Uvod u teoriju računarstva
1. MI – Tutorial – 2006/07

1. Zadatak (by b0ysee):

Jezik L nad abecedom {0, 1, 2} sadrži sve nizove u kojima nema uzastopnog ponavljanja znaka 1. Konstruirati konacni automat koji prihvaca nizove iz jezika L.

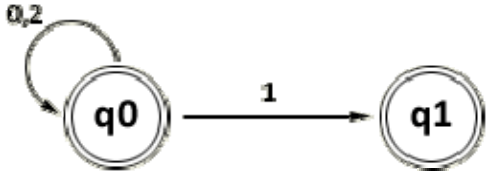
Rjesenje:

Prvo treba razmisljati sto se uopce trazi od nas u zadatku.
Imamo automat koji prima 3 moguca znaka: 0, 1 i 2. Ako prima 0 ili 2, to nam je potpuno prihvatljivo i to u bilo kojoj mjeri (000220202002202002202020 20... je savrseno prihvatljivo).
Problem se javlja ako se pojavi 2 puta za redom broj 1.
Kod zadataka koji su zadani ovako, ajmo reci, nezgrapno, bez nekih specificnih podataka najlakse je ici po redu i vidjeti sto se dogadja. Pa, tako cu ici i ovdje i to crtezom jer je to nekako najlakse za ovakve zadatke.

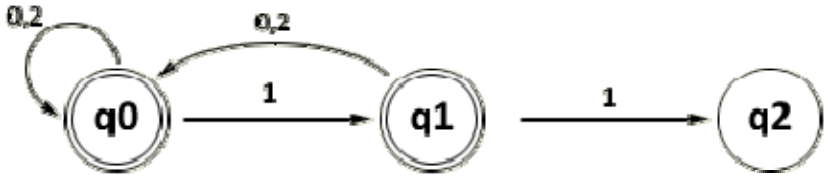
Prva stvar koju moramo imati je pocetno stanje koje, ako razmislimo o tome, mora biti prihvatljivo. Mora biti prihvatljivo jer je to stanje u kojem cemo se cijelo vrijeme zadrzavati koliko god znakova 0 ili 2 doslo.
To bi izgledalo ovako:

Sad imamo pocetno stanje koje smo nazvali q0 koje je prihvatljivo (sto se oznacava dodatnim rubom oko stanja).

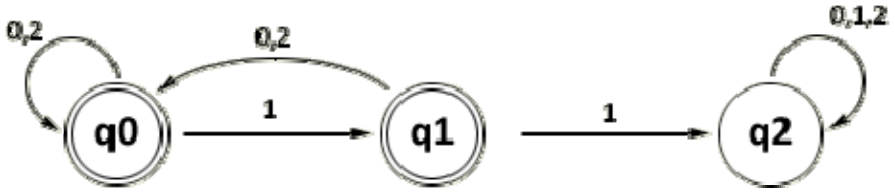
Rijesili smo problem pojavljivanja 0 i 2, ali sto ako se pojavi jedna ili vise 1?
Ako se pojavi 1 bilo bi dobro otici u novo stanje iz razloga sto moramo imati neku vrstu "brojaca". Zato ce to dodatno stanje sluziti kao brojac koji broji jedno pojavljivanje jedinice:



Kada se pojavi jedna jedinica moguca su dva sljedeca dogadjaja.
Prva moguca stvar jest da se pojavi 0 ili 2 koje su prihvatljive i omogucuju da se jedinica opet pojavi, a da automat i dalje ostane prihvatljiv. Stoga ce se taj dogadjaj smatrati kao smanjivanje "brojaca", a to smanjivanje ostvarit cemo tako da cemo se vratiti natrag u pocetno stanje.
Druga moguca stvar jest da se ponovno pojavi jedinica. Nakon toga bi trebalo nas "brojac" opet povecati, a to je najlakse tako da se ostvari jos jedno stanje!
Posto je stanje q1 i dalje bilo prihvatljivo (unutar granica zadatka) sljedece stanje vise nikako ne moze biti prihvatljivo. Razlog tome je sto se dogodilo da se ostvario jedini uvjet neprihvacanja automata, dvije uzastopne jedinice:



Ostaje jos samo odgovoriti na pitanje: "A sto ako dodjemo do stanja q2?" Odgovor bi bio - nista. U ovom slucaju, kada se dostigne to neprihvatljivo stanje vise ga nikako ne mozemo popraviti posto ce u tom nizu zauvijek ostati negdje 2 ili vise uzastopnih jedinica. Stoga cemo to oznaciti kao da se vrtimo u krug:



2. Zadatak (by b0ysee):

Konstruirati NKA koji prepoznaje sve binarne brojeve za koje vrijedi bar jedan od navedenih uvjeta:
a) u binarnom broju postoje dvije ili više uzastopnih nula
b) suma znakova je manja od 4

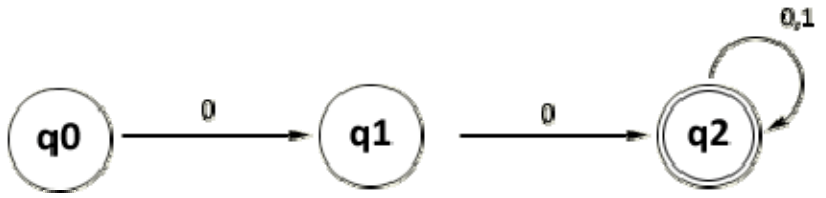
Rjesenje:

Posto u proslom zadatku nismo dobili uvjet kakav automat mora biti, tog pitanja se nisam ni dotakao. Ovdje smo dobili uvjet da nas automat mora biti tipa NKA. U proslom zadatku smo konstruirali DKA.
Koja je razlika?
Kod DKA funkcija prijelaza daje rezultat samo jedno stanje automata, dok kod NKA funkcija prijelaza daje skup stanja. To je sve, makar to dovoljno komplicira stvar. :)

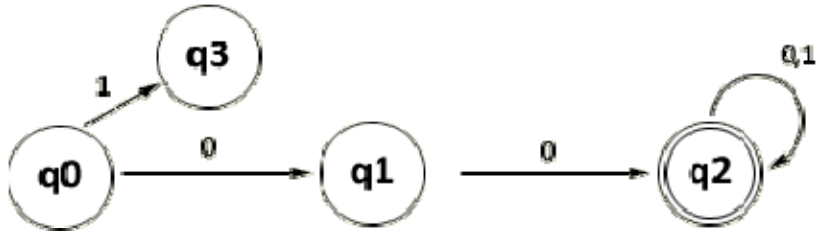
Ovdje konstruiramo NKA koji ima dva uvjeta. VRLO je bitno primjetiti kako su ta dva uvjeta odvojeni s ili. To znaci da ce nas automat ici u dva smjera, za svaki uvjet svoj smjer. U tom trenutku treba paziti na sljedece - kad dobivamo ulazni znak, recimo 0 nas ce automat paralelno ici niz grane za oba uvjeta, te pokusati odrediti prihvatljivost. Ako je bilo koje od stanja u kojima smo zavrsili prihvatljivo, ulazni niz je prihvatljiv.

Vrijeme je za zadatak. :)
Prva stvar koju treba uociti je da se radi o binarnim brojevima. Zbog toga su moguci ulazni znakovi samo 0 ili 1.
Druga stvar je ona koju smo vec spomenuli, a to je da cemo raditi dva odvojena puta, za svaki od uvjeta. Medjutim, nismo spomenuli kako ce se pojaviti dodatno stanje koje ce nam osiguravati da se nakon prvog poslanog znaka maknemo iz pocetnog stanja kako ne bi ulazili u tudju uvjetnu granu. Ovo ce biti jasnije kada se krene s rjesavanjem.

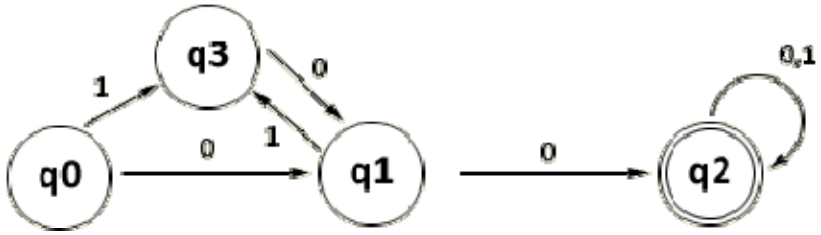
Uvjet a
Uvjet a kaze kako ce nas broj biti prihvatljiv ako se ostvari dvije ili vise uzastopnih nula. Moj prijedlog je da prvo ispunite uvjet posto tocno znate kako to mora izgledati, a tek nakon toga ispunjavate sve moguce rupe.
U nasem slucaju moramo iz pocetnog uvjeta za prvu nulu krenuti u novo stanje (princip "brojaca" iz proslog zadatka), pa za sljedecu opet krenuti u novo stanje koje ce tada biti prihvatljivo, te ce se (analogno prvom zadatku) tamo zadrzavati:



Odlicno, ostvarili smo jedan od dva uvjeta, samo sto je trenutno jos nepotpun.
Nekako najlogicniji nacin gledanja sto nedostaje je jednostavno pogledati sto je od mogucih ulaznih znakova ostalo i to pridodati postojecem grafu. U nasem slucaju nedostaje nam izlaz 1 iz q0 i izlaz 1 iz q1.
Ovdje bi bilo lijepo jednostavno vratiti u slucaju jedinice natrag u q0 (i tako smanjiti nas "brojac"), no treba misliti na gorenavedeni razlog. Naime, zelimo se u svakom slucaju maknuti uz q0 jer ga dijelimo s drugim uvjetom (i njegovom granom do koje trenutno jos nismo dosli).
Jedino sto mozemo je stvoriti jos jedno stanje u koje cemo odmah pobjeci iz q0 za slucaj jedinice, te ce to stanje "glumiti" q0:



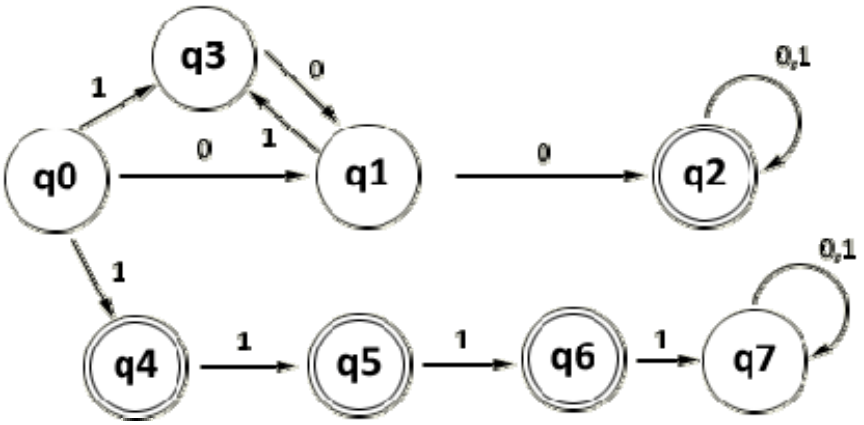
To stanje je, naravno, neprihvatljivo za nas uvjet, ali nam pomaze izbjeci neugodnosti zbog grananja uvjeta. Trenutno nam ostaje jos popuniti izlaz 1 i 0 iz q3, te izlaz 1 iz q1.
Iz q3 se takodjer ne smijemo vracati u q0, ali pogledajmo sto mozemo. Ako slucajno dodje jedinica, pa jedino logicno je ostati u istome stanju posto q3 trenutno "glumi" q0, a to bi inace radili u q0.
Ako dodje nula ici cemo u q1 jer bi to ucinio i q0.
Sto se tice q1, za izlaz 1 inace bismo isli u q0, ali posto q3 "glumi" q0, radje cemo otici u njega:



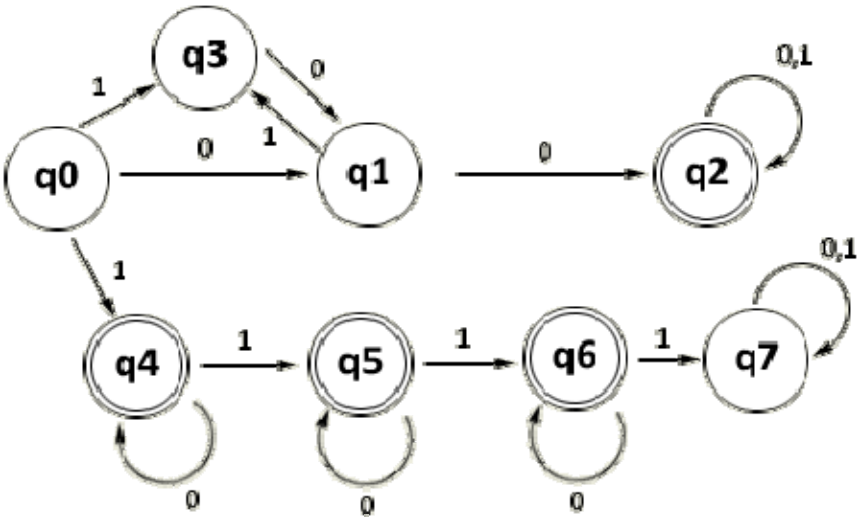
Time smo ostvarili uvjet a! :)

Uvjet b
Uvjet b nije nista tezi od uvjeta a. Jedino sto treba odmah na pocetku reci je to da cemo i ovdje "bjezati" iz stanja q0, ali otom potom. Prvo cemo pokusati ostvariti uvjet koji je zadan jer to uvijek mozemo bez razmisljanja.

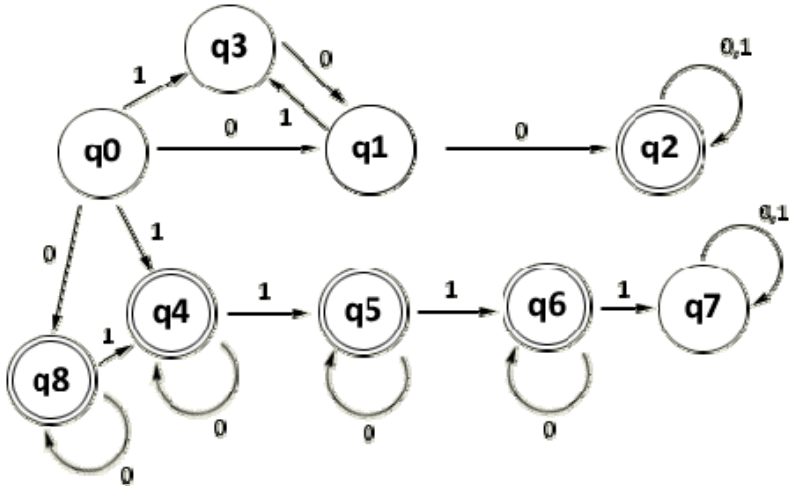
Potrebno je ostvariti novi "brojac", ali ovdje uvjet kaze da SUMA ne smije preci 4. To bi konkretno znacilo da nas "brojac" nece imati mogucnost smanjivanja, vec da ce samo rasti. ;)
Krenut cemo iz istog stanja i imati cemo 3 prihvatljiva stanja u koja cemo izravno doci dodavanjem jedinica. To radimo zato da bi dosli do krajnje granice uvjeta, tj. do sume koja ce tada iznositi 3. Ako dodamo jos jednu jedinicu, tada ce nas automat prijeci u nepovratno neprihvatljivo stanje (potpuno analogno prvom zadatku):



Sada imamo sredjen put do ispunjenja uvjeta. Posto smo rekli kako ovaj dio automata u biti ne "broji" vec "sumira", nece moci vratiti "brojac" unatrag. To znaci da ce za bilo koju 0 koja dodje u stanjima q4, q5 ili q6 on ostati u gdje je i bio:



Odlicno, ovaj automat vec izgleda vrlo dobro. :)
Jedina stvar koja sada nedostaje je "bjezanje" iz stanja nula za ovu granu (uvjeta b). To cemo ostvariti na isti nacin kao i za uvjet a. Uvest cemo novo stanje koje ce "glumiti" stanje q0. To znaci da ce za ulaz 1 otici u stanje q4 (kao sto je to q0 ucinio), a za stanje 0 ce ostati sam u sebi posto 0 ne pridonosi sumi. Takodjer, ovo ce stanje biti prihvatljivo posto je suma u tom trenutku daleko ispod 4:



Nas je NKA automat gotov! :)
Ako sada dobro pogledate sliku lako se uoci jos jedan JAKO DOBAR razlog zasto se "bjezi" iz stanja q0. To je zato sto su stanja q3 i q8, koji oba "glume" q0, razlicitih prihvatljivosti, tj. q3 nije prihvatljiv, a q8 jest!

3. Zadatak (by b0ysee):

Minimizirati zadani DKA primjenom algoritma podjele stanja (2. algoritam). Automat dodatno smanjiti pretvorbom znakova.

	a	b	c	d	
q_0	q_0	q_1	q_1	q_2	0
q_1	q_1	q_3	q_2	q_5	0
q_2	q_2	q_5	q_5	q_0	1
q_3	q_7	q_4	q_4	q_0	1
q_4	q_5	q_0	q_0	q_3	0
q_5	q_4	q_0	q_0	q_7	0
q_6	q_3	q_4	q_4	q_8	0
q_7	q_2	q_4	q_4	q_0	1
q_8	q_8	q_7	q_3	q_6	1

Rjesenje:

Minimizarnje po 2. algoritmu podijelit cemo u dva koraka.
Prvo cemo maknuti sva nedohvatljiva stanja, a nakon toga cemo ici na sam proces minimizacije.

1. korak

Zasto je tome tako?
Nema potrebe obavljati minimizaciju na cijelom skupu stanja, ako se do nekih nikada niti ne dolazi (njih se zamisli kao kruzic iz proslih zadataka, ali onak sam po sebi negdje sastrane, nebitan, tuzan i sam ili nesto :)).
Izbacivanje nedohvatljivih stanja radi se tako da se krene od pocetnog stanja, q0, te se, ovisno o ulazu, gleda u koja sve stanja mozemo preci.
Konkretno:
q0 -> za ulaz "a" ide u stanje q0, za "b" ide u q1, za "c" ide u q1 i za "d" ide u q2
Ona stanja u koja mozemo doci nekim od ulaznih znakova oznacit cemo kao dohvatljiva jer se do njih zaista i moze doci, to je najlaske samo staviti nekakvu kvacicu. :)

	a	b	c	d	
q_0 ✓	q_0	q_1	q_1	q_2	0
q_1 ✓	q_1	q_3	q_2	q_5	0
q_2 ✓	q_2	q_5	q_5	q_0	1
q_3 ✓	q_7	q_4	q_4	q_0	1
q_4	q_5	q_0	q_0	q_3	0
q_5	q_4	q_0	q_0	q_7	0
q_6	q_3	q_4	q_4	q_8	0
q_7	q_2	q_4	q_4	q_0	1
q_8	q_8	q_7	q_3	q_6	1

Sada treba uciniti isto za sva stanja koja su pod kvacicom.
q1 -> za ulaz "a" ide u q1, za "b" ide u q3, za "c" ide u q2, za "d" ide u q5
U ovom smo koraku nasli jos dva stanja koja su dostupna - q3 i q5.
Tako se obavlja za sva stanja koja dodju pod kvacicu. Na kraju kada uspijemo proci sve, potrazimo stanja koja nemaju kvacicu i njih prekrizimo!

	a	b	c	d	
q_0	q_0	q_1	q_1	q_2	0
q_1	q_1	q_3	q_2	q_5	0
q_2	q_2	q_5	q_5	q_0	1
q_3	q_7	q_4	q_4	q_0	1
q_4	q_5	q_0	q_0	q_3	0
q_5	q_4	q_0	q_0	q_7	0
q_7	q_2	q_4	q_4	q_0	1

Time smo izbacili stanja q6 i q8.
Skup stanja koji je jos uvijek u igri i ulazi u 2. korak je: {q0, q1, q2, q3, q4, q5, q7}

2. korak
Sad, vrlo je bitno prvo objasniti princip ovog algoritma.
2. algoritam radi tako da uzima sve skupove stanja koje sami navedete (mi trenutno imamo samo jedan, no to ce se promijeniti) i pokusa ih podijeliti na dva dijela. Dijeljenje radi tako da uzme dva stanja iz istog skupa i gleda pripadaju li njihova moguca stanja, koja se podudaraju (za isti ulazni znak), istom skupi, ali istom skupu u proslom koraku. Ako je to tako, on dijeli taj skup na dva dijela i sve preostale znakove iz skupa trpa ili jednome ili drugome, ovisno kako ce zadovoljiti vec navedeni uvjet.
Ako ne uspije podijeliti skup, to znaci da je taj skup vec sredjen i mozemo ga ostaviti na miru.

To vjerojatno trenutno ne zvuci najbolje, ali kroz primjer ce ici lakse.
Krenimo prvo s time da prikazemo nasu tablicu iznova:

	a	b	c	d	
q_0	q_0	q_1	q_1	q_2	0
q_1	q_1	q_3	q_2	q_5	0
q_2	q_2	q_5	q_5	q_0	1
q_3	q_7	q_4	q_4	q_0	1
q_4	q_5	q_0	q_0	q_3	0
q_5	q_4	q_0	q_0	q_7	0
q_7	q_2	q_4	q_4	q_0	1

Posto u prvom koraku nemamo prethodni korak u kojem bismo gledali da li neka dva stanja pripadaju u isti skup, koristit cemo njihov uvjet podudarnosti. Sva prihvatljiva stanja trpamo u jednu, a sva neprihvatljiva u drugu grupu.
Inace, oznaka grupa je sljedeca: Gxy gdje je x oznaka koraka koji se radi, a y grupa u trenutnom koraku.

G11 = {q0, q1, q4, q5}, G12 = {q2, q3, q7}

Prva grupa sadrzi sva stanja koju su neprihvatljiva, a druga grupa sadrzi prihvatljiva stanja.
Time je prvi korak gotov.

Krenimo na sljedeci po principu koji sam vec objasnjavao.
Moramo proci oba skupa, pa krenimo s G11:
G11 = {q0, q1, q4, q5}

Usporedjujemo po redu - q0 i q1.
- q0 za "a" ide u q0, q1 za "a" ide u "q1"
Sada dolazi do izrazaja ono sto sam rekao. Gledamo skupove u proslom koraku i pazimo nalaze li se stanja U koja idemo u istom skupu. U ovom slucaju su to stanja q0 i q1 koja su oba u skupu G11, tako da je sve u redu.
- q0 za "b" ide u q1, q1 za "b" ide u q3
Evo ga, nase prvo pravo dijeljenje! Jao srece! :)
q1 pripada skupu G11, a q3 pripada skupu G12, tako da moramo nas skup G11 ipak rastaviti na dva skupa i to:

G21 = {q0,
G22 = {q1,

Primjetite kako nijedan od skupova nije zatvoren, posto imamo jos dva stanja u G11 koja treba provjeriti. To se radi tako da se i dalje provjerava s q0. Ako mu pase (ne nadje se ni jedan neodgovarajuci stupac) ostaje u G21 s q0, a u suprotnom ide u G22.

- q0 za "a" ide u q0, q4 za "a" ide u q5
Oba su stanja iz G11, tako da je to u redu.
- q0 za "b" ide u q1, q4 za "b" ide u q0
Oba G11, sve OK.
- q0 za "c" ide u q1, q4 za "c" ide u q0
Oba G11, sve OK.
- q0 za "d" ide u q2, q4 za "d" ide u q3
Oba su stanja iz G12, tako da je i to u redu (da, dovoljno je da budu u istom skupu, nebitno je li taj skup onaj koji trenutno obradjujemo).
q0 i q4 ostaju skupa! :)

G21 = {q0, q4
G22 = {q1

Sada usporedjujemo q0 i q5. Da ne pisem opet sve, q5 ostaje s q0 i tako smo prosli sva stanja skupa G11.

G21 = {q0, q4, q5}
G22 = {q1}

Ostaje nam provjeriti G12.

G12 = {q2, q3, q7}

Da sada opet sve ne raspisujem (to sami ucinite za vjezbu), q2, q3 i q7 ce ostati u istom skupu kojeg cemo nazvati G23, tj.

G23 = {q2, q3, q7}

Treba primjetiti kako cemo sada ovaj skup samo prepisivati u daljnjim koracima posto je on prosao bez dijeljenja.
Stanje nakon 2. koraka:

G21 = {q0, q4, q5}
G22 = {q1}
G23 = {q2, q3, q7}

U treci korak ulazimo s ova tri skupa. Medjutim, treba primjetiti kako skupove G22 i G23 zapravo necemo dirati u trecem koraku posto skup G22 ima samo jedan clan, a G23 je vec oznacen kao nedjeljiv.

Usporedjujemo q0 i q4, ALI treba primjetiti da cemo sada koristiti skupove iz 2. koraka, a ne vise iz 1. (treba se sjetiti da uvijek koristimo skupove iz 1 koraka prije)
- q0 za "a" ide u q0, q4 za "a" ide u q5
Oba su stanja iz G21, tako da je to u redu.
- q0 za "b" ide u q1, q4 za "b" ide u q0
Evo greske, q1 se nalazi u G21, a q0 u G22, tako da ovo ipak moramo podijeliti.

G31 = {q0,
G32 = {q4,

Usporedjujuci q0 i q5 dolazimo do zakljucka da si ni q0 i q5 ne odgovaraju, tako da cemo i q5 strpati u G32 i konacno stanje izgleda ovako:

G31 = {q0}
G32 = {q4, q5}
G33 = {q1}
G34 = {q2, q3, q7}

Trenutno su zavrseni skupovi G31 (jedan clan), G33 (jedan clan) i G34 (jos u 2. koraku). Ostaje provjeriti skup G32 u 4. koraku.

Da opet sve ne pisem, isto je kao i prije samo sto koristite ovo zadnje stanje skupova (1 korak unazad), ispada kako su q4 i q5 zaista u istom skupu.
Konacno stanje nakon svega:

G41 = {q0}
G42 = {q4, q5}
G43 = {q1}
G44 = {q2, q3, q7}

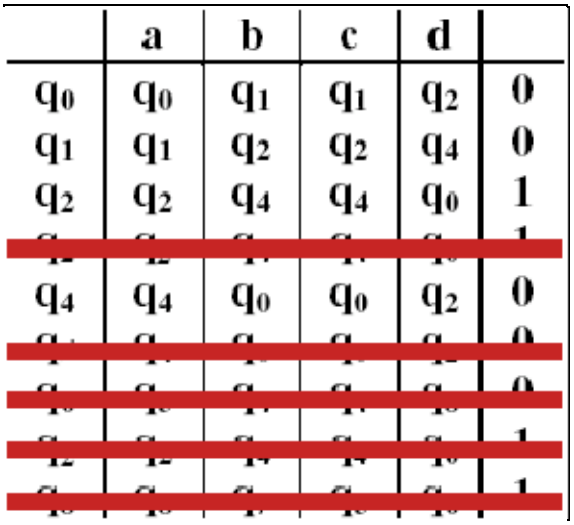
Preostaje nam samo odrediti koja su to istovjetna stanja koja se smiju preimenovati, te krizati isti retci.
Sva stanja koja su u istom skupu su istovjetna. U nasem slucaju to su q4 i q5, te q2, q3 i q7, tj.

q4 = q5
q2 = q3 = q7

Sada mozemo sve q5 u tablici preimenovati u q4, a q3 i q7 u q2 i pogledati sto imamo:

	a	b	c	d	
q_0	q_0	q_1	q_1	q_2	0
q_1	q_1	q_2	q_2	q_4	0
q_2	q_2	q_4	q_4	q_0	1
q_2	q_2	q_4	q_4	q_0	1
q_4	q_4	q_0	q_0	q_2	0
q_4	q_4	q_0	q_0	q_2	0
q_2	q_2	q_4	q_4	q_0	1

Retci 3, 4 i 7 su jednaki, tako da 4. i 7. redak mozemo prekriziti. Takodjer retci 5 i 6 su jednaki, tako da 6. mozemo prekriziti. To radimo zato sto zelimo samo po jedan primjerak svakog retka, tj. izbaciti sve duplikate.
Na kraju tablica izgleda ovako:

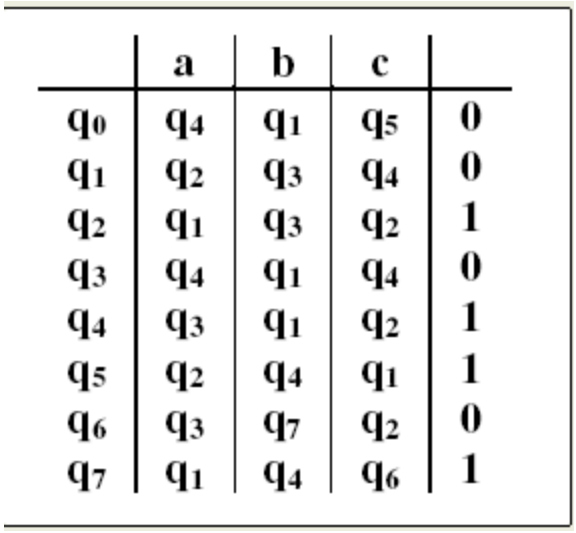


	a	b	c	d	
q_0	q_0	q_1	q_1	q_2	0
q_1	q_1	q_2	q_2	q_4	0
q_2	q_2	q_4	q_4	q_0	1
q_4	q_4	q_0	q_0	q_2	0

Naravno, na ispitu ili bilogdje napravit novu tablicu bez ovih precrtavanja, ne budite lijeni kao ja. :)

4. Zadatak (by b0ysee):

Minimizirati zadani DKA primjenom algoritma pronalaženja neistovjetnih stanja (3. algoritam).



	a	b	c	
q_0	q_4	q_1	q_5	0
q_1	q_2	q_3	q_4	0
q_2	q_1	q_3	q_2	1
q_3	q_4	q_1	q_4	0
q_4	q_3	q_1	q_2	1
q_5	q_2	q_4	q_1	1
q_6	q_3	q_7	q_2	0
q_7	q_1	q_4	q_6	1

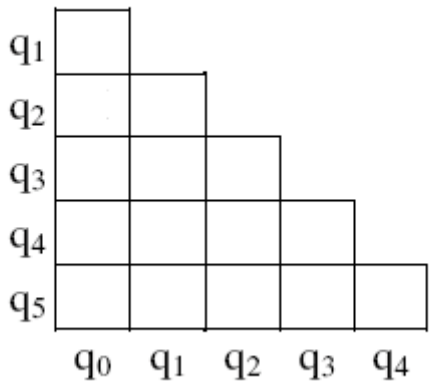
Rjesenje:

Ovaj nacin minimizacije svodi se opet na 2 koraka. Prvi je, iznova, pronalazenje istovjetnih stanja, sto necu ovdje prolaziti, posto je detaljno obradjeno u proslom zadatku. Vec cemo uzeti dohvatljiva stanja i odmah krenuti na 3. algoritam.

	a	b	c	
q_0	q_4	q_1	q_5	0
q_1	q_2	q_3	q_4	0
q_2	q_1	q_3	q_2	1
q_3	q_4	q_1	q_4	0
q_4	q_3	q_1	q_2	1
q_5	q_2	q_4	q_1	1

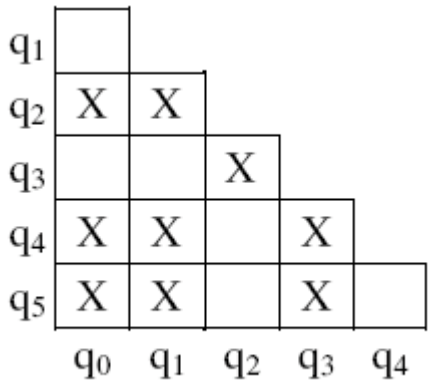
Ostaju nam stanja {q0, q1, q2, q3, q4, q5}.

3. algoritam zasniva se na grafickoj obradi ovog skupa stanja.
Tablica koja se gradi gradi se tako da je na horizontali (x-os) postavljeno (n-1) stupaca (gdje je n broj stanja u skupu :)), te po vertikali (y-os) isto tako.
Takodjer, po horizontali su postavljena stanja od 0 do n-1 (s lijeva na desno), a po vertikali od 1 do n (odozgo prema dolje). Ta tablica ima polja koja sadrze \ dijagonalu :) i sve ispod nje.
Za nas slucaj ta tablica ce izgledati ovako:



Prvo sto treba uciniti je upoznati se s tablicom. Ako gledamo prvo polje (gore lijevo) ono se odnosi na par stanja (q0, q1), te kada ce se obradjivati gledat cemo izlazna stanja za svaki moguci ulazni znak (kao sto se radilo i u 3. zadatku).
Koristeci ta izlazna stanja "putovat" cemo po tablici i pokusati odgonetnuti je li stanje istovjetno.

Da bi si olaksali posao, neka stanja mozemo oznaciti kao sigurno neistovjetna! To su svi parovi stanja koji imaju suprotan znak prihvatljivosti.
U nasem slucaju to su (q0, q2), (q0, q4), (q0, q5), (q1, q2), (q1, q4), (q1, q5), (q3, q2), (q3, q4) i (q3, q5).
Oznacimo to:



Primjetite da su parovi stanja koji nisu medjusobno istovjetni oznaceni s X.

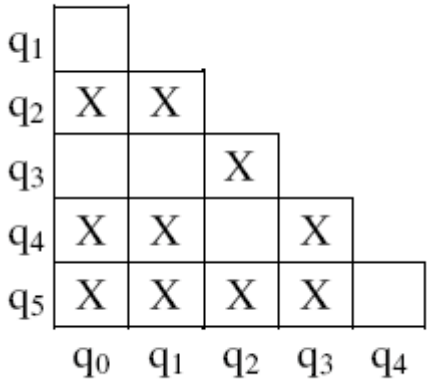
Krenimo s 1. poljem!
To je par stanja (q0, q1)
- q0 za "a" ide u q4
- q1 za "a" ide u q2
To dvoje daje par stanja (q2, q4) koji je u nasoj tablici prazno polje.
Prazno polje znaci da ovdje jos nista nije definirano, tako da cemo morati pricekati da se to stanje rijesi. To cekanje se olaksa tako da se stvori tzv. lista cekanja.
Napise se:
Lista uz (q2, q4) = ((q0, q1))
To znaci da (q0, q1) cekaju (q2,q4)
Idemo dalje:
- q0 za "b" ide u q1
- q1 za "b" ide u q3
Par stanja (q1, q3) takodjer nije definiran pa stvaramo jos jednu listu cekanja, te je sada stanje listi cekanja:
Lista uz (q2, q4) = ((q0, q1))
Lista uz (q1, q3) = ((q0, q1))
- q0 za "c" ide u q5
- q1 za "c" ide u q4
Ni taj par nije definiran, pa se opet stvara nova lista.
Lista uz (q2, q4) = ((q0, q1))
Lista uz (q1, q3) = ((q0, q1))
Lista uz (q4, q5) = ((q0, q1))
Sada smo prosli sva stanja i prvo je polje gotovo. Ono trenutno ceka na tri stanja.

Idemo na polje q0, q3.
Za ulaz "a" oba pokazuju na stanje q4, tako da nam to nista ne govori.
Za ulaz "b" oba pokazuju na q1 tako da tu isto nista ne znamo.
Za ulaz "c" dolazi par (q4, q5), te on ulazi u listu cekanja.
Lista uz (q2, q4) = ((q0, q1))
Lista uz (q1, q3) = ((q0, q1))
Lista uz (q4, q5) = ((q0, q1), (q0, q3))

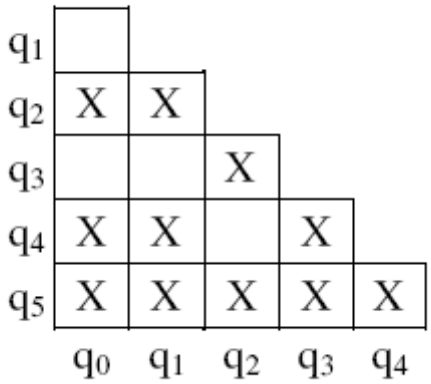
Sljedece polje je q1, q3.
Za ulaz "a" treba gledati polje (q2, q4), medjutim to polje jos ne znamo tako da se (q1, q4) pridodaje listi cekanja uz (q2, q4) i to sad izgleda ovako:
Lista uz (q2, q4) = ((q0, q1), (q1, q3))
Lista uz (q1, q3) = ((q0, q1))
Lista uz (q4, q5) = ((q0, q1), (q0, q3))
Za ulaz "b" polje gleda samo na sebe sto nam nista ne govori. :)
Za ulaz "c" oba gledaju na q4 tako da tu ne mozemo nista zakljuciti.
Lista uz (q2, q4) = ((q0, q1), (q1, q3))
Lista uz (q1, q3) = ((q0, q1))
Lista uz (q4, q5) = ((q0, q1), (q0, q3))

Krecemo na polje q2, q4.
Za ulaz "a" dobijemo par (q1, q3) tako da moramo u listu cekanja.
Lista uz (q2, q4) = ((q0, q1), (q1, q3))
Lista uz (q1, q3) = ((q0, q1), (q2, q4))
Lista uz (q4, q5) = ((q0, q1), (q0, q3))
Za ulaz "b" je ista prica, pa ne treba nista dodavati.
Za ulaz "c" je par stanja jednak tako da nema zakljucka.

Krecemo na polje q2, q5.
Za ulaz "a" imamo (q1, q2) koji nije istovjetan, tako da je ovo stanje neistovjetno i stavljamo X:

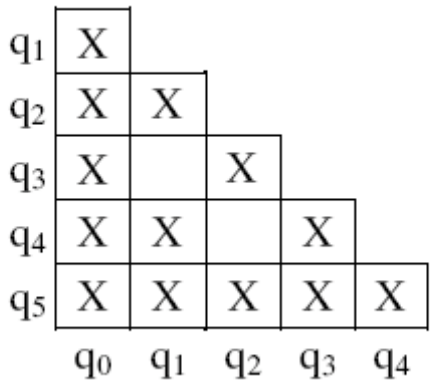


Ostajemo samo polje q4, q5.
Vec za "a" dolazimo do polja (q2, q3) koje nije istovjetno, tako da i (q4, q4) dobije X:



Sada smo prosli sva polja i vrijeme je da razrijesimo one liste. :)
Podsjetimo se:
Lista uz (q2, q4) = ((q0, q1), (q1, q3))
Lista uz (q1, q3) = ((q0, q1), (q2, q4))
Lista uz (q4, q5) = ((q0, q1), (q0, q3))

Jedino smo za q4 i q5 saznali da je neistovjetan. No, to povlaci za sobom i da su svi parovi iz njegove liste cekanja takodjer neistovjetni tako da stavljamo X na (q0, q1) i (q0, q3):



Oznacili smo sve sto smo mogli. Ona polja koja ostanu prazna predstavljaju parove stanja koji su istovjetni.
U nasem slucaju to su q3 i q1, te q4 i q2.
Vrijedi q1 = q3 i q2 = q4, te u tablici mijenjamo sve q3 u q1 i sve q4 u q2:

	a	b	c	
q_0	q_2	q_1	q_5	0
q_1	q_2	q_1	q_2	0
q_2	q_1	q_1	q_2	1
q_1	q_2	q_1	q_2	0
q_2	q_1	q_1	q_2	1
q_5	q_2	q_2	q_1	1

Retci 2 i 4 su jednaki, pa brisemo 4. redak. Retci 3 i 5 su jednaki, pa brisemo i 5. redak.

	a	b	c	
q_0	q_2	q_1	q_5	0
q_1	q_2	q_1	q_2	0
q_2	q_1	q_1	q_2	1
q_5	q_2	q_2	q_1	1

I to bi bilo sve! :)

(nastavak zadatak eventualno sutra, ne vidim vise :))

5. Zadatak (by b0ysee):

Zadani e-NKA pretvoriti u minimalni DKA.

	a	b	c	ε	
q_0	q_1	q_2	q_1	q_1,q_3	0
q_1	q_1,q_2	q_1	q_3	q_2	0
q_2	q_2	q_1	q_3	-	1
q_3	q_2	-	-	-	0

Rjesenje:

Kod pretvorke e-NKA u DKA potrebno je proci nekoliko koraka.
e-NKA prvo treba pretvoriti u NKA, nakon toga NKA u DKA, te nakon toga minimizirati DKA, ukratko:
e-NKA -> NKA -> DKA -> min(DKA)

Ima se posla. :)

e-NKA u NKA
Kod pretvorbe e-NKA u NKA treba konstruirati novu tablicu koja ce prikazit taj nas ciljni NKA.
Sto se tice stanja koje ce nova tablica sadrzavati, ona su ista kao i kod stare tablice, tj. skup q(e-NKA) = skup q(NKA).
Pocetno stanje je jednako u oba automata.
Ulazni znakovi ce takodjer biti jednaki, izuzev epsilona kojeg vadimo (i zato e-NKA postaje NKA :)).
skup ulaznih(e-NKA) = skup ulaznih(NKA) - e.
Sto se tice prihvatljivih stanja (zadnji stupac), ona se odredjuju tako da se uzme skup starih prihvatljivih stanja (sto je kod nas bilo samo stanje q2) i napravi presjek s e-okruzenjem stanja q0.

Ups. :)
Sto je e-okruzenje?
e-okruzenje (epsilon okruzenje) je skup stanja u koja se moze doci samo koristenjem epsilon prijelaza.
Pogledajmo tablicu:

	a	b	c	ε	
q_0	q_1	q_2	q_1	q_1,q_3	0
q_1	q_1,q_2	q_1	q_3	q_2	0
q_2	q_2	q_1	q_3	-	1
q_3	q_2	-	-	-	0

Iz q0 se epsilonom moze doci do q1 i q3, a iz q1 u q2 (treba do kraja dovrsiti lanac).
Kada pogledamo do kuda nas vodi epsilon zapisujemo to na nacin:
e-okruzenje(q0) = {q0, q1, q2, q3}
Primjetitie kako se i stanje samo za sebe (q0) isto zapisuje u e-okruzenje.
e-okruzenje od q1 bilo bi {q1, q2} jer ga epsilon vodi samo do q2.
e-okruzenje(q2) = {q2}, epsilon nikud ne vodi, znaci samo stanje za sebe se zapisuje.
e-okruzenje(q3) = {q3}

Sad kada je to rascisceno, mozemo natrag na zadatak.
Rekao sam kako se gleda ima li zajednickih stanja izmedju skupa trenutno prihvatljivih stanja i e-okruzenja(q0).
skup trenutno prihvatljivih stanja = {q2}
e-okruzenje(q0) = {q0,q1,q2,q3}
IMA! :)
To znaci da cemo prihvatljivim stanjima pridruziti i q0.
To je sve.
Vrijeme je da prikazemo nasu lijepu novu tablicu. :)
Sto sve znamo o njoj?
Skup stanja = {q0, q1, q2, q3}
Skup ulaznih znakova = {a, b, c}
Skup prihvatljivih stanja = {q0, q2}
pocetno stanje = q0

Jos je SAMO potrebno napraviti funkciju prijelaza, tj. vidjeti sto se kako ponasa.
Funkcija prijelaza ima formulu koja se na prvi pogled cini malo zastrasujuce, ali je DOSLOVCE samo ispisivanje na papir:
prijelaz (novog q, za ulazni znak) = e-okruzenje (prijelaz (e-okruzenje(stari q), za ulazni znak))

Na prvi pogled ce ovo izgleda monstruozno, ali kad prvi puta raspisete na papir, postaje kristalno jasno - zato, ponavaljam, raspisite to! :)

Krenimo s q0 i ulazom "a".
- prijelaz (q0,"a") = e-okruzenje (prijelaz (e-okruzenje(q0), "a"))
Prvo uvrstavamo e-okruzenje od q0 koje smo malo vise gore definirali.
- prijelaz (q0,"a") = e-okruzenje (prijelaz ({q0, q1, q2, q3}, "a"))
Sada gledamo dio: prijelaz ({q0, q1, q2, q3}, "a"), tj. stvaramo skup stanja koji ce obuhvacati sve stare prijelaze iz skupa {q0, q1, q2, q3} za ulaz "a".
Za q0 to je q1, za q1 to su q1 i q2, za q2 to je q2 i za q3 je to q2. Kad uvrstavamo, uvrstavamo uniju svih tih "mogucih" prijelaza.
Uvrstimo:
- prijelaz (q0,"a") = e-okruzenje ({q1, q2})
Ostaje jos malo mali dio, a to je gledanje e-okruzenja svih stanja iz skupa koji je ostao.
Kod nas su to q1 i q2 za koje smo to vec odredili vise gore, ali evo opet:
e-okruzenje (q1) = {q1, q2}, e-okruzenje (q2) = {q2}
Opet uvrstavamo uniju svih tih stanja koje smo dobili:
- prijelaz (q0, "a") = {q1, q2}

Eto, to nije bilo tako tesko. :)
Za vjezbu, idemo to jos obaviti za npr. q0 i c
- prijelaz (q0, "c") = e-okruzenje (prijelaz (e-okruzenje(q0), "c"))
- prijelaz (q0, "c") = e-okruzenje (prijelaz ({q0, q1, q2, q3}, "c"))
- prijelaz (q0, "c") = e-okruzenje ({q1,q3})
- prijelaz (q0, "c") = {q1, q2, q3}

Nadam se da je jasno. Ostatak stanja napravite sami, a evo i konacne tablice:

	a	b	c	
q_0	q_1,q_2	q_1,q_2	q_1,q_2,q_3	1
q_1	q_1,q_2	q_1,q_2	q_3	0
q_2	q_2	q_1,q_2	q_3	1
q_3	q_2	-	-	0

I to je nas NKA! Jeeeij! :) Jos _samo_malo_ posla. :)

NKA u DKA

Opet gradimo novu tablicu.
Krenimo s onime ocitim, pocetno stanje q0 ostaje pocetno stanje q0. :)
Sva ostala stanja ne uzimamo iz skupa starog stanja, vec iz NKA tablice i to samo ona koja su u stupcima s ulazima.
Za stupac "a" dobit cemo stanja - [q1,q2] i q2.
Za stupac "b" su stanja [q1, q2] i 0 (da, prazan skup stanja isto treba popisat)
Za stupac "c" su stanja [q1, q2, q3] i q3.
To su nasa stanja u novoj, DKA, tablici. Treba samo primjetiti kako ovdje nigdje nisu navedena stanja q0 i q1 zasebno. Stanja koja nisu navedena izbacujemo, osim ako je jedno od tih stanja pocetno (znaci, q0 ostaje :)).
Lista prihvatljivih stanja se radi tako da se prihvatljivost uzima od starih stanja, a tamo gdje imamo visestruka stanja uzimamo uniju prihvatljivosti. Recimo [q1, q2] sadrzi q1 koji nije prihvatljiv, te q2 koji je prihvatljiv i unija toga je da je to stanje prihvatljivo.
OK, imamo sve sto nam treba da oblikujemo novu tablicu,
pocetno stanje = q0
skup stanja = {q0, q2, q3, [q1,q2], [q1, q2, q3], 0}
skup prihvatljivih stanja = {q0, q2, [q1,q2], [q1,q2,q3]}
skup ulaznih znakova = {a, b, c}

	a	b	c	
$[q_0]$				1
$[q_1,q_2]$				1
$[q_1,q_2,q_3]$				1
$[q_3]$				0
$[q_2]$				1
$[\varnothing]$				0

Sada opet treba ostvariti prijelaze. Ovdje je to dosta jednostavnije nego sto je bilo u proslom korako.
Kako se prolazi kroz stanja i moguce ulazne znakove, tako se gleda u tablici iz proslog koraka unija mogucih prijelaza. Ovo je malo teze oblikovat u rijeci, pa cemo ici primjerom:
prijelaz (q0, "a") = [q1, q2]
Ovo se samo prepisuje, tj. cijeli q0 ce se samo prepisati jer tu i nije doslo do neke promjene.
Krenimo s drugim retkom;
prijelaz ([q1, q2], "a") = prijelaz(q1, "a") + prijelaz(q2, "a")
Evo, to je ono sto sam govorio, gleda se prijelaz svakog stanja iz ovog skupa stanja i onda se napravi unija. Prijelaz iz q1 za "a" sarzi stanja q1 i q2, a prijelaz iz q2 za "a" sadrzu samo q2. Unija toga je q1, q2.
prijelaz ([q1, q2], "a") = [q1,q2] + q2 = [q1, q2]

I sva ostala stanja idu po istom principu.
Konacna tablica:

	a	b	c	
$[q_0]$	$[q_1,q_2]$	$[q_1,q_2]$	$[q_1,q_2,q_3]$	1
$[q_1,q_2]$	$[q_1,q_2]$	$[q_1,q_2]$	$[q_3]$	1
$[q_1,q_2,q_3]$	$[q_1,q_2]$	$[q_1,q_2]$	$[q_3]$	1
$[q_3]$	$[q_2]$	$[\varnothing]$	$[\varnothing]$	0
$[q_2]$	$[q_2]$	$[q_1,q_2]$	$[q_3]$	1
$[\varnothing]$	$[\varnothing]$	$[\varnothing]$	$[\varnothing]$	0

Jedino sto preostaje je minimizirati ovu tablicu, a to se radi identicno kao i u 4. zadatku, tako da stvarno nema potrebe da ponavljam.

Konacno rjesenje:

	a	b	c
$[q_0]$	$[q_2]$	$[q_2]$	$[q_2]$
$[q_2]$	$[q_2]$	$[q_2]$	$[q_3]$
$[q_3]$	$[q_2]$	$[\varnothing]$	$[\varnothing]$
$[\varnothing]$	$[\varnothing]$	$[\varnothing]$	$[\varnothing]$

6. Zadatak (by b0ysee):

Konstruirati DKA koji prihvaca jezik L=L1 unija L2 (L1 presjek L2, L1-L2, L2-L1). Jezik L1 sastoji se od nizova opisanih regularnim izrazom r1=a*b*c*, a nizovi iz jezika L2 opisani su regularnim izrazom r2=a*(b+c)*.

Rjesenje:

Ideja ovog zadatka je da se prvo ostvare dva automata koji prihvacaju jezik L1 i jezik L2. Nakon toga ih se spoji u uniju, presjek ili oduzimanja (makar cemo mi to sve prikazati na jednoj tablici jer je jedina razlika u prihvatljivosti stanja).

Prije nego sto pocnemo rjesavati zadatak, volio bih se osvrnuti na regularne izraze, posto je ostalo vise manje djecja igra ako znate procitati izraz. :)

Analizirat cemo nekoliko izraza da bi dobili ideju o tome:

- r = a
Ovdje imamo regularni izraz koji ce prihvacati samo jedan ulaz "a". To bi znacilo da bi takav automat imao 3 stanja. Pocetno, neprihvatljivo, stanje iz kojeg bi za pojavljivanje znaka a skocio u drugo stanje, koje je prihvatljivo. Ako slucajno stigne jos jedan znak "a" otici ce u trece stanje koje je takodjer neprihvatljivo i ostao tamo zauvijek, bez obzira koliko a jos stiglo.
Skica:

```
       a           a
(q0) ---> ((q1)) ---> (q2)
```

- r = a*
Ovdje imamo izraz koji koristi tzv. Kleeneov operator. Taj operator kaze da ce prihvatiti bilo koji iznos znakova "a" i dalje ostati prihvatljiv.
To bi znacilo da je jezik koji ovaj izraz prihvace - L(r) = {0, a, aa, aaa, aaaa, ...}

- r = a+b
Kada dodje plus znak u igru, to znaci da se moze prihvatiti samo jedan znak, tj. ovaj konkretno automat ce prihvatiti ili "a" ili "b" kao ulaz, a sve ostalo ce biti neprihvatljivo (npr. aa je neprihvatljivo jer ima 2 znaka u nizu, ab takodjer. itd.)
Jezik koji prihvaca - L(r) = {a, b}, tj. prima samo jednoclane nizove koji sadrze ili a ili b.

- r = ab
Ovo se cita kao "a puta b", da ne bi slucajno zamijenili s Kleeneovim operatorom. ab nam kaze kako ce prihvacati **dvoclane** nizove. To bi znacilo da puta povecava broj znakova koji se prihvacaju, tocno na onaj broj clanova koliko je znakova u mnozenju i to tocno u tom redoslijedu. Recimo, abab ce primati samo ulaz "abab".
Ovdje je L(r) = {ab}

Ovo su osnove, sad se tu mogu raditi svakakvi lijepi izrazi iz toga. :)

- r = a*b
Isto kao i ab, primati ce ulaz "ab", ALI, posto se ovdje nalazi Kleeneov operator, on ce dopustiti znaku a da se proteze u rasponu od nijednog pojavljivanja do beskonacno pojavljivanja.
To bi znacilo da je L(r) = {b, ab, aab, aaab, aaaab, aaaab, ...}

- r = a*+b
Kleeneov operator ima prednost, tako da ce ovdje primati dvije mogucnosti. Ili znak a ili znak b, to je kao u gornjem primjeru a+b, medjutim, Kleeneov operator ce dopustiti znaku a ipak malo slobode, pa ce se a smjeti pojaviti u bilo kojoj svojoj formi.
L(r) = {b, a, aa, aaa, aaaa, aaaaa, ...}

- r = (a+b)(a+b)
Opa, plus i puta u istom izrazu. :) Nije to nista tesko. Ovdje to konkretno znaci da cemo imati sigurno dvoclani izraz koji na prvom mjestu moze imati ili a ili b, te na drugom mjestu isto to (Kartezijev produkt :)).
L(r) = {aa, ab, ba, bb}

- r = (a+b)*
Ovaj je malo slozeniji zato sto daje podosta mogucnosti.
Prvo, izraz u zagradi nam kaze da se tu moze stvoriti ili a ili b. Kleeneov operator kaze da se mozemo stvarati jos znakova koliko zelimo.
Primjeri: 0 - zato sto ne mora biti nijedan znak (zbog Kleeneovog operatora), a, b, aa, bb, ab, aab, aba, itd.
L(r) = {0, a, b, aa, bb, ab, ba, aaa, aab, aba, baa, bab, ...}

- r = (a+b)*aa(a+b)*
Prvo cemo vidjeti sto se nalazi u zagradama, jer one imaju najvisu prednost. Prva zagrada je ista kao i kod proslog primjera. To znaci da ce na prvi mjesto naseg ulaznog niza doci proizvoljan broj i oblik ulaza a i b.
Druga zagrada kaze isto to, ali to kaze tako da ce se taj proizvoljan broj znakova naci na kraju naseg ulaznog niza.
Sredina (aa) kaze da izmedju ova dva proizvoljna broja ulaznih podataka moraju biti dvije uzastopne nule (kao s primjerom za ab).
L (r) = {aa, aaa, aaaa, baa, abaa, abaaba, bbbbbaabbbbbbb, ...}

Nadam se da ste stekli kakvo takvo shvacanje ovoga, idemo pokusati desifrirati ove nase zadane izraze.

L1

Kaze ovako:
r1 = a*b*c*
Ovo je pljuga s obzirom na kakve smo gluposti naletavali tu gore medju primjerima. :)
a* kaze da je prvi clan prihvatljivog niza proizvoljan broj ponavljanja znaka a, b* kaze to isto za srednji clan, a c* nam to isto govori za zadnji clan.
L(r1) = {0, a, ab, ac, bc, abc, aabc, aabbc, aaabbccc, aaaaabbbbbccccc, ... }
Ako ovako gledamo, nas automat je dosta lako za ostvariti.
Ovdje bih htio napraviti malo drukciju konstrukciju automata, tj. preko tablice, jer smo to crtezom vec radili.
Sve sto za sada znamo su nasa ulazna stanja, pa cemo samo to i napraviti:

	a	b	c	PRIH

Kada radite u tablici, i dalje ostaje princip onoga da je najlakse prvo napraviti povoljan pristup, pa tek onda ici gledati sva ostala stanja.
Moramo imati pocetno stanje q0. To pocetno stanje q0 ce ostati samo u sebi za "a" (posto prvi clan niza moze biti proizvoljan broj ulaznih znakova "a". Ulazni znakovi "b" ili "c" su takodjer prihvatljivi posto Kleeneov operator dopusta da bude 0 pojavljivanja znaka pod njime. Tako ce za "b" i "c" q0 prijeci u nova, takodjer prihvatljiva stanja.

	a	b	c	PRIH
q0	q0	q1	q2	

Stanje q1 je stanje ako je primljen znak "b" sto bi znacilo da za bilo koji iznos znakova "b" automat ostaje u tom stanju. Ako slucajno dodje znak "c" prijeci cemo u stanje q2. q2 predstavlja stanje kad smo vec na znaku "c" i koliko god novih "c"-ova doslo, ostajemo u stanju q2:

	a	b	c	PRIH
q0	q0	q1	q2	1
q1		q1	q2	1
q2			q2	1

Odlicno, imamo samo tri nepopunjena stanja!
ako u stanju q1 dodje znak "a", to je lose zato sto smo dio koji dopusta znakove "a" vec prosli, tako da je to neprihvatljivo. Napraviti cemo jos jedno stanje koje ce se brinuti o tim slucajevima.
Ista stvar vrijedi i za q2 ako slucajno dodju "a" ili "b". To dvoje ce takodjer ici u to neprihvatljivo stanje.
Da sada ne crtam opet dva puta sliku, analizirajmo odmah i to novo stanje q3. q3 je stanje u kojem je doslo do greske, tj. koje kaze da je niz falican, da ne vrijedi. Iz njega se ne smije nikako izaci, pa ce on za sve ulaze vracati sam u sebe. I to je sve, nas DKA(L1) je gotov! :)

	a	b	c	PRIH
q0	q0	q1	q2	1
q1	q3	q1	q2	1
q2	q3	q3	q2	1
q3	q3	q3	q3	0

L2
Krenimo na 2. izraz.
r2 = a*(b+c)*
Ovaj je laksi od proslog. ;)
Prvo znak mora biti proizvoljan broj ponavljanja "a", a drugi proizvoljan znak pojavljanja ili b ili c.
L(r2) = {0, a, b, c, bc, ab, ac, abc, aaab, aaac, aaabcbcbbb, abcbbbcccb, ...}

Cak je i tablica laksa za ovaj slucaj, krenimo opet od osnovne forme, mogucih ulaza:

	a	b	c	PRIH

Pocetno stanje q0 ce opet ostajati samo u sebi za proizvoljan unos znakova "a", medjutim pojavljvivanje znakova "b" ili "c" bacati ce u isto novo stanje posto je svejedno dolazi li b ili c.

	a	b	c	PRIH
q0	q0	q1	q1	1

Ako dodjemo u stanje q1, bilo koji niz "b" ili "c" znakova je prihvatljiv pa ostajemo u tom stanju. Ako slucajno naleti znak "a" to je greska i treba otici u novo stanje iz kojeg se ne mozemo izvuci iz istog razloga kao i u proslom zadatku.

	a	b	c	PRIH
q0	q0	q1	q1	1
q1	q2	q1	q1	1
q2	q2	q2	q2	0

L1 operator L2
Odlicno, imamo izgradjene nase DKA automate, sada bi trebalo napraviti tablicu nad kojom cemo gledati sto operatori rade. Tablica za L1 unija L2, L1 presjek L2, L1 - L2, L2 - L1 je identicna osim u polju PRIH jer se mijenja prihvatljivost pojedinih stanja.
Ulazi u novu tablicu ce biti jednaki kao i dosadasnji (a, b i c).
Pitanje se stvara kada treba stvoriti nova stanja.
Stanja se stvaraju lagano - Kartezijevim produktom, kombinirate sva stanja iz DKA(L1), sa svim stanjima iz DKA(L2). Rezultate (funkcije prijelaza) isto kombinirate iz obje tablice.
Medjutim, ima jedan fini trik kako ne bismo morali bas sve prolaziti (to je zapravo izbjegavanje nedohvatljivih stanja).
Krenimo od prvog stanja, koje se mora tako i tako uzeti, [q0, q0].
- [q0, q0] za "a" daje [q0, q0], za "b" daje [q1, q1], za "c" daje [q2, q1]
Ovdje smo dobili prvi redak nase nove tablice, te saznali za dva nova dohvatljiva stanja: [q1,q1] i [q2, q1]

	a	b	c
q0,q0	q0,q0	q1,q1	q2,q1

- [q1, q1] ide za "a" u [q3,q2], za "b" u [q1, q1], za "c" u [q2,q1]
[q3, q2] je novo stanje koje do sada nismo imali, tako da ce se uvrstiti. [q1, q1] i [q2, q1] vec postoje.

	a	b	c
q0,q0	q0,q0	q1,q1	q2,q1
q1,q1	q3,q2	q1,q1	q2,q1

I sada na isti nacin nastavljamo tablicu dok ne dodjemo do konacnog oblika:

	a	b	c
q0,q0	q0,q0	q1,q1	q2,q1
q1,q1	q3,q2	q1,q1	q2,q1
q2,q1	q3,q2	q3,q1	q2,q1
q3,q2	q3,q2	q3,q2	q3,q2
q3,q1	q3,q2	q3,q1	q3,q1

Odlicno, nasa tablica je gotova. Ostaje samo odrediti prihvatljiva stanja za svaki od operatora.
To je najlaksi dio posla jer doslovce imate matematiku iz 1. razreda osnovne. :)

Uzmu se tablice DKA(L1) i DKA(L2) i gleda po njihovim stanjima je li nesto prihvatljivo (bitni su prvi i zadnji stupac, pa ih evo ovdje):

DKA(L1)

	PRIH
q0	1
q1	1
q2	1
q3	0

DKA(L2)

	PRIH
q0	1
q1	1
q2	0

Uzmimo prvo uniju.

L1 unija L2
Idemo po stanjima i za par stanja (ovaj najljeviji stupac) gledamo tablice za DKA(L1) i DKA(L2), te obavljamo operaciju.
Ako je bar jedno od stanja iz para koji gledamo jednako 1, tada je to prihvatljivo stanje.
Krenimo od [q0,q0] - q0 iz DKA(L1) je prihvatljivo, q0 iz DKA(L2) takodjer. Znaci prihvatljivo.
[q1, q1] - 1 unija 1 = 1
[q2, q1] - 1 unija 1 = 1
[q3, q2] - 0 unija 0 = 0
[q3, q1] - 0 unija 1 = 1
To je sve, samo treba dodati stupac PRIH i napisati tablicu do kraja:

	a	b	c	PRIH
q0,q0	q0,q0	q1,q1	q2,q1	1
q1,q1	q3,q2	q1,q1	q2,q1	1
q2,q1	q3,q2	q3,q1	q2,q1	1
q3,q2	q3,q2	q3,q2	q3,q2	0
q3,q1	q3,q2	q3,q1	q3,q1	1

L1 presjek L2
Isto kao i L1 unija L2. Dat cu samo konacni rezultat jer je postupak isti, samo sto se radi presjek medju rezultatima.

	a	b	c	PRIH
q0,q0	q0,q0	q1,q1	q2,q1	1
q1,q1	q3,q2	q1,q1	q2,q1	1
q2,q1	q3,q2	q3,q1	q2,q1	1
q3,q2	q3,q2	q3,q2	q3,q2	0
q3,q1	q3,q2	q3,q1	q3,q1	0

L1 - L2
Najobicnije oduzimanje. Malo podsjetnik na prvi razred: 1-0 = 1, 1-1 = 0, 0-1 = 0 (nema neg. vrijednosti), 0-0 = 0 :) :) :)

	a	b	c	PRIH
q0,q0	q0,q0	q1,q1	q2,q1	0
q1,q1	q3,q2	q1,q1	q2,q1	0
q2,q1	q3,q2	q3,q1	q2,q1	0
q3,q2	q3,q2	q3,q2	q3,q2	0
q3,q1	q3,q2	q3,q1	q3,q1	0

L2 - L1
No, da... :)

	a	b	c	PRIH
q0,q0	q0,q0	q1,q1	q2,q1	0
q1,q1	q3,q2	q1,q1	q2,q1	0
q2,q1	q3,q2	q3,q1	q2,q1	0
q3,q2	q3,q2	q3,q2	q3,q2	0
q3,q1	q3,q2	q3,q1	q3,q1	1

Dosta za sada, idem ViS (bljak) ucit...
Zivili! :)

8. Zadatak (by b0ysee):

Sto se osmog zadatka tice, on se sastoji samo od toga da se primjenjuju svojstva na jednadzbu koja tako i tako morate nauciti s time da je u onom PDF-u vec sve korak po korak tako da ne vidim tu jos neke dodatne mogucnosti razlaganja.

7. Zadatak (by b0ysee):

Regularnim izrazom opisati jezik koji sadrži sve nizove nad abecedom {0, 1, 2} u kojima nema uzastopnih ponavljanja znaka 0.

Rjesenje:
Kod ovog zadatka koristit cemo nase novosteceno znanje u vezi regularnih izraza. :)
Pogledajmo sto zadatak kaze. Moguci ulazni znakovi su 0, 1 i 2, sto znaci da ce samo oni i biti sadrzani u regularnom izrazu.
Zadatak kaze kako se ne smije dogoditi da se znak 0 ponavlja (makar je npr. u redu da svaki drugi znak bude 0).
Najlakse je zapoceti s onime sto nas ne brine, a to je oblikovanje izraza koji ce reci da moze doci proizvoljan broj znakova 1 i 2:

```
r = (1+2)*
```

Ovaj gornji izraz dati ce L(r) = {1, 2, 11, 22, 12, 21, 1112, 2112, 211221, ...} sto je za sada u redu.
Nadalje, moramo omoguciti da bude znakova 0. Ne mozemo samo napisati 0+1+2 posto bi onda i 0 mogla doci proizvoljan broj puta, ali se moze napisati 01 koji ce davati tocno 01 u skup mogucih nizova:

```
r = (1+2+01)*
```

L(r) = {1, 2, 01, 012, 0111, 1012, 111201, 1212101, ...}
Ovo takodjer radi odlicno medjutim, imamo nedostatak, a to je da se ne moze konstruirati primjer u kojem ce iza 0 slijediti 2. To se rjesava dodavanje znaka 02 u izraz i to:

```
r = (1+2+01+02)*
```

L(r) = {1, 2, 01, 02, 101, 102, 111101010101020202012221, ...}
Ovo izgleda vrlo dobro, ali... javlja se problem zadnjeg znaka. Sto ako zelimo konstruirati neki vrlo jednostavan niz, npr. 10?
Moramo nekako natjerati da se iza ovog, vec ostvarenog niza, moze pojaviti 0. Ako zelimo prikazati samo jednu nulu, napravit cemo izraz:

```
r = 0
```

i on ce UVIJEK sadrzavati 0. Ali, treba li nam 0 uvijek?
0 ce ponekad biti visak, pa treba omoguciti da je ona proizvoljna, a to se radi s epsilon koji prikazuje prazan znak:

```
r = 0 + epsilon
```

Sada mozemo spojiti ta dva izraza s time da zelimo ovaj izraz za 0 ili prazan znak obavezno na kraju niza. To se radi tako da se dosadasnji izraz "pomnozi" s ovim novim:

```
r = (1+2+01+02)*(0+epsilon)
```

Ostaje pogledati moze li se ovo napraviti kako alternativno. Moze, ako ove znakove 01 i 02, pretvorimo u 10 i 20, niz ce i dalje biti dobar samo sto se onda (0+epsilon) mora premjestiti na pocetak iz 2 razloga - 1. zato da bi se izbjegle 2 nule za redom, te da bi se omogucila 0 na pocetku niza:

```
r = (0+epsilon)(1+2+10+20)*
```

10. Zadatak (by jakovd):

Iz zadanog Mooreovog automata konstruirati Mealyev automat.

	0,5	1,6	2,7	3	4	λ
q0	q0	q1	q2	q3	q4	0
q1	q3	q4	q0	q1	q2	1
q2	q1	q2	q3	q4	q0	2
q3	q4	q0	q1	q2	q3	3
q4	q2	q3	q4	q0	q1	4

Rješenje:
Za početak, kratko objašnjenje Mooreovog i Mealyevog automata.
U Mooreovom automatu izlaz ovisi samo o trenutnom stanju automata.
Ulazni niz se koristi samo da bi se prešlo iz jednog stanja u drugo.
Za razliku od toga, Mealyev automat daje izlaz ovisno o tome koji je znak došao i u kojem stanju se nalazio automat u trenutku nailaska ulaznog znaka.

Izlazna funkcija Mooreovog i Mealyevog automata povezane su izrazom:
λ'(q,a)=λ(**δ(q,a)**)
Da bi dobili lijevu stranu ovog izraza (a to je izlazna funkcija Mealyevog automata) treba prvo riješiti ovaj dio desnog izraza:
δ(q,a)
To znači da za svako stanje **q** i svaki znak **a** treba odrediti u koje stanje se prelazi (**δ**).
To je jako jednostavno jer Mealyev automat ima jednake prijelaze kao i Mooreov pa treba samo prepisat taj dio iz tablice za Mooreov automat.

Dakle tablica prijelaza **δ'** za Mealyev automat izgleda ovako:

δ'	0,5	1,6	2,7	3	4
q0	q0	q1	q2	q3	q4
q1	q3	q4	q0	q1	q2
q2	q1	q2	q3	q4	q0
q3	q4	q0	q1	q2	q3
q4	q2	q3	q4	q0	q1

Sada prelazimo na dio **λ(**δ(q,a)**)** a to znači da za ova stanja koja smo našli treba odrediti kakvi su izlazi.
Za to konstruiramo drugu tablicu u kojoj stanja prijelaza zamijenjujemo sa odgovarajućim izlaznim stanjima
(to vidimo iz prvog i poslijednjeg stupca u tablici Mooreovog automata).

Tablica izlaza **λ'** za Mealyev automat:

λ'	**0,5**	**1,6**	**2,7**	**3**	**4**
q0	0	1	2	3	4
q1	3	4	0	1	2
q2	1	2	3	4	0
q3	4	0	1	2	3
q4	2	3	4	0	1

To je to. Ove dvije tablice su riješenje zadatka

11. Zadatak (by jakovd):

Iz zadanog Mealyevog automata konstruirati Mooreov automat.

δ	0	1
q_0	q_0	q_3
q_1	q_1	q_3
q_2	q_2	q_1
q_3	q_2	q_0

λ	0	1
q_0	0	0
q_1	0	1
q_2	1	1
q_3	1	0

Za svako stanje Mealyevog automata (Q-skup svih stanja) i izlazni znak (Δ-skup svih izlaznih znakova), Mooreov automat bi trebao imati po jedno stanje (kažem trebao, jer se minimizacijom dosta tih stanja može izbacit).
Q'=Q x Δ
Slijedećim postupkom izbjegavamo nepotrebno raspisivanje funkcije prijelaza za svaku od tih kombinacija i radimo samo one prijelaze do kojih se može doći iz početnog stanja.
Stanja u Mooreovom automatu se označavaju sa [q, b] a to znači da je automat u stanju koje je u slično Mealyevom stanju q s prijelazom b.
Počinjemo od početnog stanja $[q_0, 0]$. Ta nula i nije tolko bitna jer nam je to početno stanje a u njega nismo došli nikakvim prijelazom pa tu može slobodno pisat naprimjer 1 (glavno da je znak iz skupa izlaznih znakova Δ).

δ'	0	1
$[q_0, 0]$		

Onda za svaki ulazni znak gledamo u koje stanje prelazi Mealyev automat δ(q,a) i kakav je izlaz za taj prijelaz λ(q,a).

δ'	0	1
$[q_0, 0]$	$[q_0, 0]$	$[q_3, 0]$

Kad napišemo prvi red za q_0 pogledamo koja smo nova stanja dobili. Za ulazni znak 0 smo otišli u $[q_0, 0]$ (odnosno ostali i dalje u početnom stanju), a za ulazni znak 1 smo otišli u **$[q_3, 0]$**. To nam je novo stanje i sad isti taj postupak moramo napravit za to stanje.

δ'	0	1
$[q_0, 0]$	$[q_0, 0]$	$[q_3, 0]$
$[q_3, 0]$		

Kad god otiđemo u neko stanje kojeg do tad nismo susreli, upisujemo ga u tablicu i kasnije obrađujemo. I tako radimo dok ne dobijemo da jedno stanje prelazi u stanja koja smo već obradili.
Ostaje još samo napisat koji je izlaz daje taj Mooreov automat u pojedinim stanjima, a to je jednostavno ako znamo da mora davat iste izlaze kao i Mealyev automat pri prijelazima u to stanje pa samo treba prepisat ove brojeve uz nazive stanja.
Konačna tablica izgleda ovako:

δ'	0	1	λ'
$[q_0, 0]$	$[q_0, 0]$	$[q_3, 0]$	0
$[q_3, 0]$	$[q_2, 1]$	$[q_0, 0]$	0
$[q_2, 1]$	$[q_2, 1]$	$[q_1, 1]$	1
$[q_1, 1]$	$[q_1, 0]$	$[q_3, 1]$	1
$[q_1, 0]$	$[q_1, 0]$	$[q_3, 1]$	0
$[q_3, 1]$	$[q_2, 1]$	$[q_0, 0]$	1

12. Zadatak (by Shakan):

Konstruirati gramatiku nad abecedom {0, 1, 2} koja generira nizove u kojima nema uzastopnog ponavljanja podniza 01.

Rješenje:

Gramatika i DKA imaju određenu analogiju: Ako zamislimo da su nezavršni znakovi stanja (dakle, **S**, **A**, **B**, **C**, ... , gdje je **S** "Početno stanje"), produkcije prijelazi, a završni znakovi ulazni znakovi, priča postaje jednostavna. Da pojasnim:

S -> 0A bi se kod DKA moglo napisati ovako: **delta(S, 0) = A**

Dakle, mi bi mogli zamisliti DKA koji će prihvaćati upravo nizove definirane ovom gramatikom. Ja ću i u tome duhu objašnjavati pojedine produkcije, u kojima će generirani završni znakovi biti kao ulazni znakovi, koji nam dolaze jedan po jedan. Krenimo redom:

1. Za početni nezavršni znak **S** mi smijemo definirati produkcije koje će dodavati svaki završni znak (**0**, **1**, i **2**) zato što niz

smije početi bilo kojim znakom iz abecede (a **S** je početni nezavršni znak). Za svaku ćemo produkciju uzeti novi nezavršni znak

(redom **A**, **B**, **C**, **D**... kako nam budu dolazili). Dakle, pišemo:

S -> 0A

Kako je 0 početak podniza "**01**", za kojeg znamo da se ne smije ponavljati uzastopno, gramatika ga mora "prepoznati", pa

produkcija mora na kraj niza staviti novi nezavršni znak (prvi slijedeći slobodni nam je **A**) (ovo je analogno prijelazu u novo

stanje uzrokovano ulaznim znakom **0** kod DKA).

S -> 1S
S -> 2S

Ove dvije produkcije generiraju nezavršni znak **S**, kako su svi podnizovi znakova dozvoljeni osim "**0101**", nema potrebe za

generiranjem novog nezavršnog znaka (ovo je analogno zadržavanjem u istom stanju).

S -> €

Ova produkcija nam govori da niz na čijem je kraju nezavršni znak **S** odgovara jeziku koji je zadan u zadatku; ako nema više

znakova u nizu, niz će se prihvatiti (time definiramo **S** kao dozvoljeno stanje u DKA).

Time smo dovršili popis produkcija za nezavršni znak **S**, koji glasi:

S -> 0A|S1|S2|€

2. Prelazimo na idući nezavršni znak kojeg smo definirali, **A**:

A -> 0A

Ovo možemo shvatiti ovako: ako nam se nakon **0** u nizu nalazi još jedna **0**, to znači dvije stvari:
- prošla nula NIJE dio podniza "**01**"
- nova bi nula MOGLA BITI dio podniza "**01**"
Iz tog razloga produkcija dodaje na kraj nezavršni znak **A**.

A -> 1B

Ako nam je nakon nule izgenerirana **1**, time je stvoren podniz "**01**", kojega također trebamo "prepoznati", pa ova produkcija generira novi završni znak **B**.

A -> 2S

Ako definiramo generiranje znaka **2**, podniz "**01**" njime nije realiziran, te možemo uz njega generirati nezavršni znak **S** (koji nas vraća na početak priče).

A -> €

Za ovu produkciju vrijedi isto što i za produkciju **S -> €**.

Popis produkcija za nezavršni znak **A** glasi:

A -> 0A|1B|2S|€

3. Prelazimo na idući nezavršni znak, **B**:

B -> 0C

Generiranje znaka **0** otvara mogućnost generiranja drugog podniza "**01**" (ako znak **B** vrši produkciju, to znači da je neposredno prije toga generiran podniz "**01**"). To također moramo prepoznati, stoga generiramo novi nezavršni znak **C**.

B -> 1S
B -> 2S

Ukoliko generiramo znakove **1** ili **2** znamo da smo prekinuli niz "**0101**" stoga generiramo i nezavršni znak **S**.

B -> €

Za ovu produkciju vrijedi isto što i za produkciju **S -> €**.

Popis produkcija za nezavršni znak **B** glasi:

B -> 0C|1S|2S|€

4. Prelazimo na idući nezavršni znak, **C**:

C -> 0A

Generiranjem znaka **0** se prekida generiranje podniza "**0101**"; kako je **0**, međutim, početak istog podniza, ova produkcija mora generirati i nezavršni znak **A**.

C -> 1X ----> NEMA

Ako bi se uvrstila ova produkcija tada gramatika ne bi generirala niz koji pripada zadanom jeziku!

C -> 2S
C -> €

Ovo mislim da više nije potrebno komentirati.

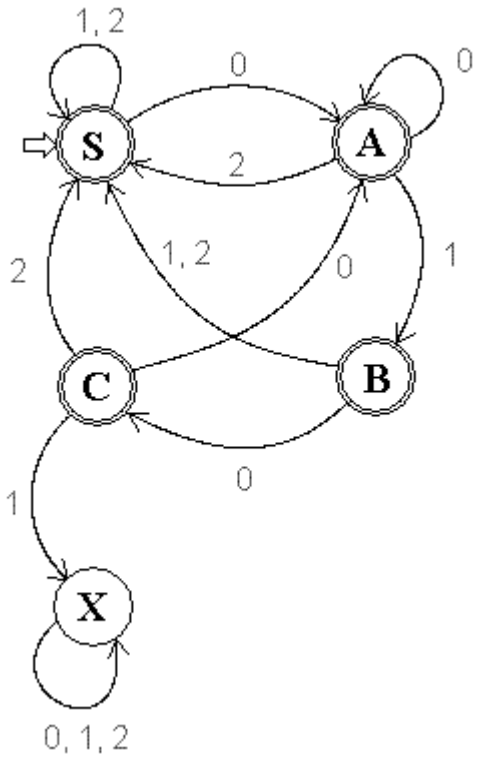
Popis produkcija za nezavršni znak **C** glasi:

C -> 0A|2S|€

5. Kako u prošlom koraku nismo definirali niti jedan novi završni znak, kažemo da smo završili sve produkcije koje definiraju gramatiku koju smo trebali osmisliti:

S -> 0A|S1|S2|€
A -> 0A|1B|2S|€
B -> 0C|1S|2S|€
C -> 0A|2S|€

Još jedno malo pojašnjenje:
Možda se nakon ovoga svega pitate: "Kako to da su ovoj gramatici ekvivalentnom DKA sva stanja prihvatljiva?". Odgovor je: nisu. Da bi konstruirali DKA koji prihvaća nizove stvorene ovom gramatikom, morali bi definirati i prijelaz u novo stanje kojim bi registrirali niz koji nije prihvatljiv. Evo primjera takvog automata:



Nadam se da ste shvatili opći princip... jer ja bogami ne znam kako bih bolje to ispričao... svaki prijedlog šaljite na PM

13. Zadatak (by Shakan):

Na osnovu zadanog DKA konstruirati konteksno neovisnu gramatiku.

	a	b	c	
q_0	q_0	q_1	q_2	1
q_1	q_2	q_0	q_1	0
q_2	q_1	q_2	q_0	0

Rješenje:

Bez da puno filozofiram zašto i kako (zato ovo i čitate, pretpostavljam), evo način rješavanja, kratko i (nadam se) jasno (mali note, znak "=>" znači "odgovara")..

Gramatika i DKA imaju analogiju (za jezik kojeg stvorimo regularnom gramatikom možemo napraviti DKA koji prihvaća nizove tog jezika):
- nezavršni znakovi odgovaraju stanjima DKA. Ako je **q0** početno stanje, u produkciji ga pišemo kao **S** ("start"); sva ostala stanja prevodimo redom u velika slova abecede:

q0 => S, **q1 => A**, **q2 => B**, općenito **Q => V**

- produkcije odgovaraju prijelazima

delta(q0, 0) = q1 => S -> aA

- završni znakovi odgovaraju ulaznim znakovima (npr. **0 => a**, **1 => b**, **2 => c** itd., općenito **T => sigma**).

Dakle, mi za svaki prijelaz unutar DKA možemo pisati analognu produkciju za gramatiku

Vrlo lagan način rješavanja ovog zadatka je slijedeći:

1) U tablici prijelaza zamijenimo nazive stanja nezavršnim znakovima prema uputama gore
2) sve ulazne znakove zamijenimo završnim znakovima tj. malim slovim abecede (osim ako već jesu, kao u ovom slučaju, mala slova abecede)

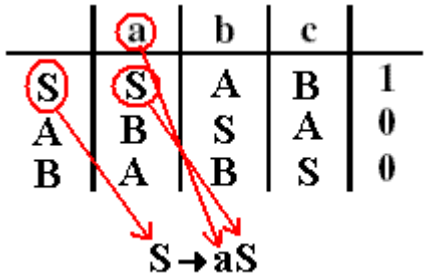
Nakon ovih promjena, tablica izgleda ovako:

	a	b	c	
S	S	A	B	1
A	B	S	A	0
B	A	B	S	0

Možemo početi rješavati:

3) čitamo prvi red zapisujući prijelaze kao produkcije koje odvajamo znakom **|**.
3.1) Čitamo prvo polje prvog retka. Ono nam kaže da znak **S** (što očitavamo s početka reda) daje produkciju **aS** (a s vrha stupca, **S** iz polja), dakle možemo pisati:

S -> aS



3.2) Prelazimo na sljedeće polje retka, iz kojeg čitamo novu produkciju znaka S, produkciju bA, koju dodajemo na kraj izraza za produkciju:

S -> aS|bA

3.3) Isto uradimo i za iduće polje

S -> aS|bA|cB

3.4) Nakon što smo dovršili sva polja u retku, ostaje nam još jedna stvar. Ako je neko stanje prihvatljivo (u zadnjem stupcu je 1), dodajemo još jednu produkciju, sa završnim znakom **€** (epsilon), na kraj liste produkcija:

S -> aS|bA|cB|€

ponavljamo korak 3) za svaki red do kraja tablice, te konačno dobijemo:

S -> aS|bA|cB|€
A -> aB|bS|cA
B -> aA|bB|cS

Eto, to je to! Ne zaboravite napisati one standardne stvari tipa **G = (V, T, P, S)** (ono što su oni napisali u svome rješenju, dakle).

14. Zadatak (by Shakan):

Iz zadane lijevo-linearne gramatike konstruirati NKA.

Kod:

```
S -> Ac         A -> Bb         B -> A
S -> Aab        A -> cab        B -> ca
S -> Ba         A -> Sb         B -> Aaba
```

Rješenje:

Ovdje je cijeli postupak manje-više očit... ali ne posve. Ipak ću ići korak po korak, objašnjavajući svaki putem.

Dakle, zadali su nam gramatiku, i očekuju od nas da im napravimo NKA koji će prihvaćati nizove proizvedene tom gramatikom. Ako ste pogledali malo rješenje koje su oni dali, vidjeli ste da ima dosta posla, čak i crtanja NKA! Problem je u tome što su nam zadali **lijevo-linearnu gramatiku**, koja konstruira nizove **od zadnjeg znaka prema prvom**, a svi NKA prihvaćaju nizove **od prvog prema zadnjem znaku**. To čini konstrukciju tog automata malo složenijom.

Rješenje? Konstruirati desno-linearnu gramatiku koja će proizvoditi obrnute nizove, konstruirati odgovarajući NKA, te onda obrnuti taj automat (više o tome poslije). Krećemo redom:

1. Obrnuti gramatiku. Jednostavno obrnemo rezultate (desne strane) svih produkcija, na sljedeći način:

S -> Ac => S -> cA
S -> Aab => S -> baA

itd. sve do zadnje produkcije, čime dobivamo novu gramatiku, G':

Kod:

```
S -> cA         A -> bB         B -> A
S -> baA        A -> bac        B -> ac
S -> aB         A -> bS         B -> abaA
```

2. Prilagoditi produkcije za stvaranje NKA. Znamo da NKA može primati samo jedan ulazni znak istovremeno. Kako sam spomenuo kod 13. zadatka, prijelaz je analogan produkciji. Kako u prijelazu sudjeluje **TOČNO JEDAN** ulazni znak, moramo preoblikovati produkcije gramatike tako da svaka u niz dodaje **TOČNO JEDAN** završni znak. To znači da ćemo stvoriti i neke nove produkcije, kao i nove nezavršne znakove.

2.1. Produkcije koje već odgovaraju ovom kriteriju preskačemo, a to su:

S -> cA
S -> aB
A -> bB
A -> bS

2.2. Ostale produkcije obrađujemo redom (obrađene se produkcije neće više rabiti, te se nakon obrade brišu iz tablice):

2.2.1. Prve na red dolaze produkcije čije desne strane ne završavaju nezavršnim znakom:

A -> bac
B -> ac

Njih sređujemo tako da im na kraj desne strane dodamo nezavršni znak, **[€]**. To je znak koji će imati samo jednu produkciju koja će umjesto njih završavati niz, **[€] -> €** (kasnije će biti jasnije zašto, za sada objašnjavam algoritam). Dakle, umjesto ovih produkcija, pišemo nove:

A -> bac[€]
B -> ac[€]

2.2.2. Nakon njih slijede jedinične produkcije. To je u našem slučaju slijedeća produkcija:

B -> A

Ovu produkcija je istovjetna SVIM produkcijama koje nezavršni znak A vrši, tako da umjesto nje pišemo (iste produkcije od A, samo zamijenimo znak s B; produkcije od A ne diramo!):

B -> bB
B -> bac[€]
B -> bS

2.2.3. Slijede produkcije koje dodaju više od jednog završnog znaka na kraj niza. To su sljedeće (uključujući one koje smo definirali u prošlim koracima):

S -> baA
A -> bac[€]
B -> ac[€]
B -> abaA

Krećemo redom:

S -> baA

Ovu ćemo produkciju zapisati kao

S -> b[aA]

gdje smo s **[]** označili novi nezavršni znak kojeg smo definirali. Kako svaki nezavršni znak mora imati produkciju, definirat ćemo ju i za njega, i to na sljedeći način:

[aA] -> aA

Ovime smo produkciju **S -> baA** rascjepkali na dvije nove, **S -> b[aA]** i **[aA] -> aA**. Prelazimo na sljedeću produkciju:

A -> bac[€]

Nju redom cjepamo na:

A -> b[ac€]*
[ac€] -> a[c€]
[c€] -> c[€]

(Kako smo za A redefinirali produkciju, ova ista vrijedi i za B, dakle B -> b[ac€])
Sličnu sudbinu dožive i ostale dvije produkcije:

B -> ac[€] =>

B -> a[c€]
[c€] -> c[€]*

te

B -> abaA =>

B -> a[baA]
[baA] -> b[aA]
[aA] -> aA*

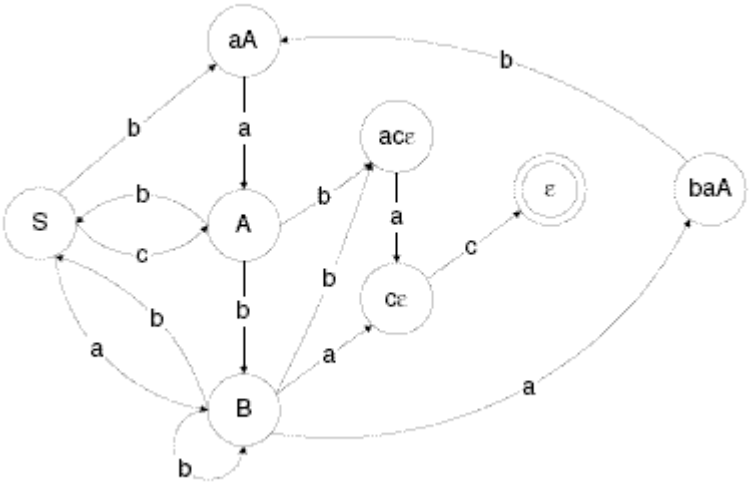
*(već definirana, ovdje sam je ostavio samo radi konzistentnosti postupka)

2.2.4. Nakon ovoga svega, popišemo i grupiramo sve produkcije koje smo izveli (moramo paziti pritom da nismo ništa izostavili):

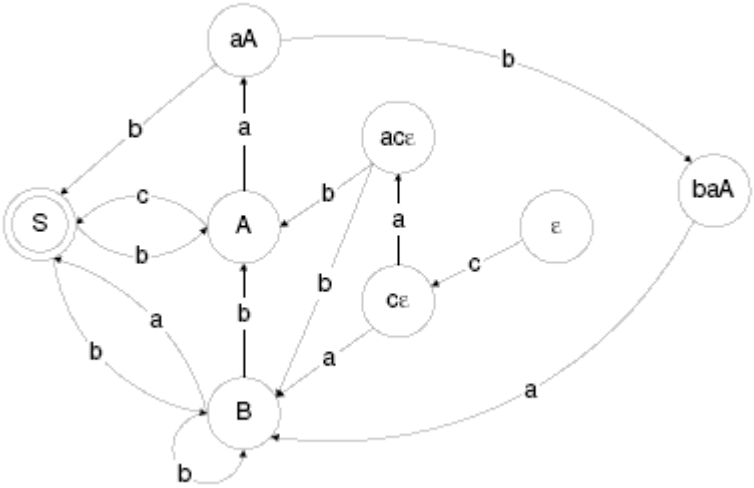
S -> aB	A -> bB	B -> a[baA]	[aA] -> aA	[€] -> €
S -> b[aA]	A -> bS	B -> a[c€]	[c€] -> c[€]	
S -> cA	A -> b[ac€]	B -> b[ac€]	[ac€] -> a[c€]	
		B -> bS	[baA] -> b[aA]	
		B -> bB		

3. Izrada NKA za G'. Ako se sjetimo 13. zadatka i analogije automata i gramatike, vidjet ćemo da u tablici gore imamo definirana sva stanja, sve ulazne znakove i sve prijelaze NKA koji će prihvaćati ovu gramatiku!
- Ulazni znakovi su: **a**, **b**, **c**
- Stanja su: **S** (početno stanje), **A**, **B**, **[aA]**, **[c€]**, **[ac€]**, **[baA]** te **[€]** (prihvatljivo stanje).
- Prijelazi su produkcije u tablici.

S ovim je podacima lako nacrtati NKA:



4. Izrada NKA za G (gramatiku zadanu u zadatku). Dakle, napravili smo gramatiku G' koja stvara nizove obrnute od onih stvorenih gramatikom G, te napravili odgovarajući NKA koji te nizove prihvaća. Ali mi želimo NKA koji prihvaća nizove gramatike G!
To se lako rješava, modifikacijom gornjeg NKA: **ZAMJENIMO** uloge početnog i prihvatljivog ("konačnog") stanja, te **OBRNEMO** smjer svih strelica (upravo zbog toga smo definirali nezavršni znak **[€]** te produkciju **[€] -> €**, kako bi stvorili **SAMO JEDNO** prihvatljivo stanje koje će u obrnutom NKA preuzeti ulogu početnog stanja (koje je također samo jedno). I to je to! Traženi NKA izgleda ovako:



15. Zadatak (by Shakan):

Zadatak:

Zadanu gramatiku G pretvoriti u lijevo-linearnu gramatiku.

Kod:

```
S -> xyA        A -> S          B -> x
S -> yxB        A -> y          B -> xS
S -> B
S -> €
```

Rješenje:

Uffff, ala su ga spetljali s ovim automatima, Bože sačuvaj... Umjesto da prtljam po tome, pokušat ću pojasniti onaj "alternativni" (i očito mnogo praktičniji) način rješavanja. Početnu gramatiku ću, kao i oni u njihovom rješenju, nazvati DLG (desno-linearna gramatika), a onu koju konstruiramo LLG (lijevo-linearna gramatika).

1. Definirati nezavršne znakove LLG. Kao prvo, potrebno je dodati novo stanje **F** koje započinje generiranje niza (dakle, buduće početno stanje). **F** će imati produkcije koje će generirati sve one nezavršne znakove čije su produkcije završavale niz u DLG, ZAJEDNO s njihovim produkcijama. Malo puno za prožvakati? Idemo to objasniti korak po korak:

Ove produkcije iz DLG završavaju niz (dakle, NE PROIZVODE druge nezavršne znakove):

S -> €
A -> y
B -> x

Mi stvaramo produkcije za znak F na temelju tih produkcija:

S -> € => F -> S
Iz produkcije **S -> €** stvaramo **F -> S(€)** (**€** se ne piše na kraju, osim ako nije jedini znak). Prva produkcija znaka F prema tome generira nezavršni znak **S** + njegovu produkciju, **€**, dakle **S€**.

A -> y => F -> Ay
B -> x => F -> Bx

2. Preoblikovati preostale produkcije DLG. Nakon prvog koraka ostale su nam ove produkcije:

Kod:

```
S -> xyA        A -> S          B -> xS
S -> yxB
S -> B
```

Krenimo redom:

S -> xyA

Preoblikovanje produkcije vršimo na sljedeći način:

2.1. Na desnoj strani produkcije zamjenimo mjesta nezavršnog znaka (**A**) i niza završnih znakova (**xy**):

xyA => Axy

Ako završnih znakova nema, ovaj se korak preskače.

2.2. Zamjenimo mjesta generirajućem (**S**) i generiranom nezavršnom znaku (**A**):

S -> Axy => A -> Sxy

Korake 2.2 i 2.3 primjenimo na sve navedene produkcije, nakon čega dobivamo ove produkcije:

Kod:

```
S -> A          A -> Sxy        B -> S
S -> Bx                         B -> Syx
```

3. Završavanje LLG. Njima dodamo produkcije znaka F, kao u produkciju S -> €, koja nam služi za dovršavanje niza (kako je sada S konačni nezavršni znak, on mora moći proizvesti barem jedan završni znak. Kako takve produkcije nakon postupka opisanog gore više nema, moramo to sami dodati. Čime smo dobili potpunu tablicu produkcija LLG (i time riješili zadatak):

Kod:

```
F -> S          S -> A          A -> Sxy        B -> S
F -> Ay         S -> Bx                         B -> Syx
F -> Bx         S-> €
```

3.1. Dodatno, možemo zamijeniti znak **F** znakom **S'** (jer je sada upravo taj znak početni nezavršni znak LLG), a znak **S** znakom **F'** (da bi se izbjegla zabuna). Tablica produkcija nakon toga izgleda ovako:

Kod:

```
S' -> F'        F' -> A         A -> F'xy       B -> F'
S' -> Ay        F' -> Bx                        B -> F'yx
S' -> Bx        F' -> €
```

Kao što se da vidjeti, mnogo brži i jednostavniji način, dakle manje mjesta za greške