Teoretične osnove računalništva

Zapiski predavanj 2010/2011

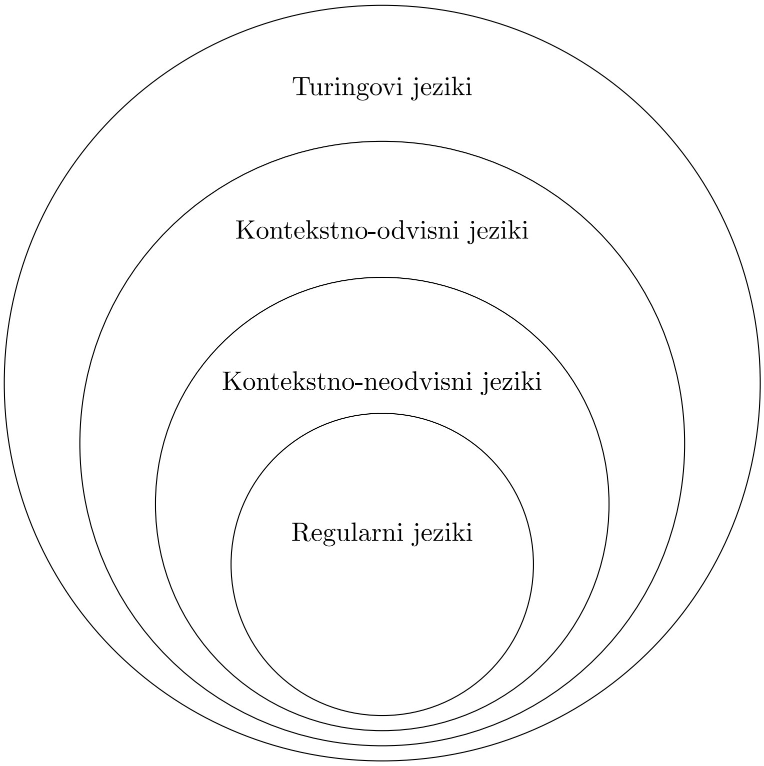
27. februar 2011

Kazalo

Poglavje 1

Uvod

1.1 Chomskyeva hiearhija



1.2 Matematične osnove

1.2.1 Dokazovanje

Dokaz s konstrukcijo

Dokaz obstoja nekega matematičnega objekta je to, da nam objekt uspe skonstruirati.

Primeri:

Primer 1: Za vsak $n > 4$, obstaja dvojiško drevo, ki ima natanko 3 liste.
Primer 2: $|\mathbb{R}| = |[0, 1)|$.

- Množici imata enako moč, kadar med njima obstaja bijektivna preslikava.

- Vsako realno število r lahko zapišemo kot:

$$r = \pm d_1 d_2 \ldots d_n . \bar{d_1} \bar{d_2} \ldots \bar{d_m} \ldots ;\ d_1 \neq 0$$

- Definiramo preslikavo:

$$\mathbb{R} \to [0,1) : r \to 0.s\bar{d_1}d_n\bar{d_2}d_{n-1}...\bar{d_n}d_1\bar{d_{n+1}}0\bar{d_{n+2}}0...$$

  kjer z s določimo predznak ($s = 0$, če $r \geq 0$ in $s = 1$, sicer).
- Vidimo:
  - $|\mathbb{R}| \leq |[0,1)|$,
  - $|\mathbb{R}| \geq |[0,1)|$, ker velja $[0,1) \subset \mathbb{R}$
- Iz tega lahko sklepamo, da velja $|\mathbb{R}| = |[0,1)|$

Dokaz z indukcijo

Če je množica induktivni razred[1], lahko z matematično indukcijo dokazujemo neko lastnost članov množice.

Induktivni razred I sestavlja:

- Baza indukcije - najbolj osnovna množica elementov (osnovni razred)
- Pravila generiranja - kako iz elementov baze gradimo nove elemente (množico)

Primeri:

Primer 1: Induktivni razred naravnih števil ($\mathbb{N}$)

- Baza: $1 \in \mathbb{N}$
- Pravila generiranja: $n \in \mathbb{N} \Longrightarrow n + 1 \in \mathbb{N}$

Primer 2: Hilbertove krivulje[2]

Dokaz s protislovjem

Vzamemo nasprotno trditev, od tiste, ki jo želimo preveriti in pokažemo, da to vodi v protislovje.

Primeri:

Primer 1: Praštevil je končno mnogo.

- Predpostavimo, da poznamo vsa praštevila:
  $P = \{2, 3, 5, ..., p\}$, kjer je p zadnje praštevilo
- Po definiciji obstajajo le praštevila in sestavljena števila (to so taka, ki jih lahko razstavimo na prafaktorje).
- Če pomnožimo vsa znana praštevila iz P in prištejemo 1 dobimo število, ki se ga ne da razstaviti na prafaktorje iz množice P:
  $q = 2 * 3 * 5 * ... * p + 1$
- Torej je q ali praštevilo (ker ni sestavljeno), ali pa število, sestavljeno iz prafaktorjev, ki jih ni v množici P.
- Oboje kaže na to, da v množici P nimamo vseh praštevil, ter, da to velja za vsako končno množico praštevil.

Primer 2: $\sqrt[3]{2}$ je racionalno število.

- Če je $\sqrt[3]{2}$ racionalno število, ga je moč zapisati kot ulomek $\frac{a}{b}$.

[1] Glej slovarček na koncu.
[2] http://en.wikipedia.org/wiki/Hilbert_curve

- Predpostavimo, da je ulomek $\frac{a}{b}$ okrajšan (torej, da velja: $GCD(a, b) = 1$):

$$\begin{aligned}\sqrt[3]{2} &= \frac{a}{b}\\ 2 &= \left(\frac{a}{b}\right)^3\\ 2b^3 &= a^3\end{aligned}$$

- Opazimo, da je a sodo število, torej lahko pišemo $a = 2k$:

$$\begin{aligned}2b &= (2k)^3\\ 2b &= 8k\\ b &= 4k\end{aligned}$$

- Ker se je pokazalo, da je tudi b sodo število, $GCD(a, b) = 1$ ne more držati, torej smo prišli v protislovje in s tem dokazali, da $\sqrt[3]{2}$ ni racionalno število.

Poglavje 2

Regularni jeziki

2.1 Uvod

Oznake

- a - simbol (niz dolžine 1)
- Σ - abeceda (končna neprazna množica simbolov)
- w - niz ali beseda (poljubno končno zaporedje simbolov $w_1 w_2 \ldots w_n$)
- $|w|$ - dolžina niza
- ε - prazen niz, $|w| = 0$
- Σ^* - vsi možni nizi abecede

Operacije

- Stik
  - Stik nizov:

$$\begin{aligned} w &= w_1 w_2 \ldots w_n \\ x &= x_1 x_2 \ldots x_m \\ wx &= w_1 w_2 \ldots w_n x_1 x_2 \ldots x_m \end{aligned}$$

  - Stik množic:

$$\begin{aligned} A &= \{w_1,\ w_2,\ \ldots,\ w_n\} \\ B &= \{x_1,\ x_2,\ \ldots,\ x_m\} \\ A \cdot B &= \{w_i x_j \mid w_i \in A \ \wedge\ x_i \in B\} \end{aligned}$$

- Potenciranje

$$\begin{aligned} &A^0 = \{\varepsilon\} \\ &A^k = A \cdot A \cdot \ldots \cdot A \ = \ \bigcirc_{i=1}^{k} A \end{aligned}$$

- Iteracija

$$A^* = A^0 \cup A^1 \cup A^2 \cdots \ = \ \bigcup_{i=0}^{\infty} A^i$$

Regularni jezik

Def.: Regularni jezik L nad abecedo Σ je poljubna podmnožica Σ^*

$$L \subseteq \Sigma^*$$

Primeri:

Primer 1: Prazen jezik: $L_1 = \{\}$
Primer 2: Jezik, ki vsebuje ε (ni prazen): $L_2 = \{\varepsilon\}$
Primer 3: Jezik, ki vsebuje nize "a, aa, ab": $L_3 = \{a, aa, ab\}$

2.2 Regularni Izrazi

Def.: Osnovni izrazi:

- $\underline{\emptyset}$ je opisuje prazen jezik $L(\underline{\emptyset}) = \{\}$
- $\underline{\varepsilon}$ opisuje jezik $L(\underline{\varepsilon}) = \{\varepsilon\}$
- $\underline{a}$ opisuje jezik $L(\underline{a}) = \{a\},\ a \in \Sigma$

Def.: Pravila za generiranje sestavljenih izrazov:

- $(r_1 + r_2)$ opisuje unijo jezikov $L(r_1 + r_2) = L(r_1) \bigcup L(r_2)$
- $(r_1\ r_2)$ opisuje stik jezikov $L(r_1\ r_2) = L(r_1) \cdot L(r_2)$
- (r^*) opisuje iteracijo jezika $(L(r))^*$

Primeri:

Primer 1: Opiši vse nize, ki se končajo z nizom 00 v abecedi $\Sigma = \{0, 1\}$.

$$r = (0 + 1)^* 00$$

Primer 2: Opiši vse nize, pri katerih so vsi a-ji pred b-ji in vsi b-ji pred c-ji v abecedi $\Sigma = \{a, b, c\}$.

$$a^* b^* c^*$$

Primer 3: Opiši vse nize, ki vsebujejo vsaj dva niza 'aa', ki se ne prekrivata v abecedi $\Sigma = \{a, b, c\}$.

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

Primer 4: Opiši vse nize, ki vsebuje vsaj dva niza 'aa' ki se lahko prekrivata v abecedi $\Sigma = \{a, b, c\}$

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

Primer 5: Opiši vse nize, ki ne vsebujejo niza 11 v abecedi $\Sigma = \{0, 1\}$

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

Primer 6: S slovensko abecedo opiši besedo "Ljubljana"v vseh sklonih in vseh mešanicah velikih in malih črk.

$$(L+l)(J+j)(U+u)(B+b)(L+l)(J+j)(A+a)(N+n)\,((A+a)(O+o)(E+e)(I+i))$$

Koliko različnih nizov opišemo s tem regularnim izrazom?

$2^8 \cdot 2^3 = 2^{11}$ nizov

2.2.1 Jezik regularnih izrazov

Def.: Jezik ki ga opisuje poljubni regularni izraz, je regularni jezik.

Primeri:

Primer 1: $\{\}$ je regularni jezik
Primer 2: $\{0^n 1^n \mid n \geqslant 0\}$ ni regularni jezik
note: regularni jeziki ne znajo držati neskončno - poljubno mnogo - informacij o prejšnjih znakih. Kasneje bomo za dokazovanje uporabljali lemo o napihovanju regularnih jezikov.

2.3 Končni avtomati

2.3.1 Nedeterministični končni avtomati z ε-prehodi

Def.: εNKA je definiran kot peterka $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, kjer je:

- Q - končna množica stanj
- Σ - vhodna abeceda
- δ - funkcija prehodov, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$
- q_0 - začetno stanje
- F - množica končnih stanj

razlaga: $P(Q) = 2^Q$ je potenčna množica stanj avtomata. To pomeni da je so v $P(Q)$ vse možne kombinacije stanj.
Recimo da se nahajamo v stanju A, potem nas funkcija prehodov δ pripelje v vsa mozna stanja do katerih pridemo iz A z določenim simbolom abecede in z vsemi ε prehodi, naprimer $\{A_1, A2, \ldots, A_n\}$. Tukaj je množica stanj $\{A_1, A2, \ldots, A_n\}$ elemnt potenčne množice $P(Q)$

2.3.2 Nedeterministični končni avtomati

Def.: NKA je definiran kot peterka $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, kjer je:

- Q - končna množica stanj
- Σ - vhodna abeceda
- δ - funkcija prehodov $\delta : Q \times \Sigma \to 2^Q$
- q_0 - začetno stanje
- F - množica končnih stanj

2.3.3 Deterministični končni avtomat

Def.: DKA je definiran kot petorka $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, kjer je:

- Q - končna množica stanj
- Σ - vhodna abeceda
- δ - funkcija prehodov, $\delta : Q \times \Sigma \to Q$
- q_0 - začetno stanje
- F - množica končnih stanj

2.3.4 Jeziki končnih avtomatov

Def.: Jezik εNKA ter NKA je definiran kot:

$$L = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

kjer je $\hat{\delta}(q, w)$ posplošena funkcija prehodov v večih korakih.

Def.: Jezik DKA je definiran kot:

$$L = \{w \mid \delta(q_0, w) \in F\}$$

Definicije želijo povedati, da so v jeziku točno tisti nizi, po katerih je iz začetnega stanja mogoče priti do nekega končnega stanja.

2.3.5 Regularne gramatike

Def.: Regularna gramatika je definirana kot četvorček $G = \langle V, T, P, S\rangle$, kjer je:

- V - množica spremenljivk oz. vmesnih simbolov, $V \subseteq \Sigma$
- T - množica znakov oz. končnih simbolov, $T \subset \Sigma$
- P - množica produkcij, $[\alpha_1 \to \alpha_2]$
- S - začetni simbol, $S \in V$

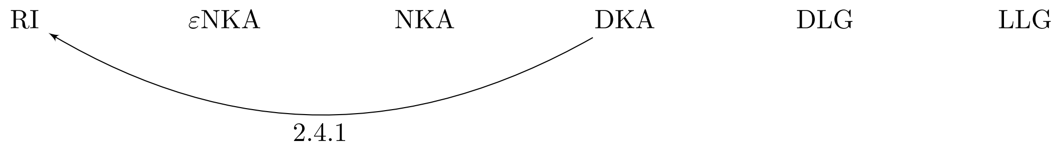
Pri tem pa regularne gramatike ločimo na levo in desno-regularne.

- Pri levih so produkcije $P \subset V \times ((V \cup \{\varepsilon\}) \cdot T^*)$
- Pri desnih so produkcije $P \subset V \times (T^* \cdot (V \cup \{\varepsilon\}))$

To pomeni, da imamo pri levo-regularnih gramatikah vmesne simbole lahko le na skrajni levi, pri desno-regularnih pa le na desni.

2.4 Prevedba med izvedbami regularnih jezikov

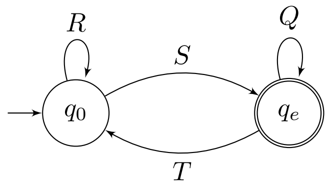
Regularni izrazi, regularne gramatike in končni avtomati so vsi enako močni in je mogoče pretvarjati med njimi. V tem odseku bomo predstavili naslednje prevedbe:



2.4.1 Končni avtomat → Regularni izraz

Končni avtomat v regularni izraz prevedemo po metodi z eliminacijo. Pri tej metodi izberemo neko vozlišče za eliminacijo, nato pa njegove sosede povežemo med seboj, tako, da na nove povezave zapišemo regularne izraze, ki opisujejo dogajanje v tistem vozlišču. Eliminacijo ponavljamo, dokler nam v avtomatu ne ostanta le dve stanji, nato pa za končni zapis uporabimo naslednji recept:

Na povezavah avtomata imamo zapisane regularne izraze R, S, Q in T,

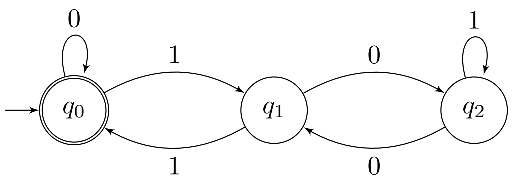


ki jih prepišemo v en sam regularni izraz oblike:

$$(R + SQ^*T)^*SQ^*$$

Primeri:

Primer 1: Zapiši DKA za preverjanje deljivosti s 3 v binarnem sistemu? Zapiši še regularni izraz.



Regularni izraz dobimo po postopku iz 2.4.1:

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

2.5 Ohranjanje regularnosti jezikov

Regularnost jezika po definiciji ohranjajo operacije:

- $L_1 \cup L_2$ - unija
- $L_1 \cdot L_2$ - stik
- L_1^* - iteracija

Obstajajo konstruktivni postopki, ki kažejo, da regularnost ohranjajo tudi:

- $L_1 \cap L_2$ - presek
  Iz avtomatov za L_1 in L_2 zgradimo t.i. produktni avtomat:

$$\begin{aligned} M_{L_1} &= \{Q_1, \Sigma, \delta_1, q_{1_0}, F_1\} \\ M_{L_2} &= \{Q_2, \Sigma, \delta_2, q_{2_0}, F_2\} \\ M_{L_1} * M_{L_2} &= \{Q_1 \times Q_2, \Sigma, \delta_*, \langle q_{1_0}, q_{2_0}\rangle, F_1 \times F_2\} \end{aligned}$$

  Namesto stanj dobimo pare stanj in moramo preveriti v kateri par pridemo, če gledamo oba stara avtomata, končna pa so tista stanja, ki so končna v obeh starih avtomatih.

$$\delta_*(\langle q_1, q_2\rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a)\rangle$$

- L^R - obrat oz. reverz
  Obrnemo vse povezave, ustvarimo novo začetno stanje, ki gre po ε v stara končna, staro začetno stanje pa postane edino končno stanje.

Regularnost ohranjajo tudi vse operacije, ki so sestavljene iz zgoraj naštetih:

- $L_1 \setminus L_2 = L_1 \cap \overline{L}_2$ - razlika
- $\overline{L} = \Sigma^* \setminus L$ - komplement
- $L_1 \veebar L_2 = (L_1 \cup L_2) \setminus (L_1 \cap L_2)$ - ekskluzivni ali

2.6 Dokazovanje regularnosti jezika

Dostikrat hočemo sestaviti regularen izraz za določen jezik in niti ne vemo ali je regularen izraz sploh regularen ali ne. Za to imamo nekaj metod za dokazovanje regularnosti jezika.

Dokazovanje da je jezik regularen

- konstrukcija končnega avtomata εNKA , NKA ali DKA, ki sprejema jezik
- zapis regularnega izraza, ki sprejema jezik.

Dokazovanje da jezik ni regularen

- lema o napihovanju regularnih jezikov
- če dokažemo da jezik ni kontekstno neodvisen - kako to naredimo bomo pokazali v naslednjem poglavju.

Opozorilo: Če zapišemo regularni izraz za jezik, ali naredimo končni avtomat, moramo dobro preveriti da ne obstaja kakšen protiprimer. Torej beseda ki je v jeziku in jo končni avtomat ali regularni izraz ne sprejme, ali obratno.
Če nam ne rata dokaza (da je ali da ni regularni jezik) do konca speljati, to ne moremo vzeti kot dokaz da nasprotno drži. Velja da v takem primeru še nič ne vemo o regularnosti jezika.

Poglavje 3

Kontekstno-neodvisni jeziki

3.1 Kontekstno-neodvisne gramatike

3.2 Skladovni avtomati

Def.: Skladovni avtomat je definiran kot sedmerka $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, kjer je:

- Q - končna množica stanj
- Σ - vhodna abeceda
- Γ - skladovna abeceda
- δ - funkcija prehodov, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$
- q_0 - začetno stanje, $q_0 \in Q$
- Z_0 - začetni skladovni simbol, $Z_0 \in \Gamma$
- F - množica končnih stanj

3.2.1 Trenutni opis

Def.: Trenutni opis je trojka $\langle q, w, \gamma \rangle \in Q \times \Sigma^* \times \Gamma^*$, pri čemer je q trenutno stanje, w preostanek vhodnega niza, ter γ trenutna vsebina sklada

3.2.2 Relacija $\vdash$

Def.: Relacija $\vdash$ nas pelje iz enega trenutnega opisa v drugega, če je ta prehod predviden v funkciji prehodov δ:

$$\langle q, aw, Z\gamma \rangle \vdash \langle p, w, \gamma'\gamma \rangle \iff \langle p, \gamma' \rangle \in \delta(q, a, Z)$$

3.2.3 Jezik skladovnega avtomata

Poglavje 4

Slovar

- Razred - razred je množica elementov, ki ga lahko podamo z naštevanjem elementov ali z opisom lastnosti (opisni ali konceptualni razredi)