

# Izračunljivost in računska zahtevnost

---

Študijsko leto 2018/19

prof. dr. Borut Robič

- Izračunljivost in računska zahtevnost
  - Študijsko leto 2018/19
  - prof. dr. Borut Robič
- 1. Uvod
  - String (beseda)
  - Abeceda
  - Jezik
  - Grafi in drevesa
  - Notacija množic
    - Osnovne operacije nad množicami:
    - Relacije med množicami
      - P-zaprta množica
  - Računski model
    - Neizračunljivi problemi
- 2. Končni avtomati in regularni izrazi
  - Sistem končni stanj
  - Deterministični končni avtomat DKA =  $(Q, \Sigma, \delta, q_0, F)$  DKA=(Q,Σ,δ,q0,F)
    - Definicija DKA
    - Delovanje DKA
  - Nedeterministični končni avtomat NKA =  $(Q, \Sigma, \delta, q_0, F)$  NKA=(Q,Σ,δ,q0,F)
    - Definicija
  - Ekvivalentnost DKA in NKA
  - Nedeterministični končni avtomati s tihimi prehodi  $\backslash varepsilon$ 
    - Definicija
  - Ekvivalentnost NKA in NKA\_  $\backslash varepsilon$ 
    - Definicija
  - Regularni izrazi
    - Definicija regularnih izrazov
    - Operacije nad regularnimi izrazi
      - Združevanje regularnih izrazov
      - Klenovo in pozitivno zaprtje
      - Ekvivalenca med končnimi avtomati in regularnimi izrazi
- 3. Regularne množice
  - Lema o napihovanju
  - Lastnosti regularnih množic
  - Odločitveni algoritmi za regularne množice
    - Praznost in neskončnost regularnih množic
    - Ekvivalentnost končnih avtomatov
  - Myhill-Nerode teorem
    - Definicija Myhill-Nerode teorema
- 4. Kontektsno neodvisne gramatike

- Definicija KNG
  - Definicija jezikov generiranih s KNG
  - Derivacijsko drevo
    - Najbolj leva in najbolj desna derivacija
  - Dvoumne gramatike
  - Poenostavljanje kontekstno neodvisnih gramatik
    - Eliminacija neuporabnih simbolov
    - Eliminacija \varepsilon produkcije
    - Eliminacija enotskih produkcij
  - Normalna oblika Chomskega
  - Normalna oblika Greibachove
  - Skladovni avtomati
    - Sestavni deli skladovnega avtomata
    - Premiki skladovnega avtomata
    - Definicija skladovnega avtomata
    - Skladovni avtomati in kontekstno neodvisni jeziki
      - Ekvivalentnost SA, ki sprejme končno stanje in prazen sklad
      - Ekvivalenca SA in KNJ
      - Moč determinističnih in nedeterminističnih SA
  - 5. Lastnosti kontekstno nedovisnih jezikov
    - Lema o napihovanju za kontekstno neodvisne jezike
    - Lastnosti zaprte nad KNJ
    - Odločitveni algoritmi za KNJ
  - 7. Turingovi stroji
    - Definicija Turingovega stroja
    - Uporaba Turingovega stroja
      - Računanje vrednosti s TS
        - Kapaciteta TS
      - Razpoznavanje množic s TS
        - Zmožnost TS
      - Generiranje množic
      - Izračunljivo preštevni jeziki
      - Odnosi med razredi jezikov
    - Modifikacije Turingovih strojev
      - Turingov stroj s končnim pomnilnikom
      - Turingov stroj z multitračnim trakom
      - Turingov stroj z dvostranskim neskončnim trakom
      - Turingov stroj z več trakovi
      - Turingov stroj z multidimenzionalnim trakom
      - Turingovi stroji z nedeterminističnim programom
    - Univerzalen Turingov stroj
      - Kodiranje Turingovega stroja TODO
      - Obstoj univerzalnega Turingovega stroja
        - Posledice obstoja univerzalnega TS
- 8. Nedoločljivost
  - Vrste računskih problemov
  - Reševanje odločitvenih problemov

- Jezik odločitvenih problemov
- Problem ustavitev
  - Dokaz o nedoločljivosti
- Osnovne vrste odločitveni problemov
  - Komplementarne množice in odločitveni problemi
- Primeri neiračunljivih problemov
- 10. Teorija računske zahtevnosti
  - Deterministični čas in prostor (DTIME, DSPACEDTIME,DSPACE)
    - Deterministična časovna zahtevnost & zahtevnostni razred DTIME
    - Deterministična prostorska zahtevnost & zahtevnostni razred DSPACE
  - Nedeterministični čas in prostor (NTIME, NSPACE)
    - Nedeterministična časovna zahtevnost & zahtevnostni razred NTIME
    - Nedeterministična prostorska zahtevnost & zahtevnostni razred NSPACE
    - Povzetek zahtevnostnih razredov
  - Kompresija traku, linearna pohitritev, redukcija števila trakov
    - Kompresija traku, linearna pohitritev
    - Redukcija števila trakov
  - Relacije med DTIME, DSPACE, NTIME, NSPACE
    - Funkcija, ki se lepo obnaša (well-behaved)
  - Razredi P, NP, PSPACE, NPSPACE
    - Osnovna relacija med razredi
  - $P =^? NPP = ?NP$
  - NP-polni in NP-težki problemi
    - Povzetek

## 1. Uvod

---

String (beseda)

niz znakov z določenim vrstnim redom\

- $|w|$  - dolžina niza  $w$
- $\epsilon$  - prazen niz
  - dolžina praznega niza je 0 ( $|\epsilon| = 0$ )
- **predpona** je kakršna koli dolžina znakov, ki se pojavi na začetku niza
- **pripona** je kakršna koli dolžina znakov, ki se pojavi na repu niza
  - prava pripona/predpona je katerakoli pripona ali predpona, ki ni enaka celotnemu nizu
- **združevanje nizov** -  $prviNiz + drugiNiz = prviNizdrugiNiz$ 
  - vedno velja:  $w\epsilon = \epsilon w = w \quad \forall w$

Abeceda

**Abeceda** je končna množica različnih znakov

Jezik

**Formalen jezik** je množica nizov iz znakov neke abecede. *Prazna množica,  $\emptyset$  in množica, ki vsebuje samo prazen niz,  $\{\epsilon\}$ , so različni jeziki.*

Množica vseh nizov abecede  $\Sigma$  je množica  $\Sigma^*$

Primer:  $\Sigma = \{a\} \Rightarrow \Sigma^* = \{\epsilon, a, aa, aaa, aaaa, \dots\}$

## Grafi in drevesa

**Neusmerjen graf** označujemo z  $G = (V, E)$ , kjer je  $V$  končna množica vozlišč in  $E$  končna množica parov vozlišč imenovanih robovi.

**Pot v grafu** je zaporedje vozlišč grafa tako, da obstaja povezava med dvema soležnima vozliščema v poti.

**Usmerjen graf** označujemo z  $G = (V, A)$ , kjer je  $V$  končna množica vozlišč in  $A$  končna množica urejenih (usmerjenih) parov vozlišč.

Urejeno pot med vozliščema  $u$  in  $v$  označimo kot  $u \rightarrow v$

**Drevo** je graf z lastnostmi:

- obstaja natanko eno vozlišče, imenovano *root* ali *koren*, ki nima predhodnika
- vsako vozlišče (razen *root*) ima natanko enega predhodnika (starša)
- otroci vozlišča so urejeni od leve proti desni

Predhodnika vozlišča imenujemo *starš*, naslednika imenujemo *sin*, če vozlišče nima nobenega sina se imenuje *list*

## Notacija množic

Množica je neurejena kolekcija objektov (članov), ki se ne ponavljajo.

Končne množice lahko definiramo tako, da navedemo **vse** člane množice.

Primer končne množice:  $\{a, b, c\}$  je končna množica.

$\{x | P(x)\}$  -  $x$  je množica, da velja  $P(x)$

$\{x \in A | P(x)\}$  -  $x$  je množica elementov množice  $A$ , da velja  $P(x)$

- če je vsak element množice  $A$  element množice  $B$  pišemo:  $A \subseteq B$  (velja tudi  $B \supseteq A$ )
- če velja  $A \subseteq B$ , vendar ne velja  $A = B$ , potem rečemo, da je  $A$  vsebovan v  $B$ , pišemo  $A \subset B$
- množici  $A$  in  $B$  sta enaki, če vsebujueta vse enake elemente:  $A = B \Leftrightarrow A \subseteq B \wedge B \subseteq A$

## Osnovne operacije nad množicami:

- **unija**  $A \cup B$ :  $\{x | x \in A \vee x \in B\}$
- **presek**  $A \cap B$ :  $\{x | x \in A \wedge x \in B\}$
- **razlika**  $A - B$ :  $\{x | x \in A \wedge x \notin B\}$
- **kartezični produkt**  $A \times B$ :  $\{(x, y) | x \in A \wedge y \in B\}$

## Relacije med množicami

Binarna relacija:

$$R = \{(a, b) | a \in A \wedge b \in B\}$$

Množica lahko ima naslednje lastnosti:

- **refleksivnost**, če velja  $aRa \forall a \in S$
- **irefleksivnost**
- **tranzitivnost**, če velja  $aRb \wedge bRc \Rightarrow aRc$
- **simetričnost**, če velja  $aRb \Leftrightarrow bRa$
- **asimetričnost**

**Ekvivalentna relacija** - če je relacija refleksivna, simetrična in tranzitivna. **Ekvivalentni razredi** so refleksivni, simetrični in tranzitivni.

### P-zaprta množica

P-zaprta množica je najmanjša množica  $R^+$ , ki še ima lastnosti  $R$ .

Formalno:

Če je  $P = \{tranzitivna\}$ , potem P-zaprtost na relaciji  $R$  definiramo kot  $R^+$ , kjer velja:

1. če  $aRb$  potem  $aR^+b$
2. če  $aR^+b \wedge bRc$  potem  $aR^+c$
3. nič ni v  $R^+$  razen, kar sledi iz 1 in 2

## Računski model

**Računski model** je formalna definicija osnovne notacije algoritmičnega računanja. Definira:

- algoritem
- okolje
- kako se algoritem izvede v danem okolju

### Neizračunljivi problemi

Obstaja več problemov kot je algoritmov za računanje. Zato obstajajo funkcije, ki so neizračunljive.

Za funkcijo  $f$  ne obstaja računalniški program (algoritem), ki bi za vsak dan element  $x$  za  $f$  izračunal vrednost  $f(x)$  in bi deloval za vse mogoče  $x$ .

## 2. Končni avtomati in regularni izrazi

### Sistem končni stanj

**Sistem končni stanj** je objekt, ki bere diskrete vhode in je lahko v kateremkoli izmed končno mnogo notranjih konfiguracij imenovanih stanja.

**Stanje** opisuje informacije, ki so potrebne za definicijo naslednjih stanj, glede na dani vhod.

Deterministični končni avtomat  $DKA = (Q, \Sigma, \delta, q_0, F)$

DKA je sestavljen iz končne množice stanj in prehodov med stanji, ki se zgodijo glede na prebrani znak vhodne abecede. Za vsak vhodni podatek je iz stanja natanko en prehod v naslednje stanje.

### Definicija DKA

Deterministični končni avtomat DKA definiramo kot petorček  $(Q, \Sigma, \delta, q_0, F)$ , kjer velja:

- $\mathcal{Q}$  je končna množica stanj,
- $\Sigma$  je vhodna abeceda s končno mnogo znakov,
- $q_0 \in \mathcal{Q}$  je začetno stanje,
- $F \subseteq \mathcal{Q}$  je množica končnih stanj,
- $\delta$  je prehodna funkcija, (primer:  $\delta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ )
- $\delta(q, a)$  je stanje

DKA  $M = (\mathcal{Q}, \Sigma, \delta, q_0, F)$  sprejme niz  $x$ , če velja  $\delta(q_0, x) = p$  za nek  $p \in F$

Jezik, ki ga sprejme DKA, je definiran kot množica:

$$L(M) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}$$

Jezik je regularna množica, če ga sprejme nek DKA.

## Delovanje DKA

Če je DKA v stanju  $q$  in prebere simbol  $a$ , potem v eni potezi:

1. vstopi v naslednje stanje  $\delta(q, a)$
2. premakne svoje okno za en simbol v desno

## Nedeterministični končni avtomat $NKA = (\mathcal{Q}, \Sigma, \delta, q_0, F)$

Če deterministični končni avtomat razširimo tako, da omogočimo nič, enega ali več prehodov iz stanja z istim vhodnim podatkom, dobimo **nedeterministični končni avtomat**.

NKA sprejme niz znakov, če obstaja "pot", ki pripelje od začetnega stanja do enega izmed končnih stanj. Pri DKA je takšna pot ena sama, pri NKA jih je lahko več - zato so NKA nerealni.

## Definicija

Nedeterministični končni avtomat opišemo s petorčkom  $(\mathcal{Q}, \Sigma, \delta, > q_0, F)$ :

- $\mathcal{Q}$  je končna množica stanj,
- $\Sigma$  je vhodna abeceda s končno mnogo vhodnimi znaki,
- $q_0 \in \mathcal{Q}$  je začetno stanje,
- $F \subseteq \mathcal{Q}$  je končna množica končnih stanj,
- $\delta$  je prehodna funkcija  $\delta : \mathcal{Q} \times \Sigma \rightarrow 2^{\mathcal{Q}}$

NKA  $M = (\mathcal{Q}, \Sigma, \delta, > q_0, F)$  sprejme niz  $x$ , če  $\delta(q_0, x)$  vsebuje nek  $p \in F$ .

Jezik, ki ga NKA sprejme, opišemo kot množico:

$$L(M) = \{x \in \Sigma^* \mid \delta(q_0, x) \text{ vsebuje stanje } p \in F\}$$

## Ekvivalentnost DKA in NKA

Vsek DKA je tudi NKA, obratno ne velja.

Vse jezike, ki jih sprejme DKA, jih sprejema tudi NKA. Obstajajo takšni jeziki, ki jih sprejema samo NKA!

## Nedeterministični končni avtomati s tihimi prehodi $\varepsilon$

Razširimo NKA s tihimi prehodom  $\varepsilon$ , to je **prehod ob praznem vhodu**.

### Definicija

Nedeterministični končni avtomat s tihimi prehodom  $\varepsilon$  ( $NKA_\varepsilon$ ) definiramo kot petorček  $(Q, \Sigma, \delta, q_0, F)$ :

- $Q$  je končna množica stanj,
- $\Sigma$  je vhodna abeceda s končno mnogo znaki,
- $q_0 \in Q$  je začetno stanje,
- $F \subseteq Q$  je končna množica končnih stanj,
- $\delta$  je prehodna funkcija  $(\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q)$

$NKA_\varepsilon$  sprejema niz znakov  $x$ , če velja, da prehodna funkcija  $\hat{\delta}(q_0, x)$  vsebuje nek  $p \in F$ .

Jezik, ki ga sprejema  $NKA_\varepsilon$  lahko opišemo z množico:

$$L(M) = \{x \in \Sigma \mid \hat{\delta}(q_0, x) \text{ vsebuje stanje } p \in F\}$$

## Ekvivalentnost NKA in $NKA_\varepsilon$

Zmožnost sprejemanja tihih prehodov  $NKA_\varepsilon$  ne omogoča sprejemanje neregularnih izrazov, zato le ta ni algoritično močnejši od NKA.

**Vsak  $NKA_\varepsilon$  lahko prevedemo v NKA brez tihih prehodov.** NKA in  $NKA_\varepsilon$  sta torej ekvivalentna.

## Regularni izrazi

Jeziki, ki jih sprejemajo končni avtomati, so **regularni izrazi**.

### Definicija regularnih izrazov

Splošne (osnovne) oblike regularni izrazov so:

1.  $a \quad L(a) = \{a\}$
2.  $\varepsilon \quad L(\varepsilon) = \{\varepsilon\}$
3.  $\emptyset \quad L(\emptyset) = \{\}$

Nad osnovnimi oblikami lahko izvajamo operacije ( $r_1, r_2 \in RI$ ):

1.  $r_1 r_2 \quad stik \quad L(r_1 r_2) = L(r_1) L(r_2)$
2.  $r_1 + r_2 \quad izbira \quad L(r_1 + r_2) = L(r_1) \cup L(r_2)$
3.  $r^* \quad Kleenovo zaprtje(ovojnica) \quad L(r^*) = L(r)^*$

### Operacije nad regularnimi izrazi

#### Združevanje regularnih izrazov

Če je  $\Sigma$  jezik in so  $L_1, L_2$  nizi iz  $\Sigma^*$ , potem izraz  $L_1 L_2$  definiramo kot množico:

$$L_1 L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$$

### Klenovo in pozitivno zaprtje

Če je  $L^0 = \{\varepsilon\}$  in  $L^i = LL^{i-1}$  za vsak  $i \geq 1$  potem definiramo Klenovo zaprtje kot množico:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Pozitivno zaprtje pa kot:

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

*Po domači: Klenovo zaprtje definira nize znakov, kjer obstaja tudi nič znakov. Pozitivno zaprtje pa takšnega niza (praznega) ne definira.*

Ekvivalenca med končnimi avtomati in regularnimi izrazi

**Jeziki, ki jih sprejemajo končni avtomati (DKA, NKA, NKA<sub>ε</sub>), so natanko jeziki, ki so definirani z regularnimi izrazi.**

1. Za vsak regularni izraz  $r$  obstaja NKA<sub>ε</sub>, ki sprejema jezik  $L(r)$
2. Za vsak DKA M obstaja regularni izraz, ki definira jezik  $L(M)$

**Sledi:** obstajajo 4 načini definiranja istega razreda jezikov, regularnih množic, to so: DKA, NKA, NKA<sub>ε</sub> in regularni izrazi.

## 3. Regularne množice

### Lema o napihovanju

**Lema o napihovanju dokazuje, da nek jezik ni regularen in/ali, da je neskončen.**

Definicija leme o napihovanju

Naj bo  $L$  regularna množica. Potem obstaja konstanta  $n$ , ki je odvisna samo od izbranega jezika  $L$ , tako da velja: Če obstaja beseda  $z$  za katero velja:

$$z \in L \wedge |z| \geq n$$

potem morajo obstajati besede  $u, v, w$  tako, da velja

$$z = uvw,$$

$$|uv| \leq n,$$

$$|v| \geq 1, \text{ in}$$

$$\forall i \geq 0 : uv^i w \in L$$

*Po domači: Vsaka dovolj dolga beseda  $z$ , ki jo KA sprejema, vsebuje podbesedo  $v$ , ki jo lahko poljubnokrat "napihnemo" in kot rezultat ponovno dobimo besedo, ki jo KA še vedno sprejme.*

Formalen zapis leme o napihovanju

$L \setminus \text{regularen} \Rightarrow (\exists n) (\forall z) [z \in L \wedge |z| \geq n \Rightarrow (\exists u, v, w) [z = uvw \wedge |uv| \leq n \wedge |v| \geq 1 \wedge (\forall i \geq 0) uv^i w \in L]$

## Lastnosti regularnih množic

Regularne množice so zaprte nad naslednjimi lastnostmi:

- unijo
- presek
- stik
- Klenovim zaprtjem
- komplementom
- substitucijo
- homomorfizmom in inverzni homomorfizem
- kvocientom

## Odločitveni algoritmi za regularne množice

Praznот and neskončnost regularnih množic

Množico  $L(M)$ , ki jo sprejema končni avtomat KA  $M$  in ima  $n$  stanj, je:

- neprazna  $\Leftrightarrow M$  sprejme besedo dolžine  $l$ , kjer je  $l < n$
- neskončna  $\Leftrightarrow M$  sprejme besedo dolžine  $l$ , kjer je  $n \leq l < 2n$

Ekvivalentnost končnih avtomatov

Dva končna avtomata sta ekvivalentna, če spejemata isti jezik

$$L(M_1) = L(M_2)$$

## Myhill-Nerode teorem

Obstaja neskončno mnogo determinističnih končnih avtomatov, ki sprejemajo enak jezik. Čeprav so enakovredni se lahko razlikujejo v  $Q, \delta, F$  komponentah.

*Myhill-Nerode teorem določa minimalen deterministični končni avtomat, ki sprejema jezik  $L$ .*

Definicija Myhill-Nerode teorema

Naslednje izjave so ekvivalentne:

1.  $L \subseteq \Sigma^*$  je regularna množica,
2.  $R_L$  je končni indeks,
3.  $L$  je unija nekih ekvivalentnih razredov pravih invariant ekvivalenčne relacije končnega indeksa

Posledica teorema je, da obstaja unikaten minimalen DKA za vsako regularno množico.

Minimalen unikaten DKA za regularno množico  $L$  je unikaten do izomorfizma (preimenovanja stanj).

## 4. Kontekstsno neodvisne gramatike

Kontekstno neodvisne gramatike so končne množice spremenljivk, ki jih imenujemo *nedeterminante*, vsaka predstavlja jezik. Jeziki predstavljeni s spremenljivkami so opisani *rekurzivno* s primitivnimi simboli imenovnimi terminali. Pravila, ki se nanašajo na spremenljivke se imenujejo produkcije.

### Definicija KNG

Kontekstno neodvisno gramatiko opišemo kot četvorček  $G = (V, T, P, S)$ , kjer je:

- $V$  končna množica spremenljivk,
- $T$  končna množica terminalov,
- $P$  končna množica produkcij, vsaka je oblike  $A \rightarrow a$ , kjer je  $A \in V$  in  $a$  beseda v jeziku  $(V \cup T)^*$
- $S$  posebna spremenljivka imenovana *start simbol*

### Definicija jezikov generiranih s KNG

Jezik generiran z KNG  $G = (V, T, P, S)$  je množica

$$L(G) = \{ w \mid w \in T^* \wedge S_G \Rightarrow^* w \}$$

### Derivacijsko drevo

Derivacije lahko predstavimo kot derivacijsko drevo. Vozlišča so označena s *terminali* ali *spremenljivkami* ali s praznim znakom ( $\epsilon$ ).

Če je notranje vozlišče označeno z  $A$  in so njegovi sinovi označeni z  $X_1, X_2, \dots, X_n$  od leve, potem je  $A \rightarrow X_1, X_2, \dots, X_n$  *produkcija*.

#### Definicija derivacijskega drevesa

Naj bo  $G = (V, T, P, S)$  KNG. Drevo imenujemo derivacijsko drevo  $G$ -ja, če velja:

- vsako vozlišče ima oznako, ki je simbol iz  $V \cup T \cup \{\epsilon\}$
- oznaka korena je  $S$
- če ima notranje vozlišče oznako  $A$ , potem velja  $A \in V$
- če ima vozlišče oznako  $A$  in so njegovi sinovi označeni z  $X_1, X_2, \dots, X_n$  od leve, potem je  $A \rightarrow X_1, X_2, \dots, X_n$  *produkcija*
- če ima vozlišče oznako  $\epsilon$ , potem je vozlišče list in edini sin svojega očeta

#### Relacija med derivacijskim drevesom in derivacijami

Če je  $G = (V, T, P, S)$  KNG, potem:

$$S_G \rightarrow^* a \Leftrightarrow \text{obstaja derivacijsko drevo za } G \text{ s krošnjo } a$$

#### Najbolj leva in najbolj desna derivacija

Najbolj leva je tista derivacija, ki v vsakem koraku derivacijskega drevesa izbere najbolj levo derivacijo.

Najbolj desna je tista derivacija, ki v vsakem koraku derivacijskega drevesa izbere najbolj desno derivacijo.

## Dvoumne gramatike

Gramatika  $G$  je *dvoumna*, če obstaja beseda, ki ima več kot eno derivacijsko drevo.

Kontekstno neodvisni jezik  $G$  je *bistveno dvoumen*, če je vsaka gramatika za  $G$  dvoumna.

Primer kontekstno bistveno dvoumnega neodvisnega jezika:

$$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

## Poenostavljanje kontekstno neodvisnih gramatik

Če je  $L$  neprazen KNJ je lahko generiran s KNG  $G$  z naslednjimi lastnostmi:

- vsaka spremenljivka in terminal  $G$ -ja je predstavljen v derivaciji neke besede v  $L$
- ne obstaja produkcija  $A \rightarrow B$ , kjer sta  $A$  in  $B$  spremenljivki
- če  $\epsilon \notin L$ , potem ni nobene produkcije oblike  $A \rightarrow \epsilon$
- če  $\epsilon \in L$  potem lahko zahtevamo:
  - vsaka produkcija ima obliko  $A \rightarrow BC$  ali  $A \rightarrow a$ , kjer so  $A, B, C$  spremenljivke,  $a$  pa terminal ali
  - vsaka produkcija v  $G$  ima obliko  $A \rightarrow aa$ , kjer je  $a$  niz znakov (lahko tudi prazen)

### Eliminacija neuporabnih simbolov

$G = (V, T, E, S)$  je gramatika. Simbol  $X$  je uporaben, če velja derivacija  $S \Rightarrow^* aXb \Rightarrow^* w$  za nek  $a, b, w \in T^*$ . Drugače je  $X$  odvečen.

Vsak neprazen kontekstno neodvisni jezik je generiran s kontekstno neodvisno gramatiko brez nepotrebnih simbolov.

### Eliminacija $\epsilon$ produkcijs

#### Definicija $\epsilon$ produkcijs

je produkcija oblike  $A \rightarrow \epsilon$

Če je  $L = L(G)$  za nek KNG  $G = (V, T, P, S)$ , potem je  $L - \{\epsilon\} = L(G')$  za nek KNG  $G'$  brez nepotrebnih simbolov in/ali  $\epsilon$  produkcijs.

### Eliminacija enotskih produkcijs

#### Definicija enotske produkcijs

je produkcija oblike  $A \rightarrow B$

Vsak kontekstno neodvisni jezik brez  $\epsilon$  je definiran z gramatiko brez nepotrebnih simbolov  $\epsilon$ -produkcijs ali enotskih produkcijs.

## Normalna oblika Chomskega

### Definicija teoremov normalnih oblik

Teroem normalne oblike navaja, da so vse kontekstno neodvisne gramatike ekvivalentne do gramatik z določenimi omejitvami in oblikami produkcijs

### Definicija normalne oblike Chomskega

Vsak kontekstno neodvisen jezik brez  $\epsilon$  je lahko generiran z gramatiko, kjer so vse produkcijs oblike

$$A \rightarrow BC \quad \text{ali} \quad A \rightarrow a$$

kjer so  $A, B, C$  spremenljivke,  $a$  pa terminal

## Normalna oblika Greibachove

### Definicija

Vsak kontekstno neodvisen jezik brez  $\epsilon$  je lahko generiran z gramatiko, kjer so vse produkcijs oblike

$$A \rightarrow a\alpha$$

kjer je  $A$  spremenljivka,  $a$  terminal in  $\alpha$  (lahko prazen) niz spremenljivk.

## Skladovni avtomati

Če končni avtomat dopolnimo s kontrolno enoto za tekoči vhodni trak in skladom dobimo **skladovni avtomat**. Tako kot imajo regularni izrazi končne avtome, imajo kontekstno neodvisne gramatike skladovne avtome.

Nedeterministični skladovni avtomat ima večjo moč od determinističnega - kontekstno neodvisni jeziki, ki jih sprejema deterministični SA, so podmnožica KNJ, ki jih sprejemajo nedeterministični SA.

### Sestavni deli skladovnega avtomata

Skladovni avtomat setavlja naslednje enote:

- vhodni trak - na njem je shranjen program, avtomat lahko na njem hrani vmesne rezultate in izhode
- kontrolna enota
- sklad - niz simbolov neke abecede, najbolj lev simbol razumemo kot vrhnji simbol sklada

### Premiki skladovnega avtomata

Skladovni avtomat ima dva možna premika: *običajen premik* in  $\epsilon$  premik.

- običajen premik:** znak je sprejet. Glede na: stanje  $q$ , vhodni simbol  $a$  in najvišji simbol na skladu  $Z$  obstaja končno mnogo možnosti.
  - $i$ -ti korak je sestavljen iz naslednjega stanja  $p$ , (lahko praznega) niza  $y_i$  (simbolov, ki zamenjajo vrhnji element na skladu)
  - po končanem izboru se okno premakne za 1 naprej

- $\varepsilon$ : od običajnega premika se razlikuje, da znak ni sprejet in na koncu okno ne napreduje za 1 naprej

## Definicija skladovnega avtomata

Skladovni avtomat (SA) definiramo kot sedmerico  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , kjer je:

- $Q$  končna množica stanj,
- $\Sigma$  vhodna abeceda,
- $\Gamma$  skladovna abeceda,
- $q_0 \in Q$  začetno stanje,
- $Z_0 \in \Gamma$  začetni znak sklada,
- $F \subseteq Q$  množica končnih stanj,
- $\delta$  je prehodna funkcija ( mapiranje iz  $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$  v končno množico  $Q \times \Gamma^*$  )

### Definicija determinističnega skladovnega avtomata

Skladovni avtomat  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  je deterministični, če prehodna funkcija  $\delta$  izpolnjuje dva dodatna pogoja za vsak  $q \in Q$  in  $Z \in \Gamma$ :

1.  $\delta(q, \varepsilon, Z) \equiv \emptyset \Rightarrow \forall a \in \Sigma : \delta(q, a, Z) = \emptyset$
2.  $\forall a \in \Sigma \cup \{\varepsilon\} : |\delta(q, a, Z)| \leq 1$

*Prvi pogoj preprečuje možnost izbire med tihim prehodom in običajnim prehodom. Drugi pogoj preprečuje možnost več kot ene možnosti v primeru tihih ali običajnih prehodov.*

**V primeru skladovnih avtomatov, vedno predpostavljamo, da gre za nedeterminističnega!**

### Trenutni opis skladovnega avtomata

Skladovni avtomat lahko v vsakem trenutku opišemo s trojčkom  $(q, \omega, \gamma)$ , kjer je  $q$  stanje,  $\omega$  niz vhodnih podatkov,  $\gamma$  niz skladovnih podatkov.

### Jeziki skladovnega avtomata

Za skladovni avtomat  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  definiramo dvoje jezikov:

- $L(M)$ , ki je sprejet s končnim stanjem

$$L(M) = \{ w \mid (q_0, w, Z_0, ) \vdash^* (p, \varepsilon, \gamma) \text{ za nek } p \in F \wedge \gamma \in \Gamma^* \}$$

- $L(M)$ , ki je sprejet s praznenjem sklada

$$N(M) = \{ w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon) \text{ za nek } p \in Q \}$$

Jezika sta ekvivalenta, če obstaja nek SA, ki sprejema jezik s končnim stanjem, obstaja nek drugi SA, ki sprejema ta isti jezik s praznjenjem sklada.

## Skladovni avtomati in kontekstno neodvisni jeziki

*Ali deterministični SA sprejmejo enak nabor (razred) jezikov kot nedeterministični SA?*

*Ne, nedeterministični SA sprejme jezik  $ww^R$ , deterministični pa ne.*

Ekvivalentnost SA, ki sprejme končno stanje in prazen sklad

Če  $L = L(M_2)$  za nek SA  $M_2$ , potem  $L = N(M_1)$  za nek SA  $M_1$ .

Če  $L = N(M_1)$  za nek SA  $M_1$ , potem  $L = L(M_2)$  za nek SA  $M_2$ .

Kjer je  $L = N(M)$  jezik, ki ga sprejema SA s praznjenjem sklada,  $L = L(M)$  pa SA s končnim stanjem.

**SA s končnim stanjem in SA s praznjenjem sklada sta enakovredna.** Če obstaja jezik, ki ga nek SA sprejme s končnim stanjem, ga nek drug SA sprejme s praznjenjem sklada.

Ekvivalenca SA in KNJ

Če je  $L$  kontektsno neodvisen jezik, potem obstaja SA  $M$ , da je  $L = N(M)$

Če je  $L = N(M)$  za nek SA  $M$ , potem je  $L$  KNJ.

Moč determinističnih in nedeterminističnih SA

Nedeterministični SA sprejema jezik  $\{ww^R \mid w \in (0+1)^*\}$ , deterministični SA ga ne.

Nedeterministični SA je močnejši od determinističnega.

## 5. Lastnosti kontekstno nedovisnih jezikov

Lema o napihovanju za kontekstno neodvisne jezike

**Neformalno:** za katerokoli dovolj dolgo besedo v jeziku  $L$  lahko najdemo dve kratki podbesedi  $v$  in  $x$ , da se ponavljajo vendar bo dana beseda še vedno v  $L$ .

Naj bo  $L$  KNJ, potem obstaja konstanta  $n$ , ki je odvisna samo od  $L$  tako, da velja: če obstaja beseda  $z$ , da velja:

$$z \in L \wedge |z| \geq n$$

potem obstajajo besede  $u, v, w, x, y$  tako, da velja:

$$z = uvwxy$$

$$|vx| \geq 1$$

$$|vwx| \leq n$$

$$\forall i \geq 0 : uv^iwx^i y \in L$$

Lastnosti zaprte nad KNJ

Lastnosti, ki so zaprte nad razredom kontekstno neodvisnih jezikov:

- unija
- povezovanje
- Kleenovo zaprtje
- substitucija (homomorfizem)
- inverzni homomorfizem
- presek z **regularno množico**

Latsnosti, ki **niso** zaprte nad razredom kontekstno neodvisnih jezikov:

- presek
- komplement

## Odločitveni algoritmi za KNJ

Obstajajo odločitveni algoritmi za KNJ, ki nam povejo ali je KNJ:

- prazen
- končen
- nekončen

## 7. Turingovi stroji

### Definicija algoritma

Algoritem za reševanje problema je končna množica *ukazov*, ki pripeljejo *procesor*, v končno mnogo korakih, od *vhodnih podatkov* problema do ustreznega *rezultata*.

### Definicija računskega modela

Računski model je definicija, ki formalizira karakteristike osnovne notacije algoritmičnega računanja - algoritem, okolje in izvedbo algoritma v danem okolju.

### Teza o izračunljivosti (Church-Turingova teza)

Osnovni koncepti računskega modela so formalizirani na način:

algoritem	Turingov program
izračun	izvedba Turingovega programa
računska funkcija	Turingova-računska funkcija

### Definicija Turingovega stroja

Osnovna izvedba Turingovega stroja ima naslednje elemente:

- kontrolno enoto, ki vsebuje Turingov program
- vhodni trak sestavljen iz celic
- premično okno nad trakom

**Trak** uporabljamo za branje in pisanje vhodnih, vmesnih in izhodnih podatkov. Razdeljen je na enako velike celice in je potencialno neskončen v eno smer.

Vsaka celica vsebuje en znak iz tračne abecede  $\Gamma = (z_1, \dots, z_t), t \geq 3$ . Med simboli je poseben simbol ( $z_i$ ), ki definira prazno celico, ob njem sta v tračni abecedi še vsaj dva simbola (1 in 2).

Vhodne podatke hranimo v *vhodni besedi*, ki se nahaja na skrajno levem koncu traku.

**Kontrolna enota** je vedno v enem izmed končno mnogo stanj  $Q = \{q_1, \dots, q_s\}, q_s \geq 1$ . Kjer je

stanje  $q_1$  začetno, nekatera stanja so končna in spadajo v podmnožico končnih stanj  $F \subseteq Q$ .

V kontrolni enoti se nahaja **Turingov program**, vsak Turingov stroj ima svoj Turingov program, ki vsebuje *prehodno funkcijo*  $\delta$  (npr.  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ )

**Okno** se lahko premakne nad katerokoli celico traku, ki ga lahko prebere ali na njega piše (pri čemer prepiše simbol kateri je bil do takrat zapisan). V enem koraku se okno lahko premakne zgolj na sosednjo celico.

## Delovanje Turingovega stroja

### Predpogoji:

1. vhodna beseda je vpisana na začetek traku
2. okno se nahaja na začetku traku
3. kontrolna enota je v začetnem stanju  $q_0$

**Od tu naprej:** TS deluje neodvisno, kot je zapisano v TP.

Če je TS v stanju  $q_i \in Q$  in prebere simbol  $z_r \in \Gamma$ :

- če je  $q_i$  končno stanje TS **stoji**
- če je  $\delta(q_i, z_r) \uparrow$  (TP nima naslednjega ukaza), TS **stoji**
- če je  $\delta(q_i, z_r) \downarrow = (q_j, z_w, D)$ , TS naredi:
  - spremeni stanje v  $q_j$
  - piše  $z_w$  skozi okno
  - premakne okno v smeri  $D \in \{Levo, Desno\}$  ali stoji na mestu  $D \in \{Stoj\}$

Formalno je Turingov stroj sedmerec  $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$

## Uporaba Turingovega stroja

### Računanje vrednosti s TS

Naj bo  $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  TS in  $k \geq 1$ . K-ta funkcija  $T$ -ja je parcialna funkcija  $\varphi_T : (\Sigma^*)^k \rightarrow \Sigma^*$  definirana kot:\

$$\varphi_T(u_1, \dots, u_k) := \begin{cases} v, & \text{if } T \text{ stoji } \wedge \text{vrača na trak } v \wedge v \in \Sigma^* \\ \uparrow, & \text{if } T \text{ ne stoji } \vee \text{trak nima besede } v \in \Sigma^* \end{cases}$$

### Kapaciteta TS

Naj bo  $\varphi : (\Sigma^*)^k \rightarrow \Sigma^*$  funkcija, potem je:

- **izračunljiva**, če obstaja za  $\exists TM$ , ki lahko izračuna  $\varphi$ , kjer koli na  $dom(\varphi) \wedge dom(\varphi) = (\Sigma^*)^k$  (obstaja TS M, ki izračuna  $\varphi$  za katerikoli argument)
- **delno izračunljiva**, če obstaja TS, ki lahko izračuna  $\varphi$  kjerkoli na  $dom(\varphi)$  (obstaja TS M, ki izračuna  $\varphi$ , kadarkoli, ko je  $\varphi$  definiran)
- **neizračunljiva**, ko ne obstaja TS, ki bi lahko izračunal  $\varphi$  kjerkoli na  $dom(\varphi)$  (ne obstaja TS, ki bi izračunal  $\varphi$ , ko je le ta definiran)

## Razpoznavanje množic s TS

Naj bo  $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  Turingov stroj in naj bo  $w \in \Sigma^*$  niz znakov. Rečemo, da  $T$  sprejme  $w$ , če  $q_1 w \vdash^* \alpha_1 p \alpha_2$  za nek  $p \in F$  in  $\alpha_1, \alpha_2 \in \Gamma^*$

Beseda je torej sprejeta, če povzroči, da Turingov stroj vstopi v končno stanje. Jezik je sprejet, če so vse besede iz tega jezika sprejete.

## Zmožnost TS

Če je  $S \subseteq \Sigma^*$  in je  $T$  Turingov stroj lahko za njega rečemo, da je:

- $S$  je **določljiv**, če  $\exists TS M$ , ki odgovori na vprašanje *Ali je  $x \in S$ ?* z **DA/NE** za vsak  $x \in \Sigma^*$
- $S$  je **pol določljiv**, če obstaja nek  $TS M$ , ki odgovori na vprašanje *Ali je  $x \in S$ ?* z **DA** za vsak  $x \in S$
- $S$  je **ne odločljiv**, če ne obstaja  $TS M$ , ki odgovori na vprašanje *Ali je  $x \in S$ ?* z **DA** za vsak  $x \in S$

## Generiranje množic

Naj bo  $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  TS. T imenujemo *generator*, če zaporedno piše na trek znake iz  $\Sigma^*$  in jih razmejuje z znakom  $\#$ . Jezik generiran s TS je množica:

$$G(T) = \{ w \mid w \in \Sigma^* \wedge T \text{ eventualno zapisi } w \text{ na trak} \}$$

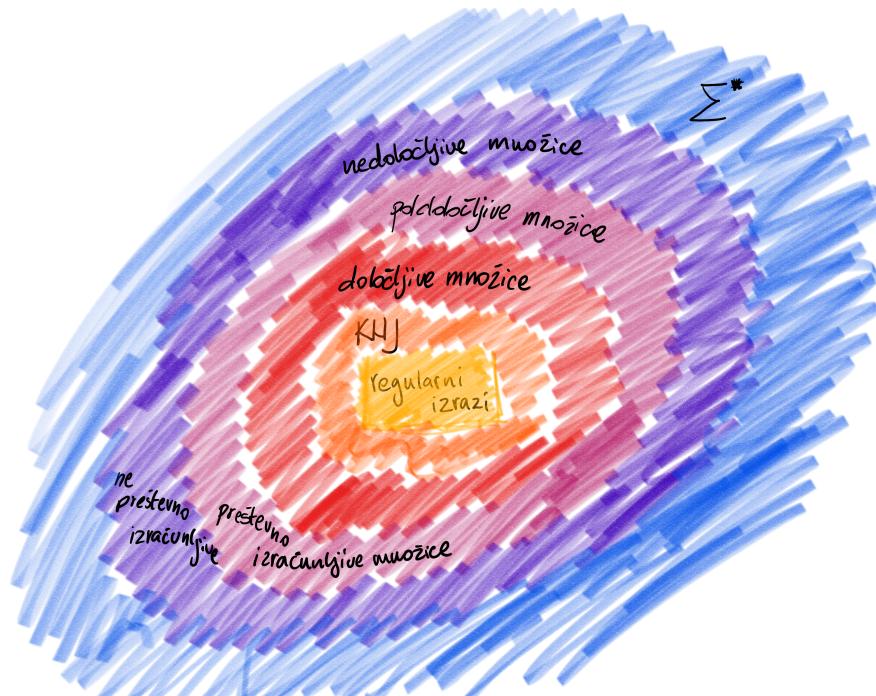
## Izračunljivo preštevni jeziki

Množica  $S$  je izračunljivo preštevna, če obstaja  $S = G(T)$  za nek TS  $T$ .

Množica je izračunljivo preštevna, če jo lahko definiramo s Turingovim strojem.

Množica  $S$  je izračunljivo preštevna  $\Leftrightarrow S$  je delno določljiva.

## Odnosi med razredi jezikov



## Modifikacije Turingovih strojev

## Turingov stroj s končnim pomnilnikom

Ta modifikacija  $V$  ima v kontrolni glavi več spominskih celic, ki hrani podatke.

### Prehodna funkcija za $V$ :

$$\delta_V : Q \times \Gamma \times \Gamma^k \rightarrow Q \times \Gamma \times \{L, R, S\} \times \Gamma^k$$

kjer je  $k$  število pomnilnih celic v kontrolni enoti

## Turingov stroj z multitračnim trakom

Turingov stroj  $V$  ima trak, ki je razdeljen na  $tk \geq 2$  sledi. Na vsaki sledi je lahko hranjen en znak tračne abecede.

### Prehodna funkcija za $V$ :

$$\delta_V : Q \times \Gamma^{tk} \rightarrow Q \times \Gamma^{tk} \times \{L, R, S\}$$

## Turingov stroj z dvostranskim neskončnim trakom

Turingov stroj  $V$  ima potencialno neskončen trak v obe smeri.

### Prehodna funkcija za $V$ :

$$\delta_V : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

## Turingov stroj z več trakovi

Turingov stroj  $V$  ima več trakov, vsak izmed trakov ima svojo bralno-pisalno glavo.

### Prehodna funkcija za $V$ :

$$\delta_V : Q \times \Gamma^{tp} \rightarrow Q \times \{\Gamma^{tp} \times L, R, S\}$$

kjer je  $tp \geq 2$  in predstavlja število trakov.

## Turingov stroj z multidimenzionalnim trakom

Turingov stroj  $V$  ima  $d$ -dimenzionalen trak. Brano-pisalna glava se lahko premakne v d smeri.

### Prehodna funkcija za $V$ :

$$\delta_V : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L_1, R_1, L_2, R_2, \dots, L_d, R_d, S\}$$

## Turingovi stroji z nedeterminističnim programom

Turingov stroj  $V$  ima nedeterministični program in v vsakem koraku določi končno mnogo alternativnih prehodov, ki se izvedejo.

**Vse modulacije Turingovih strojev so si po moči enakovredne. Kar lahko izračuna katerakoli izmed modulacij, lahko izračuna tudi osnovni Turingov stroj (ali katera druga modulacija).**

## Univerzalen Turingov stroj

Kodiranje Turingovega stroja TODO

## Obstoj univerzalnega Turingovega stroja

Idejo, da takšen TS obstaja dokažemo tako, da zgradimo TS, ki lahko simulira delovanje katerega koli drugega Turingovega stroja. Takšen TS dokažemo s teorijo izračunljivosti.

### Posledice obstoja univerzalnega TS

Kot posledica obstoja takšnega stroja se začne gradnja (in izgradnja) računskega stroja, ki bi ga lahko uporabili za več različnih namenov. Nastane **splošno namenski računalnik**.

## 8. Nedoločljivost

---

### Vrste računskih problemov

**Odločitveni problemi** (da/ne problemi): so problemi na katere lahko odgovorimo z da ali z ne (*rešitev je enogni bit*).

**Iskalni problemi:** rezultat je element dane množice  $S$  tako da ima dani element iskano lastnost  $P$  (*rezultat je element množice*).

**Problem štetja:** rezultat je število elementov množice  $S$ , ki imajo lastnos  $P$  (*rezultat je naravno število*)

**Problem generiranja:** rešitev je seznam elementov množice  $S$  z lastnostjo  $P$  (*rešitev je seznam elementov množice*).

### Reševanje odločitvenih problemov

#### Jezik odločitvenih problemov

Pokažemo povezavo med *odločitvenimi problemi* in *množicami* v štirih korakih:

1. Naj bo  $D$  odločitveni problem
2.  $d$  instanca  $D$ , v instanci problema spremenljivko zamenjamo z danim podatkom (npr.  $D$  je *Ali je  $n$  naravno število?*,  $d$  je instanca *Ali je 5 naravno število?*)
3. Iz naravnega jezika prevedemo odločitveni problem v kodo, ki jo razume Turingov stroj

$$D \rightarrow \Sigma^* \text{ naj bo kodirna funkcija}$$

4. Poишčemo vse kode pozitivnih instanc odločitvenega problema  $D$  in jih shranimo v množico  $L(D)$

Velja torej povezava med odločitvenimi problemi in množicami:

$$d \in D \text{ je pozitiven} \Leftrightarrow \langle d \rangle \in L(D)$$

Naj bo  $D$  odločitveni problem za katerega lahko rečemo, da je

- *določljiv* (ali *izračunljiv*), če je  $L(D)$  določljiva množica
- *poldoločljiv*, če je  $L(D)$  poldoločljiva množica
- *nedoločljiv* (ali *neizračunljiv*), če je  $L(D)$  nedoločljiva množica

## Problem ustavitve

Definicija problema ustavitve (*halting problem*)

Problem ustavitve  $D_{halt}$  je definiran kot:

$D_{halt}$  = "Ali se TS  $T$  z vhodno besedo  $w \in \Sigma^*$  zaustavi?"

Problem ustavitve je **nedoločljiv**.

V kolikor bi v prihodnosti zgradili algoritem, ki bi odgovoril na problem ustavitve, bi ugotovili, da ne bo bil sposoben dati odgovora za vsaj en par  $T, W$ .

Dokaz o nedoločljivosti

Preden začnemo z dokazom definiramo **dvoje množic**:

- **Univerzalni jezik**  $K_0$  je jezik problema ustavitve in je definiran kot

$$K_0 = L(D_{halt}) = \{ \langle T, w \rangle \mid T \text{ stoji na } w \}$$

- **Diagonalen jezik**  $K$  je definiran kot:

$$K = \{ \langle T, T \rangle \mid T \text{ stoji na } \langle T \rangle \}$$

Najprej z protislovjem dokažemo, da je  $K$  nedoločljiv.

Ker je  $K$  nedoločljiv je nedoločljiv tudi problem  $D_H$ , ki je podproblem  $D_{halt}$ . Torej je nedoločljiv tudi  $D_{halt}$ .

## Osnovne vrste odločitveni problemov

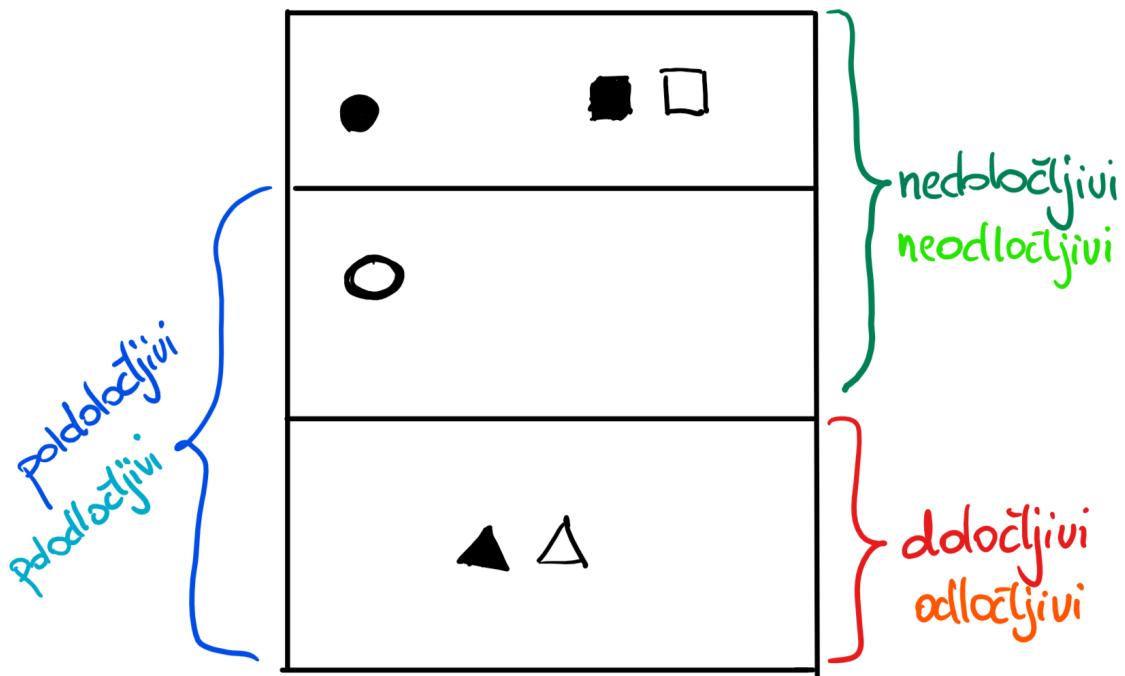
Obstajajo 3 vrste odločitvenih problemov:

- $D$  je **določljiv**: obstaja algoritem, ki reši arbitrarно instanco  $d \in D$ . Takšen algoritem imenujemo *decider* problema  $D$
- $D$  je **poldoločljiv neodoločljiv**: obstaja algoritem, ki reši arbitrarно pozitivno instanco  $d \in D$ . Takšen algoritem imenujemo *recognizer*.
- $D$  ni **poldoločljiv**: ne obstaja algoritem, ki bi rešil arbitrarно pozitivno ali negativno instanco  $d \in D$

Komplementarne množice in odločitveni problemi

- množici  $S \Delta$  in  $\bar{S} \blacktriangle$  sta *določljivi*
- množici  $S \circ$  in  $\bar{S} \cdot$  sta *nedoločljiva*, en je *poldoločljiv*, drugi je *nedoločljiv*
- množici  $S \square$  in  $\bar{S} \blacksquare$  sta *nedoločljivi* in nista *poldoločljivi*

## Vježba za množice in odločitvene probleme



### Primeri neiračunljivih problemov

- **Terminacija algoritma:** ali se algoritem termini, če vsatvimo v njega določene podatke?
- **Pravilnost algoritma:** ali je algoritem pravilen za vse vhodne podatke?
- **Obstoj krajšega algoritma:** ali obstaja algoritem ekvivalenten trenutnemu, ki je krajši?
- **"Busy beaver problem":** neformalno je "busy beaver" najbolj produktiven Turingov stroj, ki ne "ne zapravlja časa"

## 10. Teorija računske zahtevnosti

### Deterministični čas in prostor ( $DTIME$ , $DSPACE$ )

Koliko časa in/ali prostora potrebuje algoritem, da reši (odloči) odločitveni problem  $D$ ?

Koliko korakov ali celic vhodnega traku potrebuje TS, da razpozna jezik  $L(D)$  za odločitveni problem  $D$ ?

Deterministična časovna zahtevnost & zahtevnostni razred  $DTIME$

Naj bo  $M = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  deterministični TS z  $k \geq 1$  dvostranskim neskončnim trakom.

TS  $M$  ima deterministično časovno zahtevnost  $T(n)$ , če za vsak vhod  $w \in \Sigma^*$  dolžine  $n$ , naredi  $\leq T(n)$  korakov preden se zaustavi.

Jezik  $L$  ima deterministično časovno kompleksnost  $T(n)$ , če obstaja TS  $M$  s časovno kompleksnostjo  $T(n)$  tako, da velja  $L = L(M)$ .

Takšen razred jezikov definiramo kot:

$$DTIME(T(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima čas. zahtevnost } T(n) \}$$

Odločitveni problem  $D$  ima deterministično časovno kompleksnost  $T(n)$ , če ima njegov jezik  $L(D)$  časovno kompleksnost  $T(n)$ .

Takšen razred odločitvenih problemov definiramo kot:

$$DTIME(T(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima čas. zahtevnost } T(n) \}$$

## Deterministična prostorska zahtevnost & zahtevnostni razred $DSPACE$

Naj bo  $M = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  deterministični TS z enim *vhodnim trakom* in  $k \geq 1$  *delovnimi trakovi*.

TS  $M$  ima deterministično prostorko zahtevnost  $S(n)$ , če za vsako besedo  $w \in \Sigma^*$  dolžine  $n$ , uporabi preden se zaustavi, kvečjemu  $\leq S(n)$  celic na vsakem delovnem traku.

Jezik  $L$  ima deterministično prostorsko zahtevnost  $S(n)$ , če obstaja deterministični TS  $M$  z (deterministično) prostorko zahtevnostjo  $S(n)$  in velja  $L = L(M)$ .

Takšen razred jezikov definiramo kot:

$$DSPACE(S(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima prostorsko zahtevnost } S(n) \}$$

Odločitveni problem  $D$  ima deterministično prostorsko zahtevnost  $S(n)$ , če obstaja jezik  $L(D)$  z (deterministično) prostorko zahtevnostjo  $S(n)$ .

Takšen razred jezikov definiramo kot:

$$DSPACE(S(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima prostorsko zahtevnost } S(n) \}$$

## Nedeterministični čas in prostor ( $NTIME$ , $NSPACE$ )

### Nedeterministična časovna zahtevnost & zahtevnostni razred $NTIME$

Naj bo  $N = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  nedeterministični TS.

TS  $N$  ima nedeterministično časovno zahtevnost  $T(n)$ , če za vsak vhod  $w \in \Sigma^*$  dolžine  $n$ , obstaja  $\leq T(n)$  korakov preden se zaustavi.

Jezik  $L$  ima nedeterministično časovno kompleksnost  $T(n)$ , če obstaja TS  $N$  s časovno kompleksnostjo  $T(n)$  tako, da velja  $L = L(N)$ .

Takšen razred jezikov definiramo kot:

$$NTIME(T(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima nedet. čas. zahtevnost } T(n) \}$$

Odločitveni problem  $D$  ima deterministično časovno kompleksnost  $T(n)$ , če ima njegov jezik  $L(D)$  časovno kompleksnost  $T(n)$ .

Takšen razred odločitvenih problemov definiramo kot:

$$NTIME(T(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima nedet. čas. zahtevnost } T(n) \}$$

Nedeterministična prostorska zahtevnost & zahtevnostni razred  $NSPACE$

Naj bo  $N = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$  nedeterministični TS z enim vhodnim trakom in  $k \geq 1$  delovnimi trakovi.

TS  $M$  ima nedeterministično prostorko zahtevnost  $S(n)$ , če za vsako besedo  $w \in \Sigma^*$  dolžine  $n$ , obstaja, preden se zaustavi,  $\leq S(n)$  celic na vsakem delovnem traku.

Jezik  $L$  ima nedeterministično prostorsko zahtevnost  $S(n)$ , če obstaja deterministični TS  $N$  z (nedeterministično) prostorko zahtevnostjo  $S(n)$  in velja  $L = L(N)$ .

Takšen razred jezikov definiramo kot:

$$NSPACE(S(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima nedet. prostorsko zahtevnost } S(n) \}$$

Odločitveni problem  $D$  ima nedeterministično prostorsko zahtevnost  $S(n)$ , če obstaja jezik  $L(D)$  z (nedeterministično) prostorko zahtevnostjo  $S(n)$ .

Takšen razred jezikov definiramo kot:

$$NSPACE(S(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima nedet. prostorsko zahtevnost } S(n) \}$$

Povzetek zahtevnostnih razredov

### **Formalni jeziki:**

$$DTIME(T(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima čas. zahtevnost } T(n) \}$$

$$DSPACE(S(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima prostorsko zahtevnost } S(n) \}$$

$$NTIME(T(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima nedet. čas. zahtevnost } T(n) \}$$

$$NSPACE(S(n)) = \{ L \mid L \text{ je jezik} \wedge L \text{ ima nedet. prostorsko zahtevnost } S(n) \}$$

### **Odločitveni problemi:**

$$DTIME(T(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima čas. zahtevnost } T(n) \}$$

$$DSPACE(S(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima prostorsko zahtevnost } S(n) \}$$

$$NTIME(T(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima nedet. čas. zahtevnost } T(n) \}$$

$$NSPACE(S(n)) = \{ D \mid D \text{ je odločitveni problem} \wedge D \text{ ima nedet. prostorsko zahtevnost } S(n) \}$$

### **Neformalno:**

$$DTIME(T(n)) = \{\text{odločitveni problem rešljiv deterministično v času } T(N)\}$$

$$DSPACE(S(n)) = \{\text{odločitveni problem rešljiv deterministično v prostoru } S(N)\}$$

$$NTIME(T(n)) = \{\text{odločitveni problem rešljiv nedeterministično v času } T(N)\}$$

$$NSPACE(S(n)) = \{\text{odločitveni problem rešljiv nedeterministično v prostoru } S(N)\}$$

## Kompresija traku, linearna pohitritev, redukcija števila trakov

Časovno (združevanje korakov) in prostorsko (združevanje simbolov) kompleksnost lahko zmanjšamo za konstanten faktor.

### Kompresija traku, linearna pohitritev

$$DTIME(T(n)) = DTIME(cT(n))$$

$$NTIME(T(n)) = NTIME(cT(n))$$

$$DSPACE(S(n)) = DSPACE(cS(n))$$

$$NSPACE(S(n)) = NSPACE(cS(n))$$

Namesto pisanja  $DTIME(n^2)$  pišemo  $O(n^2)$

### Redukcija števila trakov

Če omejimo TS na en trak, se lahko časovna kompleksnost kvadrira, če ga omejimo na 2 trakova pa je izguba manjša.

Če  $L \in DTIME(T(n))$ , potem  $L$  sprejema eno tračni TS v  $T^2(n)$  času.

Če  $L \in NTIME(T(n))$ , potem  $L$  sprejema eno tračni TS v  $T^2(n)$  času.

Če  $L \in DTIME(T(n))$ , potem  $L$  sprejema dvo tračni TS v  $T(n) \log T(n)$  času.

Če  $L \in NTIME(T(n))$ , potem  $L$  sprejema dvo tračni TS v  $T(n) \log T(n)$  času.\

## Relacije med $DTIME$ , $DSPACE$ , $NTIME$ , $NSPACE$

Zamenjava *nedeterminističnega* algoritma z *determinističnim* lahko povzroči v najslabšem primeru eksponentno rast časovne kompleksnosti in kvadratno rast prostorske kompleksnosti.

$$DTIME(T(n)) \subseteq DSPACE(T(n))$$

kar lahko rešimo v času  $O(T(n))$  lahko rešimo na prostoru  $O(T(n))$

$$L \in DSPACE(S(n)) \wedge S(n) \geq \log_2 n \Rightarrow \exists c : L \in DTIME(c^{S(n)})$$

kar lahko rešimo na prostoru  $O(S(n))$  lahko rešimo v najslabšem primeru v času  $O(T^{S(n)})$

$$NSPACE(S(n)) \subseteq DSPACE(S^2(n)), \text{ če } S(n) \geq \log_2 \wedge S(n) \text{ se "lepo - obnaša"}$$

kar lahko rešimo nedeterministično na prostoru  $O(S(n))$  lahko rešimo deterministično na prostoru  $O(S^2(n))$

### Funkcija, ki se lepo obnaša (*well-behaved*)

TO-DO

## Razredi $P$ , $NP$ , $PSPACE$ , $NPSPACE$

Gre za zahtevnostne razrede osnovane na polinomski časovni/prostorski zahtevnosti, saj so algoritmi rešljivi v polinomskem času/prostoru definirani kot "razumski".

$$P = \bigcup_{i \geq 1} DTIME(n^i)$$

je razred vseh odločljivih problemov rešljivih v polinomskem času

$$NP = \bigcup_{i \geq 1} NTIME(n^i)$$

je razred vseh odločljivih problemov rešljivih v nedeterminističnem polinomskem času

$$PSPACE = \bigcup_{i \geq 1} DSPACE(n^i)$$

je razred vseh odločljivih problemov rešljivih na polinomskem prostoru

$$NPSPACE = \bigcup_{i \geq 1} NSPACE(n^i)$$

je razred vseh odločljivih problemov rešljivih na nedeterminističnem polinomskem prostoru

Osnovna relacija med razredi

$$P \subseteq NP \subseteq PSPACE = NPSPACE$$

Vsek deterministični TS s polinomsko časovno zahtevnostjo lahko vidimo kot trivialno nedeterminističnega.

$$P =? NP$$

Od prej vemo, da drži da k deterministično rešljivem problemu na polinomskem prosturu, nedeterminizem ne doda nič ( $PSPACE = NPSPACE$ ). Torej ali velja tudi  $P = NP$ ?

Verjetno ne drži, ker bi posledice bile preveč "presenetljive" Robič 2018.

## *NP-polni* in *NP-težki* problemi

"Najtežji" problemi v NP so definirani kot problemi  $D^*$  z lastnostmi:

- $D^* \in NP$
- $D \leq D^*$ , za vsak  $D \in NP$

Problem  $D^*$  je definiran kot NP-težki, če velja  $D \leq^p D^*$ , za vsak  $D \in NP$ .

Problem  $D^*$  je definiran kot NP-polni, če velja:

- $D^* \in NP$
- $D \leq D^*$ , za vsak  $D \in NP$

## Povzetek

Če velja  $P \equiv NP$ , potem je NP razred sestavljen iz:

