

Osnovne metode dokazovanja

Dokazovanje s konstrukcijo

Dokaz sestoji iz tega, da vedno skonstruiramo pričo temu, da ima nek matematični model določene lastnosti. Direktno najdemo tisto lastnost o kateri govoriti naša trditev oz. izjava v matematičnem jeziku.

Naj bosta a in b dve zaporedni celi števili. Dokaži, da je $a + b$ liho število.

Naj bo $b = a + 1$, torej $a + b = a + a + 1 = 2a + 1$, kar je oblike $2n + 1$, se pravi liho število.

Dokaži, da je mogoče vsako naravno število zapisati kot $2^k n$, kjer je k nenegativno število in n liho število.

$$\begin{aligned} \forall x \in \mathbb{N} : x &= 2^k \cdot n & x \text{ je liho} \Leftrightarrow x = 2^k \cdot n = 2^0 \cdot n = 1 \cdot n \\ k &> 0 & n \\ n &= 2m + 1, k \in \mathbb{N} & x \text{ je sodo} : x = 2^k \cdot n = 2^1 \cdot n = 2n \\ & & 2 \cdot 2 \cdot \frac{k}{n} \\ & & \text{sodo število lahko deli} \neq 2, \text{ če kvocient na meseč deliti} \neq 2 \rightarrow \\ & & \frac{k}{2} \text{ liho število} \\ & & x = 2^1 \left(\frac{x}{2} \right) \text{ liho število} \end{aligned}$$

Če je m liho število, ga lahko preprosto zapišemo kot $2^0 m (k = 0)$.

Če je sodo, naj bo 2^k največja potenca števila 2, ki deli m , $m = 2^k n$. Če bi bilo n sodo število, bi tudi 2^{k+1} delilo m , kar je v nasprotju s predpostavko, da je 2^k največja potenca števila 2, ki deli m . Torej je n liho število.

Na neki zabavi je 25 ljudi. Za vsako trojico ljudi velja, da se vsaj dva v tej trojici poznata. Dokaži, da na zabavi obstaja oseba, ki pozna vsaj 12 ljudi.

Če 25 ljudi spravimo v 12 škatel po 2, nam ostane ena oseba X. Če z osebo X običemo katerokoli izmed 12 škatel, dobimo trojico in lahko uporabimo dejstvo, da se vsaj dva v tej škatli poznata.

Če smo z osebo X obiskali vsako škatlo in je bila v vsakem primeru oseba X ena izmed vsaj dveh oseb, ki se poznata, je oseba X tista oseba, ki pozna vsaj 12 ljudi.

V nasprotnem primeru pa se poznata samo osebi, ki sta že bili v škatli. V tem primeru zamenjamo osebo X in eno izmed teh dveh oseb in izločimo škatlo ter nadaljujemo obiskovanje ostalih škatel z novo osebo X. Zakaj lahko škatlo izločimo? Ker smo pridelali škatlo z dvema osebama, ki se ne poznata, kar pomeni, da se bo katerakoli tretja oseba, ki jo dodamo v to škatlo, poznala z vsaj eno osebo od teh dveh (torej, četudi vmes poljubnokrat zamenjamo osebo X, vemo, da se gotovo pozna z vsaj eno osebo v vsaki od izločenih škatel).

Ko po tem postopku pregledamo vse škatle, pridemo do osebe X, ki se pozna z vsaj eno osebo iz vsake neizločene in izločene škatle, se pravi osebe, ki pozna vsaj 12 ljudi.

Pokaži, da v vsakem grafu obstajata vozlišči enake stopnje.

N je število vozlišč. Največja stopnja vozlišča je torej $n - 1$ (ciklov ne upoštevamo). Vozlišča razporedimo v škatle glede na njihovo stopnjo; škatel je torej n (od 0 do $n - 1$). Opazimo, da škatli 0 in $n - 1$ ne moreta biti hkrati zapolnjeni, kajti če je škatla $n - 1$ zapolnjena, obstaja vozlišče, ki je povezano z vsakim drugim vozliščem, torej je stopnja vsakega vozlišča vsaj 1 in obratno. Razporejujemo n vozlišč v največ $n - 1$ škatel, kar pomeni, da morata v eni škatli biti vsaj dve vozlišči, se pravi vsaj dve vozlišči morata imeti isto stopnjo.

Pokaži, da vsak graf G, za katerega je $\delta(G) \geq 2$, premore pot dolžine vsaj $\delta(G)$ ter cikel dolžine vsaj $\delta(G) + 1$.

($\delta(G)$ je minimalna stopnja vozlišč v grafu.) Konstruirajmo pot dolžine $\delta(G)$. Recimo, da smo že konstruirali pot v_1, \dots, v_k dolžine $k < \delta(G)$. Vozlišče v_k ima vsaj $\delta(G) > k$ sosedov, torej lahko izberemo soseda, ki še ni vsebovan v poti. Tako pridemo do poti dolžine $k = \delta(G)$.

Sedaj pa iz te poti pridelajmo še cikel. Če je v_1 sosed v_k , lahko pot sklenemo v cikel dolžine $\delta(G) + 1$.

V nasprotnem primeru pa nadaljujemo pot. Tokrat se osredotočimo samo na zadnjih $k - 1$ vozlišč na poti in podobno sklepamo, da ima v_k vsaj enega soseda, ki ni eden izmed teh vozlišč. Po predpostavki tudi noben sosed ni v_1 . Torej dobimo pot v_1, \dots, v_{k+1} . Če ima v_{k+1} za soseda v_1 ali v_2 , lahko sklenemo cikel dolžine $k + 2$ oziroma $k + 1$. V nasprotnem primeru pa proces ponovimo. Ker imamo končno število vozlišč, se bo ta proces gotovo ustavil.

Dokaži, da vsaka podmnožica S, {1, 2, 3, ..., 10} velikosti šest vsebuje par števil, katerih vsota je 10.

V množici S imamo štiri pare različnih števil, katerih vsota je 10 ($1 + 9, 2 + 8, 3 + 7, 4 + 6$), število 5, ki v paru s seboj da 10, ter število 10, ki ne nastopa v nobenem paru, katerega vsota je 10.

Imamo sledeče škatle: {1, 9}, {2, 8}, {3, 7}, {4, 6}, {5, 10}. Ko izberemo šest števil, moramo vsaj v eno škatlo spraviti dve števili, se pravi imamo ali obe števili v škatli z dvema različnima številoma, katerih vsota je 10, oziroma 5 in 10 v zadnji škatli (od tod potrebujemo samo 5, da dobimo $5 + 5 = 10$).

Pokaži, da za vsak algoritem za stiskanje brez izgub informacije obstajajo vhodi, ki jih algoritem ne zmanjša. Nadalje, pokaži, da če obstajajo vhodi, ki jih algoritem zmanjša, tedaj obstaja tudi vhod, ki ga algoritem poveča. Brez škode predpostavimo, da algoritem na vhod jemlje nize bitov končne dolžine.

Naj algoritem na vhod jemlje nize bitov dolžine n. Recimo, da algoritem vse nize zmanjša vsaj na dolžino $m < n$. Ampak če razporejujemo 2^n vhodnih nizov v 2^m škatel (ki predstavljajo kompresirane nize), moramo imeti v eni škatli vsaj dva niza, torej za določen kompresiran niz ne vemo, kateremu vhodnemu nizu pripada, se pravi pride do izgube informacije.

Sedaj pa recimo, da vsaj en niz zmanjšamo iz n na m bitov, vseh ostalih nizov pa ne povečamo. Vendar zahteva, da nobenih nizov ne povečamo, moramo upoštevati tudi za nize dolžine m. Ker nočemo, da se ti nizi povečajo, jih mora kompresijski algoritem slikati v množico nizov dolžine m, vendar v to množico se slika že vsaj en niz dolžine n, kar pomeni, da se mora vsaj en niz dolžine m povečati (na primer vsebovati še dodaten bit, ki pove, da to ni zgoščena vrednost niza dolžine n), drugače izgubimo informacijo.

Pokaži, da vsak graf $G = (V, E)$ premore prerez velikosti vsaj $|E|/2$.

Naj bo S_i, T_i naključna particija vozlišč grafa, kjer je vsako vozlišče z verjetnostjo $1/2$ vsebovano v S_i .

Verjetnost, da je poljubna povezava del prereza je $1/2$ (ker so vozlišča enakomerno razporejena med S_i in T_i , je verjetnost, da je končno vozlišče na drugi strani prereza kot začetno enaka $1/2$).

Pričakovano število (med vsemi particijami) povezav, ki pripadajo prerezu, je torej $|E|/2$, kar pomeni, da mora obstajati vsaj ena particija z $\geq |E|/2$ povezavami, ki pripadajo prerezu.

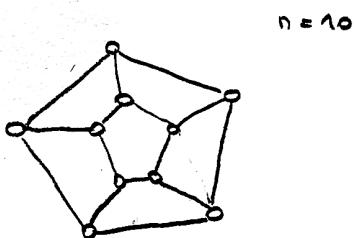
Za vsako sodo naravno število velja, da obstaja nek graf G za katerega velja, da ima vsako vozlišče tega grafa stopnjo 3. Regularni grafi - grafi, ki imajo enako število sosedov.

Pokazati moramo, kako lahko za poljuben n konstruiramo en tak graf.

Kreiramo n vozlišč in razbijemo to množico na 2 kosa (npr. 5 in 5). Na vsaki množici naredimo cikel. Vsako vozlišče iz notranjega cikla povežemo z vsakim vozliščem iz zunanjega cikla.

S tem smo dokazali, da vedno obstaja nek graf G s to lastnostjo.

$$\forall n \in \mathbb{N}, n = 2k : \exists G = \langle V, E \rangle, \forall v \in V, \deg(v) = 3$$



Dokazovanje z indukcijo

Matematična indukcija je zelo uporaben način za dokazovanje lastnosti zelo raznovrstnih množic. Če lahko neko množico zapišemo z induktivno (rekurzivno) definicijo, potem lahko poskusimo določeno lastnost te množice pokazati s pomočjo matematične indukcije.

Množice podamo induktivno tako, da najprej definiramo množice njenostavnejših elementov (t.i. bazo), nato pa definiramo pravila, s katerimi iz poljubnih elementov množice skonstruiramo nove elemente.

Induktivna definicija naravnih števil.

Baza: $1 \in \mathbb{N}$

Konstrukcija: $k \in \mathbb{N}$, potem tudi $k + 1 \in \mathbb{N}$

Množica praštevil P.

Baza: $2 \in P$

Konstrukcija: $k \in P \wedge (\forall l \in P, l < k, k \bmod l \neq 0)$, potem tudi $k \in P$

Dokaži, da bi lahko izplačal vsako vsoto denarja ≥ 8 centov, samo s kovanci 3 in 5 centov.

Dokazujemo lastnost množice naravnih števil, ki so včja ali enaka 8. Za ta števila dokazujemo, da jih lahko zapišemo kot vsoto večkratnikov števila 3 in števila 5.

1. Baza je za to množico število 8, zato najprej dokažemo, da ima to število želeno lastnost.

$$8 = 3 + 5$$

2. Sedaj pa pokažimo, da pravilo za kreiranje novega elementa v podani množici, lastnosti ne spremeni.

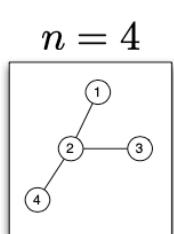
Število n naj bo že element množice, za katerega vemo, da ga lahko zapišemo kot:

$n = 3x + 5y$ (x je število kovancev za 3 cente, y je število kovancev za 5 centov). Če številu n dodamo 1 (kreiramo nov element množice), lahko iz informacije $n = 3x + 5y$ sestavimo tudi število $(n + 1)$ kot vsoto večkratnikov števila 3 in 5.

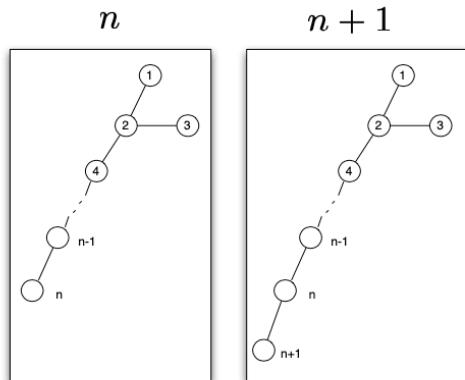
- $y > 0$: $n + 1 = 3(x + 2) + 5(y - 1)$ - odvzamemo en kovanec za 5 centov in odvzamemo 2 kovanca za 3 cente
- $x > 2$: $n + 1 = 3(x - 3) + 5(y + 2)$ - odvzamemo 3 kovance za 3 cente in odvzamemo 2 kovanca za 5 centov

Dokaži, da za $\forall n \geq 4$ obstaja dvojiško drevo z n vozlišči, od katerih so natanko 3 vozlišča listi.

Najprej pokažemo, da trditev velja za $n = 4$. Nato pokažemo, da lahko drevesu, za katerega velja zgornja lastnost, dodamo novo vozlišče, ne da bi povečali število listov. Slika 2.2 prikazuje korak izgradnje takega drevesa.



Slika 2.1: Drevo s tremi listi za $n = 4$



Slika 2.2: Induktivni korak

Dokaži, da $2^{|A|} = |2^A|$.

1. $|A| = 0$, potenčna množica prazne množice je $2^A = \{\emptyset\}$
2. $B = A \cup \{x\}$, $x \notin A$ induktivna predpostavka: $2^{|A|} = |2^A|$
 $|2^B| = |2^{A \cup \{x\}}| = |2^A| + |2^A| = 2 \cdot |2^A| = 2 \cdot 2^{|A|} = 2^{|A|+1} = 2^B$

Če je $A = \{1, 2, 3\}$, potem je $2^A = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$.

Indukcija.

$n = 0$: $2^{\emptyset} = \{\emptyset\}$. Moč te množice je 1, kar je enako 2^0 .

Induktivni korak: predpostavimo, da trditev velja za n in pokažimo, da velja tudi za $n + 1$.

Predpostavimo, da za množico $A = \{a_1, a_2, \dots, a_n\}$ velja $|2^A| = 2^{|A|}$.

Preverimo, kako je z množico $B = \{a_1, a_2, \dots, a_n, a_{n+1}\}$. Množica 2^B je sestavljena iz množic v množici 2^A (teh je po predpostavki $2^{|A|}$) in iz množic, ki jih dobimo tako, da vsaki množici v množici 2^A dodamo še element a_{n+1} (teh je prav tako $2^{|A|}$). Skupno število množic v množici 2^B je potem takem

$$2^{|A|} + 2^{|A|} = 2^{|A|+1} = 2^{|B|}.$$

Dokazovanje s protislovjem

Zelo pogosta tehnika dokazovanja izrekov je dokazovanje s protislovjem. Pri tej tehniki predpostavimo negacijo osnovnega izreka in pokažemo, da s tem pridemo v protislovje, kar pomeni, da mora izrek držati.

Dokaži, da obstaja neskončno število praštevil.

Predpostavimo, da obstaja samo končno število praštevil, recimo, da je največje praštevilo enako p_n . Sestavimo novo ševelo $q = p_1 p_2 p_3 \dots p_n + 1$. Ostanek po deljenju z vsemi praštevili (potencialnimi faktorji) je enak 1. Zato bi veljalo, da je tudi q praštevilo. S tem smo prišli v nasprotje s predpostavko, da je p_n največje praštevilo.

Dokaži, da ne obstaja pozitivna celoštevilska rešitev diofantske enačbe $x^2 - y^2 = 1$.

Recimo, da obstajata celoštevilska pozitivna x in y , ki rešita zgornjo enačbo.

$$x^2 - y^2 = (x + y)(x - y) = 1$$

Ker sta x in y celi števili, mora veljati:

1. $(x + y) = 1$ in $(x - y) = 1$, ali
2. $(x + y) = -1$ in $(x - y) = -1$.

V prvem primeru je rešitev $x = 1$, $y = 0$, v drugem primeru pa je $x = -1$, $y = 0$. V obeh primerih pridemo v protislovje s predpostavko, da sta x in y pozitivni celi števili.

Dokaži, da je $\sqrt[3]{2}$ iracionalno število.

Protislovje. Recimo, da je racionalno. To pomeni, da obstajata a in b , tako, da je število enako a/b in da velja $\gcd(a, b) = 1$. To pomeni, da lahko zapišemo $a^3 = 2b^3$.

Je lahko b sod? Ne. Torej je b lih (*). Število a je v vsakem primeru sodo. Ker je število a sodo, je število a^3 deljivo z 8. To pomeni, da je število b^3 deljivo s 4, zato je sodo in zato je tudi b sodo. To je pa v protislovju s trditvijo (*).

Moč množic

Če imamo opravka s končnimi množicami je enaka moč množic povsem intuitiven pojem. Preštejemo namreč elemente prve in druge množice, če je število elementov enako, sta množici enako močni. Povsem drugače je, če imamo opravka z neskončnimi množicami. Kako v tem primeru definiramo enako moč množic? Množici A in B sta enako močni, če lahko vsak $a \in A$ preslikamo v enolično določen $b \in B$ in obratno. Z drugimi besedami, obstajati mora bijektivna preslikava med množicama A in B. Ko imamo opravka z neskončnimi množicami, pogosto lahko pridemo do precej neintuitivnih rezultatov, npr., četudi $A \subset B$ imata množici A in B še vedno lahko enako moč. To "enakost" množic moramo zato jemati strogo po definiciji - enakost moči množic striktno pomeni zgolj obstoj bijektivne preslikave med njimi.

Ali je množica naravnih števil močnejša od množice sodih števil?

Poiskati moramo bijektivno preslikavo med tem dvojico množic. V tem primeru je bijektivna preslikava zelo enostavna: $f(n) = 2n$.

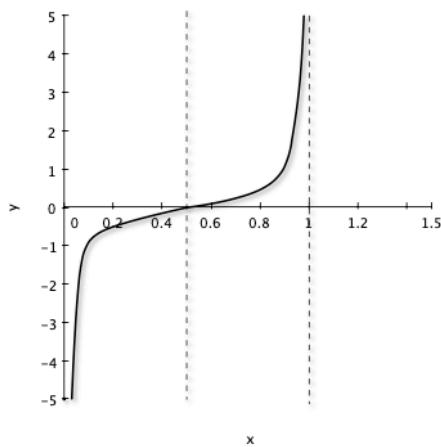
Vemo, da je f bijekcija, ker obstaja njena inverzna preslikava f^{-1} : $f^{-1}(n) = n/2$.

Množica sodih števil je torej enako močna, letudi je to prava podmnožica množice N.

Ali je množica vseh realnih števil enako močna kot množica realnih števil na intervalu $(0, 1)$?

Bijektivna preslikava je recimo:

$$\tan(\pi x - \frac{\pi}{2})$$



Slika prikazuje bijekcijo med $(0, 1)$ in \mathbb{R} .

Lahko bi zmotno mislili, da so vse neskončne množice enako močne, torej da lahko za poljuben par neskončnih množic vedno najdemo bijektivno preslikavo med njima. Vendar temu ni tako. Obstajajo množice, ki so močnejše kot je množica naravnih števil.

Ali je množica realnih števil enako močna kot množica naravnih števil?

Realnih števil je strogo več kot naravnih. **Diagonalizacija**.

Pokazali smo že, da je interval $(0, 1)$ enako močan kot množica realnih števil, zato se osredotočimo na iskanje bijekcije $f: \mathbb{N} \rightarrow (0, 1)$.

Predpostavimo, da sta ti dve množici enako močni, torej med njima obstaja bijekcija.

Predstavitev poljubne funkcije: $f(i) = 0.a_{i1}a_{i2}a_{i3} \dots$

To bijekcijo predstavimo s tabelo. V levem stolpcu zapisemo množico naravnih števil, v desnem stolpcu pa interval $(0, 1)$ - neskončno število decimalnih mest. Realno število je predstavljeno v decimalni obliki z neskončnim številom decimalnih mest (slika a). Vendar za tako predstavljenou funkcijo lahko vedno najdemo realno število, ki gotovo ni slika nobenega elementa iz \mathbb{N} .

To število $x = 0.x_1x_2x_3 \dots$ skonstruiramo tako, da

$$x_i = 1, \text{ ko je } a_{ii} \text{ sodo}$$

$$x_i = 2, \text{ ko je } a_{ii} \text{ liho}$$

Tako dobljeno število x gotovo ni v zalogi vrednosti funkcije f , saj se razlikuje od vsake slike na vsaj enim decimalnem mestu. Slika b) prikazuje konstrukcijo enega takega števila.

i pomeni, da ni v množici, 1 pa pomeni, da i pripada prestavljeni množici.

Za kakršnokoli tako preslikavo lahko pokažemo, da ne more biti bijekcija, in sicer tako, da vedno lahko najdemo eno množico, ki ni v tej tabeli. Zopet vzamemo vrednosti na diagonali in jih preprosto negiramo.

Dobljena množica je gotovo različna kot vse množice v podani tabeli.

\mathbb{N}	$\mathbb{R} : (0, 1)$	$\mathbb{R} : (0, 1)$
1	2 8 8 0 4 4 5 0 2	2 8 8 0 4 4 5 0 2
2	0 8 2 8 2 0 3	0 8 2 8 2 0 3
3	7 3 5 3 5 2 8	7 3 5 3 5 2 8
4	6 0 2 8 2 3	6 0 2 8 2 3
5	8 0 7 0 7 7	8 0 7 0 7 7
6	2 8 8 2 0	2 8 8 2 0
7	0 8	0 8
8	0 1	0 1
9	0 1	0 1
?	1 1 2 1 2 1 1 2 2	1 1 2 1 2 1 1 2 2

(a) "Bijekcija" $f: \mathbb{N} \rightarrow (0, 1)$
(b) Konstrukcija realnega števila, ki gotovo ni slika nobenega elementa iz \mathbb{N}

Slika 2.5: Različna moč množic \mathbb{N} in \mathbb{R}

Naj bosta $n, m \in \mathbb{Z}$ in $n + m \geq 11$. Dokaži, da velja $n \geq 5$ ali $m \geq 6$.

Dokaz s protislovjem.

Recimo, da trditev ne velja. To pomeni, da velja $n < 5$ in $m < 6$. Če je to res, potem je $n + m$ gotovo manjše od 11. To je pa v nasprotju s predpostavko, da je $n + m \geq 11$.

Dokaži, da je število $k \in \mathbb{Z}$ liho natanko takrat, ko je število $k^2 + 4k + 6$ liho.

Če imamo ekvivalenco jo dokažemo tako, da dokažemo obe strani.

(\rightarrow)

Predpostavimo, da je k liho. Torej ga lahko zapišemo kot $k = 2m + 1$.

Od tod sledi $k^2 + 4k + 6 = (2m + 1)^2 + 4(2m + 1) + 6 = 4m^2 + 12m + 11$.

To število je očitno liho.

(\leftarrow)

Predpostavimo, da je $k^2 + 4k + 6$ liho. Potem je $k^2 + 4k = k(k + 4)$ tudi liho. To pomeni, da je k lih; če bi bil sod, bi bil zmnožek tudi sod.

Dokaži, da je vsako celo število n^3 deljivo z 9, ali pa se od devetkratnika razlikuje za 1.

Vsako število n lahko zapišemo na enega od treh načinov:

$$(1) n = 3k$$

$$(3k)^3 = 27k^3 = 9m$$

$$(2) n = 3k + 1$$

$$(3k + 1)^3 = 27k^3 + 27k^2 + 9k + 1 = 9m' + 1$$

$$(3) n = 3k - 1$$

$$(3k - 1)^3 = 27k^3 - 27k^2 + 9k - 1 = 9m'' - 1$$

Pri vseh scenarijih smo dobili 9 krat nekaj.

To je bil dokaz z analizo primerov.

Dokaži: $x, y \in \mathbb{R}, x > 0, y > 0 \Rightarrow \frac{x+y}{2} \geq \sqrt{xy}$.

Rešimo enačbo.

$$x^2 + 2xy + y^2 \geq 4xy.$$

$$x^2 - 2xy + y^2 \geq 0. \text{ Je to res?}$$

$$(x - y)^2 \geq 0. \text{ Da, to je res!}$$

Dokaži, da ne obstaja najmanjše pozitivno racionalno število.

Recimo, da je a/b najmanjše pozitivno racionalno število. Potem je število $a/2b$ še manjše.

To pa je v protislovju s predpostavko, da je a/b najmanjše racionalno število.

Dokaži, da za vsak algoritem za stiskanje brez izgub obstajajo vhodi, ki jih algoritem ne stisne.

Vhoda dolžine 1 gotovo ne moremo stisniti.

Naj bo h višina uravnoteženega dvojiškega drevesa z N vozlišči. Dokaži: $\log N \leq h \leq \log N + 1$.

Najprej dokažemo, da ima polno uravnoteženo drevo višine h natanko $2^{h+1} - 1$ vozlišč.

Indukcija:

- baza

Če je $h = 0$, potem imamo 1 vozlišče, kar je enako $2^{0+1} - 1$. Trditev za ta primer torej velja.

- induktivni korak

Predpostavimo, da trditev velja za h in pokažemo, da velja tudi za $h + 1$. Drevo višine $h + 1$

(Q , \exists sestavljeni iz

- korena
- levega poddrevesa višine h (po induktivni predpostavki je sestavljeni iz $2^{h+1} - 1$ vozlišč)
- desnega poddrevesa višine h (velja enako kot za levo poddrevo)

Skupno število vozlišč drevesa višine $h + 1$ je torej $1 + 2 * (2^{h+1} - 1) = 2^{h+2} - 1$.

Uravnoteženo drevo višine h ima potem takem od 2^h do $2^{h+1} - 1$.

$$2^h \leq N \leq 2^{h+1} - 1.$$

Končni avtomati

Osnovni pojmi

Abeceda

- Σ : poljubna končna množica simbolov
- Σ^* : množica vseh nizov (končnih zaporedij simbolov), sestavljenih iz simbolov v množici Σ
- ϵ : prazen niz
- primer

$$\Sigma = \{0, 1\}$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$$

Niz (beseda)

- končno zaporedje simbolov iz neke abecede
- dolžina niza = $|w|$, število simbolov v nizu w

Jezik

- poljubna (končna ali neskončna) podmnožica množice Σ^* , $L \subseteq \Sigma^*$
- zanimali nas bodo predvsem neskončni jeziki
- primer
 - $\Sigma = \{0, 1\}$
 - jezik L_{111} : množica vseh nizov, ki vsebujejo podniz 111
 - Σ : množica osnovnih simbolov v programskejem jeziku
 - jezik: množica vseh sintaktičnih pravilnih programov v tem programskem jeziku

Problem

- problem: za podani jezik L in niz w ugotovi, ali velja $w \in L$
- primer
 - Ali niz 101100101 pripada jeziku L_{111} ?
 - Ali je podani javanski program sintaktično pravilen?

Deterministični končni avtomati

Avtomat bere vhodni niz in hrani podatek o trenutnem stanju. Prične v začetnem stanju. Po vrsti bere simbole, ki tvorijo vhodni niz. Ko prebere simbol, se mu stanje spremeni v skladu s funkcijo prehodov. Ko avtomat zaključi z branjem, preverimo, ali njegovo trenutno stanje pripada množici končnih stanj. Če to drži, potem avtomat vhodni niz sprejme, sicer pa ga ne sprejme. Jezik avtomata je množica vseh nizov, ki jih avtomat sprejme.

Za vsak vhodni simbol je en prehod iz vsakega stanja.

V vsakem stanju ima samo en možen prehod po vsakem simbolu.

$A =$

- Q : končna množica stanj
- Σ : končna množica vhodnih simbolov
- $\delta: Q \times \Sigma \rightarrow Q$ funkcija prehodov
 - ta funkcija mora biti totalna (definirana za vse pare stanj in simbolov)
 - $\delta(q, a) = q'$ pomeni: če je avtomat v stanju q in prebere simbol a , potem preide v stanje q'
- q_0 : začetno stanje
- F : množica končnih stanj

Povzetek

Unija in presek

1. korak: za vsak jezik nariši diagram
2. korak: za unijo/presek L_1 in L_2 nariši tabelo in nato pretvori v diagram prehodov (minimiziraj)

Presek: končna stanja so vsa tista, ki so sestavljeni iz vseh končnih

Unija: končna stanja so vsa tista, ki so sestavljeni iz vsaj enega končnega stanja

Komplement:

- DKA: zamenjamo končna in nekončna stanja
- NFA: najprej pretvorimo v DKA, nato zamenjamo končna in nekončna stanja

Deljivost

V DKA označimo stanje qr , ki predstavlja ostanek r ($0 \leq r \leq (n - 1)$).

Jezik je regularen, če je sprejet z DFA, NFA ali NFA epsilon oz. opisan z regularnimi izrazi.

Naloge

- nariši avtomat, če imaš podan jezik L
- dokaži, da je L regularen: zapis FA, ki sprejme jezik

KONEC

$Q: n+1$, črtamo spredaj
nikamor: g_0

niz 100

g_0 : začetno
 g_1 : konča z 1
 g_2 : konča z 10
 g_3 : konča z 100

	0	1
Σ	g_0	g_1
1	g_1	g_2
10	g_2	g_3
100	g_3	g_0

	10	11
100	100	101
1000	1000	1001

ZAČETEK

$Q: n+2$, [črtamo zadaj]
nikamor: DEAD

niz aba

g_0 : začetno
 g_1 : začne z a
 g_2 : začne z ab
 g_3 : začne z aba

DEAD

	a	b	NIKAMOR
Σ	g_0	g_1	DEAD
a	g_1	DEAD	g_2
ab	g_2	g_3	DEAD

aba (g_3) g_3 g_3 [FINAL]

DEAD DEAD DEAD

PODHIZ

$Q: n+1$

niz bab

g_0 : začetno
 g_1 : vsebuje b
 g_2 : vsebuje ba
 g_3 : vsebuje bab

	a	b
Σ	g_0	g_1
b	g_1	g_2
ba	g_2	g_0
bab	g_3	g_3

DIVISIBLE [group by divisors, remainder states]

ni končnih stanj

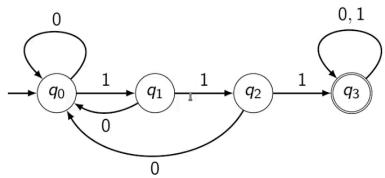
divisible by 3 : 0 remainder g_0 , 1 remainder g_1 , ...

	20,3,6,93	{1,4,73}	...
0	g_0		
1	g_1		
2	g_2		

$10 \div 3 = 3 + 1$

Sestavimo DKA za jezik L_{111} .

- $w \in L_{111} \Leftrightarrow w$ vsebuje podniz 111
- osnovna ideja: stanje nam pove, kaj smo že videli
- potrebujemo 4 stanja: q_0, q_1, q_2, q_3
 - q_0 : začetno stanje in stanje, v katerem se nahajamo, ko smo pravkar prebrali ničlo
 - q_1 : stanje, v katerem se nahajamo, ko smo pravkar prebrali eno zaporedno enico
 - q_2 : stanje, v katerem se nahajamo, ko smo pravkar prebrali dve zaporedni enici
 - q_3 : stanje, v katerem se nahajamo, ko smo pravkar prebrali tri zaporedni enice (končno stanje)



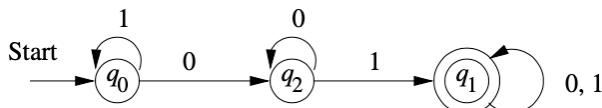
Sestavimo DKA, ki sprejme besede sestavljenje iz 0 in 1 ter vsebuje podniz 01.

$\{x01y \mid x \text{ in } y \text{ sta niza iz ničel in enic}\}$

Primeri nizov, ki so v tem jeziku: 01, 11010, 100111

Primeri nizov, ki niso v tem jeziku: ϵ , 0, 111000

$$A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$



	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

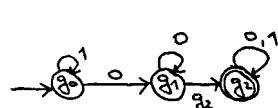
DKA ostane v stanju q_0 dokler ne sprejme prve črke 0. Po sprejemu črke 0 pride v stanje q_1 . V stanju q_1 ostanemo, dokler ne dobimo vhodne črke 1, pomeni, da sta zadnji črki bili 0 in 1, se pravi niz vsebuje 01, torej gremo v sprejemajoče stanje q_1 , kjer avtomat ostane ne glede na nadaljne vhodne črke.

Q: $n + 1$

niz: 01

- q₀: začetno
- q₁: vsebuje 0
- q₂: vsebuje 01

	0	1
Σ	q_1	q_0
0	q_1	q_2
01	q_2	q_2



Sestavimo DKA, ki sprejme naslednji jezik:

$$L = \{w \mid w \text{ ima sodo število ničel in sodo število enic}\}$$

Avtomat bo štel po modulu 2. Stanje si bo zapomnil, če je število videnih ničel oz. enic enako sodo ali liho.

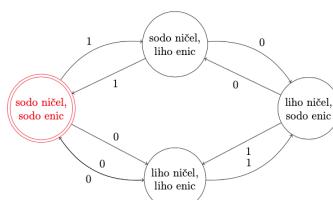
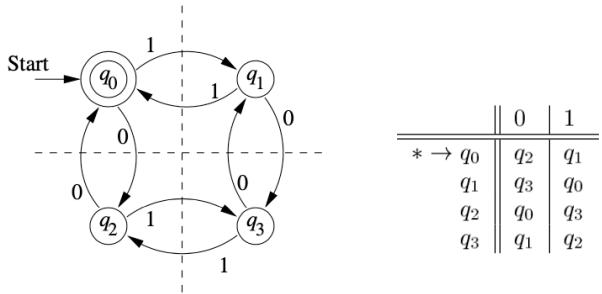
q_0 : Videli smo ničle in enice, število enic je sodo.

q_1 : Število ničel je sodo, število enic pa liho.

q_2 : Število enic je sodo, število ničel pa liho.

q_3 : Videli smo ničle in enice, število enic je liho.

$$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\})$$



Preverimo za 110101:

- $\hat{\delta}(q_0, \epsilon) = q_0$.
- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_1$.
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$.
- $\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$.
- $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$.

Naj bo $D = (Q, \Sigma, \delta, q_0, F)$ deterministični končni avtomat. Dokaži, da za vsak niz $w \in \Sigma^*$ in vsako stanje $q \in Q$ velja

$$\hat{\delta}(q, w) = \hat{\delta}(\hat{\delta}(q, x), y), \quad (1.1)$$

pri čemer je $w = xy$ za $x, y \in \Sigma^*$.

Trditev $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ dokažimo z indukcijo nad $n = |y|$. Naj bo $q' = \hat{\delta}(q, x)$. Baza indukcije: za $y = \epsilon$ velja $\hat{\delta}(q, x\epsilon) = \hat{\delta}(q', \epsilon) = q'$. Indukcijski korak: predpostavimo, da trditev velja za predpono z dolžine $n - 1$ niza $y = za$, kjer je a črka. Se pravi, predpostavimo $\hat{\delta}(\hat{\delta}(q, x), z) = \hat{\delta}(q, xz)$. Dobimo $\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), za) = \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), z), a) = \hat{\delta}(\hat{\delta}(q, xz), a) = \hat{\delta}(q, xza) = \hat{\delta}(q, xy)$.

Pokaži, da za vsako stanje $q \in Q$, vsak niz $x \in \Sigma^*$ ter vsak znak $a \in \Sigma$ velja

$$\hat{\delta}(q, ax) = \hat{\delta}(\hat{\delta}(q, a), x).$$

Najprej dokažimo, da je $\hat{\delta}(q, a) = \delta(q, a)$: $\hat{\delta}(q, a) = \hat{\delta}(q, a\epsilon) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$. Potem samo uporabimo izrek iz prejšnje naloge za $w = ax$.

Naj bo $D = (Q, \Sigma, \delta, q_0, F)$ DKA in naj bo $a \in \Sigma$ tak, da za vsako stanje $q \in Q$ velja $\delta(q, a) = q$. Pokaži, da je tedaj ali $\{a\}^* \subseteq L(D)$ ali pa $\{a\}^* \cap L(D) = \emptyset$.

Najprej lahko z indukcijo dokažemo, da velja $\hat{\delta}(q, x) = q$, kjer je $x \in \{a\}^*$ (baza indukcije je $x = \epsilon$, induksijski korak je $x' = xa$). To pomeni, da noben x ne zapusti začetnega stanja, torej je to, ali je $\{a\}^* \subseteq L$, odvisno samo od tega, ali je q_0 sprejemajoče stanje.

DKA, ki sprejema besede, ki se končajo z 000.

Q: $n + 1$, črtamo spredaj, nikamor v q_0

niz: 000

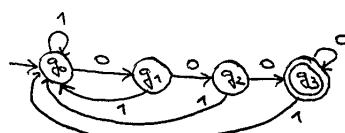
- q_0 : začetno
- q_1 : konča z 0
- q_2 : konča z 00
- q_3 : konča z 000

	0	1
Σ	q_0	q_1
0	q_1	q_2
00	q_2	q_3
000	q_3	q_0

$\rightarrow 00, q_1 \xrightarrow{0} q_2, q_2 \xrightarrow{0} q_3, q_3 \xrightarrow{0} q_0$

$\rightarrow 000, q_0 \xrightarrow{0} q_1, q_1 \xrightarrow{0} q_2, q_2 \xrightarrow{0} q_3, q_3 \xrightarrow{0} q_0$

$\rightarrow 0000 \rightarrow 000, q_0 \xrightarrow{0} q_1, q_1 \xrightarrow{0} q_2, q_2 \xrightarrow{0} q_3, q_3 \xrightarrow{0} q_0$



DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki se začnejo z nizom 0011.

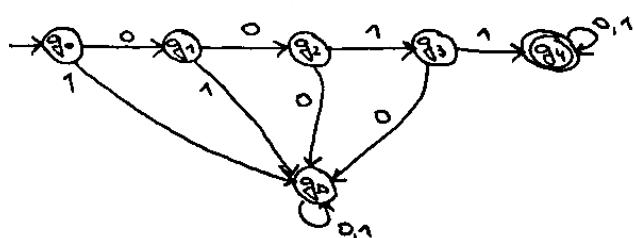
$L = \{0011w \mid w \in \{0, 1\}^*\}$

Q: $n + 2$, nikamor: dead stanje

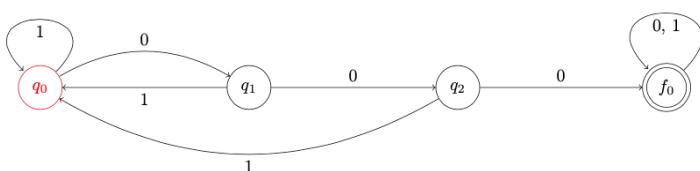
niz: 0011

- q_0 : začetno
- q_1 : začne z 0 (pomnenje)
- q_2 : začne z 00 (pomnenje)
- q_3 : začne z 001 (pomnenje)
- q_4 : začne z 0011 (vse prave besede)
- q_5 : dead stanje (vse, ki niso prave besede)

	0	1
Σ	q_0	q_1
0	q_1	q_2
00	q_2	q_3
001	q_3	q_4
0011	q_4	q_5
-	q_5	q_5



DKA, ki sprejme natanko tiste binarne nize, ki vsebujejo 3 zaporedne ničle.



DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki vsebujejo liho število ničel.

Primeri nizov, ki pripadajo jeziku: 0, 0010, 1011

Primeri nizov, ki ne pripadajo jeziku: ϵ (0 je sodo), 1, 11, 001

$w = 01100$

$(q_0, 01100) \rightarrow (q_1, 1100) \rightarrow (q_1, 100) \rightarrow (q_1, 00) \rightarrow (q_0, 0) \rightarrow (q_1, \epsilon)$



Iz vsakega stanja morajo biti definirani vsi prehodi.

Stanja določajo paritet, koliko 0 je bilo na vhodu:

$q_0 \dots$ na vhodu sodo število 0

$q_1 \dots$ na vhodu liho število 0

Izračunali smo modulo 2 (dve stanji, ker sta dva ostanka).

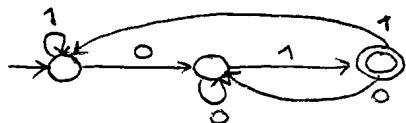
Ko dobimo na vhod 1, ostanemo v stanju q_0 , ker se ni nič zgodilo z številom 0.

Ko smo v stanju q_1 in vidimo 0, vemo, da imamo liho število 0.

DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki se končajo z 01 ali z 10.

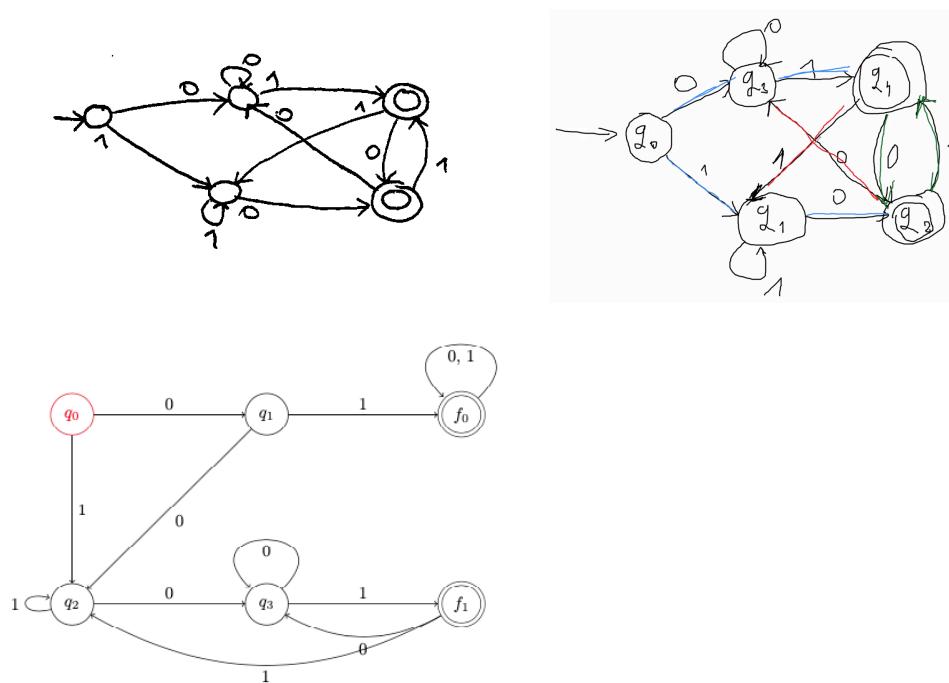
$$L = \{w \mid w \in \{0, 1\}^*, w = x01 \text{ ali } w = x10\}$$

1. Narišemo avtomat $w = x01$



Vsak začetek je dober, zato tukaj nimamo slepih stanj.

2. Napišemo avtomat za $w = x01$ ali $w = x10$

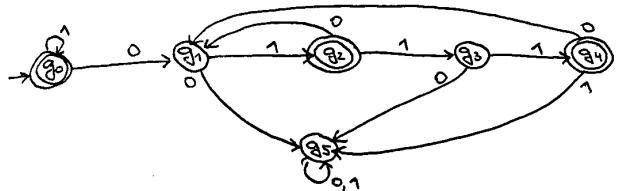


DKA, ki sprejema besede $w \in \{0, 1\}^*$, v katerem vsaki ničli sledi natanko ena enica ali pa natanko 3 enice.

Primeri nizov, ki pripadajo jeziku: ϵ (v prazni množici vsak pogoj drži), 1, 11, 01, 0111, 010111

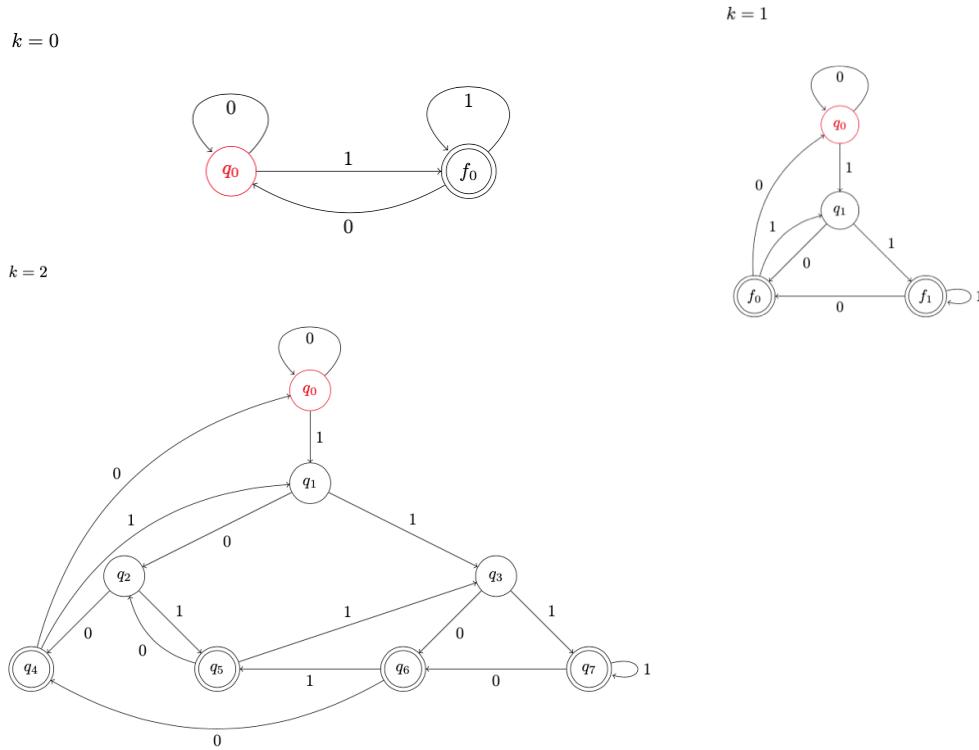
Primeri nizov, ki ne pripadajo jeziku: 0, 011, 011111

V prazni množici vsak pogoj drži \Rightarrow epsilon pripada jeziku.



$q_5 \dots$ error state

Sestavi DKA, ki sprejema jezik L_k binarnih nizov w , za katere velja $w_{n-k} = 1$, za $k < n = |w|$; poskusi za $k = 0, 1, 2$ in 3 . Kaj pa $k = 9$?



Vidimo, da je graf DKA sestavljen iz začetnega stanja in polno razvejanega binarnega drevesa. V naslednji nalogi bomo dokazali, da mora imeti vsaj 2^{k+1} stanj (opisana struktura DKA jih ima ravno toliko).

Dokaži, da za jezik L_k ne obstaja DKA D z manj kot 2^{k+1} stanji.

Vsak DKA D, za katerega je $L(D) = L_k$, ima $|Q| \geq 2^{k+1}$.

Naj bo X_l množica nizov dolžine $0 \leq l \leq k+1$, ki so enaki oziroma jih lahko dopolnemo do niza dolžine $k+1$, ki ima 1 na prvem mestu. Se pravi, na koncu vsakega niza iz L_k nastopa niz iz X_{k+1} .

Velja $X_0 = \{\epsilon\}$, $X_1 = \{1\}$, $X_2 = \{10, 11\} \dots$ Se pravi, $|X_0| = 1$ in $|X_i| = 2^{i-1}$ za $i \geq 1$. Če dokažemo, da mora vsak DKA za L_k vsebovati vsah eno stanje za vsak $x \in x_0 \cup \dots \cup x_{k+1}$, bo takoj sledilo $|Q| \geq 1 + \sum_{i=0}^k 2^i = 2^{k+1}$.

Kot prvo vemo, da si $x \in X_i$ ter $y \in X_j$ ne smeta deliti stanja, če $i \neq j$, saj zahtevata različno število vhodnih črk do dopolnitve do niza iz X_{k+1} . Zdaj pa recimo, da sta $x \neq y$ iz istega X_l ($l \geq 2$, kajti X_0 in X_1 imata samo en element). Niza se morata razlikovati vsaj za eno črko: brez izgube splošnosti vzamemo, da velja $1 = x_i \neq y_i = 0$ (pozicije v nizu se začnejo z 1). Obema nizoma dodajmo k ničel. Podaljšan x je element L_k , podaljšan y pa ne, torej nista začela v istem stanju.

Presek in unija

Ključno je, da moramo vzdrževati stanja obeh DKA v združenem DKA. Ustvariti moramo nova stanja za unijo kot neposredno množenje (kartezični produkt) prvotnih stanj. Na ta način imamo stanja za vsako kombinacijo stanj v prvotnih DKA-jih. Neposredno lahko izračunamo pravila prehoda za nov DKA.

Uporabimo kartezični produkt na jezikih avtomata, $M_1 \text{ in } M_2 \Rightarrow M = M_1 \times M_2$:

- M vsebuje pare vseh stanj iz sestavljenega DKA-ja; če ima originalni DFA naslednja stanja A, B, C in x, y, z, potem ima produkt $\{Ax, Ay, Az, Bx, By, Bz, Cx, Cy, Cz\}$
- funkcija prehodov se posodobi tako da če bi na določenem koraku, niz povzročil prehod avtomata M_1 iz stanja A v B ter povzročil prehod avtomata M_2 iz stanja x v y potem bi bil prehod avtomata iz Ax v By
- začetno stanje je par, ki je sestavljen iz začetnih stanj DKA-ja

Če sta x_1 in y_0 končna stanja M_1 in M_2 , potem bi bil presek $\{x_1y_0\}$, unija pa $\{x_1y_0, x_1y_1, x_0y_0\}$.

Presek: končna stanja so vsa tista, ki so sestavljena iz vseh končnih

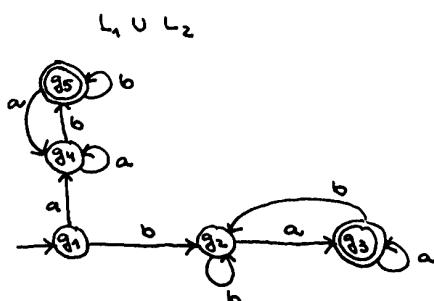
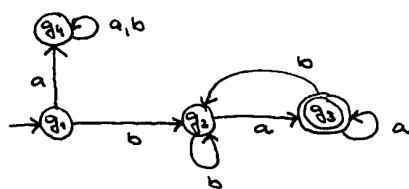
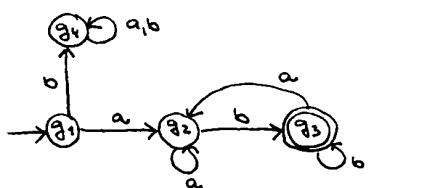
Unija: končna stanja so vsa tista, ki so sestavljena iz vsaj enega končnega stanja

DKA, ki sprejema besede $w \in \{a, b\}^*$, ki se začnejo in končajo z drugačnimi simboli.

$$L_1 = \{\text{začne z } a \text{ in konča z } b\} = \{ab, aab, aabab, \dots\}$$

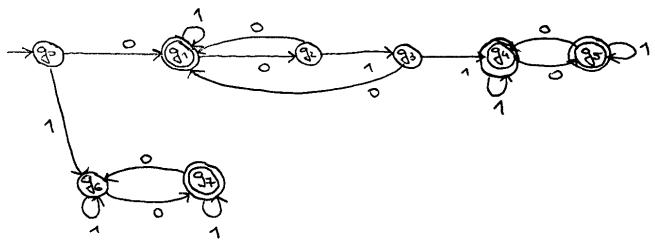
$$L_2 = \{\text{začne z } b \text{ in konča z } a\} = \{ba, bba, bbaba, \dots\}$$

$$L = L_1 + L_2 \dots \text{jezik, ki se začne in konča z drugačnimi simboli}$$

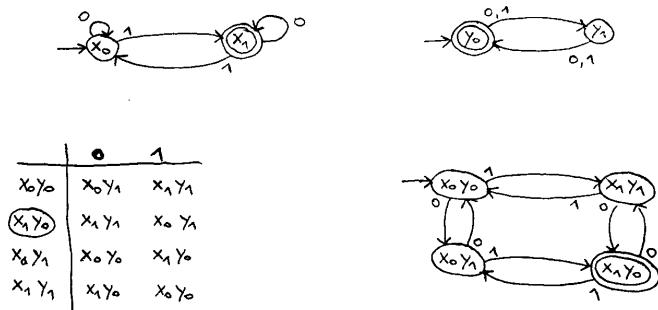


DKA, ki sprejema unijo jezikov:

besede $w \in \{0, 1\}^*$, ki vsebujejo liho število enic + besede $w \in \{0, 1\}^*$, ki se začnejo z nizom 0011



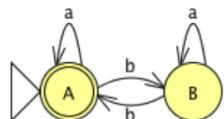
DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki imajo liho število enic in so sode dolžine.



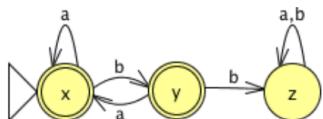
KONČNA STANJA : presek končnih stanj

DKA, ki sprejema besede $w \in \{a, b\}^*$, ki imajo sodo število b-jev in ne vsebujejo podniza bb.

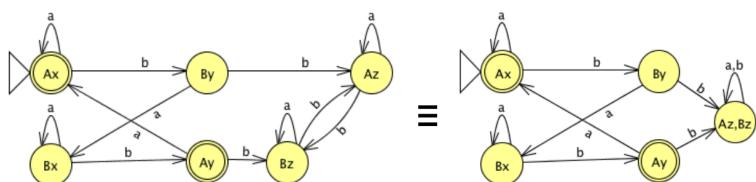
L1 = sodo število b-jev



L2 = ne vsebujejo podniza bb



Konstrukcija preseka nam da stroj z 6 stanji: $\{Ax, Ay, Az, Bx, By, Bz\}$, ki so dobljeni iz vseh parov stanj. Težji del je ugotoviti, kako smiselno narisati dobljeni DKA. Stanja lahko postavimo v mrežo 2×3 .

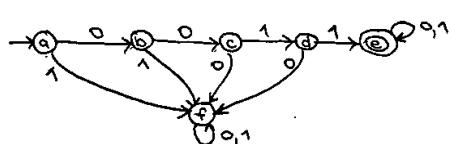
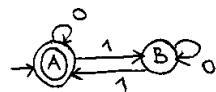


Stanja Az in Bz sta mrtva, zato ju lahko zamenjamo z enim stanjem.

DKA, ki sprejema presek jezikov:

besede $w \in \{0, 1\}^*$, ki se začnejo z nizom **0011** + besede $w \in \{0, 1\}^*$, ki vsebujejo sodo število enic

1. korak: za vsak jezik nariši diagram



2. korak: za presek L1 in L2 nariši tabelo in nato pretvori v diagram prehodov (minimiziraj)

Končna stanja so vsa tista, ki so sestavljeni iz vseh končnih.

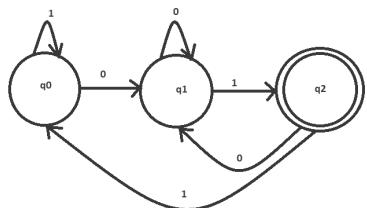
	0	1
$\rightarrow Aa$	A_b	B_f
A_b	A_c	B_f
A_c	A_f	B_d
A_d	A_f	B_e
A_e	A_e	B_e
A_f	A_f	B_f
B_a	B_b	A_f
B_b	B_c	A_f
B_c	B_f	A_d
B_d	B_f	A_e
B_e	B_e	A_e
B_f	B_f	A_f

DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki se končajo z 01 in imajo sodo število enic.

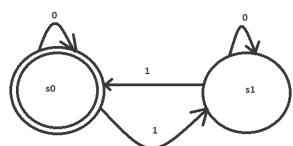
$$L_1 = \{01, 001, 101, 0101, 1001, 1101, \dots\}$$

$$L_2 = \{11, 011, 101, 110, 0011, 1100, \dots\}$$

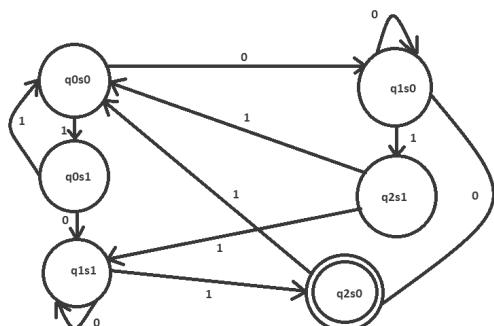
Jezik L1



Jezik L2



$$L = L_1 \cap L_2 = \{1001, 0101, 01001, 10001, \dots\}$$



Komplement

Zamenjamo končna in nekončna stanja, prehodi ostanejo isti (DFA).

Deljivost

Ko delimo število w z n , dobimo ostanek $0, 1, \dots, (n - 2)$ ali $(n - 1)$.

Če je ostanek 0 pomeni, da je w deljivo z n , drugače ni.

V DKA bomo označili stanje q_r , ki bo predstavljalo ostanek r , kjer je $0 \leq r \leq (n - 1)$ in n je število stanj v DKA.

Ko procesiramo število oz. niz w nad abecedo, je končno stanje q_r , ki implicira na $w \% n = r$.

V vsakem avtomatu je namen stanja podoben pomnilniškemu elementu.

DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki predstavljajo števila v binarnem zapisu, ki so deljiva s 3.

$$L = \{w \in \{0, 1\}^*, w \bmod 3 = 0\}$$

Primeri nizov, ki so v jeziku: $\epsilon, 0, 11, 110, 1001$

Primeri nizov, ki niso v jeziku: $1, 10$

Stanja naj bodo možni ostanki $(0, 1, 2)$

- stanje q_0 : $3k$

$$3k \mid 0 \Rightarrow 2 \times 3k = 6k$$

$$3k \mid 1 \Rightarrow 2 \times 3k + 1 = 6k + 1$$

- stanje q_1 : $3k + 1$

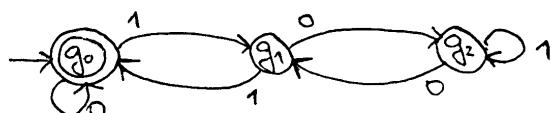
$$3k + 1 \mid 0 \Rightarrow 2(3k + 1) = 6k + 2$$

$$3k + 1 \mid 1 \Rightarrow 2(3k + 1) + 1 = 6k + 3$$

- stanje q_2 : $3k + 2$

$$3k + 2 \mid 0 \Rightarrow 2(3k + 2) = 6k + 4$$

$$3k + 2 \mid 1 \Rightarrow 2(3k + 2) + 1 = 6k + 5$$



DKA, ki sprejema besede $w \in \{0, 1\}^*$, ki predstavljajo števila v binarnem zapisu, ki so deljiva s 5.

Število povezav je enako $Q \times \Sigma = 5 \times 2 = 10$.

Za $n = 5$:

1. Stanje q_0 je lahko doseženo, če je ostanek 0. To stanje je začetno in tudi končno.
2. Stanje q_1 je lahko doseženo, če je ostanek 1.
3. Stanje q_2 je lahko doseženo, če je ostanek 2.
4. Stanje q_3 je lahko doseženo, če je ostanek 3.
5. Stanje q_4 je lahko doseženo, če je ostanek 4.

Imamo 5 stanj, se pravi 5 ostankov.

Na vsakem koraku izberemo naslednjo binarno številko.

$$L = \{w \in \{0, 1\}^*, w \bmod 5 = 0\}$$

Primeri nizov, ki so v jeziku: $\epsilon, 101, 1010, 1111$

Primeri nizov, ki niso v jeziku: 1, 10, 110

Stanja naj bodo možni ostanki (0, 1, 2, 3, 4)

- stanje $q_0: 5k + 0$

$$5k \mid 0 \Rightarrow 2 \times 5k + 0 = 10k + 0$$

$$5k \mid 1 \Rightarrow 2 \times 5k + 1 = 10k + 1$$

- stanje $q_1: 5k + 1$

$$5k + 1 \mid 0 \Rightarrow 2 \times (5k + 1) = 10k + 2$$

$$5k + 1 \mid 1 \Rightarrow 2 \times (5k + 1) + 1 = 10k + 3$$

- stanje $q_2: 5k + 2$

$$5k + 2 \mid 0 \Rightarrow 2 \times (5k + 2) = 10k + 4$$

$$5k + 2 \mid 1 \Rightarrow 2 \times (5k + 2) + 1 = 10k + 5$$

- stanje $q_3: 5k + 3$

$$5k + 3 \mid 0 \Rightarrow 2 \times (5k + 3) = 10k + 6 = 10k + 6$$

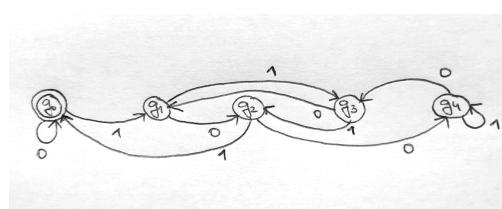
$$5k + 3 \mid 1 \Rightarrow 2 \times (5k + 3) + 1 = 10k + 7 = 10k + 7$$

- stanje $q_4: 5k + 4$

$$5k + 4 \mid 0 \Rightarrow 2 \times (5k + 4) = 10k + 8 = 10k + 8$$

$$5k + 4 \mid 1 \Rightarrow 2 \times (5k + 4) + 1 = 10k + 9 = 10k + 9$$

q_k	deljivost	ostanek
q_0	$5k + 0 \mid 0$	0
	$5k + 0 \mid 1$	1
q_1	$5k + 1 \mid 0$	2
	$5k + 1 \mid 1$	3
q_2	$5k + 2 \mid 0$	4
	$5k + 2 \mid 1$	0
q_3	$5k + 3 \mid 0$	1
	$5k + 3 \mid 1$	2
q_4	$5k + 4 \mid 0$	3
	$5k + 4 \mid 1$	4



* q_0 je začetno in končno stanje

DKA, ki sprejema besede $w \in \{0, 1, 2\}^*$, ki predstavljajo števila v trojiškem zapisu, ki so deljiva s 5.

$$L = \{w \in \{0, 1, 2\}, w \bmod 5 == 0\}$$

Stanja naj bodo možni ostanki (0, 1, 2, 3, 4)

- stanje $q_0: 5k + 0$

$$5k | 0 \Rightarrow 3 \times 5k + 0 = 15k + 0$$

$$5k | 1 \Rightarrow 3 \times 5k + 1 = 15k + 1$$

$$5k | 2 \Rightarrow 3 \times 5k + 2 = 15k + 2$$

- stanje $q_1: 5k + 1$

$$5k + 1 | 0 \Rightarrow 3 \times (5k + 1) + 0 = 15k + 3$$

$$5k + 1 | 1 \Rightarrow 3 \times (5k + 1) + 1 = 15k + 4$$

$$5k + 1 | 2 \Rightarrow 3 \times (5k + 1) + 2 = 15k + 5$$

- stanje $q_2: 5k + 2$

$$5k + 2 | 0 \Rightarrow 3 \times (5k + 2) + 0 = 15k + 6$$

$$5k + 2 | 1 \Rightarrow 3 \times (5k + 2) + 1 = 15k + 7$$

$$5k + 2 | 2 \Rightarrow 3 \times (5k + 2) + 2 = 15k + 8$$

- stanje $q_3: 5k + 3$

$$5k + 3 | 0 \Rightarrow 3 \times (5k + 3) + 0 = 15k + 9$$

$$5k + 3 | 1 \Rightarrow 3 \times (5k + 3) + 1 = 15k + 10$$

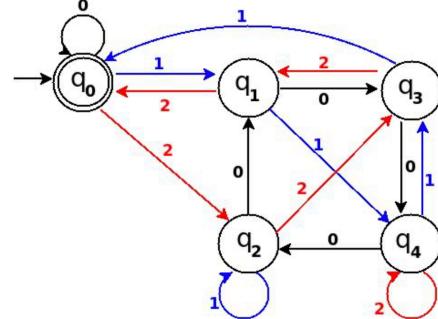
$$5k + 3 | 2 \Rightarrow 3 \times (5k + 3) + 2 = 15k + 11$$

- stanje $q_4: 5k + 4$

$$5k + 4 | 0 \Rightarrow 3 \times (5k + 4) + 0 = 15k + 12$$

$$5k + 4 | 1 \Rightarrow 3 \times (5k + 4) + 1 = 15k + 13$$

$$5k + 4 | 2 \Rightarrow 3 \times (5k + 4) + 2 = 15k + 14$$

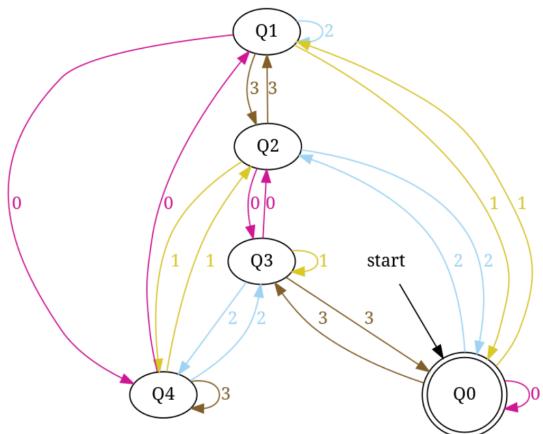


DKA, ki sprejema nize ki so deljivi s 3 ali 5.

Najprej naredimo avtomat za 3 in za 5. Nato pa združimo v avtomat, ki ima 15 stanj.

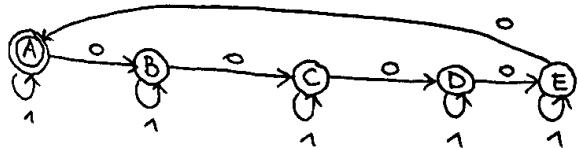
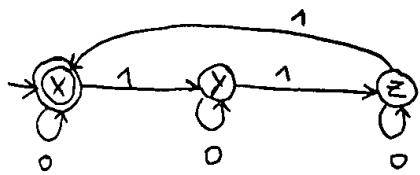
Konstruiraj DKA za števila v bazi 'b', ki so deljiva z 'n'

V takem DKA bo n stanj (za n ostankov) in 'b' (število simbolov jezika) * 'n' povezav.

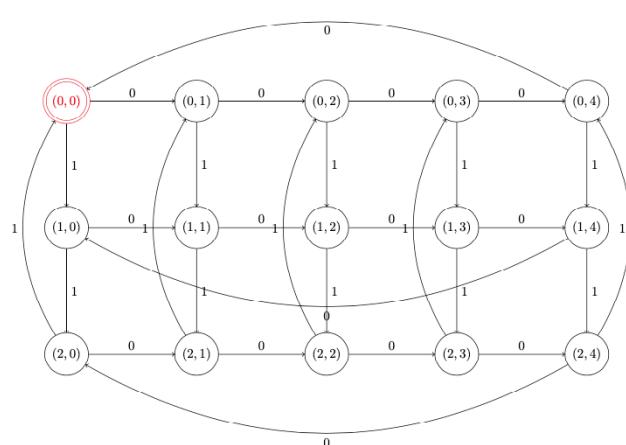


DFA accepting number strings in base 4 those are divisible by 5

Sestavi DKA, ki sprejema binarne nize, ki vsebujejo $3k$ enic in $5l$ ničel, za neka $k, l \geq 0$.



stanja	0	1
Ax	Bx	Ay
Ay	By	Az
Az	Bz	Ax
Bx	Cx	By
By	Cy	Bz
Bz	Cz	Bx
Cx	Dx	Cy
Cy	Dy	Cz
Cz	Dz	Cx
Dx	Ex	Dy
Dy	Ey	Dz
Dz	Ez	Dx
Ex	Ax	Ey
Ey	Ay	Ez
Ez	Az	Ex



Naj bo $L = \{w \in \{0, 1\}^* \mid 01 \subseteq w\}$ jezik binarnih nizov, ki vsebujejo 01 kot podniz. Dokaži, da je jezik L regularen.

PODNEZ

$Q : n+1$

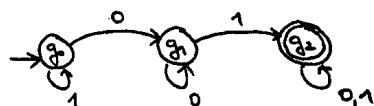
niz: 01

q_0 : začetno

q_1 : vsebuje 0

q_2 : vsebuje 01

Σ	0	1
q_0	q_1	q_0
0	q_1	q_2
01	q_2	q_2
	q_{10}	q_{11}



Jezik je regularen, če uspemo zapisati končni avtomat.

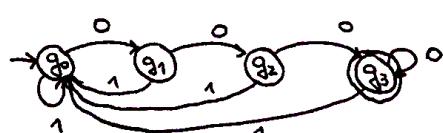
Pošči DKA, ki sprejme natanko tiste binarne nize, ki se končajo z 000.

KONEC

$Q : n+1$, črtamo spredaj
nikamor: q_0

niz: 000

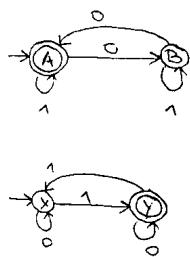
q_0 : začetno
 q_1 : konča z 0
 q_2 : konča z 00
 q_3 : konča z 000



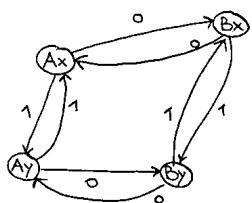
Konstruiraj DKA, ki sprejme natanko tiste binarne nize, ki vsebujejo tri zaporedne ničle (ne nujno na koncu niza).



Poisci DKA, ki vsebuje sodo število 0 in liho število 1.

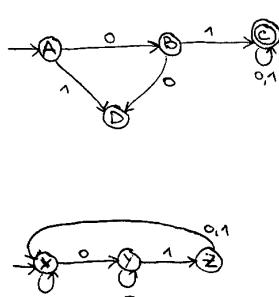


	0	1
Ax	Bx	Ay
Ay	By	Ax
Bx	Ax	By
By	Ay	Bx

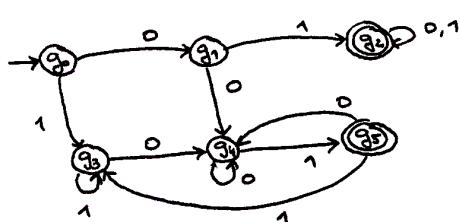


Napaka: Ay mora biti končno stanje

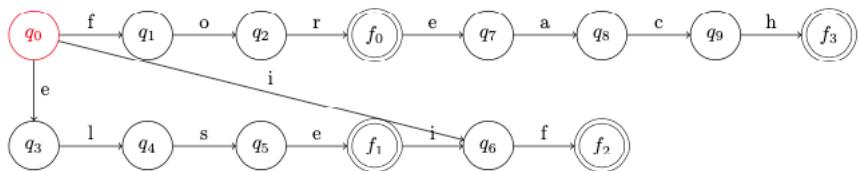
Sestavi DKA, ki ima za jezik nize, ki se začnejo ali končajo (ali oboje) z 01.



	0	1
Ax	By	Dx
Ay	By	Dz
Az	Bx	Dx
Bx	Cy	Dy
By	Cy	Dz
Bz	Cx	Dx
Cx	Cy	Cy
Cy	Cy	Cz
Cz	Cx	Cx
Dx	Dy	Dy
Dy	Dy	Dz
Dz	Dx	Dx

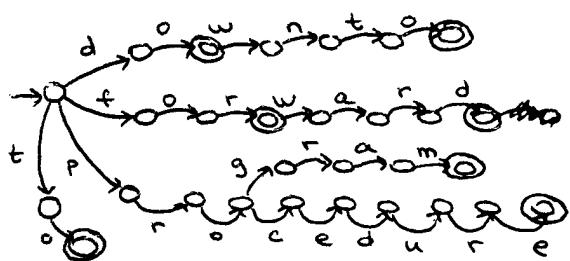


Sestavi čim manjši DKA, ki sprejme nize: for, foreach, if, elif, else.

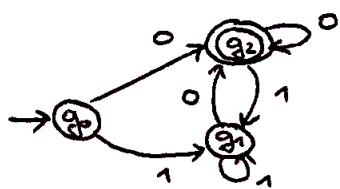


Sestavi DKA za rezervirane besede jezika Pascal.

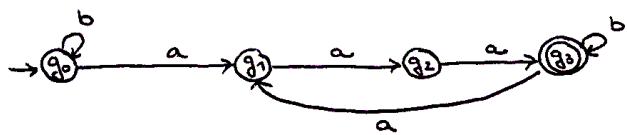
$L = \{\text{for, program, to, do, downto, procedure, forward}\}$



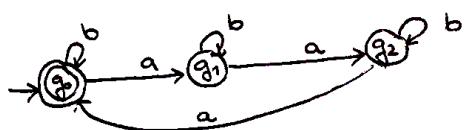
DKA, ki preveri, ali je dano binarno število sodo.



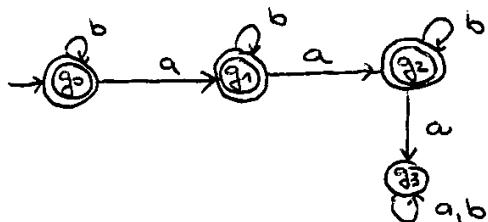
DKA, ki sprejme jezik, sestavljen iz a in b in vsebuje tri zaporedne a.



DKA, ki sprejme L, kjer so vsi nizi L takšni, da je seštevek a-jev deljiv z 3.



DKA, ki sprejme vse nize, ki nimajo več kot 2 a-ja nad abecedo a in b.

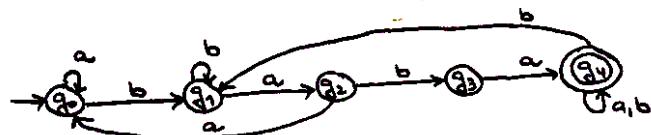


Sestavi DKA D, ki ima za jezik vse nize nad abecedo {a, b}, ki ne vsebujejo podniza baba.

Naj bo $L = L(D)$ za nek deterministični končni avtomat $D = (Q, \Sigma, \delta, q_0, F)$. Konstruiraj tak DKA $D' = (Q', \Sigma', \delta', q'_0, F')$, da bo $L(D') = \overline{L(D)}$. (To med drugim pomeni, da če je L regularen, tedaj je regularen tudi njegov komplement \overline{L} .)

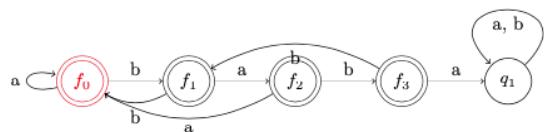
$$L = \{w \in \{a, b\}^* \mid baba \notin w\}$$

$$\overline{L} = \{w \in \{a, b\}^* \mid baba \in w\}$$



$$D = (Q, \Sigma, \delta, q_0, F)$$

$$\overline{D} = (Q, \Sigma, \delta, q_0, Q/F) \quad \text{nekončna stanja postanejo končna}$$



Nedeterministični končni avtomati

Vse komponente imajo enak pomen kot pri DKA.

Edina razlika je funkcija prehodov, ki lahko vsakemu paru stanja in simbola priredi poljubno podmnožico množice stanj.

Avtomat je lahko v več stanjih hkrati (v množici stanj).

Na začetku je v množici stanj $\{q_0\}$.

Če se nahaja v množici stanj

$$\{q_{i_1}, q_{i_2}, \dots, q_{i_k}\},$$

potem po branju simbola a preide v množico stanj

$$\delta(q_{i_1}, a) \cup \delta(q_{i_2}, a) \cup \dots \cup \delta(q_{i_k}, a)$$

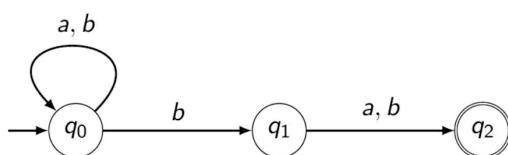
Jezik NKA je množica vseh nizov, pri katerih avtomat pristane v množici stanj, ki ima neprazen presek z množico F .

Primer avtomata

$$A = \{\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\}\}$$

Funkcija δ :

	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	\emptyset	\emptyset



Delovanje avtomata na nizu abaab:

$$\{q_0\} \xrightarrow{a} \{q_0\} \xrightarrow{b} \{q_0, q_1\} \xrightarrow{a} \{q_0, q_2\} \xrightarrow{b} \{q_0\} \xrightarrow{a} \{q_0, q_1\}$$

Delovanje avtomata na nizu bba:

$$\{q_0\} \xrightarrow{b} \{q_0, q_1\} \xrightarrow{b} \{q_0, q_1, q_2\} \xrightarrow{a} \{q_0, q_2\}$$

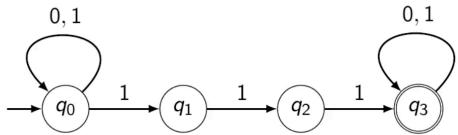
Jezik avtomata:

Zadnji simbol je lahko a ali b, predzadnji mora biti b, na začetku a ali b.

Jezik avtomata so torej vsi nizi, sestavljeni iz najmanj dveh simbolov a in b, pri katerih je predzadnji simbol enak b.

Avtomat za jezik L_{111} .

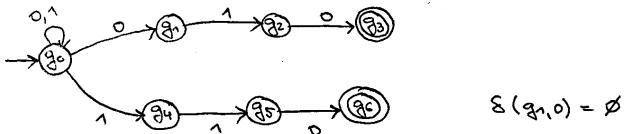
- $w \in L_{111} \Leftrightarrow w$ vsebuje podniz 111



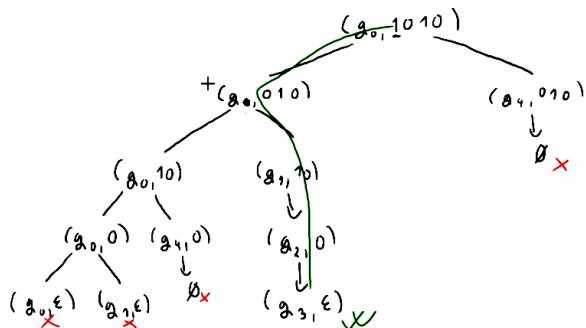
Naloge iz NKA-jev (NKA N poišči DKA D, da bo $L(D) = L(N)$.)

NKA, ki sprejema besede $w \in \{0, 1\}^*$, ki se končajo z nizom 010 ali z nizom 110.

$$L = \{w \in \{0, 1\}^* \mid w = x010 \text{ ali } w = x110\}$$



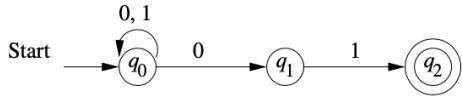
Ko prehod ni definiran, je funkcija prehodov za to stanje prazna **množica**.



Kompakten zapis izčrpnega preiskovanja.

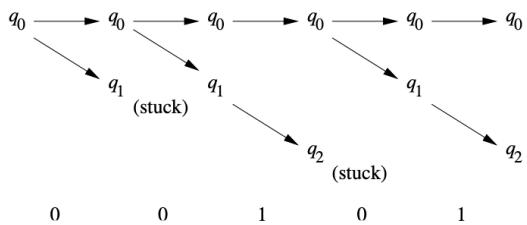
Bolj kompakten zapis jezika.

NKA, jezik, ki se konča na 01.



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	\emptyset	$\{q_2\}$
$*q_2$	\emptyset	\emptyset

00101



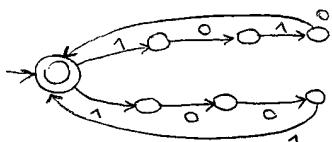
1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}.$
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}.$
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}.$
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}.$
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}.$
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}.$

NKA, ki sprejema besede, ki vsebujejo poljubno mnogo nizov 1010 in 1001 v poljubnem zaporedju.

$$L = \{(1010 + 1001)^*\}$$

Primeri nizov, ki so v jeziku: ϵ , 1010, 10101001

Primeri nizov, ki niso v jeziku: 10101, 100, 1

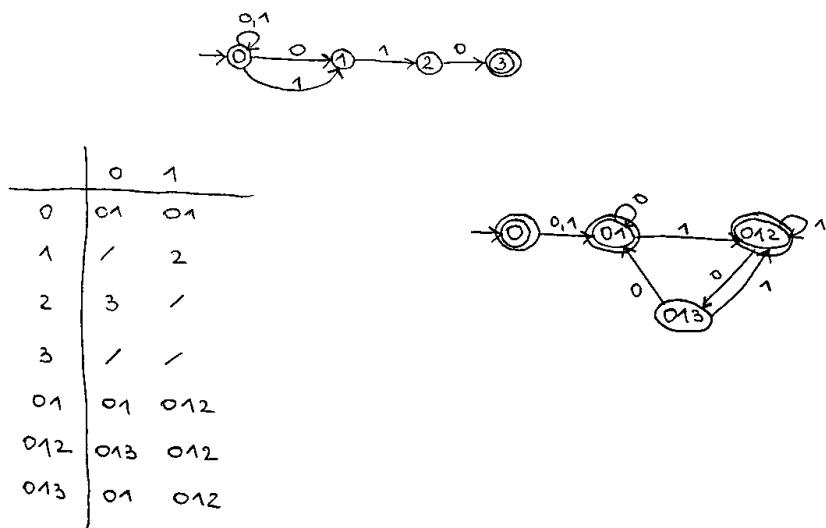


Če želimo razpoznavati neskončne besede, moramo imeti nekje cikel.

NKA, ki je komplement jezika: besede, ki se končajo z nizom 010 ali z nizom 110.

Pri DKA lahko pri komplementu samo zamenjamo končna stanja.

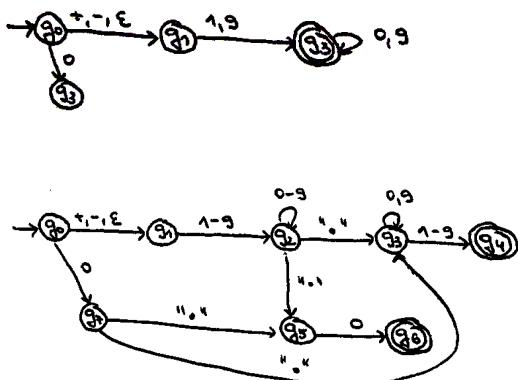
Pri NKA pa moramo najprej pretvoriti v DKA ter nato zamenjamo končna stanja (komplement stanj).



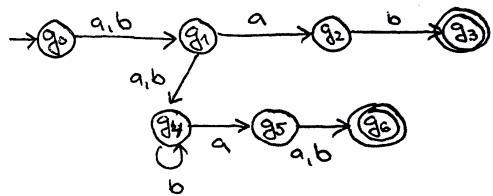
Sestavi končni avtomat, ki sprejme zapise števil v eni izmed sledečih oblik:

- Opcijski predznak (+ oz. -), kateremu sledi celo število (npr. 312, 1337, +545).
- Opcijski predznak (+ oz. -), kateremu sledi decimalno število kot zaporedje števk, znak “.”, nakar sledi zaporedje števk na decimalnih mestih (npr. 1.31, 2.7182, +3.14).

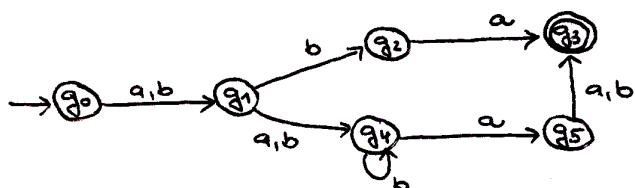
Pri tem se želimo izogniti sprejemanju zapisov, ki imajo odvečne ničle na desni oz. levi strani (npr. 0012, 1.200).



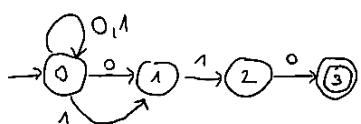
Sestavi NKA nad abecedo $\Sigma = \{a, b\}$, ki sprejme nize oblike $_b^* a _$ ali $_ ab$. Pri čemer $_$ predstavlja poljuben znak, $*$ pa nič ali več ponovitev prejšnjega znaka.



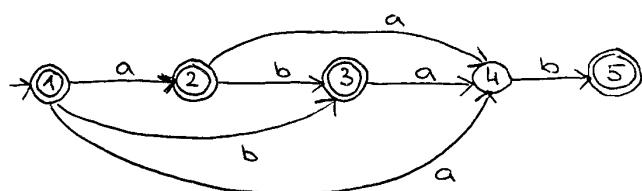
Sestavi NKA nad abecedo $\Sigma = \{a, b\}$, ki sprejme ali nize z aa in bb ali nize brez aa in brez bb.



NKA, ki sprejema komplement jezika: $w \in \{0, 1\}^*$, ki se končajo z nizom 010 ali z nizom 110.



NKA, ki sprejema besede $w \in \{a, b\}^*$, ki se začnejo z 0 ali 1 ponovitvijo simbola a, sledi 0 ali 1 ponovitev simbola b, končajo pa se z 0 ali 1 ponovitvijo niza ab.



Dan imate NKA $N = (Q, \Sigma, \delta, q_0, F)$ s stanji $Q = \{q_0, q_1, q_2, q_3\}$, končnim stanjem $F = \{q_3\}$, določen s funkcijo prehodov:

- $\delta(q_0, 0) = q_1;$
- $\delta(q_0, 1) = q_2;$
- $\delta(q_1, 1) = q_3;$
- $\delta(q_2, 0) = q_3;$
- $\delta(q_3, 0) = q_0;$
- $\delta(q_3, 1) = q_1.$

Obstaja preprosta rekurzivna enačba za preštevanje nizov dolžine k , ki jih N sprejme.

Naj bo $f(k)$ število nizov dolžine k , ki jih N sprejme. Zapišimo regularni izraz za te nize: $((01 + 10)1^{2n})^+$ za $n \in \mathbb{N}_0$. Kot prvo vidimo, da je $f(0) = f(1) = 0$, saj mora biti niz vsaj dolžine 2, ter da je $f(2) = 2$. Ker je tako člen $01 + 01$ kot člen 1^{2n} sode dolžine, so vsi sprejeti nizi sode dolžine, torej je $f(2m + 1) = 0$. Rekurzivno formulo za $f(2m)$, kjer je $m \geq 2$, dobimo tako, da upoštevamo vse n , pri katerih nam ostaneta vsaj dva znaka za naslednji člen $01 + 10$, ter primer, ko vse preostale znake porabimo za člen 1^{2n} (ko je $n = m - 1$):

$$f(2m) = 2 \left(\sum_{i=1}^{m-1} f(2i) + 1 \right)$$

Nedeterministični končni avtomati z ε - prehodi

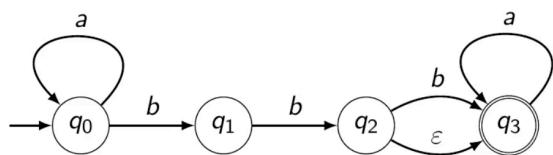
- Vse komponente imajo enak pomen kot pri NKA.
- Edina razlika je funkcija prehodov:

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$$

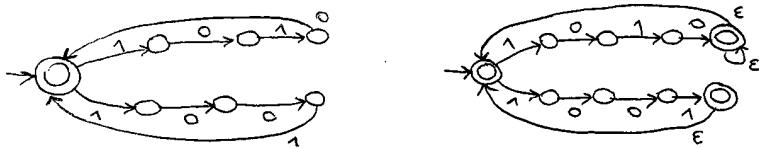
$$\delta(q, \varepsilon) = S$$

Avtomat lahko v stanju q spontano (brez branja) preide v množico stanj S (podmnožica množice stanj).

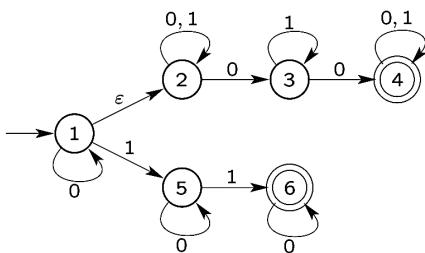
ε -NKA za jezik z abecedo a in b , ki vsebuje nize oblike $a^p b b a^q$ in $a^r b b b a^s$ za poljubne $p \geq 0, q \geq 0, r \geq 0$ in $s \geq 0$.



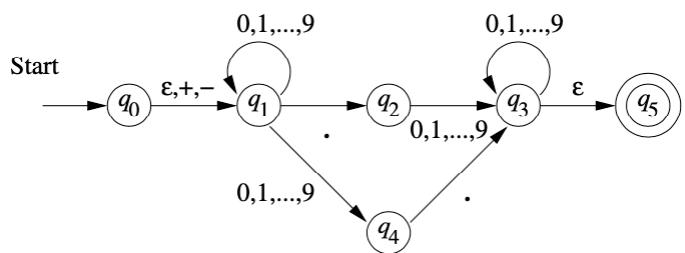
NKA- ε , ki sprejema besede $w \in \{0, 1\}^*$, ki vsebuje poljubno mnogo nizov 1010 in 1001 v poljubnem zaporedju.



NKA- ε , ki sprejema besede $w \in \{a, b\}^*$, ki vsebujejo najmanj 2 ničel ali točno 2 enici.

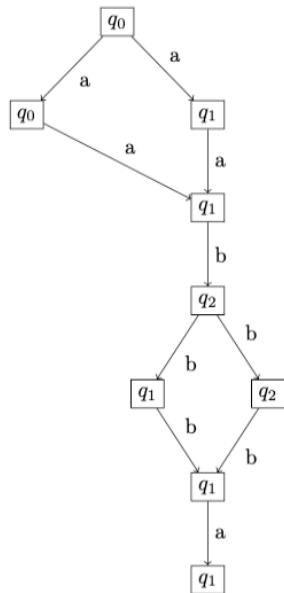
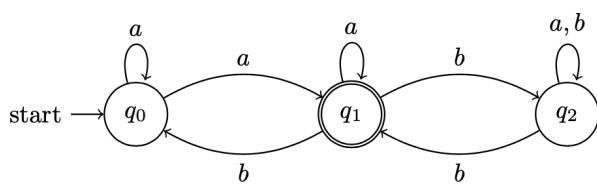


NKA- ϵ , ki sprejema decimalna števila.



	ϵ	$+, -$.	$0, 1, \dots, 9$
q_0	$\{q_1\}$	$\{q_1\}$	\emptyset	\emptyset
q_1	\emptyset	\emptyset	$\{q_2\}$	$\{q_1, q_4\}$
q_2	\emptyset	\emptyset	\emptyset	$\{q_3\}$
q_3	$\{q_5\}$	\emptyset	\emptyset	$\{q_3\}$
q_4	\emptyset	\emptyset	$\{q_3\}$	\emptyset
q_5	\emptyset	\emptyset	\emptyset	\emptyset

Za spodaj podani NKA narišite vse možnosti (kot drevo izvajanja) za sprejetje besede **aabbba**.



Končni avtomati in regularni izrazi

Regularni izrazi

Jezik je množica nizov sestavljenih iz simbolov iz abecede Σ .

Regularni izrazi so formalizem za opis regularnih jezikov.

Z **L(izraz)** bomo označili jezik, ki ga predstavlja podani regularni izraz.

Izhodiščni regularni izrazi: ϵ, \emptyset, a

- $L(\epsilon) = \{\epsilon\}$
- $L(\emptyset) = \emptyset$
- $L(a) = \{a\}$

Pravila za gradnjo regularnih izrazov

- unija (v jeziku L_1 ali v jeziku L_2 ali v obeh hkrati)

$$L(\epsilon + 1) = \{\epsilon, 1\}$$

$$L(\emptyset + R) = L(R)$$

- stik

$$L(01) = \{01\}$$

$$L((0+1) 1 (0+1)) = \{010, 011, 110, 111\}$$

$$L(\epsilon R) = L(R)$$

- Kleenova ovojnica

$$L(1^*) = \{\epsilon, 1, 11, 111, 1111, \dots\}$$

$$L((01)^*) = \{\epsilon, 01, 0101, 010101, \dots\}$$

$$L((0+1)^*) = \epsilon \cup L(0+1) \cup L((0+1)(0+1)) \cup \dots = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\} = \Sigma^*$$

Regularni izraz vseh besed nad abecedo {0, 1}, ki vsebuje podniz 111.

Vsaka beseda iz jezika na nekem mestu vsebuje podniz 111.

Rešitev: $(0 + 1)^* 111 (0 + 1)^*$

Regularni izraz za jezik nad abecedo {0, 1}, ki vsebuje kvečjemu en par zaporednih enic.

$$R = R_0 + R_1$$

R_0 : regularni izraz za besede brez zaporednih enic (konča z 0 ali 1)

$$R_0 = (0 + 10)^* + (0 + 10)^* 1$$

Uporabimo pravilo $AB + AC = A(B + C)$

$$R_0 = (0 + 10)^*(\epsilon + 1)$$

R_1 : regularni izraz za besede z natanko enim parom zaporednih enic

$$R_1 = (0 + 10)^* 11 (0 + 01)^*$$

$$R = (0 + 10)^*(\epsilon + 1) + (0 + 10)^* 11 (0 + 01)^* = (0 + 10)^*(\epsilon + 1 + 11 (0 + 01)^*)$$

Podana sta dva jezika: $L_1 = \{aab, aa, bab\}$ in $L_2 = \{aab, bbb\}$.

a) $L_1 \cup L_2$

$$L_1 \cup L_2 = \{aab, aa, bab, bbb\}$$

b) $L_1 L_2$

$$L_1 L_2 = \{aabaab, aabbbb, aaaab, aabbb, babaab, babbabb\}$$

c) $L_2 L_1$

$$L_2 L_1 = \{aabaab, aabaa, aabbab, bbbaab, bbbaa, bbbbab\}$$

d) L_1^* (nekaj primerkov besed v jeziku)

$$L_1^* = \{aab + aa + bab\}^* = \{\epsilon, aa, aabab, aaaa, babbabbab \dots\}$$

Podan je jezik za regularne izraze nad $\Sigma = \{0, 1\}$:

a) $L_1 = L(0)$

$$L_1 = \{0\}$$

b) $L_2 = L(1^*)$

$$L_2 = \{\epsilon, 1, 11, 111, \dots\} // poljubno enic$$

c) $L_3 = L((11)^*)$

$$L_3 = \{\epsilon, 11, 1111, \dots\} // sodo mnogo enic$$

d) $L_4 = L(01 + (0 + 01)^*)$

$$L_4 = \{01, \epsilon, 0, 00, 0101, 001, 0001010, 010 \dots\}$$

// za vsako iteracijo posebaj izberemo (0001010 ... 0 0 01 01 0), pred vsako enico mora stati vsaj ena 0

e) $L_5 = L(1 + 0^*)$

$$L_5 = L(1 + 0^*) = \{\epsilon, 1, 0, 00, 000, 000 \dots\}$$

f) $L_6 = L((101)^* + 01(10)^*)$

$$L_6 = \{\epsilon, 101, 101101, 01, 0110, 011010, \dots\}$$

g) $L_7 = L(0 + 10^*)$

$$L_7 = L\{0, 1, 10, 100, 1000, 10000, \dots\}$$

Regularni izraz nad $\Sigma = \{0, 1\}$, ki opisuje vse besede, ki nimajo podniza 01.

Primeri: $\epsilon, 0, 1, 00, 11, 10$

$$L(1^*0^*)$$

Regularni izraz nad $\Sigma = \{0, 1\}$, ki opisuje vse besede v katerih se izmenično izmenjujeta 0 in 1.

$$(01)^* + (10)^* + 0(10)^* + 1(01)^*$$

ALI

$$(\epsilon + 1)(01)^*(\epsilon + 0)$$

Regularni izraz nad $\Sigma = \{a, b, c\}$, ki vsebuje najmanj en a in najmanj en b.

$$_ab_$$

$$_ba__$$

$$c^*a(a + c)^*b(a + b + c)^* + c^*b(b + c)^*a(a + b + c)^*$$

Regularni izraz nad $\Sigma = \{a, b\}$, ki opisuje vse besede, ki imajo vsaj dva neprekriavoča se podniza aa.

Regularni izraz nad $\Sigma = \{a, b\}$, ki opisuje vse besede, ki imajo vsaj dva neprekriavoča podniza aa ali dva prekrivajoča se podniza aa.

1) aa aa

2) aaa

$$(a + b)^*aa(a + b)^*aa(a + b)^* + (a + b)^*aaa(a + b)^*$$

$$L_1 = (a + b)^*aa(a + b)^*aa(a + b)^*$$

$$L_2 = L_1 + ((a + \epsilon)b)^*aaa(b(a + \epsilon))^*$$

Regularni izraz nad $\Sigma = \text{ASCII}$, ki opisuje vse veljavne email naslove.

$\text{ww}^*(\text{.ww}^*)^*\text{@}\text{ww}^*(\text{.ww}^*)^*(\text{.com + si})$

Naj bo A regularni izraz za črko abecede, ki nastopa v ASCII ($A = (a + b + \dots + y + z + A + B + \dots + Y + Z)$), in naj bo D regularni izraz za top level domain ($D = \text{com} + \text{org} + \text{si} + \dots$).

$L = A^+(\text{.A}^+)^*\text{@}A^+(\text{.A}^+)^*.D$

Regularni izrazi nad abecedo $\Sigma = \{a, b\}$:

1. Besede, ki se končajo z aa.

$(a + b)^*aa$

2. Besede, ki se ne končajo z aa.

$\varepsilon + a + b + (a + b)^*(ab + ba + bb)$

3. Besede, ki vsebujejo sodo mnogo znakov a.

$(b^*ab^*ab^*)^*$

4. Besede, ki vsebujejo liho mnogo znakov a.

$b^*ab^*(b^*ab^*ab^*)^*$

5. Besede, ki vsebuje sodo mnogo a in sodo mnogo b.

$(a(bb + ba(aa)^*ab)^* (a+ba(aa)^*b) + b(aa + ab(bb)^*ba)^* (b + ab(bb)^*a)$

6. Besede, ki vsebujejo najmanj 2 a-ja ali točno 2 b-ja.

$b^*ab^*a(a + b)^* + a^*ba^*ba^*$

7. Besede, v katerih se pojavijo največ trije znaki a.

$b^*(a + \varepsilon)b^*(a + \varepsilon)b^*(a + \varepsilon)b^*$

Pretvorbe med končnimi avtomati

Pretvorba NKA => DKA

Z NKA lahko marsikateri jezik opišemo bistveno lažje kot z DKA.

Kljub temu pa ta dva formalizma opisujeta enak razred jezikov.

Za vsak NKA obstaja DKA, ki opisuje isti jezik (in obratno).

Če ima NKA n stanj, ima enakovredni DKA do 2^n stanj (v praksi je število stanj pogosto manjše, ker niso nujno vsa dosegljiva iz začetnega stanja).

NKA A želimo pretvoriti v enakovreden DKA A',

$$A = (Q, \Sigma, \delta, q_0, F)$$

$$A' = (2^Q, \Sigma, \delta', \{q_0\}, F')$$

$$\delta'(\{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}, a) = \delta(q_{i_1}, a) \cup \delta(q_{i_2}, a) \cup \dots \cup \delta(q_{i_k}, a)$$

$$F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$

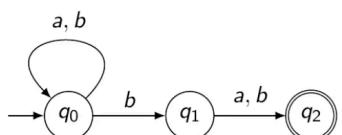
Vzdržujemo množico M generiranih stanj DKA.

Na začetku $M = \{\{q_0\}\}$.

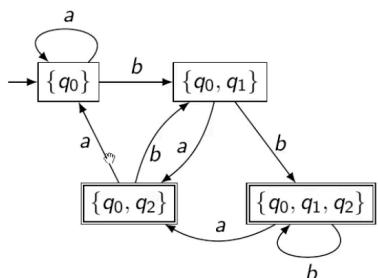
Ponavljamo, dokler ne obdelamo vseh stanj v M:

- izberemo še ne obdelano stanje S iz množice M
- izračunamo prehodno funkcijo (S, a) za vse simbole a in dobljena stanja dodamo v M

Primer



S	$\delta'(S, a)$	$\delta'(S, b)$
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$



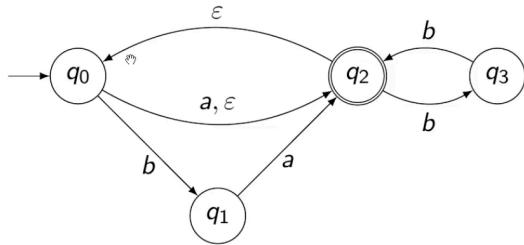
Pretvorba ε -NKA => NKA

ε -NKA je NKA z možnostjo spontanih prehodov.

Ta dodatek ne poveča izrazne moči formalizma.

ε -ovojnica stanja q

ε -CLOSURE(q) = { q } \cup { $r \in Q$ | od q do r obstaja pot s samimi ε -prehodi}



q	ε -CLOSURE(q)
q_0	{ q_0, q_2 }
q_1	{ q_1 }
q_2	{ q_0, q_2 }
q_3	{ q_3 }

ε -NKA A pretvorimo v NKA'

$$A = (Q, \Sigma, \delta, q_0, F)$$

$$A' = (Q, \Sigma, \delta', q_0, F')$$

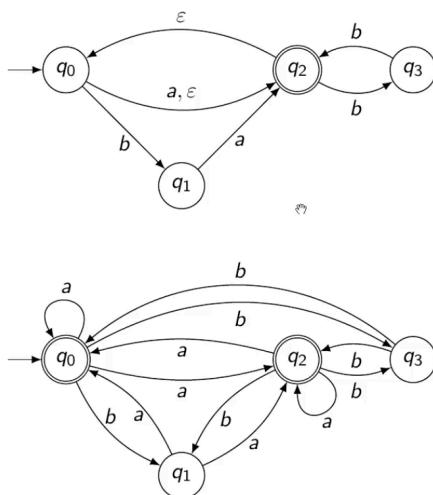
$$F' = \{q \mid \varepsilon\text{-CLOSURE}(q) \cap F \neq \emptyset\}$$

$$\text{Naj bo } \delta(\{q_{i_1}, \dots, q_{i_k}\}, a) = \delta(q_{i_1}, a) \cup \dots \cup \delta(q_{i_k}, a)$$

$$\begin{aligned} \text{Naj bo } &\varepsilon\text{-CLOSURE}(\{q_{i_1}, \dots, q_{i_k}\}) = \\ &\varepsilon\text{-CLOSURE}(q_{i_1}) \cup \dots \cup \varepsilon\text{-CLOSURE}(q_{i_k}) \end{aligned}$$

$$\delta'(q, a) = \varepsilon\text{-CLOSURE}(\delta(\varepsilon\text{-CLOSURE}(q), a))$$

Primer



q	$\delta'(q, a)$	$\delta'(q, b)$
q_0	{ q_0, q_2 }	{ q_1, q_3 }
q_1	{ q_0, q_2 }	\emptyset
q_2	{ q_0, q_2 }	{ q_1, q_3 }
q_3	\emptyset	{ q_0, q_2 }

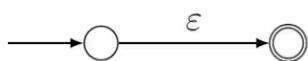
Pretvorba regularnega izraza v ε -NKA

Poljuben regularni izraz je mogoče pretvoriti v enakovreden ε -NKA in obratno.

Regularni izrazi zato opisujejo isti razred jezikov kot končni avtomati (regularni jeziki).

Pravila za pretvorbo izhodiščnih regularnih izrazov ε , \emptyset , a.

- **ε -NKA za jezik ε**



- **ε -NKA za jezik \emptyset**

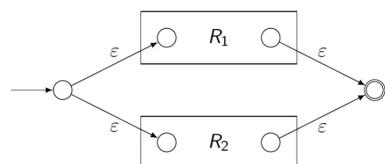


- **ε -NKA za jezik a**

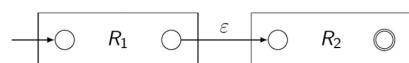


Pravila za pretvorbo izrazov $R_1 + R_2$, R_1R_2 in R^*

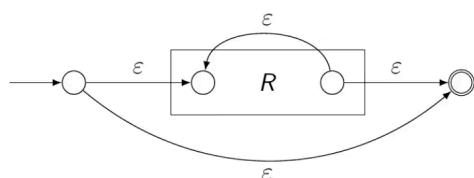
- ε -NKA za jezik $R_1 + R_2$



- ε -NKA za jezik R_1R_2

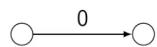


- ε -NKA za jezik R^*

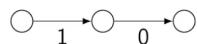


Regularni izraz $(0 + 10)^*(\varepsilon + 1)$ v enakovreden ε -NKA.

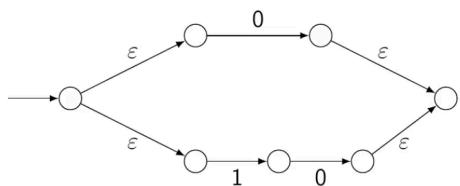
$r = 0$



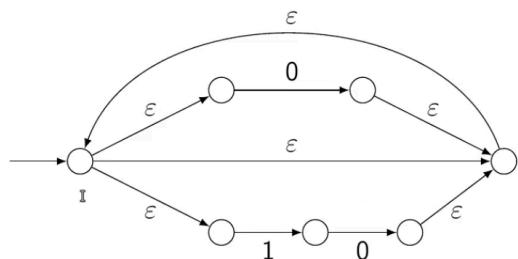
$r = 10$



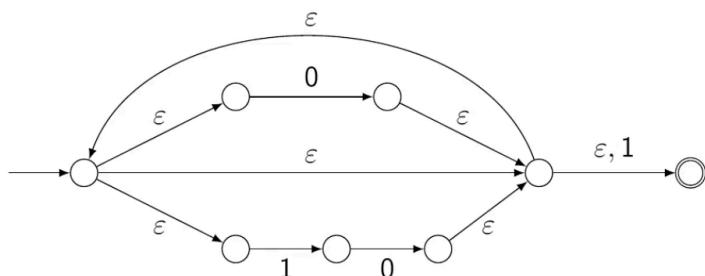
$r = (0 + 10)$



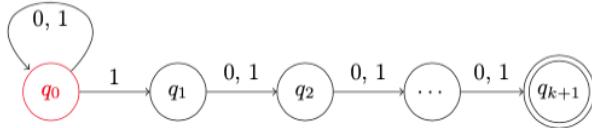
$r = (0 + 10)^*$



$r = (0 + 10)^*(\varepsilon + 1)$



Subset construction za dan NKA N sestavi tak DKA D, da je $L(D) = L(N)$. V praksi je mnogo stanj tako dobljenega DKA-ja mrtvih, zato jih lahko pobrišemo, končni DKA pa ima število stanj linearno v številu stanj NKA-ja. Poišči primer NKA-ja N, za katerega ima vsak DKA D, ki sprejme enak jezik kot N, vsaj $2^{|Q_N|-1}$ stanj. Z drugimi besedami iščemo kak primer, ki nas prisili, da imamo ekvivalenten DKA število stanj eksponentno v številu stanj NKA-ja.



$|Q_N|$ je torej $k + 2$ in dokazali smo, da mora ustrezeni DKA imeti vsaj $2^{k+1} = 2^{|Q_N|-1}$ stanj.

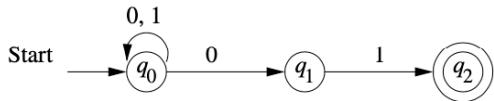
Podoben primer je na primer jezik L nizov s črkami iz $A = \{a_1, a_2, \dots, a_n\}$, ki vsebujejo največ $n - 1$ različnih črk iz A (se pravi $L = \sum_{i=0}^n (A - a_i)^*$).

NKA za ta jezik je sestavljen iz začetnega stanja q_0 , ki ima prehode v vsako od stanj p_1, \dots, p_n , kjer je p_i sprejemajoče stanje za jezik vseh nizov, ki ne vsebujejo a_i (se pravi, ima zanko za vsak $a_j \neq a_i$); in sicer pri črkah $a_j \neq a_i$ za stanje p_i (to bi lahko bili tudi ϵ -prehodi). NKA ima torej $n + 1$ stanj.

Vendar DKA mora imeti vsaj 2^n stanj, kar lahko dokažemo z uporabo Myhill-Nerodovega izreka. L lahko razbijemo na ekvivalentne razrede relacije R , kjer velja xRy natanko tedaj, ko noben od teh dveh nizov ne vsebuje črk, ki jih drug tudi ne bi vseboval. Če velja xRy in $x \notin L$, kar pomeni, da x vsebuje vseh n črk in posledično tudi $y \notin L$, je čisto vseeno kakšen z jima dodamo, kajti v vsakem primeru bosta še vedno vsebovala vse črke in bo veljalo $xz \notin L$ ter $yz \notin L$. Če velja xRy in $x \in L$ ter posledično $y \in L$, bo veljalo $xz \in L$ in $yz \in L$ natanko tedaj, ko z ne vsebuje vseh črk, ki x (in y) manjkajo do vseh črk A . Torej za vse primere velja $xzRyz$, kar pomeni, da je R res desno invariantna.

L je unija ekvivalentnih razredov nizov, ki imajo od 0 do n različnih črk iz A (razredi z od 0 do $n - 1$ črk bodo ustreznati sprejemajočim stanjem v DKA). Teh je 2^n . Po Myhill-Nerodovem izreku je to tudi minimalno število stanj v DKA.

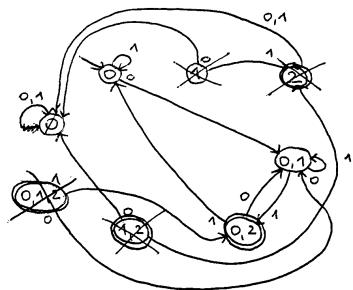
Pretvori NKA, ki sprejema nize, ki se končajo z 01 v DKA.



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	\emptyset	$\{q_2\}$
$*q_2$	\emptyset	\emptyset

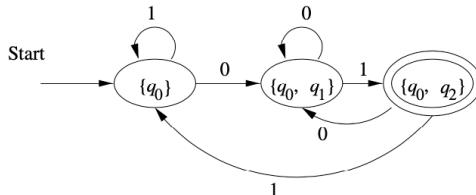
	0	1
\emptyset	\emptyset	\emptyset
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	\emptyset	$\{q_2\}$
$*\{q_2\}$	\emptyset	\emptyset
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	\emptyset	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$*D$	A	A
E	E	F
$*F$	E	B
$*G$	A	D
$*H$	E	F



Dosežemo lahko samo stanja B, E in F. Drugih 5 stanj je nedosegljivih od začetnega stanja.

	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$

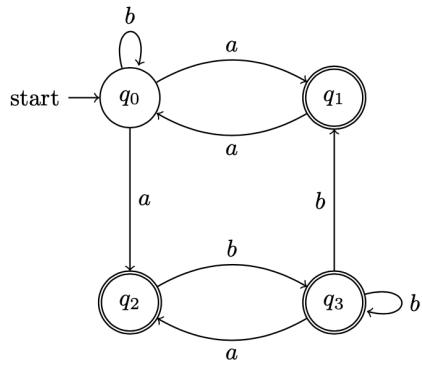


Prečrtamo stanja:

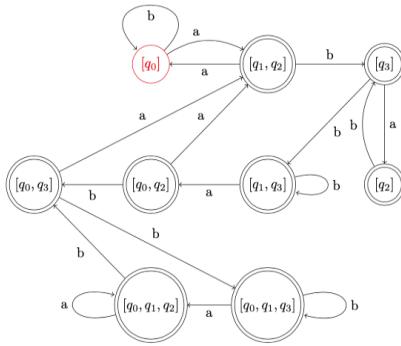
- gredo v prazno stanje
- nimajo vhodov (vhodnih povezav) ... primer stanje 0, 1, 2
- nimajo povezav za vse vhodne simbole

V praksi ne računamo vseh stanj. Po potrebi dodajamo stanja (tisto, kar se pojavi na desni in še ni na levi, dodamo na levo in izračunamo).

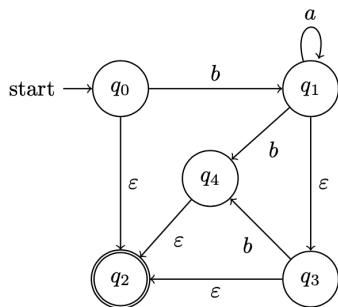
Spodaj podani NKA pretvorite v DKA.



	a	b
$\{\epsilon\}$	$\{\epsilon\}$	$\{\epsilon\}$
$\{q_0\}$	$\{q_1, q_2\}$	$\{q_0\}$
$\{q_1\}$	$\{q_0\}$	$\{\epsilon\}$
$\{q_2\}$	$\{\epsilon\}$	$\{q_3\}$
$\{q_3\}$	$\{q_2\}$	$\{q_3\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_1\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_1, q_2\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$



Spodaj podani ϵ -NKA pretvorite najprej v NKA, nato pa še v DKA.



ϵ -NKA => NKA

Če lahko z samo epsilon prehodi pridemo v končno stanje, potem je tudi to stanje končno.

$$\epsilon\text{-CL}(q_0) = \{q_0, q_3\}$$

$$\epsilon\text{-CL}(q_1) = \{q_1, q_2, q_4\}$$

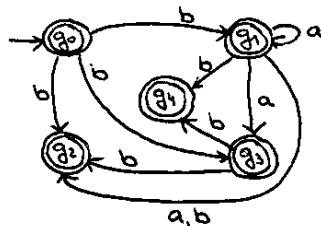
$$\epsilon\text{-CL}(q_2) = \{q_2\}$$

$$\epsilon\text{-CL}(q_3) = \{q_2, q_3\}$$

$$\epsilon\text{-CL}(q_4) = \{q_2, q_3\}$$

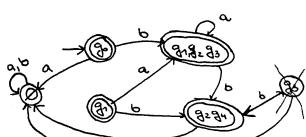
$$\epsilon\text{-CL}(\delta(\epsilon\text{-CL}(q), a))$$

q	a	b
q_0	\emptyset	q_1, q_2, q_3
q_1	q_1, q_2, q_3	q_2, q_4
q_2	\emptyset	\emptyset
q_3	\emptyset	q_2, q_4
q_4	\emptyset	\emptyset



NKA => DKA

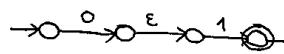
	a	b
q_0	\emptyset	q_1, q_2, q_3
q_1	q_1, q_2, q_3	q_2, q_4
q_2	\emptyset	q_2, q_4
q_3	\emptyset	\emptyset
q_4	q_1, q_2, q_3	q_2, q_4
q_1, q_2, q_3	q_1, q_2, q_3	q_2, q_4
\emptyset	\emptyset	\emptyset



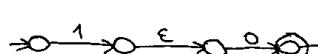
Začetno stanje je tudi $q_2 \Rightarrow$ začetno stanje mora biti
0, 2
ni q_1

Regуларни израз $(01)^*(10)^*$ по правilih pretvorite v ϵ -NKA. Nato ga poskusite še čim bolj poenostaviti.

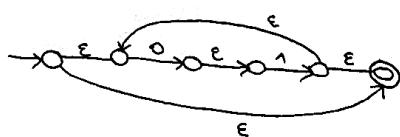
$$r = 01$$



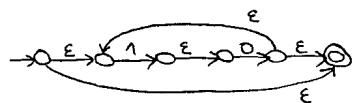
$$r = 10$$



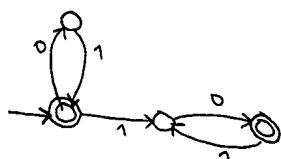
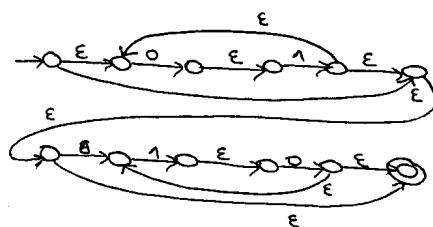
$$r = (01)^*$$



$$r = (10)^*$$

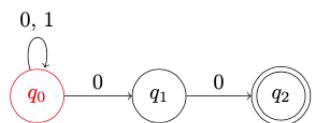
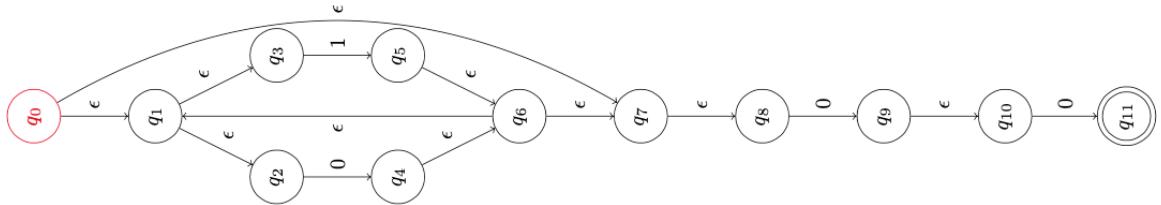


$$r = (01)^*(10)^*$$



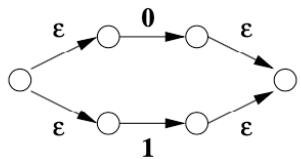
Regуларни израз в ϵ -NKA

- $(0 + 1)^*00$

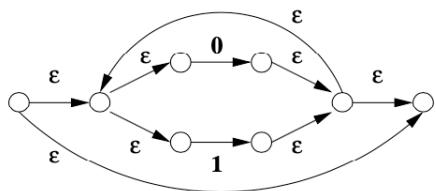


- $(0 + 1)^*1(0 + 1)$

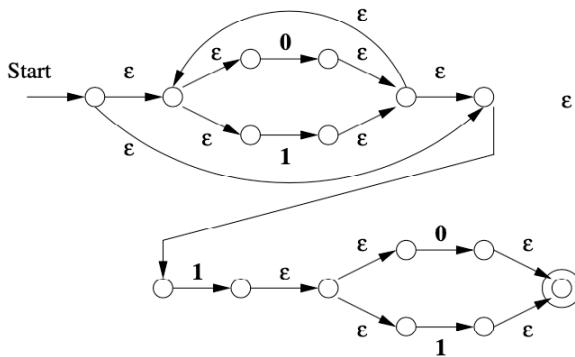
$0 + 1$



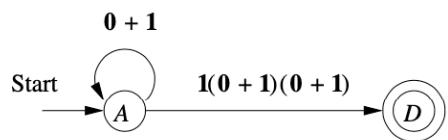
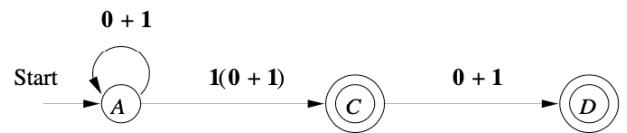
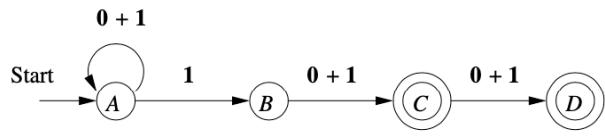
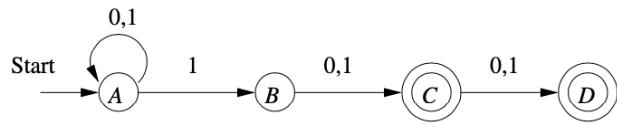
$(0 + 1)^*$



$(0 + 1)^*1(0 + 1)$



Podani NKA spremeni v regularni izraz.

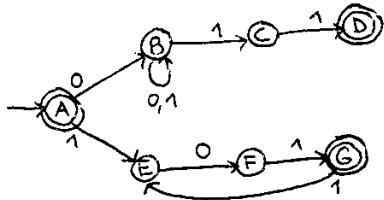


$$(0 + 1)^*1(0+1) + (0 + 1)^*1(0 + 1)(0 + 1)$$

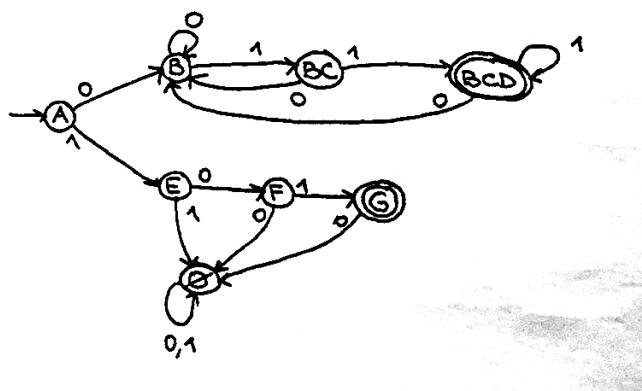
Podan je regularni izraz $0((1+0)^*11) + (101)^*$.

- Zapišite DKA, ki sprejema isti jezik, kot ga opisuje zgornji regularni izraz.
- Koliko besed dolžine 15 sprejema ta avtomat?
- Koliko besed dolžine 15 je v komplementu jezika, podanega z zgornjim regularnim izrazom?
- Zapišite za poljuben n, koliko besed dolžine n je v komplementu jezika, podanega z zgornjim regularnim izrazom.

a)



	0	1
\emptyset	\emptyset	\emptyset
$\xrightarrow{} A$	B	\emptyset
<u>B</u>	\emptyset	BC
BC	\emptyset	BCD
<u>BCD</u>	B	BCD
<u>E</u>	F	\emptyset
<u>F</u>	\emptyset	G
<u>G</u>	\emptyset	\emptyset



A je tudi končno stanje.

$$0(1+0)^*11 + (101)^*$$

Fiksni simboli: 0(1+0)^*11 + (101)^*

Nefiksni simboli so tisti, ki imajo *.

Dožina besede je $15 \Rightarrow 2^{15}$

(101)^* ... na samo en način lahko zapišemo besedo dolžine 15

Če želimo zapisati različne besede, ne smemo upoštevati fiksnih simbolov.

b) Leva stran: 2^{12} (dolžina besede - št. fiksnih simbolov)

Desna stran: 1

$$2^{12} + 1$$

c) Komplement zamenja končna stanja.

$$2^{15} - (2^{12} + 1)$$

d)

$$L: 2^{n-3}$$

$$D: \text{če } 3 \mid n \dots 1$$

sicer ... 0

n	$\#(n)$	$\#_k(n)$
0	1	$2^n - 1$
1	0	2^n
2	0	2^n
$3 \mid n$	$2^{n-3} + 1$	$2^n - 2^{n-3} - 1$
$3 \nmid n$	2^{n-3}	$2^n - 2^{n-3}$

N1 DEČIJO

$$\#(n) = \begin{cases} 2^{n-1} &; n=0 \\ 2^n &; n=1 \vee n=2 \\ 2^n - 2^{n-3} - 1 &; n=3k \\ 2^n - 2^{n-2} &; n \neq 3k \end{cases}$$

Lema o napihovanju za regularne jezike

Dokazovanje, da nekateri jeziki niso regularni.

$$\begin{aligned}
 & \forall L \in R \exists \quad \exists n \\
 & \nexists w \in L, |w| \geq n \\
 & \exists xyz, w = xyz, |xy| \leq n, |y| > 0 \\
 & \forall i \geq 0 \Rightarrow xy^i z \in L
 \end{aligned}$$

Za vsak jezik, ki je regularen obstaja nek n (za vsak jezik posebaj definiran, npr. število stanj v avtomatu, ki razpoznavata jezik).

Vsaka beseda, ki je v tem jeziku in je daljša ali enaka n , bo šla v avtomatu v nek cikel.

Za to besedo zagotovo obstaja neka trojka xyz , da je lahko beseda razstavljena na 3 kose (začetek je manjši ali enak n , y predstavlja cikel in je večji od 0).

Po ciklu lahko poljubno mnogokrat krožimo (do končnega stanja vedno pridemo).

Če jezik pripada regularnim jezikom, potem lahko za njega sestavimo končni avtomat.

Če imamo avtomat, ki ima n stanj, pomeni, da vse besede, ki so dolge n ali več, bodo v tem avtomatu morale iti v cikel.

Zanima nas 1. cikel, ki se bo pojavil na poti.

$|xy| \leq n \dots$ samo eno stanje se ponovi (drugače bi lahko dobili krajšo pot)

$|y| > 0 \dots$ cikel obstaja

Vse ostalo je lahko poljubno.



Uporaba

Za dokazovanje, da jezik L ne pripada regularnemu jeziku (ne moremo sestaviti končnega avtomata).

Dokazati moramo, da noben končni avtomat ni sposoben razpoznavati danega jezika.

Lema počne to s protislovjem.

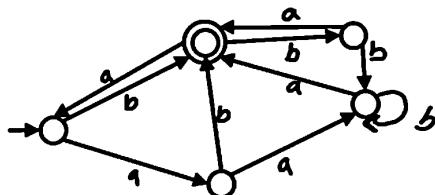
Za nek jezik pokaže, da če bi lema veljala, potem bi avtomat vedno sprejemal še nekaj besed, ki jih ne bi smel.

1. Izberemo besedo $w: |w| \geq n$

V njej mora nastopati n (n iz leme). Predpostavimo, da je jezik L regularen in da za njega obstaja nek n .

2. Za vsako možno delitev (ki ustreza omejitvam) besede na 3 komponente. S tem testiramo vse možne avtomate in načine, kam bi lahko cikel y postavili. Za vsako tako delitev moramo pokazati, da obstaja nek i (nek prehod po tem ciklu), ki nam besedo xy^iz vrže iz jezika.

3. Ni res, da za vsak i ostanemo znotraj jezika. Ne glede na to, kakšen avtomat zgradimo.



V končnem avtomatu moramo najprej identificirati n . N je najbolj enostavno število stanj avtomata (v tem primeru je 5). Prepričati se moramo, da za vse besede, ki so dolge 5 ali več, gre avtomat v cikel, medtem ko sprejema besedo.

Vzamemo neko besedo, ki je dolga 5 in je v jeziku: abba.

Besedo razbijemo na xyz , da bo veljalo: $|xy| \leq 5$, $|y| > 0$. Cikel se zgodi v zadnjih treh znakih.

Razbitje besede na 3 komponente:

$$x = ab, y = bba, z = \epsilon$$

Za jezik $a^*b(ba)^*a$ najdi n iz leme o napihovanju. Pokaži na dveh dovolj dolgih besedah, katera delitev ohranja napihnjeno besedo v jeziku.

Poiskati moramo najmanjši možni n.

Za vse besede, ki so daljše od 0 mora veljati, da obstaja nek cikel.

Cikel je dolg vsaj en znak in največ n znakov. Pojaviti se mora v prvih n znakih.

Potrebno je najti besedo, ki ne bo v jeziku.

$w = ba \dots$ fiksna beseda

- $n = 1$

$$a^*b(ba)^*a$$

$$w = ba \in L, |w| \geq 1$$

dolžina cikla bo 1:

1. ciklanje na b-ju:

$$x = \epsilon$$

$$y = b$$

$$z = a$$

$$w' = xy^iz$$

$$i = 2$$

$$bba \notin L$$

2. ciklanje na a-ju:

$$x = b$$

$$y = a$$

$$z = \epsilon$$

$$w' = xy^iz$$

$$i = 2$$

$$baa \notin L$$

Ugotovili smo, da se cikel ne mora pojaviti, ker pridemo v lemi do protislovja.

n ne more biti ena.

- $n = 2$

$$w = ba$$

$$x = \epsilon$$

$$y = ba$$

$$z = \epsilon$$

$$w' = xy^iz$$

$$i = 2$$

$$baba \notin L$$

Tudi pri $n = 2$, imamo eno besedo za katero lema ne velja.

- $n = 3$

$$w = aba$$

Vse možne delitve na x , y in z veljajo (so v jeziku).

$$w = aba, |w| \geq n = 3$$

$aba \Rightarrow aaba$ (cikel, ki je v jeziku), $aba \Rightarrow abba$, $aba = abaa$

Pomembno je, da pri eni besedi najdemo najdemo eno napihjeno besedo oz. en cikel, ki je v jeziku.

Če lema drži za vse možne besede dolžine 3, potem je очitno n enak 3.

Dokaži, da jezik $L = \{a^k b^k \mid k \geq 0\}$ ni regularen.

Končni avtomati ne moremo diktirati enakosti (npr. oklepajskih izrazov).

1. korak: izbira besede, ki je daljša od n

$$w = a^n b^n$$

$$|w| \geq n$$

2. korak: vsa možna razbitja besede (glede na omejitve postavimo različno x , y , z)

$$x = a^j$$

$$y = a^l$$

$$z = a^{n-j-l} b^n$$

$$xy^iz$$

$$i = 2:$$

$$xy^2z = a^{j+2l+n-j-l} b^n = a^{l+n} b^n$$

$$l > 0$$

L ni regularen (niz ni prave oblike)

$$i = 0:$$

$$xy^0z = a^{j+n-j-l} b^n = a^{n-l} b^n$$

$$l > 0$$

L ni regularen (niz ni prave oblike)

Dokaži, da $L = \{a^k b^m \mid k \neq m\}$ ni regularen.

Če komplement ($k = m$) ni regularen, potem tudi L ni.

Dokaži, da $L = \{1^p \mid p \text{ je praštevilo}\} \notin RJ$.

$w = 1^p$, $p \geq n$, p je praštevilo

$w = xyz$

$x = 1^i$

$y = 1^j$

$z = 1^{p-i-j}$

$$w' = xy^i z$$

$$i = p + 1: w = 1^{i+j(p+1)+p-i-j} = 1^{i+jp+j+p-i-j} = 1^{jp+p} = 1^{p(j+1)} \notin L$$

Pokazali smo za vse možne delitve, ker nismo povedali, koliko sta i in j .

Dokaži, da $L = \{a^m b^k \mid m < k\} \notin RJ$.

$w = xyz$

$w = a^n b^{n+1}$

$x = a^i$

$y = a^j$

$z = a^{n-i-j} b^{n+1}$

$$w = a^{i+j+b-i-j} b^{n+1}$$

$$i = 2:$$

$$w = a^{i+2j+n-i-j} b^{n+1} = a^{n+j} b^{n+1}$$

$$|j| \geq 1$$

$\Rightarrow L$ ni regularen

Dokaži, da $L = \{a^m b^k \mid m > k\} \notin RJ$.

$w = xyz$

$w = a^{n+1} b^n$

$x = a^i$

$y = a^j$

$z = a^{n+1-i-j} b^n$

$$w = a^{i+j+n+1-i-j} b^n$$

$$i = 2:$$

$$w = a^{i+2j+n+1-i-j} b^n = a^{n+j+1} b^{n+1}$$

$$|j| \geq 1$$

$\Rightarrow L$ ni regularen

ali

$$i = 0:$$

$$w = a^{i+n+1-i-j} b^n = a^{n-j+1} b^{n+1}$$

$$|j| \geq 1$$

$\Rightarrow L$ ni regularen

Dokaži, da $L = \{xx^R \mid x \in (0+1)^*\} \notin RJ$. Pri čemer je obrat niza x^R . Z drugimi besedami, to je jezik palindromov sode dolžine.

$$w = 0^{2n}$$

$w' = 0^{2n+(i-1)j} //$ ni dobra beseda

$$w = 0^n 1 1 0^n$$

$$x = 0^i$$

$$y = 0^j$$

$$z = 0^{n-i-j} 1 1 0^n$$

$$i = 2:$$

$$w = 0^{i+2j+n-i-j} 1 1 0^n = 0^j 1 1 0^n$$

$$j \geq 1$$

Jezik L ni regularen.

Dokaži, da $L = \{a^{k^3} \mid k \in \mathbb{N}\} \notin RJ$.

$$w = 0^{n^3}$$

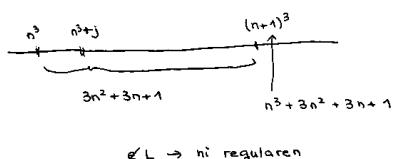
$$xy^iz = a^{n^3 + (i-1)l}$$

$$i = 2: a^{n^3 + 1}$$

$$1 \leq l \leq n$$

$n^3 + 1$ (v najboljšem primeru $n^3 + n$) ne more doseči naslednjega kuba

$$(n+1)^3 = n^3 + 3n^2 + 3n + 1$$



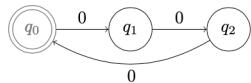
$\not\in L \rightarrow$ ni regularen

Dokaži, da je jezik $L = \{a^{3m} \mid m \in \mathbb{N}\}$ regularen (sestavi DKA) in pokaži, kako lema o napihovaju ne dokaže ničesar.

Pri tej nalogi moramo pokazati, da z lemo ne moremo dokazati, da jezik ni regularen, kar pa samo po sebi še ne pomeni, da je regularen.



V stanju q_m končajo nizi oblike 0^k , kjer je m ostanek pri deljenju k s 3.



Vzamemo niz $w = a^{3n}$, kjer je n konstanta iz leme. Ta niz je dolg vsaj n , kar pomeni, da ustreza pogoju za napihovanje. Sedaj moramo najti vsaj eno delitev $w = xyz$ (ob upoštevanju pogojev $|xy| \leq n$ in $|y| \geq 1$), tako da pri vseh napihovalnih eksponentih ostanemo v jeziku. (Pri dokazovanju, da jezik ni regularen, moramo za vsako možno delitev najti vsaj en napihovalni eksponent, pri katerem "pademo" iz jezika.)

Take delitve ni težko najti: $x = \epsilon$, $y = aaa$, $z = \text{vse ostalo}$.

Niz w' (i) = $xy^i z$ pripada jeziku za vsak eksponent i (tudi za $i = 0$), ker bo število a -jev še vedno ostalo večkratnik števila 3.

Najti moramo eno delitev, pri kateri za vse napihovalne eksponente ostanemo v jeziku.

$$L = \{a^i b^j c^{i+j} \mid i, j \geq 1\} \notin \text{RJ.}$$

Ni regularen, ker ne moremo šteti.

$$w = a^n b^n c^{2n}$$

$$x = a^i$$

$$y = a^j$$

$$z = a^{n-i-j}$$

$$w = a^{i+j+n-i-j} b^n c^{2n}$$

$$i = 2:$$

$$w = a^{i+2j+n-i-j} b^n c^{2n} = a^{j+n} b^n c^{2n}$$

$$|j| \geq 1$$

L ni regularen

Dokaži, da $L = \{b^{i^2} \mid i > 1\} \notin RJ$.

$$|w| = n^2 \geq n$$

$$n^2 + 1 \leq |xyz| \leq n^2 + 1$$

$$n^2 + 1 \leq |xyz| \leq n^2 + n + n + 1$$

$$n^2 + 1 \leq |xyz| \leq (n + 1)^2$$

$$= n^2 \leq |xyz| \leq (n + 1)^2$$

Niz leži med dvema zaporednima popolnima kvadratoma, ampak niz ni popolni kvadrat.

L ni regularen.

Dokaži, da $L = \{(10)^p 1^q \mid p, q \in N, p \geq q\} \notin RJ$.

$$w = (10)^n 1^n$$

$$|w| = 3n \geq n$$

$$w = xyz$$

Naj bo y predpona niza w z dolžino n .

$$y = (10)^{n/2} // n$$
 je sodo

$$y = (10)^{(n-1)/2} 1 // n$$
 je liho

Glede na pravili $w = xyz$ in $|xy| \leq n$

$$xy = (10)^j // 0 \leq j \leq n/2$$

$$xy = (10)^j 1 // 0 \leq j < n/2$$

Glede na pravilo $y \geq 1$ in glede na to, ali je $|xy|$ sodo ali liho

$$y \text{ je nek neprazni podniz } (10)^j, 0 \leq j \leq n/2$$

$$y \text{ je nek neprazni podniz } (10)^j 1, 0 \leq j < n/2$$

Za y so 3 možnosti

- y se začne z 0 in konča z 0

$$xy^0z = xz \text{ vsebuje } 110 \text{ kot podniz}$$

$$x = 1, y = 0, z = 10 // \text{ni del dokaza}$$

$$xy^0z \notin L \text{ (110 ni podniz nobenega niza v jeziku } L)$$

- y se začne z 1 in konča z 1

$$xy^0z = xz \text{ vsebuje } (10)^i 1 \text{ kot podniz}$$

$$x = 10, y = (10)^i 1, z = 0(10)^{n-i-2} 1^n$$

$$100 \text{ (100 ni podniz nobenega niza v jeziku } L)$$

- y se začne in konča z različnimi simboli

$$y = (10)^i \text{ ali } y = (01)^i$$

$$xy^0z = xz = (10)^{n-i} 1^n \notin L \text{ (n - i < n)}$$

L ni regularen.

$$L = \{yy \mid y \in \{0, 1\}^*\}$$

$$|w| \geq n$$

$$w = 0^n 0^n$$

$$w = xyz$$

$$x = \epsilon$$

$$y = 00$$

$$z = 0^{2n-2}$$

$$uv^iw = 0^{2k} = 0^k 0^k$$

Da beseda izpoljuje pogoj $|xy| \leq n$, mora y vsebovati samo ničle. Ampak potem ne izpoljuje pogoja $i \geq 0$, xy^iz .

Slaba beseda: $w = 0^n 0^n$

Ali ti pomeni, da je L regularen? NE! Izbrali smo slabo besedo w.

Dobro je, da imamo "mejnice" v besedi. V tem primeru sta to enici.

$$w = (01)^n (01)^n$$

$$x = \epsilon$$

$$y = 0101$$

$$z = (01)^{2n-2}$$

Ponovno smo izbrali slabo besedo.

$$w = 0^n 1 0^n 1$$

$$w = xyz$$

xy mora biti v prvi skupini 0^n

$$xy^0 z = 0^{n-|y|} 1 0^n 1$$

$|y|$ mora biti najmanj 1, kar pomeni, da ni oblike yy

$$L = \{a^n b^l a^k \mid k > n + l\}$$

$$w = a^n b a^{n+2}$$

$$w = xyz$$

$$x = a^i$$

$$y = a^m, m > 0$$

$$z = a^{n-m-i} b a^{n+2}$$

$$i = 2:$$

$$w = a^{i+3m+n-m-i} b a^{n+2} = a^{2m+n} b a^{n+2}$$

$$m \geq 1$$

=> ni regularen (niz ni prave oblike)

$$L = \{a^i b^j a^{ij} \mid i, j \geq 0\}$$

$$w = a^n b^n a^n$$

$$i = 0:$$

$$xy^0z = xz = a^{n-y} b^n a^n \text{ ni regularen}$$

$$L = \{b^2 a^n b^m a^3 \mid m, n \geq 0\}$$

Jezik je regularen, ker lahko sestavimo regularni izraz $R = b^2 a^* b^* a^3$

$$L = \{a^n b a^{3n} \mid n \geq 0\}$$

$$w = a^n b a^{3n}$$

$$y = a^k, k > 0$$

Niz ni prave oblike, $a^{n-k} b a^{3n}$

=> ni regularen

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

$$w = a^n b^n c^n$$

$$y = a^k, k > 0$$

Niz ni prave oblike, $a^{n-k} b^n c^n$

=> ni regularen

$$L = \{a^i b^n \mid i, n \geq 0, i = n \text{ ali } i = 2n\}$$

$$w = a^n b^n$$

$$y = a^k, k > 0$$

Niz ni prave oblike, $a^{n-k} b^n$

=> ni regularen

$$L = \{c^{2n}a^n b^{2n} \mid n \geq 0\}$$

$$w = c^{2n}a^n b^{2n}$$

$|xy| \leq n \dots x$ in y imata samo c -je

z ima preostanek c -jev, ki mu sledi $a^n b^{2n}$

$$x = c^j, j \geq 0$$

$$y = c^k, k \geq 0$$

$$z = c^l a^n b^{2n}, j + k + l = 2n$$

$$|y| \Rightarrow k > 0$$

$i = 2:$

$$xy = c^j c^k c^l a^n b^{2n} = c^{2n+k} a^n b^{2n}$$

$$j + k + l = 2n \Rightarrow L \text{ ni regularen (ni prave oblike)}$$

$$L = \{a^{2n}c^{3n}b^{2n} \mid n \geq 0\}$$

$$w = a^{2n}c^{3n}b^{2n}$$

Prvih n simbolov besede w so samo a -ji $\Rightarrow x$ in y bosta vsebovala samo a -je, z pa preostanek a -jev ter $c^{3n}b^{2n}$.

$$|y| > 0 \Rightarrow y \text{ bo imel vsaj en } a$$

$$x = a^j, j \geq 0$$

$$y = a^k, k \geq 1$$

$$z = a^m c^{3n} b^{2n}, m \geq 0$$

$$a^{2n}c^{3n}b^{2n} = w = xyz = a^j a^k a^m c^{3n} b^{2n} = a^{j+k+m} c^{3n} b^{2n}$$

$$j + k + m = 2n$$

$i = 2:$

$$xy^2z = a^j a^k a^k a^m c^{3n} b^{2n} = a^{2n+k} c^{3n} b^{2n}$$

$$k \geq 1$$

$$L \text{ ni regularen (ni prave oblike)}$$

$$L = \{b^i c^j a^k \mid i, j, k \geq 0 \text{ in } i + j = k\}$$

$$w = b^n c^n a^{2n}$$

Prvih n simbolov w -ja so b -ji $\Rightarrow x$ in y bosta vsebovala samo b -je, z pa preostanek b -jev ter $c^n a^{2n}$.

$$|y| > 0 \Rightarrow y \text{ bo imel vsaj en } b$$

$$x = b^j, j \geq 0$$

$$y = b^k, k \geq 1$$

$$z = b^m c^n a^{2n}, m \geq 0$$

$$b^n c^n a^{2n} = w = xyz = b^j b^k b^m c^n a^{2n} = b^{j+k+m} c^n a^{2n}$$

$$j + k + m = n$$

$i = 2:$

$$xy^2z = b^j b^k b^k b^m c^n a^{2n} = b^{n+k} c^n a^{2n}$$

$$k \geq 1$$

$$L \text{ ni regularen (ni prave oblike - število } b\text{-jev plus število } c\text{-jev ni enako številu } a\text{-jev)}$$

L = {www | w je sestavljen iz {a, b}*}

w = aⁿbⁿbⁿ

Prvih n simbolov niza w so a-ji => x in y bosta sestavljeni iz a-jev, z pa iz preostanka a-jev in baⁿbaⁿb.

|y| > 0 => y bo imel vsaj en a

$$x = a^j, j \geq 0$$

$$y = a^k, k \geq 1$$

$$z = a^m b a^n b, m \geq 0$$

$$a^n b a^n b = w = xyz = a^j a^k a^m b a^n b = a^{j+k+m} b a^n b$$

$$j + k + m = n$$

i = 2:

$$xy^2z = a^j a^k a^k a^m b a^n b = a^{n+k} b a^n b$$

L ni regularen (ni prave oblike)

L = {ww | w je sestavljen iz {a, b}*}

w = aⁿbⁿbⁿ

Prvih n simbolov niza w so a-ji => x in y bosta sestavljeni iz a-jev, z pa iz preostanka a-jev in baⁿb.

|y| > 0 => y bo imel vsaj en a

$$x = a^j, j \geq 0$$

$$y = a^k, k \geq 1$$

$$z = a^m b a^n b, m \geq 0$$

$$a^n b a^n b = w = xyz = a^j a^k a^m b a^n b = a^{j+k+m} b a^n b$$

$$j + k + m = n$$

i = 2:

$$xy^2z = a^j a^k a^k a^m b a^n b = a^{n+k} b a^n b$$

L ni regularen (ni prave oblike, k ≥ 1)

L = {w ∈ Σ* | w = w^R in |w| je sode dolžine}.

w = aⁿbⁿbⁿ

$$x = a^j, j \geq 0$$

$$y = a^k, k \geq 1$$

$$z = a^l b b a^n, j + k + l = n$$

i = 2:

$$xy^2z = a^j a^k a^k a^l b b a^n = a^{n+k+l} b b a^n$$

L ni regularen (ni prave oblike - k > 0: število a-jev na začetku in na koncu ni enako, niz ni palindrom)

L = {bⁱa^j | i ≥ j}

w = bⁿaⁿ

Najmanjše konstante n za jezike (pumping length is the length of the whole string you are pumping):

- **0*101***

N mora biti večji od 2 - jezik vsebuje nize, ki so dolžine 2 ali več (10 ne more biti napihnjen).

Trdimo, da je $n = 3$, $|w| \geq 3$. Dokažimo:

1. $w = 0 \dots 010$: pred nizom 10 je vsaj ena 0

$x = \epsilon, y = 0, z = \text{preostali niz}$

2. $w = 101 \dots 1$: po nizu 10 sledi vsaj ena 1

$x = 10, y = 1, z = \epsilon$

3. $w = 0 \dots 0101 \dots 1$: pred nizom 10 je vsaj ena 0 in po nizu 10 je vsaj ena 1

$x = \epsilon, y = 0, z = \text{preostali niz}$

ali

$x = 010, y = 1, z = \epsilon$

- **0001***

Najmanjši n (s katerim lahko napihnjemo besedo) je 4.

Niz 0000 ne more biti napihnjen $\Rightarrow n$ mora biti večji od 3.

Niz w, ki ga napihujemo je dolžine 4 ali več, tako da vsebuje vsaj eno enico.

$x = 000, y = 1, z = \text{preostanek niza}$

- **0*1***

Najmanjši n je 1. Ne more biti 0, ker niz ϵ (ki je v jeziku) ne more biti napihjen.

vsaj 1: $x = \epsilon, y = 1, z = \epsilon$

vsaj 0: $x = \epsilon, y = 0, z = \epsilon$

vsaj 1 in 0: $x = 0, y = 1, z = \epsilon; x = \epsilon, y = 0, z = \text{preostanek niza}$

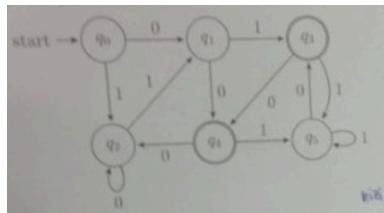
- **0*1+0+1* + 10*1**

Najmanjši n je 3. Ne more biti 2, ker niz 11 ne more biti napihjen.

- $0*1+0+1*$: $x = \epsilon, y = \text{prvi simbol besede}, z = \text{preostanek niza}$

- $10*1$: $x = 1, y = 0, z = \text{preostanek niza}$

Podan je DKA M:



Koliko je dolga najdaljša beseda, pri kateri avtomat med procesiranjem le-te ne naredi nobenega cikla? Katera beseda je to?

5 je najdaljša beseda brez cikla; 11010

Za jezik $a^*b(ba)^*a$ najdi n iz leme o napihovaju. Pokaži na dveh dovolj dolgih beseda, katera delitve ohranja napihnjeno besedo v jeziku.

$$n = \text{število fiksnih simbolov} + 1 = 2 + 1 = 3$$

Zato začnemo z $n = 3$:

1) vsaj ena a

$$x = \epsilon, y = a, z = \text{preostanek niza}$$

(napihujemo a)

=> je del jezika danega RI, saj ne skoči iz besede

primer: abbaa, aba

2) vsaj ena (ba) med nizom b in a

začetni niz = neko k-to število a-jev in naposled en b

(napihujemo ba)

$$x = \text{začetni niz} \quad y = ba \quad z = \text{preostanek niza}$$

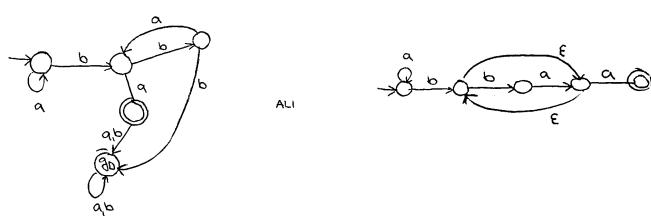
primer:

aaabbababaa

abbaa

$n = 3$, ker so vse besede v jeziku.

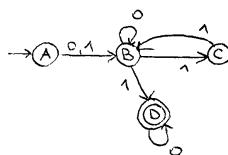
Lahko narišeš tudi avtomat.



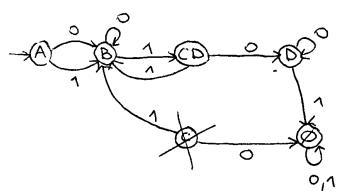
Z regularnim izrazom je podan jezik $L = (0 + 1)(11 + 0)^*10^*$.

a) Zapiši DKA za ta jezik.

b) Za L , koliko je najmanjša konstanta n iz leme o napihovanju za regularne jezike?



	0	1
A	B	B
B	B	CD
CD	D	B
C	∅	B
D	D	∅
∅	∅	∅



$$n = \text{število fiksnih simbolov} + 1 = 2 + 1 = 3$$

Dokaz

- $x = 0$ ali 1 , $y = 0$, $z = \text{preostanek niza} \Rightarrow \text{beseda je v jeziku}$
napihujemo y

n = 3

Preverimo, da lema velja za vse besede dolžine n , ki so v jeziku.

$n = 3$: 001, 010, 101, 110

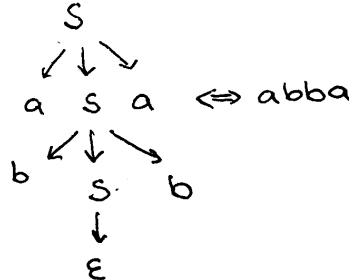
Kontekstno neodvisne gramatike

Kontekstno neodvisna gramatika za jezik palindromov nad abecedo $\Sigma = \{a, b\}$.

Gramatike rekurzivno gledajo na problem (v tem primeru palindrom).

Velike črke označujejo funkcije, mali znaki pa ustavitevne pogoje.

abaaba
 $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$
S palindrom
aSa S ... spet palindrom
 ϵ prazen niz je tudi palindrom



Kontekstno neodvisna gramatika za jezik pravilno vgnezdenih oklepajev.

Primeri: $\epsilon, (), (((),))$

$S \rightarrow (S) \rightarrow ((S)) \rightarrow (((S))) \rightarrow ((((),)))$

$S \rightarrow (S) \rightarrow (SS) \rightarrow ((S)(S)) \rightarrow (((),))()$

Rešitev:

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

$$V = \{S\}$$

$$T = \{(), ()\}$$

$$P: S \rightarrow \epsilon, S \rightarrow (S), S \rightarrow SS$$

$$S \rightarrow \epsilon \mid (S) \mid SS$$

Zapiši kontekstno neodvisno gramatiko za dane jezike:

- **0ⁿ1ⁿ**

$$S \rightarrow 0S1 \mid \epsilon$$

$$0011: S \rightarrow 0S1 \rightarrow 00S11 \rightarrow \epsilon$$

- **sodo število ničel**

Če je prvi simbol 1, imamo še vedno sodo število ničel.

Če je prvi simbol 0, potrebujemo še eno 0; potem imamo ponovno sodo število ničel.

$$S \rightarrow 1S \mid 0A0S \mid \epsilon$$

$$A \rightarrow 1A \mid \epsilon$$

ALI

$$S \rightarrow 1S \mid 0T \mid \epsilon$$

$$T \rightarrow 1T \mid 0S$$

- **(1(01)*)***

$$S = (1(01)*)^*$$

$$A = 1(01)^*$$

$$B = (01)^*$$

$$S \rightarrow \epsilon \mid A$$

$$A \rightarrow \epsilon \mid 1 B$$

$$B \rightarrow \epsilon \mid 0 1 B$$

- **00*11***

$$A = 0$$

$$B = 0^*$$

$$C = 1$$

$$D = 1^*$$

$$S \rightarrow ABCD$$

$$A \rightarrow 0$$

$$B \rightarrow 0B \mid \epsilon$$

$$C \rightarrow 1$$

$$D \rightarrow 1D \mid \epsilon$$

- binarni nizi, ki imajo sodo števil ničel in sodo število enic

$S \rightarrow 0X \mid 1Y \mid \epsilon$

$X \rightarrow 0S \mid 1Z$ (liho ničel, sodo enic)

$Y \rightarrow 1S \mid 0Z$ (liho enic, sodo ničel)

$Z \rightarrow 0Y \mid 1X$ (liho enic, liho ničel)

- $0^a 1^b 0^c, a + c = b$

$S \rightarrow TU$

$T \rightarrow 0T1 \mid \epsilon$ // prištevanje a-jev

$U \rightarrow 1U0 \mid \epsilon$ // prištevanje b-jev

Poglej število možnih primerov. Ker je b na sredini, izmenično prištevaš b in enega izmed a ali c.

a++, b++

c++, b++

- $L = \{w \in \{0,1\}^* \mid w \text{ vsebuje najmanj 3 enice}\}$

$S \rightarrow X1X1X1X$

$X \rightarrow 0X \mid 1X \mid \epsilon$

- $L = \{w \in \{0,1\}^* \mid w = w^R \text{ in } |w| \text{ je sod}\}$

$S \rightarrow 0S0 \mid 1S1 \mid \epsilon$

- $L = \{a^n b^n \mid n \geq 1\}$

$S \rightarrow aSb$

$S \rightarrow ab$

- $L = \{0^i 1^j \mid i \geq j\}$

$S \rightarrow 0S1 \mid Z$

$Z \rightarrow 0Z \mid \epsilon$

- $L = \{a^i b^j \mid i > j\}$

$S \rightarrow aSb \mid X$

$X \rightarrow aX \mid a$

- $L = \{a^i b^j \mid i \neq j\}$

$S \rightarrow aSb \mid A \mid B$

$A \rightarrow aA \mid a$ // a-jev več, b-jev manj

$B \rightarrow bB \mid b$ // b-jev več, a-jev manj

- $L = \{w \in \{0,1\}^* \mid w \text{ je lihe dolžine in sredinski simbol je } 0\}$

$S \rightarrow 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 0$

ali

$S \rightarrow ASB \mid 0$

$A \rightarrow 1 \mid 0$

$B \rightarrow 1 \mid 0$

- $L = \{a^ib^jc^k \mid i = j \text{ ali } j = k\}$

$\{a^ib^jc^k \mid i = j\}$

// X ... zagotovitev $a = b$, Y ... dodajanje c-jev

$Q \rightarrow aXbY \mid \epsilon$

$X \rightarrow aXb \mid \epsilon // a == b$

$Y \rightarrow cY \mid \epsilon$

$\{a^ib^jc^k \mid j = k\}$

// U ... zagotovitev $b = c$, W ... dodajanje a-jev

$Z \rightarrow WbUc \mid \epsilon$

$U \rightarrow bUc \mid \epsilon$

$W \rightarrow Wa \mid \epsilon$

// unija obeh pogojev

$S \rightarrow Q \mid Z$

- $L = \{a^ib^jc^k \mid i, j, k \geq 0 \text{ in } k = i + j\}$

$S \rightarrow aSc \mid X // \text{prištevanje a-jev in c-jev}$

$X \rightarrow bXc \mid \epsilon // \text{prištevanje b-jev in c-jev}$

- $L = \{a^ib^jc^k \mid i = j + k\}$

$S \rightarrow aSc \mid A // \text{prištevanje elementa, ki ga iščemo + stranski element}$

$A \rightarrow aAb \mid \epsilon // \text{skica}$

- $L = \{a^ib^jc^k \mid j = i + k\}$

$S \rightarrow AB$

$A \rightarrow aAb \mid \epsilon // a++, b++$

$B \rightarrow bBc \mid \epsilon // c++, b++$

- $L = \{a^i b^j c^k \mid i \neq j + k\}$

$$L = \{a^i b^j c^k \mid i \neq j + k\} = \{a^i b^j c^k \mid i > j + k\} + \{a^i b^j c^k \mid i < j + k\}$$

$$\{a^i b^j c^k \mid i > j + k\}$$

$S_1 \rightarrow aS_2$ // dodajanje a-jev

$S_2 \rightarrow aS_2 \mid aS_2c \mid S_3$ // dodajanje a-jev in c-jev

$S_3 \rightarrow aS_3 \mid aS_3b \mid \epsilon$ // dodajanje a-jev in b-jev

$$\{a^i b^j c^k \mid i < j + k\}$$

$S_4 \rightarrow S_5c$ // vemo, da je c zadnji simbol v nizu

$S_5 \rightarrow S_5c \mid aS_5c \mid S_6$

$S_6 \rightarrow S_6b \mid aS_6b \mid \epsilon$

$$\{a^i b^j c^k \mid i \neq j + k\}$$

$S_0 \rightarrow S_1 \mid S_4$

- $L = \{a^i b^j c^k \mid i \neq j \text{ ali } j \neq k\}$

$$\begin{array}{ccc} a \dots a & b \dots b & c \dots c \\ & i & j & k \end{array}$$

$A \rightarrow^* \text{poljubno zaporedje a-jev, a ... a}$

$C \rightarrow^* \text{poljubno zaporedje c-jev, c ... c}$

$S_1 \rightarrow^* a^i b^j \ (i > j)$ // poljubno zaporedje a-jev in b-jev

$S_2 \rightarrow^* a^i b^j \ (i < j)$

$S_3 \rightarrow^* b^j c^k \ (j > k)$

$S_4 \rightarrow^* b^j c^k \ (j < k)$

Rešitev

$S \rightarrow S_1C \mid S_2C \mid AS_3 \mid AS_4$

$A \rightarrow \epsilon \mid Aa$

$C \rightarrow \epsilon \mid Cc$

$S_1 \rightarrow a \mid aS_1 \mid aS_1b$ // a je vedno več kot b-jev

$S_2 \rightarrow b \mid S_2b \mid aS_2b$

$S_3 \rightarrow b \mid bS_3 \mid bS_3c$

$S_4 \rightarrow c \mid S_4c \mid bS_4c$

- $L = \{a^i b^j \mid i = 2j\}$

$S \rightarrow aaSb \mid \epsilon$

- $L = \{a^x b^y a^z \mid z = x + y\}$

$S \rightarrow aSa \mid L$

$L \rightarrow bLa \mid \epsilon$

- $L = \{a^x b^y a^z \mid z = x - y\}$

$S \rightarrow aSa \mid L$

$L \rightarrow aLb \mid \epsilon$

- **gramatika vseh nizov sestavljenih iz a in b, ki vsebujejo podniz baa**

$S \rightarrow LbaaL$

$L \rightarrow aL \mid bL \mid \epsilon$

- $L = \{w \mid w \text{ se začne in konča z enakim simbolom}\}$

$S \rightarrow 0A0 \mid 1A1 \mid \epsilon$

$A \rightarrow 0A \mid 1A \mid \epsilon$

- $L = \{w \mid |w| \text{ je lih}\}$

$S \rightarrow 0A \mid 1A$

$A \rightarrow 00A \mid 01A \mid 10A \mid 11A \mid \epsilon$

- $L = \{0^i 1^j 0^k \mid i + j \geq 2k\}$

$0 \dots 0 \underset{i}{1} \dots \underset{j}{1} 0 \dots 0 \underset{k}{0}$

Število začetnih ničel in enic mora biti vsaj 2-krat toliko kot je dolžina ničel na koncu.

$A \rightarrow^* 0 \dots 0$

$B \rightarrow^* 1 \dots 1 0 \dots 0$

Rešitev

$S \rightarrow AB \mid 00S0 \mid 0A1B0 \quad // \text{če dodamo spredaj 00 moramo zadaj dodati } 0^{**}$

$A \rightarrow \epsilon \mid 0A$

// lahko je prazen

// spredaj dodajamo poljubno število ničel ($i + j$ mora biti vsaj $2k$)

$B \rightarrow \epsilon \mid 1B \mid 11B0$

// lahko je prazen

// spredaj dodajamo poljubno število eni

// če dodamo zadaj 0 moramo dodati spredaj bodisi 2 enici ali 2 ničli**,

da ohranimo pravilo $i + j \geq 2k$

Opis gramatik:

- $S \rightarrow abS \mid a$

$(ab)^*a$

- $S \rightarrow aSb \mid \epsilon$

$a^n b^n, n \geq 0$

- $S \rightarrow bSb \mid A, A \rightarrow aA \mid \epsilon$

$b^n a^* b^n, n \geq 0$

- $S \rightarrow aSc \mid aBc, B \rightarrow DB \mid b \mid bB, D \rightarrow \epsilon \mid DD \mid B$

$a^n b^m c^n, m > 0, n > 0$

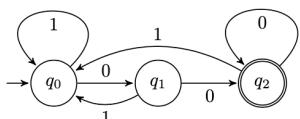
- $S \rightarrow AS \mid B, A \rightarrow aAc \mid Aa \mid \epsilon, B \rightarrow bBb \mid \epsilon$

$((a + c)^k)^* b^n$

$k \geq 0, k = i (\text{število } a\text{-jev}) + j (\text{število } c\text{-jev}), i \geq j, n = \text{sodo število}$

Nisi iz a-jev in c-jev z enako ali manj c-jev kot a-jev in noben prefix nima več c-jev kot a-jev, sledi sodo število b-jev.

Konstuiraj kontekstno neodvisno gramatiko za DFA:



$$S_0 \rightarrow 0S_1 \mid 1S_0$$

$$S_1 \rightarrow 0S_2 \mid 1S_0$$

$$S_2 \rightarrow 0S_2 \mid 1S_0 \mid \epsilon$$

Kontekstna neodvisna gramatika za jezik $0^*1(0+1)^*$ ter leva in desna izpeljava za niz 00101.

Najprej imamo poljubno število ničel, enica in nato poljubno dolgo zaporedje ničel in enic.

$$0^* \quad 1 \quad (0+1)^*$$

$$A \quad 1 \quad B$$

$$S \rightarrow A1B$$

$$A \rightarrow \epsilon \mid 0A$$

$$B \rightarrow \epsilon \mid 0B \mid 1B$$

Leva izpeljava: $S \Rightarrow A1B \Rightarrow 0A1B \Rightarrow 00A1B \Rightarrow 001B \Rightarrow 0010B \Rightarrow 00101B \Rightarrow 00101$

Desna izpeljava: $S \Rightarrow A1B \Rightarrow A10B \Rightarrow A101B \Rightarrow A101 \Rightarrow 0A101 \Rightarrow 00A101 \Rightarrow 00101$

Kontekstna neodvisna gramatika, ki generira jezik regularnega izraza $a(ab)^*bb(aa + b)^*a$.

$$\frac{a}{x} \frac{(ab)^*}{y} \frac{bb}{z} \frac{\overbrace{(aa+b)}^w)^*}{u} \frac{a}{v}$$

$S \rightarrow XYZUV$

$X \rightarrow a$

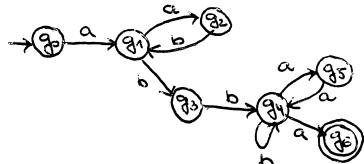
$Y \rightarrow \epsilon \mid abY$

$Z \rightarrow bb$

$U \rightarrow \epsilon \mid WU$

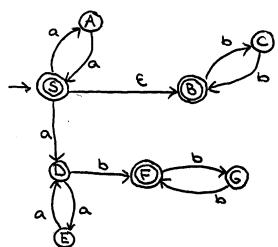
$W \rightarrow aa \mid b$

$V \rightarrow a$



Kontekstna neodvisna gramatika, ki generira jezik $L = \{a^n b^m \mid n + m \text{ je sodo}\}$.

RI: $(aa)^*(bb)^* + a(aa)^*b(bb)^*$



$S \rightarrow aA \mid B \mid aD \mid \epsilon$

$A \rightarrow aS$

$B \rightarrow bC \mid \epsilon$

$C \rightarrow bB$

$D \rightarrow aE \mid bF$

$E \rightarrow aD$

$F \rightarrow bG \mid \epsilon$

$G \rightarrow bF$

Dokaži, da so regularni jeziki vsebovani v kontekstno neodvisnih jezikih (tj., da za vsak regularni izraz E obstaja kontekstno neodvisna gramatika G, da je $L(G) = L(E)$).

RI	KNG
$R = \emptyset$	$S \rightarrow S$ $(P = \emptyset)$
$R = \epsilon$	$S \rightarrow \epsilon$
$R = a$	$S \rightarrow a$
$R = R_1 + R_2$	$S \rightarrow S_1 S_2$
$R = R_1 R_2$	$S \rightarrow S_1 S_2$
$R = R_1^*$	$S \rightarrow \epsilon SS$

Naj bo $T = \{0, 1, (,), \cup, *, \epsilon, \emptyset\}$ množica končnih simbolov.

- a) Napiši KNG, ki generira regularne izraze nad abecedo 0 in 1.

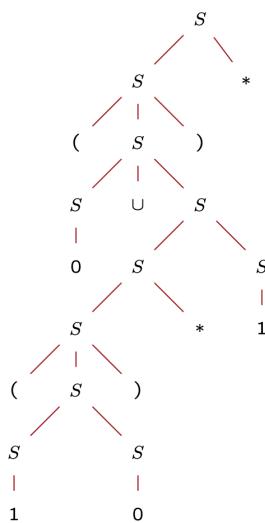
$$S \rightarrow S \cup S | SS | S^* | (S) | 0 | 1 | \emptyset | \epsilon$$

- b) Nariši drevo izpeljave za besedo $w = (0 + (10)^*1)^*$.

Terminali predstavljajo izraz.

Na vrhu je začetni simbol, nato sledi izpeljava.

$$\begin{aligned} S &\Rightarrow S^* \Rightarrow (S)^* \Rightarrow (S \cup S)^* \Rightarrow (0 \cup S)^* \Rightarrow (0 \cup SS)^* \Rightarrow (0 \cup S^*S)^* \\ &\Rightarrow (0 \cup (S)^*S)^* \Rightarrow (0 \cup (SS)^*S)^* \Rightarrow (0 \cup (1S)^*S)^* \\ &\Rightarrow (0 \cup (10)^*S)^* \Rightarrow (0 \cup (10)^*1)^* \end{aligned}$$



Naj bo L jezik binarnih nizov, ki imajo $3p$ ničel in $5q$ enic za neka $p, q \geq 0$.

- a) Zapiši KNG G , da bo $L(G) = L$.

$\epsilon, 000, 10110011, 01101011, 10001010011$

$A_{ij} \rightarrow^* 3p - i$ ničel, $5q - j$ enic ($i < 3, j < 5$) // število ničel po modulu 3 bo i , število enic po modulu 5 bo j

$S \rightarrow A_{00}$

$A_{00} \rightarrow \epsilon \mid 0A_{10} \mid 1A_{01}$

$A_{10} \rightarrow 0A_{20} \mid 1A_{11}$

$A_{20} \rightarrow 0A_{00} \mid 1A_{21}$ // če dodamo 3 ničle, smo spet na "začetku"

$A_{01} \rightarrow 0A_{11} \mid 1A_{02}$

$A_{11} \rightarrow 0A_{21} \mid 1A_{12}$

$A_{21} \rightarrow 0A_{01} \mid 1A_{22}$

$A_{02} \rightarrow 0A_{12} \mid 1A_{03}$

$A_{12} \rightarrow 0A_{22} \mid 1A_{13}$

$A_{22} \rightarrow 0A_{02} \mid 1A_{23}$

$A_{03} \rightarrow 0A_{13} \mid 1A_{04}$

$A_{13} \rightarrow 0A_{23} \mid 1A_{14}$

$A_{23} \rightarrow 0A_{03} \mid 1A_{24}$

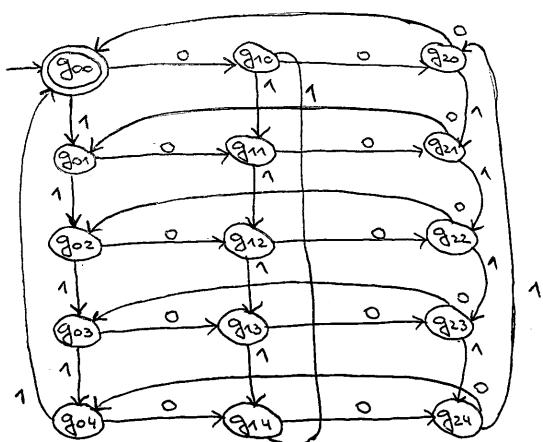
$A_{04} \rightarrow 0A_{14} \mid 1A_{00}$

$A_{14} \rightarrow 0A_{24} \mid 1A_{10}$

$A_{24} \rightarrow 0A_{04} \mid 1A_{20}$

- b) Ali je L regularen? Zakaj?

Jezik je regularen, ker lahko naredimo končni avtomat.



Za gramatiko, porojeno z $S \rightarrow SS \mid (S) \mid \epsilon$

- **poisci skrajno levo izpeljavo niza w = ((())) ()**

Vedno uporabiš najbolj levo spremenljivko.

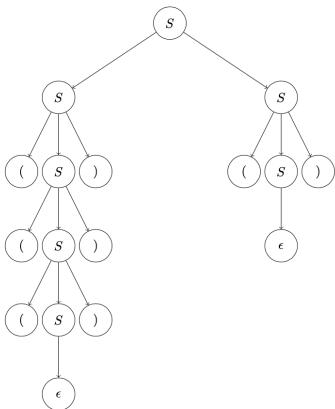
$S \rightarrow SS \rightarrow (S) S \rightarrow ((S)) S \rightarrow (((S))) S \rightarrow (((S))) (S) \rightarrow (((S))) ()$

- **poisci skrajno desno izpeljavo w = ((())) ()**

Vedno uporabiš najbolj desno spremenljivko.

$S \rightarrow SS \rightarrow S(S) \rightarrow S() \rightarrow (S)() \rightarrow ((S))() \rightarrow (((S)))() \rightarrow (((S)))()$

- **nariši drevo izpeljav**



- **vse možne izpeljave za w**

Vsaka izpeljava se začne z $S \rightarrow SS$. Za izpeljavo ((())) iz levega S potrebujemo 4 korake, za izpeljavo () iz desnega S pa 2.

Vsek korak lahko naredimo bodisi na začetku ali pa za katerim koli od korakov za izpeljavo ((())), se pravi imamo 5 možnosti.

Izmed teh možnosti izbiramo z zamenjavo in ker si morata koraka slediti v pravilnem vrstnem redu, je vseh možnih izpeljav (5×2) = 15.

- **primer nedvoumne gramatike**

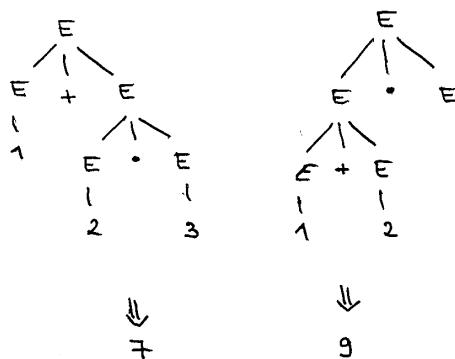
$S \rightarrow SS \mid (S) \mid ()$

Če za KNG G obstaja niz z dvema različnima drevesoma izpeljav, pravimo, da je G *dvoumna*. Jezik L je inhrenetno dvoumen, če je vsaka KNG G, za katero je $L(G) = L$, dvoumna. Ali je gramatika $E \rightarrow E + E \mid E * E \mid 0 \mid \dots \mid 9$ dvoumna?

Ker obstajata 2 različni drevesi izpeljav za en niz, je gramatika dvoumna. Zgoraj (v korenju) je "šibkejša" operacija.

$$E \rightarrow E + E \mid E \cdot E \mid 0 \mid \dots \mid 9$$

$$\omega = 1 + 2 \cdot 3$$



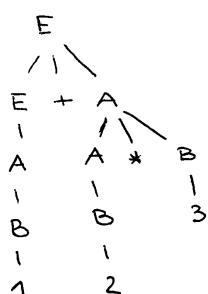
Ali lahko za jezik, ki ga opisuje gramatika iz prejšnje naloge (gramatika izrazov) zapишete nedvoumno gramatiko? Zapišite gramatiko, da bo imelo množenje višjo prioriteto kot seštevanje, ter gramatiko kjer bo imelo seštevanje višjo prioriteto kot množenje.

Gramatika, kjer ima množenje višjo prioriteto kot seštevanje:

$$E \rightarrow A \mid E + A \mid E * E$$

$$A \rightarrow B \mid A * B$$

$$B \rightarrow 0 \mid \dots \mid 9$$



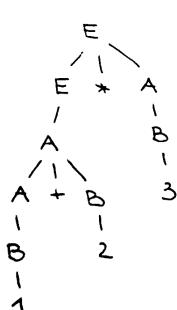
Gramatika, kjer ima seštevanje višjo prioriteto kot množenje.

$$E \rightarrow A \mid E * A$$

$$A \rightarrow B \mid A + B$$

$$B \rightarrow 0 \mid \dots \mid 9$$

$$1 + 2 \cdot 3$$

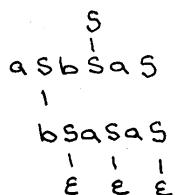
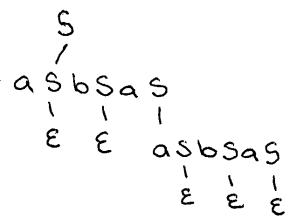


Za gramatiko

$$S \rightarrow aSaSbS \mid aSbSaS \mid bSaSaS \mid \epsilon$$

pokažite, da je dvoumna.

Niz: abaaba



Za gramatiko

$$S \rightarrow aB \mid bA$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

pokažite, da je dvoumna.

Na primer niz aabbab ima dve skrajno levi izpeljavi

$$S \rightarrow aB \rightarrow aabB \rightarrow aabbS \rightarrow aabbaB \rightarrow aabbab$$

$$S \rightarrow aB \rightarrow aaBB \rightarrow aabSB \rightarrow aabbAB \rightarrow aabbaB \rightarrow aabbab$$

Skladovni avtomati

Uvod

- ϵ -NKA z dodanim neskončnim skladom
- v vsaki potezi:
 - porabi en znak vhoda (oz. 0 v primeru ϵ -prehoda)
 - spremeni stanje
 - vrhnji simbol sklada zamenja s poljubnim zaporedjem simbolov
- $L(SA) = KNJ$
- nedeterministični SA > deterministični SA
- formalni zapis
 - $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
 - $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
- trenutni opis
 - (stanje, preostanek vhoda, sklad)
 - sklad je zapisan od vrha proti dnu

Skladovni avtomati lahko sprejemajo na podlagi dveh kriterijev:

- sprejem s končnim stanjem: avtomat sprejme besedo, ko izprazne vhod in se nahaja v končnem stanju

$$(q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha)$$

$$q \in F$$

$$\alpha \in \Gamma^*$$

- sprejem z izpraznitvijo sklada: avtomat sprejme besedo, ko izprazne vhod in hkrati izprazne sklad

$$(q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)$$

$$q \in Q$$

Sprejem z izpraznitvijo sklada

Skladovni avtomat za jezik pravilnih oglatooklepajnih izrazov.

Primer besed: $\epsilon, [], [[]], [][][], [][], \dots$

Izgradnja

- dovolj je eno stanje
- poleg Z_0 potrebujemo le še en skladovni simbol (npr. a)
- ko preberemo [, dodamo simbol a na sklad
- ko preberemo], poberemo simbol a s sklada

$$A = (\{q\}, \{[,]\}, \{Z_0, a\}, \delta, Z_0, q)$$

1. $\delta(q, [, Z_0) \rightarrow \{(q, aZ_0)\}$
2. $\delta(q, [, a) \rightarrow \{(q, aa)\}$ // na sklad dodamo a
3. $\delta(q,], a) \rightarrow \{(q, \epsilon)\}$ // a pobrišemo iz sklada
4. $\delta(q, \epsilon, Z_0) \rightarrow \{(q, \epsilon)\}$

Trenutni opis

$[][]]$

$(q, [[]], Z_0) \rightarrow (q, [][], aZ_0) \rightarrow (q,] [], aaZ_0) \rightarrow (q, [], aZ_0) \rightarrow (q,], aZ_0) \rightarrow (q, \epsilon, Z_0) \rightarrow (q, \epsilon, \epsilon)$

Sprejem s končnim stanjem

Skladovni avtomat za jezik $L = \{0^i 1^j 0^k \mid i > 0, j > 0, i + j \geq 2k\}$.

Primeri besed: 010, 00110, 0011, 001100, 000011000

Izgradnja

- ko beremo začetne ničle, smo v stanju q_0 in vsakokrat dodamo simbol a na sklad
- ko beremo enice, smo v stanju q_1 in vsakokrat dodamo simbol a na sklad
- ko beremo končne ničle, smo izmenično v stanjih q_2 in q_3 ; vsakokrat odstranimo "2 simbola a" s sklada
 - če je število ničel strogo manjše od polovice $i + j$, se sklad ne sprazne
 - če je število ničel preveliko, se sklad sprazni prezgodaj
- končno stanje: q_3

0 ... 0 1 ... 1 0 ... 0

$q_0 \quad q_1 \quad q_2 \iff q_3$ Ko beremo končne ničle se sprehajamo med stanji q_2 in q_3 .

$$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{Z_0, a\}, \delta, q_0, Z_0, \{q_3\})$$

1. $\delta(q_0, 0, Z_0) \rightarrow \{(q_0, aZ_0)\}$
2. $\delta(q_0, 0, a) \rightarrow \{(q_0, aa)\}$
3. $\delta(q_0, 1, a) \rightarrow \{(q_1, aa)\}$
i mora biti strogo večje kot 0 (na začetku je vsaj ena ničla) ... zagotovo na skladu a in ne Z_0
4. $\delta(q_1, 1, a) \rightarrow \{(q_1, aa)\}$
5. $\delta(q_1, 0, a) \rightarrow \{(q_2, \epsilon)\}$ // zadnji blok ničel
6. $\delta(q_2, \epsilon, a) \rightarrow \{(q_3, \epsilon)\}$ // preskočimo eno 0, a odstranimo
7. $\delta(q_3, 0, a) \rightarrow \{(q_2, \epsilon)\}$ // loop

Trenutni opis

0011100

$$(q_0, 0011100, Z_0) \rightarrow (q_0, 011100, aZ_0) \rightarrow (q_0, 11100, aaZ_0) \rightarrow (q_1, 1100, aaaZ_0)$$

$$\rightarrow (q_1, 100, aaaaZ_0) \rightarrow (q_1, 00, aaaaaZ_0) \rightarrow (q_2, 0, aaaaZ_0) \rightarrow (q_3, 0, aaaZ_0) \rightarrow (q_2, \epsilon, aaZ_0)$$

$$\rightarrow (q_3, \epsilon, aZ_0)$$

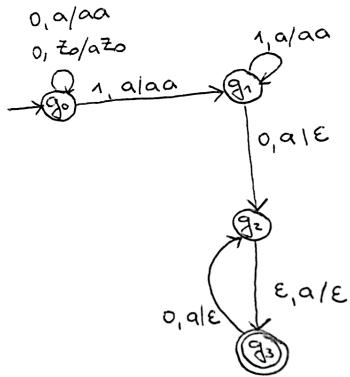
01100 ... ne pripada jeziku

$$(q_0, 01100, Z_0) \rightarrow (q_0, 1100, aZ_0) \rightarrow (q_1, 100, aaZ_0) \rightarrow (q_1, 00, aaaZ_0) \rightarrow (q_2, 0, aaZ_0)$$

$$\rightarrow (q_3, 0, aZ_0) \rightarrow (q_2, \epsilon, Z_0)$$

Sklad se prezgodaj izprazne, zato ne moremo priti do končnega stanja.

Diagram prehodov



Pretvorba KNG → SA

KNG $G = (V, T, P, S)$ lahko s preprostimi pravili pretvorimo v enakovreden SA A , ki sprejema z izpraznitvijo sklada.

$$A = (\{q\}, T, V \cup T, \delta, q, S)$$

Za vsak simbol $a \in T$ definiramo $\delta(q, a, a) = \{(q, \epsilon)\}$.

Za vsako produkcijo $A \rightarrow \alpha$ dodamo par $(q, \alpha) \vee \delta(q, \epsilon, A)$.

Pretvorimo gramatiko $S \rightarrow \epsilon \mid [S] \mid SS$ v enakovreden SA.

1. $\delta(q, [,]) \rightarrow \{(q, \epsilon)\}$
2. $\delta(q,],]) \rightarrow \{(q, \epsilon)\}$
3. $\delta(q, \epsilon, S) \rightarrow \{(q, \epsilon)\}$
4. $\delta(q, \epsilon, [S]) \rightarrow \{(q, [S])\}$
5. $\delta(q, \epsilon, SS) \rightarrow \{(q, SS)\}$

Producije 3 - 5 lahko zapišemo kot $\delta(q, \epsilon, S) \rightarrow \{(q, \epsilon), (q, [S]), (q, SS)\}$.

Trenutni opis

$[[][]]$

$(q, [[[], S]) \rightarrow 4 \rightarrow (q, [[[], [S]]) \rightarrow 1 \rightarrow (q, [[[], S]]) \rightarrow 5 \rightarrow (q, [[[], SS]))$
 $\rightarrow 4 \rightarrow (q, [[[], [S]S])) \rightarrow 1 \rightarrow (q, [[[], S]S]) \rightarrow 3 \rightarrow (q, [[[],]S]) \rightarrow 2 \rightarrow (q, [[], S])$
 $\rightarrow 4 \rightarrow (q, [[], [S]])) \rightarrow 1 \rightarrow (q, [[], S])) \rightarrow 3 \rightarrow (q, [[],]]) \rightarrow 2 \rightarrow (q, [[],]) \rightarrow 2 \rightarrow (q, \epsilon, \epsilon)$

Skladovni avtomat za jezik $L = \{w \in \{a, b\}^* \mid \#_a(w) = 3\}$.

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_1, Z_0)\} // \#a = 1$
2. $\delta(q_1, a, Z_0) \rightarrow \{(q_2, Z_0)\} // \#a = 2$
3. $\delta(q_2, a, Z_0) \rightarrow \{(q_F, \epsilon)\} // \#a = 3$

Zapišemo še za b-je. B-je ignoriramo, ostajamo v istem stanju.

4. $\delta(q_0, b, Z_0) \rightarrow \{(q_0, Z_0)\}$
5. $\delta(q_1, b, Z_0) \rightarrow \{(q_1, Z_0)\}$
6. $\delta(q_2, b, Z_0) \rightarrow \{(q_2, Z_0)\}$
7. $\delta(q_F, b, \epsilon) \rightarrow \{(q_F, \epsilon)\}$

V našem primeru smo izpraznili vhod in sklad ter smo v končnem stanju.

Skladovni avtomat za jezik $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$.

Na skladu hranimo, koliko je trenutna razlika (na skladu a-ji ali b-ji).

Stanja ne hranijo pomembnih informacij, zato imamo samo začetno in končno stanje.

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_0, aZ_0)\}$
2. $\delta(q_0, b, Z_0) \rightarrow \{(q_0, bZ_0)\}$
3. $\delta(q_0, a, a) \rightarrow \{(q_0, aa)\}$
4. $\delta(q_0, b, b) \rightarrow \{(q_0, bb)\}$
5. $\delta(q_0, a, b) \rightarrow \{(q_0, \epsilon)\} // a \text{ in } b \text{ se izničita} \Rightarrow b \text{ vržemo iz sklada (namesto } b\text{-ja zapišemo } \epsilon\text{)}$
6. $\delta(q_0, b, a) \rightarrow \{(q_0, \epsilon)\} // a \text{ in } b \text{ se izničita} \Rightarrow a \text{ vržemo iz sklada (namesto } a\text{-ja zapišemo } \epsilon\text{)}$
7. $\delta(q_0, \epsilon, Z_0) \rightarrow \{(q_F, \epsilon)\} // \text{iz sklada vržemo še } Z_0, \text{ tiki prehod ("preskočimo" vhod)}$

1. - 4.: push na sklad

5. - 6.: ujemanje (matchamo a-je in b-je)

7.: skok v končno stanje

Sled izvajanja

$w = abbaab$

$(q_0, abbaab, Z_0) \rightarrow (q_0, bbaab, aZ_0) \rightarrow (q_0, baab, Z_0) \rightarrow (q_0, aab, bZ_0) \rightarrow (q_0, ab, Z_0)$
 $\rightarrow (q_0, b, aZ_0) \rightarrow (q_0, \epsilon, Z_0) \rightarrow (q_F, \epsilon)$

Skladovni avtomat za jezik $L = \{a^n b^{2n} \mid n \geq 0\}$.

Za vsak a, ki ga vidimo na vhodu, damo na sklad dva a-ja (množenje a-jev).

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_0, aaZ_0)\}$ // na sklad 2 a-ja
2. $\delta(q_0, a, a) \rightarrow \{(q_0, aaa)\}$ // na sklad 2 a-ja
3. $\delta(q_0, b, a) \rightarrow \{(q_1, \epsilon)\}$ // zamenjamo stanje, en a vržemo iz sklada
// zamenjamo stanje, da se a-ji in b-ji ne mešajo, od q_1 ne bo več a-jev na vhodu
4. $\delta(q_1, b, a) \rightarrow \{(q_1, \epsilon)\}$ // a vržemo iz sklada
5. $\delta(q_1, \epsilon, Z_0) \rightarrow \{(q_F, \epsilon)\}$

Skladovni avtomat za jezik $L = \{a^{2n} b^n \mid n \geq 1\}$.

Obratno kot zgoraj - imamo 2x krat toliko a-jev kot b-jev: ne moremo reči, da vržemo 2 b-ja iz sklada (vedno vidimo 2 na skladu).

Moramo iti v neko drugo stanje, v katerem si zapomnimo, da v tistem stanju jemljejo dol iz sklada.

Skladovni avtomat za jezik $L = \{a^i b^j c^k \mid i + k = j\}$.

Razbijemo na dva tako jezika:

$a^i b^i \quad b^k c^k$

$x \quad y$

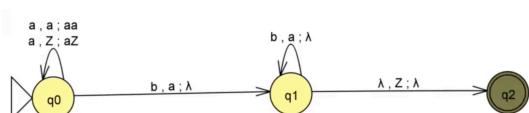
Ujemati se mora število a-jev z nekim številom b-jev, preostanek b-jev pa se mora ujemati z številom c-jev.

Ideja

Na sklad najprej postavimo a-je. Nato matchamo nekaj b-jev s tistimi a-ji, ki so bili na vhodu. Nato preostanek b-jev damo na sklad in matchamo z c-ji.

Rešitev

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_0, aZ_0)\}$
2. $\delta(q_0, a, a) \rightarrow \{(q_0, aa)\}$
3. $\delta(q_0, b, a) \rightarrow \{(q_1, \epsilon)\}$
4. $\delta(q_1, b, a) \rightarrow \{(q_1, \epsilon)\}$
5. $\delta(q_1, b, Z_0) \rightarrow \{(q_1, bZ_0)\}$
// pridemo do konca prvega dela (x)
6. $\delta(q_1, b, b) \rightarrow \{(q_1, bb)\}$
7. $\delta(q_1, c, b) \rightarrow \{(q_2, \epsilon)\}$ // zamenjamo stanje, da preprečimo, da bi se b-ji in c-ji mešali med seboj
8. $\delta(q_2, c, b) \rightarrow \{(q_2, \epsilon)\}$
9. $\delta(q_2, \epsilon, Z_0) \rightarrow \{(q_F, \epsilon)\}$



ALI pretvorba v CFG in nato PDA.

Skladovni avtomat za jezik $L = \{ w \in \{a, b\}^* \mid w = w^R, w \text{ je palindrom} \}$

Skladovni avtomati nimajo na voljo "length" funkcije. Z enim prehodom čez niz moramo razpozнати besedo.

Na nek način moramo ugotoviti, kje je sredina - PDA preklopi iz dajanja na sklad v brisanje iz sklada.

Za ta korak bomo uporabili nedeterminizem.

Na vsakem koraku imamo lahko sredino.

Predpostavimo, da vemo, kje je sredina. Na sklad postavimo prvo polovico besede.

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_0, aZ_0)\}$
2. $\delta(q_0, b, Z_0) \rightarrow \{(q_0, bZ_0)\}$
3. $\delta(q_0, a, a) \rightarrow \{(q_0, aa)\}$
4. $\delta(q_0, b, b) \rightarrow \{(q_0, bb)\}$
5. $\delta(q_0, a, b) \rightarrow \{(q_0, ab)\}$
6. $\delta(q_0, b, a) \rightarrow \{(q_0, ba)\}$

Nedeterminizem: če smo na sredini, ignoriramo preostanek vhoda

Smo na sredini in imamo lihi palindrom (a/b , ki smo ga potisnili na sklad je sredinski element):

A/b se ne ujema z nobenim drugim (a/b lahko zavrzemo). Gremo v neko drugo stanje, kjer bomo matchali levi in desni del palindroma, a/b pobrišemo iz sklad.

7. $\delta(q_0, \epsilon, a) \rightarrow \{(q_1, \epsilon)\}$
8. $\delta(q_0, \epsilon, b) \rightarrow \{(q_1, \epsilon)\}$

Smo na sredini in imamo sodi palindrom (a/b , ki smo ga potisnili na sklad je sredinski element):

Na vhodu imamo a/b , na skladu tudi a/b , ki pa je iz prve polovice (smo ga že dali na sklad). Gremo v drugo stanje in a/b zbrisemo. Te dva a-ja oz. b-ja se ujemeta (matching). Enako za b.

Na neki točki se odločimo, da smo že za en simbol čez sredino in ne bomo več brali vhoda. Tisto kar je na skladu je sredinski element, ki ga damo is sklada. Sedaj se mora ostanek ujemati. Ne ignoriramo več vhoda, gremo v q_1 in matchamo.

9. $\delta(q_0, a, a) \rightarrow \{(q_1, \epsilon)\}$
10. $\delta(q_0, b, b) \rightarrow \{(q_1, \epsilon)\}$

Ko smo na drugi polovici, pozabimo na nedeterminizem. Pometchama a-je na vhodu z a-ji na skladu. Enako z b.

11. $\delta(q_1, a, a) \rightarrow \{(q_1, \epsilon)\} // \epsilon$: ignoriranje vhoda
12. $\delta(q_1, b, b) \rightarrow \{(q_1, \epsilon)\}$
13. $\delta(q_1, \epsilon, Z_0) \rightarrow \{(q_F, \epsilon)\}$

1. - 6.: na sklad potisnemo prvo polovico besede

7. - 8.: lihi palindrom

9. - 10.: sodi palindrom

11. - 13.: ujemanje levega in desnega dela

Skladovni avtomat za jezik $L = \{0^n 1^n \mid n \geq 1\}$.

1. $\delta(q_0, 0, Z_0) \rightarrow \{(q_0, 0Z_0)\}$
2. $\delta(q_0, 0, 0) \rightarrow \{(q_0, 00)\}$
3. $\delta(q_0, 1, 0) \rightarrow \{(q_1, \epsilon)\}$
4. $\delta(q_1, 1, 0) \rightarrow \{(q_1, \epsilon)\}$
5. $\delta(q_1, \epsilon, Z_0) \rightarrow \{(q_F, Z_0)\}$

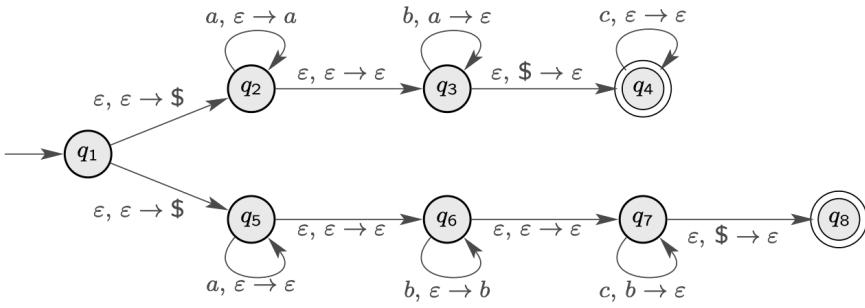
Skladovni avtomat za jezik $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ in } i = j \text{ ali } j = k\}$.

Začnemo v začetnem stanju q_0 , z začetnim simbolom Z_0 :

- $i = j = 0$ (stanje q_1)
- $i = j > 0$ (stanje q_2)
- $j = k$ (stanje q_3)

Naredili bomo skladovni avtomat, ki bo sprejel s končnim stanjem. Končna stanja bosta q_1 in q_3 .

1. $\delta(q_0, \epsilon, Z_0) \rightarrow \{(q_1, Z_0), (q_2, Z_0), (q_3, Z_0)\}$
2. $\delta(q_1, c, Z_0) \rightarrow \{(q_1, Z_0)\}$
 $i = j = 0$: predpostavjamo, da ni a-jev ali b-jev in da smo porabili vse c-je; q_1 bo končno stanje
3. $\delta(q_2, a, Z_0) \rightarrow \{(q_2, aZ_0)\} // i = j > 0$
4. $\delta(q_2, a, a) \rightarrow \{(q_2, aa)\} // i = j > 0$
5. $\delta(q_2, b, a) \rightarrow \{(q_4, \epsilon)\}$
6. $\delta(q_4, b, a) \rightarrow \{(q_4, \epsilon)\}$
7. $\delta(q_4, \epsilon, Z_0) \rightarrow \{(q_1, Z_0)\}$
v stanju q_4 pridemo do dna sklada: videli smo enako število a-jev in b-jev
8. $\delta(q_3, a, Z_0) \rightarrow \{(q_3, Z_0)\}$
 $j = k$: iz vhoda smo prebrali vse a-je; ker je $j = k = 0$ možen, je stanje q_3 tudi končno
9. $\delta(q_3, b, Z_0) \rightarrow \{(q_5, aZ_0)\} //$ ko preberemo b, začnemo šteti b-je in gremo v stanje q_5 (ni končno)
10. $\delta(q_5, b, a) \rightarrow \{(q_5, aa)\} //$ nadaljujemo s štetjem b-jev
11. $\delta(q_5, c, a) \rightarrow \{(q_6, \epsilon)\} //$ ko preberemo c-je, gremo v stanje q_6 ter matchamo c-je in b-je
12. $\delta(q_6, c, a) \rightarrow \{(q_6, \epsilon)\} //$ ko preberemo c-je, gremo v stanje q_6 ter matchamo c-je in b-je
13. $\delta(q_6, \epsilon, Z_0) \rightarrow \{(q_3, \epsilon)\} //$ v stanju q_6 pridemo do dna sklada: prebrali smo enako število b-jev in c-jev



Skladovni avtomat za jezik $L = \{0^m 1^n 2^n \mid n, m \geq 0\}$.

1. $\delta(q_1, 0, Z_0) \rightarrow \{(q_1, 0Z_0)\}$
2. $\delta(q_1, 0, 0) \rightarrow \{(q_1, 00)\}$
3. $\delta(q_1, 1, 0) \rightarrow \{(q_2, \epsilon)\}$
4. $\delta(q_2, 1, 0) \rightarrow \{(q_2, \epsilon)\}$
5. $\delta(q_2, \epsilon, Z_0) \rightarrow \{(q_2, \epsilon)\}$ // sprejmi
6. $\delta(q_1, \epsilon, Z_0) \rightarrow \{(q_2, \epsilon)\}$ // null
7. $\delta(q_1, 2, Z_0) \rightarrow \{(q_2, Z_0)\}$
8. $\delta(q_2, 2, Z_0) \rightarrow \{(q_2, Z_0)\}$

Skladovni avtomat za jezik $L = \{ww^R \mid w \text{ je beseda v } (0+1)^*\}$.

1. Začnemo v stanju q_0 , ki predstavlja "ugibanje", da še nismo na sredini; nismo še videli konca niza w , kateremu sledi njegov inverz. Ko smo v stanju q_0 , beremo simbole in jih shranujemo na sklad.
2. V vsakem trenutku, lahko "uganemo", da smo na sredini, na koncu niza w . Ko se to dejansko zgodi, bo w na skladu z desnim delom na vrhu in z levim delom na dnu sklada. To označimo tako, da gremo v stanje q_1 . Ker je avtomat nedeterminističen, smo v bistvu naredili dve "ugibanji": uganili smo, da smo videli konec w -ja obenem pa ostanemo v stanju q_1 in beremo vhodne simbole ter jih shranujemo na sklad.
3. Ko smo v stanju q_1 , primerjamo vhodne simbole z simboli na vrhu sklada. Če se ujemajo, preberemo vhodni simbol, zbrisemo simbol iz sklada in nadaljujemo. Če se ne ujemajo, nismo uganili pravilno; w -ju ni sledil njegov inverz. Ta veja umre, druge veje pa preživijo in sčasoma vodijo v sprejetje besede.
4. Če smo spraznili sklad, potem smo prebrali vhod w , ki mu je sledil njegov inverz. Sprejeli smo vhod.

Funkcija prehodov

1. $\delta(q_0, 0, Z_0) \rightarrow \{(q_0, 0Z_0)\}$
2. $\delta(q_0, 1, Z_0) \rightarrow \{(q_0, 1Z_0)\}$
3. $\delta(q_0, 0, 0) \rightarrow \{(q_0, 00)\}$
4. $\delta(q_0, 0, 1) \rightarrow \{(q_0, 01)\}$
5. $\delta(q_0, 1, 0) \rightarrow \{(q_0, 10)\}$
6. $\delta(q_0, 1, 1) \rightarrow \{(q_0, 11)\}$
7. $\delta(q_0, \epsilon, Z_0) \rightarrow \{(q_1, Z_0)\}$

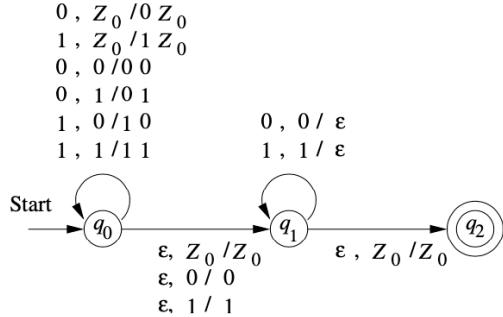
8. $\delta(q_0, \epsilon, 0) \rightarrow \{(q_1, 0)\}$

9. $\delta(q_0, \epsilon, 1) \rightarrow \{(q_1, 1)\}$

10. $\delta(q_1, 0, 0) \rightarrow \{(q_1, \epsilon)\}$

11. $\delta(q_1, 1, 1) \rightarrow \{(q_1, \epsilon)\}$

12. $\delta(q_1, \epsilon, Z_0) \rightarrow \{(q_2, Z_0)\}$

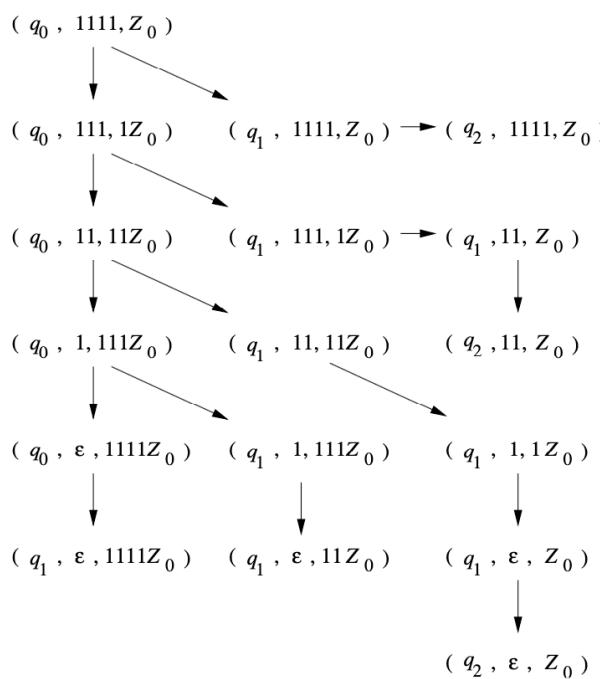


Smo v stanju q_1 , prebrali smo besedo pravilne oblike, gremo v stanje q_2 in sprejmemo

7. - 9.: spontano iz stanja q_0 v stanje q_1 , pustimo na skladu kar je na skladu

10. - 11.: v stanju q_1 matchamo vhodne simbole z simboli na skladu; če se simboli ujemajo, simbol zbrisemo iz sklada

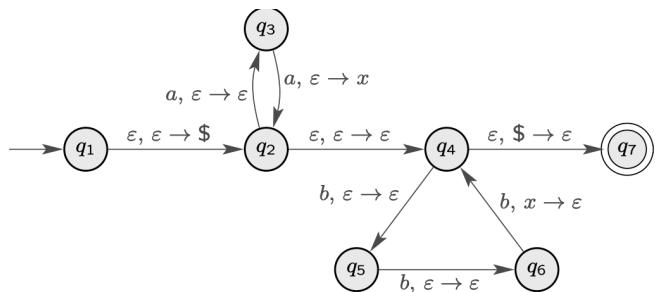
Sled izvajanja na vhodni besedi 1111.



Skladovni avtomat za jezik $L = \{a^i b^{i+2} \mid i > 0\}$.

1. $\delta(q_0, a, Z_0) \rightarrow \{(q_0, aZ_0)\}$
2. $\delta(q_1, a, a) \rightarrow \{(q_1, aa)\}$
3. $\delta(q_1, b, a) \rightarrow \{(q_2, \varepsilon)\}$
4. $\delta(q_2, b, a) \rightarrow \{(q_2, \varepsilon)\}$
5. $\delta(q_2, b, Z_0) \rightarrow \{(q_3, bZ_0)\}$
6. $\delta(q_3, b, b) \rightarrow \{(q_4, Z_0)\}$

Skladovni avtomat za jezik $L = \{a^{2n} b^{3n} \mid n \geq 0\}$.



Za KNG G, porojeno s produkcijama

$S \rightarrow 0S1 \mid A$

$A \rightarrow 1A0 \mid S \mid \varepsilon$

sestavi SA M, da bo $L(M) = L(G)$.

1. $\delta(q, 0, 0) \rightarrow \{(q, \varepsilon)\}$
2. $\delta(q, 1, 1) \rightarrow \{(q, \varepsilon)\}$
3. $\delta(q, \varepsilon, S) \rightarrow \{(q, 0S1), (q, A)\}$
4. $\delta(q, \varepsilon, A) \rightarrow \{(q, 1A0), (q, S), (q, \varepsilon)\}$

Jezik $L = \{0^n 1^m \mid n \leq m \leq 2n\}$ pretvori CFG in nato še v skladovni avtomat.

Primeri nizov, ki so v jeziku: 01, 011, 0011, 00111

CFG

$$S \rightarrow 0S1 \mid 0S11 \mid \epsilon$$

PDA

1. $\delta(q, 0, 0) \rightarrow (q, \epsilon)$
2. $\delta(q, 1, 1) \rightarrow (q, \epsilon)$
3. $\delta(q, \epsilon, S) \rightarrow \{(q, 0S1), (q, 0S11), (q, \epsilon)\}$

Formalna definicija

$\forall L \in KNJ, \exists n$

$\forall z \in L, |z| \geq n$

$\exists uvwxy, z = uvwxy, |vx| > 0, |vwx| \leq n$

$\forall i \geq 0 \Rightarrow uv^iwx^i y \in L$

Za vsak kontekstno neodvisen jezik obstaja neka konstanta n (ki je direktno posledica kontekstno neodvisne gramatike, ki opisuje kontekstno neodvisen jezik).

Za vsako besedo, ki je daljša od n-ja, mora veljati, da lahko to besedo razbijemo na 5 delov (pri lemi za regularne jezike smo jo razbili na 3 dele).

Tukaj imamo 2 ključna dela, v in x (w je samo nek ostanek).

Zopet imamo določene pogoje:

- dolžina vx mora biti večja od 0 (vsaj eden, v ali x mora biti daljši od 0)
- vwx mora biti manjši ali enak n (lahko se omejimo zgolj na tiste delitve, na 5 kosov katerih dolžina je manjša ali enaka n)

Za vsak i, ki je večji ali enak 0, lahko napihujemo v ali x (enakokrat) in bomo še vedno ostali v jeziku L.

Primerjava pogojev

- lema za regularne jezike

$|z| \geq n, z = uvw, |uv| \leq n, |v| > 0, \text{ za vsak } i \geq 0 \Rightarrow uv^i w \in L$

- lema za kontekstno neodvisne jezike

$|z| \geq n, z = uvwxy, |vwx| \leq n, |vx| > 0, \text{ za vsak } i \geq 0 \Rightarrow uv^i wx^i y \in L$

Intuitivna razlaga

Jezik je kontekstno neodvisen, če zanj obstaja kontekstno neodvisna gramatika.

Če obstaja kontekstno neodvisna gramatika, potem obstaja neko drevo izpeljave besedo iz te gramatike.

Iz produkcije S lahko pridemo v besedo z (beseda, ki jo na koncu tvorimo). Če je ta beseda dovolj dolga, pomeni, da bo vsaj ena veja v produkciji S tako dolga, da se bo neka spremenljivka morala ponoviti vsaj 2-krat.

V drevesu izpeljave obstaja pot do nekega končnega simbola, kjer se neka spremenljivka A ponovi.

A se izpelje v nek del besede z, drugi A se izpelje v nek drug del besede z, ...

Beseda z razпадne na 5 delov (u, v, w, x, y). To je naša delitev, ki opisuje točno scenarij, ko je neka pot dovolj dolga, da se A-ja ponovita. Za vse n-je, ki so dovolj dolgi, pride do takega scenarija.

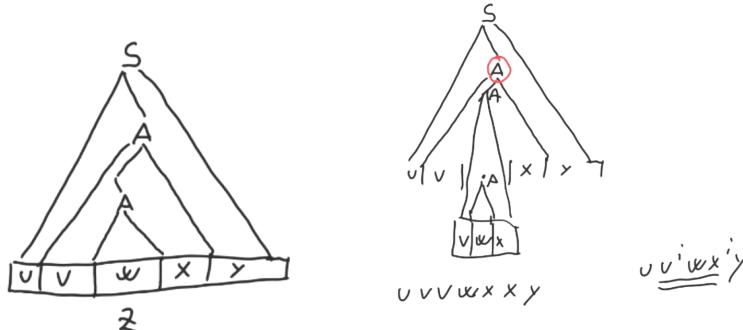
V tem primeru lahko drugi A nadomestimo z A-jem zgoraj. A se je izpeljala v v|w|x.

Celotna beseda, ki se je izpeljala iz produkcije S, je torej u v v w x y. Zopet bi lahko A izpeljali in bi dobili še en v in x.

V splošnem torej dobimo u vⁱ w x^j y (če večkrat ponovimo zamenjavo spodnjega A z zgornjim A-jem).

Omejitve:

- vx mora biti manjše ali enako n (vedno lahko zagotovimo, da vzamemo najnižjo situacijo, ko se to zgodi - najbližje dejanski besedi, ko se to zgodi)
- vx mora biti večje od 0 (če bi bila obadva prazen niz, bi sledilo, da gre A v A => redundantno drevo izpeljav; nesmiseln scenarij)



Uporaba leme

1. najdi besedo $z \in L, |z| \geq n$
2. za vsako delitev $z = uvwxy$
3. $\exists i$, da velja

$$uv^iwx^i y \notin L$$

$L \in \text{KNJ} \Rightarrow$

- obstaja n
- za vsak z, ki pripada jeziku, $|z| \geq n$
- obstaja u, v, w, x, y:

$$z = uvwxy$$

$$|vx| > 0$$

$$|vwx| \leq n$$

- za vsak i, ki je večji ali enak 0: $uv^iwx^i y \in L$

Če je L kontekstno neodvisen jezik, potem obstaja neko število n (ki ga ne poznamo).

Vsek dovolj dolgi niz (dolžine vsaj n), lahko razbijemo na 5 delov pri čemer 2. in 4. del ne smeta biti prazna in vwx morajo biti skupaj dolgi največ n.

Obstaja takšno razbitje s temi lastnostmi, da lahko v in x poljubno napihujemo ali odstranimo in vedno bomo dobili tako besedo, da bo v tem jeziku.

$a \Rightarrow b$

$\neg b \Rightarrow \neg a$

Če zanikamo desno stran, potem iz zanikane desni strani sledi zanikana leva stran.

Če ugotovimo, da desna stran ne drži, potem dokažemo, da jezik ni kontekstno nedvisen.

$L \in \text{KNJ} \Rightarrow \exists n \forall z \in L \quad \exists u, v, w, x, y, \forall i \geq 0: uv^iwx^i y \in L$

$$|z| \geq n \quad z = uvwxy$$

$$|vx| > 0$$

$$|vwx| \leq n$$

$\forall n \exists z \in L \quad \forall u, v, w, x, y, \exists i \geq 0: uv^iwx^i y \notin L$

$$|z| \geq n \quad z = uvwxy$$

$$|vx| > 0$$

$$|vwx| \leq n$$

Kako dokažemo, da nek jezik ni kontekstno neodvisen?

Začnemo z nekim n-jem (n-ja ne poznamo). Za n poiščemo nek niz z, ki je dolg vsaj n in ta n lahko uporabimo kot parameter temu nizu. Se pravi, najti moramo niz, tako da bomo za vsako delitev, ki ima te lastnosti, našli nek eksponent i, tako da bo $uv^iwx^i y$ izven jezika.

Z uporabo leme o napihovanju pokaži, da naslednji jeziki niso kontekstno neodvisni:

$$L = \{ww \mid w \in \{0, 1\}^*\}$$

Beseda se 2x ponovi v enem nizu.

Napačna rešitev

$$z = 0^n 1 0^n 1$$

$$w = 0^n 1$$

Lema o napihovanju drži za to besedo.

Delitev:

$$u = 0^i$$

$$v = 0^j$$

$$w = 0^{n-i-j} 1 0^{n-i-j}$$

$$x = 0^j$$

$$y = 0^i$$

Pri taki delitvi vsi $uv^iwx^iy \in L$.

Ne obstaja noben i pri katerem bi lahko to besedo vrgli iz jezika.

Dokazati želimo, da jezik ni kontekstno neodvisen.

Napaka je bila, da smo izbrali napačno besedo (obstaja neka delitev na 5 komponent pri kateri vedno ostajamo znotraj jezika).

Pravilna rešitev

$$z = 0^n 1^n 0^n 1^n$$

$|vwx| \leq n$ je ključna omejitev, ki bo držala vzorec v nekakšnih mejah.

Ne more se zgoditi, da bi vzorec vseboval nekaj ničel na začetku in nekaj v sredini.

Vzorec/delitev definiramo tako, da povemo kje se nahaja vwx.

- 1) vwx je homogen (vsebuje samo eno sorto simbolov, znotraj ničel bodisi enic)

vwx je nekaj ničel, $vwx = 0^i$

$$u = 0^j$$

$$v = 0^{j1}$$

$$w = 0^{j2}$$

$$x = 0^{j3}$$

$$y = 0^{n-j-i} 1^n 0^n 1^n$$

Dobimo besedo

$$uv^iwx^iy =$$

$$u^i v^{ij1} w x^{ij3} y = 0^{n+(i-1)|vx|} 1^n 0^n 1^n$$

$i = 2 \Rightarrow$ preveliko število ničel

- 2) vwx ni homogen (vsebuje 0 in 1)



- a) x ni homogen (sestavljen iz 0 in 1), npr. v in w so samo 0

pri napihanju se začnejo mešati 0 in 1 \Rightarrow ni v jeziku

- b) v ni homogen

enako kot pri a), pri napihanju se začnejo mešati 0 in 1 \Rightarrow ni v jeziku

- c) w ni homogen

prvi del besede ni enak drugemu (število 0 in 1 se razlikuje)

$$L = \{a^k b^k c^k \mid k \geq 0\}$$

- ni regularen

$$z = a^n b^n c^n$$

$$z = xyz, |xy| \leq n \text{ in } |y| \geq 0$$

$$x = a^j$$

$$y = a^k$$

$$w = a^{n-j-k} b^n c^n, j + k \leq n$$

$$z = a^j a^k a^{n-j-k} b^n c^n$$

$$i = 2: xy^2w = a^j(a^k)^2a^{n-j-k}b^n c^n = a^{n+k}b^n c^n, k \geq 1 \Rightarrow \text{niz ni prave oblike}$$

Posledično, lahko rečemo, da obstaja $i \geq 0$ za katerega beseda z ni v jeziku, $xy^iw \notin L$.

- ni KNJ:

$$z = a \dots a b \dots b c \dots c, |z| \geq n$$

$$\begin{matrix} n & n & n \end{matrix}$$

$$\text{Uganemo nek } z; z = a^n b^n c^n$$

Za vse možne delitve (vwx) moramo poiskat i , pri katerem beseda pade iz jezika, $uv^iwx^iy \notin L$.

Upoštevamo samo delitve, kjer velja $|vx| > 0$ in $|vwx| \leq n$.

$$z = uvwxy$$

- u in y lahko kar ignoriramo
- w je prostor vmes med v in x (w ne sme biti predolg)
- v in x sta pomembna (napihovanje)

Ukvarjali se bomo samo s srednjim delom (vwx) - kje se lahko pojavi znotraj niza z.

Možnost, da se vwx razteza čez vse tri skupine znakov odpade (zaradi $|vwx| \leq n$).

Lahko se razteza kvečjemu čez dve skupini znakov.

Imeli bomo 5 možnih delitev.

1) $vwx = a \dots a$ (vwx vsebovan znotraj a-jev)

poiščemo $i \geq 0$, tako da bo uv^iwx^iy izven jezika L

niz vx (ne sme biti prazen) vsebuje vsaj en a

$$v = a^p$$

$$x = a^q$$

$$p \geq 1 \text{ ali } q \geq 1$$

$$z'(i) = uv^iwx^iy$$

poiščemo $i \geq 0$, tako da bo $z'(i) \notin L$: vzamemo lahko vse i-je razen 1 (niz ostane enak); najlažje je vzeti najmanjšega

$$i = 0: z' = uv^0wx^0y = uwy \text{ (originalni niz brez v in x)}$$

$$z \quad z'(0)$$

$$\#a \quad n \quad < n \text{ v in } x \text{ skupaj vsebuje vsaj en } a \Rightarrow v \text{ in } x \text{ odstranimo} \Rightarrow \text{odstranimo vsaj en } a$$

$$\#b \quad n \quad n$$

$$\#c \quad n \quad n$$

Ker število a-jev ni več enako številu b-jev in c-jev, $z'(0)$ ne pripada jeziku.

Pri tej delitvi, ko sta v in x znotraj a-jev, smo našli takšno delitev, da pri napihovanju s tem i-jev, pademo ven iz jezika.

$$\Rightarrow z(0)' \notin L$$

2) $vwx = a \dots ab \dots b$ (vwx se razteza čez a-je in b-je)

a) $v = a^p b^q$ ($p \geq 1, q \geq 1$), $x = b^r$ ($r \geq 1$)

pri napihovanju se a-ji in b-ji prepletejo \Rightarrow rezultat ne bo prave oblike \Rightarrow ni v jeziku

b) $v = a^p$ ($p \geq 1$), $x = a^q b^r$ ($q \geq 1, r \geq 1$)

pri napihovanju se a-ji in b-ji preletejo \Rightarrow rezultat ne bo prave oblike \Rightarrow ni v jeziku

Omejili na možnosti, kjer v vsebuje samo a-je in x samo b-je.

c) $v = a^p$ ($p \geq 1$), $x = \epsilon$

To smo pokrili že pri točki 1).

d) $v = \epsilon, x = b^p$ ($p \geq 1$)

To bomo pokrili z točko 3).

e) $v = a^p$ ($p \geq 1$), $x = b^q$ ($q \geq 1$)

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uwy$$

vx vsebuje vsaj en a in vsaj en b

$\#a(z'(0)) < n //$ izgubimo vsaj en a

$\#b(z'(0)) < n //$ izgubimo vsaj en b

$\#c(z'(0)) = n$

c-jev je več kot a-jev in b-jev $\Rightarrow z'(0) \notin L$

3) $vwx = b \dots b$

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uw y$$

$$\#a(z'(0)) = n$$

$\#b(z'(0)) < n$ // izgubimo vsaj en b (v in x nista oba prazna)

$$\#c(z'(0)) = n$$

niz ni prave oblike $\Rightarrow z'(0) \notin L$

4) $vwx = b \dots b c \dots c$

Edina možnost je: $v = b^p, x = c^q, p \geq 1, q \geq 1$

Ostale možnosti privedejo do prepletanja b-jev in c-jev ali pa so pokrite druge.

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uw y$$

$$\#a(z'(0)) = n$$

$\#b(z'(0)) < n$ // vx vsebuje vsaj en b \Rightarrow izgubimo vsaj en b

$\#c(z'(0)) < n$ // vx vsebuje vsaj en c \Rightarrow izgubimo vsaj en c

niz ni prave oblike $\Rightarrow z'(0) \notin L$

5) $vwx = c \dots c$

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uw y$$

$$\#a(z'(0)) = n$$

$$\#b(z'(0)) = n$$

$\#c(z'(0)) < n$ // izgubimo vsaj en c

niz ni prave oblike $\Rightarrow z'(0) \notin L$

$$L = \{a^p b^q c^r \mid p < q \text{ in } p < r\}$$

$$z = a^n b^{n+1} c^{n+1}$$

$$z = a \dots a \quad b \dots b \quad c \dots c$$

$$n \quad n+1 \quad n+1$$

$$1) \quad vwx = a \dots a$$

$$z' (i) = uv^i wx^i y$$

$i = 0 \Rightarrow z' (0) = uv^0 wx^0 y = uw y$ // odstranimo vsaj en a iz niza \Rightarrow niz bo še "bolj" v jeziku

$$i = 2 \Rightarrow z' (2) = uv^2 wx^2 y$$

$\#a(z'(2)) \geq n + 1$ (v in x vsebujeta vsaj en a)

$\#b(z'(2)) = n + 1$

$\#c(z'(2)) = n + 1$

$$\Rightarrow z' (2) \notin L$$

$$2) \quad vwx = a \dots a b \dots b$$

$$z' (i) = uv^i wx^i y$$

$v = a^s, x = b^t, s \geq 1, t \geq 1$

$i = 0 \Rightarrow z' (0) = uv^0 wx^0 y = uw y$ // odstranimo vsaj en a ali b iz niza \Rightarrow niz bo še "bolj" v jeziku

$$i = 2 \Rightarrow z' (2) = uv^2 wx^2 y = uw y$$

$\#a(z'(2)) \geq n + 1$

$\#b(z'(2)) \geq n + 2$

$\#c(z'(2)) = n + 1$

število a-jev ne bo več strogo manj od c-jev $\Rightarrow z' (2) \notin L$

$$3) \quad vwx = b \dots b$$

$$z' (i) = uv^i wx^i y$$

$i = 0 \Rightarrow z' (0) = uv^0 wx^0 y = uw y$ // edina možnost

$\#a(z'(0)) = n$

$\#b(z'(0)) \leq n$

$\#c(z'(0)) = n + 1$

$$\Rightarrow z' (0) \notin L$$

$$4) \quad vwx = b \dots b c \dots c$$

$$z' (i) = uv^i wx^i y$$

$i = 0 \Rightarrow z' (0) = uv^0 wx^0 y = uw y$

$\#a(z'(0)) = n$

$\#b(z'(0)) \leq n$

$\#c(z'(0)) \leq n$

$$\Rightarrow z' (0) \notin L$$

5) $vwx = c \dots c$

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uwy$$

$$\#a(z'(0)) = n$$

$$\#b(z'(0)) = n + 1$$

$$\#c(z'(0)) \leq n$$

$$\Rightarrow z'(0) \notin L$$

$$L = \{a^p b^q c^r d^s \mid p, q, r, s \geq 0\}$$

$L = \{a^p b^q c^r d^s \mid p, q, r, s \geq 0\}$ ta jezik pa je KNJ.

$$z = a^n b^n c^n d^n$$

$$z = a \dots a \ b \dots b \ c \dots c \ d \dots d$$

$$\begin{array}{cccc} n & n & n & n \end{array}$$

$$1) \quad vwx = a \dots a$$

$$z'(i) = uv^i wx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0 wx^0 y = uwy$$

$$\#a(z'(0)) < n$$

$$\#b(z'(0)) = n$$

$$\#c(z'(0)) = n$$

$$\#d(z'(0)) = n$$

$$\#a < \#c \Rightarrow z'(0) \notin L$$

$$2) \quad vwx = a \dots a \ b \dots b$$

$$z'(i) = uv^i wx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0 wx^0 y = uwy$$

$$\#a(z'(0)) < n$$

$$\#b(z'(0)) < n$$

$$\#c(z'(0)) = n$$

$$\#d(z'(0)) = n$$

$$\#a < \#c \text{ in } \#b < \#d \Rightarrow z'(0) \notin L$$

$$3) \quad vwx = b \dots b$$

$$z'(i) = uv^i wx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0 wx^0 y = uwy$$

$$\#b < \#d \Rightarrow z'(0) \notin L$$

$$4) \quad vwx = b \dots b \ c \dots c$$

$$z'(i) = uv^i wx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0 wx^0 y = uwy$$

$$\#c < \#a \text{ in } \#b < \#d \Rightarrow z'(0) \notin L$$

$$5) \quad vwx = c \dots c$$

$$z'(i) = uv^i wx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0 wx^0 y = uwy$$

$$\Rightarrow z'(0) \notin L$$

6) $vwx = c \dots c d \dots d$

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uwy$$

$$\#c < \#a \text{ in } \#d < \#a \Rightarrow z'(0) \notin L$$

7) $vwx = d \dots d$

$$z'(i) = uv^iwx^i y$$

$$i = 0 \Rightarrow z'(0) = uv^0wx^0y = uwy$$

$$\#d < \#b \Rightarrow z'(0) \notin L$$

$$L = \{a^k \mid k \text{ je praštevilo}\}$$

$$z = a^p (p \geq n, p \text{ je praštevilo})$$

$z = uvwx^p, |vx| \geq 1 \Rightarrow$ ne vemo koliko a-jev v in x skupaj vsebujeta (vemo samo, da oba skupaj vsebujeta vsaj en a) \Rightarrow delamo na splošno

$$\#a(vx) = |vx| = s (s \geq 1 \text{ in } s < n)$$

$$z'(i) = uv^iwx^i y$$

$$|z'(i)| = |z| + (i - 1)|v| + (i - 1)|x| = |z| + (i - 1)(|v| + |x|) = |z| + (i - 1)|vx| = p + (i - 1)s$$

to je dolžina z-ja + kako iz z-ja dobimo $z'(i)*$

* dodamo i - 1 v-jev in še i - 1 x-sov

Poiščemo tak $i \geq 0$, da bo $z'(i) \notin L$, torej, da bo $|z'(i)|$ sestavljeni število.

Tak i je $p + 1$.

$i = p + 1$:

$$|z'(p + 1)| = p + ps = p(1 + s)$$

$$\Rightarrow z'(p + 1) \notin L$$

Produkt dveh števil ni nujno sestavljeni število (kakšen od faktorjev je lahko 1).

$p \dots$ praštevilo (večje od 1) ... bo gotovo vsaj 2

$1 + s \dots s > 0 \Rightarrow 1 + s \geq 2$

To ni praštevilo.

Za poljubno delitev, ne glede na s, smo našli napihovalni eksponent tako, da smo z napihovanjem tega eksponenta, padli iz jezika.

$$L = \{ww^Rw \mid w \in (0+1)^*\}$$

$$(0+1)^* = \{0, 1\}^*$$

$$s = \overbrace{0^k 1^k}^w \overbrace{1^k 0^k}^{w^R} \overbrace{0^k 1^k}^w = 0^k 1^{2k} 0^{2k} 1^k \in L.$$

$$k \Rightarrow n$$

$$s \Rightarrow z$$

$$z = uvxyz$$

1. Podniza v ali/in y vsebujeta nekaj simbolov iz prvega bloka n ničel.

$|vxy| \leq n$, v in y ne moreta vsebovati ničel iz drugega bloka $2n$ ničel.

$uv^0xy^0z = uxz$, ta niz mora biti oblike $0^{i_1}1^{i_2}0^{2n}1^n$, kjer $i_1 < n$ in $i_2 \leq 2n$. Če je niz uxz v L -ju, mora biti oblike $\alpha\alpha^R\alpha$. Niz uxz je oblike $0^{i_1}1^{i_2}0^{2n}1^n$ in dolžine najmanj $5n$, prvi α se mora začeti z blokom $i_1 < n$ ničel, ki jim sledi nekaj enic. $\alpha^R\alpha$ mora vsebovati vlok, ki ima največ $2i_1 < 2n$ ničel. Ampak uxz vsebuje blok $2n$ ničel, kar pa je protislovje.

2. Podniza v ali/in y vsebujeta nekaj simbolov iz prvega bloka $2n$ enic.

$|vxy| \leq n$, v in y ne moreta vsebovati enic iz drugega bloka n enic. Niz $uv^0xy^0z = uxz$, uxz mora biti oblike $0^{i_1}1^{i_2}0^{i_3}1^n$, kjer $i_1 < n$, $i_2 \leq 2n$, in $i_3 \leq 2n$. Če je uxz v jeziku L , potem mora biti oblike $\alpha\alpha^R\alpha$. Ker je uxz oblike $0^{i_1}1^{i_2}0^{i_3}1^n$ in dolžine najmanj $5n$, se mora zadnji α končati z blokom k enic, ki so pred nekaj ničlami. $\alpha\alpha^R\alpha$ mora vsebovati blok $2k$ ničel. Ampak uxz vsebuje blok $i_2 < 2n$ ničel, kar je protislovje.

3. Podniza v ali/in y vsebujeta nekaj simbolov iz drugega bloka $2n$ ničel.

$|vxy| \leq n$, v in y ne moreta vsebovati ničel iz prvega bloka k ničel.

Niz $uv^0xy^0z = uxz$, uxz mora biti oblike $0^n1^{i_1}0^{i_2}1^{i_3}$, kjer $i_1 \leq 2n$, $i_2 \leq 2n$, in $i_3 \leq n$.

Če je uxz v jeziku L , potem mora biti oblike $\alpha\alpha^R\alpha$. Ker je uxz oblike $0^n1^{i_1}0^{i_2}1^{i_3}$ in dolžne največ $5n$, se mora prvi α začeti z blokom n ničel, ki jim sledi neko število enic. $\alpha^R\alpha$ mora vsebovati blok $2n$ ničel. Ampak uxz vsebuje blok $i_2 \leq 2n$ ničel, kar pa je protislovje.

4. Podniza v ali/in y vsebujeta nekaj simbolov iz drugega bloka $2n$ enic.

$|vxy| \leq n$, v in y ne moreta vsebovati enic iz prvega bloka $2k$ enic.

Niz $uv^0xy^0z = uxz$, uxz mora biti oblike $0^n1^{2n}0^{i_1}1^{i_2}$, kjer $i_1 \leq 2n$ in $i_2 < n$.

Če je uxz v jeziku L , potem mora biti oblike $\alpha\alpha^R\alpha$. Ker je uxz oblike $0^n1^{2n}0^{i_1}1^{i_2}$ in dolžne največ $5n$, se mora drugi α končati z blokom $i_2 < n$, pred njim pa mora biti nekaj ničel.

$\alpha\alpha^R\alpha$ mora vsebovati blok največ $2i_2 < 2n$ enic. Ampak uxz vsebuje blok $2n$ enic, kar pa je protislovje.

Jezik ni CFL.

Turingovi stroji

TM je stroj, ki je sestavljen iz krmilne enote (nahaja v nekem stanju), bralno-pisalne glave (leži na neki celici traku) in neskončnega traka (razdeljen na celice).

Bralno-pisalna glava v vsakem trenutku leži na neki celici traku (bere celico traku).

V vsaki potezi

- krmilna enota spremeni stanje (lahko ostane tudi enaka)
- bralno-pisalna glava zapiše simbol za trenutno celico traku
- bralno-pisalna glava se premakne za eno mesto v desno ali levo

Funkcija prehodov se odloči glede na stanje in znaka v trenutni celici traku.

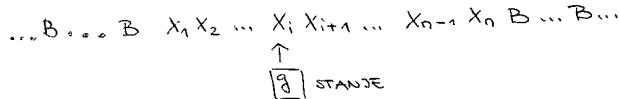
$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

$$\delta(q, X) = (q', Y, D)$$

Vzame stanje in pogleda, kateri simbol je pod bralno-pisalno glavo. Preslika v novo stanje. Zapiše oz. prepiše simbol. Premakne se levo ali desno.

Trenutni opis

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n$$



Bralno-pisalna glava oz. q se nahaja nad X_i .

Opišemo, kaj je trenutno na traku in v katerem stanju se stroj trenutno nahaja.

Predpostavke

- na traku je zapisana samo beseda w (levo in desno so zgolj presledki)
- krmilna enota je v stanju q_0
- bralno pisalna glava leži na prvem znaku besede w

M sprejme w , če se ustavi v končnem stanju.

Razlika med TM in PDA (in FA) je v tem, da se TM zaveda, kje je konec vhoda. To je mogoče, ker ima TM večjo svobodo gibanja po pomnilniku (premika se lahko levo ali desno) in vhodu. Pri PDA in FA se premikamo po vhodu samo v eni smeri.

Če želimo, da se stroj **ustavi**, potem mora biti funkcija prehodov za kakšno kombinacijo stanja in simbola nedefinirana.

Kdaj stroj sprejme besedo?

1. pogoj: stroj se mora ustaviti (stroj se ustavi, kadar funkcija prehodov ni definirana)
2. pogoj: stanje je končno

Če oba pogoja veljata, je beseda, ki smo jo podali kot vhod element jezika.

Kadar se stroj ustavi, preverimo ali je stanje v katerem se nahajamo, končno.

Turingovi stroji so v osnovi deterministični, čeprav niso njihove nedeterministične različice nič močnejše (lahko so zgolj hitrejše).

2 načina delovanja

- sprejemnik jezika

Vseeno, kaj pusti na traku, pomembno pa je, da se po branju celotne vhodne besede ustavi v končnem stanju (samo v tem primeru lahko namreč rečemo, da stroj sprejme podano vhodno besedo).

- računalnik

Zadošča, da se stroj ustavi, stanje, v katerem se ustavi, pa nas ne zanima. Pomembno pa je, kaj pusti na traku (neskončno zaporedje presledkov + iskani rezultat + neskončno zaporedje presledkov).

Turingov stroj za jezik $L = \{w\#w \mid w \in \{0, 1\}^*\}$.

Jezik, ki je sestavljen iz zaporedja ničel in enic, # ter še ene kopije tega zaporedja.

Primeri nizov, ki pripadajo jeziku: 01#01, 11010#11010

Predpostavke

Na traku Turingovega stroja imamo najprej neskončno zaporedje presledkov, vhod ter neskončno zaporedje presledkov. Turingov stroj se na začetku nahaja v začetnem stanju. Bralno-pisalna glava leži na prvi celici traku (stroj vidi prvi simbol niza). Na koncu se mora TM ustaviti v enem od končnih stanj (če je beseda v jeziku L).

Ideja

Primerjamo istoležne znake v obeh kopijah niza (če imamo na začetku 1, moramo imeti takoj za # tudi 1).

Če w ni prazen, imamo na začetku 1 ali 0. To si moramo nekako zapomniti (preidemo v ustrezno stanje, glede na to ali vidimo 0 ali 1).

Če vidimo na začetku 0, gremo v stanje q_{10} . Če vidimo na začetku 1, gremo v stanje q_{11} .

Druga komponenta indeksa stanja bo predstavljala znak, ki smo ga prebrali.

Zapomniti si moramo tudi, da smo prvi znak že videli (ko potujemo po traku, moramo zagotoviti, da se ne vrnemo že na obiskani znak). To naredimo tako, da spremenimo ta znak v znak X (npr. 1 smo že prebrali).

Sedaj se premikamo desno do istoležnega znaka v drugi polovici vhodnega niza (po znaku #).

Ko vidimo #, vemo, da smo prišli v drugo polovico niza. Ostanemo in spremenimo stanje v q_{20} ali q_{21} (odvisno, kaj smo prebrali). V tem stanju moramo preveriti znak, ki mora biti enak prvemu znaku niza.

Kateri znak smo prebrali, je sestavni del stanja. Če smo v stanju q_{20} , mora biti na začetku 0. Preverimo. Če je napačen znak, se ustavimo v ne-končnem stanju (TM ne sprejme besede). Če je znak ustrezen se prestavimo v stanju q_3 . Enico na desni strani nadomestimo z Y, tako da se potem ne vračamo nanj. Premaknemo se v levo do X (obrnemo) in preidemo v stanje q_4 . Nato spet desno v stanje q_{10} . Ponovno imamo 1 ali 0 in stanje se spremeni v q_{11} ali q_{10} . Pomikamo se desno. Ko pridemo čez #, se stanje spremeni v q_{20} ali q_{21} . Preskočimo vse Y. V odvisnosti od stanja (q_{20} ali q_{21}) preverimo, ali imamo pravi simbol. Nato se spet premikamo levo do prvega X-sa. Spremenimo stanje v q_0 . Ponavljamo ...

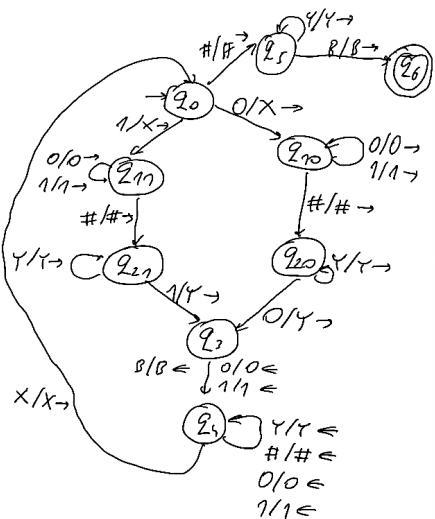
Če je niz pravilen, se bodo vsi simboli prvega niza spremenili v X, drugega niza pa v Y.

Sprehoditi se moramo čez celoten niz in preveriti, če imamo B (q_6).

Funkcija prehodov

1. $\delta(q_0, 0) \rightarrow (q_1, x, R)$ // 0 imamo na začetku
2. $\delta(q_1, 0/1) \rightarrow (q_1, 0/1, R)$
3. $\delta(q_1, \#) \rightarrow (q_2, \#, R)$
4. $\delta(q_2, x) \rightarrow (q_2, x, R)$
5. $\delta(q_2, 0) \rightarrow (q_3, x, L)$
6. $\delta(q_3, x) \rightarrow (q_3, x, L)$
7. $\delta(q_2, \#) \rightarrow (q_4, \#, L)$
8. $\delta(q_4, 0/1) \rightarrow (q_4, 0/1, L)$
9. $\delta(q_4, x) \rightarrow (q_0, x, R)$
10. $\delta(q_0, 1) \rightarrow (q_5, x, R)$ // 1 imamo na začetku
11. $\delta(q_5, 0/1) \rightarrow (q_0, 0/1, R)$
12. $\delta(q_5, \#) \rightarrow (q_6, \#, R)$
13. $\delta(q_6, x) \rightarrow (q_6, x, R)$
14. $\delta(q_6, 1) \rightarrow (q_3, x, L)$
15. $\delta(q_9, \#) \rightarrow (q_7, \#, R)$
16. $\delta(q_7, x) \rightarrow (q_7, x, R)$
17. $\delta(q_7, B) \rightarrow (q_F, B, R)$

Diagram prehodov



Stanje q_0 je na začetku, ko smo pri prvem znaku. Če preberemo 0, spremenimo stanje v q_{10} . To 0 zamenjamo z znakom X in se premaknemo desno.

Če preberemo 1, spremenimo stanje v q_{11} . To enico zamenjamo z znakom X (označimo 1) in se premaknemo desno. Enica in ničla sta simetrični. Sedaj moramo preskočiti vse enice in ničle (če vidimo 0, jo zamenjamo z X in se premaknemo desno; enako naredimo za enico).

Ko preskočimo zaporedje ničel in enice pridemo do #.

Če vidimo pri stanju q_{11} ali q_{10} znak #, # pustimo pri miru in se premaknemo desno; gremo v stanje q_{21} ali q_{20} .

Na začetku bomo v tem stanju takoj videli 0 ali 1. V nadaljnih korakih pa bomo imeli zaporedje Y (nadomestimo že prebrane znake drugega dela niza). Y preskočimo in ostanemo v istem stanju. Gremo čez vse morebitne Y in se ustavimo pri prvi 0 ali 1. Če smo v stanju q_{21} , moramo videti 1. Če smo v stanju q_{20} , moramo videti 0. V obeh primerih, če to vidimo, se premaknemo v stanje q_3 . To video 1 oz. 0 moramo zamenjati z Y in se premakniti desno. Ugotovili smo, da se prvi simbol levega dela niza ujema s prvim simbolom desnega dela niza. Sedaj smo na drugem simbolu drugega dela niza. Premakniti se moramo do zadnjega X-sa v prvem delu dela niza. Vidimo lahko 0, 1 ali B (prišli do konca drugega dela niza). V vsakem primeru se premaknemo levo in vse pustimo na miru. Premaknemo se v stanje q_4 . V stanju q_4 moramo preskočiti vse 0, 1, Y in #. Prestaviti se moramo do prvega X-sa na katerega naletimo oz. do zadnjega X-sa v prvem delu dela niza. Ko naletimo na X, se premaknemo desno in prestavimo v stanje q_0 . Potem ponovimo cikel. Ko v stanju q_0 vidimo #, ponovimo celoten postopek. Ko vidimo #, ga preskočimo in gremo desno. Pridemo v stanje q_5 in preskočimo vse Y, premaknemo se desno. Ko preskočimo vse Y, moramo preveriti, kaj je na koncu. Lahko bi se zgodilo, da je na koncu še 1 ali 0 namesto presledka. Ko vidimo presledek, se premaknemo desno in pridemo v stanje q_6 (končno stanje).

Primer izvajanja na besedi

01#01

$q_001\#01 \rightarrow Xq_{10}1\#01 \rightarrow X1q_{10}\#01 \rightarrow X1\#q_{20}01 \rightarrow X1\#Yq_31 \rightarrow X1\#q_4Y1 \rightarrow X1q_4\#Y1 \rightarrow Xq_41\#Y1 \rightarrow q_4X1\#Y1 \rightarrow Xq_01\#Y1 \rightarrow XXq_{11}\#Y1 \rightarrow XX\#q_{21}Y1 \rightarrow XX\#Yq_{21}1 \rightarrow XX\#YYq_3B \rightarrow XX\#Yq_4Y \rightarrow XX\#q_4YY \rightarrow XXq_4\#YY \rightarrow Xq_4X\#YY \rightarrow XXq_0\#YY \rightarrow XX\#q_5YY \rightarrow XX\#Yq_5Y \rightarrow XX\#YYq_5B \rightarrow XX\#YYBq_6B$

Turingov stroj, ki vhod 1^k za $k \geq 0$ preoblikuje v $1^{\lfloor k/2 \rfloor}$ (navzdol zaokroženo).

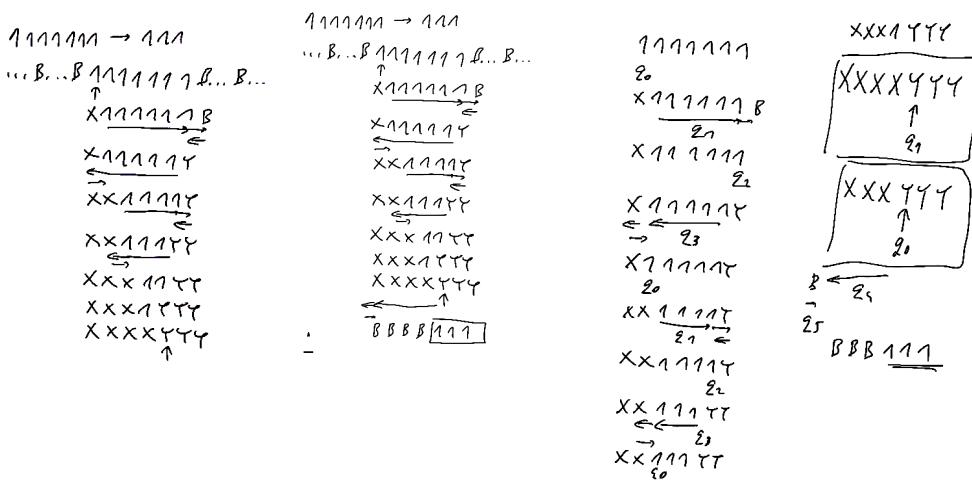
Primer: iz sedmih enic dobimo tri enice; $1111111 \rightarrow 111$

Lahko si predstavljamo, da stroj implementira matematično funkcijo - deljenje z 2.

... B ... B1111111B ... B...

Ideja

Pomagamo si z simboloma X in Y - označimo si, kaj smo že prebrali.



Prvi korak bo, da enico zamenjamo z X. Nato se sprehodimo čez vse enice, dokler ne pridemo do presledka. Ko pridemo do presledka, gremo en korak v levo, da pridemo do zadnje enice. To zadnjo enico zamenjamo z Y.

Nato se sprehodimo čez vse enice, dokler ne pridemo do X in ko pridemo do X, gremo en korak v desno.

Ideja je, da ko pridemo do prve enice, jo zamenjamo z X. Spet se sprehodimo do zadnje enice oz. Y, nato en korak levo, da pridemo do zadnje enice. Zadnjo enico ponovno zamenjamo z Y. Potem gremo spet do prve enice oz. do X, gremo korak desno in zamenjamo enico z X. Ponavljamo ...

Ko vse enice zamenjamo z X in Y, se pisalna glava nahaja nad prvim Y. Končamo.

Število Y je enako $k/2$ navzdol zaokroženo.

Sedaj moramo vse X-se pobrisati, se pravi spremeniti v presledke; Y pa v enice.

Premaknemo se levo do začetka zaporedja X-sov (do prvega presledka na levi in potem do X-sa).

Ko gremo desno, vse X-se spremenimo v presledke; vse Y pa v enice. Dobimo 3 enice, ki je rezultat funkcije.

Zanima nas, kaj ostane na traku, ko se stroj ustavi (v kateremkoli stanju).

V stanju q_0 smo vedno, ko smo na prvi enici. Enico zamenjamo z X. V stanju q_1 se premikamo desno do presledka ali pa do Y. Ko pridemo do presledka ali do Y, pridemo v stanje q_2 in se pomaknemo levo. V stanju q_2 vidimo zadnjo enico. Sedaj moramo enico zamenjati z Y. V stanju q_3 se pomikamo levo in preskočimo vse enice, dokler ne pridemo do X. Ko pridemo do X, gremo spet desno in pridemo v stanje q_0 . V stanju q_0 smo

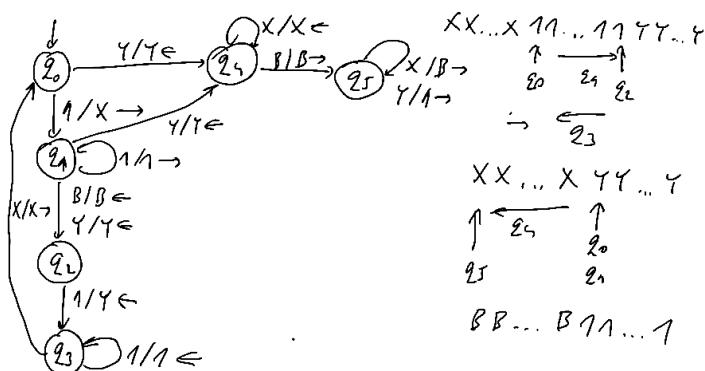
takrat, ko smo na prvi enici. To enico zamenjamo spet z X. Potem gremo spet v stanje q_1 , da pridemo do presledka ali Y ...

Na koncu pridemo do situacije XXXXYYY (nahajamo se na prvem Y). To se bo zgodilo odvisno od tega ali bo število enic liho ali sodo. Če bo število enic liho, bomo na koncu v stanju q_1 (to se bo zgodilo ravno takrat, ko 1 zamenjamo z X in bomo prišli v stanje q_1).

Če bo število enic sodo, pridemo do prvega Y v stanju q₀.

Če v stanju q_0 ali q_1 vidimo Y, je to signal, da smo že vse enice spremenili v Y ali X. Sedaj se še premaknemo levo čez vse X-se (to delamo v stanju q_4). Ko pridemo do presledka, gremo za eno mesto v desno. Pridemo v stanje q_5 , kjer zamenjamo vse X z B, vse Y pa z 1. Ko pridemo v stanju q_5 do konca, zaključimo. Na traku pustimo enice.

Diagram prehajanja stanja



V stanju q_0 smo takrat, ko smo na prvi enici. Enico moramo spremeniti v X in se pomakniti desno, premaknemo se v stanje q_1 . V stanju q_1 se premikamo desno in preskočimo vse enice dokler ne pridemo do B ali Y. Ko pridemo do presledka B ali Y pa se usmerimo levo (moramo priti nazaj do zadnje enice). Pridemo v stanje q_2 . Ko smo na zadnji enici, smo v stanju q_2 . To enico moramo zamenjati z Y, premikamo se levo, preidemo v stanje q_3 . V stanju q_3 se premikamo levo in preskakujemo enice dokler ne pridemo do prvega X. Ko pridemo do prvega X, gremo desno, da pridemo do prve enice. Pridemo nazaj v stanje q_0 in s tem zaključimo zanko. S tem delom vse enice zamenjamo z X in Y.

Če bo število na vhodu liho ali sodo, bomo vedno prišli na prvi Y. Razlika je samo v tem, v katerem stanju smo (q_0 ali q_1). Vse enice smo zamenjali z X ali z Y. Izvesti moramo samo še fazo "čiščenja". Ko vidimo Y gremo v stanje q_4 , premikamo se levo dokler ne pridemo do prvega presledka (X-se preskočimo). Ko pridemo do prvega presledka, se premaknemo desno in gremo v stanje q_5 . Smo na prvem X-su. Premikamo se desno, vse X moramo zamenjati z B-ji in Y z enicami. Ko pridemo do presledka je konec. Na traku so same enice.

Trenutni opis

- $k = 4$, sodo število enic

$q_0111 \rightarrow Xq_111 \rightarrow X1q_11 \rightarrow X11q_11 \rightarrow X111q_1B \rightarrow X11q_21 \rightarrow X1q_31Y \rightarrow Xq_311Y \rightarrow q_3X11Y$
 $\rightarrow Xq_011Y \rightarrow XXq_11Y \rightarrow XX1q_1Y \rightarrow XXq_21Y \rightarrow Xq_3XY \rightarrow XXq_0YY \rightarrow Xq_4XY \rightarrow$
 $q_4XXYY \rightarrow q_4BXXYY \rightarrow q_5XXYY \rightarrow q_5XY \rightarrow q_5YY \rightarrow 1q_5Y \rightarrow 11q_5B$

- $k = 3$, liho število enic

$q_011 \rightarrow Xq_11 \rightarrow X1q_11 \rightarrow X11q_1B \rightarrow X1q_21 \rightarrow Xq_31Y \rightarrow q_3X1Y \rightarrow Xq_01Y \rightarrow XXq_1Y \rightarrow$
 $Xq_4XY \rightarrow q_4XXY \rightarrow q_4BXXY \rightarrow q_5XXY \rightarrow q_5XY \rightarrow q_5YY \rightarrow 1q_5Y \rightarrow 11q_5B$

- vhodni niz je prazen ($k = 0$)

q_0B

- vhodni niz ima eno enico ($k = 1$)

$q_01 \rightarrow Xq_1B \rightarrow q_2X \rightarrow q_6B$

- vhodni niz ima dve enici ($k = 2$)

$q_011 \rightarrow Xq_11 \rightarrow X1q_1B \rightarrow Xq_21 \rightarrow q_3XY \rightarrow Xq_0Y \rightarrow q_4XY \rightarrow q_4BXY \rightarrow q_5XY \rightarrow q_5Y \rightarrow 1q_5B$

Turingov stroj, ki sprejema jezik $L = \{w0 \mid w \in \{0, 1\}^*\}$.

Zaporedje ničel in enic, ki se končajo na 0. Stroj bo prepoznaval sodost (vsako sodo število v binarnem zapisu ima na zadnjem mestu števko 0).

Sprehoditi se moramo torej čez celotno število in preveriti, če je zadnji simbol 0.

Sprehodimo se do konca - preskočimo vse simbole.

1. $\delta(q_0, 0) \rightarrow (q_0, 0, R)$ // preskakuj 0
2. $\delta(q_0, 1) \rightarrow (q_0, 1, R)$ // preskakuj 1

Ko pridemo do konca vhoda (do posebnega simbola blank B), gremo v novo stanje, ki bo preverilo sodost.

B pustimo na miru.

3. $\delta(q_0, B) \rightarrow (q_1, B, L)$ // konec vhoda (simbol B) - se obrnemo

Ko v stanju q_1 naletimo na 0 (edino dovoljeno), gremo v končno stanje.

4. $\delta(q_1, 0) \rightarrow (q_F, 0, R)$ // na koncu vhoda je 0

Prva dva prehoda preskakjeta vse simbole na traku. Ko ima stroj pod glavo posebni simbol B vemo, da smo prišli do konca zapisa števila. Gremo v novo stanje q_1 in se premaknemo levo na zadnji simbol števila. Če bo zadnji simbol enak 0, bo stroj skočil v stanje q_F , ki je v tem stroju končno stanje in beseda bo sprejeta.

Prehod v primeru, ko je zadnji simbol 1 sploh ni definiran, zato stroj takega števila ne bo sprejel.

Simuliranje izvajanja TM s trenutnimi opisi

Opišemo, kaj je trenutno na traku in v katerem stanju se stroj trenutno nahaja.

$q_010110 \dots 1$ je simbol, ki se nahaja tik pod glavo

- $w = 10110$

$$1q_00110 \rightarrow 10q_0110 \rightarrow 101q_010 \rightarrow 1011q_00 \rightarrow 10110q_0B \rightarrow 1011q_10 \rightarrow 10110q_F B$$

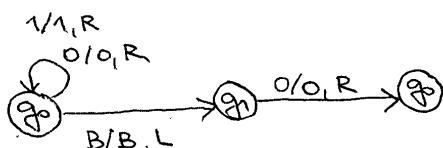
Stroj se ustavi, ker q_F nima definiranega nobenega prehoda. Beseda na vhodu je sprejeta.

- $w = 10010$

$$Bq_01001B \rightarrow 1q_00010B \rightarrow 10q_0010B \rightarrow 100q_010B \rightarrow 1001q_00B \rightarrow 1001q_1B \rightarrow 10010q_F B$$

Grafični prikaz

Simbol, ki je na traku / simbol, ki se prepiše / premik



Turingov stroj, ki opravlja aritmetične operacije:

a) številu v binarnem zapisu prišteje 1

Konstrukcijske probleme se da transformirati v odločitvene probleme.

TM damo nekaj na trak in kot rezultat dobimo, DA ali NE. Ne dobimo nekega števila.

$$L = \{w\#w \mid w \in (0, 1)^*\}$$

Ugotavljanje ali je beseda pred $\#$ enaka besedi za $\#$. Preverjanje enakosti dveh nizov.

Ta format bomo spremenili, da bomo naredili TM za prištevanje 1.

Razpoznamo besedo $w\#w'$. Če gledamo besedo w v binarnem zapisu je to enako število kot w' v binarnem zapisu. Na ta način transformiramo konstrukcijski problem v odločitveni problem ($w_b + 1 = w_b'$)

$$10111 + 1 = 11000$$

Implementirali bomo samo del, ko bomo transformirali w , $w\#w$ pa imamo že implementirano (glej zgoraj).

Problem smo prevedli na problem, ki ga znamo že reševati.

Sprehodimo se po w dokler ne naletimo na $\#$.

Ko naletimo na $\#$, gremo v novo stanje, ki bo služilo preskakovaju.

1. $\delta(q_0, 1) \rightarrow (q_0, 1, R)$
2. $\delta(q_0, 0) \rightarrow (q_0, 0, R)$
3. $\delta(q_0, \#) \rightarrow (q_1, \#, L)$

Prištevanje enice.

Če vidimo enico, ostanemo v q_1 (ker še vedno prištevamo enico), zapišemo 0 ter se premaknemo levo.

4. $\delta(q_1, 1) \rightarrow (q_1, 0, L)$

Če vidimo ničlo, prenehamo z izvajanjem in gremo v stanje q_2 , zapišemo 1 ter se premaknemo desno. Na levi nimamo kaj početi, ker se ne bo nič spremenilo.

5. $\delta(q_1, 0) \rightarrow (q_2, 1, R)$

Če v q_1 naletimo na B ($111 + 1 = 1000$).

6. $\delta(q_1, B) \rightarrow (q_2, 1, R)$

V q_2 primerjamo, ali je tisto kar je na levi strani od $\#$ enako tistemu na desni strani od $\#$.

$$\dots q_2 \dots L = \{w\#w \mid w \in (0, 1)^*\}.$$

Prištevanje enice: 4. - 6.

b) število v unarnem zapisu pomnoži z 2

1. $\delta(q_0, 1) \rightarrow (q_1, x, L)$
2. $\delta(q_1, x) \rightarrow (q_1, x, L)$
3. $\delta(q_1, B) \rightarrow (q_0, x, R)$
4. $\delta(q_0, x) \rightarrow (q_0, x, R)$
5. $\delta(q_0, B) \rightarrow (q_2, B, L)$
6. $\delta(q_2, x) \rightarrow (q_2, 1, L)$
7. $\delta(q_2, B) \rightarrow (q_F, B, R)$

c) število v unarnem zapisu prišteje 1

Unarni ali eniški.

d) število v binarnem zapisu pomnoži z 2

1. $\delta(q_1, 0) \rightarrow (q_1, 0, R)$
2. $\delta(q_1, 1) \rightarrow (q_2, 1, R)$
3. $\delta(q_1, B) \rightarrow (q_3, B, S)$
4. $\delta(q_2, 0) \rightarrow (q_2, 0, R)$
5. $\delta(q_2, 1) \rightarrow (q_2, 1, R)$
6. $\delta(q_2, B) \rightarrow (q_3, 0, S)$

Turingov stroj za jezik $L = \{a^n b^n c^n \mid n \geq 0\}$.

$L = \{a^n b^n \mid n \geq 0\}$ je CFL. Ta jezik pa ni. Skladovni avtomati ne znajo na veliko razdaljo prenašati informacije.

Niz je sestavljen iz treh delov:

a b c

Najprej razmislimo o tem, kaj če imamo niz v pravem formatu - kako potem ta niz "obvladamo".

Potem pa, kaj se zgodi, če damo vhod, ki ni v pravem formatu.

Ideja

Sprehodimo se čez vse a-je, nato vse b-je in na koncu najdemo še ujemajoče c-je. Pokazali bomo, kako prenašamo informacijo na dolge razdalje.

Ko vidimo a, ga označimo z X. Nato skočimo čez vse a-je, do prvega b-ja. Označimo ga z Y, nato preskočimo čez vse b-je. Ko naletimo na prvi c, ga označimo z Z.

Potem se vračamo nazaj do X. Pridemo do prvega a-ja, označimo drugi a z X, podobno naredimo z b-ji in c-ji. To delamo dokler ne porabimo vseh a-jev.

Implementacija prenašanja a-ja do b-ja in nato do c-ja

Začnemo v stanju q_0 , vidimo a, gremo v stanje q_1 , a označimo z X in gremo desno.

V stanju q_1 bomo prenašali a-je do b-jev.

$$1. \quad \delta(q_0, a) \rightarrow (q_1, X, R)$$

Preskočimo a-je.

$$2. \quad \delta(q_1, a) \rightarrow (q_1, a, R)$$

Če q_1 naleti na Y, ignorira Y.

$$3. \quad \delta(q_1, Y) \rightarrow (q_1, Y, R)$$

Naletimo na b, označimo b, gremo v novo stanje q_2 , zapišemo Y čez B in gremo desno.

$$4. \quad \delta(q_1, b) \rightarrow (q_2, Y, R)$$

V q_2 ignoriramo b-je ter Z-je.

$$5. \quad \delta(q_2, b) \rightarrow (q_2, b, R)$$

$$6. \quad \delta(q_2, Z) \rightarrow (q_2, Z, R)$$

V stanju q_2 naletimo na c. Gremo v novo stanje q_3 . Čez c damo Z in gremo levo.

$$7. \quad \delta(q_2, c) \rightarrow (q_3, Z, L)$$

V stanju q_3 se vračamo čez vse Z-je, b-je, Y-ne in a-je dokler ne pridemo do X-sa.

$$8. \quad \delta(q_3, Z) \rightarrow (q_3, Z, L)$$

$$9. \quad \delta(q_3, b) \rightarrow (q_3, b, L)$$

$$10. \quad \delta(q_3, Y) \rightarrow (q_3, Y, L)$$

$$11. \quad \delta(q_3, a) \rightarrow (q_3, a, L)$$

Ko q_3 naleti na X, moramo ponoviti (pridemo do stanja q_1 , kjer vzamemo nov a).

Sklenimo zanko: prenesi a do b-ja, b do c-ja, reset ...

$$12. \quad \delta(q_3, X) \rightarrow (q_0, X, R)$$

Kdaj je konec? Ko zmanjka a-jev (v stanju q_0 naletimo na Y).

$$13. \delta(q_0, Y) \rightarrow (q_4, Y, R)$$

Stanje q_4 bo preverilo, ali je še kaj ostalo na traku (npr. b-ji, c-ji) ali so naprej samo Y in Z.

$$14. \delta(q_4, Y) \rightarrow (q_4, Y, R)$$

Če bo q_4 naletel na b in ne bo definiran prehod, se ustavi brez da pride v končno stanje.

Če naletimo na Z pomeni, da so bili samo Y na traku - prečrtali smo vse b-je, ki so bili prej na traku (vsak a je imel pripadajoč b).

$$15. \delta(q_4, Z) \rightarrow (q_5, Z, R)$$

Preverimo še c-je.

$$16. \delta(q_5, Z) \rightarrow (q_5, Z, R)$$

$$17. \delta(q_5, B) \rightarrow (q_F, B, L)$$

Prenos a-jev do b-jev: 1. - 3.

Prenos b-jev do c-jev: 4. - 6.

Prehod glave nazaj, da pridemo do naslednjega a-ja (reset): 8. - 11.

Preverimo, ali je bilo na traku vse v redu prečrtano (vsak pripadajoč a je imel pripadajoč b, vsak b je imel pripadajoč c) - ni ostalo nič na traku: 13. - 17.

Namesto 15. - 17. lahko tudi:

$$15. \delta(q_4, Z) \rightarrow (q_4, Z, R)$$

$$16. \delta(q_4, B) \rightarrow (q_F, B, L)$$

Turingov stroj za jezik $L = \{a^n b^m c^{n \times m} \mid m, n > 0\}$.

Množenje v eniškem sistemu.

Jezik bomo razpoznali tako, da bomo pomnožili $n \times m$ in pogledali, če število c -jev ustreza temu zmnožku.

Primeri: abbcc, aabcc, aabbcccc

Pogledali bomo vse a-je, nato vse b-je in za vsak a bomo prenesli vse b-je na drugo stran.

Pri $a^n b^n c^n$ smo en a prenesli do b-jev, en do c-jev.

Ideja

Niz je sestavljen iz 3 delov.

Zbrišemo en a, nato vse b-je prenesemo do c-jev. Nato se vrnemo nazaj (Z-je pustimo, Y pa ne). Y-one zbrisemo in damo nazaj b-je. Gremo do naslednjega a-ja in ga zbrisemo. Gremo nazaj do b-jev in jih zbrisemo ter jih prenašamo do c-jev. Ponavljamo.

Prej (enojna zanka): prenos enega čez in vračanje

Zdaj (dvojna zanka): prenos enega, prenos b-jev do c-jev; ko zmanjka b-jev gremo nazaj ...

A nadomestimo z X. Gremo v stanje q_1 , ki gre do b-jev.

$$1. \delta(q_0, a) \rightarrow (q_1, X, R)$$

V stanju q_1 preskočimo a-je, ne potrebujemo Y (jih še ni in jih tudi ne bo).

$$2. \delta(q_1, a) \rightarrow (q_1, a, R)$$

Ko naletimo na b, gremo v stanje q_2 in desno.

$$3. \delta(q_1, b) \rightarrow (q_2, Y, R)$$

V stanju q_2 prenašamo en b čez. Ignoriramo b-je in Z-je.

$$4. \delta(q_2, b) \rightarrow (q_2, b, R)$$

$$5. \delta(q_2, Z) \rightarrow (q_2, Z, R)$$

Ko v q_2 dobimo c, smo en b že prenesli v c.

$$6. \delta(q_2, c) \rightarrow (q_3, Z, L)$$

S q_3 se vračamo nazaj do Y (iz c-jev do b-jev).

$$7. \delta(q_3, Z) \rightarrow (q_3, Z, L) // \text{preskoči } Z\text{-je}$$

$$8. \delta(q_3, b) \rightarrow (q_3, b, L) // \text{preskoči } b\text{-je}$$

Ko pridemo v q_3 do Y, smo sklenili eno zanko. Moramo nazaj v stanje, ki bo zbrisalo en b; q_1 (zamenja z Y).

To je zanka do 3.

$$9. \delta(q_3, Y) \rightarrow (q_1, Y, R)$$

Ponavljamo prenašanje b-jev na c-je (dokler so še b-ji).

Če pridemo iz q_1 na Z pomeni, da smo prenesli vse b-je na c-je. Pobrisati moramo celoten trak, tam kjer so samo b-ji. Resetirati moramo iz Y na b.

Gremo v stanje q_4 , Z pustimo na miru in gremo levo.

$$10. \delta(q_1, Z) \rightarrow (q_4, Z, L)$$

V stanju q_4 moramo videti samo Y (resetiramo jih nazaj na b). Gremo levo.

$$11. \delta(q_4, Y) \rightarrow (q_4, b, L)$$

Pridemo do a-jev in jih preskočimo (ni treba resetirati).

$$12. \delta(q_4, a) \rightarrow (q_4, a, L)$$

Ko pridemo do skrajno levega X-sa, smo sklenili še eno zanko, X pustimo na miru in gremo desno.

To je zunanja zanka za ponavljanje prenosa a-jev na b-je (začne pri 1.).

$$13. \delta(q_4, X) \rightarrow (q_0, X, R)$$

Končamo, ko zmanjka a-jev (v stanje q_0 naletimo na b). Gremo v novo stanje q_5 , kjer bomo delali preverjanje (če so vsi Z-ji prečrtani oz. zbrisani).

V q_5 ignoriramo b-je. Ko pridemo do Z-jev, gremo lahko v novo stanje. Ignoriramo Z-je in jih preskočimo.

Če naletimo na kak c, ne smemo imeti definiran prehod, saj se mora stroj ustaviti (c-jev je več kot tistih, ki smo jih prečrtali, ni ok).

Vse prečrtano (vse a-je prenesli na b-je ...), če pridemo do B. Lahko gremo v končno stanje.

$$14. \delta(q_0, b) \rightarrow (q_5, b, R)$$

$$15. \delta(q_5, b) \rightarrow (q_5, b, R)$$

$$16. \delta(q_5, Z) \rightarrow (q_6, Z, R)$$

$$17. \delta(q_6, Z) \rightarrow (q_6, Z, R)$$

$$18. \delta(q_6, B) \rightarrow (q_F, B, L)$$

q_2 : en prenos b-ja do c-ja

q_3 : reset

Prenos a-jev na b-je: 1. - 3.

Prenos b-jev na c-je: 4. - 6.

Reset (c => b): 7. - 8.

Sled

aabbcccc

Xabbcccc

XaYbZccc

XaYYZZcc

XabbZZcc

XXYbZZZc

XXYYZZZZ

XXbbZZZZ

Turingov stroj za jezik $L = \{a^{2^n} \mid n \geq 0\}$. Simulirajte izvajanje tega stroja nad besedo a^8 . Kako bi s takim (rahlo spremenjenim) strojem izračunali $\lceil \log_2 \rceil$ (računanje dvojiškega logaritma).

Najprej bomo računanje potence zapisali kot odločitveni problem.

Število a -jev je neka potenca števila 2. Besede, ki so v tem jeziku: $a, aa, aaaa, a^8$

To število je zapisano v eniškem formatu. Če bi bilo zapisano v dvojiškem: $100 \dots 0 \Rightarrow 10^*$.

Ideja

Če $100 \dots 0$ delimo z 2, bo vedno ostanek 0; dokler ne pridemo do 1.

Kriterij, da je število potenca 2: število se vedno lahko deli z 2 brez ostanka (razen takrat, ko imamo na vhodu 1)

Vzamemo število in ga delimo z 2. Ponavljamo.

Delimo tako, da prečrtamo vsak drugi znak.

a a a a
a x a x
a x x x

// za x-se se delamo kot da jih ni

Ostanek je takrat, ko je število liho. Kako vemo, da je število a -jev na vhodu sodo? Z nekim stanjem štejemo pariteto (trenutni ostanek pri deljenju z 2).

Pariteta + deljenje

Smo v stanju q_0 in vidimo a , gremo v stanje q_1 (videli smo liho število a -jev) in gremo desno.

1. $(q_0, a) \rightarrow (q_1, a, R)$

Če vidimo sodo mnogo a -jev, se vrnemo v q_1 , a prečrtamo in gremo desno.

q_1 in q_0 se izmenjujeta, dokler ne pridemo do konca.

2. $(q_1, a) \rightarrow (q_0, X, R)$

Na blank pridemo v stanju q_0 (sodo \Rightarrow ni ostanka).

3. $(q_0, B) \rightarrow (q_2, B, L)$

Vrnemo se nazaj po nizu in delimo naprej. V q_2 preskočimo a -je in X -se.

Če z q_2 na levi strani zadanemo na B , se vrnemo v stanje q_0 .

Sklenimo zanko deljenja.

4. $(q_2, a) \rightarrow (q_2, a, L)$

5. $(q_2, X) \rightarrow (q_2, X, L)$

6. $(q_2, B) \rightarrow (q_0, B, R)$

V stanju q_0 in q_1 ignoriramo X -se.

7. $(q_0, X) \rightarrow (q_0, X, R)$

8. $(q_1, X) \rightarrow (q_1, X, R)$

Kdaj končamo? Kdaj imamo res potenco števila 2? Na traku je samo ena enica.

S q_1 pridemo do B (na traku imamo liho število a -jev).

V stanju q_3 preverimo, kaj imamo na traku. X -se ignoriramo. Ko vidimo a , mora biti to edini a , ki je ostal na traku. Če je kakšno drugo število, ga stroj ne sme sprejeti. Tako za a -jem mora biti B .

9. $(q_1, B) \rightarrow (q_3, B, L)$

10. $(q_3, X) \rightarrow (q_3, X, L)$

11. $(q_3, a) \rightarrow (q_4, a, L)$

12. $(q_4, B) \rightarrow (q_F, B, R)$

Deljenje: 1. - 3.

Reset: 4. - 6.

Ignore: 7. - 8.

Trenutni opis

w = aaaa

$q_0aaa \rightarrow aq_1aaa \rightarrow aXq_0aa \rightarrow aXaq_1a \rightarrow aXaXq_0B \rightarrow aXaq_2X \rightarrow aXq_2aX \rightarrow aq_2XaX \rightarrow q_2aXaX \rightarrow q_2BaXaX \rightarrow q_0aXaX \rightarrow \dots \rightarrow aXXXq_4B \rightarrow \dots \rightarrow q_0aXXX \rightarrow \dots \rightarrow aXXXq_1B$ // preverjanje ali je na traku samo 1 $\rightarrow aXXq_3X \rightarrow aXq_3XX \rightarrow aq_3XXX \rightarrow q_3aXXX \rightarrow q_4BaXXX \rightarrow q_FaXXX$

Turingov stroj, ki sprejme jezik $L = \{0^n 1^n \mid n \geq 1\}$.

Na traku je končno zaporedje ničel in enic, pred njim in za njim pa je neskončno zaporedje praznih celic (B). TM bo spremenil 0 na X in potem 1 na Y, dokler se ne bodo ničle in enice ujemale.

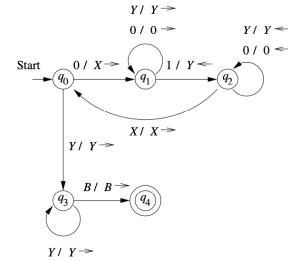
Začnemo na levi strani vhoda, vstopimo v zanko v kateri se ničle spremenijo v X, nato se premikamo desno čez ničle in Y-one, dokler ne pridemo do enice. Spremenimo enico v Y in se premikamo levo čez Y-one in ničle dokler ne najdemo X. Na tem mestu, pogledamo na 0 na desni, če jo vidimo, jo spremenimo v X in ponavljamo proces spremicanja ujemajočih enic z Y.

Če vhod ni oblike $0^* 1^*$, potem se bo TM sčasoma ustavil brez sprejetja.

Če konča s spremicanjem vseh enic v X v enakem koraku kot spremeni 1 v Y, potem je našel vhod oblike $0^n 1^n$ in sprejme.

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

State	Symbol				
	0	1	X	Y	B
q_0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q_1	$(q_1, 0, R)$	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	$(q_2, 0, L)$	—	(q_0, X, R)	(q_2, Y, L)	—
q_3	—	—	—	(q_3, Y, R)	(q_4, B, R)
q_4	—	—	—	—	—



Ko M računa, del traka, ki ga je njegova tračna glava že obiskala, bo imel zaporedje simbolov $X^* 0^* Y^* 1^*$.

Na traku bo nekaj ničel, ki so bile spremanjene v X. Sledilo bo nekaj ničel, ki še niso bile spremanjene v X.

Potem bo tam nekaj enic, ki so bile že spremanjene v Y ter nekaj enic, ki še niso bile spremanjene v Y.

Stanje q_0 je začetno stanje in M vstopi v to stanje vsakič, ko se vrne od najbolj leve ničle. Če je M v stanju q_0 in prebere 0, velja pravilo v desno-levem oglišču zgoraj. V stanju q_1 , se M premika desno čez vse ničle in Y. Če M vidi X ali B, konča. Če M vidi 1, ko je v stanju q_1 , spremeni 1 v Y in vstopi v stanje q_2 ter se začne premikati levo. V stanju q_2 se M premika levo čez ničle in Y. Ko M prispe do najbolj desnega X-sa (desni rob bloka ničel, ki se je že spremenil v X), se M vrne v stanje q_0 in se premika desno. Potem imamo dve možnosti:

- Če M vidi 0, M ponavlja zgornje korake.
- Če M vidi Y, spremeni vse ničle v X. Če so bile vse enice spremanjene v Y, potem je vhod oblike $0^n 1^n$ in M sprejme. M vstopi v stanje q_3 in se začne premikati desno čez Y. Če je prvi simbol, ki ga M vidi po Y blank pomeni, da je bilo enako število ničel in enic. M vstopi v stanje q_4 in sprejme. Če M prebere še eno enico pomeni, da je bilo preveč enic in M ne sprejme. Če M prebere 0, prav tako ne sprejme (napačna oblika).

Trenutni opis

- 0011

$q_0 0 0 1 1 \rightarrow X q_1 0 1 1 \rightarrow X 0 q_1 1 1 \rightarrow X q_2 0 Y 1 \rightarrow q_2 X 0 Y 1 \rightarrow X q_0 0 Y 1 \rightarrow X X q_1 Y 1 \rightarrow X X q_1 1 \rightarrow X X q_2 Y Y \rightarrow X q_2 X Y Y \rightarrow X X q_0 Y Y \rightarrow X X Y q_3 Y \rightarrow X X Y Y q_3 B \rightarrow X X Y Y q_4 B$

- 0010

$q_0 0 0 1 0 \rightarrow X q_1 0 1 0 \rightarrow X 0 q_1 1 0 \rightarrow X q_2 0 Y 0 \rightarrow q_2 X 0 Y 0 \rightarrow X q_0 0 Y 0 \rightarrow X X q_1 Y 0 \rightarrow X X Y q_{10} \rightarrow X X Y 0 q_1 B \rightarrow X q_2 X Y Y \rightarrow X X q_0 Y Y \rightarrow X X Y q_3 Y \rightarrow X X Y Y q_3 B \rightarrow X X Y Y q_4$

Opiši jezik naslednjega TM:

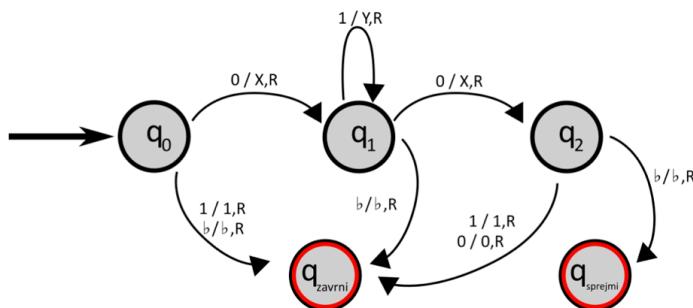
1. $\delta(q_0, 0) \rightarrow (q_1, 1, R)$
2. $\delta(q_0, 0) \rightarrow (q_1, 1, R)$
3. $\delta(q_1, B) \rightarrow (q_f, B, R)$

TM se premika samo desno. Desno se lahko premakne, če vidi 010101... na vhodnem traku.

Izmenjuje stanja q_0 in q_1 in sprejme samo, če vidi blank v stanju q_1 . To se zgodi, če vidi 0 in se premakne desno. Njegov vhod se mora končati v 0.

Sprejema regularni izraz $(01)^*0$.

Turingov stroj, ki sprejme natanko besede regularnega jezika, ki ga opišemo z regularnim izrazom 01^*0 .



Slika 2.3: Turingov stroj, predstavljen z diagramom.

Na diagramu vsaka puščica

$$\xrightarrow{0 / X, R}$$

z oznako nad njo ponazarja trenutni prebrani simbol, zapis novega simbola v celico prebranega simbola, smer premika bralne glave in prehod v naslednji trenutni opis v zaporedju. Kot primer procesiranja Turingovega stroja si oglejmo procesiranje vhodne besede 01110. Procesiranje zapišimo z zaporedjem trenutnih opisov procesiranja Turingovega stroja:

01110	$q_0 01110$
	$\xrightarrow{0 / X, R} X q_1 1110$
	$\xrightarrow{1 / Y, R} XY q_1 110$
	$\xrightarrow{1 / Y, R} XYY q_1 10$
	$\xrightarrow{1 / Y, R} XYY Y q_1 0$
	$\xrightarrow{0 / X, R} XYY Y X q_2$
	$\xrightarrow{b / b, R} XYY Y X b q_{sprejmi}$

Zaporedje trenutnih opisov procesiranja Turingovega stroja nam pove, da tak Turingov stroj sprejme vhodno besedo 01110. \triangle

Podan je TS za katerega lahko pričakujemo, da bo na vhodni trak vedno dobil 8-bitno binarno število:

1. $\delta(q_0, 0) \rightarrow (q_0, 1, R)$
2. $\delta(q_0, 1) \rightarrow (q_0, 0, R)$
3. $\delta(q_0, B) \rightarrow (q_1, B, L)$
4. $\delta(q_1, 0) \rightarrow (q_F, 1, S)$
5. $\delta(q_1, 1) \rightarrow (q_2, 0, L)$
6. $\delta(q_2, 0) \rightarrow (q_1, 1, L)$
7. $\delta(q_2, 1) \rightarrow (q_2, 1, L)$
8. $\delta(q_2, B) \rightarrow (q_F, B, S)$

- a) Zapiši izvajanje TS nad besedo **00101011**.
- b) Popravi/dodaj/odstrani en ukaz, da bo TS izračunal negativno vrednost 8-bitnega števila v dvojiškem komplementu.
- c) Popravi/dodaj/odstrani en ukaz, da bo TS izračunal negativno vrednost 8-bitnega števila v eniškem komplementu.

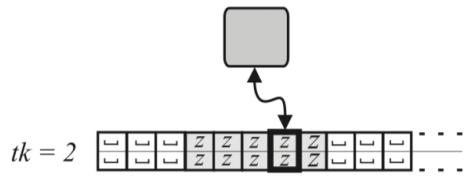
Razširitve Turingovih strojev

1. Večsledni TS

Ima eno kontrolno enoto. Trak je lahko sestavljen iz več sledi. Bralno-pisalna glava je postavljena nad vse celice vhodnega traku. V vsaki celici traku imamo lahko zapisan en simbol iz tračne abecede. Vse simbole iz celice preberemo hkrati in se na podlagi kombinacije vhodnih simbolov odločimo, kaj naredimo.

$$\delta_V : Q \times \Gamma^{tk} \rightarrow Q \times \Gamma^{tk} \times \{L, R, S\}.$$

$$\delta(q, (x, y, z)) \rightarrow (q', (x', y', z'), R)$$



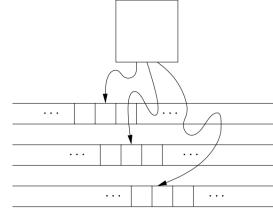
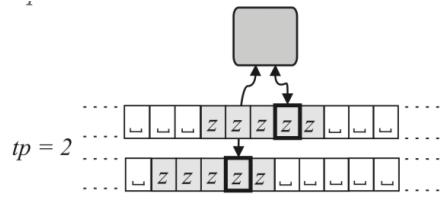
2. Večtračni TS

Ima eno kontrolno enoto (interno stanje). Ima več trakov. Vsak trak ima svojo lastno bralno-pisalno glavo, ki je neodvisna od ostalih.

Ker imamo eno kontrolno enoto, moramo hkrati prebrati vse podatke in se hkrati odločiti, kaj narediti, zapisati ter premakniti bralne-pisalne glave.

$$\delta_V : Q \times \Gamma^{tp} \rightarrow Q \times (\Gamma^{tk} \times \{L, R, S\})^{tp}.$$

$$\delta(q, (x, y, z)) \rightarrow (q', (x', L), (y', R), (z', S))$$



3. Nedeterministični TM

Zapisujemo lahko različne simbole ali pa se hkrati premikamo levo oz. desno.

Turingov program δ dodeli vsakemu (q_i, z_r) končno množico alternativnih prehodov

$$\{(q_{j_1}, z_{w_1}, D_1), (q_{j_2}, z_{w_2}, D_2), \dots\}.$$

$$\delta(q, x) \rightarrow \{(q', y, L), (q'', z, R)\}$$

Turingov stroj za jezik $L = \{a^i b^i c^i \mid i \geq 1\}$ z uporabo 4 trakov.

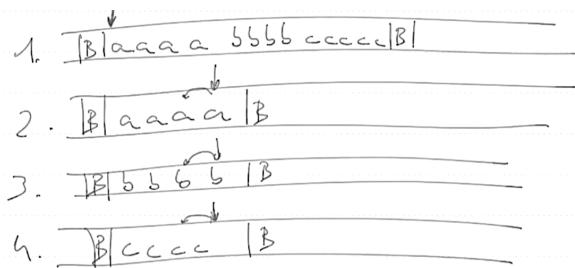
Uporabili bomo večtračni TM.

Na prvi trak damo vhodno abecedo.

Uporabimo še 3 dodatne trakove, kjer bomo zapisali enake simbole na svoj trak.

Ko končamo zapisovanje na posamezen trak, je glava na koncu. Enostavno preverimo, ali je število posameznih znakov enako.

Istočasno se premikamo v levo. Če istočasno dosežemo konec traku (prazni simbol), potem je znakov enako število.



Najprej preberemo vse a-je, dokler ne pridemo do blank (B).

1. in 2. trak: zapišemo a in se premaknemo desno; 3. in 4. trak pustimo na miru.

$$1. \delta(q_a, a, B, B, B) \rightarrow (q_a, (a, R), (a, R), (B, S), (B, S))$$

Vidimo b.

$$2. \delta(q_a, b, B, B, B) \rightarrow (q_b, (b, R), (B, S), (b, R), (B, S))$$

Preberemo vse b-je, dokler ne pridemo do blank (B).

$$3. \delta(q_b, b, B, B, B) \rightarrow (q_b, (b, R), (B, S), (b, R), (B, S))$$

Vidimo c.

$$4. \delta(q_b, c, B, B, B) \rightarrow (q_c, (c, R), (B, S), (B, S), (c, R))$$

Preberemo vse c-je, dokler ne pridemo do blank (B).

$$5. \delta(q_c, c, B, B, B) \rightarrow (q_c, (c, R), (B, S), (B, S), (c, R))$$

Po c-jih pridemo na blank (B).

$$6. (q_c, B, B, B, B) \rightarrow (q_p, (B, S), (B, L), (B, L), (B, L))$$

Novo stanje, ki ga bomo uporabili, da preverimo, ali je vseh simbolov enako število.

Z vsemi glavami se premaknemo v levo. Preverjanje, ali smo prebrali a, b in c.

$$7. (q_p, B, a, b, c) \rightarrow (q_p, (B, S), (a, L), (b, L), (c, L))$$

Konec.

$$8. (q_p, B, B, B, B) \rightarrow (q_F, (B, S), (B, S), (B, S), (B, S))$$

2-tračni Turingov stroj, ki razpoznavata jezik $L = \{a^i b^j \mid i \neq j\}$.

$\delta(q_i, T_1, T_2) \rightarrow (q_j, T_{novi1}, T_{novi2}, \text{premik}_1, \text{premik}_2)$

To pomeni, da je j večkratnik i-ja.

Primer jezika: 4a in 12b

Potrebno narediti TM z 2 trakovoma. Trakova sta neodvisna drug od drugega (vsak trak ima svojo glavo).

Stanje je eno samo.

Na prvem traku imamo na začetku zapisan vhod. Pravno-pisalna glava je na prvem znaku. Drugi trak pa je prazen (vsebuje B-je).

Ideja

Kako preverimo, da je število b-jev deljivo z številom a-jev?

A-je ali b-je damo na drug trak.

Na prvem traku bodo a-ji, na drugem pa b-ji.

B-je na prvem traku lahko izbrišemo in zamenjamo z blank-i (ni potrebno).

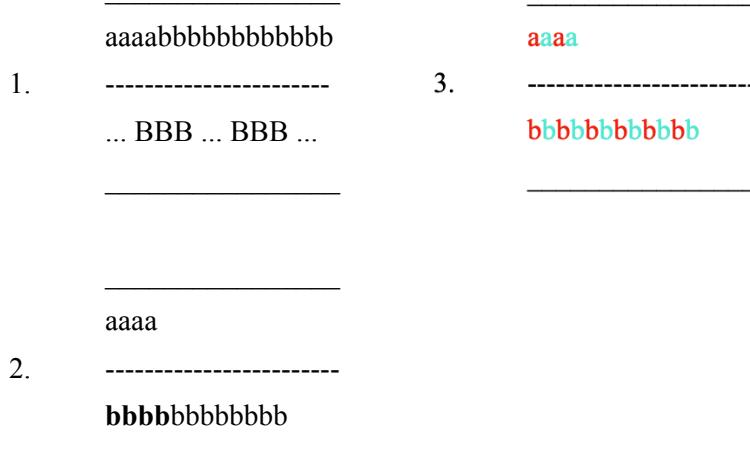
Začnemo na prvem a-ju in prvem b-ju (na obeh trakovih začnemo na začetku relevantnega dela traku).

Nato se sprehodimo čez oba trakova (a b, a b ...) do zadnjega b-ja.

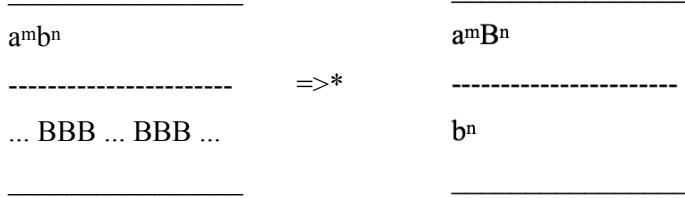
Več možnosti, ko preverjamo, če so še b-ji:

- b-jev ni več => beseda je v jeziku
- pridemo do konca a-jev in tudi do konca b-jev (v istem koraku) => beseda je v jeziku
- predčasno pridemo do konca b-jev (imamo vsaj še en a, b-jev pa ni več) => beseda ni v jeziku
- pridemo do konca a-jev ampak ne do konca b-jev (b-ji so še na traku) => vrnemo se na začetek traku a-jev, pri traku b-jev nadaljujemo => ponovno sprehod in preverjanje, če so še b-ji

1. začetno stanje
2. prepis/kopiranje, sprehod
3. preverjanje (polaganje)



Prepis:



Preverjanje (7. \Rightarrow)

Začnemo v stanju q_0 . Na prvem traku smo na začetku na a-ju, na drugem traku pa na blank-u.

Ostanemo v stanju q_0 , na prvem traku pustimo a na miru in gremo desno, na drugem traku ne naredimo ničesar (ostanemo na mestu). Se pravi, sprehodimo se čez a-je.

$$1. \delta(q_0, a, B) \rightarrow (q_0, (a, R), (B, S))$$

Prvi trak: ko pridemo do b-ja, spremenimo b na blank

Drugi trak: b zapišemo

Sprehajamo se čez b-je, na prvem traku jih brišemo, na drugem pa zapisujemo.

$$2. \delta(q_0, b, B) \rightarrow (q_0, (B, R), (b, R)) // \text{brisanje + kopiranje}$$

Prvi trak: pridemo do konca b-jev (prišli do blank-a)

Drugi trak: zapisali vse b-je iz prvega traka (v našem primeru 12), pridemo do blank-a

Na traku že imamo 4a (na prvem) in 12b (na drugem). Pozicije glav pa še niso pravilno nastavljene \Rightarrow popravimo tako, da damo glavo na prvem traku na začetek, na drugem traku pa pustimo.

Ko vidimo B, moramo iti v novo stanje. Na zgornjem traku imamo enako število B, kot na spodnjem b-jev.

$$3. \delta(q_0, B, B) \rightarrow (q_1, (B, L), (B, L)) // \text{premik na začetek obeh trakov}$$

Prvi trak: premikamo se čez B in levo \Rightarrow pridemo do a-ja

Drugi trak: premikamo se čez b in levo \Rightarrow pridemo do B

$$4. \delta(q_1, B, b) \rightarrow (q_1, (B, L), (b, L)) // \text{premik na začetek obeh trakov}$$

Prvi trak: sprehodimo se čez a-je

Drugi trak: ostanemo na mestu

$$5. \delta(q_1, a, B) \rightarrow (q_1, (a, L), (B, S)) // \text{premik na začetek obeh trakov}$$

V 5. koraku smo preskočili vse a-je. Sedaj smo tik pred prvim a-jem in prvim b-jem. Gremo v novo stanje in se premaknemo desno (smo na prvem a-ju in b-ju).

$$6. \delta(q_1, B, B) \rightarrow (q_2, (B, R), (B, R)) // \text{premik na začetek obeh trakov}$$

Sedaj lahko začnemo s "preverjanjem". Smo v stanju q_2 , obe glavi sta na začetku a-jev in b-jev.

Prvi trak: premikamo desno

Drugi trak: z b-ji nič ne počnemo, premikamo samo desno

$$7. \delta(q_2, a, b) \rightarrow (q_2, (a, R), (b, R))$$

Situacija:

- prvi trak: imamo še a-je
- drugi trak: pridemo do B

Beseda ni v jeziku. Gremo v posebno stanje (q_N). Lahko pustimo tudi nedefinirano.

Na koncu bomo iz q_0 bodisi prišli v q_F (končno stanje) ali v q_N (ne-končno stanje).

$$8. \delta(q_2, a, B) \rightarrow (q_N, (a, S), (B, L)) // \text{beseda ni v jeziku}$$

Situacija:

- prvi trak: pridemo do konca a-jev
- drugi trak: pridemo do konca b-jev

Beseda je v jeziku.

$$9. \delta(q_2, B, B) \rightarrow (q_F, (B, S), (B, S)) // \text{beseda je v jeziku}$$

Situacija:

- prvi trak: pridemo do konca a-jev (pridemo na B) => gremo na začetek a-jev
- drugi trak: imamo še b-jev => ostanemo na mestu

Nismo končali z preverjanjem.

Načeloma vedno, ko spremenimo smer, gremo v novo stanje. Za isto kombinacijo stanj in simbolov, ne smemo imeti več prehodov (stroj je determinističen).

$$10. \delta(q_2, B, b) \rightarrow (q_3, (B, L), (b, S)) // \text{na začetek a-jev}$$

Prvi trak: vidimo a

Drugi trak: vidimo b

$$11. \delta(q_3, a, b) \rightarrow (q_3, (a, L), (b, S)) // \text{na začetek a-jev}$$

Prvi trak: pridemo do B => korak v desno in v stanje q_2 (na začetek preverjanja)

Drugi trak: vidimo b

$$12. \delta(q_3, B, b) \rightarrow (q_2, (B, R), (b, S)) // \text{na začetek a-jev}$$

Rešitev

1. $\delta(q_0, a, B) \rightarrow (q_0, (a, R), (B, S))$
2. $\delta(q_0, b, B) \rightarrow (q_0, (B, R), (b, R)) // \text{brisanje + kopiranje}$
3. $\delta(q_0, B, B) \rightarrow (q_1, (B, L), (B, L)) // \text{premik na začetek obej trakov}$
4. $\delta(q_1, B, b) \rightarrow (q_1, (B, L), (b, L)) // \text{premik na začetek obej trakov}$
5. $\delta(q_1, a, B) \rightarrow (q_1, (a, L), (B, S)) // \text{premik na začetek obej trakov}$
6. $\delta(q_1, B, B) \rightarrow (q_2, (B, R), (B, R)) // \text{premik na začetek obej trakov}$
7. $\delta(q_2, a, b) \rightarrow (q_2, (a, R), (b, R))$
8. $\delta(q_2, a, B) \rightarrow (q_N, (a, S), (B, L)) // \text{beseda ni v jeziku}$
9. $\delta(q_2, B, B) \rightarrow (q_F, (B, S), (B, S)) // \text{beseda je v jeziku}$
10. $\delta(q_2, B, b) \rightarrow (q_3, (B, L), (b, S)) // \text{na začetek a-jev}$
11. $\delta(q_3, a, b) \rightarrow (q_3, (a, L), (b, S)) // \text{na začetek a-jev}$
12. $\delta(q_3, B, b) \rightarrow (q_2, (B, R), (b, S)) // \text{na začetek a-jev}$

2-tračni Turingov stroj, ki razpoznavajo jezik $L = \{ap \mid p \text{ je praštevilo}\}$.

Razpoznamo moramo praštevilsko število a-jev.

Primer besede, ki je v jeziku: 7 a-jev

Začetek

- prvi trak: vhodna beseda
- drugi trak: blank-i

aaaaaaa

... BBB ... BBB ...

Kako preverimo, ali je praštevilsko mnogo a-jev?

Kdaj bo neko število praštevilo? Kadar ne bo deljivo z 2, 3, 4, 5, n - 1. Če se nobeno deljenje ne izzide potem je število praštevilo.

Prvi trak: damo 2 a-ja

Drugi trak: a-je skopiramo v b-je

LOOP:

1. primerjamo število a-jev in b-jev

Če je število a-jev enako število b-jev, potem je beseda v jeziku.

Če je število a-jev in b-jev različno, nadaljujemo.

2. poženemo stroj iz prejšnje naloge (oz. gremo v stanje q_2)

a) 1. možnost: pridemo v stanje q_F

beseda ni v jeziku

b) 2. možnost: pridemo v stanje q_N

število a-jev ni bilo deljivo z 2, ampak vseeno je lahko deljivo s 3, ...

dodamo en a

aa

----- => v stanje q_2 prejšnjega stroja => 2 možnosti => q_N =>

bbbbbbb

aaa

----- => v stanje q_2 prejšnjega stroja => 2 možnosti => q_N =>

bbbbbbb

aaa

bbbbbbb

aaaa

bbbbbbb

Posebni primeri

1. $\delta(r_0, B, B) \rightarrow (r_N, (B, S), (B, S))$
2. $\delta(r_0, a, B) \rightarrow (r_1, (a, R), (B, S))$
3. $\delta(r_1, B, B) \rightarrow (r_N, (B, S), (B, S))$
4. $\delta(r_1, a, B) \rightarrow (r_2, (a, B), (B, S))$

a na prvem traku => b na drugem traku + aa na traku 1

Začnemo v stanje r_2 . A-je skopiramo na drugi trak v b-je. Nato a-je spremenimo v blank-e. Na prvi trak damo nato 2 a-ja.

5. $\delta(r_2, a, B) \rightarrow (r_2, (B, R), (b, R))$

Na prvem traku pridemo do blank-a (na drugem je tudi B). Na drugem traku se moramo premakniti na začetek b-jev.

6. $\delta(r_2, B, B) \rightarrow (r_3, (B, S), (B, L))$

Prvi trak je prazen. Na drugem traku pa se premikamo na začetek b-jev.

7. $\delta(r_3, B, b) \rightarrow (r_3, (B, S), (b, L))$

Ko pridemo do blank na drugem traku, gremo v stanje r_4 .

Na prvi trak zapišemo a. Na drugem traku (samo b-ji) pa gremo desno.

8. $\delta(r_3, B, B) \rightarrow (r_4, (a, R), (B, R))$

Sedaj napišemo še en a in se premaknemo levo (na začetek a-jev).

9. $\delta(r_4, B, b) \rightarrow (r_5, (a, L), (b, S))$

Sedaj imamo na prvem traku 2 a-ja, na drugem traku pa imamo toliko b-jev kot je bilo na začetku na prvem traku a-jev (v našem primeru 7).

Poženemo podprogram (program iz prejšnje naloge), da preverimo, ali je število a-jev in b-jev enako. To naredimo v stanju r_5 .

10. $\delta(r_5, a, b) \rightarrow (r_5, (a, R), (b, R))$

#a == #b

11. $\delta(r_5, B, B) \rightarrow (r_F, (B, S), (B, S))$ // beseda je v jeziku

#a != #b

Na prvem traku pridemo do B (do konca b-jev), na drugem imamo še b-je. Gremo nazaj na začetek.

12. $\delta(r_5, B, b) \rightarrow (r_6, (B, L), (b, L))$

13. $\delta(r_6, a, b) \rightarrow (r_6, (a, L), (b, L))$

Na obeh trakovih pridemo do B.

Prvi trak: 2 a-ja

Drugi trak: 7 b-jev

Gremo v stanje q_2 prejšnjega stanja. Pridemo lahko bodisi v q_F ali q_N .

14. $\delta(r_6, B, B) \rightarrow (q_2, (B, R), (B, R))$

Če pridemo v stanje q_F (oba trakova imata samo B), vemo, da beseda ni v jeziku.

15. $\delta(q_F, B, B) \rightarrow (r_N, (B, S), (B, S))$

Če pridemo v stanje q_N , povečamo število a-jev (prvi trak: smo na a-jih, drugi trak: smo na b-jih).

Prestaviti se moramo na začetek a-jev in na začetek b-jev.

Premikamo se levo do začetka a-jev. Število a-jev mora biti manjše kot število b-jev.

$$16. \delta(q_N, a, b) \rightarrow (r_7, (a, L), (b, L)) // \text{na prvi trak dodamo a}$$

$$17. \delta(r_7, a, b) \rightarrow (r_7, (a, L), (b, L)) // \text{na prvi trak dodamo a}$$

Prvi trak: pridemo do B (ostanemo), drugi trak: pridemo do nekega b => se premikamo še levo

$$18. \delta(r_7, B, b) \rightarrow (r_7, (B, S), (b, L)) // \text{na prvi trak dodamo a}$$

Na obeh trakovih pridemo do B. Dodamo a na prvi trak. Gremo v stanje q_2 .

$$19. \delta(r_7, B, B) \rightarrow (r_5, (a, S), (B, R)) // \text{preverimo } \#a == \#b$$

Ponavljamo postopke.

V stanju r_7 se premikamo levo do začetka a-jev in nato do začetka b-jev na spodnjem traku. Nato dodamo še en a. Na koncu končamo na začetku novega zaporedja a-jev. Na drugem traku pa smo na začetku seznama b-jev. Ponovno gremo v stanje r_5 . (preverimo $\#a == \#b$). Če je $\#a == \#b$, potem je beseda v jeziku. Če ne ponovimo postopek (gremo v stanje q_2 ...).

Rešitev

$$1. \delta(r_0, B, B) \rightarrow (r_N, (B, S), (B, S))$$

$$2. \delta(r_0, a, B) \rightarrow (r_1, (a, R), (B, S))$$

$$3. \delta(r_1, B, B) \rightarrow (r_N, (B, S), (B, S))$$

$$4. \delta(r_1, a, B) \rightarrow (r_2, (a, L), (B, S))$$

$$5. \delta(r_2, a, B) \rightarrow (r_2, (B, R), (b, R))$$

$$6. \delta(r_2, B, B) \rightarrow (r_3, (B, S), (B, L))$$

$$7. \delta(r_3, B, b) \rightarrow (r_3, (B, S), (b, L))$$

$$8. \delta(r_3, B, B) \rightarrow (r_4, (a, R), (B, R))$$

$$9. \delta(r_4, B, b) \rightarrow (r_5, (a, L), (b, S))$$

$$10. \delta(r_5, a, b) \rightarrow (r_5, (a, R), (b, R))$$

$$11. \delta(r_5, B, B) \rightarrow (r_F, (B, S), (B, S))$$

$$12. \delta(r_5, B, b) \rightarrow (r_6, (B, L), (b, L))$$

$$13. \delta(r_6, a, b) \rightarrow (r_6, (a, L), (b, L))$$

$$14. \delta(r_6, B, B) \rightarrow (q_2, (B, R), (B, R))$$

$$15. \delta(q_F, B, B) \rightarrow (r_N, (B, S), (B, S))$$

$$16. \delta(q_N, a, b) \rightarrow (r_7, (a, L), (b, L))$$

$$17. \delta(r_7, a, b) \rightarrow (r_7, (a, L), (b, L))$$

$$18. \delta(r_7, B, b) \rightarrow (r_7, (B, S), (b, L))$$

$$19. \delta(r_7, B, B) \rightarrow (r_5, (a, S), (B, R))$$

Turingov stroj s tremi trakovi, ki sešteje števili, podani v binarnem zapisu. Rezultat naj bo zapisan na traku #3.

Vsako vhodno število je zapisano na svojem traku (#1 in #2).

Kako seštejemo dve števili?

Števili morata biti desno poravnani, gremo od desne proti levi.

$$\begin{array}{r} 1011 \\ 111110 \\ \hline 1001001 \end{array}$$

Začetek

Prvi trak: prvo število

Drugi trak: drugo število

Tretji trak: rezultat (na začetku prazen, blank-i)

$$\begin{array}{r} 1011 \\ 111110 \\ \hline B \dots B \end{array}$$

Ideja

Premaknemo se na konec obeh trakov. Potem se premikamo levo in seštevamo.

Prenos bo posebno stanje.

Uporabljali bomo simbole a in b, ki sta lahko 0 ali 1 (poenostavimo delo, da ni potrebno pisati 4 prehodov).

Prvi trak: karkoli

Drugi trak: karkoli

Tretji trak: ničesar oz. B

Premikamo se desno.

$$1. \delta(q_0, a, b, B) \rightarrow (q_0, (a, R), (b, R), (B, S)) // \text{ premik do zadnje števke}$$

Če je število na prvem traku krajše, bomo prišli najprej do B-ja na prvem traku, drugače pa na drugem.

Prvo število krajše.

$$2. \delta(q_0, B, b, B) \rightarrow (q_0, (B, S), (b, R), (B, S)) // \text{ premik do zadnje števke}$$

Drugo število krajše.

$$3. \delta(q_0, a, B, B) \rightarrow (q_0, (a, R), (B, S), (B, S)) // \text{ premik do zadnje števke}$$

Na obeh trakovih smo prišli do B, ki direktno sledi številu. Premaknemo se levo in v stanje q_1 .

$$4. \delta(q_0, B, B, B) \rightarrow (q_1, (B, L), (B, L), (B, S)) // \text{ premik do zadnje števke}$$

Na prvem in drugem traku smo na zadnji števki. Začnemo s seštevanjem.

Seštevanje: premikamo se od desne proti levi

q_1 : ni prenosa

q_2 : je prenos

Imamo 0 in 0. Ostanemo v stanju q_1 . Pomikamo se levo.

$$5. \delta(q_1, 0, 0, B) \rightarrow (q_1, (0, L), (0, L), (0, L))$$

Imamo 0 in 1. Ostanemo v stanju q_1 . Pomikamo se levo.

$$6. \delta(q_1, 0, 1, B) \rightarrow (q_1, (0, L), (1, L), (1, L))$$

$$7. \delta(q_1, 1, 0, B) \rightarrow (q_1, (1, L), (0, L), (1, L))$$

Imamo 1 in 1. Vsota dveh enic je 1, ampak imamo prenos enice. To si zapomnimo, da gremo v drugo stanje in sicer q_2 .

$$8. \delta(q_1, 1, 1, B) \rightarrow (q_2, (1, L), (0, L), (0, L))$$

Enake možnosti imamo tudi v q_2 .

Gremo v stanje q_1 , ker ni več prenosa.

$$9. \delta(q_2, 0, 0, B) \rightarrow (q_1, (0, L), (0, L), (1, L)) // 0 + 0 + 1$$

$$10. \delta(q_2, 0, 1, B) \rightarrow (q_2, (0, L), (1, L), (0, L)) // 0 + 1 + 1$$

$$11. \delta(q_2, 1, 0, B) \rightarrow (q_2, (1, L), (0, L), (0, L)) // 1 + 0 + 1$$

$$12. \delta(q_2, 1, 1, B) \rightarrow (q_2, (1, L), (1, L), (1, L)) // 1 + 1 + 1$$

Če bi bili števili enako dolgi, bi lahko tu končali.

Če TM deluje kot računalnik, ni potrebno, da gre v končno stanje. Zanima nas samo, kaj pusti na traku, ko konča.

$$13. \delta(q_1, B, 0, B) \rightarrow (q_1, (B, S), (0, L), (0, L))$$

$$14. \delta(q_1, B, 1, B) \rightarrow (q_1, (B, S), (1, L), (1, L))$$

$$15. \delta(q_1, 0, B, B) \rightarrow (q_1, (B, S), (0, L), (0, L))$$

$$16. \delta(q_1, 1, B, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$$

$$17. \delta(q_2, B, 0, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$$

$$18. \delta(q_2, B, 1, B) \rightarrow (q_2, (B, S), (1, L), (0, L))$$

$$19. \delta(q_2, 0, B, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$$

$$20. \delta(q_2, 1, B, B) \rightarrow (q_2, (B, S), (0, L), (0, L))$$

21. $\delta(q_2, B, B, B) \rightarrow (q_1, (B, S), (B, S), (1, L))$

Rešitev

1. $\delta(q_0, a, b, B) \rightarrow (q_0, (a, R), (b, R), (B, S))$
2. $\delta(q_0, B, b, B) \rightarrow (q_0, (B, S), (b, R), (B, S))$
3. $\delta(q_0, a, B, B) \rightarrow (q_0, (a, R), (B, S), (B, S))$
4. $\delta(q_0, B, B, B) \rightarrow (q_1, (B, L), (B, L), (B, S))$
5. $\delta(q_1, 0, 0, B) \rightarrow (q_1, (0, L), (0, L), (0, L))$
6. $\delta(q_1, 0, 1, B) \rightarrow (q_1, (0, L), (1, L), (1, L))$
7. $\delta(q_1, 1, 0, B) \rightarrow (q_1, (1, L), (0, L), (1, L))$
8. $\delta(q_1, 1, 1, B) \rightarrow (q_2, (1, L), (0, L), (0, L))$
9. $\delta(q_2, 0, 0, B) \rightarrow (q_1, (0, L), (0, L), (1, L))$
10. $\delta(q_2, 0, 1, B) \rightarrow (q_2, (0, L), (1, L), (0, L))$
11. $\delta(q_2, 1, 0, B) \rightarrow (q_2, (1, L), (0, L), (0, L))$
12. $\delta(q_2, 1, 1, B) \rightarrow (q_2, (1, L), (1, L), (1, L))$
13. $\delta(q_1, B, 0, B) \rightarrow (q_1, (B, S), (0, L), (0, L))$
14. $\delta(q_1, B, 1, B) \rightarrow (q_1, (B, S), (1, L), (1, L))$
15. $\delta(q_1, 0, B, B) \rightarrow (q_1, (B, S), (0, L), (0, L))$
16. $\delta(q_1, 1, B, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$
17. $\delta(q_2, B, 0, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$
18. $\delta(q_2, B, 1, B) \rightarrow (q_2, (B, S), (1, L), (0, L))$
19. $\delta(q_2, 0, B, B) \rightarrow (q_1, (B, S), (0, L), (1, L))$
20. $\delta(q_2, 1, B, B) \rightarrow (q_2, (B, S), (0, L), (0, L))$
21. $\delta(q_2, B, B, B) \rightarrow (q_1, (B, S), (B, S), (1, L))$

Za nedeterministični Turingov stroj N, podan s prehodi

1. $\delta(q_0, 0) \rightarrow \{(q_0, 1, R)\}$
2. $\delta(q_0, 1) \rightarrow \{(q_1, 0, R)\}$
3. $\delta(q_1, 0) \rightarrow \{(q_1, 0, R), (q_0, 0, L)\}$
4. $\delta(q_1, 1) \rightarrow \{(q_1, 1, R), (q_0, 1, L)\}$
5. $\delta(q_1, B) \rightarrow \{(q_2, B, R)\}$,

kjer je B prazni tračni simbol ter $F = \{q_2\}$, simuliraj drevo izvajanj za spodnja vhodna niza:

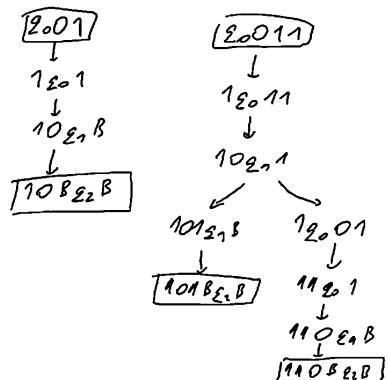
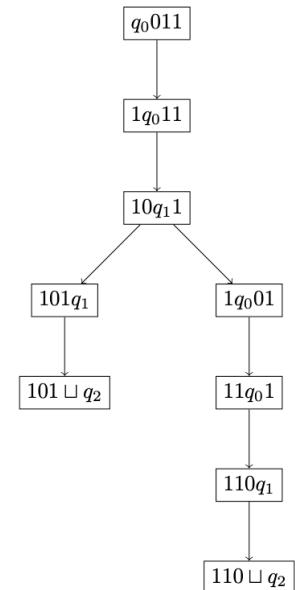
- 01
- 011

Za 01: $q_001 \rightarrow 1q_01 \rightarrow 10q_1B \rightarrow 10Bq_2B$

Za 011:

q_0011
 \rightarrow
 $1q_011$
 \rightarrow
 $10q_11$

- $101q_1B \rightarrow 101Bq_2B$
- $1q_001 \rightarrow 11q_01 \rightarrow 110q_1B \rightarrow 110Bq_2B$



Nedeterministični Turingov stroj (poljubno mnogo trakov), ki razpoznavajo jezik

$L = \{ww \mid w \in \{0, 1\}^*\}$.

Uporabili bomo 3 trakove.

Ideja

Pomikamo se desno po znakih in sproti mečemo simbole na drugi trak.

Na neki točki "uganemo", da smo prišli do sredine in gremo v drugo stanje. Od takrat naprej zapisujemo simbole na drugi trak.

Ko se sprehajamo po prvi polovici prvega traka, kopiramo na trak 2. Ko gremo čez polovico prvega raku, kopiramo na tretji trak.

0 0 1 0 0 1

0 0 1

0 0 1

Kopiramo simbole na drugi trak ali pa se na neki točki odločimo, da gremo v novo stanje (uganemo sredino) q_1 .

1. $\delta(q_0, a, B, B) \rightarrow \{(q_0, (a, R), (a, R), (B, S)) \text{ // prepisujemo na T2}$
 $(q_1, (a, S), (B, S), (B, S))\} \text{ // uganemo sredino vhodne besede}$

Začnemo s prepisovanjem simbolov na tretji trak.

2. $\delta(q_1, a, B, B) \rightarrow \{(q_1, (a, R), (B, S), (a, R))\}$

Ko pridemo do konca, gremo v novo stanje. Premaknemo se levo.

Z zgornjim trakom se ne ukvarjam več. Drugi in tretji trak: na zadnjem znaku vhoda.

Premikamo se levo in primerjamo znake med seboj.

3. $\delta(q_1, B, B, B) \rightarrow \{(q_2, (B, S), (B, L), (B, L))\} \text{ // prepisujemo na T3}$
4. $\delta(q_2, B, a, a) \rightarrow \{(q_2, (B, S), (a, L), (a, L))\}$
5. $\delta(q_2, B, B, B) \rightarrow \{(q_F, (B, S), (B, S), (B, S))\} \text{ // beseda ni v jeziku}$

(Lahko) nedeterministični Turingov stroj, ki preveri, ali je podano unarno število sestavljenlo (tj. ni praštevilo). Pri tem lahko uporabite poljubno (končno) število trakov.

Sestavljenlo število je število večje od 1, ki ni praštevilo (zato enic ne bomo sprejeli). Na prvem traku bo z vhodom predstavljenlo število n, ki ga ne bomo spremajali, na drugem pa predstavljenlo število m ≥ 2 , za katerega poskušamo, če deli vhodno število. Preverimo samo števila manjša od n. Za začetek kopiramo n na drugi trak in mu odstranimo zadnjo enico.

Nato bi lahko deterministično zaporedoma preverjali, če je m še večje od 1 (če ni, preidemo v nedefinirano stanje, kar ustavi TM), poskušali, če m deli n (v tem primeru preidemo v končno stanje) in zmanjševali m. Vendar imamo na voljo nedeterminizem, tako da lahko s prehodi samo opišemo postopek sprejema nedeterministično izbranega $2 \leq m < n$, ki deli n.

Zagotovimo, da je m ≥ 2 (obenem izločimo n = 1 in sprejmemo n = 2):

$$T = (\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}, \{1\}, \{1, \sqcup\}, q_1, \sqcup, \{q_9\})$$

$$\delta(q_1, 1, B) \rightarrow (q_2, (1, R), (1, R))$$

$$\delta(q_2, 1, B) \rightarrow (q_3, (1, R), (1, R))$$

$$\delta(q_3, B, B) \rightarrow (q_9, (B, S), (B, S))$$

Nedeterministično izberemo m ≥ 2 :

$$\delta(q_3, 1, B) \rightarrow \{(q_4, (1, R), (1, R)), (q_5, (1, S), (B))\}$$

$$\delta(q_4, 1, B) \rightarrow \{(q_4, (1, R), (1, R)), (q_5, (1, S), (B))\}$$

Izločimo m $\geq n$ (predstavitev m se mora končati pred predstavljivo n, drugače preidemo v nedefinirano stanje):

$$\delta(q_5, 1, B) \rightarrow (q_6, (1, L), (B, L))$$

Vrnemo se na začetek:

$$\delta(q_6, 1, 1) \rightarrow (q_6, (1, L), (1, L))$$

$$\delta(q_6, B, B) \rightarrow (q_7, (B, R), (B, R))$$

Napredujemo po m črk naprej v n-ju, dokler ne pridemo do konca n-ja, nato preidemo v končno stanje q9:

$$\delta(q_7, 1, 1) \rightarrow (q_7, (1, R), (1, R))$$

$$\delta(q_7, 1, B) \rightarrow (q_8, (1, R), (B, L))$$

$$\delta(q_7, B, B) \rightarrow (q_9, (B, S), (B, S))$$

$$\delta(q_8, 1, 1) \rightarrow (q_8, (1, S), (1, L))$$

$$\delta(q_8, 1, B) \rightarrow (q_6, (1, S), (B, R))$$

Pokaži, da so CE jeziki zaprti za unijo ter presek. Pokaži, da so za to operacijo zaprti tudi odločljivi jeziki.

Naj bosta L_1 in L_2 izračunljivo preštevna (CE) jezika. Jezik je izračunljivo prešteven natanko tedaj, ko je (vsaj) polodločljiv.

Ker sta L_1 in L_2 polodločljiva obstajata Turingova stroja T_1 in T_2 , ki v končnem številu korakov sprejmeta x v primeru, da je x v ustreznem jeziku.

Dokažimo, da je $L_{\text{unija}} = L_1 \cup L_2$ polodločljiv jezik in posledično bo sledilo, da je L_{unija} tudi izračunljivo prešteven.

Pričakovost besede x v L_{unija} lahko ugotovimo: T_1 in T_2 inicializiramo z vhodom z , nato izmenjajoče naredimo en korak na T_1 in en korak na T_2 .

Če je $x \in L_{\text{unija}}$, bo v končnem številu korakov en izmed teh dveh Turingovih strojev sprejel x . V tem primeru naš proces sprejme x . Po Church-Turingovi tezi obstaja Turingov stroj T , ki opravlja ta proces, torej obstaja Turingov stroj, ki ugotavlja pričakovost L_{unija} , se pravi L_{unija} je polodločljiv jezik.

Podobno sklepamo za jezik L_{presek} : če velja $x \in L_{\text{presek}}$, bosta v končnem številu korakov oba sprejela x (počakati moramo, da oba sprejmeta x).

Tudi odločljivi jeziki so zaprti za obe operacije. Pri uniji se bosta tudi v primeru, ko x ne pripada L_{unija} oba stroja v končnem številu korakov ustavila in zavrnila x (počakati moramo, da oba zavrneta x). Tudi v preseku se bo v primeru, da x ne pripada L_{presek} , en Turingov stroj, ki zavrne x , v končnem številu korakov ustavil in zavrnil x s čimer se tudi naš proces ustavi in zavrne x .

Turingov stroj za jezik $L = \{w\#w \mid w \in (0, 1)^*\}$ z dvema trakovoma.

$$\delta(q_0, 1, B) \rightarrow (q_1, 1, R, 1, R)$$

$$\delta(q_0, 0, B) \rightarrow (q_0, 0, R, 0, R)$$

$$\delta(q_0, \#, B) \rightarrow (q_1, \#, S, B, L)$$

$$\delta(q_1, \#, 1/0) \rightarrow (q_1, \#, S, 1/0, L)$$

$$\delta(q_1, \#, B) \rightarrow (q_2, \#, S, B, R)$$

$$\delta(q_2, 1, 1) \rightarrow (q_2, 1, R, 1, R)$$

$$\delta(q_2, 0, 0) \rightarrow (q_2, 0, R, 0, R)$$

$$\delta(q_2, B, B) \rightarrow (q_F, B, S, B, S)$$

Podan je jezik $L = \{xy \mid |x| > 1, y vsebuje x kot podniz, v jeziku 0 in 1\}$. (nepreverjeno)

a) 3 besede, ki pripadajo jeziku in 3 besede, ki temu jeziku ne pripadajo

pripadajo: 0101, 10010, 11010110

ne pripadajo: 01, 10, 00

$1100011x = 11y = 00011$

b) TM:

$$\delta(q_0, 0) \rightarrow (q_1, 0, R) \quad \delta(\{q_3, q_4, q_5, q_6\}, \{0, 1\}) \rightarrow (\{q_3, q_4, q_5, q_6\}, \{0, 1\}, R)$$

$$\delta(q_0, 1) \rightarrow (q_2, 1, R)$$

$$\delta(q_1, 0) \rightarrow (q_3, 0, R)$$

$$\delta(q_1, 1) \rightarrow (q_4, 1, R)$$

$$\delta(q_2, 0) \rightarrow (q_5, 0, R)$$

$$\delta(q_2, 1) \rightarrow (q_6, 1, R)$$

$$\delta(q_3, 0) \rightarrow (q_7, 0, R)$$

$$\delta(q_4, 0) \rightarrow (q_8, 0, R)$$

$$\delta(q_6, 1) \rightarrow (q_8, 1, R)$$

$$\delta(q_7, 0) \rightarrow (q_F, 0, R)$$

$$\delta(q_8, 1) \rightarrow (q_F, 1, R)$$

Ta TM ne razpozna zgoraj podanega jezika L, vendar ga z majhno spremembo lahko spremenimo v pravilnega. S spremembo ali dodajanjem ali brisanjem samega δ prehoda naredite pravilen TM.

{ } pomeni, da gre lahko q_3 v q_4 ali q_5, q_6, q_7

0, 0, R ali 1, 1, R

Vse kar TM zagotavlja je, da je beseda daljša od 1. Ostalo pa je prepusteno nedeterminizmu.

Nekaj stanj je notri, 2 možni črki na traku.

Imamo lahko katerokoli izmed 4 stanj ter 2 črki. Gremo lahko v katerekoli kombinacijo teh 4 stanj x 2 črk.

$$\delta(q_3, 1) \rightarrow (q_4, 0, R) \text{ in } \delta(q_3, 0) \rightarrow (q_4, 1, R)$$

$$\text{Dodano, } \delta(q_5, 1) \rightarrow (q_7, 1, R)$$

c) Simulirajte (s trenutnimi opisi) izvajanje popravljenega stroja nad besedo 011101.

$$q_0011101 \rightarrow 0q_111101 \rightarrow 01q_41101 \rightarrow 011q_4101 \rightarrow 01110q_81 \rightarrow 011101q_F$$

Odločitveni problemi, univerzalni TS in prevedbe

Jeziki in problemi

S pomočjo Turingovega stroja lahko ugotovimo, ali določena beseda pripada določenemu jeziku.

Turingove stroje pa lahko obravnavamo kot modele algoritmov ali računalnikov (npr. rešujemo probleme kot je iskanje maksimuma v tabeli, urejanje tabele, ...).

Vez med jeziki in problemi je zelo tesna.

Jezik je množica besed nad določeno abecedo (predpostavili bomo abecedo $\{0, 1\}$).

Računski problem je vprašanje, na katerega poskušamo odgovoriti s pomočjo algoritma.

Eden izmed problemov je **odločitveni problem**, pri katerem je odgovor DA ali NE in ga lahko preoblikujemo v enakovreden jezik.

Univerzalni Turingov stroj

Prevedba problema v enakovreden jezik, kodiranje TM oz. opis TM

Univerzalni Turingov stroj je čisto navaden Turingov stroj, ki kot vhod sprejme opis nekega drugega Turingovega stroja **M** in njegov vhod (**besedo w**), ter ta stroj na podanem vhodu simulira. Če beseda pripada jeziku tega podanega stroja, potem se ustavi v nekem končnem stanju.

Če potegnemo vzporednico s sodobnimi računalniki, je opis Turingovega stroja ekvivalenten programski kodi, vhod pa vhodnim parametrom tega programa.

Kodiranje TM oz. opis TM

Kodirna abeceda naj bo $\{0,1\}$.

$$0 = z_1, 1 = z_2, \sqcup = z_3$$

$$\delta(q_i, z_j) = (q_k, z_\ell, D_m) \text{ zakodiramo navodilo z besedo } K = 0^i 1^j 10^k 10^\ell 10^m$$

kjer $D_1 = L, D_2 = R, D_3 = S$.

Tako zakodiramo vsako navodilo δ .

Iz obstoječih kod $K_1, K_2, \dots, K_r \langle \delta \rangle = 111K_111K_211 \dots 11K_r111$

Univerzalni TM sprejema besede oblike: kodiran TM, ki mu sledi w
 $\langle TM \rangle \langle w \rangle$

Zapiši TS za jezik $L = \{0^{2k+1} : k \geq 0\}$ in ga zakodiraj po zgoraj opisanem postopku.

$$(q_1, 0) \rightarrow (q_3, 0, R)$$

$$(q_3, 0) \rightarrow (q_1, 0, R)$$

$$(q_3, B) \rightarrow (q_2, B, L)$$

Koda zgoraj opisanega stroja pa je:

$$111 \underbrace{01010001010}_{(q_1,0) \rightarrow (q_3,0,R)} 11 \underbrace{00010101010}_{(q_3,0) \rightarrow (q_1,0,R)} 11 \underbrace{000100100100100}_{(q_3,B) \rightarrow (q_2,B,L)} 111$$

Indeks besede je pretvorba besede v desetiški sistem.

Indeks Turingovega stroja je pretvorba opisa Turingovega stroja v desetiški sistem.

Če Turingov stroj nima pravilnega opisa, gre za Turingov stroj, ki se za vsako besedo ustavi v nekončnem stanju.

Odločljivost

Jezik L je **odločljiv (rekurziven)**, če obstaja TS, ki se za vsak

- $w \in L$ ustavi v končnem stanju
- $w \notin L$ ustavi v nekončnem stanju

Če je jezik odločljiv, je tudi polodločljiv.

Jezik L je **polodločljiv (rekurzivno nešteven, RE, Turingov)**, če obstaja TS, ki se za vsak $w \in L$ ustavi v končnem stanju.

Če je jezik polodločljiv, je lahko tudi neodločljiv.

Jezik L je **neodločljiv (non-RE)**, če ni polodločljiv.

Nekaj lastnosti

1. Komplement odločljivega jezika je odločljiv.
2. Unija dveh odločljivih jezikov je odločljiv jezik.
3. Unija dveh polodločljivih jezikov je polodločljiv jezik.
4. Če je jezik L polodločljiv in je njegov komplement L tudi polodločljiv jezik, potem sta oba odločljiva.

Če imamo dva jezika L in L komplement, sta te dva jezika lahko:

- oba odločljiva
- noben ni polodločljiv
- eden je polodločljiv (in ni odločljiv), drugi pa ni polodločljiv

Odločljivi, polodločljivi, neodločljivi



Prevedbe

S pomočjo **prevedbe** lahko za neznan jezik/problem dokažemo, da je **neodločljiv** ali da **ni niti polodločljiv**.

Prevedba jezika/problema L_1 na jezik/problem L_2 je postopek, s katerim

- pozitivne primerke problema (besede v jeziku) L_1 preslikamo v pozitivne primerke problema (besede v jeziku) L_2
- negativne primerke problema (besede izven jezika) L_1 preslikamo v negativne primerke problema (besede izven jezika) L_2
- $w \in L_1 \Leftrightarrow f(w) \in L_2$

Če neka beseda pripada jeziku L_1 potem prevedba te besede (funkcija f) pripada jeziku L_2 .

Če beseda ne pripada jeziku L_1 potem prevedba te besede ne pripada jeziku L_2 .

Prevedba problema L_1 na problem L_2 je izdelava algoritma za problem L_1 s pomočjo algoritma za problem L_2 . Pri **dokazovanju neodločljivosti** uporabljam prevedbe zato, da nas privedejo do protislovja. Tipičen postopek dokazovanja neodločljivosti s prevedbami lahko opišemo z nekaj koraki:

1. Vemo, da za problem L_1 ne obstaja algoritem.
2. Predpostavimo, da za problem L_2 obstaja algoritem.
3. S pomočjo algoritma L_2 sestavimo algoritem za problem L_1 .
4. To pa nas privede do protislovja s točko 1. Predpostavka, da za problem L_2 obstaja algoritem je bila torej napačna.

Prevedba mora biti **izračunljiva**: obstajati mora Turingov stroj (algoritem), ki jo izvrši

Se pravi mora obstajati Turingov stroj, ki izvrši to prevedbo oz. se ustavi za vsako besedo, ki je v jeziku L_1 oz. za vsako besedo, ki ni v jeziku L_1 . V končnem času zapiše prevedbo te besede na trak.

L_1 prevedemo na L_2 .

Če je L_1 neodločljiv, je tudi L_2 neodločljiv.

Če L_1 ne-polodločljiv, je L_2 ne-polodločljiv.

Če nas za neznan jezik L_x zanima, ali je neodločljiv (oz. ne-polodločljiv), vzamemo znan neodločljiv (oz. ne-polodločljiv jezik) L in zgradimo prevedbo $L \Rightarrow L_x$.

Znan neodločljiv problem/ne-polodločljiv problem prevedemo na neznan problem.

znan problem \Rightarrow neznan problem

Prevedba $L_1 \rightarrow L_2$ pomeni, da je L_2 vsaj toliko "težak" kot L_1 in L_1 vsaj toliko "lahek" kot L_2 .

L_1 odločljiv $\rightarrow > L_2$ karkoli

L_1 polodločljiv $\rightarrow L_2$ polodločljiv ali neodlodločljiv

L_1 neodločljiv $\rightarrow L_2$ neodločljiv

L_2 odločljiv $\rightarrow L_1$ odločljiv

L_2 polodločljiv $\rightarrow L_1$ odločljiv ali polodločljiv

L_2 neodločljiv $\rightarrow L_1$ karkoli

"Programerski" primer

L₁: iskanje minimuma tabele

L₂: urejanje tabele

Če prevedemo problem iz L₁ na L₂ ($L_1 \Rightarrow L_2$) imamo stroj, ki zna tabele urejati (stroj za jezik L₂).

M(L₂) sprejme neko tabelo in jo uredi.



Problem L₁ želimo prevesti na problem L₂:

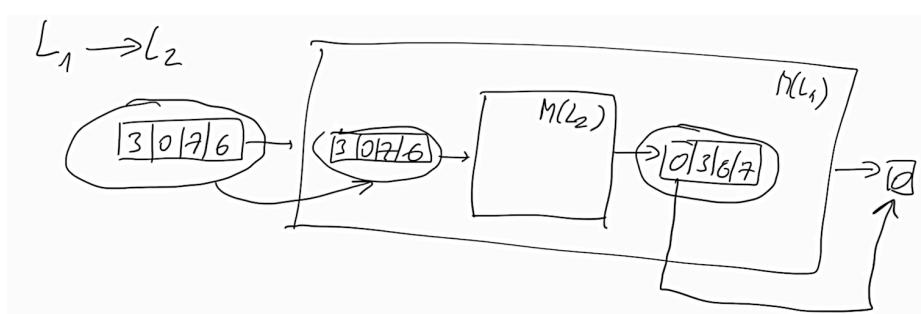
Kako zgradimo stroj za jezik L₁, pri čemer si pomagamo s strojem za jezik L₂?

Stroj za jezik L₁ sprejme neko tabelo in mora vrniti najmanjši element te tabele.

Kako izpeljemo to prevedbo?

Tabelo, ki jo podamo na vhod M (L₁) - stroj, ki išče minimum, podamo stroju za urejanje tabele M (L₂), nato stroj za urejanje vrne urejeno tabelo.

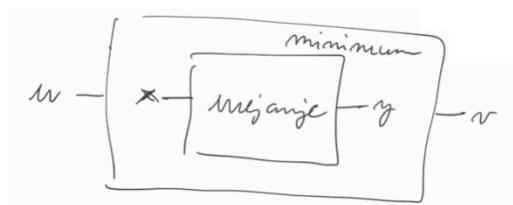
Sedaj vzamemo le še prvi element urejene tabele in to je rezultat stroja M (L₁).



Na ta način lahko prevedemo problem iskanja minimuma na problem urejanja tabele.

Imamo stroj za urejanje tabele, ki sprejme nek vhod x in kot rezultat vrne y.

Sedaj smo s pomočjo tega stroja za urejanje zgradili stroj za iskanje minimuma, ki sprejme vhod u in kot rezultat vrne v.



Kako smo naredili to prevedbo?

x = u

v = y [0] // prvi element tabele

Podano imamo množica A in funkcijo s: $s: s \rightarrow \mathbb{Z}^+$.

Ali obstaja razbitje množice A na dve podmnožici A' in A'', za katere velja $A' \cup A'' = A$ in $A' \cap A'' = \emptyset$ ter

$$\sum_{a \in A'} s(a) = \sum_{a \in A''} s(a)?$$

1. Definirati moramo kodiranje problema. Problem predstavimo kot niz, kjer so zapisane velikosti predmetov v eniškem zapisu, ločene z znakom #. Dogovorimo se, da začnemo niz tudi z znakom #.

Problem

$$A = \{1, 2, 3\}$$

$$s(1) = 2, s(2) = 3, s(3) = 2$$

bi npr. zakodirali z nizom: #11#111#11

2. Zapišemo nedeterministični TS, ki sprejema vse naloge zgoraj definiranega problema.

$$(q_0, \#, B, B) \rightarrow (q_1, (\#, B), (B, R), (S, S)) // \text{nedet. izberemo prepis na 2. ali 3. trak}$$

$$(q_0, \#, B, B) \rightarrow (q_2, (\#, B), (B, R), (S, S)) // \text{nedet. izberemo prepis na 2. ali 3. trak}$$

$$(q_1, 1, B, B) \rightarrow (q_1, (1, 1), (B, R), (R, S)) // \text{trenutni element na drugi trak}$$

$$(q_2, 1, B, B) \rightarrow (q_2, (1, B), (1, R), (S, R)) // \text{trenutni element na tretji trak}$$

$$(q_1, \#, B, B) \rightarrow (q_0, (\#, B), (B, S), (S, S)) // \text{ponovimo izbiro}$$

$$(q_2, \#, B, B) \rightarrow (q_0, (\#, B), (B, S), (S, S)) // \text{ponovimo izbiro}$$

$$(q_1, B, B, B) \rightarrow (q_3, (B, B), (B, S), (L, L)) // \text{konec vhoda - izbrali smo } A'$$

$$(q_2, B, B, B) \rightarrow (q_3, (B, B), (B, S), (L, L)) // \text{konec vhoda - izbrali smo } A'$$

$$(q_3, B, 1, 1) \rightarrow (q_3, (B, 1), (1, S), (L, L)) // \text{preverimo če sta 2. in 3. trak enaka}$$

$$(q_3, B, B, B) \rightarrow (q_F, (_, _), (_, _), (_, _)) // \text{preverimo če sta 2. in 3. trak enaka}$$

Drugi trak hrani izbrano množico A' , tretji trak pa množico $A \setminus A'$. Za vsak element množice se stroj nedeterministično odloči ali prepisati predmet (oz. velikost predmeta) na drugi trak ali na tretji trak. Ko zaključi nedeterministično izbiro mora še preveriti ali sta velikosti A' in $A \setminus A'$ enaki. Če sta enaki, je stroju uspelo najti ustrezno razbitje množice A.

ALI

Primer

$$A = \{2, 5, 7, 3, 1\}$$

$$A' = \{7, 2\}, A'' = \{5, 3, 1\}$$

Vhod: 00#00000#0000000#000#0#

Preiskovalni prostor je, da moramo preizkušati vse možne podmnožice in pogledati, če se seštejejo v drugo polovico.

Rešujemo z 3-tračnim TM.

Prvi trak: vhod

Drugi trak: množica A'

Tretji trak: množica A''

Ideja

Za vsak element se ned. odločimo, ali spada v A' ali v A'' .

Ena odločitev, kam kaj spada bo tista, ki bo problem rešila. Nedeterminizem bo generiral vse možne podmnožice. Na koncu preverimo, če je vsota A' enaka vsoti A'' - se vrnemo nazaj po obeh trakovih hkrati in preverjamo, če imamo enako število ničel.

Realizacija problema z TM

Začnemo z prepisovanjem na 2. ali 3. trak.

V stanju q_1 prepisujemo element na 2. trak, v stanju q_2 pa na 3. trak.

1. $(q_0, a, B, B) \rightarrow (q_1, (a, R), (a, R), (B, S))$ // nedet. izberemo prepis na 2. trak
2. $(q_0, a, B, B) \rightarrow (q_2, (a, R), (B, S), (a, R))$ // nedet. izberemo prepis na 3. trak

Prepisujemo na 1. ali 2. trak.

3. $(q_1, a, B, B) \rightarrow (q_1, (a, R), (a, R), (B, S))$ // prepisovanje A' ali A''
4. $(q_2, a, B, B) \rightarrow (q_2, (a, R), (B, S), (a, R))$ // prepisovanje A' ali A''

Ko naletimo na $\#$, se vrnemo nazaj na q_0 , kjer spet naredimo nedeterministično odločitev - ali prepišemo naslednji znak, ali ne (zanka, ki generira drevo z 2^n listi).

5. $(q_1, \#, B, B) \rightarrow (q_0, (\#, R), (B, S), (B, S))$ // prepisovanje A' ali A''
6. $(q_2, \#, B, B) \rightarrow (q_0, (\#, R), (B, S), (B, S))$ // prepisovanje A' ali A''

Z obema trakova se pomaknemo levo. Oba stojita na zadnjem a-ju. V stanju q_3 preverjamo, ali je na 2. in 3. traku enako število a-jev.

7. $(q_0, B, B, B) \rightarrow (q_3, (B, S), (B, L), (B, L))$ // preverjanje, ali $A' == A''$
8. $(q_3, B, a, a) \rightarrow (q_3, (B, S), (a, L), (a, L))$ // preverjanje, ali $A' == A''$

Če je a-jev enako, dobimo istočasno B.

9. $(q_3, B, B, B) \rightarrow (q_F, (_, _), (_, _), (_, _))$ // preverjanje, ali $A' == A''$

Preveri, ali je podano število največje v nekem zaporedju števil.

Poglejmo si, kako tak problem definiramo in kako ga s Turingovim strojem rešimo.

Jezik, ki bi ga radi prepoznali:

$$L = \{x \# a_1 \# a_2 \# \dots \# a_n \mid x \text{ je največji element v zaporedju } a_1 \dots a_n\}$$

Ševela naj bodo v tej besedi zakodirana v eniškem sistemu, primer besede, ki pripada temu jeziku je npr.

1111#11#1#1111. Postopek prepoznavanja bomo izvedli z dvotračnim Turingovim strojem. Na drugem traku bomo hranili trenutno najdaljši element. Vsakič, ko bomo naleteli na znak #, drugi trak prevrtnimo na začetek in začnemo z zapisovanjem novega elementa, če bo ta daljši od prejšnjega najdaljšega, se bo zapisal na trak. Ko pregledamo celotno zaporedje, imamo na drugem traku zapisan najdaljši element zaporedja. Oba trakova prevrtnimo na začetek in preverimo, če sta najdaljši dobljeni element in x enaka.

$$(q_0, 1, B) \rightarrow (q_0, (1, B), (R, S)) // \text{preskoči } x$$

$$(q_0, \#, B) \rightarrow (q_1, (\#, B), (R, S)) // \text{preskoči } x$$

$$(q_1, 1, B) \rightarrow (q_1, (1, 1), (R, R)) // \text{trenutni element na drugi trak}$$

$$(q_1, 1, 1) \rightarrow (q_1, (1, 1), (R, R)) // \text{trenutni element na drugi trak}$$

$$(q_1, \#, 1) \rightarrow (q_2, (\#, 1), (S, L)) // \text{prevrtnimo drug trak}$$

$$(q_1, \#, B) \rightarrow (q_2, (\#, B), (S, L)) // \text{prevrtnimo drug trak}$$

$$(q_2, \#, 1) \rightarrow (q_2, (\#, 1), (S, L)) // \text{prevrtnimo drug trak}$$

$$(q_2, \#, B) \rightarrow (q_1, (\#, B), (R, R)) // \text{prevrtnimo drug trak}$$

$$(q_1, B, 1) \rightarrow (q_3, (B, 1), (L, L)) // \text{na koncu zaporedja}$$

$$(q_1, B, B) \rightarrow (q_3, (B, B), (L, L)) // \text{na koncu zaporedja}$$

$$(q_3, 1, 1) \rightarrow (q_3, (1, 1), (L, L)) // \text{prevrtnimo oba trakova}$$

$$(q_3, 1, B) \rightarrow (q_3, (1, B), (L, L)) // \text{prevrtnimo oba trakova}$$

$$(q_3, \#, 1) \rightarrow (q_3, (\#, 1), (L, L)) // \text{prevrtnimo oba trakova}$$

$$(q_3, \#, B) \rightarrow (q_3, (\#, B), (L, L)) // \text{prevrtnimo oba trakova}$$

$$(q_3, B, B) \rightarrow (q_4, (B, B), (R, R)) // \text{prevrtnimo, če sta } x \text{ in 2. trak enaka}$$

$$(q_4, 1, 1) \rightarrow (q_4, (1, 1), (R, R)) // \text{prevrtnimo, če sta } x \text{ in 2. trak enaka}$$

$(q_4, \#, B) \rightarrow (q_F, (_, _), (_, _))$ // prevrtnimo, če sta x in 2. trak enaka

Vidimo lahko, da med reševanjem odločitvenega problema dejansko moramo najti največje število, kar je tista rešitev, ki jo v realnosti največkrat potrebujemo. Pri tem problemu lahko dobimo malo občutka, da odločitveni problemi niso tako zelo daleč od t.i. konstrukcijskih problemov, kjer rešitve ne zgolj preverjamo, ampak jo moramo zgraditi.

Problem 0-1 nahrbtnika je klasični optimizacijski problem, ki se velikokrat pojavlja tudi v praksi.

Poglejmo si, kako tak optimizacijski problem definiramo kot odločitveni in kako ga rešujemo z nedeterminističnim Turingovim strojem. Problem 0-1 nahrbtnika je definiran na naslednji način: imamo množico predmetov $A = \{a_1, a_2, \dots, a_n\}$ in funkcijo velikosti teh predmetov

$$s : A \rightarrow \mathbb{Z}^+$$

ter funkcijo vrednosti predmetov

$$v : A \rightarrow \mathbb{Z}^+.$$

Podano imamo tudi celo število V , ki predstavlja prostornino našega nahrbtnika, ki ga hočemo čim bolj donosno napolniti. Iščemo torej podmnožico $A' \subseteq A$, za katero velja:

$$\sum_{a \in A'} s(a) \leq V,$$

pri čemer želimo najti tako podmnožico A' , da bo vrednost

$$\sum_{a \in A'} v(a)$$

čimvečja. Tak problem enostavno predelamo v odločitvenega tako, da mu dodamo en dodaten parameter K in postavimo vprašanje: ali obstaja taka podmnožica A' , ki zadošča omejitvi kapacitete nahrbtnika, in velja

$$\sum_{a \in A'} v(a) \geq K.$$

Ta prevedba je primerek klasične transformacije optimizacijskega problema v odločitveni problem.

1. Najprej definirajmo kodiranje za nalogo. Vsa števila bomo zaradi enostavnosti zapisovali v eniškem sistemu:

$$V\#K\$s_1\$v_1\$s_2\$v_2#\dots\$s_n\$v_n$$

2. Turingov stroj

($q_0, 1, B, B$) \rightarrow ($q_0, (1, B), (B, R), (S, S)$) // preskočimo V in K
($q_0, \#, B, B$) \rightarrow ($q_1, (\#, B), (B, R), (S, S)$) // preskočimo V in K
($q_1, 1, B, B$) \rightarrow ($q_1, (1, B), (B, R), (S, S)$) // preskočimo V in K
($q_1, \#, B, B$) \rightarrow ($q_2, (\#, B), (B, R), (S, S)$) // izbira ali prepišemo a_i (q_2) ali ne (q_3)
($q_1, \#, B, B$) \rightarrow ($q_3, (\#, B), (B, R), (S, S)$) // izbira ali prepišemo a_i (q_2) ali ne (q_3)
($q_2, 1, B, B$) \rightarrow ($q_2, (1, 1), (B, R), (R, S)$) // prepišemo s_i na drugi trak
($q_2, \$, B, B$) \rightarrow ($q_4, (\$, B), (B, R), (S, S)$) // prepišemo s_i na drugi trak
($q_4, 1, B, B$) \rightarrow ($q_4, (1, 1), (B, R), (R, S)$) // prepišemo v_i na tretji trak
($q_4, \#, B, B$) \rightarrow ($q_1, (\#, B), (B, S), (S, S)$) // prepišemo v_i na tretji trak
($q_3, 1, B, B$) \rightarrow ($q_3, (1, B), (B, R), (S, S)$) // preskočimo ta predmet
($q_3, \$, B, B$) \rightarrow ($q_3, (\$, B), (B, R), (S, S)$) // preskočimo ta predmet
($q_3, \#, B, B$) \rightarrow ($q_1, (\#, B), (B, S), (S, S)$) // preskočimo ta predmet
(q_4, B, B, B) \rightarrow ($q_5, (B, B), (B, L), (L, L)$) // končali smo izbiro rešitve

$(q_3, B, B, B) \rightarrow (q_5, (B, B), (B, L), (L, L))$ // končali smo izbiro rešitve

Nedeterministično smo na drugi in tretji trak izbrali podmnožico A' . Naslednja faza Turingovega stroja je preverjanje, če ta rešitev ustreza podanim omejitvam.

$(q_5, 1/\$/\#, 1, 1) \rightarrow (q_5, (1/\$/\#, B), (B, L), (L, L))$ // prevrtnimo trakove

$(q_5, 1/\$/\#, 1, B) \rightarrow (q_5, (1/\$/\#, B), (B, L), (L, S))$ // prevrtnimo trakove

$(q_5, 1/\$/\#, B, 1) \rightarrow (q_5, (1/\$/\#, B), (B, L), (S, L))$ // prevrtnimo trakove

$(q_5, 1/\$/\#, B, B) \rightarrow (q_5, (1/\$/\#, B), (B, L), (S, S))$ // prevrtnimo trakove

$(q_5, B, B, B) \rightarrow (q_6, (B, B), (B, R), (R, R))$ // preverimo, če drugi trak $\leq V$

$(q_6, 1, 1, 1) \rightarrow (q_6, (1, 1), (1, R), (R, S))$ // preverimo, če drugi trak $\leq V$

$(q_6, 1, B, 1) \rightarrow (q_6, (1, B), (1, R), (S, S))$ // preverimo, če drugi trak $\leq V$

$(q_6, \#, B, 1) \rightarrow (q_7, (\#, B), (1, R), (S, S))$ // preverimo, če tretji trak $\geq K$

$(q_7, 1, B, 1) \rightarrow (q_7, (1, B), (1, R), (S, S))$ // preverimo, če tretji trak $\geq K$

$(q_7, \#, B, 1) \rightarrow (q_F, (_, _), (_, _), (_, _))$ // preverimo, če tretji trak $\geq K$

$(q_7, \#, B, B) \rightarrow (q_F, (_, _), (_, _), (_, _))$ // preverimo, če tretji trak $\geq K$

Naj bo A množica predmetov z vrednostjo v: $A \Rightarrow Z^+$ in maso m: $A \Rightarrow Z^+$.

Iz množice A želimo izbrati nabor predmetov, ki ne bo presegal mase M in bo imel čim večjo skupno vrednost. Namig: optimizacijski problem prevedi na odločitvenega – ali obstaja nabor predmetov s skupno vrednostjo V, ki ne presega mase M?

S tem, da smo dodali V smo spremenili optimizacijski problem v odločitveni problem.

$$A = \{ (v_1, m_1), (v_2, m_2), \dots, (v_R, m_R) \}$$

$$M = 3$$

$$V = 2$$

$$A' \subseteq A$$

$$\sum_{i \in A'} m_i \leq M, \quad \sum_{i \in A'} v_i \geq V$$

KODIRANJE (primera zgoraj)

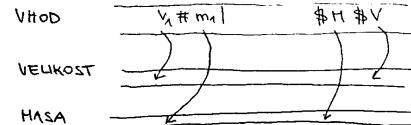
$v_1 \# m_1 \mid v_2 \# m_2 \mid \dots \# M \# V$
 $0 \# 00 \mid 00 \# 000 \mid 00 \# 0 \# 000 \# 00$

Ideja

Prvi trak: vhod

Drugi trak: vrednost predmetov, ki smo jih izbrali

Tretji trak: masa trenutno izbranih predmetov



Preveriti moramo, kateri predmeti gredo v nahrbtnik.

Nedeterministično bomo izbrali, ali predmet gre v nahrbtnik ali ne gre.

Ko gre v nahrbtnik, prepišemo v_1 na drugi trak, m_1 pa na tretji trak.

Na vsakem traku hranimo trenutno vsoto vseh vrednosti ter maso predmetov, ki smo jih izbrali.

Če predmeta ne damo v nahrbtnik, ignoriramo vhod do naslednje navpičnice, nič ne zapisujemo.

Ko na koncu pridemo do mase, najprej preverimo, če je na tretjem traku manj m-jev kot na 1. traku (M).

Nato preverimo, ali je vrednost večja, kot tista ki je podana kot vhod.

1. $(q_0, 0, B, B) \rightarrow (q_1, (0, R), (0, R), (B, S))$ // nedet., predmet v nahrbtnik
2. $(q_0, 0, B, B) \rightarrow (q_2, (0, R), (B, S), (B, S))$ // nedet., ignorira vhod
3. $(q_2, \#/0, B, B) \rightarrow (q_2, (\#/0, R), (B, S), (B, S))$ // ignorira
4. $(q_1, 0, B, B) \rightarrow (q_1, (0, R), (0, R), (B, S))$ // prepis vrednosti
5. $(q_1, \#, B, B) \rightarrow (q_3, (\#, R), (B, S), (B, S))$
6. $(q_3, 0, B, B) \rightarrow (q_3, (0, R), (B, S), (0, R))$ // prepis mase
7. $(q_2, 1, B, B) \rightarrow (q_0, (1, R), (B, S), (B, S))$ // vrnemo na začetek, ponovitev zanke (ned.)

8. $(q_3, 1, B, B) \rightarrow (q_0, (1, R), (B, S), (B, S))$ // vrnemo na začetek, ponovitev zanke (ned.)

Kdaj končamo z nedeterminističnim zbiranjem in začnemo s preverjanjem? Ko pridemo do \$.

9. $(q_2/q_3, \$, B, B) \rightarrow (q_4, (\$, R), (B, S), (B, L))$

10. - 14: preverjanje mase ($M > \text{sum}$ (izbrani predmeti))

10. $(q_4, 0, B, 0) \rightarrow (q_4, (0, R), (B, S), (0, L))$

11. $(q_4, 0, B, B) \rightarrow (q_5, (0, R), (B, S), (B, S))$

12. $(q_4, \#, B, B) \rightarrow (q_6, (\$, R), (B, L), (B, S))$

13. $(q_5, 0, B, B) \rightarrow (q_5, (0, R), (B, S), (0, S))$

14. $(q_5, \#, B, B) \rightarrow (q_6, (\$, R), (B, L), (B, S))$

15. - 17.: preverjanje vrednosti

15. $(q_6, 0, 0, B) \rightarrow (q_6, (0, R), (0, L), (B, S))$

16. $(q_6, B, 0, B) \rightarrow (q_F, (_, _), (_, _), (_, _))$

17. $(q_6, B, B, B) \rightarrow (q_F, (_, _), (_, _), (_, _))$

Zapišite TS, ki sprejme opise TS, ki imajo vsaj en prehod iz nekončnega v končno stanje.

$L = \{\langle M \rangle : M \text{ ima vsaj en prehod iz nekončnega v končno stanje}\}$

Eno binarno število prestavlja en Turingov stroj.

$\delta(q_x, \text{karkoli}) \rightarrow (q_2, \text{karkoli}, \text{karkoli})$

$x \neq 2$

Z regularnimi izrazi:

$(0 + 0000^*)100*100100*100*$

Naj bo $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ Turingov stroj, funkcija prehodov pa naj bo dana z

- $\delta(q_1, 1) = (q_3, 0, R)$
- $\delta(q_3, 0) = (q_1, 1, R)$
- $\delta(q_3, 1) = (q_2, 0, R)$
- $\delta(q_3, B) = (q_3, 1, L).$

Zapiši kodo tega Turingovega stroja.

$q_1: 0, q_2: 00, q_3: 000$

$0: 0, 1: 00, B: 000$

$L: 0, R: 00, S: 000$

$\langle T \rangle = 11101001000101001100010101001001100010010010100110001000100010010111$

Recimo, da je odločitveni problem P_1 prevedljiv na odločitveni problem P_2 . Pokaži, da tedaj velja

1. če je P_1 neodločljiv, tedaj je P_2 neodločljiv
2. če P_1 ni polodločljiv, tedaj tudi P_2 ni polodločljiv

(1) P_1 neodločljiv $\Rightarrow P_2$ neodločljiv

Trditve se lotimo s protislovjem.

Predpostavka: P_2 je odločljiv

Če je P_2 odločljiv in imamo takšno prevedbo narejeno, bo ta odgovor tudi odgovor za P_1 .

P_2 odločljiv $\Rightarrow P_1$ odločljiv

nesmiselno, ker vemo da je P_1 neodločljiv, predpostavka je bila napačna

Problem P_2 mora bit vsaj toliko težek kot problem P_1 .

Če je P_1 neodločljiv, potem je neodločljiv tudi P_2 (P_2 je težji od prvega).

(2) P_1 ni polodločljiv $\Rightarrow P_2$ ni polodločljiv

Polodločljivi problemi so podmnožica neodločljivih problemov.

Predpostavka: P_2 je polodločljiv

Če je P_2 polodločljiv in takšna prevedba obstaja (problema sta prevedljiva), potem znamo izhod iz M_2 problema P_2 uporabiti kot izhod za M_1 problema P_1 .

Če imamo polodločljiv problem P_2 , ki ima TM M_2 (ustavi se pri vseh pozitivnih primerkih), potem mora biti tudi P_1 polodločljiv.

P_1 bi moral biti neodločljiv \Rightarrow protislovje

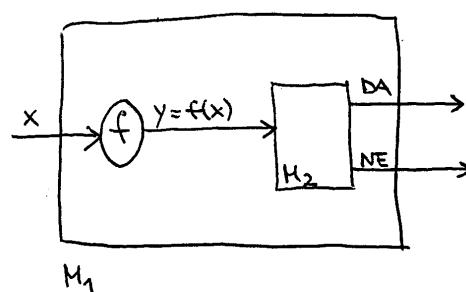
Če imamo prevedbo iz P_1 na P_2 potem mora biti P_2 vsaj toliko težek kot P_1 .

Če je P_1 odločljiv, potem je P_2 lahko odločljiv, polodločljiv ali neodločljiv.

Če je P_1 polodločljiv, potem P_2 ne more biti odločljiv (če bi bil P_2 odločljiv, bi bil odločljiv tudi P_1).

Če je P_1 neodločljiv, potem je P_2 neodločljiv.

Če nas zanima, kakšen je nek jezik (odločljiv, polodločljiv, neodločljiv) bomo uporabili eno izmed zgornjih trditev in na ta način prevedli problema (enega na drugega).



Diagonalni jezik ni polodločljiv.

$$L_d = \{w_i \mid w_i \notin L(M_i)\}$$

Vsek Turingov stroj se da zakodirati kot neko binarno število. Zato lahko Turingove stroje uredimo glede na število, ki ga njegova koda predstavlja. Seveda je takih urejenosti neskončno različnih, kar pa za sam dokaz ni pomembno, važno je, da taka urejenost obstaja. Besede iz Σ^* ravno tako lahko uredimo glede na neko urejenost. Vse možne jezike, ki jih lahko prepoznavamo s Turingovimi stroji tako lahko zapišemo v tabeli, kot je prikazana na spodnji sliki.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	
M_1	1	0	0	1	1	1	1	0	...
M_2	1	0	0	1	1	0	0	0	...
M_3	0	0	0	0	0	0	1	0	...
M_4	0	1	1	0	0	1	1	0	...
M_5	0	0	1	0	1	0	0	1	...
M_6	1	1	1	1	1	0	1	1	...
M_7	1	1	1	1	0	0	1	0	...
M_8	0	0	1	0	1	1	0	0	...
:	:	:	:	:	:	:	:	:	...

Celica (i, j) v tej tabeli predstavlja odgovor Turingovega stroja M_i , če dobi na vhodu besedo w_j : če je odgovor 0, beseda ne pripada temu jeziku, če pa je odgovor 1, potem beseda pripada jeziku, ki ga predstavlja posamezna vrstica. Pokazati pa moramo, da obstaja jezik, ki ga gotovo ni v tej tabeli. Tak jezik skonstruiramo tako, da pogledamo vrednosti na diagonali te tabele (osenčene celice na sliki) in vzamemo ravno nasprotne vrednosti. Tako dobimo diagonalni jezik, ki vsebuje vse tiste besede w_i , ki niso v jeziku $L(M_i)$, oz. bolj formalno: $L_d = \{w_i \mid w_i \notin L(M_i)\}$

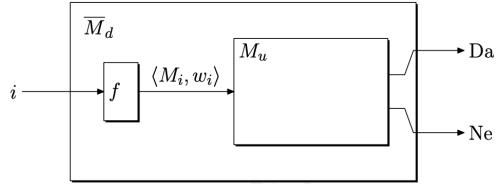
Ta jezik gotovo ni v zgornji tabeli (torej ni polodločljiv oz. Turingov), saj bi moral zanj obstajati Turingov stroj M_k (stroj na k-ti poziciji). Vendar bi ta stroj besedo w_k sprejemal natanko takrat, ko bi jo stroj M_k zavrnil. S tem smo pa prišli do protislovja, kar dokazuje, da tak stroj res ne obstaja.

Dokaži, da je jezik $L_u = \{(M, w) \mid w \in L(M)\}$ neodločljiv.

Neodločljivost jezika L_u bomo dokazali s prevedbo jezika $\neg L_d$ na L_u .

$$\neg L_d \rightarrow L_u$$

Predpostavimo, da je jezik L_u odločljiv - z njegovo pomočjo sestavimo algoritem za $\neg L_d$.



Vhod hipotetičnega stroja $\neg M_d$ predstavlja neko število i , ki je element jezika $\neg L_d$ natanko takrat, ko stroj M_i sprejme besedo w_i . Funkcija f iz vhodnega števila i zgenerira opis stroja M_i in besede w_i in ta dva opisa poda kot vhod univerzalnemu stroju M_u . Če bi stroj M_u znal odločati o vseh možnih vhodih, bi znal tudi stroj $\neg M_d$. Vemo pa, da je $\neg M_d$ neodločljiv. Prišli smo do protislovja. Lu ne more biti odločljiv.

Dokaži, da so odločljivi jeziki zaprti za unijo, presek, komplement in Kleenovo zaprtje.

Vemo, da sta jezika L_1 in L_2 odločljiva.

Obstaja stroj M_1 , ki sprejema L_1 in se vedno ustavi in vrne DA ali NE.

Obstaja stroj M_2 , ki sprejema L_2 in se vedno ustavi in vrne DA ali NE.

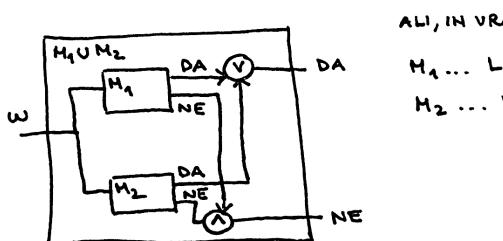


Unija dveh odločljivih jezikov je odločljiv jezik.

Če na kateremkoli izhodu M_1 ali M_2 dobimo DA, je rezultat unije DA.

Če hkrati na izhodu M_1 in M_2 dobimo NE, je rezultat unije NE.

Na ta način dobimo stroj za unijo teh dveh jezikov.



ALI, IN VRATA

$M_1 \dots L_1$
 $M_2 \dots L_2$

Imamo funkcijo bool M_1 , ki dobi x in vrne true ali false.

```
bool M1 (x)
```

Imamo funkcijo bool M_2 , ki dobi x in vrne true ali false.

```
bool M2 (x)
```

Nato naredimo funkcijo $M_1_unija_M_2$, ki dobi x in vrne true ali false.

```
bool M1_unija_M2 {  
    return M1 (x) || M2 (x)  
}
```

Ker sta L_1 in L_2 odločljiva, obstajata Turingova stroja M_1 in M_2 , ki odločita, ali velja $w \in L_1$ oziroma $w \in L_2$. Torej je tudi $L_1 \cup L_2$ odločljiv, saj lahko naredimo Turingov stroj M' , ki simulira M_1 in M_2 pri vhodu w v M' (oba se ustavita v končnem številu korakov) in na koncu se odloči, ali velja $w \in L_1 \cup L_2$.

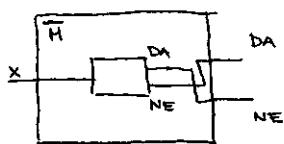
Presek dveh odločljivih jezikov je odločljiv jezik.

```
bool M1_presek_M2 {
    return M1(x) && M2(x)
}
```

Komplement odločljivega jezika je tudi odločljiv jezik.

Če je L odločljiv, obstaja Turingov stroj M , ki odloči, ali je $w \in L$. Potem je tudi $\neg L$ odločljiv, saj lahko naredimo Turingov stroj M' , ki požene M (ta se ustavi v končnem številu korakov, drugače L ne bi bil odločljiv) in vrne obraten odgovor, se pravi, odloči, ali velja $w \in \neg L$.

Izhod notranje škatle "negiramo".



Kleenovo zaprtje

Imamo jezik L , ki je odločljiv. Se pravi, imamo nek stroj M , ki zna reči DA in NE.

Za L^* nas zanima kakšen bi bil stroj M^* .

Uporaba nedeterminizma.

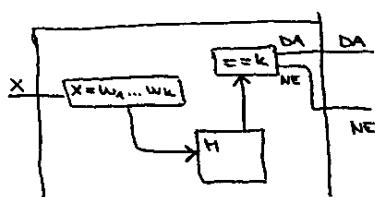
Imamo eno veliko škatlo, v kateri bomo uporabili M . Škatla dobi na vhod x (poljuben niz).

Iz tega niza moramo zgraditi nedeterministično razbitje.

Vsako od teh besed $w_1 \dots w_k$ damo na vhod stroju, ki zna razpoznavati M . Stroj M povežemo na preverjalnik.

Ideja je v tem, da znamo z nedeterminizmom preveriti vse možnosti.

Kleenovo zaprtje v bistvu pomeni, da moramo preveriti vsa možna razbitja iz katerih je lahko beseda x sestavljena.



Naj bosta L in njegov komplement $\neg L$ polodločljiva jezika. Dokaži, da sta L in $\neg L$ odločljiva.

Če je L polodločljiv, potem za njega obstaja stroj M, ki zna reči samo DA.

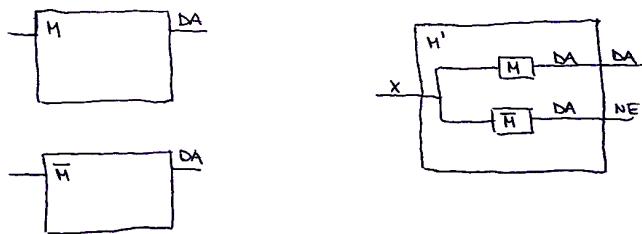
Za L komplement imamo nek drug stroj, ki zna reči DA.

Sestavimo nov stroj M' in $L(M') = L$.

Oba stroja, M in M komplement uporabimo v stroju M'. Vhod x damo obema vhod.

Ko M reče DA, M' reče DA. Ko komplement M reče DA, M' reče NE.

Dobimo oba odgovora in zato vemo, da je odločljiv.



Naj bo L_{ne} jezik Turingovih strojev M, za katere velja $L(M) \neq \emptyset$. Dokaži, da je L_{ne} polodločljiv jezik, ki ni odločljiv.

$$L_{ne} = \{\langle M \rangle \mid L(M) \neq \emptyset\}$$

L_{ne} je jezik vseh opisov TM-jev, ki sprejmejo vsaj eno besedo.

Dokaz polodločljivosti

Naredimo Turingov stroj, ki vzporedno simulira M pri večih vhodih in potrdi, da je $M \in L_{ne}$, ko se ena izmed simulacij ustavi z nepraznim izhodom. Torej obstaja Turingov stroj, ki potrdi, da je $L \in L_{ne}$.

Množica vseh možnih besed je števno neskončna (sistematicno generiramo vse možne besede (0, 1, 01, 10, ...)).

Torej obstaja Turingov stroj, ki potrdi, da je $L \in L_{ne}$. L_{ne} je polodločljiv.

$M_{ne}(M)$:

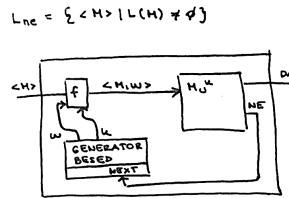
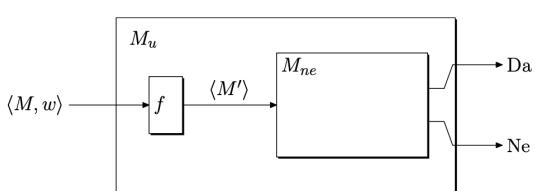
```
while (true) {
    w = gen();
    if (Mu(M, w)) // Mu poženemo na vhodu M in besedi, ki smo jo zgenerirali
        return true;
}
```

Dokaz neodločljivosti s prevedbo

Znan jezik (vemo, da je neodločljiv) L_u bomo prevedli na neznan jezik L_{ne} .

$$L_u \rightarrow L_{ne}$$

Predpostavimo, da je L_{ne} odločljiv.



Če par opisa ($\langle M \rangle, w$) pripada jeziku L_u potem mora prevedba para ($\langle M \rangle, w$) pripadati jeziku L_{ne} . (je izračunljiva). In obratno.

$w \in L_1 \Leftrightarrow f(w) \in L_2$ // za negativen in pozitiven primerek

$$(\langle M \rangle, w) \in L_u \Leftrightarrow f((\langle M \rangle, w)) \in L_{ne}$$

DA, če $w \in L(M) \Leftrightarrow DA$, če $L(M') \neq \emptyset$

$w \in L(M) \Leftrightarrow L(M') \neq \emptyset$

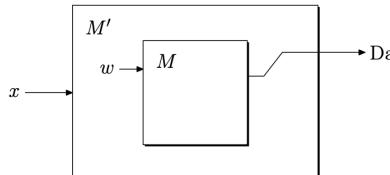
NE, če $w \notin L(M) \Leftrightarrow NE$, če $L(M') = \emptyset$

$w \notin L(M) \Leftrightarrow L(M') = \emptyset$

Iz vhoda $\langle M, w \rangle$ stroja M_u moramo sestaviti kodo novega Turingovega stroja M' .

Jezik stroja M' je odvisen od tega ali M sprejema besedo w ali ne ($\{\langle M, w \rangle \mid w \in L(M)\}$):

$$L(M') = \begin{cases} \Sigma^* & w \in L(M) \\ \emptyset & w \notin L(M) \end{cases}$$



$$L(M') \neq \emptyset \Leftrightarrow w \in L(M).$$

Kodo novega stroja damo na vhod stroja za jezik L_{ne} , ki bo ta opis sprejel natanko takrat, ko bi stroj M sprejel besedo w .

S strojem za jezik L_{ne} , bi lahko na zgoraj opisani način prepoznavali jezik L_u , kar smo pa že pokazali, da ni mogoče (jezik L_u je neodločljiv). Predpostavka je bila napačna. Iz tega sledi, da jezik L_{ne} tudi ne more biti odločljiv.

Naj bo L_e jezik Turingovih strojev M, za katere velja $L(M) = \emptyset$.

Dokaži, da prazni jezik ni polodločljiv.

Je jezik vseh opisov TM-jev, ki ne sprejmejo nobene besede.

1. rešitev

$\neg L_e$ je polodločljiv. Če je jezik $\neg L_e$ polodločljiv in njegov komplement L_e polodločljiv, potem sta oba odločljiva. Vemo pa, da $\neg L_e$ ni odločljiv $\Rightarrow L_e$ ne sme biti niti polodločljiv.

2. rešitev

Prevedba

$$\neg L_u (\text{ni polodločljiv}) \rightarrow L_e$$

Predpostavimo, da je L_e polodločljiv.

Iz vhoda $\langle M, w \rangle$ stroja $\neg M_u$ moramo sestaviti kodo novega Turingovega stroja M' .

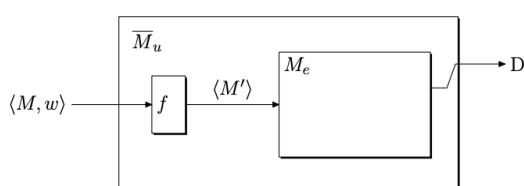
Jezik stroja M' je odvisen od tega ali M sprejema besedo:

$$\neg L_u = \{\langle M, w \rangle \mid w \notin L(M)\} \quad L(M') = \emptyset \Leftrightarrow w \notin L(M).$$

$$L_e = \{\langle M \rangle \mid L(M) = \emptyset\}$$

Stroj M' bo odgovoril DA, ko M ne bo sprejel besede.

Kodo stroja M' damo na vhod hipotetičnega stroja M_e in ta stroj bo kodo M' sprejel natanko takrat, ko beseda w ne bo v jeziku stroja M.

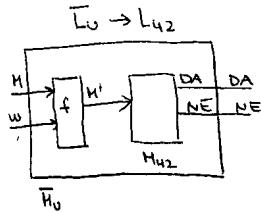


Z strojem za jezik L_{ne} , bi lahko prepoznali jezik $\neg L_u$, kar vemo, da je ne mogoče. Iz tega sledi, da jezik L_{ne} ne more biti polodločljiv. Je neodločljivo nepolodločljiv.

Za podani L_{42} dokaži, da ni polodločljiv. (Namig: prevedi $\neg L_u$ na L_{42} . Ne pozabi pokazati, kakšne M' generira funkcija f.)

$$L_{42} = \{< M >; |L(M)| = 42\}$$

L_{42} predstavlja jezik TM-jev, ki sprejemajo 42 različnih besed.



$$\neg L_u \text{ (ni polodočljiv)} \rightarrow L_{42}$$

Predpostavimo, da je L_{42} polodločljiv.

Iz vhoda $\langle M, w \rangle$ stroja M_u moramo sestaviti kodo novega Turingovega stroja M' , ki ga bomo dali kot vhod v M_{42} .

$\neg M_u$ vrne DA, ko beseda ne pripada jeziku M ($\{\langle M, w \rangle \mid w \notin L(M)\}$).

$$|S| = 42$$

$$|T| \neq 42 \text{ oz. } \Sigma^*$$

$$\text{DA, če } w \notin L(M) \iff \text{DA, če } L(M') = S$$

$$w \notin L(M) \iff L(M') = S$$

$$\text{NE, če } w \in L(M) \iff \text{NE, če } L(M') = \Sigma^*$$

$$w \in L(M) \iff L(M') = \Sigma^*$$

Polodločljiv (odgovor DA):

$$w \notin L(M) \iff |L(M')| = 42$$

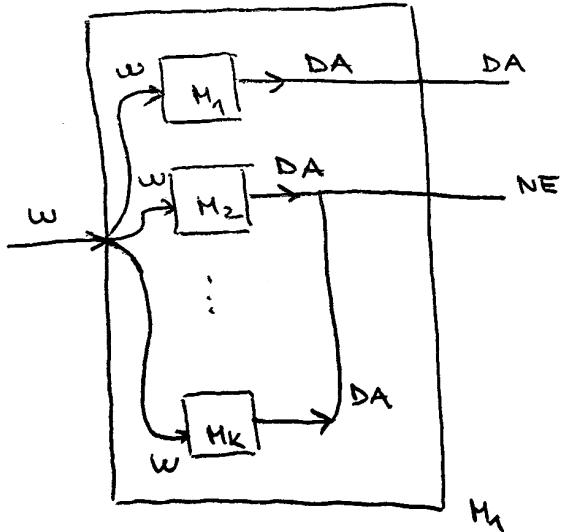
$$L(M') =$$

- $S; w \notin L(M)$ // sprijeme (samo tiste besede iz množice S (velikost množice je 42))
- $\Sigma^*; w \in L(M)$ // ne sprijeme/ne ustavi

Z strojem za jezik L_{42} , bi lahko prepoznali jezik $\neg L_u$, kar vemo, da je ne mogoče. Iz tega sledi, da jezik L_{42} ne more biti polodločljiv. Jezik je neodločljivo nepolodločljiv.

Naj bo Σ abeceda in naj bo $L_1 \cup L_2 \cup \dots \cup L_k = \Sigma^*$ razbitje vseh končnih nizov nad Σ , tj., za $i \neq j$ je $L_i \cap L_j = \emptyset$. Nadalje naj bo vsak od L_i polodločljiv.

Dokaži, da tedaj sledi, da so vsi od L_i odločljivi.



Vse možne besede razbijemo na jezike, $L_1 \cup L_2 \cup \dots \cup L_k$.

Unija vseh jezikov je celotna množica.

Če izberemo poljubno besedo, potem bo ta beseda spadala v natanko enega izmed teh jezikov.

Dokazati moramo, da so vsi te jeziki odločljivi.

L_1 in L_2 sta tu komplementa.

Za vsakega izmed teh jezikov zgradimo Turingov stroj. Vsaj en izmed teh strojev bo za poljubno besedo rekel DA.

Iz teh strojev naredimo nov stroj, ki bo odločljiv.

V nov stroj damo kot vhod besedo w in paralelno poženemo vse TM-je (besedo damo na vhod vsem strojem).

Če je beseda v M_1 , vrne DA.

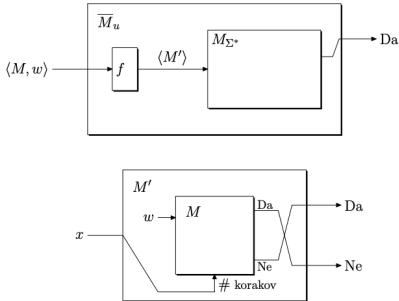
Če bo katerikoli drug stroj vrnil NE, potem vrnemo NE.

Vemo, da se bo vsaj eden od teh strojev zaključil, ker vemo da so vsi stroji polodločljivi in neka poljubna beseda spada vsaj v enega izmed teh jezikov. Vsaj eden izmed teh strojev vrne DA.

Dokaži, da jezik $L_{\Sigma^*} = \{\langle M \rangle \mid L(M) = \Sigma^*\}$ ni polodločljiv, neodločljiv.

$$\neg L_u \rightarrow L_{\Sigma^*}$$

Iz vhoda M in w sestavimo stroj M' , ki kot "podprogram" požene stroj M na besedi w . Vhodna beseda x v stroj M' pa določa koliko korakov naj izvede stroj M nad besedo w .



S tem dosežemo, da se ta stroj nikoli ne zacikla in vedno dobimo bodisi odgovor DA (stroj M sprejme besedo w v največ $|x|$ korakih) bodisi odgovor NE (stroju v $|x|$ korakih ne uspe sprejeti besede w). Ta dva odgovora speljemo na ravno nasprotna odgovora stroja M' . Analizirajmo kakšen jezik sprejema M' v odvisnosti od tega kaj naredi M nad besedo w .

1. Če stroj M ne sprejme besede w , potem tudi pri vsakem številu korakov izvajanja reče NE. Ker pa smo izhode stroja M povezali na obratne izhode stroja M' , ta stroj za vsak vhod reče DA.
2. Če stroj M sprejme besede w , potem gotovo obstaja neko končno število korakov (neka vhodna beseda x) v katerem stroj M reče DA, torej stroj M' reče NE.

$$L(M') = \begin{cases} \Sigma^* & w \notin L(M) \\ \neq \Sigma^* & w \in L(M). \end{cases}$$

Ali je neodločljiv?

Da.

Nanj prevedemo komplement $\neg L_u = \{\langle M, w \rangle \mid w \notin L(M)\}$, za katerega vemo, da je neodločljiv. Zgraditi moramo torej stroj $M'(w')$, tako da bo veljalo $L(M') = \Sigma^* \iff w \notin L(M)$.

1. rešitev:

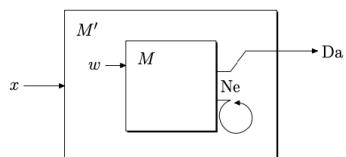
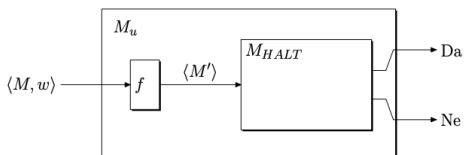
Naj bo $M_u(M, w)$ univerzalni Turingov stroj in naj bo $M'(w') = \neg M_u(M, w)$. Iz tega sledi implikacija $w \in L(M) \Rightarrow L(M') \neq \Sigma^*$, oziroma ekvivalentno, $L(M') = \Sigma^* \Rightarrow w \notin L(M)$. Implikacija v drugo smer pa ne velja, saj se lahko zgodi, da se M_u ne ustavi. Rešitev leži prav v omejitvi delovanja M_u .

2. rešitev:

Naj bo $M_u(M, w, k)$ univerzalni Turingov stroj, ki sprejme vhod, če M sprejme w v manj kot k korakih. Naj bo $M'(w') = \neg M_u(M, w, |w'|)$. Če $w \notin L(M)$, bo M' sprejel vhod ne glede na w' , zato $L(M') = \Sigma^*$. Tudi prejšnja implikacija v tem primeru še vedno drži, saj v primeru $w \in L(M)$ vedno obstaja w' zadostne dolžine.

Dokaži, da jezik $L_{HALT} = \{\langle M \rangle \mid M \text{ se ustavi na vseh vhodih}\}$ ni odločljiv.

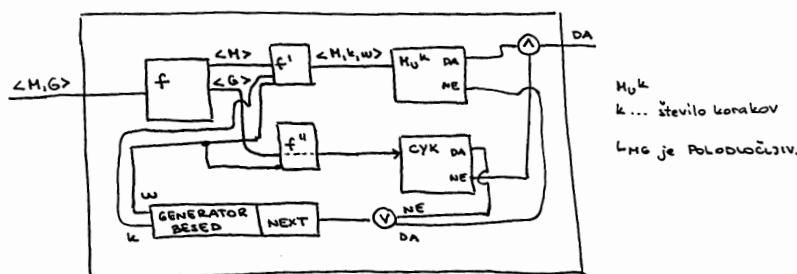
Iz vhoda $\langle M, w \rangle$ in w sestavimo ...



Kakšne vrste je jezik, če stroj TM M sprejme vsaj en niz, ki ga neodvisno-kontekstna gramatika ne?

$$L_{NG} = \{\langle M, G \rangle \mid L(M) \setminus L(G) \neq \emptyset\}$$

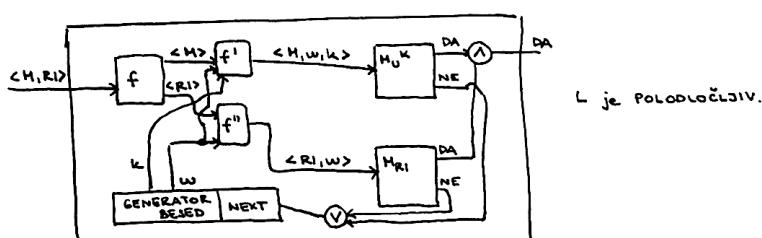
$$L_{NG} = \{\langle M, G \rangle \mid L(M) \setminus L(G) \neq \emptyset\}$$



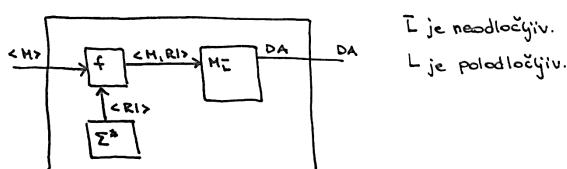
Kakšne vrste je jezik, če je presek M z RI ni prazna množica?

Kaj pa njegov komplement?

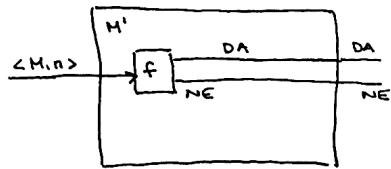
$$L = \{\langle M, RI \rangle \mid L(M) \cap L(RI) \neq \emptyset\}$$



$$\bar{L} = \{\langle M, RI \rangle \mid L(M) \cap L(RI) = \emptyset\}$$



Kakšne vrste je jezik $L = \{\langle M, n \rangle, |M| > n\}$, M je opis TM-ja, $n \geq 1$.



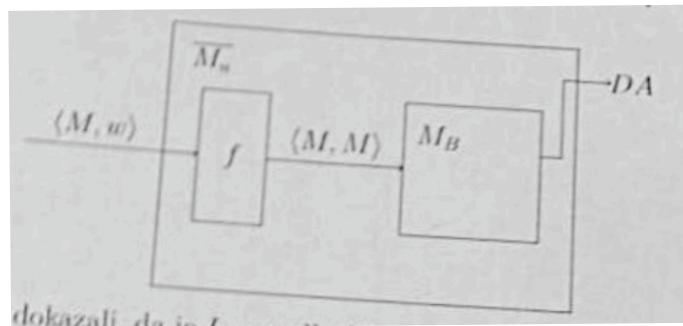
```
function f(<M, n>) {
    m = M.length();
    if (m > n)
        return true;
    else
        return false;
}
```

Jezik L je odločljiv (lahko napišemo program).

Naj bosta podana jezika L_a in L_b . Vemo zgolj to, da je jezik L_a polodločljiv.

- a) Uspemo narediti pretvorbo $L_a \rightarrow L_b$. Kaj vemo o odločljivosti problema L_b ?
- b) Uspemo narediti pretvorbo $L_b \rightarrow L_a$. Kaj vemo o odločljivosti problema L_b ?
- c) Denimo, da je jezik $L_b = \{\langle M_1, M_2 \rangle; |L(M_1) \geq |L(M_2)|\}$.

Trdimo, da smo dokazali, da je L_b neodločljiv problem. Ali imamo prav?



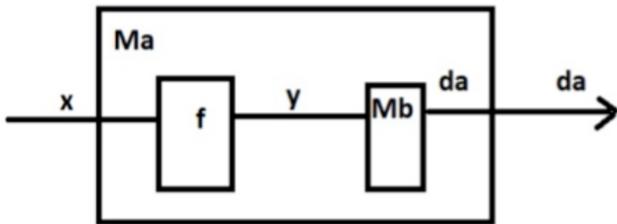
L_a je lahko tudi odločljiv.

- a) Karkoli.
- b) Vsaj polodločljiv (polodločljiv, odločljiv).

By Čibej:

Če vemo zgolj to, da je jezik L_a polodločljiv, v bistvu je mogoče tudi odločljiv. Če naredimo prevedbo $L_a \rightarrow L_b$ je lahko L_b karkoli. Če naredimo prevedbo $L_b \rightarrow L_a$, pa je jezik L_b vsaj polodločljiv.

Prikazan je poskus prevedbe.



Kaj mora veljati za x in y , da bo slika predstavljal veljavno priredbo?

x pripada L_a iff y pripada L_b .

L_b je neodločljiv jezik. Kaj nam to pove o L_a ?

L_a je kvečjemu tako težek kot L_b (karkoli).

L_a je polodločljiv jezik. Kaj nam to pove o L_b ?

L_b je polodločljiv ali nepolodločjivo neodločljiv.

Zapiši program M' , da bo veljalo:

```
if (|L(M')| = 17)
    x je element L(M')
else if (|L(M')| = 42)
    x ni element L(M')
```

```
def M'(x):
    if x pripada L_a
        return true
    else if Ma(M, w)
        return x pripada L_b
```

A, B množici besed

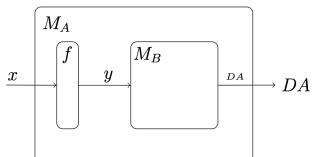
$|A| = 17, |B| = 25, A \cap B = \emptyset$

$A \rightarrow B$

L_a je polodločljiv jezik. Kaj nam to pove o L_b ?

Polodločljiv ali nepolodločjivo neodločljiv.

Kaj mora veljati za funkcijo f na spodnji sliki, da je prevedba $M_A \rightarrow M_B$ veljavna?



Funkcija f mora biti izračunljiva.

Podan je jezik: $L = \{< M > \mid L(M) \cap \text{PRIMES} \neq \emptyset\}$ kjer je PRIMES množica vseh besed, katerih dolžina je praštevilo.

- a) Zapišite tri Turingove stroje (shematsko), ki temu jeziku pripadajo, in dva, ki temu jeziku ne pripadata.
- b) Kakšen jezik je to - odločljiv, neodločljiv, polodločljiv?

Zapišite tri Turingove stroje (shematsko), ki temu jeziku pripadajo, in dva, ki temu jeziku ne pripadata.

$\text{PRIMES} = \{w \mid |w| \text{ je praštevilo}\}$

Jezik vsebuje vse TM-je oz. njihove opise, katerih jezik vsebuje vsaj eno praštevilo.

Primeri strojev, ki pripadajo jeziku L :

- stroj, ki sprejme samo 111 (njegov jezik je $\{111\}$)

$$\delta(q_0, 1) \rightarrow \delta(q_1, 1, R)$$

$$\delta(q_1, 1) \rightarrow \delta(q_2, 1, R)$$

$$\delta(q_2, 1) \rightarrow \delta(q_F, 1, R)$$

- stroj, ki sprejme samo 11 (njegov jezik je $\{11\}$)

$$\delta(q_0, 1) \rightarrow \delta(q_1, 1, R)$$

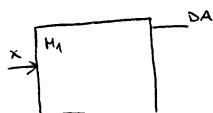
$$\delta(q_1, 1) \rightarrow \delta(q_F, 1, R)$$

- stroj, ki sprejme samo 1

$$\delta(q_0, 1) \rightarrow \delta(q_F, 1, R)$$

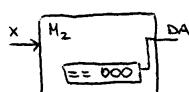
- vse kar dobi, reče DA

$$L(M_1) = \Sigma^*$$



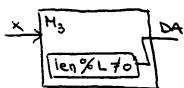
- sprejme samo eno besedo: preveri, ali je vhod enak 000 (če je, reče DA)

$$L(M_2) = \{000\}$$



- stroj, ki kot vhod sprejme x in pogleda, kakšna je dolžina x-sa

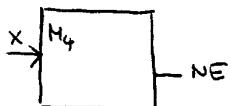
$$L(M_3) = \{\text{besede lihe dolžine; } 0, 1, 000, 001, \dots\}$$



Primeri strojev, ki ne pripadajo jeziku L:

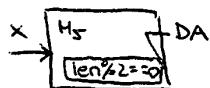
- stroj, ki vsak vhod zavrne

$$L(M_4) = \emptyset$$



- stroj, ki preveri, ali je dolžina vhodnega niza soda

$$L(M_5) = \{00, 10, 0000, \dots\}$$



- psevdokoda

```

bool M5 (x) {
    return length(x) % 2 == 0;
}
  
```

Kakšen jezik je - odločljiv, neodločljiv, polodločljiv?

L_{PRIMES} ... jezik PRIMES

Kaj pomeni, da je je presek neprazen? Obstaja neka beseda, ki je v obeh jezikih v $L(M)$ in PRIMES.

PRIMES je odločljiva množica (lahko napišemo program, npr. v Javi).

Splošno

- polodločljiv

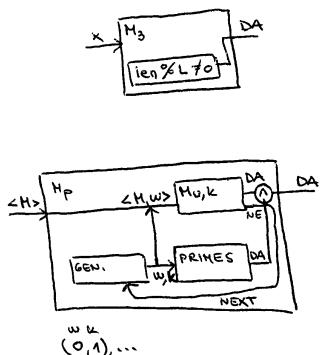
Moramo najti vsaj en niz, ki pripada jeziku.

- neodločljiv

Presek mora biti prazen. Neodločljiv, ker ne moremo garantirati, da ne obstaja v neskončnosti nek niz, za katerega bi rekli DA.

Za vsa praštevila simuliramo, če pripadajo tudi jeziku $L(M)$. Če najdemo kako število, ki pripada, potem se ustavimo in vrnemo DA.

Kot vhod v M_p dobimo stroj M . Stroj M bomo poskušali spraviti čez univerzalni TM. Obenem bomo uporabljali stroj za praštevila oz. za prepoznavanje PRIMES.



Ali je polodločljiv?

Generiramo vse možne nize in jih sistematično dajamo na izhod (w). W damo na vhod PRIMES, ki bo preveril ali je w praštevilske dolžine. UTM dobi kot vhod program iz vhoda in besedo w iz generatorja. Če PRIMES in UTM vrneta DA, potem stroj M_p vrne DA.

Vendar to ne deluje.

Problem: UTM se lahko pri neki besedi zacikla; ne bomo dobili odgovora DA, čeprav ga bi morali.

UTM dopolnimo še z enim k-jem.

Zdaj ne bo več simuliral programa M na besedi w , ampak ga bo simuliral do k-korakov. Posledično, UTM se bo vedno ustavil.

Generator ne generira samo besede ampak generira še neko število korakov k, ki ga damo na vhod UTM.

To je stroj, ki se vedno ustavi, če je odgovor DA.

Posledično je L_p **vsaj polodločljiv**.

Ali je (ne) odločljiv?

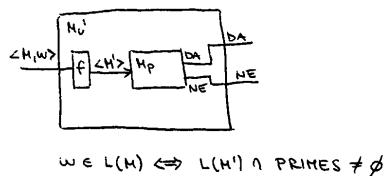
Ali obstaja TM, ki bi sprejel opis stroja M in bi rekel, da ta stroj M sprejme vsaj eno praštevilo ali stroj M ne sprejme nobenega praštevila? Za vsako vprašanje bi odgovor DA ali NE v končnem času.

Prevedba univerzalnega TM na naš problem L_p .

$$L_u \rightarrow L_p$$

Napisali bomo algoritem, ki ne obstaja (nikoli se ne bo zacikljal, vedno bo vrnil DA ali NE).

Znotraj M_u' bomo uporabili hipotetični L_p .



Glede na to, da smo zvezali DA in NE istočasno, mora hkrati veljati

$$w \in L(M) \iff L(M') \cap \text{PRIMES} \neq \emptyset$$

$$w \notin L(M) \iff L(M') \cap \text{PRIMES} = \emptyset$$

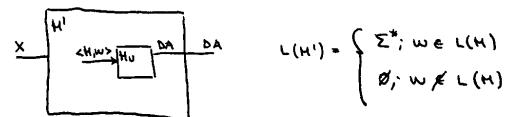
Če bi bil odločljiv, potem bi bil odločljiv tudi UTM.

$$L_p \text{ odločljiv} \Rightarrow L_u \text{ odločljiv}$$

$$L_u \text{ ni odločljiv} \Rightarrow L_p \text{ ni odločljiv}$$

To je dokaz, da je M_p največ polodločljiv (ni čisto odločljiv).

L_p ni odločljiv.



Drugi način prevedbe $L_u \rightarrow L_p$

```
import M_p, M'_u
bool (Mu (M, w)) {
    M' = "bool M' (x) {
        return M'_u (M, w)
    }"
    return M_p (M')
}
```

Ugotovi, ali je jezik odločljiv (R), polodločljivo neodločljiv (RE), nepolodločljiv (not RE).

- $L = \{\langle M \rangle \mid M \text{ je TM in obstaja vhod nad katerim se } M \text{ ustavi v manj kot } |\langle M \rangle| \text{ korakih}\}$.

Odločljiv.

(Število korakov je končno, M se bo zagotovo ustavil in sprejel).

- $L = \{\langle M \rangle \mid M \text{ je TM in } |L(M)| \leq 3\}$.

Nepolodločljiv (dokaz z $\neg L_{halt}$).

- $L = \{\langle M \rangle \mid M \text{ je TM in } |L(M)| \geq 3\}$.

Polodločljivo neodločljiv, neodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in sprejema vsa soda števila}\}$.

Nepolodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in } L(M) \text{ je končen}\}$.

Nepolodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in } L(M) \text{ je nekončen}\}$.

Nepolodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in } L(M) \text{ je števen}\}$.

Odločljiv. To so jeziki vseh TM-jev.

- $L = \{\langle M \rangle \mid M \text{ je TM in } L(M) \text{ nešteven}\}$.

Odločljiv. Prazna množica, saj taki jeziki ne obstajajo.

- $L = \{\langle A, B \rangle \mid A \text{ in } B \text{ sta TM-ja in } \epsilon \in L(A) \cup L(B)\}$.

Polodločljivo neodločljiv (A in B sta polodločljiva), neodločljiv.

- $L = \{\langle A, B \rangle \mid A \text{ in } B \text{ sta TM-ja in } \epsilon \in L(A) \cap L(B)\}$.

Polodločljivo neodločljiv (A in B sta polodločljiva), neodločljiv.

- $L = \{\langle A, B \rangle \mid A \text{ in } B \text{ sta TM-ja in } \epsilon \in L(A) \setminus L(B)\}$.

Nepolodločljiv.

- $L = \{\langle M, x \rangle \mid M \text{ je TM, } x \text{ je niz, obstaja TM } M', x \notin (M \cap L(M'))\}$.

Polodločljivo neodločljiv (M' zavrže vse vhode $\Rightarrow M$ je jezik vseh TM-jev).

- $L = \{\langle M \rangle \mid M \text{ je TM, obstaja vhod na katerem se } M \text{ ustavi v največ 1000 korakih}\}$.

Odločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM, obstaja vhod čigar dolžina je manjša od 100 in na katerem se } M \text{ ustavi}\}$.

Nepolodločljiv, neodločljiv.

- $L = \{\langle M \rangle \mid |M| < 1000\}$

Odločljiv (Vsi opisi Turingovih strojev nad fiksno abecedo končne dolžine).

- $L = \{\langle M \rangle \mid M \text{ je TM in se ustavi na palindromih}\}$

Nepolodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in } L(M) \cap \{a^{2^n} \mid n \geq 0\} \text{ je prazen}\}$

Nepolodločljiv.

- $L = \{\langle M, k \rangle \mid M \text{ je TM in } |\{w \in L(M) : w \in a^*b^*\}| \geq k\}$

Polodločljivo neodločljiv, neodločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in se ustavi na (najmanj) dveh nizih različne dolžine}\}$

Polodločljivo neodločljiv, neodločljiv.

- $L = \{\langle M, x, k \rangle \mid M \text{ je TM in se ne ustavi na } x \text{ within } k \text{ korakov}\}$

Odločljiv.

- $L = \{\langle M \rangle \mid M \text{ je TM in } |L(M)| \text{ je praštevilo}\}.$

Ni polodločljiv.

- $L = \{\langle M \rangle \mid \text{obstaja } x \in \Sigma^*, \text{ tako da za vsak } y \in L(M) \text{ velja, da } xy \notin L\}$

Ni polodločljiv.

- $L = \{\langle M \rangle \mid \text{obstaja } x, y \in \Sigma^*, \text{ tako da } x \in L(M) \text{ ali } y \notin L(M)\}$

Odločljiv.

- $L = \{\langle M \rangle \mid \text{obstaja TM } M' \text{ da velja } \langle M \rangle \neq \langle M' \rangle \text{ in } L(M) = L(M')\}$

Odločljiv.

- $L = \{\langle M \rangle \mid M \text{ ne sprejme noben niz } w001\}$

Nepolodločljiv.

- $L = \{\langle M, x \rangle \mid x \text{ je prefix } \langle M \rangle\}$

Odločljiv (preveri ali je niz x prefix niza $\langle M \rangle$).

- $L = \{\langle A, B, C \rangle \mid L(A) = L(B) \cup L(C)\}$

Nepolodločljiv.

- $L = \{\langle A, B, C \rangle \mid L(A) \subseteq L(B) \cup L(C)\}$

Nepolodločljiv.

- $L = \{\langle A \rangle \mid \text{obstaja TM } B \text{ in } C, \text{ da velja } L(A) \subseteq L(B) \cup L(C)\}$

Odločljiv.

Izpitne naloge

Naloga 1

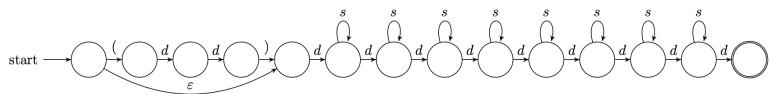
Napišite regularni izraz za prepoznavanje telefonskih številk. Vsaka telefonska številka je sestavljena iz 9 števk, ki so lahko poljubno ločene s presledki. Na začetku je lahko prisotna še območna koda, sestavljena iz dveh števk v oklepaju.

$$d = 0 + 1 + 2 + \dots + 9$$

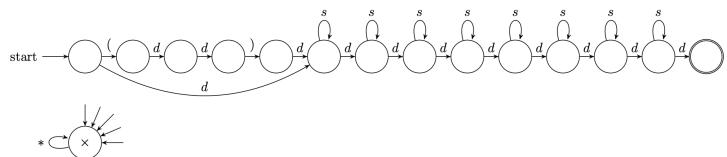
$$s = \dots$$

$$(("dd")^* + \varepsilon)ds^*ds^*ds^*ds^*ds^*ds^*ds^*ds^*d$$

Dobljeni regularni izraz pretvorite v ε -NKA.



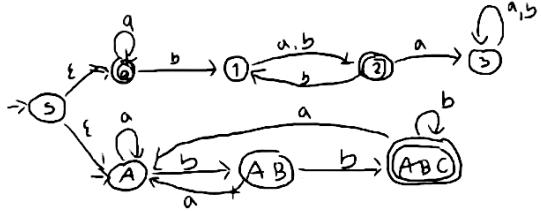
ε -NKA pretvorite v DKA.



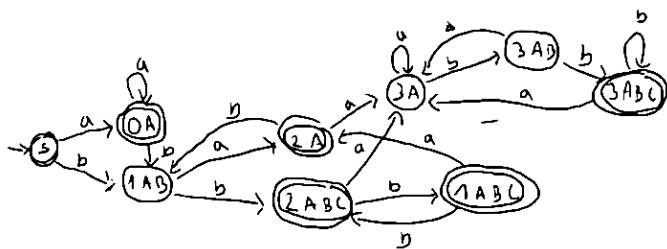
Naloga 2

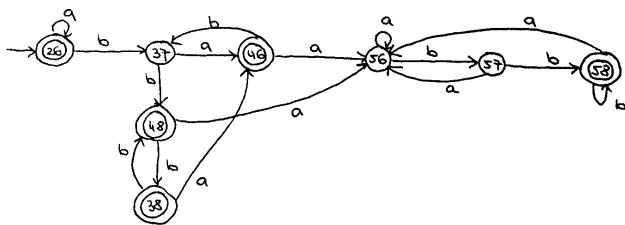
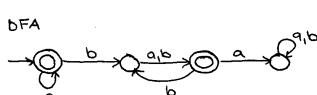
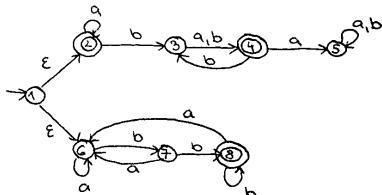
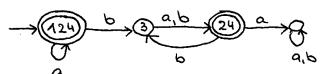
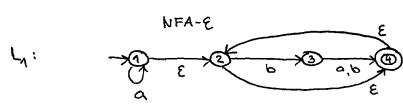
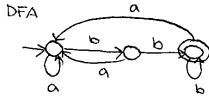
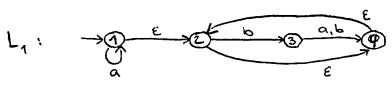
Naj bo L_1 podan z regularnim izrazom $a^*(ba + bb)^*$, L_2 pa z regularnim izrazom $(a + b)^*bb$.

- Zapiši DKA za L_1 in L_2 .
- Zapiši DKA za $L_1 \cup L_2$.



Znebimo se ϵ , da dobimo NKA - pogledamo, kam lahko pridemo iz S z navadnimi simboli preko ϵ . Pretvorimo v DKA.





NKA => DKA

V vsakem stanju imamo pri DKA dve možnosti (a in b). Pri vsakem stanju je potrebno upoštevati vse možne prehode.

Začnemo v stanju 1.

Gledamo tudi ε -prehode.

$\delta(1, \varepsilon) : 1, 2, 4 \Rightarrow$ začetno stanje in tudi končno stanje (vsebuje vsaj eno stanje, ki je končno)

$\delta(1, a): 1 \Rightarrow 1, 2, 4$

$\delta(2, a)$: /

$\delta(4, a)$: /

$$\delta(\{1, 2, 4\}, b) : 3$$

$\delta(3, a): 2, 4 \Rightarrow$ končno stanje (4)

$\delta(3, b)$: 2,4

$\delta(\{2, 4\}, a)$: dead state

$$\delta(\{2, 4\}, b): 3$$

Ko narišemo avtomat, preverimo če smo prav narisali.

$$L_1 \cup L_2$$

Vsa stanja v katera lahko pri ϵ -prehodih pridemo iz začetnega stanja prvotnega avtomata so začetna stanja.

Začetno stanje: 1, 2, 6

$\delta(\{1, 2, 6\}, a) : 2, 6 \Rightarrow 1$ lahko črtamo, ker je umetno začetno stanje \Rightarrow končno stanje

$\delta(\{2, 6\}, b) : 3, 7$

$\delta(\{3, 7\}, a) : 4, 6 \Rightarrow$ končno stanje

$\delta(\{3, 7\}, b) : 4, 8 \Rightarrow$ končno stanje

$\delta(\{4, 6\}, a) : 5, 6$

$\delta(\{4, 6\}, b) : 3, 7$

$\delta(\{5, 6\}, a) : 5, 6$

$\delta(\{5, 6\}, b) : 5, 7$

$\delta(\{5, 7\}, a) : 5, 6$

$\delta(\{5, 7\}, b) : 5, 8$

$\delta(\{5, 8\}, a) : 5, 6$

$\delta(\{5, 8\}, b) : 5, 8$

$\delta(\{4, 8\}, a) : 5, 6$

$\delta(\{4, 8\}, b) : 3, 8$

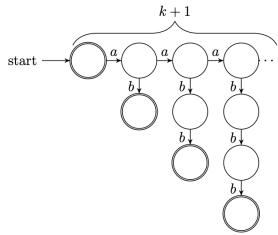
$\delta(\{3, 8\}, a) : 4, 6$

$\delta(\{3, 8\}, b) : 4, 8$

Naloga 3

Dokažite, da je jezik $\{a^n b^n \mid n \leq k\}$ regularen za poljuben k .

Sestavimo NKA za ta jezik:



Je jezik $\{a^n b^{2n} c^n \mid n \geq 0\}$

i. regularen?

Ne. Uporabimo lemo o napihovanju za regularne jezike. Izberemo prizerno besedo $w = a^n b^{2n} c^n$. Glede na lemo se mora nekje med prvimi n znaki pojaviti cikel, ki ga lahko napihujemo in tako dobimo besedo $w' = a^{n+j(\alpha-1)} b^{2n} c^n$, kjer je $j > 0$ dolžina cikla in $\alpha \geq 0$ število prehodov cikla. Za poljuben $\alpha \neq 1$ beseda w' ni v jeziku, zato jezik ni regularen.

ii. kontekstno neodvisen?

Ne. Uporabimo lemo o napihovanju za kontekstno neodvisne jezike. Izberemo prizerno besedo $w = a^n b^{2n} c^n$. Del, ki ga napihujemo, se lahko nahaja v celoti med a-ji, b-ji ali c-ji, ali pa se razteza prek ene izmed mej, med a-ji in b-ji ali med b-ji in c-ji. V vsakem izmed teh primerov po napihovanju beseda ni več v jeziku.

iii. odločljiv?

Da, obstaja Turingov stroj, ki razpoznavajo jezik. Ideja: Uporabimo stroj s tremi trakovi.

Najprej preverimo, če je beseda sestavljena iz zaporedja a-jev, b-jev in c-jev v tem vrstnem redu. Nato enake simbole razdelimo na enake trakove. Zdaj lahko preverimo, ali je število a-jev enako številu b-jev, potem pa še, ali je število b-jev dvakratnik števila a-jev.

iv. rekurziven?

Da, rekurziven in odločljiv sta sinonima.

Dokažite, da je $\{a^i b^j a^{i+j} \mid i, j > 0\}$ determinističen kontekstno neodvisen jezik.

Zgradimo skladovni avtomat, ki sprejema s praznim skladom:

$$\delta(q_0, a, Z_0) = (q_0, XZ_0)$$

$$\delta(q_0, a, X) = (q_0, XX)$$

$$\delta(q_0, b, X) = (q_1, XX)$$

$$\delta(q_1, b, X) = (q_1, XX)$$

$$\delta(q_1, a, X) = (q_2, \epsilon)$$

$$\delta(q_2, a, X) = (q_2, \epsilon)$$

Naloga 4

Je mogoče, da je jezik L polodločljiv, njegov komplement L pa neodločljiv? Če je, podajte primer.

Je mogoče, na primer univerzalni jezik L_u in njegov komplement $L_{\bar{u}}$.

Ali je jezik $\{\langle M \rangle \mid L(M) = \Sigma^*\}$

i. regularen?

Ne, ker je neodločljiv.

ii. odločljiv?

Ne, ker je neodločljiv.

iii. polodločljiv?

Ne, ker je neodločljiv.

iv. neodločljiv?

Da.

Naloga 5

Naj bo L jezik binarnih nizov, ki imajo $3p$ ničel in $5q$ enic za neka $p, q \geq 0$.

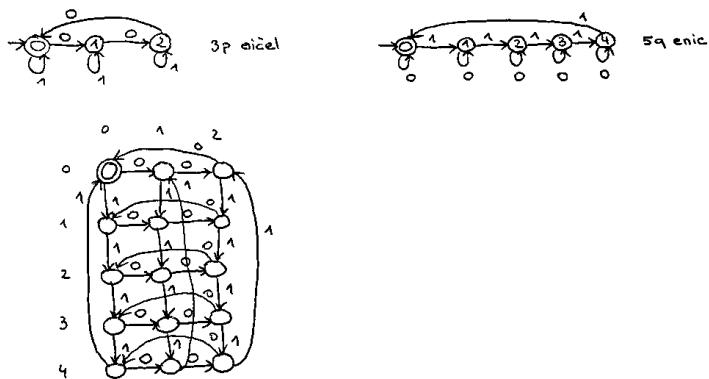
- Zapiši KNG G , da bo $L(G) = L$.
- Zapiši SA P , da bo $L(P) = L$.
- Ali je L regularen? Zakaj?

1. rešitev

- Narišemo lahko končni avtomat in rečemo, da je to skladovni avtomat.

Naredimo končen avtomat za $3p$ ničel in $5q$ enic (deljivost s 3 in 5).

Za presek naredimo produktni avtomat ($5 * 3$ stanj).



- Iz končnega avtomata lahko zelo enostavno sestavimo gramatiko.

Vsako stanje predstavlja eno spremenljivko gramatike.

$A_{00}, A_{01}, \dots A_{02}$

....

A_{42}

$S \rightarrow A_{00}$

Nato za vsako stanje naredimo produkcije.

$A_{00} \rightarrow 0A_{01} // \text{po 0 gre v stanje } A_{01}$

$A_{00} \rightarrow 1A_{10}$

$A_{00} \rightarrow \epsilon$

...

Ali pa napišemo, kar pravilo.

$A_{ij} \rightarrow 0A_{i(j+1)\%3} \mid 1A_{(i+1)\%5j} \mid \epsilon$

- DA, ker smo narisali končni avtomat pri a).

2. rešitev

a) Zapiši KNG G, da bo $L(G) = L$.

$$S \rightarrow A_{00}$$

Začetni simbol bo A_{00} (nobena ničla in nobena enica ne manjka).

A_{ij} : manjka še $(3 - i)$ ničel in $(5 - j)$ enic

$A_{00} \rightarrow \epsilon \mid 0A_{10} \mid 1A_{01} \mid \dots$ v jeziku je prazen niz, eno ničlo že imamo (2 manjkata), eno enico že imamo (4 manjkajo)

$$A_{10} \rightarrow 0A_{20} \mid 1A_{11}$$

$$A_{20} \rightarrow 0A_{00} \mid 1A_{21}$$

$$A_{01} \rightarrow 0A_{11} \mid 1A_{02}$$

$$A_{11} \rightarrow 0A_{21} \mid 1A_{12}$$

$$A_{21} \rightarrow 0A_{01} \mid 1A_{22}$$

...

$$A_{04} \rightarrow 0A_{14} \mid 1A_{00}$$

$$A_{14} \rightarrow 0A_{24} \mid 1A_{10}$$

Splošneje

$$G = (N, T, P, S), T = \{0, 1\}$$

$$N = \{A_{ij} \mid 0 \leq i \leq 2 \text{ in } 0 \leq j \leq 4\}$$

P:

$$S \rightarrow A_{00}$$

$$A_{00} \rightarrow \epsilon$$

$$A_{ij} \rightarrow 0A_{(i+1)\bmod 3, j} \mid 1A_{i, (j+1)\bmod 5}$$

b) Zapiši SA P, da bo $L(P) = L$.

Za pretvorbo gramatike v avtomat obstaja sistematičen postopek.

Naredili bomo avtomat s praznitvijo sklada.

$$A_{10} \rightarrow 0A_{20} \mid 1A_{11}$$

$\delta(q, \epsilon, S) \rightarrow \{(q, A_{00})\}$ // Na začetku ima na skladu simbol S. Nato S pobrišemo in ga nadomestimo z A_{00} .

$\delta(q, \epsilon, A_{00}) \rightarrow \{(q, \epsilon)\}$ // Če vidimo ϵ in imamo na skladu A_{00} , potem se sklad sprazne.

$\delta(q, 0, A_{ij}) \rightarrow \{(q, A_{(i+1)\bmod 3, j})\}$ // Če vidimo 0 in imamo na skladu A_{00} , potem se sklad sprazne.

$\delta(q, 1, A_{ij}) \rightarrow \{(q, A_{i, (j+1)\bmod 5})\}$

Na skladu bo vedno en simbol - pove nam, kateri je trenutno nekončni simbol.

c) Ali je L regularen? Zakaj?

Gramatike, ki je takšne oblike definira regularni jezik.

$A \rightarrow \epsilon$

$A \rightarrow a$

$A \rightarrow aB$

