Izračunljivost in Računska Zahtevnost

Izreki, dokazi in definicije

January 6, 2025

Contents

1 Osnovni matematični pojmi

Definicija: *Model računanja je* formalna definicija osnovnih pojmov in sestavin algoritmičnega računanja. Model računanja strogo definira:

- pojem algoritma,
- okolje, ki je potrebno za izvedbo algoritma,
- izvajanje algoritma v tem okolju.

2 Končni avtomati in regularni izrazi

2.1 Deterministični končni avtomat

Definicija: *Deterministični končni avtomat* (DKA) je peterka $(Q,\sum,\delta,q_0,F)$ kjer so:

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

Torej je, $\delta(q, a)$ stanje, v katerega DKA preide iz stanja q, če je prebral simbol a.
Opomba: δ je program DKA. Vsak DKA ima svoj specifični program δ.

Definicije:
DKA $M = (Q, \sum, \delta, q_0, F)$ sprejme besedo (niz) x, če je $\delta(q_0, x) = p$ za neki $p \in F$.

Jezik, sprejet z DKA M je množica L(M) vseh besed, ki jih sprejme M
$L(M) = \{x \in \sum^* | \delta(q_0, x) \in F\}$

Jezik L' je regularen (oz. je regularna množica),
če L' sprejme kak DKA (tj. če $\exists$ DKA M : L' = L(M)).

2.2 Nedeterminizem

Vprašanje: Če je dana vhodna beseda x = $a_1 a_2 \ldots a_n$, kdo bo ugotovil ali obstaja zaporedje prehodov, ki se konča v končnem stanju?
Odgovor: NKA sam!

Vprašanje: Kako bo NKA to ugotovil?
Odgovor: NKA ni realističen model računanja: predpostavljamo namreč, da NKA to vedno pravilno ugane. Lahko si mislimo, da ima NKA magično sposobnost, da med več možnimi prehodi vedno izbere pravega (če je med njimi kakšen tak), tj. tistega,ki ga vodi k sprejetju. Če pravega prehoda ni, NKA to magično ugotovi in se ustavi.
Zaradi te svoje magične lastnosti je NKA neuresničljiv, nerealističen model.

Vprašanje: Čemu koristi nerealistični model računanja NKA?
Odgovora:

- NKA in tudi drugi nedeterministični modeli računanja, ki jih bomo še spoznali, koristijo pri določanju spodnjih meja časa, potrebnega za rešitev računskega problema. Podlaga pri tem je naslednji razmislek:
  Če reševanje danega problema P z nedeterminističnim modelom računanja M zahteva T časa, potem bo reševanje problema P s kakršnokoli deterministično različico D modela M zahtevalo vsaj T časa (ker D nima magične sposobnosti pravilnega ugibanja).
- Pogosto je za dani problem P lažje sestaviti NKA (ali drug nedeterministični model M). Ko je M sestavljen, poskušamo iz M sestaviti ekvivalentno deterministično različico D (ekvivalentno v smislu, da tudi D reši problem P, čeprav v nekem drugem času).

2.2.1 Nedeterministični končni avtomat

Definicija: Nedeterministični končni avtomat (NKA) je peterka $(Q, \sum, \delta, q_0, F)$ kjer so:

- Q končna množica stanj,
- $\sum$ (končna) vhodna abeceda,
- $q_0 \in Q$ začetno stanje,
- $F \subseteq Q$ množic končnih stanj
- δ funkcija prehodov tj. $\delta : Q \times \sum \to 2^Q$
  Torej so v množici $\delta(q, a)$ natanko tista stanja, v katera NKA lahko preide iz stanja q, če je prebral simbol a.

Opomba: δ is je program NKA. Vsak NKA ima svoj specifični program δ.

Definicije:
NKA $M = (Q, \sum, \delta, q_0, F)$ sprejme besedo (niz) x,
če $\delta(q_0, x)$ vsebuje kako končno stanje $p \in F$ (tj. $\delta(q_0, x) \cap F \neq \emptyset$).

Jezik, sprejet z NKA M je množica $L(M)$ vseh besed, ki jih sprejme M:
$L(M) = \{x \in \sum^* \mid \delta(q_0, x)$ vsebuje stanje v $F\}$

2.2.2 Ekvivalentnost DKA in NKA

Vsak DKA je tudi (trivialen) NKA.

Izrek: Naj bo L jezik, ki ga sprejme dani NKA M. Potem obstaja DKA M' , ki sprejme L

Nepotrebno: $\mid Q_{M'} \mid \leq 2^{|Q_M|}$

2.2.3 Končni avtomati s tihimi (ε) prehodi (NKA_ε)

Definicija: NKA s tihimi prehodi (NKA_ε) je peterka $(Q, \sum, \delta, q_0, F)$ kjer so:

- Q končna množica stanj,
- $\sum$ vhodna abeceda,
- $q_0 \in Q$ začetno stanje,
- $F \subseteq Q$ množica končnih stanj
- δ funkcija prehodov, tj. $\delta : Q \times (\sum \cup \{\varepsilon\}) \to 2^Q$
  V množici, $\delta(q, a)$ so natanko tista stanja, kamor NKA lahko preide iz stanja q po povezavah, ki so označene z a ali ε.

NKA_ε $M = (Q, \sum, \delta, q_0, F)$ sprejme besedo (niz) x, če $\delta(q_0, x)$ vsebuje stanje v F.

Definicije:
Jezik, sprejet z NKA_ε $M = (Q, \sum, \delta, q_0, F)$ je množica L(M)vseh besed, ki jih sprejme M, torej
$L(M) = \{x \in \sum^* \mid \delta(q_0, x) = p \wedge p \in F\}$

Množico vseh stanj, dosegljivih iz stanja q samo s tihimi prehodi, imenujemo ε-Closure(q).

2.2.4 Ekvivalentnost NKA in NKAε

Izrek: Naj bo L jezik, ki ga sprejme dani NKA_ε M. Potem obstaja NKA M', ki sprejme L.

2.3 Regularni izrazi

2.3.1 Stik

Definicija: Naj bo $\sum$ abeceda ter L_1 in L_2 množici besed iz $\sum^*$. Stik L_1L_2 jezikov L_1 in L_2, je jezik
$L_1L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$
Besede jezika L_1L_2 dobimo tako, da besedi x iz L_1 pripnemo besedo y iz L_2, po vseh možnih x, y.

2.3.2 Kleeneovo in tranzitivno zaprtje

Definicija: Naj bo $L \subseteq \sum^*$. Definirajmo $L^0 = \{\varepsilon\}$ in $L^i = LL^{i-1}$ za $i \geq 1$.
Kleeneovo zaprtje L^* jezika L je jezik

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

tranzitivno zaprtje L^+ jezika L pa jezik

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

L^* je množica besed, nastalih s stikom končno mnogo (tudi nič) besed iz L.
L^+ je množica besed, nastalih s stikom končno mnogo (toda ne nič) besed iz L.
Opomba: L^+ vsebuje $\varepsilon \Leftrightarrow L$ vsebuje ε.

2.3.3 Regularni izraz

Definicija: Naj bo $\sum$ abeceda. Regularni izraz (r.i.) nad $\sum$ in jezik, ki ga r.i. predstavlja, sta definirana induktivno takole:

1. $\emptyset$ je r.i.; predstavlja jezik $\emptyset$;
2. ε je r.i.; predstavlja jezik $\{\varepsilon\}$;
3. Za vsak $a \in \sum$, je a r.i.; predstavlja jezik $\{a\}$;
4. Če sta r in s r.i. in predstavljata jezika R in S, potem:
   - $(r + s)$ je r.i.; predstavlja jezik $R \cup S$; (unija jezikov R in S)
   - (rs) je r.i.; predstavlja jezik RS; (stik jezikov R in S)
   - (r^*) je r.i.; predstavlja jezik R^*. (Kleeneovo zaprtje jezika R)

Opomba. Osnovni r.i. so definirani eksplicitno (1,2,3), vsi ostali pa induktivno (4). Pravimo, da so take definicije induktivne. V nadaljevanju jih bomo še srečali. Lastnosti tako definiranih objektov lahko dokazujemo z matematično indukcijo.

2.3.4 Dogovori

- Veliko oklepajev v r.i. postane nepotrebnih, če upoštevamo , da ima operacija * prednost pred operacijo stik, ta pa prednost pred +.
- Regularne izraze oblike rr^* krajše zapišemo z r^+.
- Regularne izraz oblike $rrr \ldots r$, kjer je staknjenih k r.i. r, okrajšamo z r^k.
- Običajno bomo jasno razlikovali med r.i. in jezikom, ki ga r.i.predstavlja. Zato bomo z $L(r)$ označili jezik, ki ga predstavlja r.i. r.

2.3.5 Ekvivalentnost končnih avtomatov in regularnih izrazov

Razred jezikov, sprejetih s končnimi avtomati, je identičen razredu jezikov, predstavljivih z regularnimi izrazi.

Izrek: Naj bo r poljuben regularni izraz. Potem obstaja NKA$_\varepsilon$ ki sprejme jezik $L(r)$.

Ideja dokaza: Z indukcijo po številu operatorjev v r.i. bomo dokazali, da lahko za poljuben r sestavimo NKA $_\varepsilon$ $M = (Q, \sum, \delta, q_0, \{f_0\})$ z enim končnim stanjem f_0 (in brez prehodov iz njega), da velja $L(M) = L(r)$.
Opomba: NKA$_\varepsilon$ z enim samim končnim stanjem nam bodo omogočili enostavno sestavljanje manjših NKA$_\varepsilon$ v večje. Zahteva po enem končnem stanju pa ne zmanjša splošnosti izreka (Zakaj? Poljuben NKA lahko pretvorimo v ekvivalentni NKA$_\varepsilon$ z enim končim stanjem.)

Izrek: Naj bo M poljuben DKA.
Tedaj obstaja regularni izraz r, ki predstavlja jezik $L(M)$.
Ideja dokaza:

- Jezik $L(M)$ ki ga sprejema M, zapišemo kot unijo končno mnogo množic.
- Vsaka množica ustreza nekemu končnemu stanju avtomata M in vsebuje natanko tiste besede, ki privedejo M iz začetnega stanja v to končno stanje

- Te množice sestavimo na induktiven način (tj. večje izrazimo z manjšimi), za vsako pa tudi induktivno sestavljamo regularni izraz, ki jo predstavlja.
- Regularni izraz, ki predstavlja jezik $L(M)$, je potem vsota regularnih izrazov, ki predstavljajo induktivno definirane množice.

3 Lastnosti regularnih jezikov

3.1 Lema o napihovanju (za regularne jezike)

Naj bo L regularni jezik.
Potem obstaja konstanta n (odvisna samo od L), da velja naslednje: če je z poljubna beseda z lastnostjo

$z \in L$ in $|z| \geq n$ (obstaja beseda z, ki je daljša ali enaka n)

potem obstajajo besede u, v, w da velja

$z = uvw$, (sestavljena iz u,v,w)

$|uv| \leq n$,

$|v| \geq 1$, (srednja beseda dolga vsaj 1)

$\forall i \geq 0 : uv^i w \in L$. (to besedo lahko poljubno ponavljamo)

Kakšen je n? Dokaz leme bo razkril, da je n lahko enak številu stanj v DKA, ki sprejme L. Kakšna je beseda v? Iz zgornjih lastnosti u, v, w enostavno sledi, da je $1 \leq |v| \leq n$

Intuitivno:
Če je dana poljubna zadosti dolga beseda z, ki jo sprejme neki končni avtomat, je v z blizu njenega začetka neka neprazna in ne zelo dolga podbeseda v, ki jo znotraj z lahko ponovimo poljubno (končno) mnogokrat, a bo končni avtomat napihnjeno besedo $z' = uvvv...vw$ tudi sprejel.

Formalno:
L *regularen* $\implies (\exists n)(\forall z)(z \in L \wedge |z| \geq n \Rightarrow (\exists u, v, w)(z = uvw \wedge |uv| \leq n \wedge |v| \geq 1 \wedge (\forall i \geq 0)uv^i w \in L))$

Uporaba leme o napihovanju:
Dokazovanje, da neki jezik ni regularen.
$(\exists z)(\forall u, v, w)(\exists i \geq 0)uv^i w \notin L \Rightarrow L$ ni regularen (za 'dobre' n,z,u,v,w)

Če za dani L dokažemo, da leva stran '⇒' velja, potem L ni regularen jezik.

Posledica: Obstajajo formalni jeziki, ki niso regularni. To pomeni, da teh jezikov ne sprejme noben končni avtomat! Zato bomo potrebovali močnejši model računanja, ki bo sposoben sprejeti te jezike.

3.2 Zaprtost za operacije unija, stik, Kleeneovo zaprtje

Izrek: Razred regularnih jezikov je zaprt za operacije unija, stik in Kleeneovo zaprtje.
Intuitivno: Unija $L_1 \cup L_2$ in stik $L_1 L_2$ poljubnih regularnih jezikov L_1, L_2 sta tudi regularna jezika, in Kleeneovo zaprtje L^* poljubnega regularnega jezika L je tudi regularni jezik.

3.3 Zaprtost za operaciji komplement in presek

Izrek: Razred regularnih jezikov je zaprt za operaciji komplement in presek.
Intuitivno: Komplement $\sum^* - L$ poljubnega regularnega jezika L je tudi regularen jezik.
Presek $L_1 \cap L_2$ poljubnih regularnih jezikov L_1, L_2 je tudi regularen jezik.

3.4 Closure under substitution and homomorphism

Let Σ, Δ be alphabets. A *substitution* is a function f that maps each symbol of Σ to a language over Δ; i.e. $f(a) \subseteq \Delta^*$ for each a $\in \Sigma$. We extend f to words in Σ^* by defining $f(\varepsilon) = \varepsilon$ and $f(wa) = f(w)f(a)$; and then to languages by defining

$$f(L) = \bigcup_{x \in L} f(x)$$

The definition of substitution says nothing about the kind of the set L and the sets $f(a)$, $a \in \Sigma$. What if we additionally require that L and all $f(a)$, $a \in \Sigma$ are regular? Is then $f(L)$ regular too?

A homomorphism is a substitution h such that $h(a)$ contains a *single word* for each $a \in \Sigma$. We extend h to words and languages as in the case of substitution. The *inverse homomorphic image* of a word w is the set $h^{-1}(w) = \{x \mid h(x) = w\}$ and of a language L is the set $h^{-1}(L) = \{x \mid h(x) \in L\}$.

The class of regular sets is closed under substitution, homomorphism and inverse homomorphism.

Proof idea:

- (*substitution*) Let L and all $f(a)$, $a \in \Sigma$ be regular sets. Let L be denoted by r.e. r and $f(a)$ by r_a.
  Idea: replace each occurence of a by r by r_a. Then prove that the resulting r.e. r' denotes $f(L)$.
  (Use induction on the number of operators in r'.)
- (*homomorphism*) Closure under homomorphism follows directly from closure under substitution
  (because every homomorphism is by definition a (special) substitution)
- (*inverse homomorphism*) Let L be regular and h a homomorphism. We want to prove that $h^{-1}(L)$ is regular. Let M be DFA accepting L. We want to construct a DFA M' such that M' accepts $h^{-1}(L)$ *iff* M accepts L.
  Idea: construct M' so that when M' reads $a \in \Delta$, it simulates M on $h^{-1}(L)$.

3.5 Zaprtost za operacijo kvocient

Kvocient jezikov L_1 in L_2 je jezik L_1/L_2 definiran takole:
$L_1/L_2 = \{x \mid \exists y \in L_2 : xy \in L_1\}$.
Intuitivno: L_1/L_2 vsebuje predpone tistih besed jezika L_1, katerih pripadajoče pripone so besede v jeziku L_2.

Razred regularnih jezikov je zaprt za operacijo kvocient s poljubnim jezikom (deliteljem).

4 Odločitveni problemi in algoritmi za regularne jezike

Računski problemi, ki sprašujejo po odgovoru DA/NE, so odločitveni problemi, algoritmi, ki rešujejo odločitvene probleme, pa odločitveni algoritmi.

4.1 Praznost in končnost regularnih jezikov

Izrek: Jezik $L(M)$ ki ga sprejme končni avtomat M z n stanji, je:

1. *neprazen* $\Leftrightarrow$ M sprejme neko besedo dolžine l, kjer je $l < n$.
2. *neskončen* $\Leftrightarrow$ M sprejme neko besedo dolžine l, kjer je $n \leq l < 2n$

Algoritem sistematično generira besede dolžin l v intervalu $[0, n-1]$ oz. $[n, 2n-1]$; po vsaki generirani besedi preveri, ali bi jo M sprejel.
Za poljuben vhod M se algoritma ustavita in vrneta odgovor DA ali NE.

4.2 Ekvivalentnost končnih avtomatov

Definicija: Končna avtomata M_1 in M_2 sta ekvivalentna če sprejmeta isti jezik, torej če je $L(M_1) = L(M_2)$.
Izrek: Obstaja algoritem, ki odloči, ali sta končna avtomata $M1$ in $M2$ ekvivalentna.
Dokaz:
Naj bosta M_1 in M_2 končna avtomata in $L_1 = L(M_1)$ ter $L_2 = L(M_2)$. Definirajmo jezik L_3 takole:

$$L_3 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

L_3 je regularen jezik (zaradi lastnosti zaprtosti), zato ga sprejme neki končni avtomat M_3. Za M_3 pa lahko dokažemo naslednje

$$M_3 \text{ sprejme besedo} \Leftrightarrow L_1 \neq L_2.$$

Torej moramo samo še preveriti, ali je M_3 neprazen regularen jezik.

5 Myhill-Nerodejev izrek in najmanjši ekvivalentni končni avtomati

5.1 Relacija R_L

DEFINICIJA: Bodi $L \subseteq \Sigma^*$ poljuben jezik. Definirajmo relacijo R_L na Σ^* takole:

$$xR_Ly \Leftrightarrow (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L).$$

Besedi $x, y \in \Sigma^*$ sta v relaciji R_L natanko tedaj, ko sta njuni poljubni razširitvi xz, yz bodisi obe v L bodisi izven L. R_L je *ekvivalenčna relacija*. Torej, R_L razdeli L v *ekvivalenčne razrede*. Njihovemu številu pravimo indeks relacije R_L. Ta je lahko končen ali neskončen.

5.2 Relacija R_M

Definicija: Bodi $M = (Q, \Sigma, \delta, q_0, F)$ DKA. Definirajmo relacijo R_M na Σ^* takole:

$$xR_My \Leftrightarrow \delta(q_0, x) = \delta(q_0, y).$$

Besedi $x, y \in \Sigma^*$ sta v relaciji R_M natanko tedaj, ko privedeta M iz začetnega stanja q_0 v isto stanje q. R_M je *ekvivalenčna relacija*. Razdeli Σ^* v ekvivalenčne razrede, po en razred za vsako stanje q ki je dosegljivo iz q_0. Index relacije R_M je končen (ker je Q končna množica).Jezik $L(M)$ je (očitno) unija ekvivalenčnih razredov relacije R_M , ki ustrezajo (dosegljivim) končnim stanjem $q \in F$.
R_M je desno invariantna, ker zanjo velja tole:

$$xR_My \Rightarrow \forall z \in \Sigma^* : xzR_Myz$$

5.3 Myhill-Nerodejev izrek

Izrek: Naslednje izjave so ekvivalentne:

1. $L \subseteq \Sigma^*$ je regularen jezik.
2. R_L ima končen indeks.
3. L je unija (nekaterih) ekvivalenčnih razredov (neke, tj. R_M) desno invariantne ekvivalenčne relacije s končnim indeksom.

Uporaba: Izrek je koristen, ko za dani L uspemo dokazati eno (vseeno katero) od zgornjih izjav. Potem takoj (brez dokaza) držita tudi ostali dve izjavi, in tako razkrijeta nove, dotlej nepoznane lastnosti jezika L.

Posledica Myhill-Nerodejevega izreka je, da ima vsak regularni jezik (v bistvu en sam) najmanjši deterministični končni avtomat, ki sprejema ta jezik.

5.4 Najmanjši DKA

Izrek: Najmanjši DKA, ki sprejme dani regularni jezik L, je enolično določen do izomorfnosti (tj.do preimenovanja stanj) natančno.

Ideja dokaza:

- Bodi L regularen jezik. Po Myhill-Nerodejevem izreku ima R_L končno mnogo ekvivalenčnih razredov. Označimo z $[x]$ ekv.razred, kjer je $x \in \Sigma^*$. Množica vseh ekvivalenčnih razredov relacije R_L je tedaj $\{[x] \mid x \in \Sigma^*\}$.

- Konstruirajmo DKA $M = (Q, \Sigma, \delta, q_0, F)$ na sledeči način:

  $Q := \{[x] \mid x \in \Sigma^*\}$; (stanja predstavljajo ekv. razrede relacije R_L)

  $\delta([x], a) := [xa]$, za $a \in \Sigma$

  $q_0 := [\varepsilon]$ (začetno stanje predstavlja ekv. razred $[\varepsilon]$, kjer je prazna beseda)

  $F := \{[x] \mid x \in L\}$ (končna stanje predstavljajo ekv. razredi, kjer je sprejete besede)

- *Opombe:* $\delta(q_0, w) = \delta(q_0, a_1 a_2 \dots a_n) = [a_1 a_2 \dots a_n] = [w]$.
  Torej M sprejme w natanko tedaj, ko $[w] \in F$. To pa pomeni, da M sprejme L. Iz dokaza Myhill-Nerodejevega izreka pa sledi, da je tako konstruirani M najmanjši končni avtomat, ki sprejme L.

6 Kontekstno-neodvisne gramatike in jeziki

6.1 Kontekstno-neodvisna gramatika

Definicija: Kontekstno-neodvisna gramatika (KNG) je četverka $G = (V, T, P, S)$, kjer so:

- V končna množica vmesnih simbolov,
- T končna množica končnih simbolov,
- P končna množica produkcij oblike $\cdots \to \ldots$ kjer je,
  "$\ldots$" na levi strani znaka "$\to$" poljuben vmesni simbol iz V in
  "$\ldots$" na desni strani znaka "$\to$" poljubna beseda iz jezika $(V \cup T)^*$;
- S je poseben vmesni simbol, imenovan začetni simbol.

Definicije:
Naj bo $A \to \beta$ produkcija in $\alpha, \gamma \in (V \cup T)^*$ poljubni besedi.

- Produkcijo $A \to \beta$ uporabimo na besedi tako, da v tej besedi zamenjamo A z β. Takrat rečemo, da smo iz $\alpha A \gamma$ neposredno izpeljali $\alpha\beta\gamma$ z uporabo produkcije $A \to \beta$ (oziroma, da smo $\alpha\beta\gamma$ razvili neposredno iz $\alpha A \gamma$ z uporabo produkcije $A \to \beta$).
- Besedi α_i in α_j sta v relaciji $\alpha_i \ {}_G\Rightarrow \alpha_j$, če je α_i neposredno izpeljiv iz α_i z uporabo neke produkcije gramatike G.
- Naj bodo $\alpha_1, \alpha_2, \ldots, \alpha_m \in (V \cup T)^*, m \geq 1$, nizi.
  Če velja $\alpha_1 \ {}_G\Rightarrow \alpha_2 \wedge \alpha_2 \ {}_G\Rightarrow \alpha_3 \wedge \cdots \wedge \alpha_{m-1} \ {}_G\Rightarrow^* \alpha_m$
  potem to zapišemo z $\alpha_1 \ {}_G\Rightarrow^* \alpha_m$ in rečemo, da iz α_1 izpeljemo α_m po gramatiki G (oz. da je α_m izpeljiv iz α_1 po gramatiki G).
  Opomba: Relacija ${}_G\Rightarrow^*$ je refleksivno in tranzitivno zaprtje relacije ${}_G\Rightarrow$.

6.2 Jezik kontekstno-neodvisne gramatike

Definicija: Jezik kontekstno-neodvisne gramatike $G = (V, T, P, S)$ je

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

$L(G)$ je množica vseh končnih nizov, ki so izpeljivi iz začetnega simbola S po gramatiki G.

Definicije:
Jezik L je kontekstno-neodvisen (KNJ), če je $L = L(G)$ za neko (katerokoli) KNG G

Niz $\alpha \in (V \cup T)^*$ imenujemo stavčna oblika, če velja $S \ {}_G\Rightarrow^* \alpha$.

Gramatiki G_1 in G_2 sta ekvivalentni, če velja $L(G_1) = L(G_2)$.

6.3 Drevesa izpeljav

Definicija: Bodi $G = (V, T, P, S)$ poljubna KNG. Drevo D_G je drevo izpeljav po gramatiki G, če velja naslednje:

1. Vsaka točka $v \in D_G$ je označena z vmesnim ali končnim simbolom $V \cup T \cup \{\varepsilon\}$.
2. Koren drevesa $v \in D_G$ je označen z začetnim simbolom S.
3. Če je $v \in D_G$ notranja točka, je označena z vmesnim simbolom ($\in V$).
4. Če je $v \in D_G$ označena z vmesnim simbolom, npr. A, in so vsi njeni sinovi od leve v desno označeni z $X_1, X_2, \ldots, X_k$, potem je $A \to X_1 X_2 \ldots X_k$ produkcija ($\in P$).
5. Če je $v \in D_G$ označena z ε, potem je v list v D_G in je edini sin svojega očeta.

Opomba: Če je $v \in D_G$ označena s končnim simbolom ($\in T$), je v list v D_G (vendar ne nujno edinček).

6.4 Rob drevesa

Definicija: Oznake listov, izpisane med *premim obhodom* drevesa izpeljav D_G, tvorijo rob drevesa D.

6.5 Poddrevo

Definicija: Poddrevo drevesa izpeljav D_G v točki v je drevo s korenom v točki v ter vsemi točkami in povezavami, ki v D_G sledijo točki v. Če je v označena z $A \in V$, rečemo, da je poddrevo v tej točki A-drevo.
Poddrevo ima lastnosti drevesa izpeljav, le da oznaka v njegovem korenu ni nujno začetni simbol S gramatike.

6.5.1 Zveza med izpeljivostjo (po G) in robovi (v D_G)

Izrek: Naj bo $G = (V, T, P, S)$ poljubna KNG. Tedaj velja:
$S \ {}_G \Rightarrow^* \alpha$ če in samo če obstaja drevo izpeljav D_G z robom α.
Ideja dokaza: Indukcija po številu notranjih točk drevesa D_G.

6.5.2 Skrajno leva (desna) izpeljava (po G)

Definicija: Izpeljava stavčne oblike po gramatiki G je skrajno leva, če na vsakem koraku uporabimo produkcijo, ki razvije skrajno levi vmesni simbol trenutne stavčne oblike.
Izpeljava stavčne oblike po gramatiki G je skrajno desna, če na vsakem koraku uporabimo produkcijo, ki razvije skrajno desni vmesni simbol trenutne stavčne oblike.

6.5.3 Dvoumnost

Definicija: KNG G je dvoumna, če obstaja $w \in L(G)$, ki ima več kot eno drevo izpeljav.
Ekvivalenta definicija: KNG G je dvoumna, če ima kaka beseda $w \in L(G)$ več kot eno skrajno levo (desno) izpeljavo po G.

Definicija: KNJ L je bistveno dvoumen, če je vsaka KNG za jezik L dvoumna.

6.6 Normalne oblike kontekstno-neodvisnih gramatik

Obstajajo načini, s katerimi lahko omejimo obliko produkcij KNG, ne da bi pri tem zmanjšali "moč" KNJ. Natančneje: če je L neprazen KNJ, potem je L jezik (neke) gramatike G ($L = L(G)$), ki ima naslednje lastnosti:

- Vsak vmesni in končni simbol gramatike G nastopa v izpeljavi vsaj ene besede $w \in L$.
- V gramatiki G ni produkcij oblike $A \to B$.
- Če $\varepsilon \notin L$, potem v G ni produkcij oblike $A \to \varepsilon$.
- Če $\varepsilon \notin L$, potem je v G
  - vsaka produkcija oblike $A \to BC$ ali oblike $A \to b$
    (normalna oblika Chomskega) kjer sta A, B, C poljubna vmesna simbola in b poljuben končni symbol

  ali

  - vsaka produkcija oblike $A \to b\gamma$
    (normalna oblika Greibachove) kjer je γ poljuben niz vmesnih simbolov ($\gamma \in V^*$) in b poljuben končni simbol ($b \in T$)).

6.6.1 Odstranitev odvečnih simbolov

Definicija: Bodi $G = (V, T, P, S)$ KNG. Simbol X je potreben, če obstaja izpeljava $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ kjer sta α, β, in $w \in T^*$. Če X ni potreben, je odvečen.
Lemi:

- Lema 1. Poljubni KNG $G = (V, T, P, S)$ kjer $L(G) \neq \emptyset$, lahko efektivno priredimo ekvivalentno KNG $G' = (V', T, P', S)$ kjer za vsak vmesni simbol $A \in V'$ obstaja $w \in T^*$ da velja $A \Rightarrow^* w$.
  (tj. iz vsakega vmesnega simbola A se da izpeljati po G' nek končni niz w).
- Lema 2. Poljubni KNG $G' = (V', T, P', S)$, lahko efektivno priredimo ekvivalentno KNG $G'' = (V'', T, P'', S)$ kjer za vsak (vmesni ali končni) simbol $X \in V'' \cup T$ obstajata $\alpha, \beta \in (V'' \cup T)^*$ da velja $A \Rightarrow^* \alpha X \beta$
  (tj., vsak vmesni ali končni simbol X se pojavi v neki izpeljavi po G'' iz S).

Uporaba Leme 1 in nato Leme 2 priredi KNG G ekvivalentno G'' brez odvečnih simbolov (Uporaba najprej leme 2 in nato 1 v nekaterih primerih ne odstrani vseh odvečnih simbolov.)

Izrek: Vsak neprazen KNJ generira neka KNG, ki je brez odvečnih simbolov.

6.6.2 Odstranitev ε-produkcij

Definicija: ε-produkcija je produkcija oblike $A \to \varepsilon$.
Če je $\varepsilon \in L(G)$, očitno ne moremo iz G odstraniti vseh ε-produkcij (ker potem ε ne bi bil več v $L(S)$).
Če pa $\varepsilon \notin L(G)$, lahko odstranimo vse ε-produkcije iz G.

Izrek: Če je $L = L(G)$ za neko KNG $G = (V, T, P, S)$, potem jezik $L - \{\varepsilon\}$ generira neka KNG G' ki je brez ε-produkcij. (in brez odvečnih simbolov).

Ideja dokaza:

- Za vsak vmesni simbol $A \in V$ ugotovi, ali $A \Rightarrow^* \varepsilon$. Če velja, rečemo, da je simbol A uničljiv. Naj bo U množica vseh uničljivih vmesnih simbolov gramatike G.
- Vsako produkcijo $B \to X_1X_2 \dots X_n$ nadomesti z novimi produkcijami, ki nastanejo, ko iz $X_1X_2 \dots X_n$ izločamo simbole iz podmnožic množice U; med nove produkcije pa ne štej $B \to \varepsilon$ (tudi če so vsi $X_1X_2 \dots X_n$ uničljivi).

6.6.3 Odstranitev enotskih produkcij

Definicija: Enotska produkcija je produkcija oblike $A \to B$.
Pozor: $A \to a$, kjer je a končni simbol, ni enotska produkcija, prav tako tudi $A \to \varepsilon$ ni enotska produkcija.

Izrek: Vsak KNJ brez ε lahko generira KNG, ki je brez enotskih produkcij, in brez ε-produkcij ter odvečnih simbolov.

6.6.4 Normalna oblika Chomskega

Izrek: Vsak KNJ, ki ne vsebuje ε, se da generirati s KNG, katere produkcije so oblike

$$A \to BC$$
ali
$$A \to a$$

Tu so A, B, C poljubni vmesni simboli in a poljuben končni simbol gramatike.

TLDR "Ideja dokaza" p.143:

> Vhodna KNG naj bo brez odvečnih simbolov, enotskih produkcij in ε-produkcij
>
> Zamenjaj vse končne simbole tako, da so oblike: $A \to a$ (en vhod $\to$ en izhod)
>
> Vse produkcije z več kot dvema (≥ 3) simboloma v manjše, kjer je maksimalno število simbolov v produkciji 2 (npr: $A \to BC$).

6.6.5 Normalna oblika Greibachove

Izrek: Vsak KNJ, ki ne vsebuje ε, se da generirati s KNG, katere produkcije so oblike:

$$A \to b\gamma$$

Tu je $A \in V$ poljuben vmesni simbol, $b \in T$ poljuben končni simbol in $\gamma \in V^*$ poljuben niz vmesnih simbolov. *"dokaz" p.145*

6.6.6 Bistveno dvoumni kontekstno-neodvisni jeziki

Definicija: KNJ L je bistveno dvoumen, če: $(L = L(G) \wedge G$ je KNG$) \Rightarrow G$ je dvoumna.
Ekvivalenta definicija: KNJ L je dvoumen, če je vsaka KNG G, ki ga opisuje, dvoumna.
Izrek (in primer dvoumnosti): KNJ $L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \bigcup \{a^n b^m c^m d^n \mid n, m \geq 1\}$ je bistveno dvoumen.

7 Skladovni avtomati

Definicija: Skladovni avtomat (SA) je sedmerka $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$,
kjer so

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

Opomba: δ je program SA. Vsak SA ima svoj specifični δ.

7.1 Poteze skladovnega avtomata

Prehod:

- Navadna poveza: $\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$ pomeni, da SA v stanju q, z vhodnim simbolom a v oknu in skladovnim simbolom Z na vrhu sklada, nedeterministično izbere i $(1 \leq i \leq m)$ in stori naslednje: preide v stanje p_i, zamenja Z z nizom γ_i in premakne okno na naslednjo calico traku.
- Tiha poteza: $\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$ pomeni, da SA v stanju q, s skladovnim simbolom Z na vrhu sklada in neodvisno od vhodnega simbola v oknu, nedeterministično izbere i $(1 \leq i \leq m)$ in stori naslednje: preide v stanje p_i in zamenja Z z nizom γ_i ter okna ne premakne naprej (ostane na istem mestu).

Dogovori: skrajno levi simbol v γ_i je na vrhu sklada. Vhodne simbole bomo označevali z $a, b, c, \ldots$, nize vhodnih simbolov z $u, v, w, \ldots$, skladovne simbole $B, C, \ldots$, nize skladovnih simbolov pa z α, β, γ

7.2 Trenutni opisi (TO)

Definicije: Trenutni opis (TO) je trojka (q, w, γ), kjer je q stanje, w niz vhodnih simbolov in γ niz skladovnih simbolov

- Če je $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ SA, rečemo, da TO $(q, ax, Z\beta)$ neposredno preide v $TO(p_i, x, \gamma_i\beta)$, kar zapišemo kot:
$$(q, ax, Z\beta) \vdash (p_i, x, \gamma_i\beta),$$
če $\delta(q, a, Z)$ vsebuje (p_i, γ_i). Tu je namesto vhodnega simbola a lahko tudi ε.

- Refleksivno-tranzitivno zaprtje relacije $_M \vdash$ je relacija $_M \vdash^*$ Če velja $I \ _M \vdash^* \ J$ rečemo, da TO I preide v TO J. Če I prede v J v k neposrednih prehodih, to pišemo kot $I_M \vdash^k J$.

Kadar bo jasno, kateri SA M je mišljen, bomo indeks M opustili.

7.3 Jeziki, ki jih sprejemajo SA

Definicija: Vsak SA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ sprejme dva jezika:

- $L(M)$, jezik sprejet s končnim stanjem, ki je definiran kot
  $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \gamma)$ kjer je $p \in F$ in $\gamma \in \Gamma^*\}$
- $N(M)$, jezik sprejet s praznim skladom, ki je definiran kot
  $N(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)$ kjer je $p \in Q\}$.

$L(M)$ vsebuje besedo w, če se M po branju w lahko (nedeterminizem!) znajde v kakem končnem stanju.
$N(M)$ vsebuje besedo w, če ima M po branju w lahko (nedeterminizem!) prazen sklad.
Opomba: Pri sprejemanju s praznim skladom so končna stanja nepomembna, zato je takrat lahko $F = \emptyset$.

7.4 Deterministični skladovni avtomati

Definicija: SA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je deterministično, če za δ velja:
Za vsak par $q \in Q$ in $Z \in \Gamma$:

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

Kaj to pomeni?
Pogoj 1 preprečuje izbiranje med navadnimi in tihimi potezami.
Pogoj 2 preprečuje izbiranje pri navadnih potezah in izbiranje pri tihih potezah.
Opomba: Ker je SA po definiciji nedeterminističen, bomo deterministične SA označevali z DSA (deterministični skladovni avtomati).

7.5 Skladovni avtomati in kontekstno-neodvisni jeziki

7.5.1 Ekvivalenca sprejemanja s končnim stanjem in praznim skladom

Izrek: Če je $L = L(M_2)$ za nek SA M_2, potem $L = N(M_1)$ za nek (drugi) SA M_1.
Ideja dokaza: Naj bo $L = L(M_2)$ poljuben jezik, Konstruiramo SA M_1 ki simulira M_2 pri tem pa še zbriše svoj sklad vsakokrat, ko bi M_2 vstopil v kako končno stanje. Zato je $L = N(M_1)$

Izrek: Če je $L = N(M_1)$ za nek SA M_1, potem $L = L(M_2)$ za nek (drugi) SA M_2.
Ideja dokaza: Naj bo $L = L(M_1)$ poljuben jezik, Konstruiramo SA M_2 ki simulira M_1 pri tem pa še vstopi v končno stanje vsakokrat, ko bi M_1 izpranil svoj sklad. Zato je $L = L(M_2)$

Posledica: Razred jezikov, ki jih sprejmejo SA s končnim stanjem, je enak razredu jezikov, ki jih sprejemejo SA s praznim skladom.

7.5.2 Zveza med skladovnimi avtomati in kontekstno-neodvisnimi jeziki

Izrek: Če je L KNJ, potem obstaja SA M, da je $L = N(M)$.
Ideja dokaza: Naj bo L poljuben KNJ. L lahko generira KNG G v Greibachovi normalni obliki. Konstruiramo SA M ki simulira skrajno leve izpeljave po G. (M naj sprejema s praznim skladom) Sledi, da je $L = N(M)$

Izrek: Če je $L = N(M)$ za neki SA M, potem je L KNJ.
Ideja dokaza: Naj bo M poljuben SA. Konstruiramo KNG G tako, da skrajno leva izpeljava besede x simulira (opisuje) delovanje SA M na vhodni besedi x. Sledi, da je $L = L(G)$, torej je L KNJ.

Posledica: Razred vseh jezikov, sprejetih s SA, je enak razredu vseh KNJ.

7.5.3 Deterministični in nedeterministični SA

DSA je po sposobnosti sprejemanja šibkejši od SA.

$\{ww^R \mid w \in (0+1)^*\}$ (jezik palindromov) sprejme neki SA in noben DSA.

8 Lastnosti kontekstno-neodvisnih jezikov

8.1 Lema o napihovanju za KNJ

Naj bo L KNJ. Potem obstaja konstanta n (odvisna samo od L) da velja naslednje:
če je z poljubna beseda z lastnostjo

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

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

$$\begin{aligned} & z = uvwxy \wedge \\ \wedge\ & |vwx| \leq n \ \wedge \\ \wedge\ & |vx| \geq 1 \quad \wedge \\ \wedge\ & \forall i \geq 0 : uv^i wx^i y \in L \end{aligned}$$

Intuitivno: Če je L KNJ in $z \in L$ zadosti dolga beseda ($|z| \geq n$), ima z dve podbesedi v in x, ki sta si blizu ($|vwx| \leq n$) in ne obe prazni ($|vx| \geq 1$), ki ju lahko (obe enakokrat) končno mnogokrat ponovimo, a bo dobljena napihnjena beseda še vedno v jeziku L.

Formalno:
L je KNJ $\implies$ $(\exists n)(\forall z)(z \in L \wedge |z| \geq n \Rightarrow (\exists u, v, w, x, y)(z = uvwxy \wedge |vwx| \leq n \wedge |vx| \geq 1 \wedge (\forall i \geq 0) uv^i wx^i y \in L))$

Uporaba leme o napihovanju:
Dokazovanje, da neki jezik ni kontekstno-neodvisen.
$(\exists z)(\forall u, v, w, x, y)(\exists i \geq 0) uv^i wx^i y \notin L \Rightarrow L$ ni kontekstno-neodvisen (za 'dobre' n,z,u,v,w,x,y)

Obstajajo formalni jeziki, ki niso KNJ!! Zato bomo potrebovali močnejši model računanja, ki bo sposoben sprejeti te jezike.

8.2 Lastnosti zaprtosti za KNJ

Izrek: Razred KNJ je zaprt za operacije

- unija,
- stik,
- Kleeneovo zaprtje,
- substitucija (in homomorizem),
- inverzni homomorfizem

Izrek: Razred KNJ NI zaprt za operacije

- presek (razen če: izrek spodaj),
- komplement

Izrek: Če je L KNJ in R regularen jezik, potem je $L \cap R$ KNJ.

8.3 Odločitveni problemi in algoritmi za KNJ

8.3.1 Praznost in končnost KNJ

Izrek: Obstajata odločitvena algoritma za ugotavljanje, ali je KNJ:

1. prazen;
2. končen.

Ideja dokaza: Naj bo $G = (V, T, P, S)$ KNG

- Za ugotavljanje praznosti jezika $L(G)$ moramo ugotoviti, ali S generira vsaj en niz končnih simbolov.
- Za ugotavljanje končnosti jezika $L(G)$ konstruiramo KNG $G' = (V', T, P', S)$ v normalni obliki Chomskega brez odvečnih simbolov, ki generira jezik $L(G) - \{\varepsilon\}$. (Seveda: $L(G')$ je končen natanko tedaj, ko je $L(G)$ končen.) Nato narišemo usmerjen graf, kjer točke predstavljajo vmesne simbole iz V', iz točke A v točko B pa obstaja usmerjena povezava, če v P' obstaja produkcija oblike $A \to BC$ ali $A \to CB$ (za poljuben vmesni simbol C). Potem velja: $L(G')$ je končen natanko tedaj, ko je graf acikličen. Nato preverimo, ali je graf acikličen.

8.3.2 Pripadnost KNJ

Problem pripadnosti (za KNG) je vprašanje

"Ali je $x \in L(G)$ kjer je $G = (V, T, P, S)$ dana KNG in $x \in T^*$ dana beseda?"

(časovno neučinkovit) Algoritem:

1. Pretvori G v normalno obliko Greibachove G'. (Opomba: $L(G') = L(G) – \{\varepsilon\}$)
2. Če je $x = \varepsilon$ potem preveri ali $S \underset{G}{\Rightarrow}^* \varepsilon$
   sicer: sistematično pregleduj eno za drugo skrajno leve izpeljave besed dolžine $|x|$, dokler bodisi ne najdeš izpeljave besede x ali pa si neuspešno pregledal vse možne izpeljave.

Obstaja učinkovitejši algoritem, t.i. algoritem CYK (Cocke-Younger-Kasami) ki,

- je razvit z metodo dinamičnega programiranja
- zahteva $O(n^3)$ časa. (n = $|x|$)
- (informativno) opisan na: *p.182-183*

9 Turingov stroj

Definicija: (algoritem intuitivno) Algoritem za reševanje problema je končna množica ukazov, ki vodijo izvajalca, da v končnem številu korakov dobi iz vhodnih podatkov rešitev problema.

Zgodovina: *p.188*

Definicija: (model računanja) Model računanja formalno karakterizira osnovne pojme algoritmičnega računanja, tj. algoritem, njegovo okolje in njegovo izvajanje v tem okolju.

Nekaj modelov računanja:

- μ-rekurzivne funkcije
- splošne rekurzivne funkcije
- λ-račun
- Turingov stroj
- Postov stroj
- Markovski algoritmi

Večina raziskovalcev se je ogrela za Turingov stroj, ker je z najbolj popolno in razumljivo utemeljitvijo definiral osnovne pojme algoritmičnega računanja.

Vendar: našteti modeli so ekvivalentni, torej kar zmore izračunati eden, zmorejo tudi ostali.

9.1 Teza o izračunljivosti

Teza o izračunljivosti (Church-Turingova teza): Intuitivni osnovni pojmi algoritmičnega računanja so ustrezno formalizirani (strogo definirani) takole:

- intuitivni pojem "algoritma" formalizira Turing program;
- intuitivni pojem "računanja" formalizira izvajanje Turingovega programa;
- intuitivni pojem "izračunljive funkcije" formalizira funkcija, izračunljiva s Turingovim programom.

Pomen teze: Teza o izračunljivosti je vzpostavila most (zvezo) med intuitivnim razumevanjem osnovnih pojmov "algoritem", "računanje" in "izračunljivost" in matematičnim, formalno definiranim razumevanjem teh pojmov s pomočjo modelov računanja. Teza je odprla vrata matematični obravnavi intuitivnih osnovnih pojmov računanja.

9.2 Turingov stroj

Definicija. (Turingov stroj) Turingov stroj (osnovna različica) ima naslednje enote: nadzorno enoto, v kateri je Turingov program; trak, ki ga sestavljajo celice; premično okno na traku, ki je povezano z nadzorno enoto.
V splošnem: TS je sedmerka $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$, kjer so:

- Q končna množica stanj,
- Σ vhodna abeceda,
- Γ tračna abeceda,
- δ funkcija prehodov, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$
- q_1 začetno stanje ($q_1 \in Q$),
- $\sqcup$ prazni simbol,
- F množica končnih stanj ($F \subseteq Q$).

Podrobno po enotah:

- Trak služi pisanju in branju vhodnih podatkov ter vmesnih in končnih rezultatov. Razdeljen je na enake celice in je potencialno neskončen v eno (desno) smer; tj. vsakokrat, ko je to potrebno, se (samodejno) podaljša s končno mnogo celicami.)

  Vsaka celica vsebuje nek tračni simbol iz tračne abecede $\Gamma = \{z_1, \ldots, z_t\}, t \geq 3$. Simbol z_t je poseben: označuje, da je celica prazna. Običajno ga pišemo kot $\sqcup$ in imenujemo simbol za prazen prostor. Poleg $\sqcup$ sta v tračni abecedi še vsaj dva simbola: 0 and 1. (Ne da bi se s tem omejili, se dogovorimo, da je $z_1 = 0$ in $z_2 = 1$.)

Vhodni podatki so zapisani v vhodni besedi; ta je sestavljena iz simbolov vhodne abecede Σ, za katero velja $\{0,1\} \subseteq \Sigma \subseteq \Gamma - \{\sqcup\}$. Sprva je vhodna beseda vpisana v skrajno levih celicah (tj. na začetku traku), vse ostale celice traku pa so prazne (v vsaki je $\sqcup$).

- Nadzorna enota je vedno v kakem stanju iz končne množice stanj $Q = \{q_1, \ldots, q_s\}, s \geq 1$. q_1 je začetno stanje. Nekatera stanja so končna; zbrana so v množici $F \subseteq Q$. ostala stanja so nekončna. Kadar indeks stanja ni pomemben, včasih uporabimo tudi oznaki q_{yes} oz. q_{no} za označevanje končnega oz. nekončnega stanja.

  V nadzorni enoti je Turingov program (TP), ki usmerja delovanje enot TS. Turingov program P je značilen za izbrani TS, tj. različni TS imajo različne TP. TP je parcialna funkcija $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$, imenovana funkcija prehodov.
  Pozor: TS je po definiciji deterministično: za vsak par (stanje, tračni simbol) ima na voljo kvečjemu eno potezo.

  TP δ lahko predstavimo s tabelo $\Delta = Q \times \Gamma$, kjer velja

  - $\Delta[q_i, z_r] = (q_j, z_w, D)$ če $\delta(q_i, z_r) = (q_j, z_w, D)$ je ukaz v δ,
  - $\Delta[q_i, z_r] = 0$ če $\delta(q_i, z_r) \uparrow$ (nedefinirano).

  Brez zmanjšanja splošnosti lahko predpostavimo, da iz q_{no}, vedno obstaja prehod, iz q_{yes} pa ne.
- Okno se lahko premakne (korakoma, preko vmesnih celic) na poljubno celico. Nadzorna enota lahko iz celice pod oknom prebere simbol, v celico pod oknom pa lahko tudi vpiše simbol (tj. zamenja prejšnjega z novim). V enem koraku se okno lahko premakne le na sosednjo celico.
- Pred zagonom TS velja naslednje:

  a) vhodna beseda je zapisana na začetku traku (tj. v skrajno levih celicah);
  b) okno je premaknjeno na začetek traku (tj. nad skrajno levo calico);
  c) nadzorna enota se nahaja v začetnem stanju.

- Odslej naprej TS deluje samostojno, mehanično in korakoma kot mu narekuje TP. Če je TS v stanju $q_i \in Q$ in vidi v celici pod oknom simbol $z_r \in \Gamma$, potem:

  if q_i je končno stanje, then TS se ustavi
  else, if $\delta(q_i, z_r) \uparrow$ (tj. TP nima ukaza za par q_i, z_r), then TS se ustavi
  else, if $\delta(q_i, z_r) \downarrow = (q_j, z_w, D)$, TS stori naslednje:

  a) preide v stanje q_j
  b) vpiše z_w skozi okno
  c) Premakne okno na sosednjo celico v smer D (levo (če $D = L$); desno (če $D = R$)), ali pa pusti okno, tam kjer je (stoj (če $D = S$))

9.3 Trenutni opis TS

trenutni opis (TO) Turingovega stroja je $I = \alpha_1 q \alpha_2$, s tem da se beseda α_1 začne na začetku traka in konča en znak pred simbolom, ki je pod oknom, q je trenutno stanje TS, α_2 pa se začne s simbolom pod oknom in konča s skrajno desnim nepraznim simbolom.
TO je "trenutna slika enot Turingovega stroja" (angl. snapshot) med zaporednima ukazoma.

TO I neposredno preide v TO J – kar zapišemo $I \vdash J$ – če v TP stroja T obstaja ukaz, katerega izvedba spremeni I to J. Refleksivno tranzitivno zaprtje relacije $\vdash$ je relacija $\vdash^*$; če velja $I \vdash^* J$, rečemo, da TO I preide v TO J.

9.4 Raba Turingovega stroja

Turingovi stroji izvajajo tri osnovne računske naloge:

- Računanje vrednosti funkcij
  "Za dano funkcijo φ in argumente $a_1, \ldots, a_k$, izračunaj $\varphi(a_1, \ldots, a_k)$."
- Razpoznavanje množic
  "Za dano množico S in objekt x ugotovi, ali velja ali ne velja $x \in S$."
- Generiranje množic
  "Generiraj zaporedje $x_1, x_2, x_3, \ldots$, ki ima natančno elemente dane množice S."

9.4.1 Računanje vrednosti funkcij s TS

Vsak TS T inducira, za vsak $k \geq 1$, funkcijo $\varphi_T^{(k)}$ ki preslika k besed v besedo.

Definirajmo funkcijo $\varphi_T^{(k)}$:
Definicija: Bodi $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ TS in $k \geq 1$. k-mestna lastna funkcija stroja T je parcialna funkcija $\varphi_T : (\Sigma^*)^k \to \Sigma^*$, definirana kot sledi:
Če vhodne podatke stroja T sestavlja k besed $u_1, \ldots, u_k \in \Sigma^*$, potem je vrednost funkcije $\varphi_T^{(k)}$ pri argumetnih $u_1, \ldots, u_k$ podana s

$$\varphi_T^{(k)}(u_1, \ldots, u_k) := \begin{cases} v, & \text{če se } T \text{ ustavi } \wedge \text{ vrne na traku besedo } v \wedge v \in \Sigma^*; \\ \uparrow, & \text{če se } T \text{ ne ustavi } \vee \text{ na traku ne vrne besede iz } \Sigma^*. \end{cases}$$

Interpretacija argumentov of $u_1, \ldots, u_k$ in rezultata v je odvisna od področja uporabe.

V praksi pogosto naletimo na obratno nalogo:
"Za dano k-mestno funkcijo $\varphi : (\Sigma^*)^k \to \Sigma^*$, najdi TS T, ki računa njene vrednosti."
Drugače: za dano k-mestno funkcijo φ, konstruiraj TS T da bo $\varphi = \varphi_T^{(k)}$.

Definicija: Naj bo $\varphi : (\Sigma^*)^k \to \Sigma^*$ funkcija. Tedaj:

- φ je izračunljiva, če $\exists$ TS, ki izračuna φ povsod na $dom(\varphi) \wedge dom(\varphi) = (\Sigma^*)^k$ (=totalna);
- φ je parcialna izračunljiva (p.i.), če $\exists$ TS, ki izračuna φ povsod na $dom(\varphi)$;
- φ je neizračunljiva, če $\neg\exists$ TS, ki izračuna φ povsod na $dom(\varphi)$.

Opomba: Izračunljiva funkicja je p.i. funkcija, ki je celo totalna .

9.4.2 Razpoznavanje množic

Vsak TS T sprejme nek jezik.

Definicija: Bodi $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ TS in $w \in \Sigma^*$ beseda. T sprejme w če $q_1 w \vdash^* \alpha_1 p \alpha_2$, za neko stanje $p \in F$ in $\alpha_1\alpha_2 \in \Gamma^*$.
Jezik, ki ga sprejme TS T,je množica $L(T) = \{w \mid w \in \Sigma^* \wedge T \text{ sprejme } w\}$.

Torej: Beseda je sprejeta, če se T, ko jo prebere, znajde v kakem končnem stanju. Jezik, ki ga T sprejme, pa je množica vseh takih besed.

V praksi pogosto naletimo na obratno nalogo:
"Za dano množico $S \subseteq \Sigma^*$, najdi TS T, ki sprejme S."
Definicija: Naj bo $S \subseteq \Sigma^*$ jezik. Potem:

- S je odločljiv, če $\exists$ TS, ki odgovori DA/NE na vpr. "Ali je $x \in S$?" za poljuben $x \in \Sigma^*$;
- S je polodločljiv, če $\exists$ TS, ki odgovori DA na vpr. "Ali je $x \in S$?" za poljuben $x \in S$;
- S je neodločljiv, če $\neg\exists$ TS, ki bi odgovoril DA/NE na vpr. "Ali je $x \in S$?" za poljuben $x \in \Sigma^*$

Kaže, da pri nekaterih S ne moremo algoritmično odločiti na vprašanje "$x \in^? S$?". Zakaj?

Naj bo $S = L(T)$ polodločljiv in $x \in \Sigma^*$ poljubna vhodna beseda. Potem:

- če x je v S, potem se bo T gotovo ustavil (in odgovoril DA).
- če x ni v S, potem se lahko zgodi, da se T ne bo ustavil (in zato nikoli odgovoril NE).

Torej: dokler T računa, ne moremo vedeti

- ali se bo T nekoč ustavil (če ga pustimo računati) in odgovoril DA ali NE,
- ali pa bo T računal brez prestanka in zato nikoli odgovoril (niti DA niti NE)

9.4.3 Generiranje množic s TS

Nekateri TS T generirajo jezike.

Definicija: Naj bo $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ TS. Pravimo, da je T generator, če na trak izpiše zaporedje (nekih) besed iz Σ^* razmejenih s simbolom $\# \in \Gamma - \Sigma$.
Jezik, ki ga generira T je množica $G(T) = \{w \mid w \in \Sigma^* \wedge T \text{ izpiše } w\}$.

V praksi pogosto naletimo na obratno nalogo:
"Za dano množico $S \subseteq \Sigma^*$, najdi TS T, ki generira natanko njene elemente."
Drugače: za dano množico S konstruiraj TS T, da bo $G(T) = S$.

9.4.4 Izračunljivo preštevni (i.p.) jeziki (množice)

Denimo, da se množica S da generirati. Torej se bo vsak njen posamični element slejkoprej izpisal. Izberimo poljuben $x \in S$. Recimo, da se x (prvič) izpiše kot n-ti po vrsti, kjer je $n \in \mathbb{N}$. Torej lahko govorimo o prvem, drugem, tretjem ... n-tem izpisanem (generiranem) elementu množice S. Torej vsakemu elementu množice S pripada natanko eno naravno število (njegova zaporedna številka v zaporedju). Pravimo, da se S da oštevilčiti (enumerable).

Definicija: Množica S je izračunljivo preštevna (i.p.), če obstaja TS T, da je $S = G(T)$.
Drugače: S je izračunljivo preštevna, če jo lahko generira kak Turingov stroj

Izrek: S je izračunljivo preštevna natanko tedaj, ko S je polodločljiva.

Opomba: Če je taka S neskončna, se ne bo nikoli izpisala v celoti, toda vsak njen posamični element se bo izpisal v končnem času.

9.4.5 Povzetek

Doslej smo definirali že nekaj vrst formalnih jezikov (oz. množic):
regularne, kontekstno-neodvisne, odločljive, polodločljive (i.p.), neodločljive jezike.

Regularni jeziki $\subset$ Kontekstno-neodvisni jeziki $\subset$ Odločljivi jeziki $\subset$ Polodločljivi jeziki $\subset \Sigma^*$
Neodločljivi jeziki so preostanek Σ^*, ki niso v odločljivih jezikih.

9.5 Različice Turingovega stroja

Osnovni model TS je ekvivalenten vsem različicam TS.
Pomen in uporaba različic TS: Lažje reševanje problemov (dokaz obstoja TS za dani problem P) in hitrejši ter manj prostorsko potratni.

Osnovni TP: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$

9.5.1 TS s končnim pomnilnikom

Ta različica (varianta) V ima v nadzorni enoti končno velik pomnilnik, ki lahko hrani $k \geq 1$ tračnih simbolov in uporablja med računanjem.
TP je $\delta_V : Q \times \Gamma \times \Gamma^k \to Q \times \Gamma \times \{L, R, S\} \times \Gamma^k$.

9.5.2 TS z večslednim trakom

Ta različica V ima trak razdeljen na $tk \geq 2$ vzporednih sledi (angl. tracks). Na vsaki sledi so simboli tračne abecede Γ. Okno vidi tk-terko simbolov, po enega na vsaki sledi.
TP je $\delta_V : Q \times \Gamma^{tk} \to Q \times \Gamma^{tk} \times \{L, R, S\}$.

9.5.3 TS z neomejenim trakom

Ta različica V ima trak, ki je (potencialno) neomejen v obe smeri.
TP je $\delta_V : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.

9.5.4 Večtračni TS

Ta različica V ima $tp \geq 2$ neomejenih trakov. Vsak trak ima svoje okno, ki se lahko premika neodvisno od ostalih.
TP je $\delta_V : Q \times \Gamma^{tp} \to Q \times (\Gamma \times \{L, R, S\})^{tp}$.

9.5.5 TS z večdimenzionalnim trakom

Ta različica V ima d-dimenzionalen trak, $d \geq 2$. Okno se lahko premika v d dimenzijah, tj. $2d$ smereh $L_1, R_1, L_2, R_2, \ldots, L_d, R_d$.
TP je $\delta_V : Q \times \Gamma \times \{L_1, R_1, L_2, R_2, \ldots, L_d, R_d, S\}$.

9.5.6 Nedeterministični TS

Ta različica V ima Turingov program δ ki vsakemu paru priredi (q_i, z_r) končno množico možnih prehodov $\{(q_{j1}, z_{w1}, D_1), (q_{j2}, z_{w2}, D_2), \ldots\}$. stroj V (nadzorna enota) pa izbere enega med njimi.
TP je $\delta_V : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,S\}}$

Vprašanje: Kako V izbere enega izmed možnih prehodov? Odgovor: enako kot pri NKA - ima magično sposobnost pravilnega odločanja.

9.6 Univerzalni Turingov stroj

9.6.1 Kodiranje Turingovih strojev

Naj bo $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$, poljuben osnovni model TS. Stroj T želimo zakodirati, tj. predstaviti z besedo nad neko kodirno abecedo. Kako to storimo?

- Naj bo kodirna abeceda množica $\{0, 1\}$.
- Zakodirali bomo samó funkcijo δ, To bomo storili tako, da se bodo iz kode funkcije δ dali enostavno prebrati/izluščiti parametri Q, Σ, Γ, F, ki tudi določajo T
  To storimo takole:
  1. Če je $\delta(q_i, z_j) = (q_k, z_l, D_m)$ ukaz v funkciji δ, ga predstavimo z besedo
     $$K = 0^i 10^j 10^k 10^l 10^m$$
     kjer so $D_1 = L, D_2 = R, D_3 = S$.
  2. Tako predstavimo vsak ukaz funkcije δ.
  3. Iz dobljenih predstavitev $K_1, K_2, \ldots, K_r$ sestavimo kodo $\langle\delta\rangle$ funkcije δ takole:
     $$\langle\delta\rangle = 111K_1 11K_2 11 \ldots 11K_r 111$$
- Koda $\langle T\rangle$ stroja T je potem kar koda $\langle\delta\rangle$ njegovega programa, tj., $\langle T\rangle := \langle\delta\rangle$.

9.6.2 Oštevilčenje Turingovih strojev

Kodo $\langle T\rangle$ Turingovega stroja lahko interpetiramo kot dvojiški zapis nekega naravnega števila. Temu številu pravimo indeks Turingovega stroja T.

Nekatera naravna števila niso indeksi strojev.

- Dogovor: Vsako naravno število, čigar dvojiški zapis ni oblike, ki jo imajo kode Turingovih strojev, bo indeks posebnega, umetno uvedenega TS, imenovanega prazni TS. Progam δ tega TS je povsod nedefiniran; tj. pri vsaki vhodni besedi se ta TS takoj ustavi (v 0 korakih).

$\longrightarrow$ Vsako naravno število je indeks natanko enega Turingovega stroja.

Za poljubni dani $n \in \mathbb{N}$, lahko iz n izluščimo komponente Q, Σ, Γ, F ki skupaj z δ natančno določijo konkretni TS T, ki ima indeks n.

TS z indeksom n označimo s T_n.

S tem smo oštevilčili (enumerirali) osnovne Turingove stroje. Vsakemu osnovnemu TS pripada natančno določen indeks $n \in \mathbb{N}$ in vsakemu natančno določen osnovni TS.
Z drugimi besedami: naredili smo bijekcijo med $\mathbb{N}$ in TS.

9.6.3 Obstoj univerzalnega Turingovega stroja

Trditev: Obstaja Turingov stroj U, ki lahko izračuna vse, kar se da izračunati s katerimkoli (drugim) Turingovim strojem.

Ideja dokaza: Konstruirati je potrebno poseben TS, ki je sposoben simulirati katerikoli drugi TS. Dokazovanje, da tak poseben stroj obstaja, si olajšamo s Tezo o izračunljivosti (v naslednjih dveh korakih):

a) Zamislili si bomo nek stroj Z in zapisali intuitivni algoritem, ki naj ga izvaja stroj Z

b) Nato se bomo sklicali na Tezo o izračunljivosti, rekoč: "Teza o izračunljivosti zagotavlja, da obstaja Turingov stroj U, ki počne isto kot opisani algoritem na zamišljenem stroju Z."

9.6.4 Dokaz: Obstoj univerzalnega Turingovega stroja

a) Stroj Z si zamislimo kot večtračni TS (ki nima Turingovega programa).

- Vhodni trak vsebuje vhodno besedo, ki ima dva dela: kodo $\langle T \rangle$ poljubnega TS $T = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$, in poljubno besedo w (vh. besedo za T).
- Delovni trak je spočetka prazen. Z ga bo uporabljal natanko tako, kot bi T uporabljal svoj trak pri dani vhodni besedi w.
- Pomožni trak je spočetka prazen. Z ga bo uporabljal za zapisovanje trenutnega stanja, v katerem bi bil T v danem trenutku, in za preverjanje, ali je to stanje končno stanje stroja T.
- Nadzorna enota stroja Z naj izvaja naslednji intuitivo zasnovan algoritem:
  1. Preveri, ali je vhodna beseda oblike $\langle T, w \rangle$, kjer je $\langle T \rangle$ koda nekega osnovnega TS. Če ni, se ustavi.
  2. Iz $\langle T \rangle$ izlušči F in zapiši $\langle q_1, F \rangle$ na pomožni trak.
  3. Kopiraj w na delovni trak in postavi okno na začetek tega traku.
  4. (Naj bo na pomožnem traku $\langle q_i, F \rangle$ v oknu na delovnem traku pa z_r).
     Če je $q_i \in F$, se ustavi. (T bi se ustavil v končnem stanju q_i).
  5. Na vhodnem traku išči v kodi $\langle T \rangle$ zapis ukaza $\delta(q_i, z_r) = \ldots$
  6. Če ne najdeš, se ustavi (T bi se ustavil v nekončnem stanju q_i).
  7. (Ukaz $\delta(q_i, z_r) = \ldots$ je bil najden in prebran; ukaz označimo $\delta(q_i, z_r) = (q_j, z_w, D)$.
     Na delovnem traku izpiši simbol z_w in premakni okno v smeri D.
  8. Na pomožnem traku nadomesti q_i v $\langle q_i, F \rangle$ s q_j.
  9. Skoči na korak 4.

b) Zgornji algoritem lahko izvede tudi človek. Po Tezi o izračunljivosti zato obstaja Turingov stroj $U = (Q_U, \Sigma_U, \Gamma_U, \delta_U, q_1, \sqcup, F_U)$ čigar program δ_U izvaja ravno naš intuitivni algoritem. Ta TS je iskani univerzalni Turingov stroj (UTS).

9.6.5 Konstrukcija univerzalnega Turingovega stroja

Kakšen je najmanjši UTS? (mera je dejansko število ukazov v programu δ_U)

Razred $UTS(s, t)$ $(s, t \geq 2)$ vsebuje vse UTS, ki imajo s stanj in t tračnih simbolov.

Med temi ima (trenutno) univerzalni Turingov stroj v UTS(4,6) najmanj ukazov: 22.

9.6.6 Pomen univerzalnega Turingovega stroja

Turingovo odkritje univerzalnega Turingovega stroja je bil teoretični dokaz, da je splošno-namenski računski stroj vsaj načeloma možen.

Turing pa je bil prepričan, da se dá tak računski stroj tudi dejansko sestaviti: Možno je fizično realizirati računski stroj, ki zmore izračunati vse, kar je izračunljivo na kateremkoli drugem TS (oz. kateremkoli drugem fizičnem računskem stroju). Turing je torej napovedal napravo, ki ji danes pravimo splošno-namenski računalnik.

9.7 Prvi osnovni rezultati

Izreki: Naj bodo S, A, B poljubne množice. Velja naslednje:

a) S je odločljiva $\Rightarrow$ S je polodločljiva

b) S je odločljiva $\Rightarrow$ $\overline{S}$ je odločljiva

c) S in $\overline{S}$ sta polodločljivi $\Rightarrow$ S (in $\overline{S}$) je odločljiva (Postov izrek)

d) A in B sta polodločljivi $\Rightarrow$ $A \cap B$ in $A \cup B$ sta polodločljivi

e) A in B sta odločljivi $\Rightarrow$ $A \cap B$ in $A \cup B$ sta odločljivi

Dokazi na koncu dokumenta.

10 Neodločljivost

10.1 Vrste računskih problemov in primeri

Definirajmo naslednje štiri vrste računskih problemov:

- Odločitveni problemi (imenovani tudi da/ne problemi). Rešitev odločitvenega problema je odgovor DA ali NE (torej en bit).
  - Ali dana množica števil S vsebuje kako praštevilo?
  - Ali dani graf $G(V, E)$ vsebuje kak Hamiltonov cikel?
- Problemi iskanja. Pri dani množici S in lastnosti P je rešitev element $x \in S$, ki ima lastnost P.
  - V dani množici števil S poišči največje praštevilo.
  - V danem uteženem grafu $G(V, E, c)$ poišči najkrajši Hamiltonov cikel.
- Problemi preštevanja. Pri dani množici S in lastnosti P je rešitev število elementov množice S, ki imajo lastnost P.
  - Koliko praštevil je v dani množici števil S?
  - Koliko Hamiltonovih ciklov je v danem grafu $G(V, E)$?
- Problemi generiranja. Pri dani množici S in lastnosti P je rešitev zaporedje (seznam) elementov množice S, ki imajo lastnost P.
  - Izpiši vsa praštevila, ki so v dani množici števil S
  - Izpiši vse Hamiltonove cikle, ki so v danem grafu $G(V, E)$.

10.2 Reševanje računskih problemov

Ali lahko uporabimo pridobljeno znanje o treh osnovnih računskih nalogah, ki jih izvaja Turingov stroj, pri reševanju računskih problemov?

10.2.1 Jezik odločitvenega problema

Definicija: Jezik odločitvenega problema D je množica $L(D)$, definirana z

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

Obstaja tesna zveza med odločitvenimi problemi in množicami, ki omogoča, da prevedemo vprašanja o odločitvenih problemih na vprašanja o množicah. Zvezo bomo konstruirali v štirih korakih.

1. Naj bo D poljuben odločitveni problem.
2. Naj bo d primerek problema D. (d pozitiven oz. negativen, če je odgovor na d DA oz. NE.)
3. Naj bo koda : $D \to \Sigma^*$ kodirna funkcija. (predstavitev d, ki je razumljiva stroju)
4. Zberimo kode vseh pozitivnih primerkov problema D v množici $L(D)$.

Velja naslednja ekvivalenca: (sklicovanje na to ekvivalenco = Ω)

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

To je iskana povezava med odločitvenimi problemi in množicami (formalnimi jeziki).

Kaj smo s tem dosegli:
Reševanje odločitvenega problema D prevedemo na razpoznavanje množice $L(D)$ v Σ^*.

Vprašanje: Kaj nam razpoznavnost jezika $L(D)$ pove o izračunljivosti problema D?

- $L(D)$ je odločljiv $\Rightarrow$ Obstaja algoritem, ki na poljuben $d \in D$, odgovori DA/NE..
  Dokaz: Naj bo L(D) odločljiv. Tedaj $\exists$ TS, ki za poljuben $\langle d\rangle \in \Sigma^*$ odloči, ali je/ni $\langle d\rangle \in L(D)$. Nato Ω.
- $L(D)$ je polodločljiv $\Rightarrow$ Obstaja algoritem, ki,
  - na vsak pozitiven $d \in D$, odgovori DA;
  - na negativen $d \in D$, odgovori NE ali pa sploh ne odgovori.

  Dokaz: Naj bo L(D) polodločljiv. Potem $\exists$ TS, ki sprejme vsak $\langle d\rangle$, kjer $\langle d\rangle \in L(D)$; če $\langle d\rangle \notin L(D)$, TS zavrne $\langle d\rangle$ ali pa se ne ustavi. Potem uporabi Ω.
- $L(D)$ je neodločljiv $\Rightarrow$ Ni algoritma, ki bi na poljuben $d \in D$, odgovoril DA/NE.
  Dokaz: Naj bo $L(D)$ neodločljiv. Tedaj $\neg\exists$ TS, ki za poljuben $\langle d\rangle \in \Sigma^*$ odloči, ali je/ni $\langle d\rangle \in L(D)$. Nato Ω.

Razširitev z množic na odločitvene probleme:
Definicija: Naj bo D odločitveni problem. Tedaj

- problem D je odločljiv (ali izračunljiv) če je jezik $L(D)$ odločljiv;
- problem D je polodločljiv če je jezik $L(D)$ polodločljiv;
- problem D je neodločljiv (ali neizračunljiv) če je jezik $L(D)$ je neodločljiv.

Terminologija:
Namesto odločljiv/neodločljiv problem lahko rečemo tudi izračunljiv/neizračunljiv problem. Vendar je slednje poimenovanje bolj splošno: nanaša se lahko na vse vrste računskih problemov, ne le na odločitvene. Izraza rešljiv/nerešljiv sta še splošnejša: lahko se nanašata na računske in tudi ne-računske probleme (npr. psihološke, službene, družbene, politične ...).

10.3 Neizračunljiv problem – Problem ustavitve

Definicija: Problem ustavitve D_{Halt} je odločitveni problem

$$D_{Halt} = \text{“Ali se TS } T \text{ pri vhodni besedi } w \in \Sigma^*, \text{ ustavi?”}$$

Izrek: Problem ustavitve D_{Halt} je neodločljiv.
Opomba: Posledica je, da ni algoritma, ki bi bil sposoben za poljuben par T, w odgovoriti DA/NE na vprašanje “Ali se T pri vhodni besedi w ustavi?” Drugače: Vsak algoritem za reševanje Problema ustavitve, ki bi ga razvili danes ali v prihodnosti, bo zagotovo odpovedal (ne vrnil odgovora) pri vsaj enem paru T, w.

10.4 Dokaz: Neizračunljiv problem – Problem ustavitve

Definicija: Univerzalni jezik, označen s K_0, je jezik Problema ustavitve, torej,

$$K_0 = L(D_{Halt}) = \{\langle T, w\rangle \mid T \text{ se pri vhodu } w \text{ ustavi}\}.$$

Drugi jezik nastane iz jezika K_0, ko zahtevamo, da je $w := \langle T\rangle$.
Definicija: Diagonalni jezik, označen s K, je definiran s

$$K = \{\langle T, T\rangle \mid T \text{ se pri vhodu } \langle T\rangle \text{ ustavi}\}$$

Opomba:

- K je jezik odločitvenega problema D_H = “Ali se TS T pri vhodni besedi $\langle T\rangle$ ustavi?”
- D_H je podproblem problema D_{Halt} (ker je dobljen iz D_{Halt} z omejitvijo w na $w = \langle T\rangle$).

Načrt dokaza:

- dokazali bomo, da je K neodločljiva množica (lema spodaj).
- potem bo sledilo, da je tudi K_0 neodločljiva množica;
- to pa bo pomenilo, da je D_{Halt} neodločljiv problem.

Lema: K je neodločljiva množica.

Dokaz s protislovjem:
Predpostavimo, da bi bila množica K odločljiva. Potem bi obstajal TS D_K ki bi za poljuben T, odgovoril na vprašanje $\langle T, T\rangle \in ? K$:

$$D_K(\langle T, T\rangle) = \begin{cases} \text{DA}, & \text{če } T \text{ se nad } \langle T\rangle \text{ ustavi;} \\ \text{NE}, & \text{če } T \text{ se nad } \langle T\rangle \text{ ne ustavi;} \end{cases}$$

Zdaj pa konstruirajmo nov TS S: Naš namen je sestaviti S tako, da bo – ko bo za vhod dobil svojo lastno kodo $\langle S\rangle$, S razkril nesposobnost D_K da pravilno napove, ali se S nad $\langle S\rangle$ ustavi. (p.252)

Torej D_K ni zmožen pravilno odgovoriti na vprašanje $\langle S, S\rangle \in^? K$, To nasprotuje predpostavki, da je K odločljiva množica (in D_K pripadajoči TS). Predpostavka ne velja - K je neodločljiva množica.

10.5 Osnovne vrste odločitvenih problemov

10.5.1 Obstajajo neodločljive množice, ki so polodločljive

Izrek: K_0 je polodločljiva množica.

> Dokaz: Poiskati moramo TS, ki sprejme K_0. Zamisel je sledeča. Iskani TS naj za poljubnen dani par $\langle T, w\rangle$, simulira T nad w. Če se simulacija konča – torej se T nad w ustavi – naj TS odgovori DA in se ustavi. Če tak TS obstaja, bo odgovoril DA natanko takrat, ko je $\langle T, w\rangle \in K_0$. Tak TS pa že poznamo: univerzalni Turingov stroj U. Torej je K_0 polodločljiva množica.
>
> *Opomba:* Zato se K_0 imenuje univerzalni jezik.

Neposredna posledica zadnjih dveh izrekov je: *Postov izrek/Korolar:* K_0 je neodločljiva (a še vedno) polodločljiva množica.
Na podoben način dokažemo enako za K.

10.5.2 Obstajajo neodločljive množice, ki niso polodločljive

Izrek: $\overline{K_0}$ ni polodločljiva množica.

> Dokaz: Denimo, da bi bila $\overline{K_0}$ polodločljiva. (Dokazali smo že, da je polodločljiva K_0), Potem bi bili K_0 in $\overline{K_0}$ obe polodločljivi in zato bi bili po obe odločljivi. To bi bilo v protislovju s korolarjem K_0 (Postov izrek). Sklep: $\overline{K_0}$ ni polodločljiva množica.

Na enak način dokažemo, da tudi $\overline{K}$ ni polodločljiva množica.

10.6 Vrste odločitvenih problemov

Vidimo, da je vsak odločitveni problem D ene od treh vrst:

- D je odločljiv
  Obstaja algoritem, ki reši (odgovori z DA/NE na) poljuben primerek $d \in D$.
  Takemu algoritmu včasih rečemo odločevalnik (angl. decider) za problem D.
- D je polodločljiv (a neodločljiv)
  Ni algoritma, ki bi rešil poljuben primerek $d \in D$. Obstaja pa algoritem, ki reši poljuben pozitiven $d \in D$, tj. odgovori DA če in samo če je d pozitiven primerek. (Pri vsaj enem negativnem primerku pa se nikoli ne ustavi).
  Takemu algoritmu včasih rečemo razpoznavalnik (angl. recognizer) za D.
- D ni polodločljiv
  Ni algoritma, ki bi rešil poljuben $d \in^? D$ še več, ni niti algoritma, ki bi rešil poljuben pozitiven $d \in D$. Vsak algoritem za D bo odpovedal (se nikoli ustavil) pri vsaj enem pozitivnem in vsaj enem negativnem primerku problema D.

10.7 Komplementarne množice in pripadajoči odločitveni problemi

Iz prejšnjih izrekov sledi, da so za odločljivost množice S in komplenta $\overline{S}$ naslednje tri možnosti

- S in $\overline{S}$ sta obe odločljivi.
- S in $\overline{S}$ sta obe neodločljivi, pri čemer je ena polodločljiva, druga pa ne.
- S in $\overline{S}$ sta obe neodločljivi, pri čemer nobena ni polodločljiva.

Enako velja za odločljivost pripadajočih odločitvenih problemov.

10.8 Drugi neizračunljivi problemi

V tem razdelku bomo našteli nekaj izbranih neizračunljivih problemov, urejenih po področjih (iz računalništva in matematike). Za nobenega od njih ne obstaja algoritem, ki bi ga zmogel rešiti v popolnosti (rešiti poljuben primerek)

- Problemi o algoritmih in programih (p.262)

  - USTAVLJIVOST ALGORITMOV (PROGRAMOV)
  - PRAVILNOST ALGORITMOV (PROGRAMOV)
  - KRAJŠI EKVIVALENTNI PROGRAM
- Problemi o jezikih in gramatikah (p.263)
  - DVOUMNOST KONTEKSTNO-NEODVISNIH GRAMATIK
  - EKVIVALENTNOST KONTEKSTNO-NEODVISNIH GRAMATIK
  - RAZNE LASTNOSTI KNG IN KNJ
- Problem o izračunljivih funkcijah (p.264)
  - RAZNE LASTNOSTI IZRAČUNLJIVIH FUNKCIJ
- Problemi iz teorije števil, algebre in matematične analize (p.265)
  - REŠLJIVOST DIOFANTSKIH ENAČB
  - NIČELNI PRODUKT MATRIK
  - REALNE NIČLE FUNKCIJ
- Problemi tlakovanja (p.266)
  - TLAKOVANJE POLIGONOV
  - TLAKOVANJE POTI
- POSTOV KORESPONDENČNI PROBLEM (p.267)

10.8.1 Busy beaver problem

Povedano neformalno: garač je najbolj produktiven TS med TS iste vrste.
Mislimo na Turingove stroje, ki ne porabljajo časa s pisanjem simbolov, različnih od 1, in tudi ne z zadrževanjem okna na obiskani celici. Pa jih razdelimo v razrede $\tau_n, N = 1, 2, \ldots$ tako da bo τ_n vseboval vse take TS, ki imajo enako število stanj.

Definicija: Naj bo τ_n $(n \geq 1)$ razred vseh TS, ki imajo:

- neomejen trak v obe smeri;
- n nekončnih stanj (skupaj s q_1) in eno končno stanje q_{n+1};
- vhodno abecedo $\Sigma = \{0, 1\}$ in tračno abecedo $\Gamma = \{0, 1, \sqcup\}$;
- Program δ , ki na trak vedno izpiše le 1 in okno vedno premakne L ali R.

Izrek (Radó): Pri vsakem $n \geq 1$, je v τ_n končno mnogo Turingovih strojev.

Definicija. TS $T \in \tau_n$ je ustavljiv, če se pri praznem vhodu ε ustavi.
Izrek (Radó): Za vsak $n \geq 1$, obstaja v τ_n ustavljiv T. Torej je v vsakem razredu τ_n vsaj eden in kvečjemu $|\tau_n|$) ustavljivih TS.

Sledi, da v τ_n obstaja ustavljiv TS T^* ki zapusti na traku, ko se ustavi, največje število simbolov 1 med vsemi ustavljivimi TS v τ_n. Temu T^* rečemo n-garač.

PROBLEM GARAČA: Naj bo $T \in \bigcup_{i \geq 1} \tau_i$ poljuben TS. Vprašanje: "Ali je T garač?" (tj. "Ali obstaja $n \geq 1$, pri katerem je $T = n$-garač?")

Definicija. Garačeva funkcija $s(n)$ je definirana takole:
$s(n) =$ 'število simbolov 1, ki jih zapusti n-garač na traku, ko se ustavi'.
Izrek: Garačeva funkcija je neizračunljiva.

11 Računska zahtevnost

11.1 Deterministični čas in prostor (DTIME, DSPACE)

11.1.1 Deterministična časovna zahtevnost & razredi DTIME

Definicija: Naj bo $M = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ DTS s $k \geq 1$ neomejenimi trakovi. Pravimo, da ima DTS M (deterministično) časovno zahtevnost $T(n)$ če za vsako vhodno besedo $w \in \Sigma^*$ dolžine $|w| = n$, naredi kvečjem $T(n)$ korakov (potez), preden se ustavi.

- V definiciji velja naslednja predpostavka: M prebere celo vhodno besedo w; Posledica: $T(|w|) \geq |w| + 1$, tj. $T(n) \geq n + 1$. Sledi: $T(n)$ vsaj linearna funkcija.

Torej TS M (det.) časovne zahtevnosti $T(n)$ (zagotovo) odgovori na vprašanje $w \in^? L(M)$ v kvečjemu $T(|w|)$ korakih.

Definicija: Jezik L ima (deterministično) časovno zahtevnost $T(n)$, če obstaja DTS M z (det). časovno zahtevnostjo $T(n)$, za katerega je $L = L(M)$.
Razred vseh jezikov z (det.) čas. zahtevnostjo $T(n)$ je

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

Intuitivno, DTIME($T(n)$) vsebuje natanko vse jezike L, pri katerih se dá odgovor na poljubno vprašanje $w \in^? L$ det. deterministično izračunati v času $\leq T(|w|)$.
Zgornji definiciji ustreza (zaradi Ω) podobna definicija, ki se nanaša na odločitvene probleme: $L(D)$ (p.277).

11.1.2 Deterministična prostorska zahtevnost & razredi DSPACE

Definicija: Naj bo $M = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ DTS z 1 vhodnim trakom in $k \geq 1$ delovnimi trakovi. Pravimo, da ima DTS M (deterministično) prostorsko zahtevnost $S(n)$ za vsako vhodno besedo $w \in \Sigma^*$ dolžine $|w| = n$, M porabi kvečjemu $S(n)$ celic na vsakem delovnem traku, preden se ustavi.

- Opomba: celice na vhodnem traku se pri porabi prostora ne upoštevajo.
- V definiciji velja naslednja predpostavka: M uporabi vsaj eno celico na vsakem delovnem traku (tisto, ki je pod oknom pred zagonom M). Sledi: $S(n) \geq 1$.

Torej TS M (det.) prostorske zahtevnosti $S(n)$ (zagotovo) odgovori na vprašanje $w \in^? L(M)$ na kvečjemu $S(|w|)$ celicah vsakega od delovnih trakov.

Definicija. Jezik L ima (deterministično) prostorsko zahtevnost $S(n)$, če obstaja DTS M z (det). prostorsko zahtevnostjo $S(n)$, za katerega je $L = L(M)$.
Razred vseh jezikov z (det.) prostorsko zahtevnostjo $S(n)$ je

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

Intuitivno: DSPACE($S(n)$) vsebuje natanko vse jezike L, pri katerih se dá odgovor na poljubno vprašanje $w \in^? L$ deterministično izračunati na $\leq S(|w|)$ celicah (na vsakem od delovnih trakov).
Zgornjima definicijama ustrezata (zaradi Ω) podobni definiciji za DA/NE probleme $L(D)$ (p.279).

11.2 Nedeterministični čas in prostor (NTIME, NSPACE)

11.2.1 Nedeterministična časovna zahtevnost & razredi NTIME

Definicija: Naj bo $N = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ nedeterministični TS s $k \geq 1$ trakovi. Pravimo, da ima NTS N nedeterministično časovno zahtevnost $T(n)$, če za vsako vhodno besedo $w \in \Sigma^*$ of dolžine $|w| = n$, obstaja izračun, v katerem N naredi kvečjemu $T(n)$ korakov (potez), preden se ustavi.

- V definiciji velja naslednja predpostavka: N prebere celo vhodno besedo w; Posledica: $T(|w|) \geq |w| + 1$, tj. $T(n) \geq n + 1$. Sledi: $T(n)$ je vsaj linearna funkcija..

Torej NTS N nedet. časovne zahtevnosti $T(n)$ (zagotovo) odgovori na vprašanje $w \in^? L(N)$ v kvečjemu $T(|w|)$ korakih.

Definicija. Jezik L ima nedeterministično časovno zahtevnost $T(n)$, če obstaja NTS N z nedet. časovno zahtevnostjo $T(n)$, tako da je $L = L(N)$.
Razred vseh takih jezikov je

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

Intuitivno: NTIME($T(n)$) vsebuje natanko vse jezike L pri katerih se dá odgovor na poljubno vprašanje $w \in^? L$ nedeterministično izračunati v času $\leq T(|w|)$.. Zgornji definiciji ustreza (zaradi Ω) podobna definicija, ki se nanaša na odločitvene probleme $L(D)$ (p.282).

11.2.2 Nedeterministična prostorska zahtevnost & razredi NSPACE

Definicija. Naj bo $N = (Q, \Sigma, \Gamma, \delta, q_1, \sqcup, F)$ NTS z 1 vhodnim trakom in $k \geq 1$ delovnimi trakovi. Rečemo, da ima NTS N nedeterministično prostorsko zahtevnost $S(n)$, če za vsako vhodno besedo $w \in \Sigma^*$ dolžine $|w| = n$, obstaja izračun, v katerem N porabi kvečjemu $S(n)$ celic na vsakem delovnem traku, preden se ustavi.

- Opomba: celice na vhodnem traku se pri porabi prostora ne upoštevajo.
- V definiciji velja naslednja predpostavka: N uporabi vsaj eno celico na vsakem delovnem traku (tisto, ki je pod oknom pred zagonom N). Sledi $S(|w|) \geq 1$.

Torej NTS N nedet. prostorske zahtevnosti $S(n)$ (zagotovo) odgovori na vprašanje $w \in^? L(N)$ na kvečjemu $S(|w|)$ celicah vsakega od delovnih trakov.

Definicija. Jezik L ima nedeterministično sprostorsko zahtevnost S(n), če obstaja NTS N z nedet. prostorsko zahtevnostjo S(n), tako da je L = L(N).
Razred vseh takih jezikov je

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

Intuitivno: NSPACE($S(n)$) vsebuje natanko vse jezike L pri katerih se dá odgovor na poljubno vprašanje $w \in^? L$ nedeterministično izračunati na $\leq S(|w|)$ celicah.
Zgornjima definicijama ustrezata (zaradi Ω) podobni definiciji za DA/NE probleme $L(D)$ (p.284).

11.3 Povzetek razredov zahtevnosti

Razredi z vidika formalnih jezikov in TS:

$\text{DTIME}(T(n)) = \{L \mid L \text{ je jezik} \wedge L \text{ ima (det.) časovno zahtevnost } T(n)\}$
$\text{DSPACE}(S(n)) = \{L \mid L \text{ je jezik} \wedge L \text{ ima (det.) prostorsko zahtevnost } S(n)\}$
$\text{NTIME}(T(n)) = \{L \mid L \text{ je jezik} \wedge L \text{ ima nedet. časovno zahtevnost } T(n)\}$
$\text{NSPACE}(S(n)) = \{L \mid L \text{ je jezik} \wedge L \text{ ima nedet. prostorsko zahtevnost } S(n)\}$

Isti razredi z vidika odločitvenih (odl.) problemov in algoritmov:

$\text{DTIME}(T(n)) = \{D \mid D \text{ je odl. problem} \wedge L(D) \text{ ima (det) časovno zahtevnost } T(n)\}$
$\text{DSPACE}(S(n)) = \{D \mid D \text{ je odl. problem} \wedge L(D) \text{ ima (det.) prostorsko zahtevnost } S(n)\}$
$\text{NTIME}(T(n)) = \{D \mid D \text{ je odl. problem} \wedge L(D) \text{ ima nedet. časovno zahtevnost } T(n)\}$
$\text{NSPACE}(S(n)) = \{D \mid D \text{ je odl. problem} \wedge L(D) \text{ ima nedet. prostorsko zahtevnost } S(n)\}$

Intuitivno:

$\text{DTIME}(T(n)) = \{\text{vsi odl. problemi, rešljivi z det. algoritmom v času } T(n)\}$
$\text{DSPACE}(S(n)) = \{\text{vsi odl. problemi, rešljivi z det. algoritmom na prostoru } S(n)\}$
$\text{NTIME}(T(n)) = \{\text{vsi odl. problemi, rešljivi z nedet. algoritmom v času } T(n)\}$
$\text{NSPACE}(S(n)) = \{\text{vsi odl. problemi, rešljivi z nedet. algoritmom na prostoru } S(n)\}$

11.4 Stiskanje traku, linearna pohitritev in zmanjšanje števila trakov

11.4.1 Stiskanje traku

Zakodiramo lahko več simbolov z enim simbolom iz večje tračne abecede.

Korolar: Za poljuben $c > 0$ je:

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

11.4.2 Linearna pohitritev

Združimo več zaporednih korakov v nov, večji korak (večji ukaz).

Možno, če sta izpolnjena dva pogoja:

- TS mora imeti vsaj 2 trakova (tj. $k > 1$),
- Veljati mora $\inf_{n\to\infty} T(n)/n = \infty$

  Intuitivno: $T(n)$ mora naraščati vsaj malo hitreje kot n. Potem bo po branju vhodne besede ostalo vsaj malo časa za računanje.

Korolar: Če $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$, potem je za poljuben $c > 0$:

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

11.4.3 Povzetek

Pri $c > 0$ in dveh razumnih pogojih velja:

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: "D je v razredu DTIME(n^2)",
rečemo tudi: "D ima (det.) časovno zahtevnost reda $O(n^2)$".

11.4.4 Zmanjšanje števila trakov v TS

Časovna zahtevnost:

Če $L \in$ DTIME($T(n)$) pri $k > 1$, potem je $L \in$ DTIME($T^2(n)$) pri $k = 1$
Če $L \in$ NTIME($T(n)$) pri $k > 1$, potem je $L \in$ NTIME($T^2(n)$) pri $k = 1$
Če $L \in$ DTIME($T(n)$) pri $k > 2$, potem je $L \in$ DTIME($T(n)logT(n)$) pri $k = 2$
Če $L \in$ NTIME($T(n)$) pri $k > 2$, potem je $L \in$ NTIME($T(n)logT(n)$) pri $k = 2$

Prostorska zahtevnost (zmanjšanje števila trakov v TS ne vpliva na prostorsko zahtevnost TS):

Če $L \in$ DSPACE($S(n)$) pri $k > 1$, potem je $L \in$ DSPACE($S(n)$) pri $k = 1$
Če $L \in$ NSPACE($S(n)$) pri $k > 1$, potem je $L \in$ NSPACE($S(n)$) pri $k = 1$

11.5 Relacije med razredi DTIME, DSPACE, NTIME, NSPACE

11.5.1 Relacije med razredi zahtevnosti različnih vrst

Izreki:

- $DTIME(T(n)) \subseteq DSPACE(T(n))$
  Kar se da rešiti deterministično v času reda $O(T(n))$, se da rešiti deterministično na prostoru reda $O(T(n))$
  Enako velja tudi za nedet. zahtevnosti: $NTIME(T(n)) \subseteq NSPACE(T(n)))$
- $L \in$DSPACE$(S(n)) \wedge S(n) \geq log_2 n \Rightarrow \exists c : L \in$DTIME$(c^{S(n)})$
  Kar se da rešiti deterministično na prostoru reda $O(S(n))$, se da rešiti deterministično v času reda $O(c^{S(n)})$. (Konstanta c je odvisna od L.)
- $L \in$NTIME $(T(n)) \Rightarrow \exists c : L \in$DTIME$(c^{T(n)})$
  Kar se da rešiti nedeterministično v času reda $O(T(n))$, se da rešiti deterministično v času reda $O(c^{T(n)})$.
  Posledica: zamenjava nedeterminističnega algoritma z ekvivalentnim determinističnim algoritmom povzroči kvečjemu eksponentno povečanje časa, ki je potreben za izračun rešitve primerka problema.
- Savitchev izrek: NSPACE$(S(n)) \subseteq$ DSPACE$(S^2(n))$, če $S(n) \geq log_2 n \wedge S(n)$ popolnoma prostorsko predstavljiva.
  Kar se da rešiti nedeterministično na prostoru reda $O(S(n))$, se da rešiti deterministično na prostoru reda $O(S^2(n))$.
  Posledica: zamenjava nedeterminističnega algoritma z ekvivalentnim determinističnim algoritmom povzroči kvečjemu kvadratno povečanje protora, ki je potreben za izračun rešitve primerka problema.

11.5.2 Predstavljive funkcije zahtevnosti

Definicija: Funkcija $S(n)$ je prostorsko predstavljiva, če obstaja TS M s prostorsko zahtevnostjo $S(n)$ in naslednjo lastnostjo: za vsak $n \in N$ obstaja vhodna beseda dolžine n, pri kateri M med računanjem uporabi natanko $S(n)$ celic traku. Če pa za vsak $n \in N$ pri vsaki vhodni besedi dolžine n stroj M uporabi natanko $S(n)$ celic, rečemo, da je $S(n)$ popolnoma prostorsko predstavljiva.

Definicija: Funkcija $T(n)$ je časovno predstavljiva, če obstaja TS M s časovno zahtevnostjo $T(n)$ in naslednjo lastnostjo: za vsak $n \in N$ obstaja vhodna beseda dolžine n, pri kateri M med računanjem napravi natanko $T(n)$ korakov. Če pa za vsak $n \in N$ pri celo vsaki vhodni besedi dolžine n stroj M napravi natanko $T(n)$ korakov, rečemo, da je $T(n)$ popolnoma časovno predstavljiva.

Opomba: Enakovreden izraz za predstavljivo funkcijo je verna funkcija

11.6 Razredi P, NP, PSPACE, NPSPACE

Zahteva računanja po nekem računskem viru (času, prostoru, energiji, procesorjih,...) je razumna, če je navzgor omejena s polinomom.

11.6.1 P, NP, PSPACE, NPSPACE

Definicija: Razredi zahtevnosti P, NP, PSPACE in NPSPACE so definirani:

- P= $\bigcup_{i\geq 1}$DTIME(n^i)
  razred vseh odločitvenih problemov, ki so deterministično rešljivi v polinomsko omejenem času
- NP= $\bigcup_{i\geq 1}$NTIME(n^i)
  razred vseh odločitvenih problemov, ki so nedeterministično rešljivi v polinomsko omejenem času
- PSPACE= $\bigcup_{i\geq 1}$DSPACE(n^i)
  razred vseh odločitvenih problemov, ki so deterministično rešljivi na polinomsko omejenem prostoru
- NPSPACE= $\bigcup_{i\geq 1}$NSPACE(n^i)
  razred vseh odločitvenih problemov, ki so nedeterministično rešljivi na polinomsko omejenem prostoru

11.6.2 Osnovne relacije (med P, NP, PSPACE, NPSPACE)

Izrek. Veljajo naslednja vsebovanja: P $\subseteq$ NP $\subseteq$ PSPACE = NPSPACE

Dokazi:

- (P $\subseteq$ NP) Vsak DTS polinomske časovne zahtevnosti je trivialni NTS (vsak ukaz da na izbiro kvečjemu en prehod) enake (polinomske) časovne zahtevnosti.
- (NP $\subseteq$ PSPACE) Če je $L \in$ NP potem (po definiciji) $\exists k$ da je $L \in$ NTIME(n^k).
  Zato je (po enem od osnovnih izrekov) $L \in$ NSPACE(n^k), in zato (po Savitchevem izreku) $L \in$ DSPACE(n^{2k}). Sledi (iz definicije PSPACE), da je $L \in$PSPACE.
- (PSPACE = NPSPACE) Izrek dokažemo v dveh korakih: najprej $\subseteq$ potem $\supseteq$
  - (PSPACE $\subseteq$ NPSPACE) Vsak DTS polinomske prostorske zahtevnosti je trivialni NTS (v katerem vsak ukaz da na izbiro ≤ 1 prehod) enake (polinomske) prostorske zahtevnosti.
  - (NPSPACE $\subseteq$ PSPACE) NPSPACE =(definicija NPSPACE)= $\bigcup_{i\geq 1}$ NSPACE(n^i) $\subseteq_{Savitchev\ izrek}$ $\bigcup_{i\geq 1}$ DSPACE(n^{2i}) $\subseteq_{def.\ PSPACE}$ PSPACE.

11.7 Problem P $=^?$ NP

Pravkar smo dokazali PSPACE = NPSPACE
⇒ Kadar je prostorska zahtevnost polinomska, nedeterminizem ne poveča računske moči.

Vemo že, da je P⊆NP, Ali je P=NP?
⇒ Če je časovna zahtevnost polinomska, ali nedeterminizem ne poveča računske moči?

Zato poskušamo dokazati, da je P⊆NP
z iskanjem "najtežjega" problema v NP in pokažemo da ta problem ni v P.

11.7.1 Prevedbe problemov

Zamisel: Denimo, da bi v NP obstajal D^*, na katerega bi se dal "hitro prevesti" vsak $D \in$ NP. Prevedbo definirajmo takole: problem D se dá "hitro" prevesti na problem D^*,

- če obstaja funkcija $r : D \to D^*$
- ki "hitro" preslika poljuben primerek $d \in D$ v primerek $r(d) \in D^*$
- tako da se rešitev s primerka $r(d)$ dá "hitro" transformirati v rešitev "?" primerka d.
  Intuitivno: Reševanje problema D bi se dálo "hitro" nadomestiti z reševanjem problema D^*. Zato bi bil D kvečjemu tako težek kot D^*! (Tu nič ne govorimo o težavnosti reševanja D^*.)

Če bi tak $D^* \in$ NP obstajal, bi bil vsak drug $D \in$ NP kvečjemu tako težek kot D*;
z drugimi besedami: D^* bi bil najtežji problem v NP (ali pa eden od njih).

11.7.2 Polinomske-časovne prevedbe

Definirajmo: "hitro" = v deterministočnem polinomskem času.

Problem $D \in$NP je polinomsko-časovno prevedljiv na problem D', tj. $D \leq^p D'$, če obstaja deterministični TS M polinomske časovne zahtevnosti, ki preslika poljuben primerek $d \in D$, v primerek $d' \in D'$, tako da velja: d je pozitiven $\Leftrightarrow$ d' je pozitiven.

Relaciji $\leq^p$ rečemo polinomska-časovna prevedba (polinomska prevedba, kadar je jasno, da gre za čas).

Intuitivno: M v polinomskem času nadomesti, $d \in D$ z $d' \in D'$ ki ima enak odgovor kot d.
Kako to stori? M na podlagi vhodne besede $\langle d \rangle$ vrne v času poly($|\langle d \rangle|$) besedo $M(\langle d \rangle)$,
za katero velja $\langle d \rangle \in L(D) \Leftrightarrow M(\langle d \rangle) \in L(D')$ (Tu je seveda $M(\langle d \rangle) = \langle d' \rangle$, koda primerka d'.)

11.8 NP-polni in NP-težki problemi

Videli smo, da je najtežji problem v NP vsak problem D^*, ki izpolnjuje pogoja:

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

Definicija: Problem D^* je NP-težek, če:

- za vsak $D \in$ NP velja $D \leq^p D^*$.

Definicija: Problem D^* je NP-poln, če velja:

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

Torej: problem D^* je NP-poln $\Leftrightarrow$ problem je v razredu NP $\wedge$ problem je NP-težek.

11.8.1 Obstaja NP-poln problem: SAT

Definicija. Logični izraz (Boolov izraz) induktivno definiramo takole:

- Logične spremenljivke $x_1, x_2, \ldots$ so logični izrazi,
- Če sta E, F logična izraza, so $\neg E$, $E \vee F$, in $E \wedge F$ logični izrazi.

Definicija: Logični izraz E je izpolnljiv, če obstaja prireditev logičnih vrednosti RESNIČNO/NERESNIČNO njegovim spremenljivkam, da ima E logično vrednost RESNIČNO.

Definicija: Problem izpolnljivosti je odločitveni problem
SAT = "Ali je logični izraz E izpolnljiv?"

Izrek (Cook-Levin): SAT je NP-poln problem

(Torej: Za problem D* lahko vzamemo SAT.)

11.8.2 Dokazovanje NP-polnosti problemov

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

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

Intuitivno: Če je problem D prevedljiv v polinomskem času na kak problem v P (oz. v NP), potem je tudi sam D P (oz. v NP).

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

Korolar: Velja sledeče:

- 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

Korolar nam razkrije metodo za dokazovanje NP-težkosti oz. NP-polnosti problema $D^\star$:

- Za dokaz, da je $D\star$ NP-težek: Nek znano NP-težki problem D^* prevedi v polinomskem času na $D\star$.
- Za dokaz, da je $D\star$ NP-poln: Nek znano NP-polni problem D^* prevedi v polinomskem času na $D\star$ in dokaži, da je $D\star$ v NP.

11.8.3 Primeri NP-polnih problemov

S to metodo je bila dokazana NP-polnost oz. NP-težkost že nekaj tisoč problemov. Spodaj navajamo samo tri (za več glejte npr. Wikipedia: List of NP-complete problems).

- RAZDELITEV (PARTITION)
  - Primerek: Končna množica A naravnih števil.
  - Vprašanje: Ali obstaja podmnožica $B \subseteq A$, da je $\sum_{a \in B} a = \sum_{a \in A-B} a$?
- HAMILTONOV CIKEL (HAMILTONIAN CYCLE)
  - Primerek: Graf $G(V, E)$.
  - Vprašanje: Ali $G(V, E)$ vsebuje Hamiltonov cikel?
- KOŠI (BIN PACKING)
  - Primerek: Končna množica U naravnih števil in naravni števili c in k
  - Vprašanje: Ali obstaja razdelitev množice U na disjunktne množice $U_1, U_2, ..., U_k$, tako da je vsota števil v vsaki U_i kvečjemu c?

11.8.4 Povzetek

Če je P $\neq$ NP, potem so razmere v razredu NP naslednje:

- NPC je razred vseh NP-polnih problemov
- NPI je razred vseh NP-vmesnih problemov.
  Izrek (Ladner): Če P$\neq$NP, potem v NP obstaja problem, ki ni niti v P niti v NPC.
  Rekli bomo, da je tak problem NP-vmesen. Kandidat za NP-vmesni problem je odločitveni problem "Ali je n sestavljeno število?"

Če je P $\neq$ NP, potem velja: Če je problem v NPC ali NPI, potem njegova deterministična časovna zahtevnost ni polinomsko omejena.

- Pravimo, da so problemi, ki so v razredu P, obvladljivi (tractable). Ostali izračunljivi problemi so neobvladljivi (intractable).
- Izjema so problemi v razredih NPC in NPI. Za te probleme še ni znano, ali so obvladljivi ali neobvladljivi (domnevamo pa, da so neobvladljivi).

12 Neuradni dokazi odločljivosti množic

a) S je odločljiva $\Rightarrow$ S je polodločljiva

S je odločljiva $\Rightarrow$ $\exists$TS M, ki odg. DA/NE na vpr. "$x \in^? S$" za $\forall x \in \Sigma^*$
$S \subseteq \Sigma^* \Rightarrow$ $\exists$TS (isti M), ki odg. DA na vpr. "$x \in^? S$" za $\forall x \in S \Rightarrow S$ je polodločljiva. $\square$

b) S je odločljiva $\Rightarrow$ $\overline{S}$ je odločljiva

S je odločljiva $\Rightarrow$ $\exists$TS M, ki odg. DA/NE na vpr. "$x \in^? S$" za $\forall x \in \Sigma^*$
Zgradimo TS $\overline{M}$ za odločanje $\overline{S}$, ki vrne DA ko M vrne NE in vrne NE ko M vrne DA.
$\overline{M}$ odg. DA/NE na vpr. "$x \in^? \overline{S}$" za $\forall x \in \Sigma^* \Rightarrow \overline{S}$ je odločljiva. $\square$

c) S in $\overline{S}$ sta polodločljivi $\Rightarrow$ S je odločljiva (Postov izrek)

$\overline{S}$ in S sta polodločljivi $\Rightarrow$ $\exists$TS $M, \overline{M}$, ki odgovorita DA na vpr. "$x \in^? S/\overline{S}$" za $\forall x \in S/\overline{S}$
Zgradimo TS M' za odločanje S, ki vsebuje M in $\overline{M}$ in v vsakem svojem koraku naredi en korak M ter en korak $\overline{M}$. M' vrne odgovor glede na to kateri TS se je ustavil prvi (DA če M, NE če $\overline{M}$).
Ker se bo za vsak vhod ($S \cup \overline{S} = \Sigma^*$) vedno ustavil eden od TS,
bo M' odg. DA/NE za $\forall x \in \Sigma^*$ $\Rightarrow$ S je odločljiva (prav tako $\overline{S}$). $\square$

d) A in B sta polodločljivi $\Rightarrow$ $A \cap B$ in $A \cup B$ sta polodločljivi

A in B sta polodločljivi $\Rightarrow$ $\exists$TS M_A, M_B, ki odg. DA na vpr. "$x \in^? A/B$" za $\forall x \in A/B$
Zgradimo TS $M_\cup$ za odločanje $A \cup B$, ki vsebuje M_A in M_B in v vsakem svojem koraku naredi en korak M_A ter en korak M_B in vrne DA če vsaj eden od TS vrne DA.
$M_\cup$ odg. DA na vpr. "$x \in^? A \cup B$?" za $\forall x \in A \cup B \Rightarrow A \cup B$ je poldoločljiva.

Zgradimo TS $M_\cap$ za odločanje $A \cap B$, ki vsebuje M_A in M_B in najprej zažene enega od TS, če ta odg. DA, zaženemo še drugega. Če sta oba odg. DA vrne DA, sicer NE.
$M_\cap$ odg. DA na vpr. "$x \in^? A \cap B$?" za $\forall x \in A \cap B \Rightarrow A \cap B$ je polodločljiva. $\square$

e) A in B sta odločljivi $\Rightarrow$ $A \cap B$ in $A \cup B$ sta odločljivi

A in B sta odločljivi $\Rightarrow$ $\exists$TS M_A, M_B, ki odg. DA/NE na vpr. "$x \in^? A/B$" za $\forall x \in \Sigma^*$
Zgradimo TS $M_\cup$ za odločanje $A \cup B$, ki vsebuje M_A in M_B in zažene oba TS dokler ne vrneta odg. Če je vsaj en odgovor DA vrne DA, sicer NE.
$M_\cup$ odg. DA/NE na vpr. "$x \in^? A \cup B$?" za $\forall x \in \Sigma^* \Rightarrow A \cup B$ je odločljiva.

Zgradimo tak TS $M_\cap$ za odločanje $A \cap B$, ki vsebuje M_A in M_B in zažene oba TS dokler ne vrneta odg. Če sta oba odg. DA vrne DA, sicer NE.
$M_\cap$ odg. DA/NE na vpr. "$x \in^? A \cap B$?" za $\forall x \in \Sigma^* \Rightarrow A \cap B$ je odločljiva. $\square$