Prevođenje programskih jezika – beleške sa predavanja
Konačni automati

Milan Banković

*Matematički fakultet,
Univerzitet u Beogradu

Jesenji semestar 2025/26.

Pregled

Uvod u automate

Definicija 1

Automati su formalni modeli izračunavanja koji omogućavaju prepoznavanje jezika, tj. ispitivanje da li data reč pripada nekom fiksiranom jeziku.

Šta ovo znači?

- Za svaki fiksirani jezik konstruiše se poseban automat koji prepoznaje samo taj jezik
- Automat na ulazu prihvata reč, a na izlazu daje odgovor: da (pripada) ili ne (ne pripada jeziku automata)
  - Otuda je automat model procedure odlučivanja za problem pripadanja jeziku
- Automati su bliski Tjuringovim mašinama, ali im je moć prepoznavanja manja:
  - Tjuringova mašina predstavlja model proizvoljnog algoritma i pomoću nje se može opisati proizvoljna procedura (polu)odlučivanja
  - Automati predstavljaju model uže klase algoritama za prepoznavanje jezika koji pripadaju određenim klasama (poput regularnih jezika, kontekstno slobodnih jezika, i td.)
- Sa druge strane, automati su jednostavniji od Tjuringovih mašina
  - Složenost izračunavanja automata je, u slučaju determinističkih automata, linearna u odnosu na dužinu reči na ulazu
  - Jednom definisani automat se relativno lako može implementirati u većini programskih jezika

Vrste automata

Neke značajne klase automata:

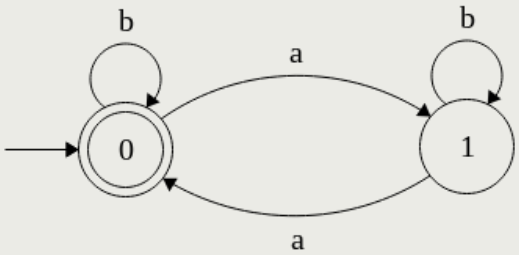
- Konačni automati (engl. finite state automata) – prepoznaju regularne jezike
- Potisni automati (engl. push-down automata) – prepoznaju kontekstno slobodne jezike
- Linearno ograničeni automati (engl. linear bounded automata) – prepoznaju kontekstno zavisne jezike
  - U kontekstno zavisnim gramatikama leve strane pravila mogu biti proizvoljne niske iz $(\Sigma \cup N)^*$, uz uslov da desna strana pravila ne bude kraća od leve
  - Na primer, možemo imati pravilo $aSb \longrightarrow acb$, što znači da se simbol S može zameniti sa c samo kada se nalazi u datom ,,kontekstu" (između a i b)
  - Sa druge strane, kontekstno slobodno pravilo $S \longrightarrow c$ omogućava zamenu S sa c uvek, bez obzira na kontekst
  - Klasa kontekstno zavisnih jezika sadrži u sebi sve kontekstno slobodne jezike koji ne sadrže praznu reč
- Potisni automati sa dva steka – ekvivalentni su sa Tjuringovim mašinama

Pregled

Konačni automati

Primer

Konačni automat se može predstaviti u obliku grafa:

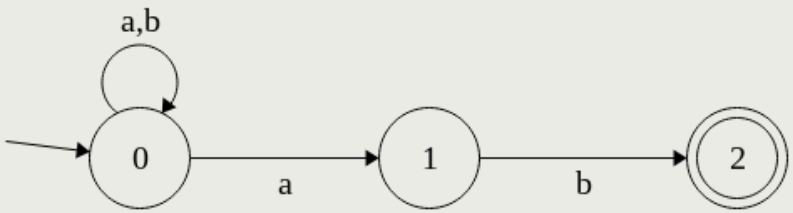


Čvorovi grafa predstavljaju stanja kojih ima konačno mnogo. Lukovi grafa predstavljaju prelaze. Prelazi su označeni simbolima azbuke Σ (u našem primeru $\Sigma = \{a, b\}$). Stanja u koja ulazi strelica koja ne izlazi ni iz jednog stanja se nazivaju početna stanja (stanje 0 kod nas). Stanja koja su označena dvostrukim kružićem se nazivaju završna stanja (opet stanje 0 kod nas). Kao što se vidi, stanje može istovremeno biti i početno i završno.
Automat polazi iz nekog početnog stanja i čita slovo po slovo reči $w \in \Sigma^*$ sa ulaza. Za svako pročitano slovo, automat koristi neki od lukova koji je označen tim slovom da pređe iz tekućeg stanja u neko sledeće stanje. Reč je prepoznata automatom ako se nakon čitanja svih slova reči automat nalazi u završnom stanju.
Automat iz ovog primera prepoznaje sve reči koje imaju paran broj slova a u sebi.

Konačni automati

Primer

Konačni automat sa sledeće slike:

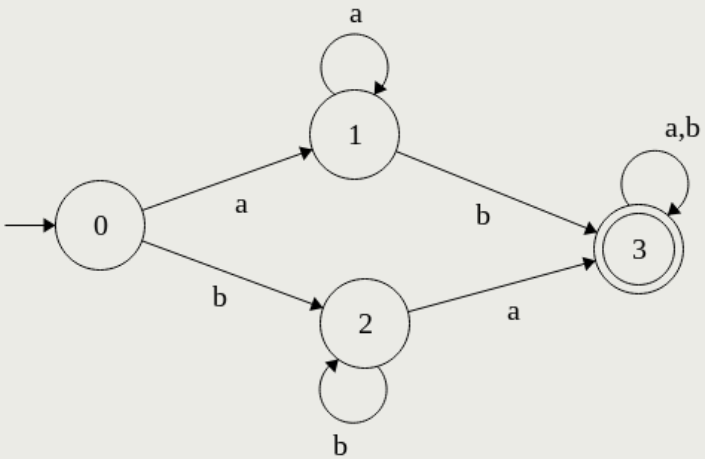


prepoznaje sve reči nad azbukom $\Sigma = \{a, b\}$ *koje se završavaju sa ab. Ono što primećujemo na ovom primeru je da kada se nalazimo u stanju* 0*, a na ulazu se nalazi slovo a, tada imamo dve mogućnosti: ili da ostanemo u stanju* 0 *ili da pređemo u stanje* 1*. Ovo naš automat čini nedeterminističkim.*

Konačni automati

Primer

Konačni automat sa sledeće slike:



prepoznaje jezik svih reči nad azbukom $\Sigma = \{a, b\}$ koje sadrže bar jedno slovo a i bar jedno slovo b. Ovaj automat je deterministički i potpun: za svako stanje i svako slovo azbuke postoji tačno jedan luk koji izlazi iz tog stanja i obeležen je tim slovom.

Konačni automati

Definicija 2

Konačni automat je uređena petorka $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$, *gde je:*

- Σ – *konačna azbuka automata*
- Q – *konačni skup stanja*
- $I \subseteq Q$ – *skup početnih stanja*
- $F \subseteq Q$ – *skup završnih stanja*
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ – *skup prelaza*

Prelaz $(q, a, r) \in \Delta$ *ćemo obično zapisivati kao* $q \xrightarrow{a} r$. *Izračunavanje* c *u automatu* $\mathcal{A}$ *je niz prelaza* $c : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} q_n$. *Pritom, reč* $w = a_1 a_2 \ldots a_n$ *nazivamo etiketom izračunavanja* c *(u oznaci* $w = |c|$*). Izračunavanje* c *ćemo kraće zapisivati sa* $c : q_0 \xRightarrow{w} q_n$. *Izračunavanje* c *je uspešno ako kreće iz početnog stanja i završava u završnom stanju. Jezik automata* $\mathcal{A}$ *(u oznaci* $L(\mathcal{A})$*) je skup etiketa svih uspešnih izračunavanja u automatu:*

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists c : q_0 \xRightarrow{w} q_n, q_0 \in I, q_n \in F\}$$

Za jezik kažemo da je prepoznatljiv ako postoji konačni automat koji ga prepoznaje. Skup svih prepoznatljivih jezika nad Σ *označavamo sa* $P(\Sigma)$.

Konačni automati

Primedba

Primetimo da je u definiciji automata skup prelaza Δ podskup od $Q \times (\Sigma \cup \{\varepsilon\}) \times Q$: ovo znači da etiketa prelaza može biti i prazna reč ε. Takve prelaze nazivamo ε-prelazi.

Pregled

Potpuni deterministički konačni automati

Definicija 3

Automat je potpun ako za svako stanje $q \in Q$ i svaki simbol $a \in \Sigma$ postoji prelaz $(q, a, r) \in \Delta$, za bar jedno $r \in Q$.

Upotpunjavanje konačnog automata

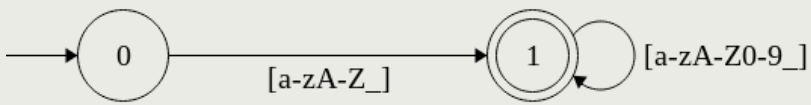
Svaki nepotpuni automat se može upotpuniti dodavanjem stanja greške *Err*:

- stanje greške je nezavršno
- svi prelazi koji nedostaju se usmeravaju ka stanju greške
- iz stanja greške se za sve simbole iz Σ prelazi u to isto stanje

Potpuni deterministički konačni automati

Primer

Automat sa sledeće slike:

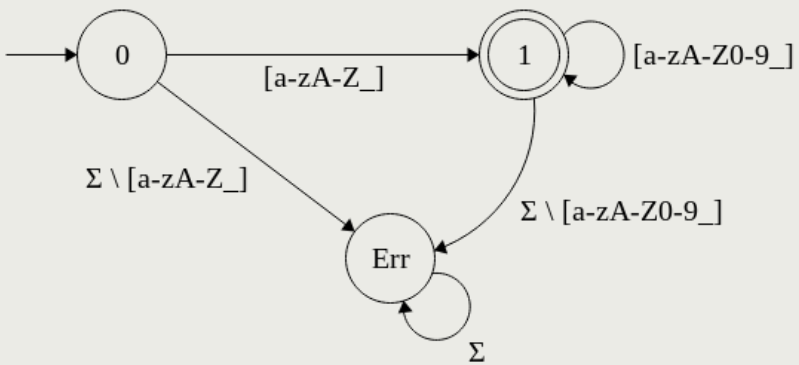


prepoznaje jezik svih identifikatora u C-u. Ipak, ovaj automat je nepotpun – ako naiđe simbol koji se ne očekuje na tom mestu u identifikatoru, automat neće moći da napravi prelaz (kažemo i da automat zaglavljuje).

Potpuni deterministički konačni automati

Primer

Automat na sledećoj slici (nad azbukom ASCII karaktera) dobija se upotpunjavanjem automata iz prethodnog primera:



Ovaj (potpuni) automat takođe prepoznaje jezik svih identifikatora u jeziku C.

Potpuni deterministički konačni automati

Primedba

U prethodnom primeru, stanje *Err* je predstavljalo stanje greške. Ovo stanje je nezavršno i ulaskom u to stanje ostajemo u njemu zauvek. U to stanje se ulazi u slučaju nedozvoljenog prefiksa, tj. prefiksa ulaza koji ne predstavlja prefiks ni jedne reči jezika koji prepoznajemo.

- Ne moraju svi automati imati stanje greške
- Stanje greške postoji u slučaju da za jezik koji prepoznajemo postoji neki nedozvoljeni prefiks
  - Na primer, kod identifikatora, čim naiđe neki karakter koji se ne može pojaviti u identifikatoru ulazimo u stanje greške – naredni karakteri na ulazu ne mogu ,,popraviti" štetu koja je nastala nailaskom tog karaktera
- Postoje jezici kod kojih za svaki prefiks iz Σ^* postoji reč jezika koja ima taj prefiks:
  - Na primer, u jeziku koji sadrži paran broj slova *a*, uvek možemo dopisati još jedno *a* da broj pojavljivanja tog slova učinimo parnim
  - Dakle, uvek ima nade da će simboli koji nailaze u nastavku ,,popraviti" trenutno stanje

Potpuni deterministički konačni automati

Napomene

- Potpunost automata je veoma bitno svojstvo, jer mnoga teorijska razmatranja o automatima pretpostavljaju potpunost
- U praksi, stanje greške je uglavnom implicitno – kada nemamo prelaz, prekidamo dalje čitanje ulaza i prijavljujemo grešku

Potpuni deterministički konačni automati

Definicija 4

Konačni automat je deterministički ako su ispunjeni sledeći uslovi:

- *postoji tačno jedno početno stanje*
- *nema ε-prelaza*
- *za svako $q \in Q$ i svako $a \in \Sigma$ postoji najviše jedno stanje $q' \in Q$ takvo da je $(q, a, q') \in \Delta$.*

Napomena

Ako je automat deterministički i potpun, tada za svako $q \in Q$ i svako $a \in \Sigma$ postoji tačno jedno stanje $q' \in Q$ takvo da je $(q, a, q') \in \Delta$. Tada možemo definisati funkciju prelaska:

$$\delta(q, a) = q' \Leftrightarrow (q, a, q') \in \Delta$$

Potpuni deterministički konačni automati

Teorema 1

Ako je automat deterministički, tada za svaku reč $w \in \Sigma^*$ *postoji najviše jedno izračunavanje* $c : p \stackrel{w}{\Longrightarrow} q$ *sa etiketom* w*, takvo da je* $p \in I$*.*

Teorema 2

Ako je automat potpun i deterministički tada za svaku reč $w \in \Sigma^*$ *postoji tačno jedno izračunavanje* $c : p \stackrel{w}{\Longrightarrow} q$ *sa etiketom* w*, takvo da je* $p \in I$*.*

Potpuni deterministički konačni automati

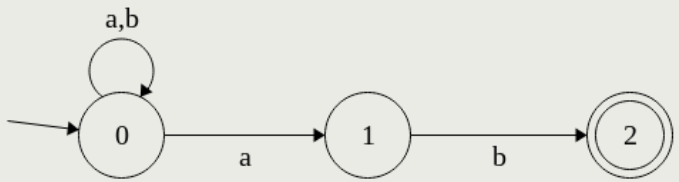
Napomene

- Deterministički automat omogućava prepoznavanje reči u linearnom vremenu u odnosu na dužinu reči
- Sa druge strane, izvršavanje nedeterminističkih automata se može simulirati vraćanjem unazad (engl. backtracking), što u najgorem slučaju daje eksponencijalnu složenost
- Otuda je poželjno da automat koji koristimo bude deterministički

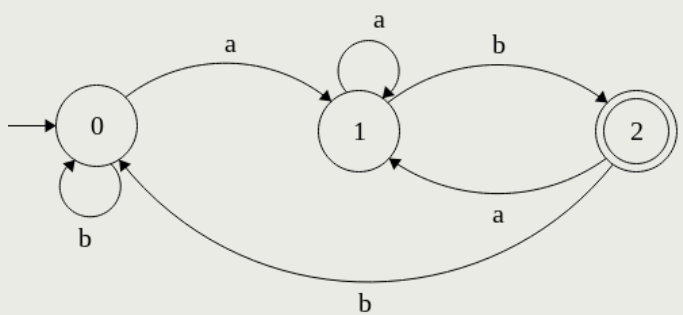
Potpuni deterministički konačni automati

Primer

Setimo se automata koji prepoznaje reči koje se završavaju sa ab:



Konstatovali smo da je ovaj automat nedeterministički. Ekvivalentan deterministički automat je dat na sledećoj slici:



Potpuni deterministički konačni automati

Da li uvek možemo popraviti nedeterminističnost?

- U prethodnom primeru smo do ekvivalentnog determinističkog automata došli *ad-hoc* pristupom
- Da li postoji sistematski pristup determinizaciji konačnih automata?
  - Da li za svaki nedeterministički konačni automat postoji njemu ekvivalentan deterministički?

Potpuni deterministički konačni automati

Konstrukcija po podskupovima

Neka je dat nedeterministički konačni automat $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$. Njemu ekvivalentan potpun i deterministički konačni automat $\overline{\mathcal{A}} = (\Sigma, \mathbb{P}Q, \{I\}, \mathcal{F}, \delta)$ konstruišemo na sledeći način:

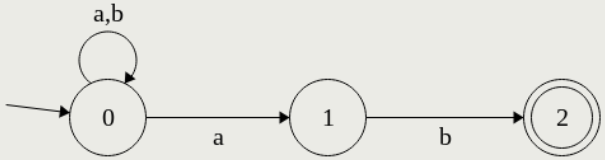
- skup stanja automata $\overline{\mathcal{A}}$ je $\mathbb{P}Q$ – skup svih podskupova od Q
- skup početnih stanja je skup $\{I\}$
- skup završnih stanja je $\mathcal{F} = \{R \subseteq Q \mid R \cap F \neq \varnothing\}$
  - dakle, završno stanje je svaki podskup od Q koji sadrži bar jedno završno stanje automata $\mathcal{A}$
- funkcija prelaska automata $\overline{\mathcal{A}}$ je definisana na sledeći način:

$$\delta(R, a) = \{q \mid \exists r \in R.\ r \stackrel{a}{\Rightarrow} q\}$$

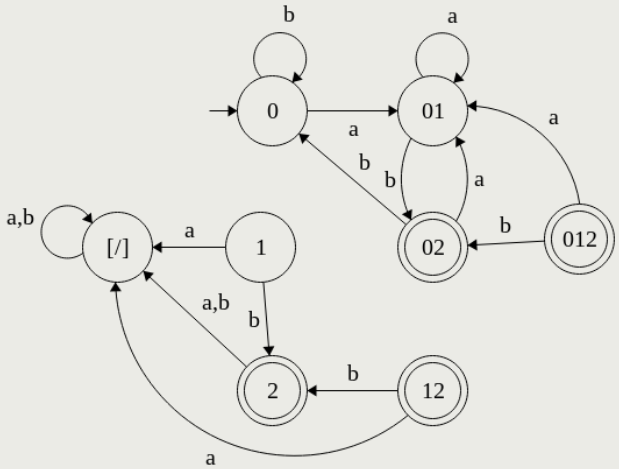
Potpuni deterministički konačni automati

Primer

Posmatrajmo ponovo nedeterministički automat za jezik svih reči koje se završavaju sa ab:



Konstrukcijom po podskupovima dobija se sledeći automat:



Potpuni deterministički konačni automati

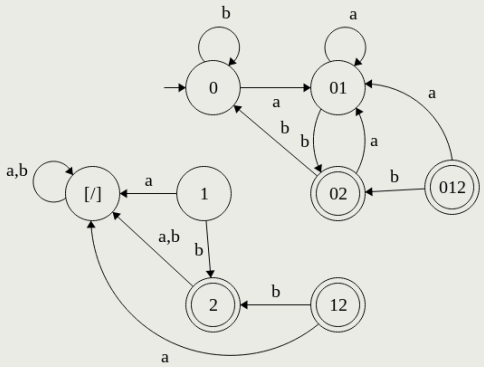
Nedostižna stanja

- U determinističkom automatu iz prethodnog primera postoje stanja koja su nedostižna iz početnog stanja
- Za automat kažemo da je potkresan (engl. pruned) ako nema nedostižnih stanja
- Primenom konstrukcije po podskupovima često se dobijaju nepotkresani automati
- Zbog toga nakon konstrukcije po podskupovima moramo da izvršimo potkresivanje:
  - Obilaskom grafa automata polazeći iz početnog stanja označavamo sva dostižna stanja
  - Stanja koja nisu označena kao dostižna se eliminišu iz automata

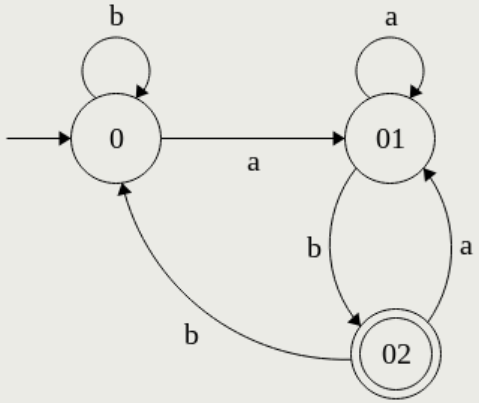
Potpuni deterministički konačni automati

Primer

Posmatrajmo ponovo deterministički automat dobijen u prethodnom primeru:



Potkresivanjem ovog automata dobijamo automat:



Ovaj automat odgovara automatu koji smo dobili ranije ad-hoc pristupom.

Potpuni deterministički konačni automati

Može li bez potkresivanja?

- Konstrukcijom po podskupovima (direktno po definiciji) dobijamo automat sa $2^{|Q|}$ stanja
- U mnogim slučajevima, nakon potkresivanja, dobija se automat sa znatno manje stanja
- Otuda je mnogo zgodnije da se umesto potkresivanja odmah konstruišu samo podskupovi koji odgovaraju dostižnim stanjima

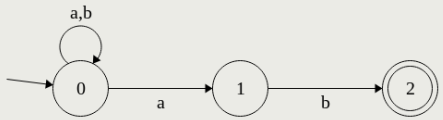
Algoritam konstrukcije po podskupovima

- Neka je red stanja S inicijalno prazan
- Najpre formiramo početno stanje automata $\overline{\mathcal{A}}$ (skup I svih početnih stanja polaznog automata $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$)
- Početno stanje dodajemo u skup stanja automata $\overline{\mathcal{A}}$ (kao jedino njegovo početno stanje), kao i u red stanja S
- Dokle god je red stanja S neprazan:
  - Skidamo naredno stanje R iz reda S
  - Za svaki simbol $a \in \Sigma$ formiramo skup stanja $R_a = \{q' \in Q \mid \exists q \in R.\ q \stackrel{a}{\Rightarrow} q'\}$
  - Ako stanje R_a ne postoji već u automatu $\overline{\mathcal{A}}$, dodajemo ga u skup stanja, dodajemo prelaz (R, a, R_a) i dodajemo R_a u red S
  - Ako stanje R_a već postoji u skupu stanja, tada samo dodajemo prelaz (R, a, R_a) u automat
- Postupak se završava kada red S postane prazan
- Završna stanja novog automata su sva stanja koja sadrže bar jedno završno stanje polaznog automata $\mathcal{A}$

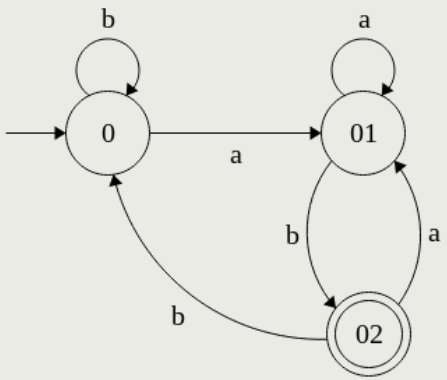
Potpuni deterministički konačni automati

Primer

Polazeći od nedeterminističkog automata:



možemo najpre konstruisati početno stanje (podskup koji se sastoji iz svih početnih stanja polaznog automata). U našem primeru, to je skup $\{0\}$*. Zatim konstruišemo skup koji sadrži sva stanja u koja se može stići preko simbola a iz nekog od stanja u početnom stanju – u našem primeru to je skup* $\{0,1\}$*. Slično, skup svih stanja u koje se može stići simbolom b iz nekog od stanja u početnom stanju će biti skup* $\{0\}$ *(to stanje već imamo). Sada za novodobijeno stanje* $\{0,1\}$ *ponavljamo sličan postupak po svakom od simbola azbuke: za a dobijamo skup* $\{0,1\}$*, dok za b dobijamo skup* $\{0,2\}$*. Ponavljanjem postupka za stanje* $\{0,2\}$ *dobijamo postojeća stanja* $\{0,1\}$ *(za a) i* $\{0\}$ *(za b). Ovim se algoritam završava i dobija se rezultat:*



Potpuni deterministički konačni automati

Konstrukcija po podskupovima i upotpunjavanje

- Algoritam konstrukcije po podskupovima automatski vrši i upotpunjavanje automata, ukoliko on prethodno nije bio potpun
- Ukoliko iz nekog stanja polaznog automata nema prelaza po nekom simbolu azbuke, tada će algoritam konstrukcije po podskupovima proizvesti prazan skup stanja polaznog automata
  - Prazan skup je podskup svakog skupa, pa je po definiciji jedno od mogućih stanja automata koji konstruišemo
  - Iz praznog skupa će svi prelazi voditi ponovo u njega samog, jer nećemo imati stanje u njemu iz koga bismo mogli da odemo u neko drugo stanje polaznog automata
  - Prazan skup će biti nezavršno stanje, jer ne sadrži ni jedno završno stanje polaznog automata
  - Otuda, prazno stanje igra ulogu stanja greške
- Ovo stanje će se automatski kreirati ako je jezik takav da sadrži nedozvoljene prefikse
- VAŽNO: Ne kreirati ručno stanje greške u automatu pre determinizacije, jer će ga algoritam determinizacije sam kreirati, ako je potrebno

Potpuni deterministički konačni automati

Šta sa ε-prelazima?

- Iako po definiciji automati mogu imati i ε-prelaze, u dosadašnjim primerima ih nismo imali
- ε-prelazi se obično dobijaju u nekim standardnim konstruktivnim postupcima koje ćemo kasnije učiti
- Automati koji sadrže ε-prelaze su po definiciji nedeterministički
- Pitanje je na koji način se konstrukcija po podskupovima može primeniti na automate sa ε-prelazima?
  - Prvi način je da se najpre oslobodimo ε-prelaza, pa da onda primenimo konstrukciju po podskupovima
  - Drugi način je da se eliminacija ε-prelaza integriše u postupak konstrukcije po podskupovima
- Ovaj drugi pristup je češći, ali ćemo ipak jednim primerom najpre ilustrovati prvi pristup

Potpuni deterministički konačni automati

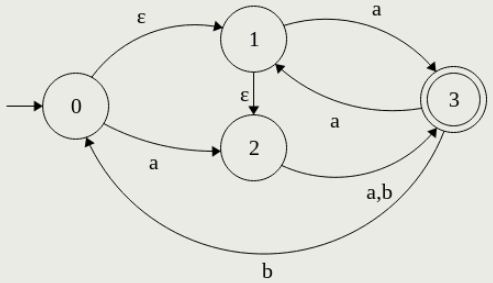
Algoritam eliminacije ε-prelaza

- Za svako stanje $q \in Q$ formiramo njegovo ε-zatvorenje $\varepsilon(q) = \{q' \in Q \mid q \stackrel{\varepsilon}{\Rightarrow} q'\}$
- Novi automat će za skup stanja imati skup $\varepsilon(Q) = \{\varepsilon(q) \mid q \in Q\}$ (skup svih ε-zatvorenja)
- Prelaz $(\varepsilon(q), a, \varepsilon(r))$ će u novom automatu postojati akko postoji prelaz (q', a, r) u starom automatu, za neko $q' \in \varepsilon(q)$
- Početna stanja novog automata su ε-zatvorenja početnih stanja polaznog automata
- Završna stanja novog automata su sva ε-zatvorenja koja sadrže bar jedno završno stanje polaznog automata

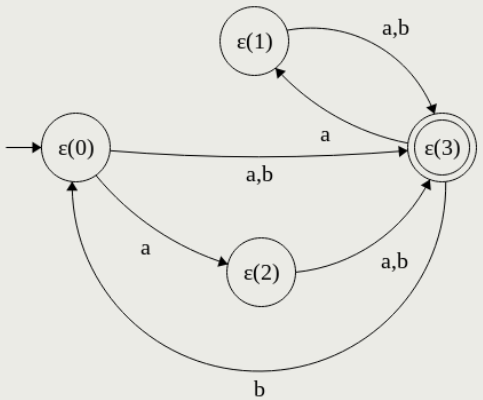
Potpuni deterministički konačni automati

Primer

Posmatrajmo automat:



Imamo da je $\varepsilon(0) = \{0, 1, 2\}$, $\varepsilon(1) = \{1, 2\}$, $\varepsilon(2) = \{2\}$, $\varepsilon(3) = \{3\}$. *Ova* ε*-zatvorenja biće stanja novog automata. Formiranjem odgovarajućih prelaza, dobijamo:*

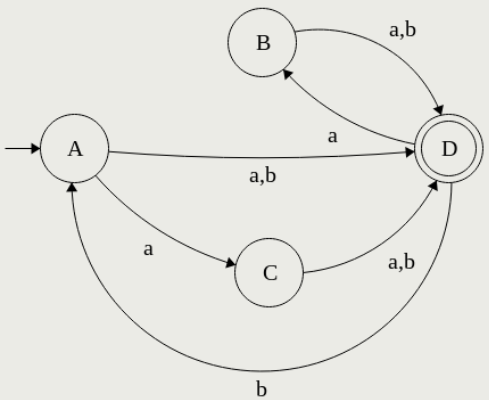


Ovaj automat nema ε*-prelaza, ali je nedeterministički.*

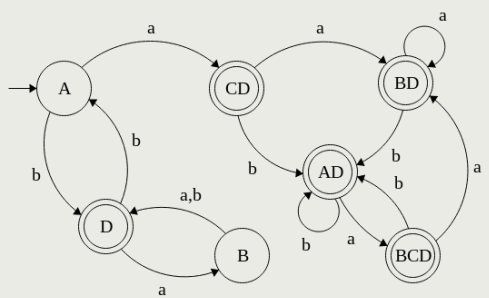
Potpuni deterministički konačni automati

Primer

Zbog jednostavnosti notacije, preimenujmo stanja automata dobijenog u prethodnom primeru:



Njegovom determinizacijom (konstrukcijom po podskupovima), dobijamo:



Potpuni deterministički konačni automati

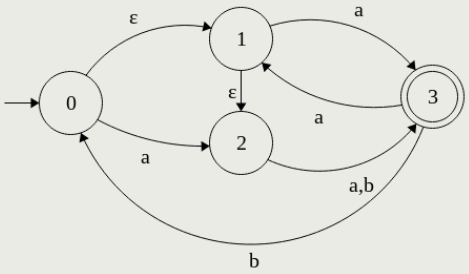
Determinizacija bez prethodnog uklanjanja ε-prelaza

- Pažljivom analizom algoritma konstrukcije po podskupovima vidimo da njemu ε-prelazi ne smetaju
- Naime, setimo se kako smo konstruisali stanje R_a u koje prelazimo iz stanja R po simbolu a: $R_a = \{q' \in Q \mid \exists q \in R.\ q \stackrel{a}{\Rightarrow} q'\}$
  - Ključno u ovoj formulaciji je da imamo izračunavanje $q \stackrel{a}{\Rightarrow} q'$, a ne prelaz $q \stackrel{a}{\rightarrow} q'$
  - Otuda, dozvoljena su i višestruka ,,preskakanja" po ε-prelazima, kako pre, tako i posle prelaza po simbolu a
  - U nekim knjigama, konstrukcija po podskupovima se definiše restriktivnije (koriste se prelazi umesto izračunavanja): u takvoj formulaciji se algoritam može koristiti samo na automate bez ε-prelaza
  - Naša verzija algoritma je proširena tako da u sebi sadrži i algoritam eliminacije ε-prelaza
- Da bismo efikasno konstruisali automat $\overline{\mathcal{A}}$, potrebna su nam ε-zatvorenja:
  - Početno stanje automata $\overline{\mathcal{A}}$ čini unija ε-zatvorenja svih početnih stanja polaznog automata $\mathcal{A}$
  - Svaki put kada u R_a dodamo neko stanje q', dodajemo i sva stanja iz njegovog ε-zatvorenja

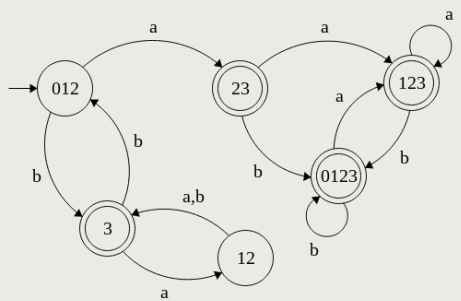
Potpuni deterministički konačni automati

Primer

Krenimo ponovo od automata:



Setimo se i njegovih ε-zatvorenja: $\varepsilon(0) = \{0, 1, 2\}$, $\varepsilon(1) = \{1, 2\}$, $\varepsilon(2) = \{2\}$, $\varepsilon(3) = \{3\}$. Početno stanje čini ε-zatvorenje početnog stanja 0 polaznog automata: to je skup $\{0, 1, 2\}$. Sada postupak konstrukcije po podskupovima primenjujemo kao i ranije, s tim što uvek zajedno sa nekim stanjem u skup dodajemo i sva stanja iz njegovog ε-zatvorenja. Na primer, iz stanja 3 se pomoću simbola a može stići u stanje 1, kao i u sva stanja iz $\varepsilon(1)$, a tu je još i stanje 2. Otuda će svaki prelaz po a iz stanja koje sadrži stanje 3 obavezno pored stanja 1 sadržati i stanje 2. Rezultat koji se dobija na kraju je:



Potpuni deterministički konačni automati

Teorema 3

Jezik L je prepoznatljiv akko postoji potpuni deterministički konačni automat (PDKA) koji ga prepoznaje.

Dokaz

Teorema sledi iz gore opisanog postupka kojim se za svaki automat može konstruisati njemu ekvivalentan PDKA.

Važna napomena

Ova teorema ima veoma značajne posledice na teoriju konačnih automata: ona nam govori da su nam deterministički automati dovoljni i da nedeterminizmom u slučaju konačnih automata ništa ne dobijamo, u smislu izražajnosti. U praksi je ovo veoma pozitivan rezultat, jer deterministički automati ne zahtevaju vraćanje unazad u implementaciji, te omogućavaju prepoznavanje reči u linearnom vremenu (jednim prolaskom kroz reč).

Pregled

Osobine prepoznatljivih jezika

Teorema 4

Ako je jezik L prepoznatljiv, tada je i njegov komplement $\Sigma^* \setminus L$ prepoznatljiv.

Dokaz

Posmatrajmo PDKA (potpuni deterministički konačni automat) $\mathcal{A}$ koji prepoznaje jezik L (ovakav automat uvek postoji). Posmatrajmo automat $\mathcal{A}^c$ koji je identičan kao i polazni, s tim što su njegova završna stanja postala nezavršna i obratno. Setimo se da je u svakom PDKA izračunavanje za svaku reč $w \in \Sigma^*$ jedinstveno i ono nas odvodi u jednoznačno određeno stanje $q(w) \in Q$. Kako automat $\mathcal{A}^c$ ima isto početno stanje i iste prelaze kao i $\mathcal{A}$, stanje $q(w)$ biće isto u ovom automatu kao i u automatu $\mathcal{A}$ (za svaku reč w). Međutim, kako smo ovde zamenili završna i nezavršna stanja, sledi da će automat $\mathcal{A}^c$ prepoznavati one i samo one reči koje automat $\mathcal{A}$ ne prepoznaje.

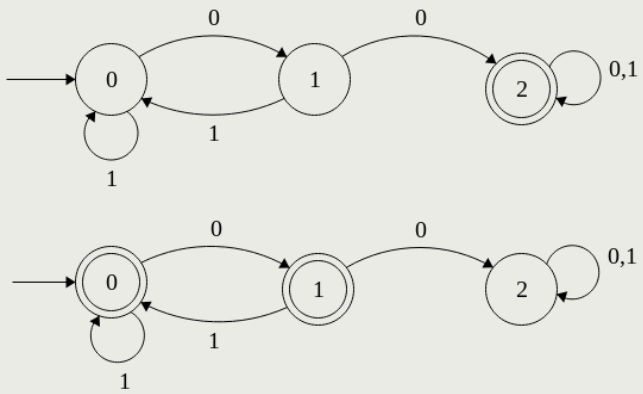
Primedba

Primetimo da opisanu konstrukciju ne možemo primeniti na automate koji nisu potpuni i deterministički, jer za njih ne važi ona primedba o jedinstvenosti izračunavanja (tj. izračunavanje za neku reč w može da ne postoji, ili da postoji, a da ne bude jedinstveno).

Osobine prepoznatljivih jezika

Primer

Posmatrajmo automate:

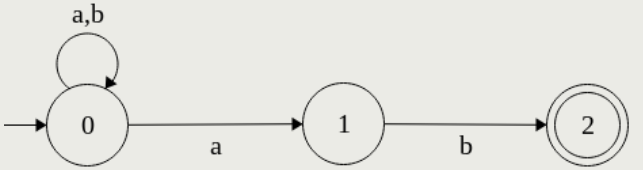


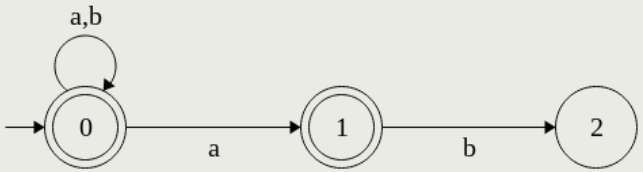
Gornji automat je PDKA koji prepoznaje jezik svih binarnih reči koje sadrže dve uzastopne nule. Donji automat prepoznaje komplementarni jezik – jezik svih reči koje ne sadrže dve uzastopne nule.

Osobine prepoznatljivih jezika

Primer

Posmatrajmo automate:





Gornji automat je nedeterministički automat koji prepoznaje jezik svih reči nad azbukom $\{a, b\}$ koje se završavaju sa ab. Kako ovaj automat nije PDKA, opisana transformacija nad njim ne daje ispravan rezultat – donji automat prepoznaje jezik Σ^*, što nije komplement polaznog jezika.

Osobine prepoznatljivih jezika

Teorema 5

Ako su jezici L_1 i L_2 prepoznatljivi, tada su i jezici $L_1 \cup L_2$, $L_1 \cap L_2$ i $L_1 \setminus L_2$ prepoznatljivi.

Dokaz

Neka su $\mathcal{A}_1 = (\Sigma, Q_1, i_1, F_1, \delta_1)$ i $\mathcal{A}_2 = (\Sigma, Q_2, i_2, F_2, \delta_2)$ PDKA koji prepoznaju jezike L_1 i L_2, respektivno. Formirajmo automat $\mathcal{A} = (\Sigma, Q_1 \times Q_2, (i_1, i_2), F, \delta)$, gde je $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. Skup završnih stanja F zavisi od tražene skupovne operacije:

- *Za uniju: $F = \{(q_1, q_2) \mid q_1 \in F_1 \vee q_2 \in F_2\}$*
- *Za presek: $F = \{(q_1, q_2) \mid q_1 \in F_1 \wedge q_2 \in F_2\}$*
- *Za razliku: $F = \{(q_1, q_2) \mid q_1 \in F_1 \wedge q_2 \notin F_2\}$*

Lako se može proveriti da opisani automat prepoznaje odgovarajući jezik.

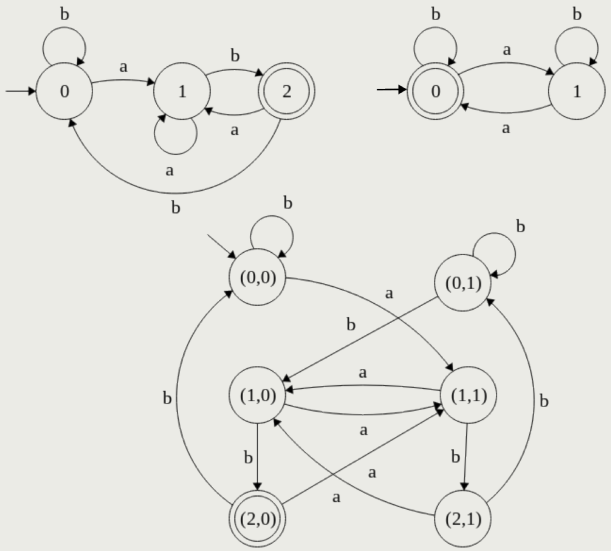
Napomena

Primetimo da se opisanim postupkom može dobiti nepotkresan automat, iako su polazni automati bili potkresani. Otuda nakon ove konstrukcije može biti neophodno dodatno potkresivanje.

Osobine prepoznatljivih jezika

Primer

Posmatrajmo automate na slici:



Gornji levi automat je PDKA koji prepoznaje jezik svih reči koje se završavaju sa ab, dok je gornji desni PDKA automat koji prepoznaje jezik svih reči koje sadrže paran broj slova a. Automat koji prepoznaje presek ova dva jezika (tj. reči koje ispunjavaju i jedan i drugi uslov) dat je u donjem delu slike.

Osobine prepoznatljivih jezika

Teorema 6

Ako su jezici L_1 i L_2 prepoznatljivi, tada je i jezik $L_1 \cdot L_2$ prepoznatljiv.

Teorema 7

Ako je jezik L prepoznatljiv, tada je i jezik L^* prepoznatljiv.

Napomena

Dokazi ove dve teoreme slede iz Tompsonove konstrukcije koju ćemo raditi kasnije, te ih ovde izostavljamo.

Teorema 8

Svaki regularan jezik je prepoznatljiv (tj. $R(\Sigma) \subseteq P(\Sigma)$).

Dokaz

Lako se mogu konstruisati automati koji prepoznaju jezike $\varnothing$, $\{\varepsilon\}$ i $\{a\}$. Sada iz zatvorenosti klase prepoznatljivih jezika za uniju, dopisivanje i Klinijevo zatvorenje sledi teorema.

Pregled

Konstrukcija automata po regularnom izrazu

Kako odrediti automat koji prepoznaje dati regularni jezik?

- Iz prethodne teoreme sledi da se svaki automat opisan regularnim izrazom može prepoznati nekim konačnim automatom
- Ostaje pitanje efektivne konstrukcije takvog automata
- U nastavku izlažemo dve takve konstrukcije: Tompsonovu i Gluškovljevu konstrukciju

Tompsonova konstrukcija

Definicija 5

Automat je normalizovan ako:

- *ima tačno jedno početno i jedno završno stanje*
- *ni jedan luk mu ne ulazi u početno stanje niti izlazi iz završnog stanja*
- *iz svakog stanja izlazi ili tačno jedan luk sa etiketom iz Σ ili najviše dva luka sa etiketom ε*

Teorema 9

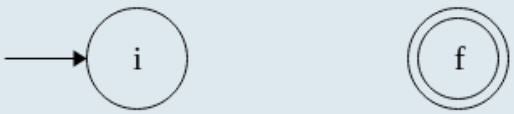
Za svaki regularni jezik L postoji normalizovan konačni automat koji ga prepoznaje.

Dokaz

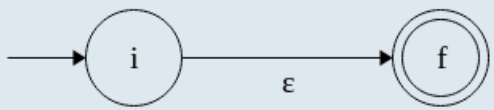
Dokaz sledi iz opisa konstrukcije u nastavku koja je poznata i kao Tompsonova konstrukcija.

Tompsonova konstrukcija

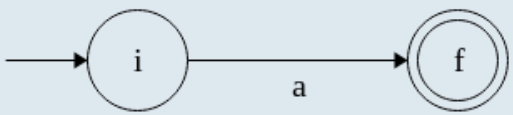
Jezik $\varnothing$



Jezik $\{\varepsilon\}$

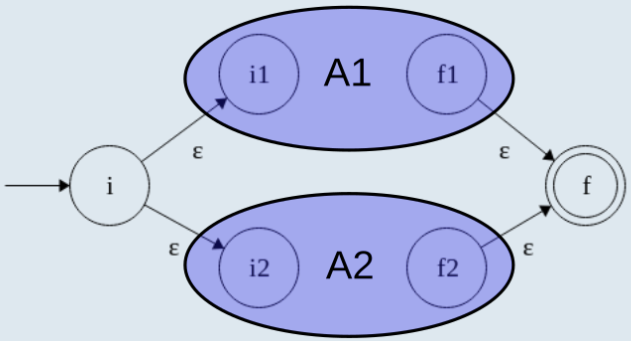


Jezik $\{a\}$

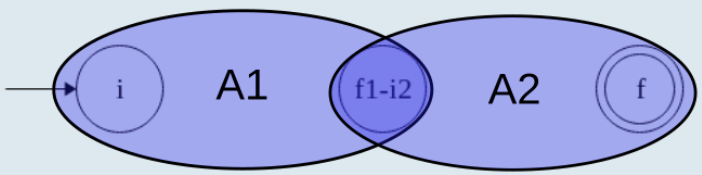


Tompsonova konstrukcija

Jezik $L_1 \cup L_2$

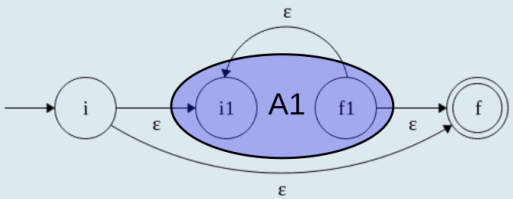


Jezik $L_1 \cdot L_2$

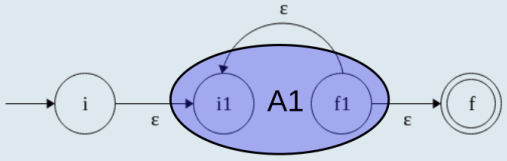


Tompsonova konstrukcija

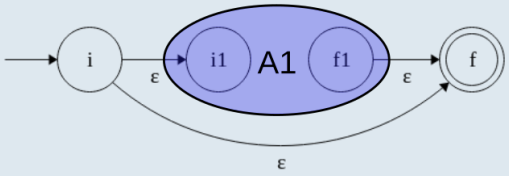
Jezik L^*



Jezik L^+



Jezik $L^?$



Tompsonova konstrukcija

Definicija 6

Obim regularnog izraza r (u oznaci $|r|$) definišemo na sledeći način:

- $|\varnothing| = |\varepsilon| = |a| = 1$
- $|r_1 \mid r_2| = |r_1| + |r_2| + 1$
- $|r_1 r_2| = |r_1| + |r_2|$
- $|r^*| = |r^+| = |r^?| = |r| + 1$

Dakle, u pitanju je ukupan broj simbola u izrazu, ne računajući zagrade.

Teorema 10

Neka je dat regularni izraz r obima m sa k simbola iz Σ u sebi. Automat dobijen Tompsonovom konstrukcijom je normalizovan i ima najviše $2m$ stanja i najviše $4m$ prelaza, od čega je tačno k ne-ε-prelaza.

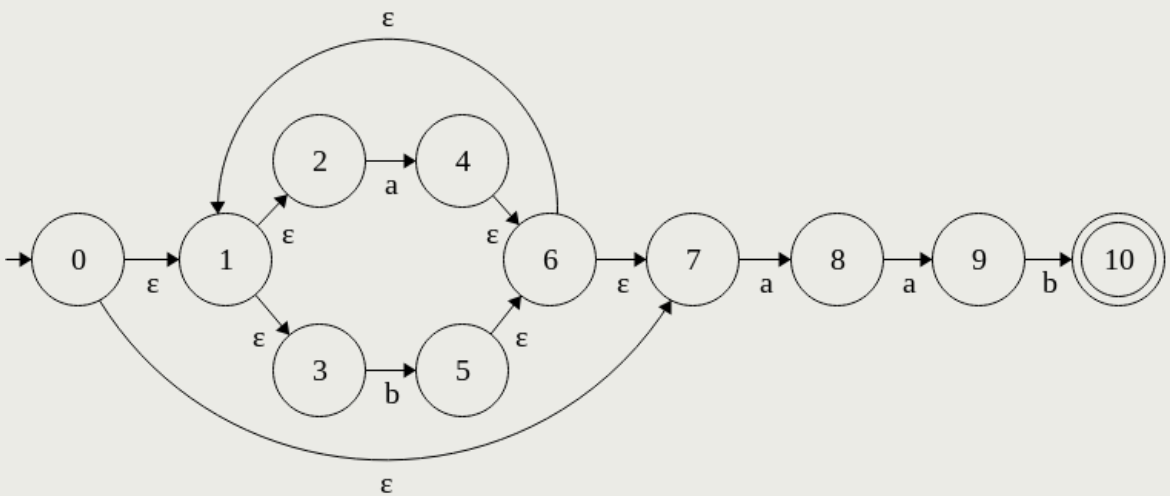
Dokaz

Ne-ε-prelazi odgovaraju slovima (tj. elementima azbuke Σ) u izrazu, pa je njihov broj jednak k. Svaki simbol u regularnom izrazu (ne računajući zagrade) uvodi tačno dva nova stanja pa ukupan broj stanja ne može biti veći od $2m$ (može biti manji, jer operacijom dopisivanja stapamo dva stanja u jedno). Takođe, svaki simbol uvodi najviše četiri nova prelaza. Normalizovanost sledi iz činjenice da svaki od konstruktivnih koraka čuva normalizovanost.

Tompsonova konstrukcija

Primer

Neka je dat regularni izraz: $(a|b)^*aab$. *Automat dobijen Tompsonovom konstrukcijom na osnovu ovog izraza je:*

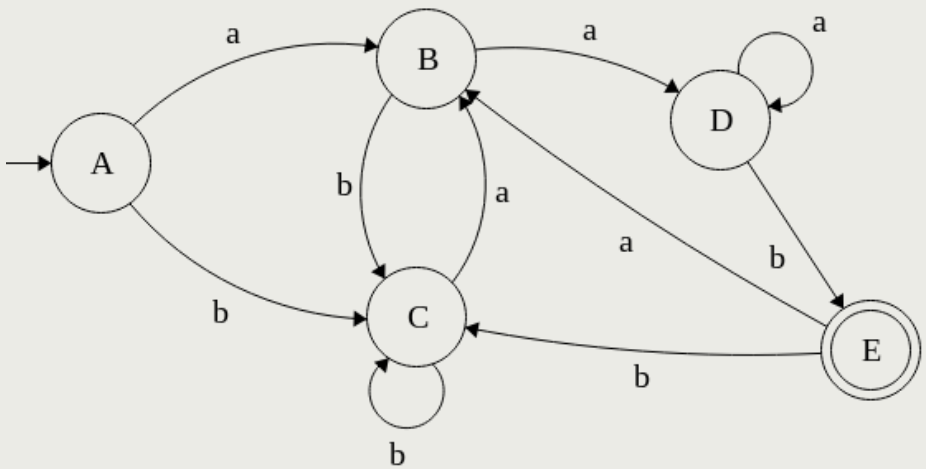


Ovaj automat je nedeterministički. Da bismo ga determinizovali, najpre određujemo ε*-zatvorenja:* $\varepsilon(0) = \{0, 1, 2, 3, 7\}$, $\varepsilon(1) = \{1, 2, 3\}$, $\varepsilon(2) = \{2\}$, $\varepsilon(3) = \{3\}$, $\varepsilon(4) = \{1, 2, 3, 4, 6, 7\}$, $\varepsilon(5) = \{1, 2, 3, 5, 6, 7\}$, $\varepsilon(6) = \{1, 2, 3, 6, 7\}$, $\varepsilon(7) = \{7\}$, $\varepsilon(8) = \{8\}$, $\varepsilon(9) = \{9\}$, $\varepsilon(10) = \{10\}$.

Tompsonova konstrukcija

Primer

Nakon što smo odredili ε-zatvorenja, primenjujemo konstrukciju po podskupovima i dobijamo PDKA:

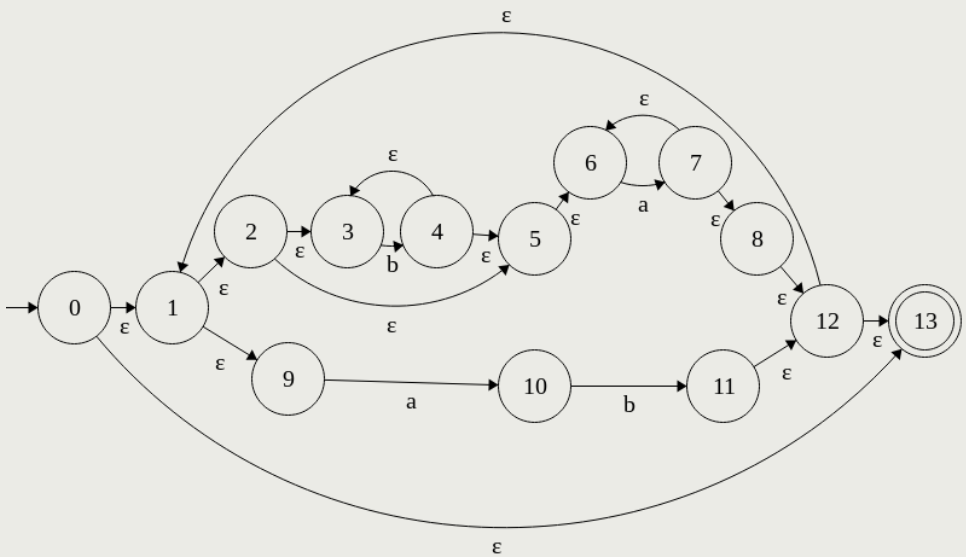


pri čemu je $A = \varepsilon(0) = \{0, 1, 2, 3, 7\}$, $B = \varepsilon(4) \cup \varepsilon(8) = \{1, 2, 3, 4, 6, 7, 8\}$, $C = \varepsilon(5) = \{1, 2, 3, 5, 6, 7\}$, $D = \varepsilon(4) \cup \varepsilon(8) \cup \varepsilon(9) = \{1, 2, 3, 4, 6, 7, 8, 9\}$ *i* $E = \varepsilon(5) \cup \varepsilon(10) = \{1, 2, 3, 5, 6, 7, 10\}$. *Primetimo da broj stanja dobijenog konačnog automata nije tako veliki, s obzirom da u polaznom automatu imamo samo nekoliko ne-ε-prelaza koji nam efektivno određuju stanja PDKA (svako stanje je unija nekih od ε-zatvorenja stanja na kojima se završavaju ne-ε-prelazi polaznog automata). Ti prelazi su u ovom primeru* $2 \xrightarrow{a} 4$, $3 \xrightarrow{b} 5$, $7 \xrightarrow{a} 8$, $8 \xrightarrow{a} 9$, $9 \xrightarrow{b} 10$.

Tompsonova konstrukcija

Primer

Neka je dat regularni izraz: $(ab|b^*a^+)^*$. *Automat dobijen Tompsonovom konstrukcijom na osnovu ovog izraza je:*

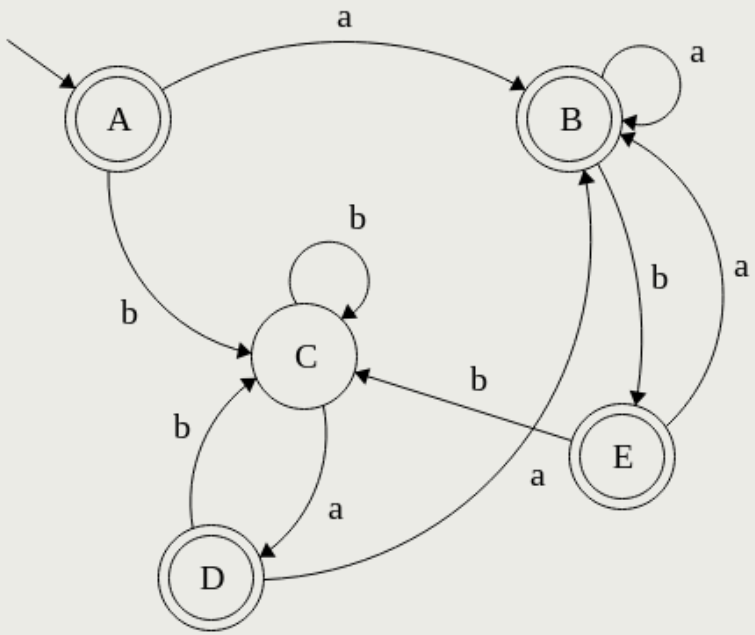


Ne-ε*-prelazi u ovom automatu su* $3 \xrightarrow{b} 4$, $6 \xrightarrow{a} 7$, $9 \xrightarrow{a} 10$, $10 \xrightarrow{b} 11$, *a* ε*-zatvorenja su:*
$\varepsilon(0) = \{0, 1, 2, 3, 5, 6, 9, 13\}$, $\varepsilon(1) = \{1, 2, 3, 5, 6, 9\}$, $\varepsilon(2) = \{2, 3, 5, 6\}$, $\varepsilon(3) = \{3\}$,
$\varepsilon(4) = \{3, 4, 5, 6\}$, $\varepsilon(5) = \{5, 6\}$, $\varepsilon(6) = \{6\}$, $\varepsilon(7) = \{1, 2, 3, 5, 6, 7, 8, 9, 12, 13\}$,
$\varepsilon(8) = \{1, 2, 3, 5, 6, 8, 9, 12, 13\}$, $\varepsilon(9) = \{9\}$, $\varepsilon(10) = \{10\}$, $\varepsilon(11) = \{1, 2, 3, 5, 6, 9, 11, 12, 13\}$,
$\varepsilon(12) = \{1, 2, 3, 5, 6, 9, 12, 13\}$, $\varepsilon(13) = \{13\}$.

Tompsonova konstrukcija

Primer

Nakon primene konstrukcije po podskupovima dobijamo sledeći automat:

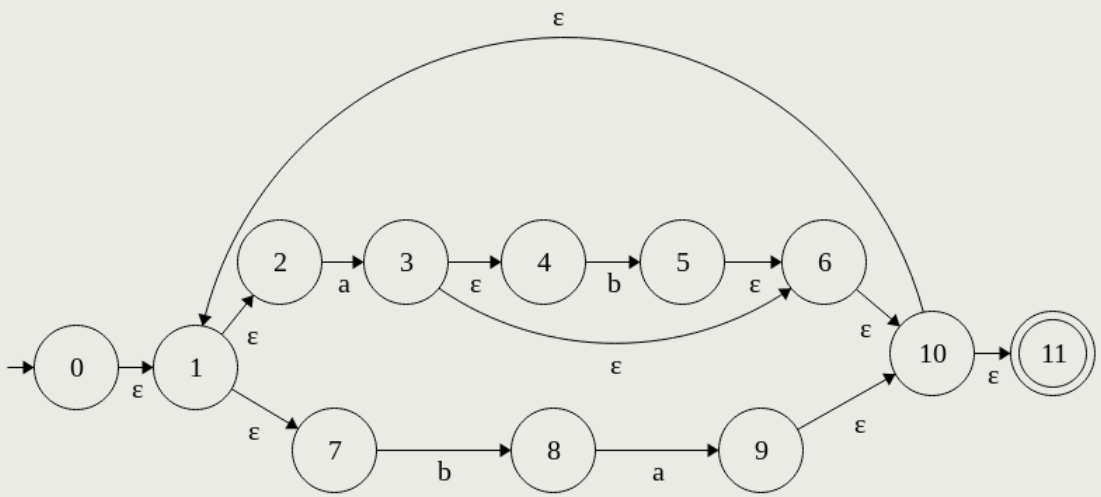


gde je $A = \varepsilon(0) = \{0, 1, 2, 3, 5, 6, 9, 13\}$, $B = \varepsilon(7) \cup \varepsilon(10) = \{1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13\}$,
$C = \varepsilon(4) = \{3, 4, 5, 6\}$, $D = \varepsilon(7) = \{1, 2, 3, 5, 6, 7, 8, 9, 12, 13\}$,
$E = \varepsilon(4) \cup \varepsilon(11) = \{1, 2, 3, 4, 5, 6, 9, 11, 12, 13\}$.

Tompsonova konstrukcija

Primer

Neka je dat regularni izraz: $(ab^?|ba)^+$. *Automat dobijen Tompsonovom konstrukcijom na osnovu ovog izraza je:*

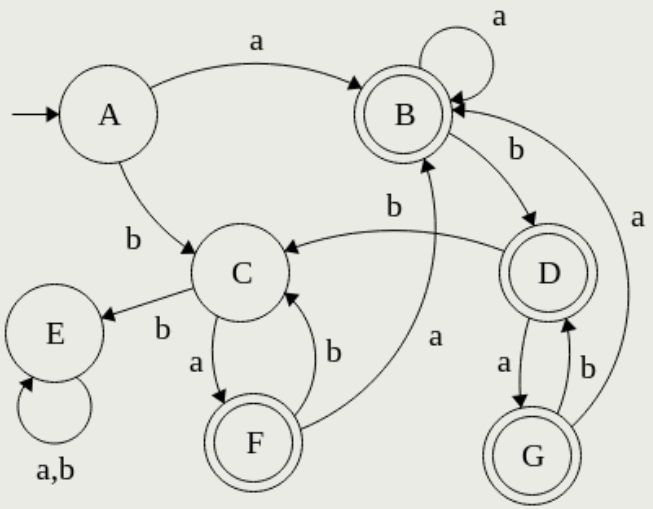


Ne-ε*-prelazi u ovom automatu su* $2 \xrightarrow{a} 3$, $4 \xrightarrow{b} 5$, $7 \xrightarrow{b} 8$, $8 \xrightarrow{a} 9$, *a* ε*-zatvorenja su:*
$\varepsilon(0) = \{0, 1, 2, 7\}$, $\varepsilon(1) = \{1, 2, 7\}$, $\varepsilon(2) = \{2\}$, $\varepsilon(3) = \{1, 2, 3, 4, 6, 7, 10, 11\}$,
$\varepsilon(4) = \{4\}$, $\varepsilon(5) = \{1, 2, 5, 6, 7, 10, 11\}$, $\varepsilon(6) = \{1, 2, 6, 7, 10, 11\}$, $\varepsilon(7) = \{7\}$,
$\varepsilon(8) = \{8\}$, $\varepsilon(9) = \{1, 2, 7, 9, 10, 11\}$, $\varepsilon(10) = \{1, 2, 7, 10, 11\}$, $\varepsilon(11) = \{11\}$.

Tompsonova konstrukcija

Primer

Nakon primene konstrukcije po podskupovima dobijamo sledeći automat:



gde je $A = \varepsilon(0) = \{0, 1, 2, 7\}$, $B = \varepsilon(3) = \{1, 2, 3, 4, 6, 7, 10, 11\}$, $C = \varepsilon(8) = \{8\}$, $D = \varepsilon(5) \cup \varepsilon(8) = \{1, 2, 5, 6, 7, 8, 10, 11\}$, $E = \varnothing$, $F = \varepsilon(9) = \{1, 2, 7, 9, 10, 11\}$, $G = \varepsilon(3) \cup \varepsilon(9) = \{1, 2, 3, 4, 6, 7, 9, 10, 11\}$. *Primetimo da je* E *stanje greške.*

Konstrukcija Gluškova

Konstrukcija Gluškova

Neka je dat regularni izraz r. Primenjujemo sledeći postupak:

- Izraz r transformišemu u linearizovani oblik:
  - ovaj oblik dobijamo tako što redom sa leva na desno indeksiramo slova iz Σ koja se nalaze u izrazu r indeksima počev od 1
  - Na primer, linearizovani oblik izraza $(a|b)^*aab$ je $(a_1|b_2)^*a_3a_4b_5$
- Za svaki od indeksa u linearizovanom obliku izraza r formiramo po jedno stanje označeno tim indeksom
  - dodatno, imamo i stanje 0 koje je početno stanje
- Ako je indeksom i u linearizovanom obliku izraza r indeksiran simbol $s \in \Sigma$, tada će svi prelazi koji ulaze u stanje i imati etiketu s
  - iz početnog stanja 0 imamo prelaze ka svim stanjima koja odgovaraju simbolima kojima može počinjati neka reč jezika $L(r)$
  - iz stanja $i > 0$ imamo prelaze ka svim stanjima koja odgovaraju simbolima koji mogu slediti neposredno nakon simbola koji odgovara stanju i u nekoj reči jezika $L(r)$
- Završna stanja dobijenog automata će biti sva ona stanja koja odgovaraju simbolima kojima se može završiti neka reč jezika $L(r)$
  - specijalno, stanje 0 će biti završno akko $\varepsilon \in L(r)$

Konstrukcija Gluškova

Teorema 11

Automat dobijen konstrukcijom Gluškova ima $k + 1$ stanje, gde je k broj simbola iz Σ koja se pojavljuju u izrazu r (pri čemu se svako pojavljivanje nekog simbola posebno broji). Pritom, dobijeni automat nema ε-prelaza.

Dokaz

Trivijalno sledi iz opisa konstrukcije.

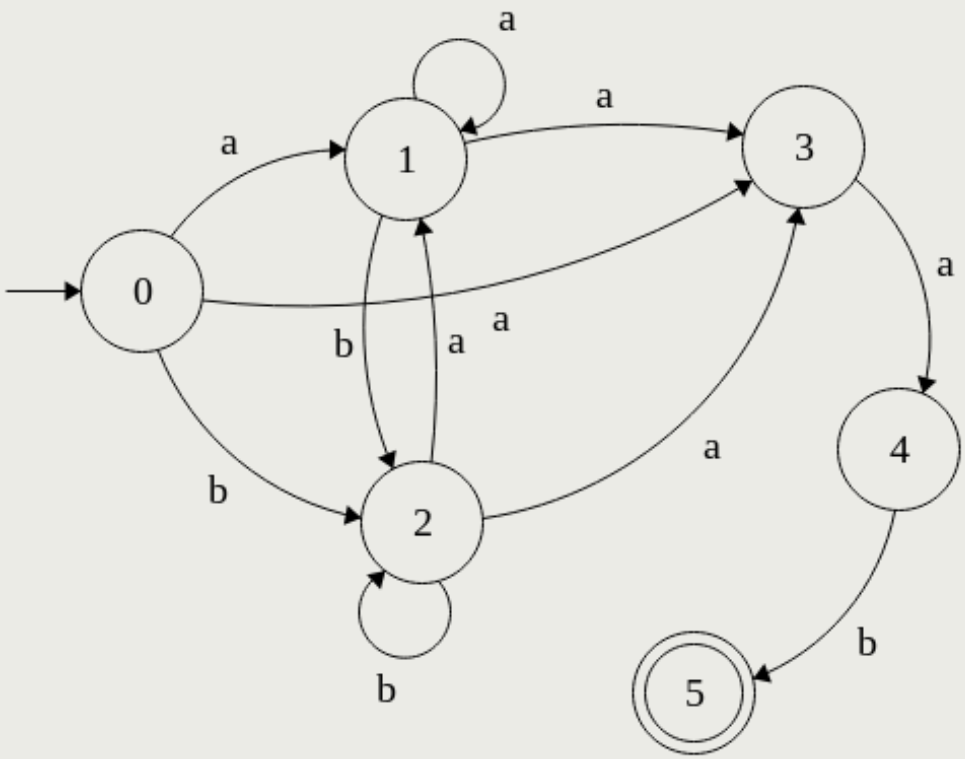
Primedba

Iako dobijeni automat nema ε-prelaza, on ne mora biti (i najčešće nije) deterministički.

Konstrukcija Gluškova

Primer

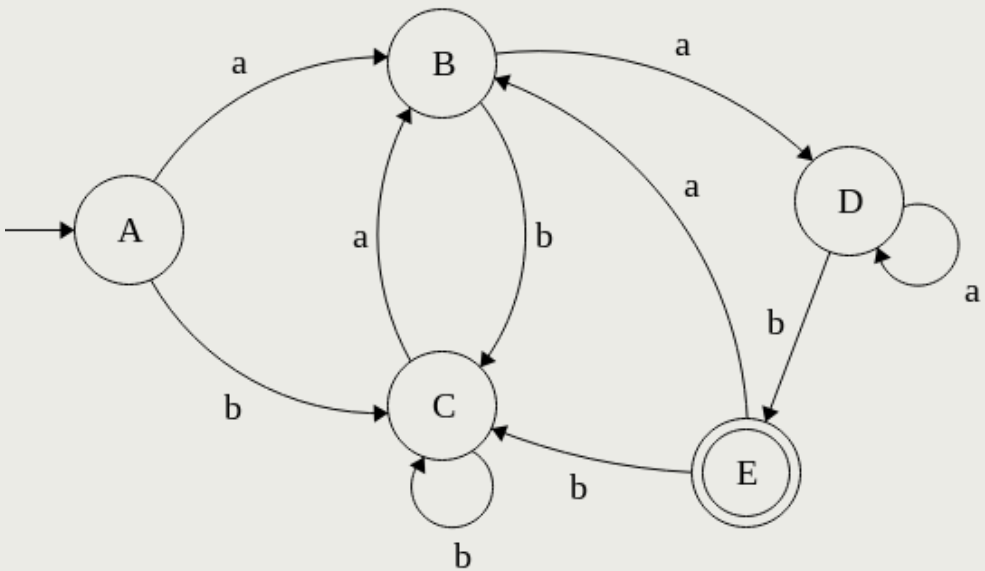
Neka je dat regularni izraz: $(a|b)^*aab$. *Linearizovani oblik ovog izraza je* $(a_1|b_2)^*a_3a_4b_5$, *a automat dobijen konstrukcijom Gluškova je:*



Konstrukcija Gluškova

Primer

Primenom konstrukcije po podskupovima dobijamo PDKA:

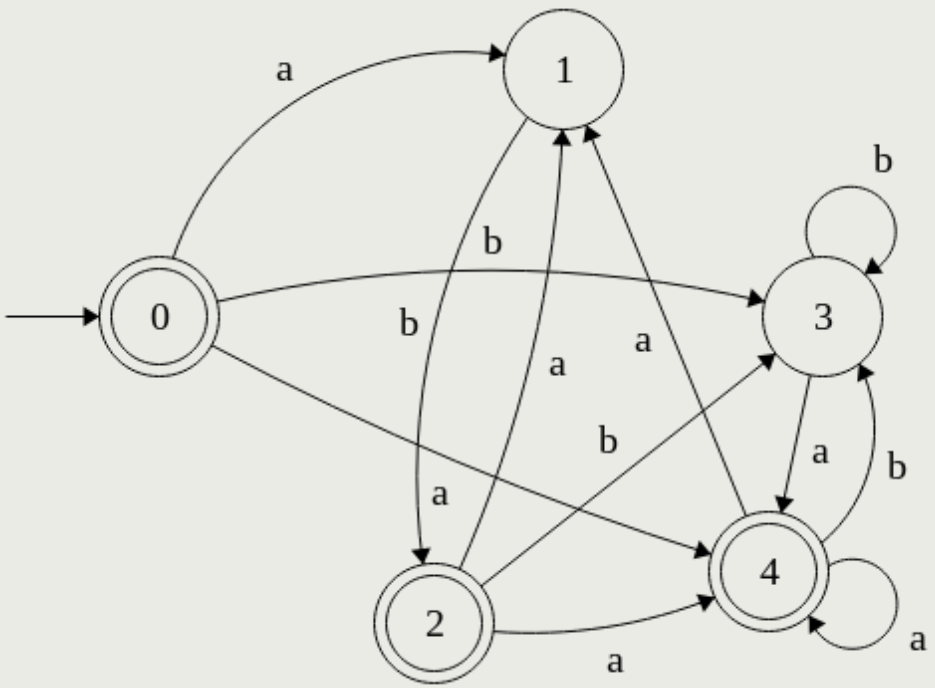


gde je $A = \{0\}$, $B = \{1, 3\}$, $C = \{2\}$, $D = \{1, 3, 4\}$, $E = \{2, 5\}$.

Konstrukcija Gluškova

Primer

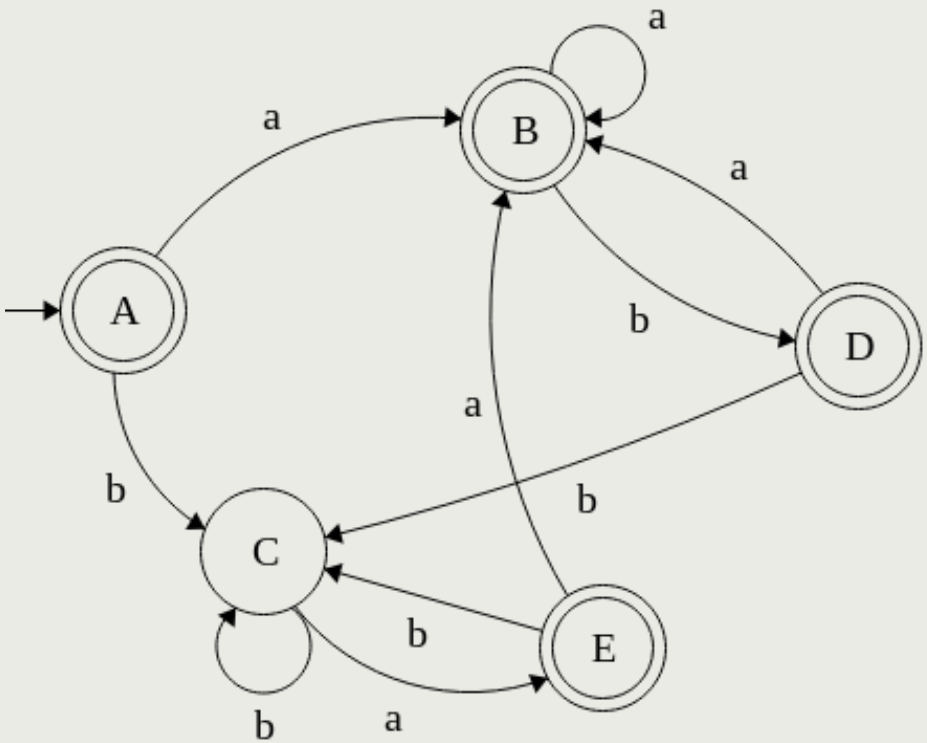
Neka je dat regularni izraz: $(ab|b^*a^+)^*$. *Linearizovani oblik ovog izraza je* $(a_1b_2|b_3^*a_4^+)^*$, *a automat dobijen konstrukcijom Gluškova je:*



Konstrukcija Gluškova

Primer

Primenom konstrukcije po podskupovima dobijamo PDKA:

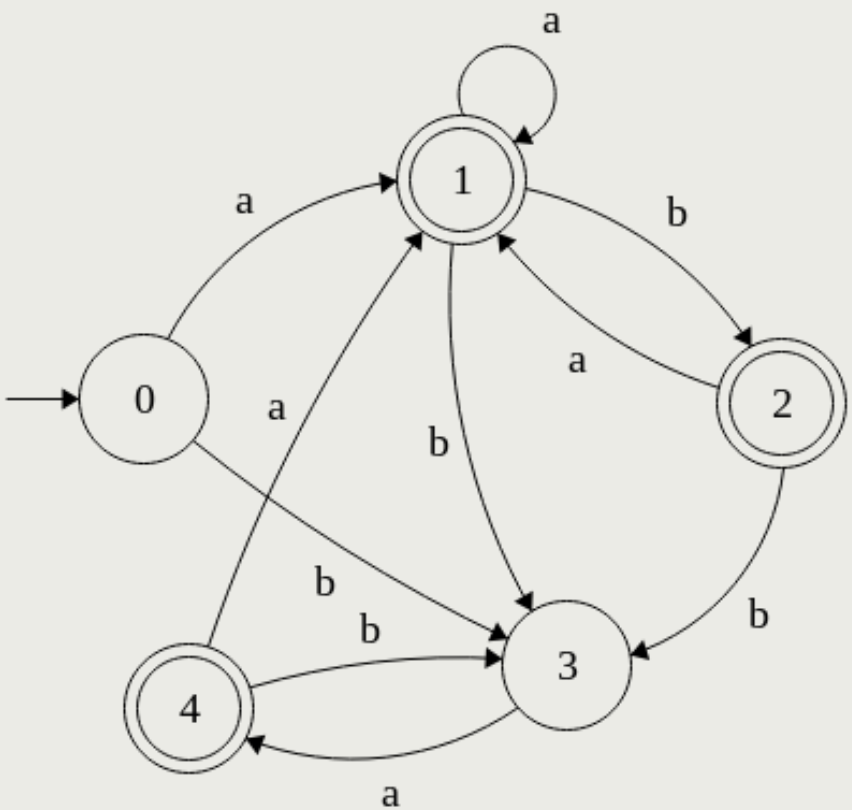


gde je $A = \{0\}$, $B = \{1, 4\}$, $C = \{3\}$, $D = \{2, 3\}$, $E = \{4\}$.

Konstrukcija Gluškova

Primer

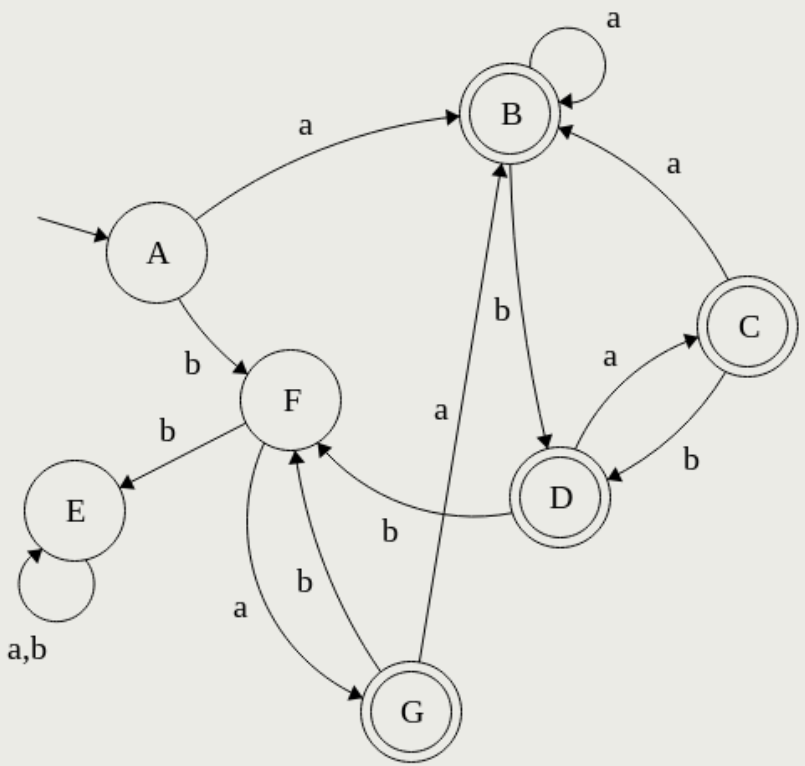
Neka je dat regularni izraz: $(ab^?|ba)^+$. *Linearizovani oblik ovog izraza je* $(a_1 b_2^?|b_3 a_4)^+$, *a automat dobijen konstrukcijom Gluškova je:*



Konstrukcija Gluškova

Primer

Primenom konstrukcije po podskupovima dobijamo PDKA:



gde je $A = \{0\}$, $B = \{1\}$, $C = \{1, 4\}$, $D = \{2, 3\}$, $E = \varnothing$, $F = \{3\}$ *i* $G = \{4\}$.
Primetimo da je E *stanje greške.*

Tompson vs. Gluško

Tompsonova konstrukcija

- Pogodna je za automatizaciju, jer prati strukturu izraza i primenjuje jasno definisane konstruktivne korake
- Automat koji se dobija obično ima veliki broj ε-prelaza, kao i veliki broj stanja
- Ovo predstavlja poteškoću u postupku determinizacije prilikom ručne primene („na papiru”), ali ne i prilikom implementacije

Gluškovljeva konstrukcija

- Nije pogodna za automatizaciju, ali je lakša za ručnu primenu
- Konstrukcijom se ne dobijaju ε-prelazi, a broj stanja dobijenog automata je obično znatno manji nego u slučaju Tompsonove konstrukcije
- Postupak determinizacije je otuda znatno lakši nad automatom koji je dobijen Gluškovljevom konstrukcijom

Pregled

Minimizacija konačnih automata

Da li je PDKA za neki jezik jedinstven?

- Odgovor na ovo pitanje je NE
  - Vratiti se na primer sa slajda 31: dva različita postupka determinizacije (sa prethodnom eliminacijom ε-prelaza i bez nje) su nam dala dva različita PDKA
  - ova dva automata su imala različit broj stanja
- Ova činjenica otežava situaciju prilikom ispitivanja ekvivalentnosti regularnih izraza
  - to što smo za dva regularna izraza dobili različite PDKA i dalje ne znači da ta dva izraza predstavljaju različite jezike
- Ukoliko zahtevamo da broj stanja PDKA bude najmanji mogući, da li ćemo tada imati jedinstvenost?

Definicija 7

PDKA je minimalan (u oznaci MPDKA), ako ne postoji PDKA sa manjim brojem stanja koji prepoznaje isti jezik.

Minimizacija konačnih automata

Definicija 8

Jezik stanja $q \in Q$ *(u oznaci* L_q*) automata* $\mathcal{A} = (\Sigma, Q, i, F, \delta\}$ *definišemo kao skup svih reči koje su etikete nekog izračunavanja koje polazi iz stanja* q *i završava u nekom završnom stanju automata. Formalno:* $L_q = \{w \mid \exists c\ f.\ c : q \overset{w}{\Longrightarrow} f \wedge f \in F\}$.

Napomena

Kako je u PDKA izračunavanje sa etiketom w koje polazi iz q jedinstveno, važiće $w \in L_q$ akko je stanje f u koje se stiže tim jedinstvenim izračunavanjem završno.

Primedba

Ako je $i \in Q$ početno stanje PDKA $\mathcal{A}$, tada je $L_i = L(\mathcal{A})$.

Definicija 9

Stanja p *i* q *nekog automata su nerazlikujuća akko je* $L_p = L_q$.

Minimizacija konačnih automata

Nerazlikujuća stanja predstavljaju redundantna stanja u automatu

Svaka dva (ili više) nerazlikujuća stanja se mogu stopiti u jedno stanje, čime se broj stanja smanjuje

Definicija 10

Nerodova ekvivalencija je relacija ekvivalencije među stanjima automata definisana na sledeći način: $p \sim q \Leftrightarrow L_p = L_q$.

Važna napomena

Može se pokazati da je automat čija su stanja klase ekvivalencije relacije $\sim$ (tzv. količnički automat) PDKA sa najmanjim mogućim brojem stanja koji prepoznaje isti jezik kao i polazni automat:

- Ovaj automat je jedinstven do na izomorfizam
- Otuda imamo postupak odlučivanja o ekvivalentnosti dva regularna izraza:
  - primenimo Tompsonovu (ili Gluškovljevu) konstrukciju na oba regularna izraza
  - determinizujemo dobijene konačne automate
  - za oba automata odredimo Nerodovu ekvivalenciju $\sim$ i formiramo količničke automate
  - polazni izrazi su ekvivalentni akko su dobijeni MPDKA izomorfni
- Pored toga, minimizacija automata ima i praktični značaj (lakše se programira, zauzima manje memorije, i sl.)

Minimizacija konačnih automata

Kako odrediti relaciju $\sim$

Ideja je da se relacija $\sim$ formira iterativno:

- Neka je $L_q^{(k)} = \{w \mid w \in L_q \wedge |w| \leq k\}$
- Neka je $p \sim_k q \Leftrightarrow L_p^{(k)} = L_q^{(k)}$

Sada važe sledeća tvrđenja:

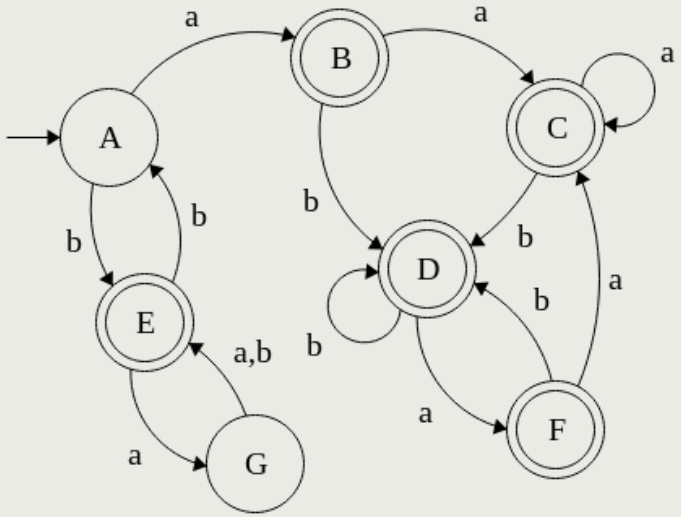
- $\forall k \in \mathbb{N}.\ p \sim q \Rightarrow p \sim_k q \Rightarrow p \sim_{k-1} q$
- $\forall k \in \mathbb{N}.\ \sim_{k-1} \supseteq \sim_k \supseteq \sim$
- dakle, imamo: $\sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \sim_3 \supseteq \ldots \supseteq \sim$
- $\sim = \bigcap_{k=0}^{\infty} \sim_k$
- $\exists k \in \mathbb{N}.\ \sim_k = \sim_{k+1}$
- ako za neko k važi da je $\sim_k = \sim_{k+1}$, tada će važiti $\sim = \sim_k$
- $p \sim_0 q$ akko su ili oba ova stanja završna ili oba nezavršna
- $p \sim_1 q \Leftrightarrow p \sim_0 q \wedge (\forall a \in \Sigma.\ \delta(p, a) \sim_0 \delta(q, a))$
- $p \sim_2 q \Leftrightarrow p \sim_1 q \wedge (\forall a \in \Sigma.\ \delta(p, a) \sim_1 \delta(q, a))$
- $p \sim_k q \Leftrightarrow p \sim_{k-1} q \wedge (\forall a \in \Sigma.\ \delta(p, a) \sim_{k-1} \delta(q, a))$ (indukcijom)

Postupak minimizacije se sada svodi na iterativno određivanje relacija $\sim_0, \sim_1, \sim_2, \ldots$ dokle god ima promena, tj. dokle god je $\sim_k \neq \sim_{k+1}$. Kada to više nije slučaj, dobijena relacija je upravo Nerodova ekvivalencija $\sim$. Opisani postupak poznat je i kao Murov algoritam.

Minimizacija konačnih automata

Primer

Dat je automat na slici:



Formirajmo $\sim_k$ *relacije:*

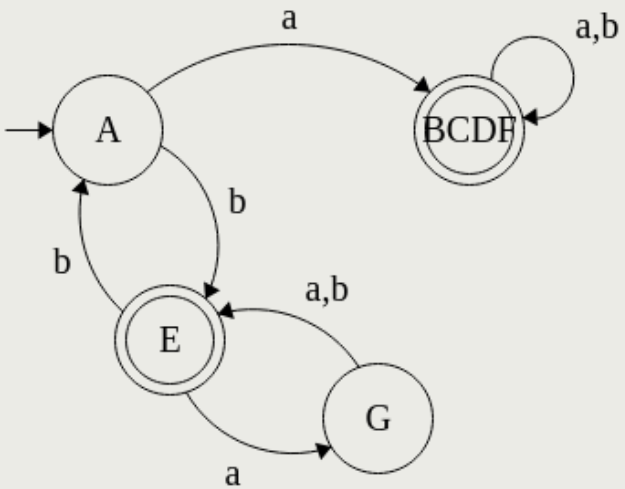
$\sim_0$:	$\{A, G\}$	$\{B, C, D, E, F\}$
$\sim_1$:	$\{A, G\}$	$\{E\}, \{B, C, D, F\}$
$\sim_2$:	$\{A\}, \{G\}$	$\{E\}, \{B, C, D, F\}$
$\sim_3$:	$\{A\}, \{G\}$	$\{E\}, \{B, C, D, F\}$

Kako je $\sim_2 = \sim_3$, *sledi da je* $\sim = \sim_2$.

Minimizacija konačnih automata

Primer

Koristeći klase ekvivalencije dobijene relacije $\sim$ kao stanja, dobijamo sledeći količnički automat:

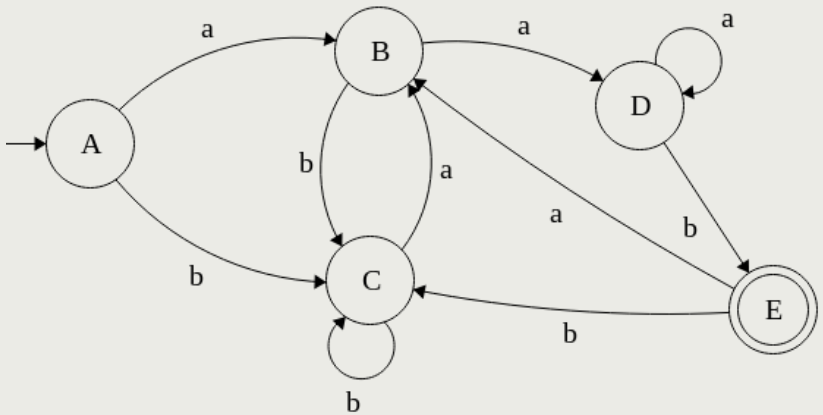


Ovaj automat je MPDKA ekvivalentan polaznom. Početno stanje ovog automata odgovara klasi ekvivalencije kojoj pripada početno stanje polaznog automata. Završna stanja ovog automata su sva stanja koja se sastoje iz završnih stanja polaznog automata.

Minimizacija konačnih automata

Primer

Dat je automat na slici:



Formirajmo $\sim_k$ relacije:

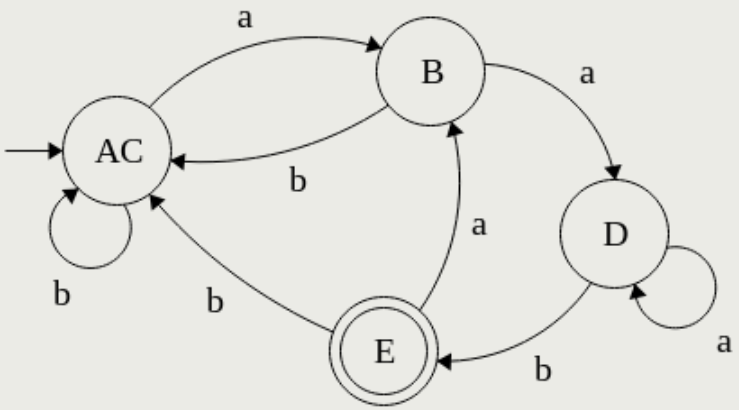
$$\begin{array}{lc|c} \sim_0: & \{A,B,C,D\} & \{E\} \\ \sim_1: & \{A,B,C\},\{D\} & \{E\} \\ \sim_2: & \{A,C\},\{B\},\{D\} & \{E\} \\ \sim_3: & \{A,C\},\{B\},\{D\} & \{E\} \end{array}$$

Kako je $\sim_2 = \sim_3$, sledi da je $\sim = \sim_2$.

Minimizacija konačnih automata

Primer

Koristeći klase ekvivalencije dobijene relacije $\sim$ kao stanja, dobijamo sledeći količnički automat:

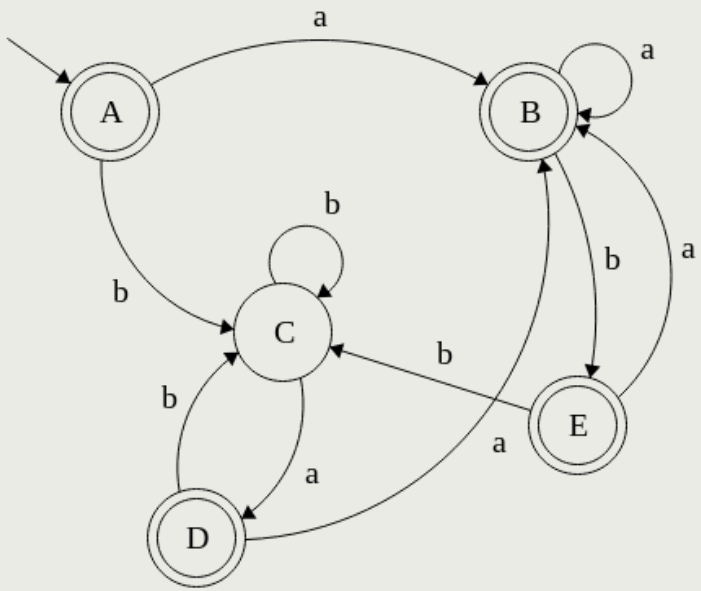


Ovaj automat je MPDKA ekvivalentan polaznom.

Minimizacija konačnih automata

Primer

Dat je automat na slici:



Formirajmo $\sim_k$ *relacije:*

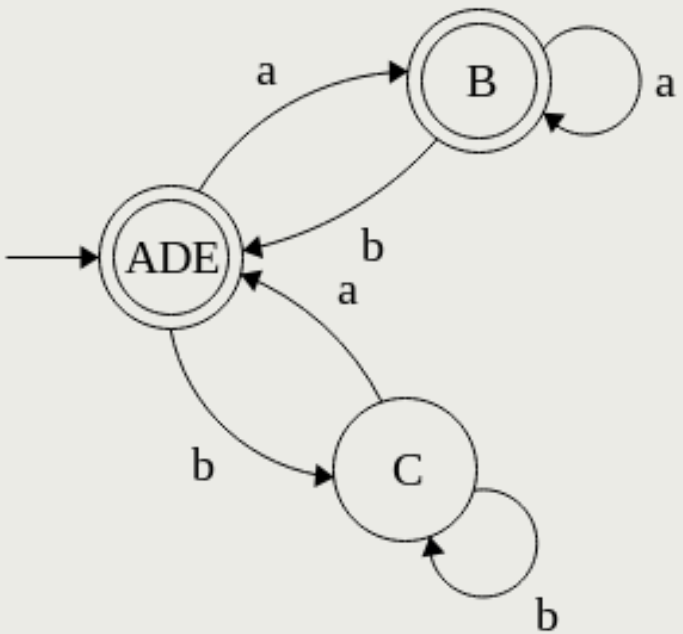
$$\begin{array}{lc|c} \sim_0: & \{C\} & \{A, B, D, E\} \\ \sim_1: & \{C\} & \{B\}, \{A, D, E\} \\ \sim_2: & \{C\} & \{B\}, \{A, D, E\} \end{array}$$

Kako je $\sim_1 = \sim_2$*, sledi da je* $\sim = \sim_1$.

Minimizacija konačnih automata

Primer

Koristeći klase ekvivalencije dobijene relacije $\sim$ kao stanja, dobijamo sledeći količnički automat:

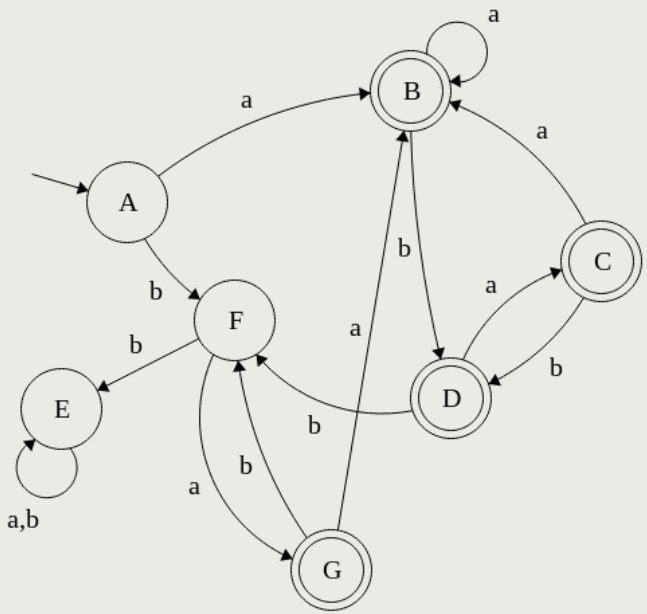


Ovaj automat je MPDKA ekvivalentan polaznom.

Minimizacija konačnih automata

Primer

Dat je automat na slici:



Formirajmo $\sim_k$ relacije:

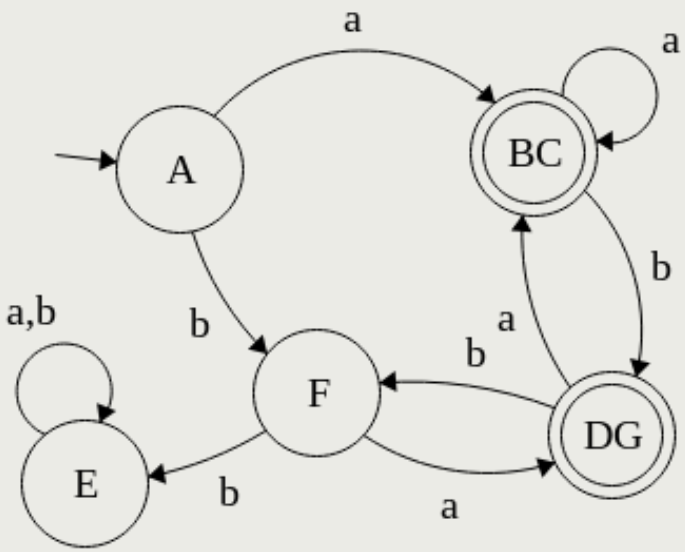
$\sim_0$:	$\{A, E, F\}$	$\{B, C, D, G\}$
$\sim_1$:	$\{A, F\}, \{E\}$	$\{B, C\}, \{D, G\}$
$\sim_2$:	$\{A\}, \{F\}, \{E\}$	$\{B, C\}, \{D, G\}$
$\sim_3$:	$\{A\}, \{F\}, \{E\}$	$\{B, C\}, \{D, G\}$

Kako je $\sim_2 = \sim_3$, sledi da je $\sim = \sim_2$.

Minimizacija konačnih automata

Primer

Koristeći klase ekvivalencije dobijene relacije $\sim$ kao stanja, dobijamo sledeći količnički automat:



Ovaj automat je MPDKA ekvivalentan polaznom.

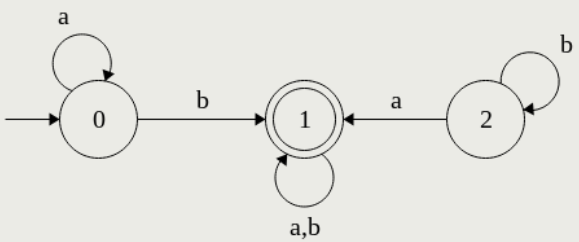
Minimizacija konačnih automata

Važna napomena

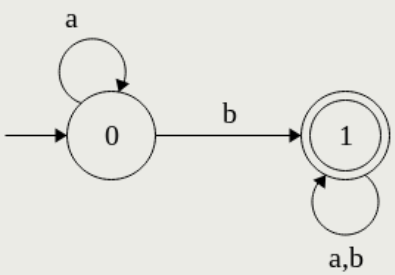
- Murov algoritam ne uklanja nedostižna stanja, ukoliko postoje
- Otuda je pre postupka minimizacije potrebno potkresati automat, da bi rezultat bio ispravan

Primer

Posmatrajmo automat:



Lako se vidi da u ovom automatu nema nerazlikujućih stanja. Ipak, on nije minimalni PDKA, jer i automat:



prepoznaje isti jezik.

Pregled

Konstrukcija regularnog izraza za dati automat

Da li je svaki prepoznatljiv jezik regularan?

- Ranije smo ustanovili da je svaki regularan jezik prepoznatljiv ($R(\Sigma) \subseteq P(\Sigma)$)
- Da li važi obrnuta inkluzija: $P(\Sigma) \subseteq R(\Sigma)$?
- Drugim rečima, da li za svaki konačni automat postoji regularni izraz koji opisuje isti jezik?
- Odgovor na ovo pitanje je DA
- Dokaz ove činjenice sledi iz postojanja algoritama koji konstruišu regularni izraz na osnovu automata:
  - Metod eliminacije stanja (izložen u nastavku)
  - Algoritam zasnovan na sistemu jednačina pridruženih automatu
  - Algoritam Mek Notona i Jamade

Metod eliminacije stanja

Definicija 11

Konačni automat kod koga etikete prelaza mogu biti proizvoljni regularni izrazi nad Σ (a ne samo simboli iz Σ i ε) nazivamo uopšteni konačni automat.

Metod eliminacije stanja

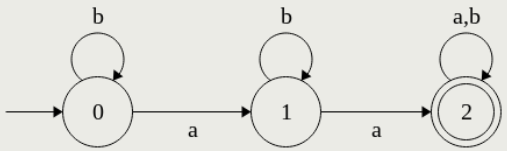
Neka je $\mathcal{A}$ proizvoljan PDKA. Posmatrajmo $\mathcal{A}$ kao uopšteni konačni automat. Označimo sa R_{pq} regularni izraz kojim je etiketiran prelaz od p do q (ako postoji). Na ovaj automat primenjujemo sledeći postupak:

- Dodajemo jedno novo početno stanje i i jedno novo završno stanje f (ovo postaju jedino početno, odnosno završno stanje)
  - dodajemo luk od i do (starog) početnog stanja automata $\mathcal{A}$ sa etiketom ε
  - dodajemo lukove od svih starih završnih stanja automata $\mathcal{A}$ do stanja f sa etiketom ε
- Eliminišemo jedno po jedno stanje automata (izuzev i i f) u proizvoljnom poretku na sledeći način:
  - označimo stanje koje želimo da eliminišemo sa q
  - neka su p i r dva stanja različita od q takva da postoji lukovi $p \xrightarrow{R_{pq}} q$ i $q \xrightarrow{R_{qr}} r$
  - ova dva luka zamenjujemo novim lukom $p \xrightarrow{R_{pq}R_{qq}^{*}R_{qr} \mid R_{pr}} r$
  - opciono, ako neki od lukova $q \xrightarrow{R_{qq}} q$ (petlja) i $p \xrightarrow{R_{pr}} r$ ne postoji, odgovarajući deo izraza se izostavlja
  - ovo uradimo za svaka takva dva stanja p i r (ne obavezno međusobno različita)
- Kada ostanu samo stanja i i f, luk koji ih spaja biće etiketiran regularnim izrazom R_{if} koji opisuje jezik polaznog automata

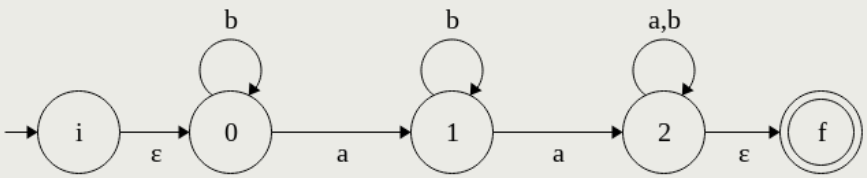
Metod eliminacije stanja

Primer

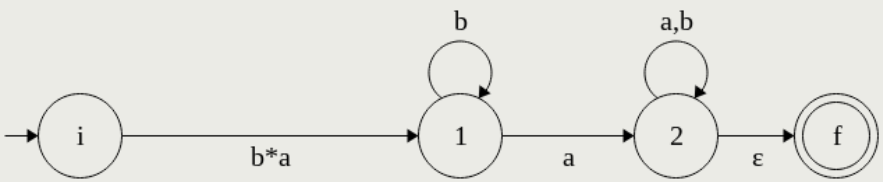
Posmatrajmo automat na slici:



Uvedimo najpre nova početna i završna stanja:



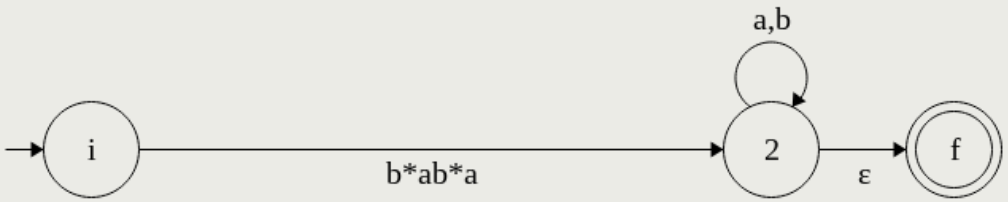
Zatim eliminišemo stanje 0*, tako što formiramo novi luk od* i *do* 1*:*



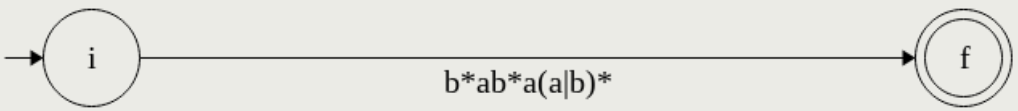
Metod eliminacije stanja

Primer

(nastavak) Zatim eliminišemo stanje 1 *tako što formiramo luk od* i *do* 2



Najzad eliminišemo i stanje 2*, uvođenjem luka od* i *do* f*:*

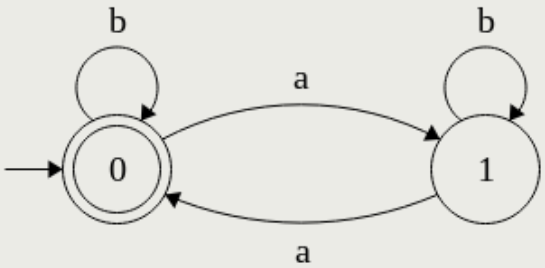


Regularni izraz nad ovim lukom opisuje jezik polaznog automata.

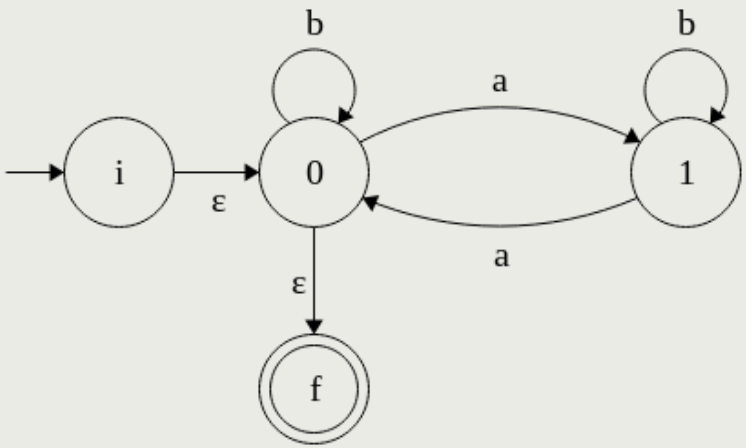
Metod eliminacije stanja

Primer

Posmatrajmo automat na slici:



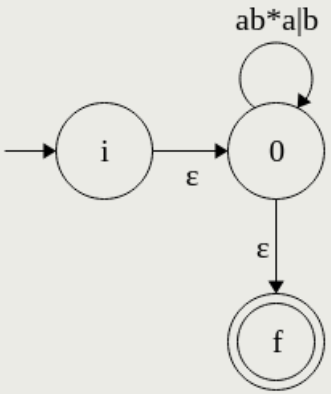
Uvedimo najpre novo početno stanje i i novo završno stanje f :



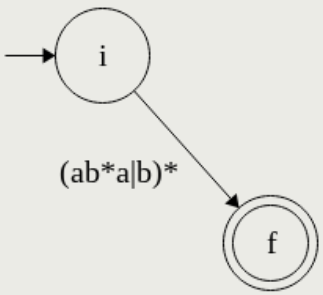
Metod eliminacije stanja

Primer

(nastavak) Sada eliminišemo stanje 1*:*



a zatim i stanje 0*:*



Dobijeni regularni izraz opisuje jezik automata.

Metod eliminacije stanja

Napomena

- Iako je teorijski potpuno svejedno u kom poretku se eliminišu stanja, dobijeni regularni izraz će zavisiti od izabranog poretka
  - Setimo se da možemo imati različite regularne izraze koji opisuju isti jezik
- U praksi, izborom ,,pravog" poretka dobijamo jednostavniji regularni izraz
  - Ne postoji egzaktan metod koji nam određuje poredak koji daje najjednostavniji izraz
  - Dobra heuristika: eliminišemo prvo stanje koje zahteva dodavanje najmanjeg broja novih lukova
  - Zadatak: pokušajte da u prethodnom primeru prvo eliminišete stanje 0

Metod eliminacije stanja

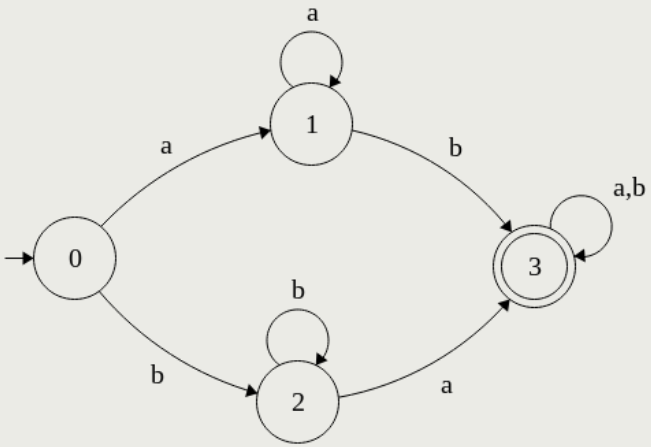
Napomena

- Novo početno stanje i je neophodno uvoditi da bismo postigli da ni jedan luk ne ulazi u početno stanje automata
  - Ako je to slučaj već u polaznom automatu, tada se stanje i ne mora uvoditi, što pojednostavljuje postupak
- Slično, novo završno stanje f se uvodi da bismo postigli da imamo tačno jedno završno stanje iz koga ne izlazi ni jedan luk
  - ako je ovo ispunjeno u polaznom automatu, onda stanje f ne moramo uvoditi

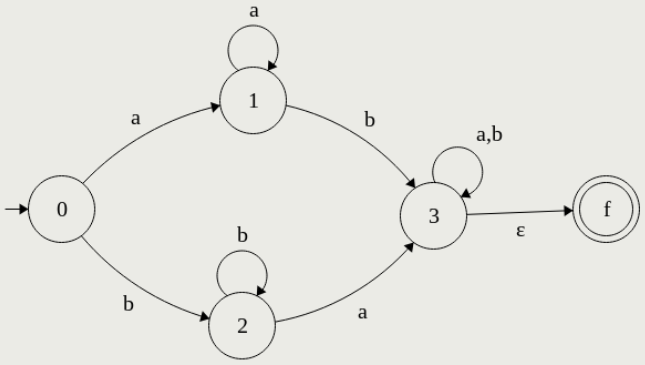
Metod eliminacije stanja

Primer

Posmatrajmo automat na slici:



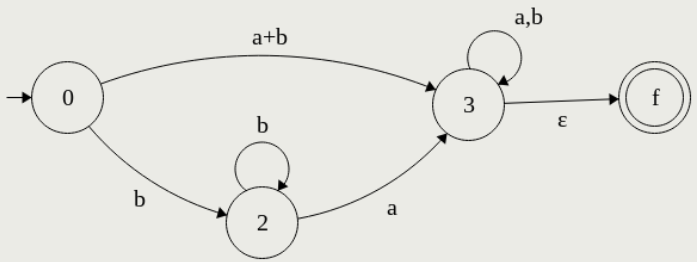
Uvedimo novo završno stanje f (novo početno stanje nije neophodno, jer u 0 *ne ulazi ni jedan luk):*



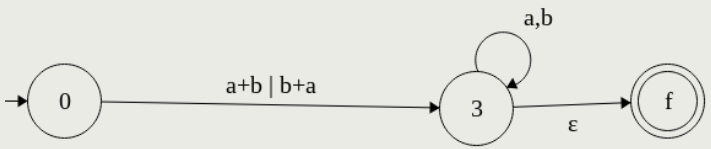
Metod eliminacije stanja

Primer

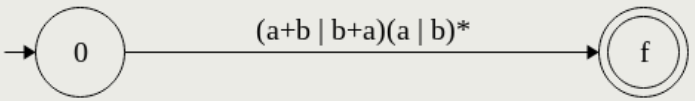
(nastavak) Sada eliminišemo stanje 1*:*



a zatim i stanje 2*:*



Najzad, eliminišemo i stanje 3*:*

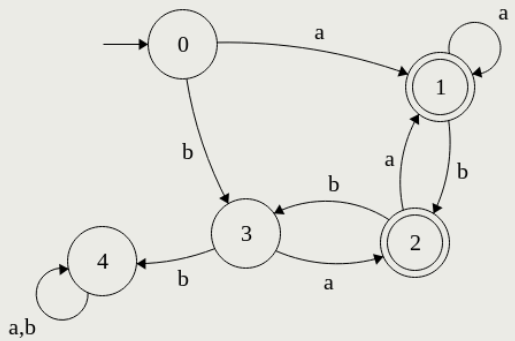


Dobijeni regularni izraz opisuje jezik automata.

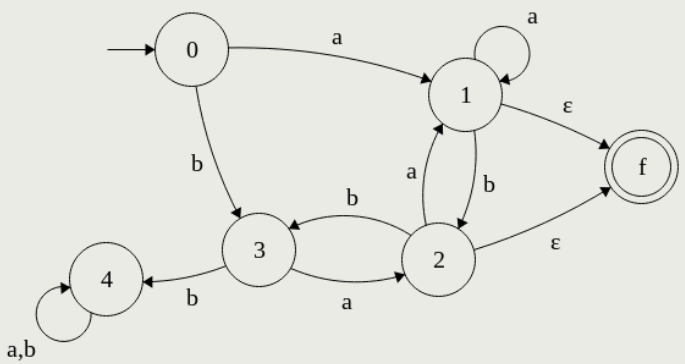
Metod eliminacije stanja

Primer

Posmatrajmo automat na slici:



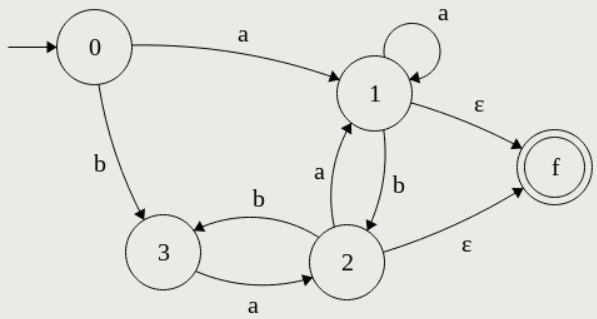
Uvedimo novo završno stanje f (novo početno stanje nije neophodno, jer u 0 ne ulazi ni jedan luk):



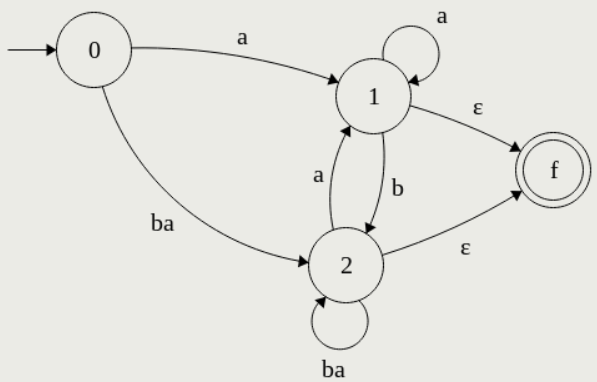
Metod eliminacije stanja

Primer

(nastavak) Sada eliminišemo stanje 4 *(„stanje greške"):*



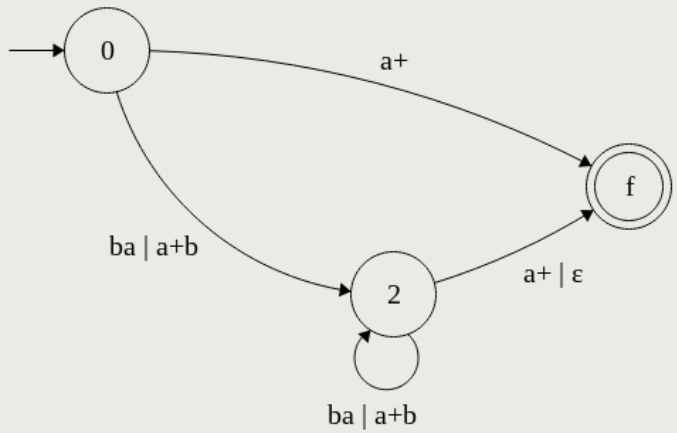
a zatim i stanje 3*:*



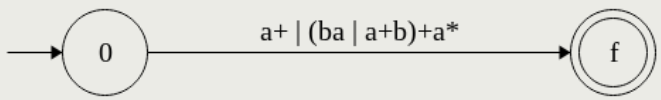
Metod eliminacije stanja

Primer

(nastavak) Zatim eliminišemo stanje 1*:*



a zatim i stanje 2*:*



Dobijeni regularni izraz opisuje jezik automata.

Klinijeva teorema

Teorema 12

Klase regularnih i prepoznatljivih jezika se poklapaju, tj.: $R(\Sigma) = P(\Sigma)$.

Dokaz

Implikacija $R(\Sigma) \subseteq P(\Sigma)$ *je dokazana ranije (npr. Tompsonova konstrukcija). Metod eliminacije stanja je dokaz da važi i drugi smer* $P(\Sigma) \subseteq R(\Sigma)$. *Otuda važi gornja skupovna jednakost.*

Napomena

Ova teorema poznata je i kao Klinijeva teorema i najznačajnija je teorema u teoriji regularnih jezika i konačnih automata.

Jedan zanimljiv primer

Šta je starije, automat ili izraz?

- U većini slučajeva nam je zgodnije da najpre jezik opišemo regularnim izrazom, pa da za njega konstruišemo automat
- Ipak, ponekad nije tako lako opisati jezik regularnim izrazom, ali je lako napraviti automat koji ga prepoznaje
- Tada možemo napraviti automat, a onda metodom eliminacije stanja dobiti regularni izraz (ako nam je uopšte potreban)

Jedan zanimljiv primer

Primer

Želimo da regularnim izrazom opišemo jezik svih binarnih brojeva koji su deljivi sa 3. *Da bismo to uradili, posmatraćemo na koji se način menja ostatak pri deljenju sa* 3 *kada na neki binarni broj dopišemo cifru* 0 *ili* 1. *Označimo sa* w *tekuću binarnu reč. Neka je* $w0$ *odnosno* $w1$ *reč koja se dobija dopisivanjem* 0 *odnosno* 1 *na* w. *Označimo sa* x *binarni broj koji je predstavljen zapisom* w. *Tada zapis* $w0$ *predstavlja broj* $2x$, *a* $w1$ *predstavlja broj* $2x + 1$. *Sada važi:*

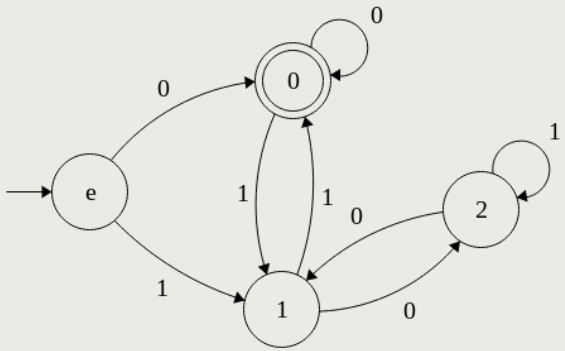
- *Ako je* $x \mod 3 = 0$, *tada je* $2x \mod 3 = 0$, *a* $2x + 1 \mod 3 = 1$
- *Ako je* $x \mod 3 = 1$, *tada je* $2x \mod 3 = 2$, *a* $2x + 1 \mod 3 = 0$
- *Ako je* $x \mod 3 = 2$, *tada je* $2x \mod 3 = 1$, *a* $2x + 1 \mod 3 = 2$

Imajući ovo u vidu, formiramo automat (na sledećem slajdu).

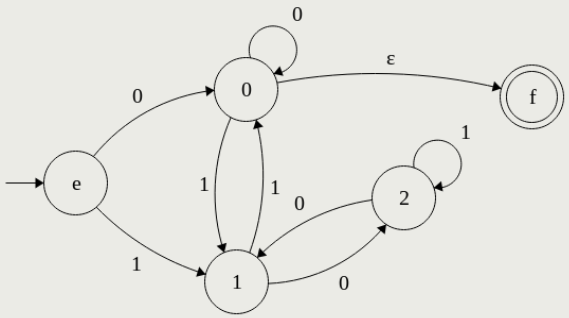
Jedan zanimljiv primer

Primer

(nastavak) Traženi automat je:



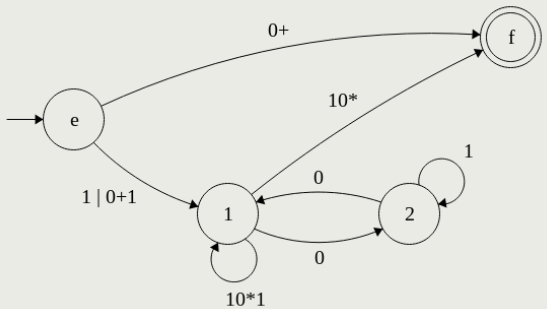
Da bismo dobili regularni izraz na osnovu ovog automata, dodajemo najpre novo završno stanje:



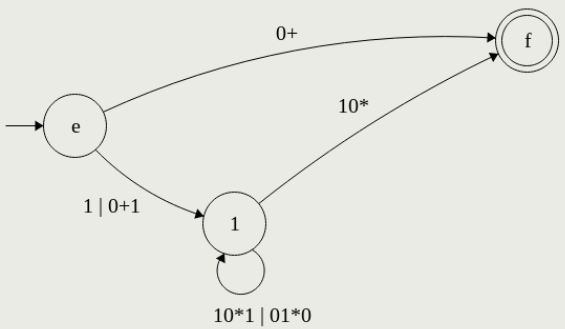
Jedan zanimljiv primer

Primer

(nastavak) Sada eliminišemo stanje 0



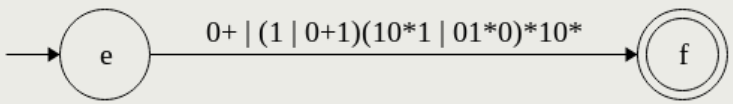
a zatim i stanje 2*:*



Jedan zanimljiv primer

Primer

(nastavak) Na kraju eliminišemo stanje 1*:*



Dobijeni regularni izraz opisuje traženi jezik.

Pregled

Lema o razrastanju

Setimo se leme o razrastanju koju smo ranije naveli bez dokaza:

Lema 1

Neka je L regularan jezik. Tada postoji neko $p \in \mathbb{N}$ (koje zavisi samo od jezika L), takvo da za svaku reč $w \in L$ za koju je $|w| \geq p$ važi da se w može predstaviti u obliku $w = xzy$, gde je $|z| \geq 1$, $|xz| \leq p$ i $xz^k y \in L$ za svako $k \in \mathbb{N}_0$.

Sada kada poznajemo konačne automate, možemo da dokažemo ovu lemu sasvim jednostavno.

Lema o razrastanju

Dokaz

Neka je $\mathcal{A}$ jedinstveni MPDKA koji prepoznaje dati regularni jezik L. Neka je p broj stanja ovog automata i neka je w bilo koja reč jezika L dužine bar p. Postoji jedinstveno izračunavanje $c : s \stackrel{w}{\Longrightarrow} f$, gde je s početno, a f jedno od završnih stanja automata $\mathcal{A}$. Kako je reč w dužine bar p, ovo izračunavanje će bar dva puta proći kroz jedno isto stanje, tj. izračunavanje c mora biti oblika: $s \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \dots \xrightarrow{a_i} q_i \xrightarrow{a_{i+1}} q_{i+1} \xrightarrow{a_{i+2}} \dots \xrightarrow{a_j} q_i \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} \dots \xrightarrow{a_n} f$, gde je $w = a_1 a_2 \dots a_n$. Pritom, prvo ponavljanje stanja mora biti nakon najviše p koraka, tj. $j \leq p$. Sada možemo uzeti $x = a_1 \dots a_i$, $z = a_{i+1} \dots a_j$ i $y = a_{j+1} \dots a_n$. Za ovakvu podelu reči važiće $|z| \geq 1$ i $|xz| = j \leq p$. Pritom, reč z automat vodi od stanja q_i ponovo do stanja q_i, pa se ova reč može na tom mestu ponavljati 0 ili više puta, ne remeteći ostatak izračunavanja. Otuda će svaka reč oblika $xz^k y$ biti u jeziku L ($k = 0, 1, 2 \dots$).