

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

Sadržaj

21. DIGITALNI AUTOMAT	2
21.1. SUSTAV S UPRAVLJANJEM	2
21.2. SVOJSTVA AUTOMATA 1. DIO	2
21.3. SVOJSTVA AUTOMATA 2. DIO	2
22. APSTRAKTNI MODEL DIGITALNOG AUTOMATA.....	3
22.1. AUTOMAT 1. DIO	3
22.2. AUTOMAT 2. DIO	3
22.3. SINTEZA AUTOMATA	4
23. ZADAVANJE AUTOMATA	5
23.1. PRISTUPI ZADAVANJU AUTOMATA	5
23.2. VRSTE ULAZNE SEKVENCE	5
23.3. POSTUPAK ZADAVANJA KORAK PO KORAK	6
23.4. PRIMJENA POSTUPKA KORAK PO KORAK	6
24. EKVIVALENTNOST AUTOMATA	7
24.1. ODNOSI JEDNAKOSTI MEDU AUTOMATIMA	7
24.2. DEFINICIJA EKVIVALENTNOSTI AUTOMATA	7
24.3. DEFINICIJA EKVIVALENTNOSTI STANJA	7
24.4. NUŽAN I DOVOLJAN UVJET	8
24.5. MINIMIZACIJA PRIMITIVNE TABLICE	8
25. NAPREDNI POSTUPCI MINIMIZACIJE AUTOMATA	8
25.1. HM ALGORITAM	8
25.2. PU ALGORITAM	9
26. STRUKTURNA SINTEZA AUTOMATA.....	10
26.1. MODEL REALIZACIJE AUTOMATA	10
26.2. KODIRANJE AUTOMATA.....	10
26.3. TABLICA AUTOMATA S KODOVIMA	11
26.4. SINTEZA KONKRETNOG AUTOMATA	12
27. AUTOMATI I ALGORITMI.....	12
27.1. PROGRAMABILNI AUTOMAT	12
27.2. ALGORITAM	13
27.3. TURINGOV STROJ.....	13
28. AUTOMATI I JEZICI	14
28.1. ZNAČAJ ANALIZE JEZIKA	14
28.2. KOMPLEKSNOŠĆ ALGORITAMA	14
28.3. IZRAČUNLJIVOST	15
28.4. TAKSONOMIJA AUTOMATA I JEZIKA	16
29. ALGEBRA DOGAĐAJA	17
29.1. ELEMENTARNI I SLOŽENI DOGAĐAJI	17
29.2. OPERATORI ALGEBRA DOGAĐAJA.....	17
30. ZADAVANJE AUTOMATA REGULARNIM IZRAZOM	18
30.1. ZADAVANJE AUTOMATA S POMOCU RI	18
30.2. INDEKSIRANJE RI	19
30.3. DOBIVANJE STRUKTURE AUTOMATA IZ RI	20

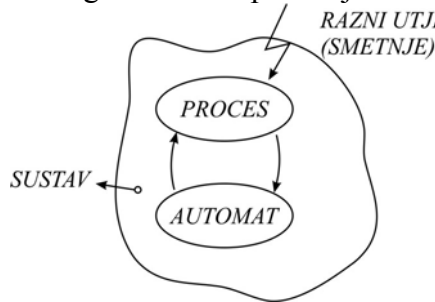
III. DIO - DIGITALNI SUSTAVI I STRUKTURE

21. DIGITALNI AUTOMAT

21.1. Sustav s upravljanjem

- definirati sustav s upravljanjem
- skica strukture sustava s upravljanjem
- funkcija cilja
- uvjeti uspješnosti sustava

Sustav s upravljanjem je sustav koji čine proces i automat. Na proces djeluje okolina remeteći njegov rad, dok automat pokušava održati proces u optimalnim uvjetima rada, kako bi mogao ostvariti postavljenu funkciju cilja.



Uvjeti uspješnosti sustava:

Automat mjeri stanje procesa i razlučuje sva njegova bitna stanja (mjerljivost) i posjeduje ugrađeno znanje o procesu (poznaje svojstva procesa). Na osnovi sadašnjih i prethodnih događaja u procesu, može se odrediti optimalni niz akcija kojima treba dovesti proces u optimalni režim rada. Da bi to bilo moguće, mora raspolagati i s dovoljnim skupom akcija da bi mogao kompenzirati bilo koji predvidljivi utjecaj

okoline. Da bi upravljanje moglo funkcionirati proces mora biti upravljiv.

21.2. Svojstva automata 1. dio

- definirati konačnost
- definirati diskretnost
- definirati digitalnost

Konačnost – ima konačan broj stanja, konačnu memoriju

Diskretnost – radi u diskretnom vremenu

Digitalnost – raspolaze digitalnim ulazima i izlazima

21.3. Svojstva automata 2. dio

- definirati determiniranost
- definirati specificiranost
- definirati sinkronost

Determiniranost – jednoznačno obavlja svoju funkciju

Specificiranost

- potpuno: mogući svi nizovi ulaznih događaja (očekuje proizvoljni niz ulaza)
- nepotpuno: mogući su samo neki nizovi ulaznih događaja

Sinkronost – diskretno vrijeme definirano taktnim signalom

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

22. APSTRAKTNI MODEL DIGITALNOG AUTOMATA

22.1. Automat 1. dio

- automat kao petorka
- definirati skupove U , I i S

Automat kao petorka:

Automat se opisuje petorkom: $A = \langle U, I, S, \delta, \lambda \rangle$ i naziva se apstraktni automat jer je definiran matematičkim elementima, skupovima i funkcijama.

Definirati skupove U , I i S :

U je skup ulaznih simbola ili ulazni alfabet koji su u stvarnom automatu kodirani kodnim riječima ulaznih varijabli X :

$$U = \{u_1, u_2, \dots, u_p\}$$

$$X = \{x_1, x_2, \dots, x_e\}$$

$$2^e \geq p$$

I je skup izlaznih simbola ili izlazni alfabet koji su u stvarnom automatu kodirani kodnim riječima izlaznih varijabli Y :

$$I = \{i_1, i_2, \dots, i_q\}$$

$$Y = \{y_1, y_2, \dots, y_m\}$$

$$2^m \geq q$$

S je skup unutarnjih stanja automata koja su u stvarnom automatu kodirana kodnim riječima varijabli Z (a to su kodne riječi stanja bistabila memorije):

$$S = \{s_1, s_2, \dots, s_n\}$$

$$Z = \{z_1, z_2, \dots, z_k\}$$

$$2^k \geq n$$

22.2. Automat 2. dio

- definirati funkcije δ i λ
- zapisivanje automata

Funkcija prijelaza δ određuje slijedeća stanja automata na osnovi sadašnjeg stanja i sadašnjeg ulaza:

$$\delta : \quad s^{n+1} = \delta(s, u)^n \quad S \times U \rightarrow S$$

Funkcija izlaza λ određuje sadašnji izlaz automata. Razlikujemo Mealy i Moore model automata:

$$\lambda : \quad i^n = \begin{cases} \lambda(s, u)^n & \text{Mealy} \\ \lambda(s)^n & \text{Moore} \end{cases} \quad \begin{matrix} S \times U \rightarrow I \\ S \rightarrow I \end{matrix}$$

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

Zapisivanje automata:

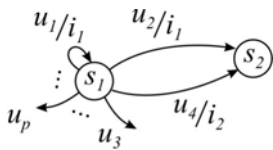
Automat se zapisuje tablicom prijelaza i izlaza, i usmjerenim grafom.

Tablica prijelaza i izlaza za Mealy automat:

	s^{n+1}	i^n
	u_1, u_2, \dots, u_p	u_1, u_2, \dots, u_p
s_1		
s_2	s_j	i_k
\vdots		
s_n		

svako mjesto u tablici odgovara jednom paru s, u .

Usmjereni graf za Mealyev model automata:



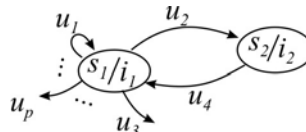
Čvorovi su stanja, usmjerene duljine su prijelazi. Uz duljine pišemo izlaze jer ovise o stanju i ulazu.

Tablica prijelaza i izlaza za Moore automat:

	s^{n+1}	i^n
	u_1, u_2, \dots, u_p	u_1, u_2, \dots, u_p
s_1		
s_2	s_j	i_k
\vdots		
s_n		

izlazi ovise samo o stanju s .

Usmjereni graf za Mooreov model automata:



Čvorovi su stanja, usmjerene duljine su prijelazi. Uz duljine pišemo izlaze jer ovise samo o stanju.

22.3. Sinteza automata

- definirati faze sinteze
- definirati korake za svaku fazu sinteze

Sinteza automata se obavlja u dvije faze, kroz apstraktnu i strukturnu sintezu. Apstraktna se odnosi na definiranje automata kao matematičkog modela, a strukturna se odnosi na sintezu konkretnog digitalnog sklopa.

Koraci sinteze su:

- apstraktna sinteza

- zadavanje automata
- minimizacija automata

- strukturna sinteza

- kodiranje stanja, ulaza i izlaza
- uvrštavanje kodova, prepoznavanje
 - tablica prijelaza za pojedine bistabile
 - tablica istine za izlazne varijable
- realizacija automata
 - općim bistabilima i logičkim vratima
 - MD strukturom i D bistabilima (MDD)

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

23. ZADAVANJE AUTOMATA

23.1. pristupi zadavanju automata

- definirati tri transformacije
- navesti konkretne postupke zadavanja

Definirati tri transformacije:

- Zadavanje automata kao **transformatora sekvence** provodi se kroz postavljanje pravila o transformaciji ulazne na izlaznu sekvencu. (matematičke gramatike)
- Zadavanje automata kao **akceptora sekvence** provodi se kroz prepoznavanje sekvenci koje izazivaju pojavu nekog simbola na izlazu. Govorimo o transformaciji ulazne sekvence na izlazni simbol. (jezik regularnih izraza)
- Zadavanje automata postupkom **korak po korak** provodi se kroz analizu svakog mogućeg para stanje-ulazni simbol. Govorimo o transformaciji ulaznog simbola, uz stanje, na izlazni simbol. (metoda potpunog stabla)

Navesti konkretne postupke zadavanja:

Automat možemo zadati na tri načina:

- potpunim stablom
- tablicom prijelaza i izlaza
- regularnim izrazom

23.2. Vrste ulazne sekvence

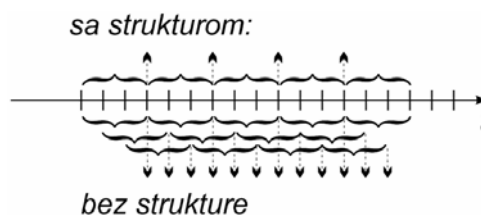
- definirati vrste ulazne sekvence
- skicirati vremenski dijagram odlučivanja

Definirati vrste ulazne sekvence:

Sekvenca bez strukture je beskonačna sekvenca ulaznih simbola, kod koje se tražena sekvenca može pojaviti u proizvoljnom trenutku. Nekađ su čak moguća preklapanja sekvenci na koje automat reagira, pa moramo odlučiti hoće li preklopljena sekvenca djelovati kao sekvenca koja nije preklopljena. Automat mora u svakom trenutku biti u stanju voditi računa o bilo kojem početnom dijelu sekvence koji se upravo dogodio.

Sekvenca sa strukturom je beskonačna sekvenca ulaznih simbola, koja se sastoji od beskonačnog niza konačnih sekvenci. Tražena sekvenca se ovdje može dogoditi nakon što se dogodi prethodna. Automat mora analizirati ulazni niz sinkrono s pojavom konačnih sekvenci. Konačne sekvence mogu biti iste duljine, ali ne moraju. Ako se koriste sekvence različite duljine, održavanje je moguće samo ako kraća sekvenca nije sadržana na početku niti jedne od dužih sekvenci.

Vremenski dijagram ulaznih sekvenci sa i bez strukture:



III. DIO - DIGITALNI SUSTAVI I STRUKTURE

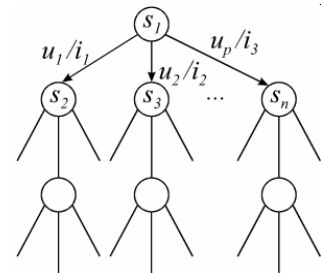
23.3. Postupak zadavanja korak po korak

- definirati postupak
- definirati stablo i potpuno stablo
- svojstva automata s grafom potpunog stabla

Definirati postupak:

Zadavanje automata postupkom **korak po korak** provodi se kroz analizu svakog mogućeg para stanje-ulazni simbol. Govorimo o transformaciji ulaznog simbola, uz stanje, na izlazni simbol. (metoda potpunog stabla)

Potpuno stablo je karakterizirano zasebnim prijelazima u nova stanja za svaki par stanja i ulaznog simbola. Karakteristika potpunog stabla je što za svaki niz događaja (za svaku ulaznu sekvencu) generira zasebno stanje.



Svojstva automata s grafom potpunog stabla:

Velik broj sekvenci za automat nema značenje, a neke sekvence imaju isto značenje kao i druge. Stoga je nepotrebno realizirati automat koji će razlučivati svaku ulaznu sekvencu za sebe. Naprotiv, nastoji se koristiti automat koji ima najmanju razlučivost, dovoljnu za izvršavanje funkcije. Takav automat ima najmanji broj stanja.

23.4. Primjena postupka korak po korak

- svojstva stabla za različite modele automata i sekvence
- tretiranje preklapanja sekvence

Svojstva stabla za različite modele automata i sekvence:

MOORE automat, sekvenca SA strukturom: Od početka rada automat analizira konačnu ulaznu sekvencu i donosi odluku u jednom od stanja posljednje razine. Ta stanja se nazivaju akceptorska stanja, jer je sekvenca prepoznata i simbol generiran. Prvi sljedeći simbol je već prvi simbol sljedeće konačne sekvence.

MOORE automat, sekvenca BEZ strukture: Struktura potpunog stabla i odlučivanje automata od početka rada su jednaki kao za sekvencu sa strukturom. Međutim, nakon donošenja odluke automat prelazi u stanja posljednje razine ($n+1$), zato jer u obzir uzima i prethodnih $n-1$ simbola, kao da je počeo iz početka.

MEALY automat, sekvenca SA strukturom: Ovaj automat izlaz generira na osnovi stanja i ulaznog simbola (graf ima svega n razina, jednu manje nego Moore).

Nakon što je primljen posljednji simbol donosi se odluka bez prijelaza u novo stanje.

MEALY automat, sekvenca BEZ strukture: Graf je sličan kao i sa strukturom, osim što prijelazi posljednje razine stanja idu u istu, posljednju razinu. Na isti način kontroliramo prijelaze akceptorskih stanja.

Tretiranje preklapanja sekvence:

Posebno se kontroliraju prijelazi iz akceptorskih stanja. Ako želimo spriječiti preklapanje, za ta stanja odredimo prijelaze u stanja s_2 i s_3 kao za sekvencu sa strukturom.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

24. EKVIVALENTNOST AUTOMATA

24.1. Odnosi jednakosti među automatima

- vrste jednakosti među automatima
- definicija minimalnosti automata
- objašnjenje razlike broja stanja

Dva automata koji obavljaju istu funkciju (mogu biti različiti po strukturi) su ekvivalentna. Postojanje ekvivalentnog automata objašnjava se sposobnošću razlučivanja ulazne sekvence. Ako dva automata nisu jednaka, a rade istu funkciju, jedan od njih nepotrebno razlučuje ulazne sekvence, iako na kraju za njih donosi istu odluku kao da ih nije razlučio.

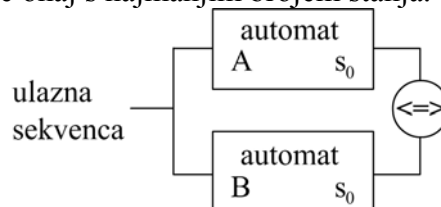
Minimizacija se provodi da se pronađe optimalni apstraktni automat, koji će rezultirati minimalnim sekvencijalnim sklopom. Budući da automat s najmanjom razlučivošću ima najmanji **broj stanja**, proglašavamo ga minimalnim. Cilj minimizacije je pronaći minimalni automat ekvivalentan zadanom, odnosno ako je zadani već minimalan, dokazati njegovu minimalnost.

24.2. Definicija ekvivalentnosti automata

- blok shema testa
- definicija ekvivalentnosti automata

Dva automata (A i B) su ekvivalentna ako počnu raditi iz početnog stanja s_0 , te za istu **proizvoljnu** sekvencu na ulazu daju **istu** sekvencu na izlazu.

Ekvivalentni automati se ne mogu razlikovati u skupovima ulaznih i izlaznih simbola, jer neće biti zadovoljen zahtjev istovjetnosti ulazne i izlaznih sekvenci. Oni se mogu razlikovati samo u skupu stanja, a minimalan je onaj s najmanjim brojem stanja.

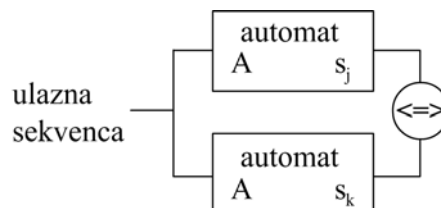


24.3. Definicija ekvivalentnosti stanja

- blok shema testa
- definicija ekvivalentnosti stanja

Stanja koja nepotrebno razlučuju ulazne sekvence, jer za njih automat donosi istu odluku, nazivaju se **ekvivalentna stanja**.

Dva stanja automata A (s_1 i s_2) su ekvivalentna ako automat počne raditi iz jednog pa iz drugog stanja, te za istu **proizvoljnu** sekvencu na ulazu da u oba testa **istu** sekvencu na izlazu.



III. DIO - DIGITALNI SUSTAVI I STRUKTURE

24.4. Nužan i dovoljan uvjet

- definirati nužan uvjet ekvivalencije
- definirati dovoljan uvjet ekvivalencije

Nužan uvjet ekvivalencije kaže da su dva stanja potencijalno ekvivalentna ako su im reci u tablici izlaza automata identični. Ovaj uvjet osigurava da u svakom paru testova prvi simbol izlazne sekvence bude isti.

Dovoljan uvjet ekvivalencije kaže da su dva stanja ekvivalentna ako je zadovoljen nužan uvjet, te ako su im reci u tablici prijelaza automata isti. Ovaj uvjet osigurava da u svakom paru testova, iz promatranih stanja automat prijeđe u isto vrijeme. Time će automat u nastavku testa proći kroz istu trajektoriju stanja, pa je time osigurano da i ostali simboli izlazne sekvence budu isti.

24.5. Minimizacija primitivne tablice

- definirati postupak minimizacije
- definirati minimalizaciju primitivne tablice
- mane jednostavnog postupka minimizacije primitivne tablice

Postupak minimizacije primitivne tablice provodi se nad primitivnom tablicom neposrednom primjenom nužnog i dovoljnog uvjeta ekvivalentnosti stanja. Postupak polazi od pretpostavke da su sva stanja neekvivalentna, te primjenom uvjeta traži moguća ekvivalentna stanja.

Neposredna primjena nužnog i dovoljnog uvjeta ima dvije mane. Prva je u tome što je dovoljan uvjet prestrog, jer ne vodi računa o mogućim neotkrivenim ekvivalentnostima sljedećih stanja. Ovo je moguće djelomično kompenzirati iterativnom primjenom uvjeta ekvivalentnosti. Druga mana proizlazi iz prve, a to je da primjena uvjeta ne može uvijek dati minimalni automat.

Postupak primitivne tablice provodi se u koracima:

1. traže se stanja s istim recima u tablici prijelaza i izlaza;
2. precrtaju se svi redci ekvivalentnih stanja osim jednoga;
3. sve oznake precrtanih stanja zamijenimo oznakom onog neprecrtanog;
4. nastavi se postupak dok ima ekvivalentnih stanja.

25. NAPREDNI POSTUPCI MINIMIZACIJE AUTOMATA

25.1. HM algoritam

- definicija klasa i zatvorenosti
- postupak minimizacije HM algoritmom

Huffman-Mealy algoritam pretpostavlja da su stanja za koja je zadovoljen **nužan uvjet** ekvivalentna. **Klasa** je podskup stanja iz skupa **S** za koja pretpostavljamo da su ekvivalentna. Klasa je **zatvorena** ako sadrži samo jedno stanje ili ako sadrži više stanja koja sva imaju iste prijelaze u klase.

Postupak minimizacije H-M algoritmom:

1. Definiramo primarne klase na osnovu nužnog uvjeta ekvivalentnosti
2. Za sva stanja unutar klase odredimo prijelaze u klase
3. Kontroliramo zatvorenost klasa (isti prijelazi u klase ili samo jedno stanje)
4. Razbijamo otvorene klase i ponavljamo (2) (prema istim prijelazima u klase)
5. Kada su sve klase zatvorene, svaku zamijenimo s jednim stanjem, te ispišemo tablicu minimalnog automata.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

25.2. PU algoritam

- definicija implikacije
- definicija ekvivalentnosti
- tablica implikanata
- postupak minimizacije tablicom implikanata

Paul-Unger algoritam:

Implikacija među skupovima:

Skup S_p impliciran je skupom S_{ri} , ako S_{ri} sadrži sva stanja u koja prelaze stanja iz S_p za promatrani ulaz u_i .

Pri tom skupova S_{ri} ima "p", koliko ima ulaznih simbola.

Ekvivalentnost: S_p sadrži ekvivalentna stanja:

- ako je zadovoljen nužan uvjet ekvivalencije
- ako su svi njemu pripadni S_{ri} ekvivalentni
(svodi se na zadovoljavanje nužnog uvjeta)

Skupove S_p formiramo sistematski 2 po 2 stanja (ispitujemo ekvivalentnost svakog sa svakim)

Tablica implikanata je trokutasta matrica bez dijagonale jer vrijedi

$s_i \Leftrightarrow s_i$...svako stanje ekvivalentno samo sebi pa ne trebamo gledat dijagonalu

$s_i \Leftrightarrow s_j \Rightarrow s_j \Leftrightarrow s_i$...zbog komutativnosti ekvivalencije ne trebamo cijelu matricu nego samo jednu njenu trokutastu polovicu. Tablica ima **n-1** redaka i stupaca. Svako mjesto odgovara jednom paru s_i, s_j . U mjesta tablice upisujemo implikante, a to su skupovi S_{ri}

Postupak minimizacije provodimo:

1. Formiramo trokutastu matricu $(n-1) \times (n-1)$

2. Upišemo implikante

$S_{ri} \Rightarrow$ zadovoljen nužan, ne i dovoljan uvjet

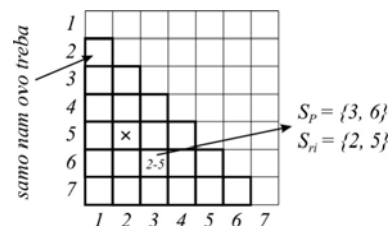
X \Rightarrow nužan uvjet nije zadovoljen (neekv.)

$\checkmark \Rightarrow$ zadovoljeni nužan i dovoljan uvjet (ekv.)

3. Ispitujemo kontradikcije, upisujemo X za otkrivene neekvivalentnosti

4. Ponavljamo (3) ako ima novih neekvivalentnosti

5. Ispisujemo tablicu minimalnog automata. Neprekrižena polja znače ekvivalentna stanja.



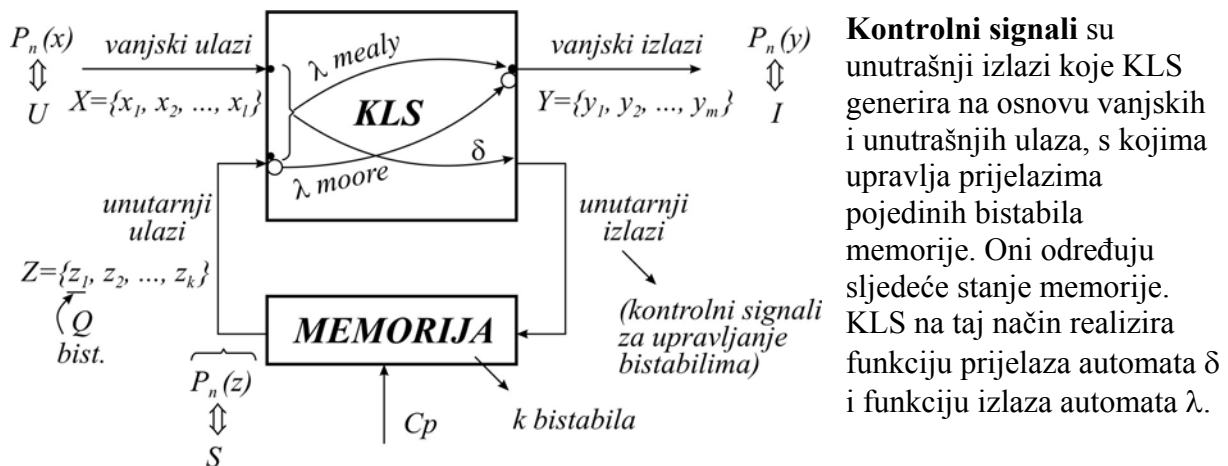
III. DIO - DIGITALNI SUSTAVI I STRUKTURE

26. STRUKTURNA SINTEZA AUTOMATA

26.1. Model realizacije automata

- blok shema modela
- namjena signala
- veza s apstraktnim automatom

Strukturnu sintezu automata vršimo prema **modelu automata** koji se sastoji od kombinacijsko logičke strukture (KLS) i memorije.



Veza s apstraktnim automatom ostvaruje se kroz kodiranje ulaznih i izlaznih simbola, te kroz kodiranje automata.

26.2. Kodiranje automata

- kodiranje ulaza i izlaza
- problem kodiranja stanja
- strategija kodiranja stanja

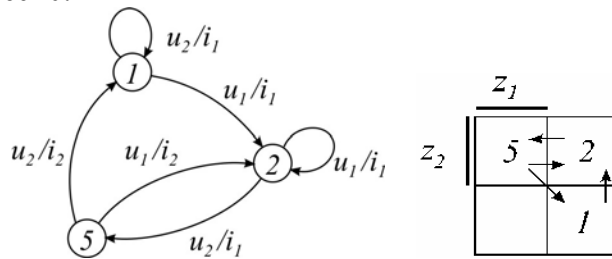
Kodiranje ulaza i izlaza ovisi o okolini automata (kompatibilnosti izvorišta i odredištu informacije), pa često zadani kodovi ulaza i izlaza moraju biti usklađeni.

	s^{n+1}		i^n		U	x_1	I	y_1
	u_1	u_2	u_1	u_2	u_1	0	i_1	0
1	2	1	1	1	u_2	1	i_2	1
2	2	5	1	1				
5	2	1	2	2				

Problem kodiranja stanja – neposredno utječe na veličinu KLS jer raspored nula i jedinica određuju mogućnost minimizacije Booleovih funkcija koje definiraju KLS. Ne postoji egzaktni postupak kodiranja koji daje minimalan sklop.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

Strategija kodiranja stanja – obavlja se po **kriteriju susjednosti**. Stanjima između kojih postoji mogućnost prijelaza dodjeljujemo susjedne kompleksije ili kompleksije sa što manjom distancom. U postupku kodiranja koristimo Veitchev dijagram. Početno stanje obično se kodira kodnom riječi 0.



26.3. Tablica automata s kodovima

- transformiranje tablice
- tablica kao cjelina
- prepoznati dijelovi tablice

Transformiranje tablice

Koristimo jednodimenzionalnu tablicu prijelaza i izlaza apstraktnog automata kao osnovicu za upis kodova, nabranjem parova stanje-ulazni simbol.

S	U	s^{n+1}	i^n
s_1	u_1		
	u_2		
	\vdots		
	u_p		
s_2	u_1		
	u_2		
	\vdots		
	u_p		
\vdots			

Nakon upisa kodova (umjesto simbola i stanja) kodne riječi su poslagane u prirodnom binarnom nizu.

s^n				u^n				s^{n+1}				i^n			
z_1	z_2	\dots	z_k	x_1	x_2	\dots	x_ℓ	z_1	z_2	\dots	z_k	y_1	y_2	\dots	y_m
0	0	\dots	0	0	0	\dots	0								
0	0	\dots	0	0	0	\dots	1								
		\vdots				\vdots		0	0	\dots	0	0	0	\dots	0
1	1	\dots	1	1	1	\dots	0								
1	1	\dots	1	1	1	\dots	1								

Ukupna kodna riječ $zz\dots xx$ čini **lijevu stranu tablice**. **Desnu stranu tablice** prvo čine prijelazi (kodne riječi stanja bistabila u sljedećem diskretnom trenutku), zatim kodne riječi izlaza u sadašnjem trenutku. Svaka kodna riječ stanja sljedećeg trenutka ovisi o kodnoj riječi stanja, te ulaza i izlaza u sadašnjem trenutku. O lijevoj strani ovisi i svaki pojedini bit desne strane.

Prepoznati dijelovi tablice su tablice prijelaza bistabila z i tablice istine izlaznih varijabli y .

	z_1	z_2	x_1	z_1	z_2	y
s_1	0	0	0	0	1	0
			1	0	0	0
s_2	0	1	0	0	1	0
			1	1	1	0
R	1	0	0	R		R
			1			
s_5	1	1	0	0	1	1
			1	0	0	1

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

26.4. Sinteza konkretnog automata

- moguća struktura sklopovlja
- kriteriji sinteze konvencionalnog automata
- kriteriji sinteze MDD strukture

Moguća struktura sklopovlja – integrirani krugovi niske razine integracije (bistabili i logička vrata), integrirani krugovi srednje razine integracije (multiplekseri, demultiplekseri i registri) i programibilne logičke strukture s ugrađenim D bistabilom u izlaznoj makro ćeliji (GAL).

Sinteza memorije se provodi izborom vrste i izračunavanjem broja bistabila **po kriteriju jednoznačnosti kodiranja stanja automata**, a sinteza KLS se provodi korištenjem prepoznatih dijelova tablice prijelaza i izlaza automata.

Veličina multipleksera i demultipleksera se određuje **po kriteriju kvadratičnosti matrice**. Broj multipleksera jednak je broju bistabila **k** uvećanom za broj izlaznih varijabli **p**:

$$M = k + p$$

Zbog jednostavnosti postupka i smanjenja mogućnosti pogreške, redosljed varijabli treba biti usklađen s lijevom stranom kodirane tablice prijelaza i izlaza automata.

27. AUTOMATI I ALGORITMI

27.1. Programabilni automat

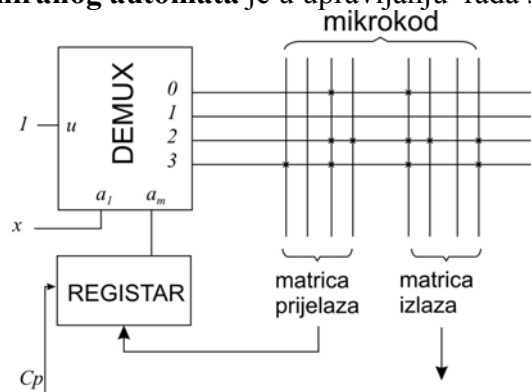
- Wilkiesov model automata
- mikroprogramirani automat
- primjena mikroprogramiranog automata

Wilkiesov model automata je model mikroprogramiranog automata realiziranog MDD strukturom.

Mikroprogramirani automat je automat realiziran MDD strukturom. Sadržaj jednog retka matrice naziva se **mikroinstrukcija**, a sadržaj matrice je **mikroprogram**.

Automat aktivira jedan redak matrice i određuje sljedeće stanje (matrica prijelaza) i izlazne signale (matrica izlaza) ovisno o ulazima i stanju. Matricom prijelaza može se slijedno izvršavati niz uzastopnih redaka ili skakanje naprijed ili natrag u neki drugi dio matrice.

Primjena mikroprogramiranog automata je u upravljanju rada sabirnice i ALU jedinicom.



III. DIO - DIGITALNI SUSTAVI I STRUKTURE

27.2. Algoritam

- definirati algoritam
- primjena algoritma
- istovjetnost algoritma i automata

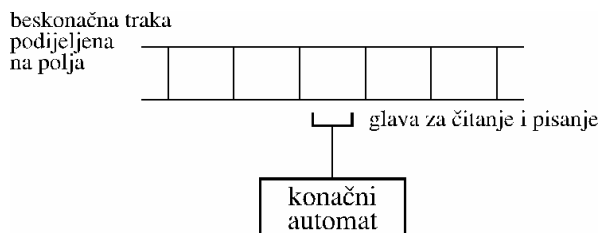
Algoritam je skup transformacija ili instrukcija čijom se primjenom, u konačnom broju redaka, može doći do rješenja bilo kojeg problema iz promatranog skupa (klase) problema.

Primjena algoritma u rješavanju načina sinteze automata i mogućnost njegove primjene.

Istovjetnost algoritma i automata: Svaki se algoritam može zamisliti kao stroj(automat) koji izvodi niz zadanih instrukcija. Svakom algoritmu pripada stroj, a svakom stroju algoritam.

27.3. Turingov stroj

- definirati Turingov stroj
- definirati jezik petorki
- primjena Turingovog stroja



Turingov stroj je algoritam, odnosno model koji raspolaže beskonačnom memorijom, glavom za čitanje/upisivanje (R/W), mogućnošću pomaka glave lijevo ili desno i konačnim digitalnim automatom koji izvršava program.

Rad Turingovog stroja zadan je petorkom: $\langle S_i, T_i, T_j, L, S_j \rangle$

Prva dva elementa (S_i , T_i) daju sadašnje stanje i simbol glave. Ostali elementi opisuju akciju koja će se izvršiti. T_j je novi simbol, S_j je sljedeće stanje automata, dok L i R specificiraju lijevo odnosno desno kretanje, a S zaustavljanje. Nizom ovakvih petorki opisujemo rad Turing-ovog automata. Uočimo da jedna petorka odgovara retku tablice prijelaza i izlaza automata kojem su ulazni i izlazni alfabet identični.

Primjena Turingovog stroja je u pretpostavci mogućnosti realiziranja "bilo kojeg automata", u izračunu svih parcijalno-rekurzivnih funkcija.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28. AUTOMATI I JEZICI

28.1. Značaj analize jezika

- ekvivalentnost programibilnih strojeva (algoritama)
- modeliranje procesa postizanja rješenja
- prepoznavanje pripadnosti rijeci jeziku L

Ekvivalentnost programibilnih strojeva (algoritama):

Dva računala (programabilna stroja) su ekvivalentna ako mogu riješiti isti skup (klasu) problema, bez obzira koliko vremena trebalo pojedinačnom računalu.

Modeliranje procesa postizanja rješenja:

Rješenje problema je preslikavanje problema u rezultat.

Problem i rezultat **su opisani** ulaznim i izlaznim nizovima znakova (riječima), stoga se rješenje problema može **modelirati** ispitivanjem da li riječ koja opisuje problem pripada skupu riječi-jeziku L (Language).

Prepoznavanje pripadnosti riječi jeziku L:

Za prepoznavanje pripadnosti riječi jeziku L koristimo strojeve:

- automate određenih sposobnosti (vrste) za konkretan skup (vrstu) jezika
- određene strukture za konkretan jezik.

28.2. Kompleksnost algoritama

- mjera kompleksnosti
- vrste kompleksnosti

Mjera kompleksnosti:

Mjera kompleksnosti je vrijeme postizanja rješenja u smislu izvršenja jednog koraka Turingovog stroja (TM). Pkušavamo izračunati odnos između broja koraka i duljine ulaznog niza znakova, a taj odnos označavamo s $O(g(n))$, gdje je n = duljina ulaznog niza.

Vrste kompleksnosti:

Ako postoji rješenje:

- pitanje je da li je ostvarivo u stvarnom vremenu
- raspoloživom tehnologijom

Problem kompleksnosti jednostavno rješavamo modelima na bazi Turingovog stroja.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28.3. Izračunljivost

- vrste izračunljivosti
- mogućnost donošenja odluke

Vrste izračunljivosti:

Prihvatljivost rješenja ovisi o prognozi vremena izvršenja:

ovisnost	formula	prognoza
beskonačno	$O(\infty)$	nema algoritamskog rješenja
eksponencijalna:	$O(k^n)$	loša
polinomska	$O(n^k)$	dobro
linearna	$O(kn)$	vrlo dobro
logaritamska	$O(\log(n))$	izvrsno

Mogućnost donošenja odluke:

Pitamo se:

- da li je niz član jezika L (TR, Turing Recognizable)
- da li niz ne pripada jeziku L (co-TR, Complementary Turing Recognizable)

Nekad u konačnom $O()$ vremenu možemo odgovoriti samo na jedno ili ni na jedno pitanje. Pitamo da li je niz $x \in L$ TR i pritom nije co-TR:

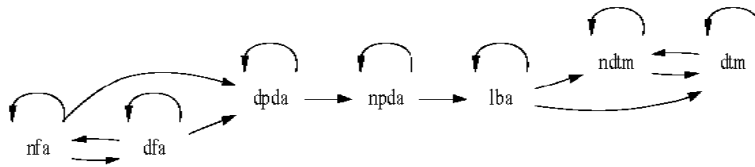
TR	Co-TR	odluka
0	0	ne postoji
0	1	ne postoji
1	0	ne postoji
1	1	postoji, eksponencijalno

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

28.4. Taksonomija automata i jezika

- odnos snage vrsta automata
- ekvivalentnost vrsta automata
- odnos automata i jezika

Odnos snage vrsta automata:



DFA (Deterministic Finite Automata)

NFA (Non Deterministic Finite Automata)

DPDA (Deterministic Push Down Automata)

NPDA (Non Deterministic Push Down Automata)

LBA (Linear Bounded Automata)

DTM (Deterministic Turing Machine)

NDTM (Non Deterministic Turing Machine)

Ekvivalentnost vrsta automata:

Ekvivalentnost raznih vrsta automata:



Odnos automata i jezika:

Jezik L je skup sekvenci izgrađenih od članova skupa simbola Σ .

Za neki skup simbola Σ jezika može biti beskonačno, jer karakteristična sekvenca može biti beskonačna. Nad jezicima su definirane operacije: unije, presjeka, razlike, simetrične razlike, pripajanja, eksponenciranja, iteracije i negacije. Ideja o funkciji automata zabilježava se nekim od standardnih jezika. Automat na osnovi ulazne riječi (sekvence slova ulaznog alfabeta) generira izlaznu riječ (sekvenca slova izlaznog alfabeta).

Jezici za pojedine automate:

- DFA, NFA - regularni jezici
- DPDA – determinirani CFL
- NPDA, PDA – CFL itd.

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

29. ALGEBRA DOGAĐAJA

29.1. Elementarni i složeni događaji

- definicija elementarnog događaja
- elementarni događaj i diskretno vrijeme
- složeni događaji
- ulaz automata i algebra događaja

Definicija elementarnog događaja: nedjeljivi događaj, dogodi se kao cjelina.

Elementarni događaj i diskretno vrijeme: dogodi se samo JEDAN u trenutku diskretnog vremena.

Složeni događaji:

- nastaje kao kombinacija elementarnih događaja
- u suštini je vremenski niz - sekvenca - elementarnih

ULAZ AUTOMATA:

- ulazi u automat daju mu informaciju o okolini
- ulazni simboli dolaze u vremenskim nizovima
- to su sekvence u diskretnom vremenu
- te smo sekvence nazivali nizovima DOGAĐAJA

ALGEBRA DOGAĐAJA:

- algebra koja uzima u obzir vremenske odnose
- bavi se nizovima DOGAĐAJA
- uvodi ALGEBARSKE OPERACIJE među događajima

29.2. Operatori algebre događaja

- definicija i svojstva operatora algebre događaja
- definicija regularnog izraza i događaja

Definicija i svojstva operatora algebre događaja:

DISJUNKCIJA: $R = R_1 \vee R_2$

- složeni događaj od dva ili više (elementarnih) događaja
- dogodi se kad se dogodi JEDAN OD disjunktivno vezanih članova (događaja)
- traje koliko traje disjunktivni član koji se dogodio
- vrijedi svojstvo komutativnosti: $R_1 \vee R_2 = R_2 \vee R_1$

KONJUNKCIJA: $R = R_1 \& R_2$

- složeni događaj od dva ili više (elementarnih) događaja
- dogodi se kad se dogode SVI konjunktivno vezani članovi onim redom kojim su zapisani
- traje koliko traju SVI članovi
- opisuje SEKVENCU događaja
- ne vrijedi svojstvo komutativnosti: $R_1 \& R_2 \neq R_2 \& R_1$

NEGACIJA: $R = \overline{R_1}$

- događaj koji se dogodi kad se NE dogodi negirani događaj
- to je dakle skup SVIH događaja OSIM negiranog događaja
- problem je u određivanju POTPUNOG SKUPA događaja (može biti beskonačan)
- stoga se u praksi ne koristi
- definicija regularnog izraza i događaja

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

REGULARNI IZRAZI:

- to su KONAČNI algebarski izrazi u okvirima algebre događaja

REGULARNI DOGAĐAJ:

- svaki složeni događaj koji možemo izraziti konačnim algebarskim izrazom - regularnim izrazom

30. ZADAVANJE AUTOMATA REGULARNIM IZRAZOM

30.1. Zadavanje automata s pomoću RI

- pristup zadavanju automata s RI
- tehnika pisanja RI za sekvencu sa strukturom
- tehnika pisanja RI za sekvencu bez strukture

Ulazni simbol automata je elementarni događaj. Sekvenca ulaznih simbola je složeni događaj. Promatramo automat kao akceptor sekvence. Stoga trebamo opisati traženu sekvencu, ali i sekvence koje nisu tražene jer ih automat mora odbaciti.

Regularnim izrazom opisujemo ulaznu sekvencu tako da izdvojimo:

- dio sekvence koji nije tražena sekvenca
- dio sekvence koji jest tražena sekvenca

Ako ima više akceptorskih izlaznih simbola pišemo zasebni regularni izraz za svaki simbol.

Tehnika pisanja RI za sekvencu sa strukturom:

- iteracijskom zagradom obuhvatimo sekvence koje NISU tražene
- običnom zagradom obuhvatimo sekvence koje su tražene za promatrani akceptorski (aktivni) izlazni simbol
- pišemo više izraza ako je više akceptorskih izlaznih simbola
- automat u svim diskretnim trenucima, u kojima nije donio odluku, na izlazu daje neutralni (neaktivni) izlazni simbol

Tehnika pisanja RI za sekvencu bez strukture:

- analiziramo dijelove ulazne sekvence
- iteracijskom zagradom obuhvatimo sekvence do duljine tražene koje NISU početak tražene
- upišemo prvi simbol tražene sekvence
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon prvog simbola koje nisu početak tražene, ali čiji je zadnji simbol prvi simbol tražene sekvence (ako takvih ima)
- upišemo drugi simbol tražene sekvence
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon drugog simbola koje nisu početak tražene, ali čija su zadnja dva simbola prva dva simbola tražene sekvence (ako takvih ima)
- upišemo treći simbol tražene sekvence
- iteracijskom zagradom obuhvatimo ostatke sekvenci nakon trećeg simbola koje nisu početak tražene, ali čija su zadnja tri simbola prva tri simbola tražene sekvence (ako takvih ima)
- itd. do ispisa tražene sekvence

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

30.2. Indeksiranje RI

- definicija mjesta i vrste mjesta
- analogija mjesta i rada automata
- osnovno indeksiranje
- pravila za rasprostiranje indeksa

Mjesta unutar regularnog izraza mogu biti:

Osnovna - ona koja se nalaze s lijeve strane obične i iteracijske zagrade i s desne strane slova.

Predosnovna - ona koja se nalaze s desne strane obične i iteracijske zagrade i s lijeve strane slova. Između dva slova (simbola) mjesto je istovremeno osnovno i predosnovno, smatramo ga **osnovnim**. Između dvije zagrade mjesto je istovremeno osnovno i predosnovno, smatramo ga **predosnovnim**.

Analogija mjesta i rada automata:

Osnovna mjesta odgovaraju **stanjima** automata. To su stanja u koja dolazimo nakon određene sekvence ulaznih događaja. **Predosnovna** mjesta odgovaraju **prelazima** automata, a to su stanja iz kojih idemo u prelaze. **Završno** mjesto u RI odgovara akceptorskom stanju digitalnog automata.

Osnovno indeksiranje:

Osnovna mjesta označimo kratkim okomitim crtama i indeksiramo u prvom redu ispod izraza. Predosnovna mjesta označimo dugim okomitim crtama i indeksiramo u drugom redu ispod izraza. Osnovnim mjestima dodijelimo vlastite indekse, npr. redom 1, 2 ... Na predosnovna mjesta rasprostiremo indekse osnovnih mjesta koristeći 5 pravila.

Pravila za rasprostiranje indeksa su:

1. indeks mjesta ispred **obične ili iteracijske** zagrade rasprostiremo na početna predosnovna mjesta disjunktivno vezanih članova unutar zagrade (zato jer se zagrada - disjunkcija dogodi, kad se dogodi jedan od disjunktivno vezanih članova)
2. indeks mjesta ispred iteracijske zagrade rasprostiremo na predosnovno mjesto **iza** zagrade (zato jer iteracija sadrži i praznu sekvencu)
3. indeks završnih mjesta disjunktivno vezanih članova unutar **obične ili iteracijske** zagrade rasprostiremo na predosnovno mjesto **IZA** zagrade (zato jer se zagrada - disjunkcija dogodi, kad se dogodi jedan od disjunktivno vezanih članova)
4. indekse svih mjesta koja smo rasprostirali na predosnovno mjesto **iza iteracijske** zagrade rasprostiremo na početna predosnovna mjesta disjunktivno vezanih članova unutar zagrade (zato jer iteracija dozvoljava proizvoljno ponavljanje)
5. indeks završnog mjesta regularnog izraza rasprostiremo na ona predosnovna mjesta, na koja se je rasprostirao indeks početnog mjesta regularnog izraza. Iznimno, ako za sekvencu bez strukture dozvoljavamo preklapanje, indeks završnog mjesta rasprostiremo na mjesta na koja se rasprostirao indeks mjesta u koje dolazimo nakon dijela sekvence koji se preklapa

III. DIO - DIGITALNI SUSTAVI I STRUKTURE

30.3. Dobivanje strukture automata iz RI

- razrješenje izlaznog simbola
- razrješenje nastavka rada automata
- redukcija indeksa
- ispis primitivne tablice automata

Razrješenje izlaznog simbola (ovisno o vrsti automata):

Za **MOORE** akceptorski izlazni simbol dodijelimo završnom **osnovnom** mjestu regularnog izraza. Za **MEALY** akceptorski izlazni simbol dodijelimo posljednjem **predosnovnom** mjestu regularnog izraza, a to je uvijek mjesto **ispred** posljednjeg simbola tražene sekvence. Za sva ostala mjesta **podrazumijevamo neutralni** izlazni simbol.

Razrješenje nastavka rada automata:

Indeks završnog mjesta regularnog izraza rasprostiremo na ona predosnovna mjesta, na koja se je rasprostirao indeks početnog mjesta regularnog izraza. Iznimno, ako za sekvencu bez strukture dozvoljavamo preklapanje, indeks završnog mjesta rasprostiremo na mjesta na koja se rasprostirao indeks mjesta u koje dolazimo nakon dijela sekvence koji se preklapa.

Pravila za redukciju indeksa:

Indekse **svih** mjesta koji su se rasprostirali na **isto** predosnovno mjesto zamijenimo jednim indeksom, pod uvjetom da su im dodijeljeni **isti** izlazni simboli. Indekse **svih** mjesta u koja dolazimo iz **istog** predosnovnog mjesta s **istim** ulaznim simbolom zamijenimo jednim indeksom.

Ispis primitivne tablice automata:

Primitivnu tablicu ispišemo tako da preostale indekse osnovnih mjesta uzmemo za stanja automata, a njihove prelaze odredimo preko ulaznih simbola.