet **podudarnosti:** p i q moraju biti prihvatljiva ili neprihvatljiva
- uvjet **napredovanja:** vrijedi: $\delta(p,a)= \delta(q,a)$

Tranzitivnost istovjetnosti: iz p=q i q=r slijedi p=r

Strategija redukcije broja stanja:
- grupu istovjetnih stanja zamijeniti jednim
- prvo istovjetna stanja označiti istim imenom
- sve prijelaze označiti tim imenom
- u skupu Q ostaviti samo jedno stanje
- po potrebi funkciju prijelaza

Za određivanje broja stanja koristimo neki od 3 algoritma:
- primitivne tablice
- Huffman-Meally.jev algoritam
- tablica implikanata (Paul-Ungerov algoritam)

3. MINIMIZACIJA KONAČNOG AUTOMATA

3.1. Algoritam primitivne tablice

- koraci algoritma
- primjer

Određivanje istovjetnosti stanja (npr. p i q) **Prvim algoritmom (primitivnim tablicama):**
- primjenjuje uvjete istovjetnosti iterativno
- neučinkovit, zahtjeva ispitivanje svih parova stanja

Koraci algoritma:
1. provjeri uvjet podudarnosti
2. za svaki znak formiraj zasebni stupac
3. provjeri uvjet napredovanja i za svaki novi par različitih stanja stvori novi redak
4. provjeri uvjet podudarnosti za svaki novi par, ako uvjet nije zadovoljen prvi par nije istovjetan
5. ako nema novog para, prvi par je istovjetan

Primjer:

	c	d	⊥
po	po	p3	0
p1	p2	p5	0
p2	p2	p7	0
p3	p6	p7	0
p4	p1	p6	1
p5	p6	p5	0
p6	p6	p3	1
p7	p6	p3	0

	c	d
po, p1	po, p2	p3, p5
po, p2	po, p2	p3, p7
p3, p5	p6	p5, p7
p3, p7	p6	p3, p7
p5, p7	p6	p3, p5

p0=p1=p2
p3=p5=p7

	c	d	⊥
po	po	p3	0
p3	p6	p3	0
p4	p1	p6	1
p6	p6	p3	1

Stanje p4 je nedostupno.

3.2. Algoritam Huffman-Mealy

- koraci algoritma
- primjer

Određivanje istovjetnosti stanja (npr. p i q) **Drugim algoritmom (Huffman-Mealy):**
- dijeli skup stanja Q na podskupove korištenjem uvjeta podudarnosti

Koraci algoritma:

1. podijeli Q na dva podskupa na osnovu uvjeta podudarnosti (pripadnost skupu F)
2. provjeri zatvorenost podskupova: podijeli podskup tako da su u novom podskupu p i q:
   $\delta(p,a) \in Gi \ \wedge \ \delta(q,a) \in Gi$
3. ako nema novih podskupova, stanja u podskupovima su istovjetna

Primjer:

	c	d	⊥
po	po	p3	0
p1	p2	p5	0
p2	p2	p7	0
p3	p6	p7	0
p4	p1	p6	1
p5	p6	p5	0
p6	p6	p3	1
p7	p6	p3	0

po, p1, p2, p3, p5, p7 ; p4, p6
po, p1, p2 ; p3, p5, p7 ; p4 ; p6

po = p1 = p2
p3 = p5 = p7

ISTI AUTOMAT!

3.3. Algoritam tablice implikanata

- koraci algoritma
- primjer

Određivanje istovjetnosti stanja (npr. p i q) **Trećim algoritmom (tablica implikanata):**
Traži neistovjetna stanja **koracima algoritma**:

1. označi sve neistovjetne parove stanja na osnovu uvjeta podudarnosti (pripadnost skupu F)
2. za sve neoznačene parove *p, q* za sve ulazne simbole:
   ako je $\delta(p,a)$, $\delta(q,a)$ označen označi *p, q*
   označi rekurzivno sve parove u listi *p,q* i dalje
   inače za sve znakove *a*
   ako je $(\delta(p,a) != \delta(q,a))$
   dodaj p, q u listu $\delta(p,a)$, $\delta(q,a)$ (implikanti)

Primjer:

	c	**d**	⊥
p o	p o	p 3	0
p 1	p 2	p 5	0
p 2	p 2	p 7	0
p 3	p 6	p 7	0
p 4	p 1	p 6	1
p 5	p 6	p 5	0
p 6	p 6	p 3	1
p 7	p 6	p 3	0

p1							
p2							
p3	x	x	x				
p4	**x**	**x**	**x**	**x**			
p5	x	x	x		**x**		
p6	**x**	**x**	**x**	**x**	x	**x**	
p7	x	x	x		**x**		**x**
	p0	**p1**	**p2**	**p3**	**p4**	**p5**	**p6**

p0=p1=p2
p3=p5=p7

3.4. Nedohvatljiva stanja

- definicija nedohvatljivih stanja
- algoritam eliminacije
- algoritam postizanja DKA s minimalnim brojem stanja

Stanje p je **nedohvatljivo** ako ne postoji niz w: $\delta(q_0,w) = p$.
Odbacivanjem nedohvatljivih stanja dobije se istovjetni DKA s manjim brojem stanja.
Koraci:

1. u listu dohvatljivih stanja DS upiši q_0
2. proširi DS sa skupom stanja $\{p \mid p=\delta(q_0,a)$, za sve $a\Sigma\in\}$
3. za sva stanja $q_i \in$ DS proširi DS sa skupom stanja
   $\{p \mid p=\delta(q_i,a) \wedge p\notin DS$, za sve $a\ \Sigma\in\}$

DKA s minimalnim brojem stanja:

- Odbacivanjem istovjetnih i nedostupnih stanja dobije se istovjetni **DKA s minimalnim brojem stanja.**
- Ne postoji drugi DKA koji prihvaća isti jezik, a koji ima manje stanja
- Optimalno je najprije odbaciti nedostupna stanja pa onda tražiti i odbaciti istovjetna stanja.

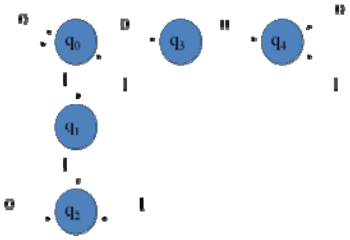
4. NEDETERMINISTIČKI KONAČNI AUTOMAT (NKA)

4.1. Pristup i rad NKA

- opis NKA
- primjer i rad NKA
- formalna definicija NKA

Opis NKA (NFA): Za razliku od funkcije prijelaza DKA koja za jedno stanje i jedan znak određuje prijelaz u jedinstveno stanje, nova funkcija prijelaza određuje prijelaz u skup stanja.

Primjer NKA: prihvaća nizove koji imaju barem dvije uzastopne nule ili jedinice



Rad NKA opisuje se na sljedeći način:
- svaki put kada postoji mogućnost prijelaza u više različitih stanja, stvori se toliki broj DKA koji paralelno obrađuju ulazni niz.
- niz se prihvaća ako staza makar jednog DKA završi prihvatljivim stanjem.

Formalna definicija:
NKA čini:

$$\mathbf{nka = (Q, \Sigma, \delta, q_0, F)}$$

Q - konačan skup stanja
Σ - konačan skup ulaznih znakova
δ - funkcija prijelaza δ: $Q \rightarrow \Sigma \times 2^Q$
$q_0 \in Q$ - početno stanje
$F \in Q$ - skup prihvatljivih stanja

4.2. Definicija NKA

- formalna definicija NKA
- proširena fija prijelaza NKA
- prihvaćanje ulaznog niza
- fija prijelaza NKA nad skupom stanja

NKA čini:

$$\mathbf{nka = (Q, \Sigma, \delta, q_0, F)}$$

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

Definira se **fja**: $\hat{\delta} : Q \times \Sigma^* \to 2^Q$, koja definira stanje automata nakon čitanja ulaznog niza.
Vrijedi:

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

(2) $\hat{\delta}(q, wa) = P = \left\{ p \middle| r \in \hat{\delta}(q, w) \Rightarrow p \in \delta(r, a) \right\}$

Vrijedi:
- iz (1) da automat mijenja stanje ako se desi ulazni znak
- uvrštavanjem u (2) $w=\varepsilon$ dobije se: $\hat{\delta}(q, \varepsilon a) = \hat{\delta}(q, a) = P = \left\{ p \middle| p \in \delta(q, a) \right\} = \delta(q, a)$

Slijedi da su obje fje iste, tj. opisuju iste prijelaze.

Nadalje proširimo funkciju: $\delta : 2^Q \times \Sigma^* \to 2^Q$
- koja definira stanje automata nakon čitanja ulaznog niza polazeći iz nekog od podskupa stanja $P \in 2^Q$
- vrijedi: $\delta(P, w) = \bigcup_{q \in P} \delta(q, w); \; P \subseteq Q$

NKA prihvaća niz ako je u P makar jedno stanje iz F: $\delta(q_0, x) = P; \quad P \cap F \neq \varnothing$
NKA prihvaća skup L(nka) $\subseteq \Sigma^*$: $L(nka) = \left\{ x \middle| \delta(q_0, x) \cap F \neq \varnothing \right\}$
Za nizove koji nisu u skupu L(nka) kaže se da ih NKA ne prihvaća

4.3. Izgradnja DKA iz zadanog NKA

- definicija algoritma
- problem nedostupnih stanja
- primjer izgradnje

Izgradnja DKA iz zadanog NKA:
- Za bilo koji NKA M={Q,Σ,δ,q0,F} moguće je izgraditi istovjetni DKA M'={Q',Σ',δ',q0',F'}
- NKA i DKA su istovjetni ako prihvaćaju isti jezik L(M)=L(M')
- ako je Q={q_0,...,q_i} izgradimo Q'=2^Q, [p_0,...,p_j] ∈ Q', p_k ∈ Q;
  pa vrijedi: Q'={[∅], [q_0],..., [q_i], [q_0, q_1],..., [q_0, q_j],..., [q_0,...,q_j] }
- F'=skup svih [p_0,...,p_j] gdje je barem jedan p_k ∈ F
- početno stanje je q_0'=[q_0]
- funkcija prijelaza DKA jest: δ'([p_0,...,p_l],a) = [r_0,...,r_j] ako i samo ako je: δ({p_0,...,p_l},a) = {r_0,...,r_j}
- **mnoga stanja dobivenog DKA su nedostupna**, pa ih eliminiramo
- DKA minimiziramo radi učinkovite programske realizacije

Primjer izgradnje:
Ako je NKA *M*={ q_0,q_1},{0,1}, *δ*, q_0,{q_1}). Fja prijelaza NKA *M* prikazana je tablicom:

	0	1	⊥
qo	{qo,q1 }	{q1}	0
q1	{ }	{qo,q1}	1

DKA $M'=(Q', \Sigma', \delta', q_0', F')$ gradimo na sljedeći način:

1) $Q'=\{[\varnothing], [q_0], [q_1], [q_0, q_1]\}$,
2) $F'=\{[q_1], [q_0, q_1]\}$,
3) $q_0'=[q_0]$.
4) Fija prijelaza δ' *je:*

$\delta'([q_0], 0)=[q_0, q_1]$,
$\delta'([q_0], 1)=[q_1]$,
$\delta'([q_1], 0)=[\varnothing]$,
$\delta'([q_1], 1)=[q_0, q_1]$,
$\delta'([q_0, q_1], 0)=[q_0, q_1]$,
$\delta'([q_0, q_1], 1)=[q_0, q_1]$,
$\delta'([\varnothing], 0)=[\varnothing]$,
$\delta'([\varnothing], 1)=[\varnothing]$.

Izgrađeni DKA $M'=(Q', \{0,1\}, \delta', [q_0], \{[q_1], [q_0, q_1]\})$:

	0	1	⊥
qo	[qo,q1]	[q1]	0
q1	[∅]	[qo,q1]	1
[qo,q1]	[qo,q1]	[qo,q1]	1
[∅]	[∅]	[∅]	0

4.4. Istovjetnost NKA i DKA

- dokaz istovjetnosti

Istovjetnost NKA i DKA:

Neka je DKA M' izgrađen na temelju zadanog NKA M. Želi se dokazati da automati M i M' prihvaćaju isti jezik L(M) = L(M').

Dokazuje se **indukcijom** da za bilo koji niz $w\in\Sigma^*$ vrijedi:

$\delta'([q0], w)=[r0,..., rj]$ ako i samo ako je $\delta(q0, w)=\{r0,..., rj\}$.

Indukcija se zasniva na duljini niza x:

- najprije dokažemo za $|w|=0$, tj. $w=\varepsilon$:
- pretpostavimo da vrijedi za $x\in\Sigma^*$, a onda dokažemo za xa, $a\in\Sigma$

Na temelju pretpostavke vrijedi:

- (1) $\delta'([q0], x)=[p0,..., pj]$ ako i samo ako je $\delta(q0, x)=\{p0,..., pj\}$
- $\delta'([p0,...,pl],a) = [r0,...,rj]$ ako i samo ako je $\delta(\{p0,...,pl\},a) = \{r0,...,rj\}$
- (2) $\delta'([p0,..., pj], a)=[r0,..., rj]$ ako i samo ako je $\delta(\{p0,..., pj\}, a)=\{r0,..., rj\}$
- (3) $\delta'([q_0], xa)=[r_0,..., r_j]$ ako i samo ako je $\delta(q_0, xa)=\{r_0,..., r_j\}$
- obzirom da je $\delta'([q_0], w) \in F'$ ako i samo ako $\delta(q_0, w)$ sadrži makar jedno stanje iz F
- slijedi da NKA M i DKA M' prihvaćaju isti jezik odnosno L(M)=L(M')

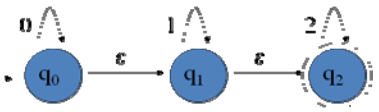
5. EPSILON-NEDETERMINISTIČKI KONAČNI AUTOMAT (ε-NKA)

5.1. Pristup i rad ε-NKA

- opis ε-NKA
- primjer i rad ε-NKA
- formalna definicija ε-NKA

Opis ε-NKA: (nedeterministički konačni automat s ε-prijelazima)
- NKA zasnovan na proširenoj fiji prijelaza
- omogućuje konačnom automatu da promijeni stanje, a da ne pročita niti jedan ulazni znak.

Primjer:
- prihvaća nizove koji:
  - Započinju proizvoljnim brojem nula
  - Nastavljaju proizvoljnim brojem jedinica
  - Završavaju proizvoljnim brojem dvojki



Rad ε-NKA: ε-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 slijed prijelaza uključuju se i ε-prijelazi.

Formalna definicija:

ε-nka=(Q, Σ, δ, q_0, F)

Q - konačan skup stanja
∑ - konačan skup ulaznih znakova
δ - funkcija prijelaza:

$\delta: Q\times(\Sigma\cup\{\varepsilon\})\rightarrow 2^Q$
$q_0 \in Q$ - početno stanje
$F \in Q$ - skup prihvatljivih stanja

5.2. Definicija NKA

- formalna definicija ε-NKA
- definicija ε-okruženja
- definicija proširene fije prijelaza

Formalna definicija:

ε-nka=(Q, Σ, δ, q_0, F)

Q - konačan skup stanja
∑ - konačan skup ulaznih znakova
δ - funkcija prijelaza:

$\delta: Q\times(\Sigma\cup\{\varepsilon\})\rightarrow 2^Q$
$q_0 \in Q$ - početno stanje
$F \in Q$ - skup prihvatljivih stanja

ε-OKRUŽENJE(q): stanju $q \in Q$ dodjeljujemo skup $R \subseteq Q$ tako da R sadrži sva ona stanja u koja q prelazi isključivo ε-prijelazom:
ε-OKRUŽENJE(q) = {p | p jest q ili ε-NKA prelazi iz q u p isključivo ε-prijelazom}
Fja $\hat{\delta}(q,w)$ jest **proširenje fje prijelaza** na argumente: $\hat{\delta}: Q\times\Sigma^* \rightarrow 2^Q$.

Fja $\hat{\delta}$ definira se pomoću fije *ε-OKRUŽENJE* (uključujemo i ε-prijelaze):

$$\hat{\delta}(q,\varepsilon)=\varepsilon-\text{OKRUŽENJE}\{q\}$$

$$\hat{\delta}(q,wa)=\varepsilon-\text{OKRUŽENJE}\{P\};\qquad P=\left\{p\middle|r\in\hat{\delta}(q,w)\Rightarrow p\in\delta(r,a)\right\}$$

Za skupove stanja vrijedi:

$$\delta(R,a)=\bigcup_{q\in R}\delta(q,a)$$

$$\hat{\delta}(R,w)=\bigcup_{q\in R}\hat{\delta}(q,w)$$

5.3. Izgradnja NKA iz zadanog ε-NKA

- definicija algoritma izgradnje
- primjer izgradnje

Izgradnja NKA iz zadanog ε-NKA:

- Za bilo koji ε-NKA M={Q,Σ,δ,q_0,F}moguće je izgraditi istovjetni NKA M'={Q',Σ',δ',q_0',F'}
- ε- NKA i NKA su istovjetni ako prihvaćaju isti jezik L(M)=L(M')
- izgradimo Q'= Q ={q_0,...,q_i}
- početno stanje je q_0'=q_0
- F'=F∪{q_0} ako ε-OKRUŽENJE(q_0) sadrži barem jedno stanje p_k∈ F, inače F'=F
- $\delta'(q,a)=\hat{\delta}(q,a)$

Primjer:

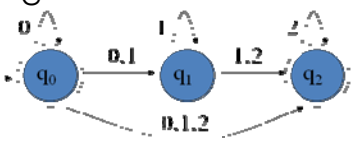
- izgradimo Q' = Q = {q_0, q_1, q_2}
- početno stanje je q_0'=q_0
- F' = F∪{q_0} = {q_2}∪{q_0} = {q_0, q_2} jer je:
  ε-OKRUŽENJE(q_0)∩F = {q_0, q_1, q_2}∩{q_2} = {q_2}
- funkcija δ':
  $\delta'(q_0,0)=\{q_0,q_1,q_2\}$ jer je $\hat{\delta}(q_0,0)=\varepsilon-\text{OKRUŽENJE}\left(\delta\left(\hat{\delta}(q_0,\varepsilon),0\right)\right)=\{q_0,q_1,q_2\}$

ITD.

Dobije se NKA:

	0	1	2	⊥
qo	{ qo, q1, q2},	{q1, q2}	{q2}	1
q1	∅	{q1, q2},	{q2}	0
q2	∅	∅	{q2}	1

Dijagram:



5.4. Istovjetnost ε-NKA i NKA

- dokaz istovjetnosti

Dokazuje se indukcijom s obzirom na duljinu niza x. Dokaz se izvodi u dva koraka.
U prvom koraku dokazuje se da vrijedi: $\delta'(q_0,x)=\hat{\delta}(q_0,x)$

U drugom koraku dokazuje se da δ'(q_o, x) sadrži stanje iz F' ako i samo ako $\hat{\delta}$ (q_o, x) sadrži stanje iz skupa F.
(detaljniji izvod u knjizi, str. 38)

6. KONAČNI AUTOMATI SA IZLAZOM

6.1. Moore automat

- ideja izlaza
- formalna definicija Moore automata
- primjer

DKA (DFA) s izlazom:

- **izlaz** je ograničen funkcijom (0,1) koja označava da li se niz prihvaća ili odbacuje
- **funkcija izlaza proširuje se na dva načina:**
  - **Moore = izlaz je funkcija stanja**
  - Mealy = izlaz je funkcija stanja i ulaza
- za Mealy je moguće izgraditi istovjetni Moore i obrnuto
- Mealy i Moore automati su istovjetni ako za bilo koji ulazni niz daju jednake izlazne nizove

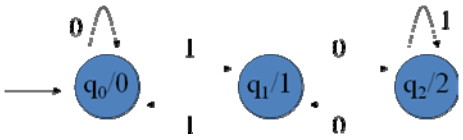
Formalna definicija:

$$MoDka = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

– Q je skup stanja, $q_0 \in Q$ početno stanje
– Σ je konačan skup ulaznih znakova,
– Δ je konačan skup izlaznih znakova,
– δ je funkcija prijelaza $Q \times \Sigma \rightarrow Q$,
– λ je funkcija izlaza $Q \rightarrow \Delta$

DKA je poseban slučaj Moore automata sa $\Delta=\{0,1\}$

Primjer: Za niz koji predstavlja binarni broj potrebno je izgraditi Moore automat koji za pročitani prefiks niza daje ostatak dijeljenja broja s 3.



6.2. Mealy automat

- formalna definicija Mealy automata
- primjer

DKA (DFA) s izlazom:

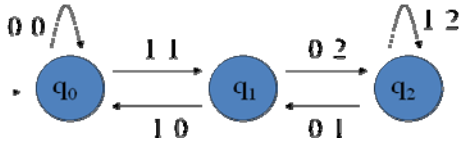
- **izlaz** je ograničen funkcijom (0,1) koja označava da li se niz prihvaća ili odbacuje
- **funkcija izlaza proširuje se na dva načina:**
  - Moore = izlaz je funkcija stanja
  - **Mealy = izlaz je funkcija stanja i ulaza**
- za Mealy je moguće izgraditi istovjetni Moore i obrnuto
- Mealy i Moore automati su istovjetni ako za bilo koji ulazni niz daju jednake izlazne nizove

Formalna definicija:

$MeDka = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

– Q je skup stanja, $q_0 \in Q$ početno stanje
– Σ je konačan skup ulaznih znakova,
– Δ je konačan skup izlaznih znakova,
– δ je funkcija prijelaza $Q \times \Sigma \to Q$,
– λ je funkcija izlaza $Q \times \Sigma \to \Delta$

Primjer: Za niz koji predstavlja binarni broj potrebno je izgraditi Mealy automat koji za pročitani prefiks niza daje ostatak dijeljenja broja s 3.



6.3. Konstrukcija Mealy iz zadanog Moore DKA

- definicija istovjetnosti
- izgradnja funkcije izlaza
- primjer

Konstrukcija Mealy iz zadanog Moore automata

– za prazni niz ε Moore M daje, a Mealy M' ne daje izlaz
– istovjetnost se definira: $bT_{M'}(w) = T_M(w); \qquad b = \lambda(q_0)$
– za niz nakon početnog simbola b Moore automata
– za Mealy automat gradi se izmijenjena funkcija izlaza:

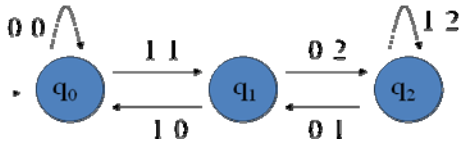
$$\lambda'(q,a) = \lambda(\delta(q,a)); \quad \forall q \in Q, a \in \Sigma$$

Primjer: Konstrukcija Mealy iz zadanog Moore automata

- Za niz koji predstavlja binarni broj potrebno je izgraditi Moore automat koji za pročitani prefiks niza daje ostatak dijeljenja broja s 3:

$$\lambda'(q_0,0) = 0; \quad \lambda'(q_0,0) = \lambda(\delta(q_0,0)) = \lambda(q_0) = 0$$
$$\lambda'(q_0,1) = 1; \quad \lambda'(q_0,1) = \lambda(\delta(q_0,1)) = \lambda(q_1) = 1$$
$$\lambda'(q_1,0) = 2; \quad \lambda'(q_1,0) = \lambda(\delta(q_1,0)) = \lambda(q_2) = 2$$
$$\lambda'(q_1,1) = 0; \quad \lambda'(q_1,1) = \lambda(\delta(q_1,1)) = \lambda(q_0) = 0$$
$$\lambda'(q_2,0) = 1; \quad \lambda'(q_2,0) = \lambda(\delta(q_2,0)) = \lambda(q_1) = 1$$
$$\lambda'(q_2,1) = 2; \quad \lambda'(q_2,1) = \lambda(\delta(q_2,1)) = \lambda(q_2) = 2$$

- nacrtamo graf:



6.4. Konstrukcija Moore iz zadanog Mealy DKA

- definicija istovjetnosti
- izgradnja Moore automata
- primjer

Dva su automata istovjetna ako za bilo koji ulazni niz daju jednake izlazne nizove.
Istovjetnost Mealyevog automata M' i Mooreovog automata M definira na sljedeći način:

$$b\,T_{M'}(w) = T_M(w); \qquad b = \lambda(q_0)$$

za niz nakon početnog simbola b Moore automata.

Konstrukcija Moore iz zadanog Mealy:

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

Primjer:
-skup stanja Q' je $Q' = \{[q_0,0],[q_0,1],[q_0,2],[q_1,0],[q_1,1],[q_1,2],[q_2,0],[q_2,1],[q_2,2]\}$
-početno stanje q_0' je $[q_0,0]$ i funkcija prijelaza je:

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

-odbacimo nedostupna stanja i dobijemo polazni Moore:

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

-funkcija izlaza je:

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

7. REGULARNI JEZICI I IZRAZI

7.1. Definicija regularnih izraza

- regularni jezik i izraz
- algoritam sinteze DKA

Regularni jezik je **regularan** ako postoji konačni automat koji ga prihvaća.
Regularni izraz je izraz kojim definiramo regularni jezik.
Algoritam sinteze DKA:

1. regularni jezik *K*
2. opis jezika *K* regularnim izrazima *r*: *L(r)=K*
3. izgraditi ε-NKA *M* za koji vrijedi: *L(M)=L(r)*
4. izgraditi NKA *M* ' za koji vrijedi: *L(M ')=L(M)*
5. izgraditi DKA *M* '' za koji vrijedi: *L(M '')=L(M ')*
6. izgraditi minimalni DKA *M* ''' za koji vrijedi:
7. *L(M ''')=L(M '')=L(M ')=L(M)=L(r)=K*

7.2. Rekurzivna pravila za RI

- navesti rekurzivna pravila
- pravila asocijativnosti i prednosti
- istovjetnost i svojstva RI

Rekurzivna pravila:

1. ∅ jest RI i označava jezik L(∅)={}
2. ε jest RI i označava jezik L(ε)= {ε}
3. ∀a∈Σ, a jest RI i označava jezik L(a)={a}
   (ista iznaka "a" znači znak, niz i RI)
4. ako su r i s RI koji označavaju L(r) i L(s) vrijedi:
   - (r)∨(s) jest RI i označava jezik L((r)∨(s))= L(r)∪L(s)
     (operator ∨ nekad se označava sa + ili |)
   - (r)(s) jest RI i označava jezik L((r) (s))= L(r)L(s)
   - (r)* jest RI i označava jezik L((r)*)= (L(r))*

Pravila asocijativnosti i prednosti:

1. unarni operator ***** jest lijevo asocijativan i najveće je prednosti
2. binarni operator nadovezivanja je lijevo asocijativan i veće je prednosti od operat. ∨
3. binarni operator ∨ jest lijevo asocijativan i najmanje je prednosti

Istovjetnost RI: Dva RI *r* i *s* su istovjetna ako definiraju iste jezike: L(r)=L(s) [piše se *r*=*s*].
Svojstva RI:

1. r ∨ s = s ∨ r — ∨ jest komutativno
2. r ∨ (s ∨ t) = (r ∨ s) ∨ t — ∨ jest asocijativno
3. (rs)t = r(st) — nadovezivanje jest asocijativno
4. r(s ∨ t) = rs ∨ rt
   (s ∨ t)r = sr ∨ tr — distributivnost nadovezivanja nad ∨
5. εr = rε =r — ε jest neutralni element nadovezivanja

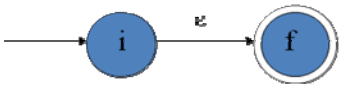
6. $r^* = (r \vee \varepsilon)^*$ relacija između $\vee$ i *
7. $r^{**} = r^*$ idempotentnost

7.3. Konstrukcija e-NKA iz RI

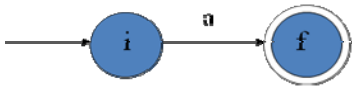
– pravila elementarnih automata

p1. za RI ∅ koji označava jezik L(∅)={} izgradimo: ε-NKA M=({i,f}, Σ, {}, i, {f})

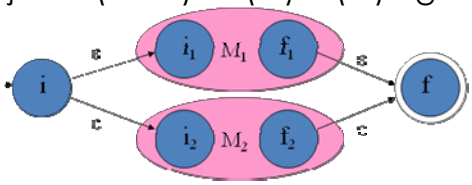
p2. za RI ε koji označava jezik L(ε)={ε} izgradimo: ε-NKA M=({i,f}, Σ, {δ(i,ε)=f}, i, {f})



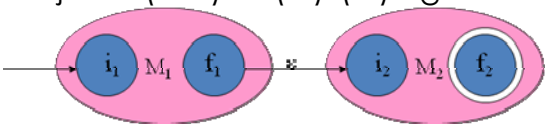
p3. za RI a koji označava jezik L(a)={a} izgradimo: ε-NKA M=({i,f}, Σ, {δ(i,a)=f}, i, {f})



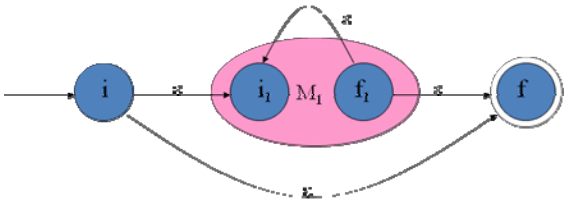
p4. za RI **r1∨r2** koji označava jezik L(r1∨r2) = L(r1)∪L(r2) izgradimo ε-NKA M



p5. za RI **r1r2** koji označava jezik L(r1r2) = L(r1)L(r2) izgradimo ε-NKA M



p6. za RI **r1*** koji označava jezik L(r1*) = L(r1)* izgradimo ε-NKA M



p7. za RI (r) koji označava jezik L((r)) = L(r) izgradimo ε-NKA M za RI (r) tako da uzmemo

ε-NKA M1 za RI r

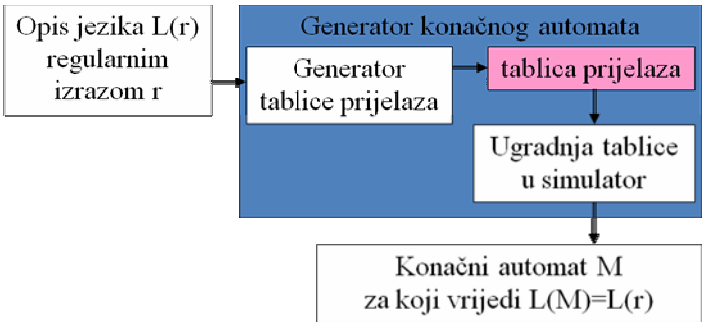
7.4. Generator konačnog automata

- generator i simulator
- struktura ostvarenja

Generator KA (konačnog automata): ostvaruje dio ili cjelokupnu pretvorbu RI u DKA

Simulator:
-ovisno o željenom konačnom automatu izgradi se program simulator
-simulator radi na upisanoj tablici prijelaza (izravni način zapisa stanja, samo mijenjamo tablice)
-simulator čita znakove ulaza i računa prijelaz u stanje na osnovu tablice prijelaza

Struktura ostvarenja:
Generator KA gradi tablicu prijelaza na temelju RI. Tablica prijelaza se ugradi u program simulator.

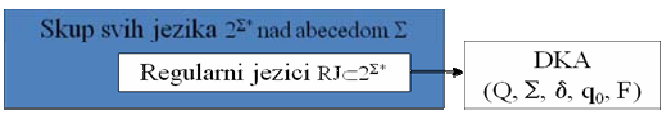


8. SVOJSTVA REGULARNIH JEZIKA

8.1. Klase jezika

- skup svih jezika
- položaj regularnih jezika
- zatvorenost klase jezika
- unija, nadovezivanje, Kleene

Skup svih jezika:



- 2^{Σ^*} označava skup svih jezika nad abecedom Σ, a svaki jezik $L \subseteq \Sigma^*$ je član tog skupa $L \in 2^{\Sigma^*}$.
Regularni jezici su u skupu $RJ \subset 2^{\Sigma^*}$.
Zatvorenost klase jezika:

- definira se obzirom na operacije nad jezicima
- klasa je zatvorena ako primjenom operacije dobijemo jezik u istoj klasi

RJ su zatvoreni obzirom na **uniju, nadovezivanje i Kleeneov operator** što slijedi iz definicije regularnih izraza.

8.2. Zatvorenost regularnih jezika

- zatvorenost obzirom na komplement
- zatvorenost obzirom na presjek

Zatvorenost obzirom na komplement:

1. neka DKA $M = (Q, \Sigma, \delta, q0, F)$ prihvaća $L(M) \in RJ$
2. za komplement jezika $L(M)^c$ izgradimo: DKA $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$,koji prihvaća jezik:

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

Zatvorenost obzirom na presjek:

- koristimo zatvorenost unije i komplementa, te DeMorganovo pravilo: $L \cap N = ((L \cap N)^c)^c = (L^c \cup N^c)^c$

8.3. Regularne definicije

- zatvorenost obzirom na supstituciju
- regularne definicije

Zatvorenost obzirom na supstituciju: Regularni jezici zatvoreni su obzirom na **supstituciju.** Svojstvo supstitucije (zamjene) omogućava pojednostavljeno zapisivanje **regularnih definicija**

1. neka je $R \subseteq \Sigma^*$; $R \in RJ$
2. pridružimo znaku $a \in \Sigma$ RJ $R_a \subseteq \Delta^*$ tako da niz $a_1a_2...a_n$ zamijenimo nizom $w_1w_2...w_n$
3. dobiveni jezik f(R) je regularan
4. dovoljno je R i R_a opisati regularnim izrazima

Regularne definicije:

- imenima dodjeljujemo RI, odnosno zamjenjujemo s RI
- oblik regularnih definicija je:

$$d_1 \rightarrow r_1$$
$$d_2 \rightarrow r_2$$
$$.....$$
$$d_n \rightarrow r_n$$

9. SVOJSTVO NAPUHAVANJA

9.1. Definicija svojstva napuhavanja

- problem dugačkih nizova
- ponavljanje stanja
- prihvaćanje dugačkog niza

Svojstvo napuhavanja: pogodno je za dokazivanje neregularnosti nekih jezika, kao i za dokazivanje ispravnosti raznih algoritama kojima se utvrđuje nepraznost regularnog jezika, beskonačnost regularnog jezika, itd...

Ponavljanje stanja:
Ako DKA koji prima ulazni niz koji je duži od broja stanja automata, barem će se jedno stanje u nizu ponavljati.

Prihvaćanje dugačkog niza:
- bilo koji dovoljno dugački niz $z \in L(M)$ može se rastaviti na podnizove: $z=uvw$
- podniz v moguće je proizvoljan broj puta ponoviti, jer je $uv^iw \in L(M)$, a M odgovarajući DKA
- ako RJ sadrži dovoljno dugačak niz $z=uvw$, onda taj jezik sadrži beskonačni skup nizova uv^iw

9.2. Dokaz neregularnosti

- pristup dokazu neregularnosti
- primjer
- nepraznost i beskonačnost regularnog jezika

Pristup dokazu neregularnosti:
- ako je L regularan, postoji **n** takav da je moguće
  - bilo koji niz $z \in L$ gdje je $|z|>n$ rastaviti na podnizove $z=uvw$ tako da je:
    $|uv| \leq n$ i $1 \leq |v|$
  - za bilo koji $i \geq 0$ niz $uv^iw \in L$
  - pokazuje se da n nije veći od broja stanja minimalnog DKA koji prihvaća jezik L

Primjer:
- jezik $K = \{0^{\ell^2} \mid \ell \in N; \ell \geq 1\}$ nije regularan
  - pretpostavi se da je L regularan jezik
  - neka **n** odgovara dokazu neregularnosti i neka je $z=0^{n^2}$ niz jezika L: $|z|=n^2$, $|z|>n$
  - prema dokazu neregularnosti niz **z** rastavlja se na podnizove uvw; $1 \leq |v| \leq |uv| \leq n$
  - treba utvrditi da li je niz uv^iw element jezika L za bilo koji **i**
  - ako je **i**=2 onda i $|v| \leq |uv| \leq n$ onda je $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$
  - budući da je $(n^2+n) < (n+1)^2$ vrijedi:
    $n^2 < |uv^2w| < (n+1)^2$ tj. $|uv^2w|$ nije kvadrat cijelog broja
  - bez obzira na n i na podjelu uvw, uv^2w nije član jezika; posljedično L je neregularan.

Algoritmi odlučivanja:

- nepraznost regularnog jezika

- L(M) je neprazan ako DKA M prihvaća niz duljine $|z|<n$; n je broj stanja DKA M

(ako je u skupu dohvatljivih stanja jedno prihvatljivo, L(M) je neprazan)

- beskonačnost regularnog jezika

- L(M) je beskonačan ako DKA M prihvaća niz duljine $n \leq |z| < 2n$; n je broj stanja DKA M
- promatra se graf DKA M i dobije se DKA M' tako da se izuzmu neprihvatljiva stanja za koje ne postoji staza u prihvatljivo

(L(M) je beskonačan ako graf DKA M' ima barem jednu zatvorenu petlju)

10. REGULARNA GRAMATIKA

10.1. Kontekstno neovisna gramatika

- formalna definicija gramatike
- oznake u formalnoj gramatici
- kontekstno neovisna gramatika
- relacija primjene produkcija

Formalna gramatika:
Formalna gramatika je skup pravila kojima nezavršne znakove zamjenjujemo završnim znakovima koristeći produkcije.
Relacijom $\Rightarrow$ označavamo primjenu jednog pravila, dok **relacijom** $\Rightarrow^*$ **ili** $\overset{*}{\Rightarrow}$ označavamo primjenu više pravila.
Gramatiku **formalno definiramo** na svojstvima **kontekstno neovisnih gramatika**
Oznake u formalnoj gramatici:

1. A,B,C,D,E...S su nezavršni znakovi gramatike
2. a,b,c...0, 1, 2..., su završni znakovi gramatike
3. X,Y,Z su završni ili nezavršni znakovi
4. u, v, w, x, y i z označavaju nizove završnih znakova
5. α, β, γ označavaju nizove završnih i nezavršnih znakova

-ima li više produkcija za isti nezavršni znak koristimo |, npr. $A \rightarrow a$ i $A \rightarrow b$ piše se $A \rightarrow a \mid b$.

Kontekstno neovisna gramatika

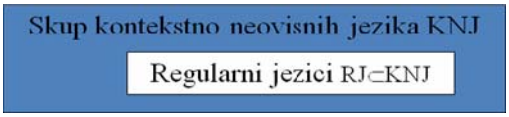
- je uređena četvorka: ***G=(V, T, P, S)***
  - V - konačni skup nezavršnih znakova
  - T - konačni skup završnih znakova $V \cap T = \varnothing$
  - P - konačni skup produkcija oblika $A \rightarrow \alpha$
  - S - početni nezavršni znak
- produkcije gramatike su najčešće u BNF obliku (Backus-Naur Form-sustav oznaka)

10.2. Formalna gramatika i jezici

- generiranje jezika
- kontekstno neovisni jezici
- generativno stablo

Generiranje jezika:
- G=(V, T, P, S) **generira jezik** $L(G) = \{w \mid w \in T^*; S \Rightarrow_G^* w\}$
- gramatike G_1 i G_2 su istovjetne ako generiraju iste jezike tj. ako vrijedi $L(G_1) = L(G_2)$.

Kontekstno neovisne jezike generira kontekstno neovisna **gramatika.**



Generativno stablo:
- za G = (V, T, P, S) stablo je generativno ako
  1. čvorove označimo znakovima $V \cup T \cup \{\varepsilon\}$
  2. korijen stabla označen je početnim nezavršnim znakom S
  3. unutrašnji čvorovi označeni su nezavršnim $A \in V$
  4. za čvor A i djecu $X_1, X_2, \ldots X_n$ vrijedi produkcija iz P $A \rightarrow X_1X_2 \ldots X_n$
  5. znakom ε označava se isključivo list stabla; taj list je jedino dijete svog roditelja
  6. listovi stabla označeni su znakovima skupa $T \cup \{\varepsilon\}$ i čitani slijeva na desno čine generirani niz jezika L(G)

- **$S \Rightarrow_G^*$ w** vrijedi samo ako postoji generativno stablo.

10.3. Regularna gramatika

- definicija
- konstrukcija regularne gramatike iz DKA
- odnos regularne gramatike i DKA

Regularna gramatika:
- regularna gramatika generira regularne jezike
- ujedno je i kontekstno neovisna gramatika
- konstruiranjem gramatike za regularni jezik zadan DKA dokazujemo da je gramatika regularna

Konstrukcija regularne gramatike iz DKA:
- za regularni jezik zadan s DKA $M = (Q, \Sigma, \delta, q_0, F)$ gradi se kontekstno neovisna gramatika
  G = (V, T, P, S) tako da je L(M) = L(G).
- primjenjujemo pravila:
  - **T = Σ**; završni znakovi gramatike su ulazni znakovi automata
  - **V = Q**; nezavršni znakovi su stanja automata
  - **S = q_0**; početno stanje je početni nezavršni znak
  - na temelju prijelaza DKA $\delta(A, a) = B$ gradimo produkciju $A \rightarrow aB$
  - za prihvatljiva stanja $A \in F$ gradimo produkcije $A \rightarrow \varepsilon$

Odnos regularne gramatike i DKA:
Istovjetnost G i DKA: **prihvaća** li DKA isti jezik koji **generira** G, DKA i G su istovjetni: L(DKA) = L(G).

10.4. Sinteza NKA iz regularne gramatike

- istovjetnost regularne gramatike i DKA
- konstrukcija NKA iz jednostavne gramatike
- izvor nederminiranosti

Istovjetnost regularne gramatike i DKA: **prihvaća** li DKA isti jezik koji **generira** G, DKA i G su istovjetni: L(DKA) = L(G).

Konstrukcija NKA za jednostavni G:
- koristimo gramatiku s produkcijama oblika: $A \rightarrow aB$ i $C \rightarrow \varepsilon$
- to su upravo oblici koji nastaju konstrukcijom gramatike na osnovu DKA; pravila su:
  - **Σ = T**; završni znakovi gramatike su ulazni znakovi automata
  - **Q = V**; nezavršni znakovi su stanja automata
  - **q_0 = S**; početno stanje je početni nezavršni znak
  - na temelju produkcije $A \rightarrow aB$ gradi se prijelaz DKA $\delta(A, a) = \delta(A, a) \cup B$ jer su moguće višestruke produkcije iz A,a
  - ako postoji produkcija $A \rightarrow \varepsilon$, stanje A je prihvatljivo; $A \in F$

10.5. Desno linearna gramatika

- definicija desno linearne gramatike
- konstrukcija NKA iz DLG

Desno linearna gramatika je regularna gramatika koja ima najviše jedan nezavršni znak na desnoj strani:
- $A \rightarrow wB$ ili $A \rightarrow w$
- $A,B \in V$; $w \in T^*$

Konstrukcija NKA iz DLG:
Složenu produkciju oblika : $S \rightarrow A$ rastavimo na jednostavne: $S \rightarrow$ desne strane svih produkcija od A. Ako postoje produkcije $A \rightarrow cA$ i $A \rightarrow \varepsilon$ bit će $S \rightarrow cA$, $S \rightarrow \varepsilon$ jer S preko A generira sve međunizove od A.

10.6. Lijevo linearna gramatika

- definicija lijevo linearne gramatike
- konstrukcija e-NKA iz LLG

Lijevo linearna gramatika je regularna gramatika koja ima najviše jedan nezavršni znak na lijevoj strani: $A \rightarrow Bw$ ili $A \rightarrow w$; $A, B \in V$; $w \in T^*$

Konstrukcija e-NKA iz LLG:
Neka je G=(V, T, P, S) LLG.
Konstrukcija ε-NKA M', koji prihvaća jezik L(M')=L(G), se obavlja na način:
1) Iz LLG G se konstruira DLG G'=(V, T, P', S) na način da se skup produkcija P gramatike G preuredi tako da se desne strane produkcija napišu obrnutim

redoslijedom.
Vrijedi $L(G') = L(G)^R$.

2) Na temelju DLG G' se konstruira NKA M koji prihvaća jezik $L(M)= L(G') = L(G)^R$
3) NKA M se preuredi tako da ima samo jedno prihvatljivo stanje.
4) Na temelju NKA M se izgradi ε-NKA M' koji prihvaća jezik $L(M')=L(M)^R=L(G')^R=L(G)$. Početno stanje M' je prihvatljivo stanje od M, a prihvatljivo od M' je početno od M.
5) Funkcije prijelaza NKA M' se grade zamjenom smjera usmjerenih grana u dijagramu stanja.