

Izračunljivost in računska zahtevnost

1) Osnove

- **niz/beseda** = končno mnogo staknjenih simbolov iz abecede
- **dolžina niza** = $|w|$, število simbolov, ki sestavlja w
- **abeceda** Σ = končna množica simbolov
- **formalni jezik L** = množica nizov simbolov neke abecede
- **drugi jezik Σ^*** = množica vseh nizov simbolov nad fiksno abecedo Σ
- **model računanja:** FA, PDA, TM

Model računanja (računski model) je stroga matematična definicija, ki formalno opisuje osnovne pojme algoritemskega računanja (algoritmom, njegovo okolje in njegovo izvajanje v okolju).

2) Končni avtomati

- deterministični končni avtomat, DFA
 - definicija DFA

Deterministični končni avtomat (DFA) je peterka: $(Q, \Sigma, \delta, q_0, F)$

- ◊ Q je končna množica stanj
- ◊ Σ je končna množica vhodne abecede
- ◊ $q_0 \in Q$ je začetno stanje
- ◊ $F \subseteq Q$ je množica končnih stanj
- ◊ δ je funkcija prehodov $\delta : Q \times \Sigma \rightarrow Q$
- $\delta(q, a)$
vsakemu paru (stanje, simbol) priredi neko stanje

Za vsak vhodni simbol je točno 1 prehod iz vsakega stanja.

- razširjena funkcija prehodov

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

$$\hat{\delta}(q, \varepsilon) = q$$

$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$



Za vse nize w in vhodne simbole a .

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

- niz sprejet z DFA

Niz x je sprejet z DFA M , če $\delta(q_0, x) = p$ za $p \in F$

Če niz x privede M iz začetnega v končno stanje p .

- jezik sprejet z DFA

Jezik L sprejet z DFA M je množica $L(M) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}$

Množica vseh nizov, ki privedejo avtomat iz začetnega v končno stanje.

- jezik je regularna množica (regularen jezik)

Jezik L je regularna množica (regularen jezik), če je sprejet z nekim DFA.

if \exists DFA $M : L' = L(M)$

Če je jezik L $L(M)$ za kakšen DFA M rečemo, da je L regularni jezik. Regularni jeziki so jeziki, ki jih prepoznajo DFA.

- nedeterministični končni avtomat, NFA

- definicija NFA

Nedeterministični končni avtomat (NFA) je peterka: $(Q, \Sigma, \delta, q_0, F)$

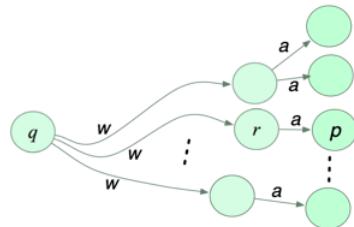
- ◊ Q je končna množica stanj
- ◊ Σ je končna množica vhodne abecede
- ◊ $q_0 \in Q$ je začetno stanje
- ◊ $F \subseteq Q$ je množica končnih stanj
- ◊ δ je funkcija prehodov $\delta : Q \times \Sigma \rightarrow 2^Q$

0, 1 ali več prehodov iz stanja glede na enak vhodni simbol (lahko je v več stanjih naenkrat).

- razširjena funkcija prehodov

$$\hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q$$

$$\begin{aligned}\hat{\delta}(q, \varepsilon) &= \{q\} \\ \hat{\delta}(q, wa) &= \{p \in Q \mid \exists r \in \hat{\delta}(q, w) : p \in \delta(r, a)\}\end{aligned}$$



- niz sprejet z NFA

Niz x je sprejet z NFA M , če $\delta(q_0, x)$ vsebuje nek $p \in F$
(i.e. $\delta(q_0, x) \cap F \neq \emptyset$).

- jezik sprejet z NFA

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

- enakovrednost DFA in NFA

Vsek DFA je tudi NFA. Za vsak NFA obstaja enakovreden DFA (ki sprejme enak jezik kot NFA).

L je jezik sprejet z NFA M. Obstaja DFA M', ki sprejme natanko ta jezik L.

- NFA z tihimi prehodi

- definicija NFA z tihimi prehodi

NFA z ϵ -prehodi (NFA $_{\epsilon}$) je peterka: $(Q, \Sigma, \delta, q_0, F)$

- Q je končna množica stanj
 - Σ je končna množica vhodne abecede
 - $q_0 \in Q$ je začetno stanje
 - $F \subseteq Q$ je množica končnih stanj 
 - δ je funkcija prehodov $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$

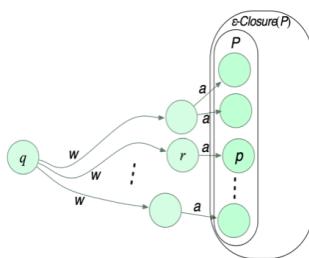
- razširjena funkcija prehodov

$$\hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q$$

- ◆ $\hat{\delta}(q, \varepsilon) = \varepsilon\text{-Closure}(q)$
 - ◆ For $w \in \Sigma^*$ and $q \in \Sigma$ we have

$$\hat{\delta}(g, wa) \equiv \varepsilon\text{-Closure}(P)$$

where $P = \{p \mid \exists r \in \hat{\delta}(q, w) : p \in \delta(r, a)\}$



- epsilon zaprtje

ε - zaprtje(q): množica stanj, ki so dosegljiva iz stanja q z samo ε -prehodi

- niz sprejet z NFA tihimi prehodi

Niz x je sprejet z NFA $_{\epsilon}$ M, če $\hat{\delta}(q_0, x)$ vsebuje nek $p \in F$

- jezik sprejet z NFA tihimi prehodi

Jezik sprejet z NFA ϵ -M je množica $L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \text{ contains a state in } F\}$

- enakovrednost NFA z in NFA brez tihih prehodov

Vsek NFA je tudi NFA ϵ . Za vsak NFA ϵ obstaja enakovredni NFA (v smislu, da sprijemimo enak jezik).

L je množica sprejeta z NFA_ε M. Obstaja NFA M', ki sprejme L.

3) Regularni izrazi

- **stik**

Stik L_1, L_2 , označen z L_1L_2 , je množica

$$L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$$

- **klinovo zaprtje**

Naj bo $L \subseteq \Sigma^*$, $L^0 = \{\varepsilon\}$ in $L^i = LL^{i-1}$ za $i \geq 1$.

Klinovo zaprtje (zaprtje) L , označeno z L^* , je množica

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

L^* je množica besed, ki so strukturirane z stikom poljubno številom besed iz L .

- **pozitivno zaprtje**

Pozitivno zaprtje L , označeno z L^+ , je množica

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

L^+ je množica besed, ki so strukturirane z stikom poljubno številom besed (izključeno je nič besed).

- **regularni izrazi**

Regularni izrazi (r.i.) nad Σ (in množice, ki jih označuje) so induktivno definirani:

- 1) \emptyset je r.i.; označuje prazno množico, \emptyset
- 2) ε je r.i.; označuje množico (z enim elementom, prazna beseda) $\{\varepsilon\}$
- 3) Za vsak $a \in \Sigma$, a je r.i.; označuje množico oz. jezik, v katerem je simbol a $\{a\}$
- 4) Če sta r in s regularna izraza in predstavljata jezika R in S , potem
 - a) $(r + s)$ je r.i.; označuje množico $R \cup S$ (unija R in S)
 - b) (rs) je r.i.; označuje množico RS (stik R in S)
 - c) (r^*) je r.i.; označuje množico R^* (Kleenovo zaprtje R)

- **enakovrednost končnih avtomatov in regularnih izrazov**

Za poljubni regularni izraz r obstaja NFA $_{\varepsilon}$, ki sprejme jezik $L(r)$.

Za poljubni DFA M obstaja regularni izraz r , ki opisuje $L(M)$.

Jeziki, ki so sprejeti s končnimi avtomati so natanko jeziki, ki so opisani z regularnimi izrazi.

Vsi definirajo enak razred jezikov, regularne množice.

4) Lastnosti regularnih množic

- lema o napihanju za regularne množice

Za dokazovanje, da so določeni jeziki neregularni.

Naj bo L regularna množica.

V tem primeru obstaja neka konstanta n (odvisna samo od L), tako da velja sledeče:

(*predpostavka*) če je z poljubna beseda

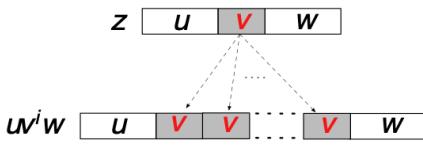
$$z \in L \text{ in } |z| \geq n$$

(*posledice*) potem za besede u, v, w , velja:

$$z = uvw$$

$$|uv| \leq n$$

$$|v| \geq 1 \text{ (v srednjem delu je vsaj 1 znak)}$$



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

n je največje število stanj najmajšega FA, ki sprejme L

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

- **značilnosti zaprtja regularnih množic**

Operacije na jezikih, ki ohranljajo regularne množic. Za dokazovanje, da so določeni jeziki regularni.

- unija
- stik
- Klinovo zaprtje
- komplement: $\Sigma^* - L$
- presek
- zamenjava
- homorfizem
- kvocient

- odločitveni algoritmi za regularne množice

- praznost in končnost regularnih množic oz. jezikov

Množica $L(M)$ sprejeta z FA M z n stanji je:

1) ***neprazna***, če M sprejme besedo dolžine ℓ , kjer je $\ell < n$.

Poglej, če je kakšna beseda dolžine $\ell < n$ v $L(M)$.

2) ***neskončna***, če M sprejme besedo dolžine ℓ , kjer je $n \leq \ell < 2n$.

Poglej, če je kakšna beseda dolžine $n \leq \ell < 2n$ v $L(M)$.

- enakovrednost končnih avtomatov

Dva končna avtomata M_1 in M_2 sta enakovredna, če spremeta enak jezik: $L(M_1) = L(M_2)$.

- **Myhill-Nerode izrek in minimizacija FA**

Za dokazovanje, da so določeni jeziki neregularni.

- **minimalni DFA**

DFA, ki ima izmed vseh DFA, ki sprejmejo L , najmanjše število stanj.

- **relacija nad jezikom, R_L**

Naj bo $L \subseteq \Sigma^*$ poljuben jezik. Relacija R_L na Σ^* : $xR_Ly \iff \forall z \in \Sigma^*: xz \in L \Leftrightarrow yz \in L$.

Dve besedi $x, y \in \Sigma^*$ sta v relaciji R_L natanko takrat, ko sta njuni poljubni razširitvi xz, yz ali obe v L ali obe zunaj L . R_L je ekvivalenčna relacija. R_L razdeli L v ekvivalenčne razrede. Število ekvivalenčnih razredov je indeks od R_L . Na splošno, indeks R_L je končen ali neskončen.

- **relacija nad DFA, R_M**

Naj bo $M = (Q, \Sigma, \delta, q_0, F)$ DFA. $xR_My \iff \delta(q_0, x) = \delta(q_0, y)$.

Relacija R_M na Σ^* : $xR_My \Rightarrow \forall z \in \Sigma^*: xzR_Myz$

Dve besedi $x, y \in \Sigma^*$ sta v relaciji R_M če končni avtomat M iz začetnega stanja q_0 pride do istega stanja $q \in Q$ pri obeh besedah. R_M je ekvivalenčna relacija. Razdeli Σ^* v ekvivalenčne razrede (stanja q , ki so dosegljiva iz q_0). Število razredov je indeks R_M . Indeks R_M je končen ($|Q|$ končna). $L(M)$ je unija teh ekvivalenčnih razredov, kar ustreza končnim stanjem $q \in F$.

R_M je desni invariant.

- **Myhill Nerode izrek**

Izrek (Myhill-Nerode) Naslednje trditve so ekvivalentne (čim ena velja, tudi drugi dve veljata):

- 1) $L \subseteq \Sigma^*$ je regularna množica.
- 2) Relacija R_L , ki temu jeziku pripada, ima končni indeks R_L .
(končno število ekvivalenčnih razredov)
- 3) L je unija nekaj ekvivalenčnih razredov desne invariančne ekvivalenčne relacije z končnim indeksom R_M (množica stanj je končna).

Posledica: obstaja edinstven DFA z minimalnimi stanji za vsako regularno množico.

5) Kontekstno neodvisne gramatike

• kontekstno neodvisne gramatike

- Kaj naredi gramatiko kontekstno neodvisno?

Uporaba produkcij ni odvisna od konteksta (aSb, a in b ne vplivata na produkcijo S-ja)

- CFG

Kontekstna neodvisna gramatika (CFG) je četverica: $G = (V, T, P, S)$ kjer:

- V je končna množica spremenljivk
- T je končna množica terminalov
- P je končna množica produkcij
vsaka je oblike $A \rightarrow \alpha$, kjer je $A \in V$ in α je beseda iz jezika $(V \cup T)^*$
- S je posebna spremenljivka imenovana začetni simbol

- neposredna izpeljava

Naj bo $A \rightarrow \beta$ produkcija in $\alpha, \gamma \in (V \cup T)^*$ poljubna niza.

$\alpha A \gamma$ neposredno izpelje $\alpha \beta \gamma$, če **uporabimo** produkcijo $A \rightarrow \beta$ na nizu $\alpha A \gamma$

Dva niza sta povezana z relacijo $G \Rightarrow$, če prvi neposredno izpelje drugega z uporabo produkcije v G.

- jezik gramatike CFG G

Je množica

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

Jezik gramatike G je množica besed sestavljenih iz samo terminalov, ki jih lahko v končno mnogih korakih izpeljemo iz začetnega simbola gramatike G.

- kontekstno neodvisen jezik

Jezik L je kontekstno neodvisen (CFL), če je $L(G)$ za nek CFG G.

- stavčna oblika

Niz $\alpha \in (V \cup T)^*$ je stavčna oblika, če $S \xrightarrow{G}^* \alpha$.

Niz je stavčna oblika, če v gramatiki G iz začetnega simbola S lahko izpeljemo ta niz (vsi nizi ki jo lahko izpeljemo v gramatiki).

- ekvivalentni gramatiki

Dve gramatiki G_1 in G_2 sta ekvivalentni, če $L(G_1) = L(G_2)$.

- drevesa izpeljav

- drevo izpeljave za CFG

Naravni opis izpeljave stavčne oblike gramatike.

Drevo je drevo izpeljave za CFG G, če:

1. Vsako vozlišče v ima oznako, ki je simbol iz $V \cup T \cup \{\epsilon\}$.
2. Oznaka korena je S.
3. Če je vozlišče v notranje in označeno z A, potem mora biti A v V.
4. Če je vozlišče v označeno z A in so vozlišča v_1, \dots, v_k sinovi vozlišča A (od leve proti desni) z oznakami X_1, \dots, X_k potem je $A \rightarrow X_1 \dots X_k$ produkcija v P.
5. Če ima vozlišče v oznako ϵ , potem je list in edini sin svojega očeta.

- rezultat drevesa izpeljave

Rezultat drevesa izpeljave je niz, ki ga dobimo, če preberemo oznake listov od leve proti desni.

α je rezultat drevesa izpeljave za $G = (V, T, P, S)$ iff $S \xrightarrow{G}^* \alpha$.

- poddrevo drevesa izpeljave

Poddrevo drevesa izpeljave je neko vozlišče drevesa skupaj z njenimi potomci, povezavami in njihovimi oznakami. Če je koren poddrevesa označen z A, potem se to poddrevo imenuje A-drevo.

- povezava med drevesi izpeljav in izpeljavami

$S \xrightarrow{G}^* \alpha$ iff obstaja drevo izpeljave za G z rezultatom α .

- leva in desna izpeljava

Izpeljava je **leva**, če je na vsakem koraku izpeljave uporabljeni produkciji nad skrajno levo spremenljivko v trenutni stavčni obliku.

Izpeljava je **desna**, če je na vsakem koraku izpeljave uporabljeni produkciji nad skrajno desno spremenljivko v trenutni stavčni obliku.

- dvoumna CFG

Če beseda w pripada $L(G)$ za CFG G, potem ima beseda vsaj eno drevo izpeljave (w ima edinstveno levo in desno izpeljavo).

CFG G je **dvoumna**, če ima beseda več kot eno drevo izpeljave.

- bistveno dvoumen CFL

CFL L je **bistveno dvoumen**, če je vsaka CFG za L dvoumna.

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

- **poenostavitev CFG/omejitev oblik produkcij**

Vsak neprazen CFL ($L - \{\epsilon\}$) je generiran z CFG brez nepotrebnih simbolov, ϵ -produkcijs in enotskih produkcijs.

- **nepotreben/potreben simbol**

Simbol X je **potreben**, če obstaja izpeljava $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ za neke α, β in $w \in T^*$.

Drugače je X **nepotreben**.

- **epsilon produkcija**

Epsilon produkcija je produkcija oblike $A \rightarrow \epsilon$.

- **enotska produkcija**

Enotska produkcija je produkcija oblike $A \rightarrow B$.

- **normalna oblika Chomskega**

Vsak CFL brez ϵ je lahko generiran z gramatiko v kateri je vsaka produkcija oblike:

$$A \rightarrow BC$$

$$A \rightarrow a$$

A, B, C so spremenljivke, a je terminal.

- **normalna oblika Greibachove**

Vsak CFL brez ϵ je lahko generiran z gramatiko v kateri je vsaka produkcija oblike:

$$A \rightarrow b\gamma$$

A je spremenljivka, b je terminal, γ je niz spremenljivk.

6) Skladovni avtomati

- definicije

- PDA

Skladovni avtomat (PDA) je sedmerica $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

- ◊ Q končna množica stanj
- ◊ Σ končna množica vhodne abecede
- ◊ Γ končna množica skladovne abecede
- ◊ $q_0 \in Q$ začetno stanje
- ◊ $Z_0 \in \Gamma$ začetni simbol
- ◊ $F \subseteq Q$ množica končnih stanj
- ◊ δ funkcija prehodov

$$Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

- PDA prehodi

- regularen (običajen) prehod

$$\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \dots, (p_m, \gamma_m)\}$$

PDA je v stanju q , prebere vhodni simbol a , vrhnji simbol na skladu je Z .

Za vsak $1 \leq i \leq m$ lahko vstopi v stanje p_i , zamenja simbol Z na skladu z nizom γ_i in premakne okno za en simbol.

- ε -prehod

$$\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \dots, (p_m, \gamma_m)\}$$

PDA je v stanju q , neodvisno od vhodnega simbola, vrhnji simbol na skladu je Z .

Za vsak $1 \leq i \leq m$ lahko vstopi v stanje p_i , zamenja simbol Z na skladu z nizom γ_i in ne premakne okna.

- trenutni opis PDA

Trenutni opis (ID) je trojka (q, w, γ) , kjer je q stanje, w je niz vhodnih simbolov, ki jo mora stroj še prebrati in γ je niz skladovnih simbolov.

Če je $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ PDA,
potem lahko ID $(q, ax, Z\beta)$ neposredno preide v ID $(p_i, x, \gamma_i\beta)$,
če $\delta(q, a, Z)$ vsebuje (p_i, γ_i) . (a je lahko vhodni simbol ali ε)
 $(q, ax, Z\beta) \xrightarrow{M} (p_i, x, \gamma_i\beta)$

- jeziki sprejeti s PDA

L je sprejet s praznim skladom nekega PDA iff L je sprejet s končnim stanjem nekega PDA.

- **L (M)** ... jezik sprejet s končnim stanjem

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \gamma) \text{ for some } p \in F \text{ and } \gamma \in \Gamma^*\}$$

L (M) vsebuje besedo w, če po branju besede w, M (nedeterminizem) pride v končno stanje.

Zaporedje prehodov povzroči, da PDA vstopi v končno stanje.

- **N (M)** ... jezik sprejet s praznim skladom

$$N(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon) \text{ for some } p \in Q\}.$$

N (M) vsebuje besedo w, če ima M po branju besede w (nedeterminizem) prazen sklad.

Zaporedje prehodov povzroči, da PDA sprazne svoj sklad.

- osnovni izrek PDA

L je sprejet s PDA iff L je CFL.

- deterministični PDA

PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je determinističen, če δ izpolnjuje dva pogoja za vsak $q \in Q$ and $Z \in \Gamma$:

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

Pogoj 1 prepreči možnost *izbire med ε -premikom in regularnim premikom*.

Pogoj 2 prepreči možnost *izbire v primeru ε -premika in možnost izbire v primeru regularnega premika*.

- **skladovni avtomati in kontekstno neodvisni jeziki**

- **enakovrednost sprejetja s končnim stanjem in s praznim skladom**

Razred jezikov sprejetih s PDA končnim stanjem je enak kot razred jezikov sprejetih s PDA praznim skladom.

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

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

- **enakovrednost PDA in CFL**

Razred jezikov sprejetih s PDA je točno razred CFL.

Če je L CFL, potem obstaja PDA M , da velja $L = N(M)$.

Če je $L = N(M)$ za PDA M , potem je L CFL.

- **deterministični in nedeterministični PDA**

Deterministični PDA je manj močan kot nedeterministični PDA.

(obstajajo CFL, ki niso sprejeti z nobenim determinističnim PDA)

7) Značilnosti kontekstno neodvisnih jezikov

- lema o napihovanju za CFL: za dokazovanje, da določeni jeziki niso kontekstno neodvisni

Naj bo L CFL.

Obstaja konstanta n (odvisno samo od jezika L) tako, da velja:

če je z beseda, za katero velja

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

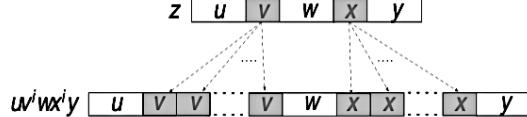
potem obstajajo besede u, v, w, x, y in velja

$$z = uvwxy$$

$$|vxy| \geq 1$$

$$|vwx| \leq n$$

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



- **značilnosti zaprtja:** za dokazovanje, da so/niso določeni jeziki kontekstno neodvisni; operacije, ki ohranijo CFL

Razred CFL je **zaprt** za:

- unijo
- stik
- Klinovo zaprtje
- zamenjavo (homomorfizem)
- inverzni homomorfizem

Razred CFL **ni zaprt** za:

- presek
- komplement

Ampak: razred CFL je zaprt za presek z *regularno množico*:

Izrek

Če je L CFL in R regularna množica, potem je $L \cap R$ CFL.

- **odločitveni algoritmi**

- **neprazen**

$L(G)$ je neprazen iff začetni simbol S generira kakšen niz terminalov.

- **končen**

Gramatika mora biti v normalni obliki Chomskega in brez nepotrebnih spremenljivk.

$L(G)$ je končen iff nastali graf nima ciklov.

- **problem pripadnosti za CFG**

Obstaja odločitveni algoritem, ki za poljuben CFG G in poljubno besedo $x \in T^*$ odgovori na vprašanje "Ali je x člen $L(G)$?".

8) Turingov stroj

- teza o izračunljivosti (Church-Turingova teza)

Algoritem je formaliziran z Turingovim programom, **računanje** je formalizirano z izvajanjem Turingovega programa v Turingovem stroju in **izračunljiva funkcija** je formalizirana z Turingovo izračunljivo funkcijo.

Vse kar je možno izračunati v intuitivnem smislu, je mogoče izračunati z enim od standardnih univerzalnih modelov računanja in obratno.

- **Turingov stroj**

- **o Turingovem stroju**

Turingov stroj ima kontrolno enoto, ki vsebuje Turingov program. Njegov trak je sestavljen iz celic (tračni simbol).

TM je sedmerica $T = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$.

Q ... končna množica stanj // kontrolna enota

Σ ... končna množica vhodne abecede

Γ ... končna množica tračne abecede

δ ... funkcija prehodov: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ // Turingov program

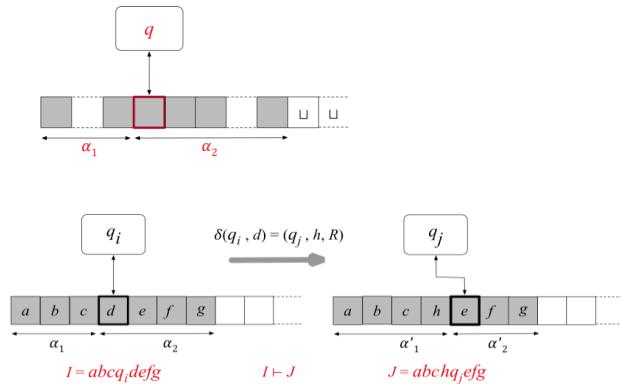
q_1 ... začetno stanje

B ... prazen simbol

F ... množica končnih stanj

- trenutni opis TM

Trenutni opis je niz $I = \alpha_1 q \alpha_2$, če je trenutna konfiguracija TM takšna, da je okno nad prvim simbolom niza α_2 in α_2 se konča na najbolj desnem nepraznem simboli.



Kdaj se I lahko neposredno spremeni v J .

ID I se lahko neposredno spremeni v ID J $I \vdash J$ -- če obstaja navodilo v TM programu, čigar izvedba spremeni I v J .

Ali je relacija \vdash neposredne spremembe tranzitivna?

Ni tranzitivna, saj iz $a \vdash b$ in $b \vdash c$ ne sledi $a \vdash c$.

Relacija opisuje posamezno spremembo, ki definira začetno in končno stanje v vsakem koraku.

Kaj pomeni Kleenejevo zaprtje \vdash^* .

Vsi trenutni opisi TM, ki se jih da po preslikovalni funkciji po $x \geq 0$ potezah iz nekega trenutnega stanja doseči.

- **uporaba Turingovega stroja**

- **računanje vrednosti funkcij**

Naj bo $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ TM in $k \geq 1$.

K-mestna lastna funkcija: parcialna funkcija, ki preslika k besed v eno besedo

$$\varphi_T : (\Sigma^*)^k \rightarrow \Sigma^*$$

$$\varphi_T(u_1, \dots, u_k) =$$

- v : T se ustavi in ima na traku besedo $v \in \Sigma^*$
- \uparrow : T se ne ustavi ali trak nima besede v Σ^*

Dana je funkcija φ , kontruirati moramo TM T za katerega velja $\varphi_T = \varphi$.

Funkcija φ

- **izračunljiva:** $\exists \text{TM, ki lahko izračuna } \varphi \text{ kjerkoli na dom}(\varphi) \wedge \text{dom}(\varphi) = (\Sigma^*)^k$

(obstaja tak TM, ki lahko izračuna vrednosti φ za vsak argument)

- **parcialno izračunljiva:** $\exists \text{TM ki lahko izračuna } \varphi \text{ kjerkoli na dom}(\varphi)$

(obstaja tak TM, ki lahko izračuna vrednosti φ kadarkoli je φ definirana)

- **neizračunljiva:** ne obstaja TM, ki bi lahko izračunal φ kjerkoli na dom(φ)

(ne obstaja noben tak TM, ki bi lahko izračunal vrednosti φ kadarkoli je φ definirana)

- **razpoznavanje pripadnosti množicam**

Dana je množica $S \subseteq \Sigma^*$, poišči TM T, ki sprejme S. Dan je jezik (množica) S, konstruirati moramo TM T, za katerega velja $L(T) = S$.

Beseda w je sprejeta z TM T, če $q_i w \vdash^* \alpha_1 p \alpha_2, p \in F, \alpha_1 \alpha_2 \in I^*, w \in \Sigma^*$.

(Beseda je sprejeta s TM T, če se po končno mnogo korakih TM ustavi v nekem končnem stanju).

Jezik L sprejet z TM T je množica $L(T) = \{w \mid w \in \Sigma^* \wedge w \text{ je sprejeta z } T\}$.

Množica S $\subseteq \Sigma^*$

- **odločljiva:** če obstaja TM, ki odgovori z DA/NE na "Je $x \in S?$ " za vsak $x \in \Sigma^*$
- **polodločljiva:** če obstaja TM, ki odgovori z DA na "Je $x \in S?$ " kadarkoli $x \in S$
- **neodločljiva:** če ne obstaja TM, ki odgovori z DA/NE na "Je $x \in S?$ " za vsak $x \in \Sigma^*$

- **generiranje množic**

Dana je množica S , generiraj seznam x_1, x_2, \dots , ki so točno elementi S .

Dana je množica S , konstruiraj TM T za katerega velja $G(T) = S$.

TM T je **generator**, če piše na trak besede iz Σ^* v zaporedju in razmejevano z $\#$.

Jezik generiran z TM T je množica $G(T) = \{w \mid w \in \Sigma^* \wedge T \text{ sčasoma zapiše } w \text{ na trak}\}$.

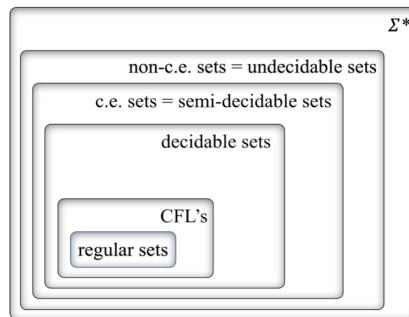
Množica S je **preštevna**, če elemente množice lahko naštejemo (damo v neko zaporedje).

Izračunljivo preštevne množice

Množica S je **izračunljivo preštevna** (c.e.), če je $S = G(T)$.

(množica S lahko generiramo z Turingovim strojem)

Množica S je **izračunljivo preštevna iff S je polodločljiva**.



- **razširitve Turingovih strojev**

Osnovni model T lahko izračuna vse, kar lahko izračuna V (razširitveni TM).

- **TM s končnim pomnilnikom**
- **TM z večslednim trakom**
- **TM z dvosmernim trakom**
- **TM z večtračnim trakom**
- **TM z večdimenzionalnim trakom**
- **TM z nedeterminističnim programom**

Turingov program δ dodeli vsakemu (q_i, z_r) končno množico alternativnih prehodov.

Z njim definiramo minimalno število korakov, ki jih potrebujemo, da izračunamo rešitev (če rešitev obstaja).

- **kodiranje TM**

TM T želimo predstaviti z besedo nad neko kodirno abecedo.

Kodiranje TP δ :

1. Če je $\delta(q_i, z_j) = (q_k, z_\ell, D_m)$ navodilo za δ , potem zakodiramo navodilo z besedo

$$K = 0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$$

kjer $D_1 = L$, $D_2 = R$, $D_3 = S$, $z_1 = 0$, $z_2 = 1$, $z_3 = \sqcup$

2. Tako zakodiramo vsako navodilo δ .

3. Iz obstoječih kod K_1, K_2, \dots, K_r konstruiramo kodo $\langle \delta \rangle$: $\langle \delta \rangle = 111K_111K_211 \dots 11K_r111$

- **oštevilčenje TM**

$\langle T \rangle$ je binarna koda nekega naravnega števila in predstavlja indeks Turingovega stroja T.

Vsako naravno število je indeks točno enega Turingovega stroja.

Naravno število katerega binarna koda ni v zahtevani obliki je indeks praznega Turingovega stroja.

Oštevilčenje Turingovih strojev: n gre čez vsa naravna števila in dobimo zaporedje

Turingovih strojev: T_0, T_1, T_2, \dots

- **univerzalni TM**

Univerzalni TM U izračuna vse kar je izračunljivo s kakšnim drugim Turingovim strojem.

Kot vhod sprejme TM T (njegovo kodo) in njegov vhod (besedo w) ter simulira izvajanje TM T nad vhodno besedo w.

9) Neodločljivost

- **računski problemi**

odločitveni problemi (DA ali NE), **problemis iskanja** (element množice),
problemis štetje (naravno število), **problemis generiranja** (zaporedje elementov množice)

- **odločitveni problemi**

- **jezik odločitvenega problema**

D je odločitveni problem.

Primerek **d** problema D je lahko **pozitiven/negativen**, če je odgovor za primerek d DA/NE.

Kodirna funkcija: $D \rightarrow \Sigma^*$, spremeni vsak primerek d odločitvenega problema D v

besedo $w \in \Sigma^*$.

$\langle d \rangle$ je množica kod vseh primerkov problema D.

Jezik odločitvenega problema D je množica

$L(D) = \{ \langle d \rangle \in \Sigma^* \mid d \text{ je pozitiven primerek problema } D \}$.

Reševanje odločitvenega problema D lahko prevedemo v razpoznavanje množice $L(D)$.

Odgovor za primerek d problema D lahko najdemo, če določimo kako je $\langle d \rangle$ povezan z $L(D)$.

$d \in D$ je pozitiven $\iff \langle d \rangle \in L(D)$

- kaj nam **razpoznavnost $L(D)$** pove o rešljivosti **problema D**

- $L(D)$ je **odločljiva** $\iff D$ je **odločljiv (izračunljiv)**

Obstaja algoritem, ki za vsak $d \in D$, odgovori z DA ali NE.

Odločitveni problem D je odločljiv, če je $L(D)$ odločljiva množica.

- $L(D)$ je **polodločljiva** $\iff D$ je **polodločljiv**

Obstaja algoritem, ki

- za vsak pozitivni primerek $d \in D$, odgovori z DA
 - za negativen primerek $d \in D$ v končnem času lahko odgovori z NE ali pa sploh ne odgovori

Odločitveni problem D je polodločljiv, če je $L(D)$ polodločljiva množica.

- $L(D)$ je **neodločljiva** $\iff D$ je **neodločljiv (neizračunljiv)**

Ne obstaja algoritem, ki bi za vsak $d \in D$ odgovoril z DA ali NE.

Odločitveni problem D je neodločljiv, če je $L(D)$ neodločljiva množica.

- **problem ustavitve (neodločljiv problem)**

- **problem ustavitve, D_{Halt}**

Problem ustavitve je **neodločljiv**.

Ne obstaja algoritem, ki bi za poljuben TM T in besedo w odgovoril na vprašanje "Ali se T ustavi na w?" z DA ali NE.

- **univerzalni jezik, neodločljiv**

Jezik, ki pripada odločitvenemu problemu ustavitve.

$$L_u = L(D_{\text{Halt}}) = \{ \langle T, w \rangle \mid T \text{ se ustavi na } w \}$$

- **diagonalni jezik, neodločljiv**

Jezik, ki pripada odločitvenemu problemu **D_H** (podproblem problema ustavitve, w = ⟨T⟩).

$$L_d = \{ \langle T, T \rangle \mid T \text{ se ustavi na } \langle T \rangle \}$$

Dokažemo, da je L_d neodločljiv posledično je tudi njegov pripadajoči problem D_H neodločljiv. Ker pa je to podproblem bolj splošnega problema ustavitve, je tudi D_{halt} neodločljiv problem.

- **osnovne vrste odločitvenih problemov**

- **neodločljivi in nepolodločljivi**

(algoritem vedno odpove na nekem primerku, ki je poz. ali neg.)

- komplement L_u (komplement D_{halt})
- komplement L_d (komplement D_H)

- **neodločljivi in polodločljivi**

(dobimo vsaj odgovore DA)

- L_u (D_{Halt})
- L_d (D_H)

Razred vseh odločitvenih problemov je sestavljen iz

- razred **odločljivih** problemov
- razred **neodločljivih** problemov
- razred **polodločljivih** problemov (vsebuje vse odločljive in nekaj neodločljivih problemov)

Odločitveni problem D je lahko

- **odločljiv**

Obstaja algoritem, ki lahko reši poljubni primerek $d \in D$.

- **polodločljivo neodločljiv**

Ne obstaja algoritem, ki bi rešil poljuben primerek $d \in D$ ampak obstaja algoritem, ki lahko reši poljubnen pozitiven primerek $d \in D$.

- **nepolodločljiv**

Za poljuben algoritem obstaja tak pozitiven in negativen primerek $d \in D$, da ju algoritem ne more rešiti.

- **S in njen komplement (enako velja za odločitvene probleme)**

- S in komplement S sta **odločljiva**
- S in komplement S sta **neodločljiva**; en je **pododločljiv**, drugi **nepolodločljiv**
- S in komplement S sta **neodločljiva**; **noben** od njiju ni polodločljiv

- **izreki iz teorije izračunljivosti (odločljivost in polodločljivost množic/jezikov)**

S, A in B so poljubne množice.

$$S \text{ je odločljiva} \implies S \text{ je polodločljiva}$$

$$S \text{ je odločljiva} \implies \overline{S} \text{ je odločljiva}$$

$$S \text{ in } \overline{S} \text{ sta polodločljivi} \implies S \text{ je odločljiva}$$

$$S \text{ in } \overline{S} \text{ sta polodločljivi} \implies \overline{S} \text{ je odločljiva}$$

$$A \text{ in } B \text{ sta polodločljivi} \implies A \cap B \text{ in } A \cup B \text{ sta polodločljivi}$$

$$A \text{ in } B \text{ sta odločljivi} \implies A \cap B \text{ in } A \cup B \text{ sta odločljivi}$$

Komplement polodločljivo neodločljivega jezika ne more biti polodločljiv.

10) Teorija računske zahtevnosti

Koliko časa ali prostora potrebuje algoritom, da reši odločitveni problem D?

Koliko korakov ali tračnih celic potrebuje TM, da prepozna jezik odločitvenega problema L(D)?

• deterministična časovna zahtevnost (razred zahtevnosti DTIME)

- TM M

M ima deterministično časovno zahtevnost **T(n)**, če za vsako vhodno besedo $w \in \Sigma^*$

dolžine n, M naredi kvečjemu $\leq T(n)$ korakov pred ustavljivijo.

- jezik L in razred vseh takih jezikov

Jezik L ima deterministično časovno zahtevnost **T(n)**, če obstaja DTM M, ki sprejme jezik L v času, ki je kvečjemu T(n) in velja $L = L(M)$.

DTIME(T(n)) = {L | L je jezik \wedge L ima deterministično časovno zahtevnost T(n)}

(Vsi jeziki L, za katere je problem $w \in L$ deterministično rešljiv v kvečjemu $\leq T(|w|)$ času)

- odločitveni problem in razred vseh takih odločitvenih problemov

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

DTIME(T(n)) = {D | D je odločitveni problem \wedge D ima deterministično časovno

zahtevnost T(n)}

(odločitveni problemi, ki so deterministično rešljivi v času T(n))

• nedeterministična časovna zahtevnost (razred zahtevnosti NTIME)

- TM N

N ima nedeterministično časovno zahtevnost **T(n)**, če za vsako vhodno besedo $w \in \Sigma^*$

dolžine n obstaja nek izračun, ki naredi kvečjemu $\leq T(n)$ korakov pred ustavljivijo.

- jezik L in razred vseh takih jezikov

Jezik L ima nedeterministično časovno zahtevnost **T(n)**, če obstaja NTM N, ki sprejme jezik L v času, ki je kvečjemu T(n) in velja $L = L(N)$.

NTIME(T(n)) = {L | L je jezik \wedge L ima nedeterministično časovno zahtevnost T(n)}

- odločitveni problem in razred vseh takih odločitvenih problemov

Odločitveni problem D ima nedeterministično časovno zahtevnost **T(n)**, če ima njegov jezik $L(D)$ nedeterministično časovno zahtevnost T(n).

NTIME(T(n)) = {D | D je odločitveni problem \wedge D ima nedeterministično časovno zahtevnost T(n)}

- deterministična prostorska zahtevnost (razred zahtevnosti DSPACE)
 - TM M

M ima deterministično prostorsko zahtevnost $S(n)$, če za vsako vhodno besedo $w \in \Sigma^*$ dolžine n , M uporabi $\leq S(n)$ celic na vsakem delovnem traku pred ustavljivoj.
(odloči se $w \in L(M)$ na prostoru $\leq S(n)$)

- jezik L in razred vseh takih jezikov
- Jezik L** ima deterministično prostorsko zahtevnost $S(n)$, če obstaja DTM M deterministične prostorske zahtevnosti $S(n)$, da velja $L = L(M)$.
- DSPACE(S(n))** = $\{L \mid L \text{ je jezik} \wedge L \text{ ima deterministično prostorsko zahtevnost } S(n)\}$
(Množica jezikov L za katere je problem $w \in L$ deterministično rešljiv na $\leq S(|w|)$ prostoru)
- odločitveni problem in razred vseh takih odločitvenih problemov
- Odločitveni problem D** ima deterministično prostorsko zahtevnost $S(n)$, če ima njegov jezik $L(D)$ deterministično prostorsko zahtevnost $S(n)$.
- DSPACE(S(n))** = $\{D \mid D \text{ je odločitveni problem} \wedge D \text{ ima deterministično prostorsko zahtevnost } S(n)\}$
(odločitveni problemi, čigar primerki d so lahko deterministično rešljivi na $\leq S(|d|)$ prostoru)

- nedeterministična prostorska zahtevnost (razred zahtevnosti NSPACE)
 - TM N

N ima nedeterministično prostorsko zahtevnost $S(n)$, če za vsako vhodno besedo $w \in \Sigma^*$ dolžine n obstaja nek izračun, ki uporabi $\leq S(n)$ celic na vsakem delovnem traku pred ustavljivoj.

 - jezik L in razred vseh takih jezikov

Jezik L ima nedeterministično prostorsko zahtevnost $S(n)$, če obstaja NTM N nedeterministične prostorske zahtevnosti $S(n)$ in velja $L = L(N)$.

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

 - odločitveni problem in razred vseh takih odločitvenih problemov

Odločitveni problem D ima nedeterministično prostorsko zahtevnost $S(n)$, če ima njegov jezik $L(D)$ nedeterministično prostorsko zahtevnost $S(n)$.

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

- **stiskanje trakov, linearne pohitritev, zmanjševanje števila trakov**

Prostorsko/časovno zahtevnost je lahko vedno zmanjšana s konstantnim faktorjem (z zakodiranjem večih tračnih simbolov v enega/z združitvijo nekaj korakov v enega).

- **stiskanje trakov (prostor)**

Če ima jezik L prostorsko zahtevnost $S(n)$, potem ima za vsako konstanto $c > 0$ L prostorsko zahtevnost $c S(n)$.

$$\mathbf{DSPACE}(S(n)) = \mathbf{DSPACE}(cS(n))$$

in

$$\mathbf{NSPACE}(S(n)) = \mathbf{NSPACE}(cS(n))$$

- **linearne pohitritev (čas)**

Izpolnjena morata biti **2 pogoja**: $k > 1$ in $\inf_{n \rightarrow \infty} T(n)/n = \infty$ ($T(n)$ raste hitreje kot n).

Če ima jezik L časovno zahtevnosti $T(n)$, potem ima za vsako konstanto $c > 0$ L časovno zahtevnost $cT(n)$.

$$\mathbf{DTIME}(T(n)) = \mathbf{DTIME}(cT(n))$$

in

$$\mathbf{NTIME}(T(n)) = \mathbf{NTIME}(cT(n))$$

$$\mathbf{DTIME}(0.33 n^2) = \mathbf{DTIME}(n^2) = \mathbf{DTIME}(4n^2) = \dots$$

D ima deterministično časovno zahtevnost reda $O(n^2)$.

- **zmanjševanje števila trakov (k)**

- **zmanjševanje števila k vpliva na časovno zahtevnost**

Če je $L \in \mathbf{DTIME}(T(n))$, potem je jezik L sprejet v času $O(T^2(n))$ z 1-tračnim TM.

Če je $L \in \mathbf{NTIME}(T(n))$, potem je jezik L sprejet v času $O(T^2(n))$ z 1-tračnim NTM.

Če je $L \in \mathbf{DTIME}(T(n))$, potem je jezik L sprejet v času $O(T(n) \log T(n))$ z 2-tračnima TM.

Če je $L \in \mathbf{NTIME}(T(n))$, potem je jezik L sprejet v času $O(T(n) \log T(n))$ z 2-tračnima NTM.

- **zmanjševanje števila k ne vpliva na prostorsko zahtevost**

Če je L sprejet s k -delovnimi traki TM prostorske zahtevnosti $S(n)$, potem je L sprejet z 1-delovnim trakom TM prostorske zahtevnosti $S(n)$.

- relacije med DTIME, DSPACE, NTIME in NSPACE

Med

- razredi enake vrste

$\text{RAZRED}(f_1(n)) \subseteq \text{RAZRED}(f_2(n)) \subseteq \dots$ za funkcije $f_i(n)$, $i = 1, 2, \dots$

$\text{DTIME}(n) \subseteq \text{DTIME}(n^2) \subseteq \dots$ // če je rešljivo v n^2 času, je rešljivo tudi v n času

- različnimi razredi

- $\text{DTIME}(T(n)) \subseteq \text{DSPACE}(T(n))$

Kar je lahko rešljivo v času $O(T(n))$, je lahko rešljivo tudi na prostoru $O(T(n))$.

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

Kar je lahko rešljivo na prostoru $O(S(n))$, je lahko rešljivo (*največ*) v času $O(c^{S(n)})$.

(c je odvisen od L .)

- $L \in \text{NTIME}(T(n)) \Rightarrow \exists c : L \in \text{DTIME}(c^{T(n)})$

Kar je lahko rešljivo v *nedeterminističnem* času $O(T(n))$, je lahko rešljivo v *največ determinističnem* času $O(c^{T(n)})$.

Zamenjava nedeterminističnega algoritma z determinističnim povzroči največ eksponentno rast v času, ki je potreben za reševanje problema.

- $\text{NSPACE}(S(n)) \subseteq \text{DSPACE}(S^2(n))$, če $S(n) \geq \log_2 n \wedge S(n)$ je lepa, pohlevna funkcija

Kar je lahko rešljivo na *nedeterminističnem prostoru* $O(S(n))$, je lahko rešljivo na *determinističnem prostoru* $O(S^2(n))$.

Zamenjava nedeterminističnega algoritma z determinističnim povzroči največ kvadratno rast na prostoru, ki je potreben za reševanje problema.

- lepe, pohlevne funkcije zahtevnosti

- prostorsko predstavljiva funkcija $S(n)$

Funkcija $S(n)$ je **prostorsko predstavljiva**, če obstaja TM M prostorske zahtevnosti $S(n)$, da za vsak n obstaja vhod dolžine n na katerem M uporabi točno $S(n)$ tračnih celic.

Funkcija $S(n)$ je **popolnoma prostorsko predstavljiva**, če M za vsak n uporabi točno $S(n)$ celic za vsak vhod dolžine n .

- časovno predstavljiva funkcija $T(n)$

Funkcija $T(n)$ je **časovno predstavljiva**, če obstaja TM M časovne zahtevnosti $T(n)$, da za vsak n obstaja vhod dolžine n na katerem M uporabi točno $T(n)$ prehodov.

Funkcija $S(n)$ je **popolnoma časovno predstavljiva**, če M za vsak n naredi točno $T(n)$ prehodov za vsak vhod dolžine n .

- **razredi zahtevnosti P, NP, PSPACE, NPSPACE**

Razredi zahtevnosti DTIME(T(n)), NTIME(T(n)), DSPACE(S(n)), NSPACE(S(n)), ki imajo funkcije zahtevnosti T(n) in S(n) so **polinomi** (zahteve računanja za računske vire - čas in prostor sta obravnavana kot smiselna, če sta omejena z polinomom).

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

je razred vseh odločitvenih problemov rešljivih *v determinističnem polinomskev času*

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

je razred vseh odločitvenih problemov rešljivih *nedeterministično v polinomskev času*

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

je razred vseh odločitvenih problemov rešljivih *deterministično na polinomskev prostoru*

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

je razred vseh odločitvenih problemov rešljivih *nedeterministično na polinomskev prostoru*

i (1 do neskončnosti, velikost primerka, ki se rešuje; npr. n^2 , nek polinom od n-ja)

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

- **P ⊆ NP**

Vsek deterministični TM polinomske časovne zahtevnosti si lahko predstavljam kot (trivialni) *nedeterministični TM enake časovne zahtevnosti*.

- **NP ⊆ PSPACE**

Če je $L \in NP$, potem obstaja $\exists k$, da je $L \in NTIME(n^k)$.

$L \in NSPACE(n^k) \dots$ Savitch ... $L \in DSPACE(n^{2k})$.

Sledi $L \in PSPACE$.

- **PSPACE = NPSPACE**

$PSPACE \subseteq NPSPACE$

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

Savitch ... $\bigcup_{i \geq 1} DSPACE(n^i) \subseteq PSPACE$

Če je prostorska zahtevnost polinomska, nedeterminizem ne doda nič k računski moči.

Savitch: $L \in NSPACE(n^k) \subseteq L \in DSPACE(n^{2k})$.

- vprašanje $P =? NP$

Če je $P = NP$, potem je vsak $D \in NP$ deterministično rešljiv v polinomskem času.

Je res, da če je časovna zahtevnost polinomska, potem nedeterminizem ne doda nič k računski moči?

Prevladuje prepričanje, da je $P \neq NP$ ($P \subsetneq NP$).

- polinomsko-časovna prevedba

Problem $D \in NP$ je **polinomsko-časovno prevedljiv** na problem D' , $D \leq^p D'$, če obstaja deterministični TM M polinomske časovne zahtevnosti, ki za poljuben $d \in D$ vrne $d' \in D'$, tako da je d pozitiven $\iff d'$ je pozitiven.

Relacija \leq^p je **polinomsko-časovna prevedba**.

- NP-težek problem

Problem D^* je **NP-težek**, če

- za vsak $D \in NP$: $D \leq^p D^*$ (D^* je ali ni v NP)

- NP-poln problem

Problem D^* je **NP -poln**:

- $D^* \in NP$
- za vsak $D \in NP$: $D \leq^p D^*$ (D^* je NP-težek)

Če je problem NP-poln, potem je tudi NP-težek. Obratno ne velja.

Primer NP-polnega problema: problem SAT (problem izpolnjivosti), particije

- dokazovanje NP-polnih problemov

Naj bo $D \leq^p D'$. Potem:

- $D' \in P \Rightarrow D \in P$
- $D' \in NP \Rightarrow D \in NP$

Vsek problem D , ki je lahko \leq^p -zmanjšan na problem v P (ali v NP), je tudi v P (ali v NP).

Relacija \leq^p je tranzitivna: $D \leq^p D' \wedge D' \leq^p D'' \Rightarrow D \leq^p D''$.

(Če prvi problem s prevedbo prevedemo na drug problem in drug problem z drugo prevedbo na tretjega, potem zagotovo obstaja prevedba, ki bo prvega prevedla na tretjega.)

D^* je NP-težek $\wedge D^* \leq^p D^\star \Rightarrow D^\star$ je NP-težek

D^* je NP-poln $\wedge D^* \leq^p D^\star \wedge D^\star \in NP \Rightarrow D^\star$ je NP-poln

- **P ≠ NP**

NPC je razred vseh **NP-polnih** problemov.

NPI je razred vseh **NP-vmesnih** problemov (problem v NP, ki ni niti v P niti v NPC).

Noben problem v NPC ali NPI nima zahtevnosti polinomskega časa.

Problemi v P so obvladljivi. Drugi računski problemi so neobvladljivi (za NPC in NPI ni jasno).

