Formalni jezici i jezični procesori I

VREMENSKA I PROSTORNA KOMPLEKSNOST

prof. dr. sc. Sanda Martinčić - Ipšić

smarti@inf.uniri.hr

Složenost jezika

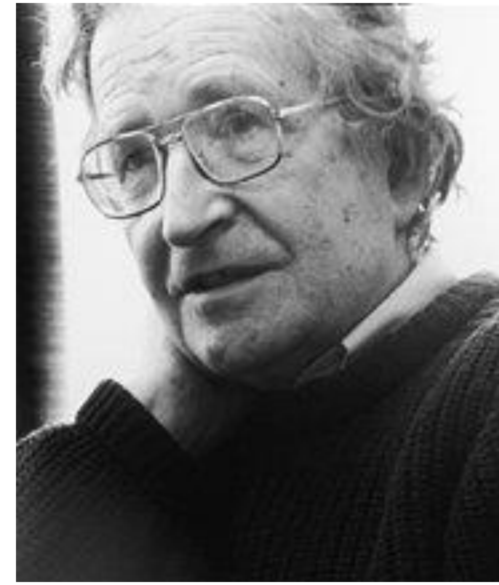
- strukturna složenost jezika
  - složenost automata koji jezik prihvaća
  - **Chomskyeva hijerahija jezika**
- složenost prihvaćanja jezika
  - vrijeme i prostor potrebni da se jezik prihvati
  - **vremenska i prostorna kompleksnost**

Strukturna složenost jezika

- ako su A i B dvije klase jezika i ako je A pravi podskup od B onda vrijedi:
  - automat koji prihvaća jezike iz klase A jednostavniji je (po strukturi) od automata koji prihvaća jezik iz klase B
  - produkcije gramatike koje generira jezik iz klase A su jednostavnije od produkcija gramatike koja generira jezik iz klase B

- jezici iz klase A su jednostavnije strukturne složenosti od klase B

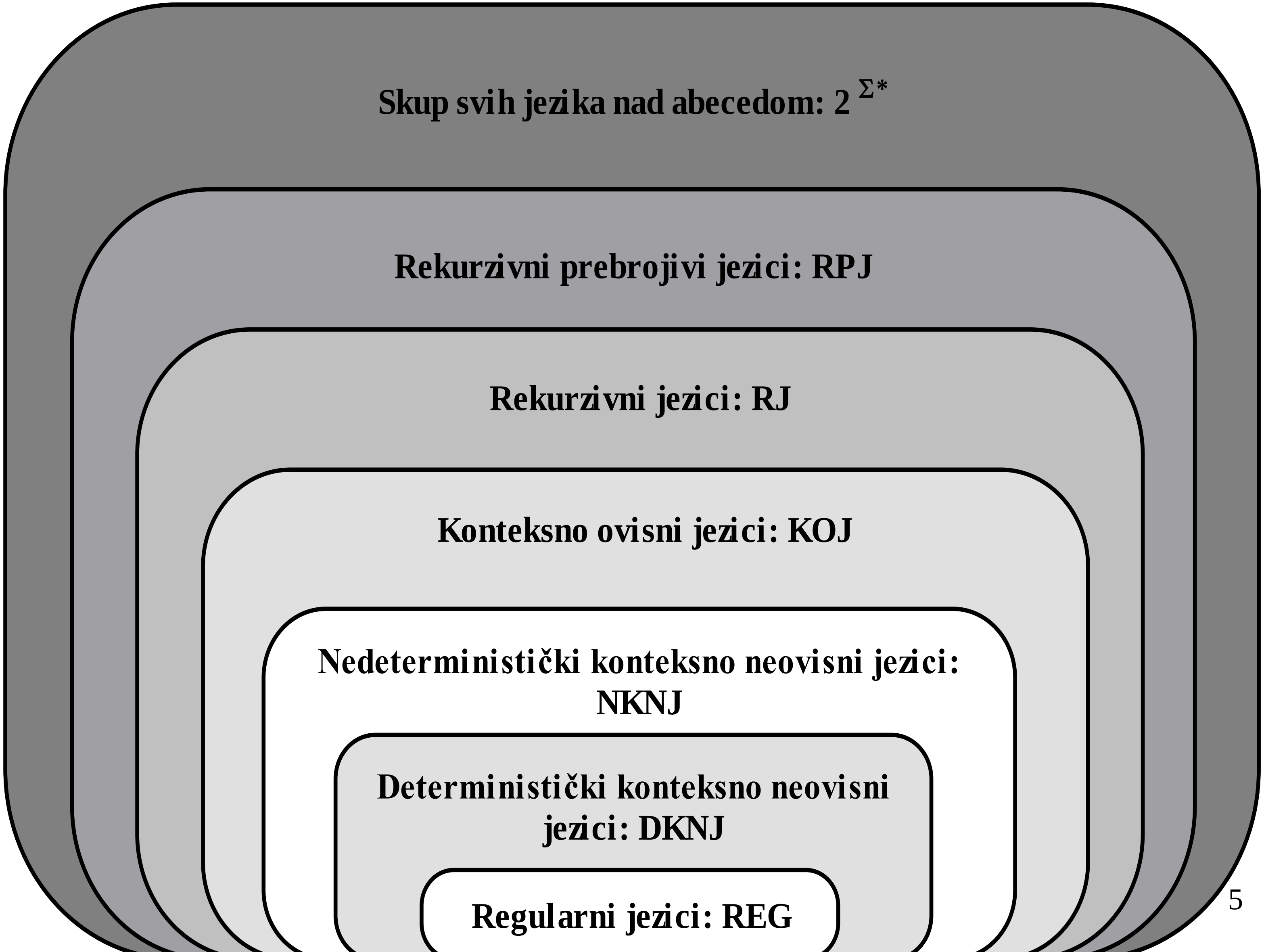
Noam Chomsky



- (krajem 1950-tih) istražuje modele prirodnih jezika
- definira hijerarhiju složenosti jezika s obzirom na strukturnu složenost
- CNO,...
- nacitiraniji živući znanstvenik 1980-92

- danas: a leading critic of US foreign policy has made him controversial

Chomskyjeva hijerarhija jezika



Chomskyjeva hijerarhija jezika II

Skup svih jezika nad abecedom: 2^{Σ^*}

Dijagonalni jezik $L_d \in 2^{\Sigma^*}$ i $L_d \notin RPJ$

Rekurzivni prebrojivi jezici: RPJ

Univerzalni jezik $L_u \in RPJ$ i $L_u \notin RJ$

Rekurzivni jezici: RJ

$L_r \in RJ$ i $L_r \notin KOJ$

Konteksno ovisni jezici: KOJ

$L1=\{ww \mid w \in (0+1)^* \text{ i } |w| > 1\}$ $L1 \in KOJ$ i $L1 \notin NKNJ$

Nedeterministički konteksno neovisni jezici: NKNJ

$L2=\{ww^R \mid w \in (0+1)^* \text{ i } |w| > 1\}$ $L2 \in NKNJ$ i $L2 \notin DKNJ$

Deterministički konteksno neovisni jezici: DKNJ

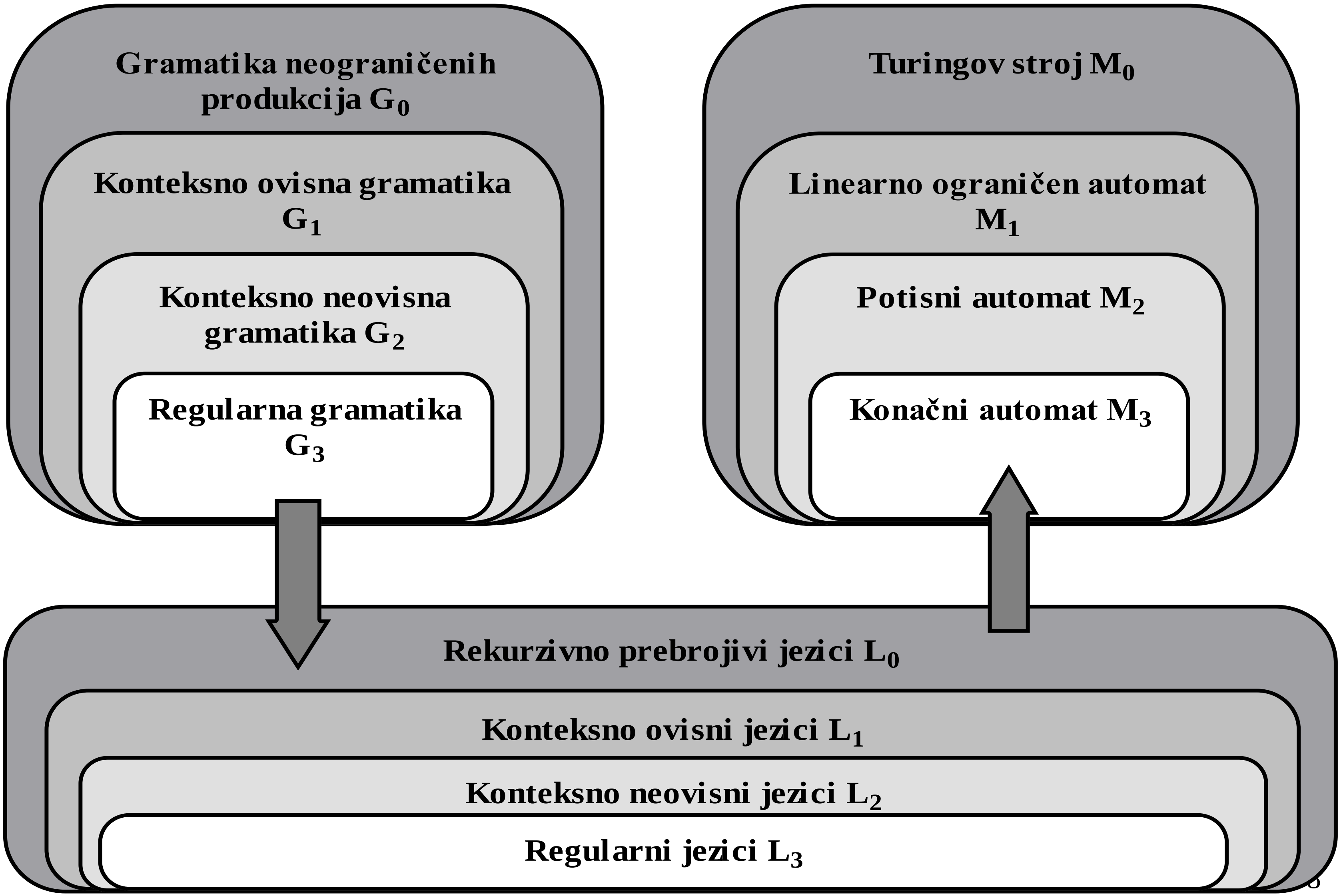
$L3=\{w2w^R \mid w \in (0+1)^* \text{ i } |w| > 1\}$ $L3 \in DKNJ$ i $L3 \notin REG$

Regularni jezici: REG

Hijerarhija automata i gramatika I

- zasniva se na istovjetnostima:
  - regularnih jezika, konačnih automata i regularnih gramatika
  - kontekstno neovisnih jezika, potisnog automata i kontekstno neovisnih gramatika
  - kontekstno ovisnih jezika, linearno ograničenog automata i kontekstno ovisnih gramatika **(nismo radili)**
    - broj znakova desne strane **veći ili jednak** broju znakova lijeve strane produkcije
  - rekurzivno prebrojivih jezika, Turingovog stroja i gramatike neograničenih produkcija

Hijerarhija automata i gramatika II



Istovjetnost

Gramatika

1. Gramatika neograničenih produkcija $G_0=(V,T,P,S)$

   $\alpha \rightarrow \beta$

   nema ograničenja

2. Kontekstno ovisna gramatika $G_1=(V,T,P,S)$

   $\alpha \rightarrow \beta,\ |\alpha| \leq |\beta|$

   ograničen br. znakova

3. Kontekstno neovisna gramatika $G_2=(V,T,P,S)$

   $A \rightarrow \alpha$

   jedan znak s lijeve str,

4. Regularna gramatika $G_3=(V,T,P,S)$

   $A \rightarrow wB$ i $A \rightarrow w$ ili

   $A \rightarrow Bw$ i $A \rightarrow w$

   lijevo ili desno linearna

Automat

1. Turingov stroj $M_0=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$
2. Linearno ograničen stroj $M_1=(Q,\Sigma,\Gamma,\delta,q_0, ¢,\$,F)$
3. Potisni automat $M_2=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$
4. Konačni automat $M_3=(Q,\Sigma,\delta,q_0,F)$

Jezik

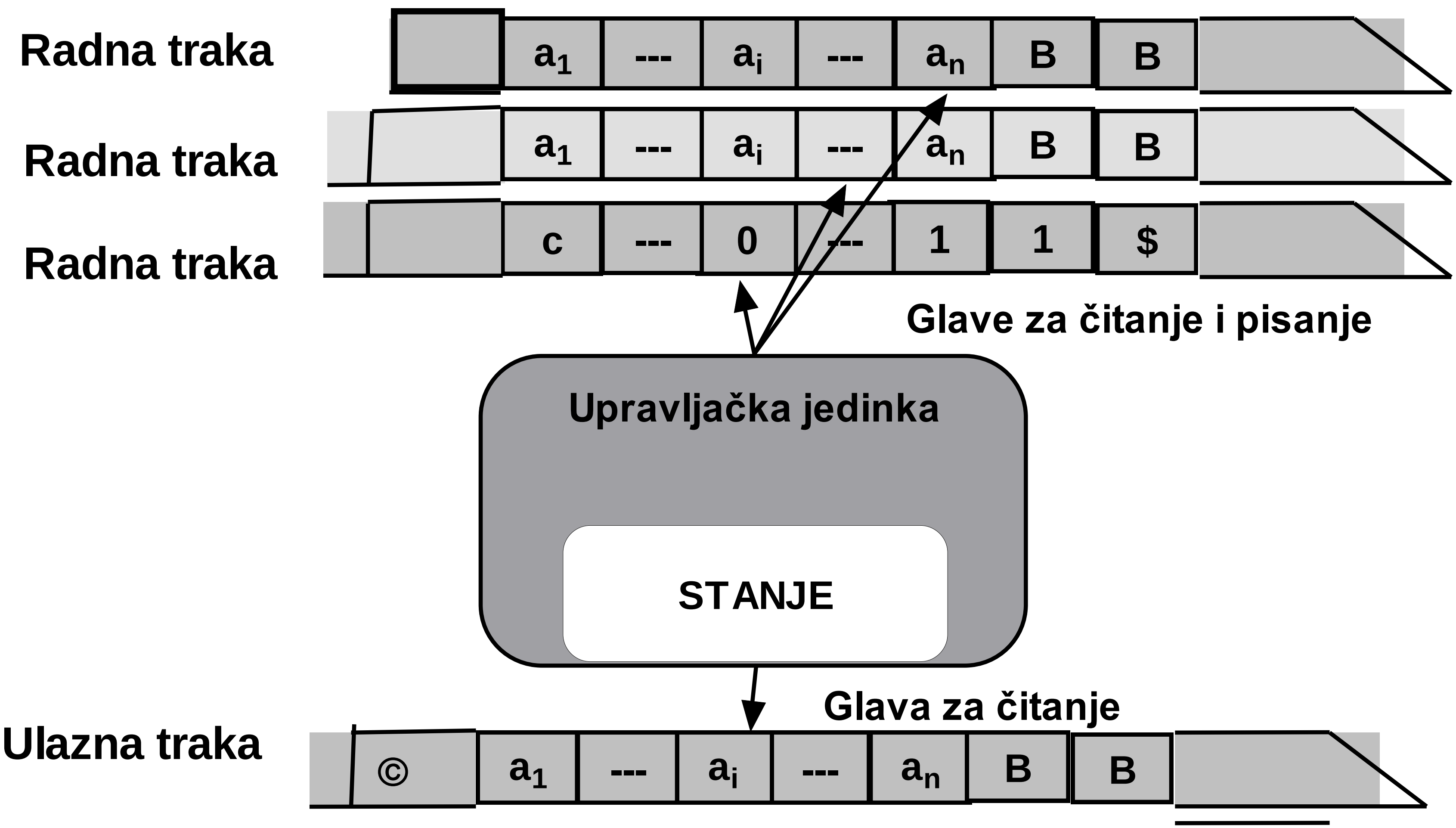
1. Rekurzivno-prebrojiv jezik $L_0= L(G_0)=L(M_0)$
2. Kontekstno ovisan jezik $L_1= L(G_1)=L(M_1)$
3. Kontekstno neovisan jezik $L_2= L(G_2)=L(M_2)$
4. Regularan jezik $L_3= L(G_3)=L(M_3)$

Složenost prihvaćanja jezika

- ovisi od veličine trake i vremena da automat prihvati jezik
- zbog hijerarhije jezika i automata TS je osnovni automat za ocjenu složenosti prihvaćanja svih klasa jezika
  - **veličina trake**: broj ćelija koje se tijekom rada koriste
  - **vrijeme**: broj pomaka glave TS-a
    - jedan pomak je jedna jedinica vremena

Prostorna složenost prihvaćanja jezika

- neizravan deterministički TS s k polubeskonačnih traka

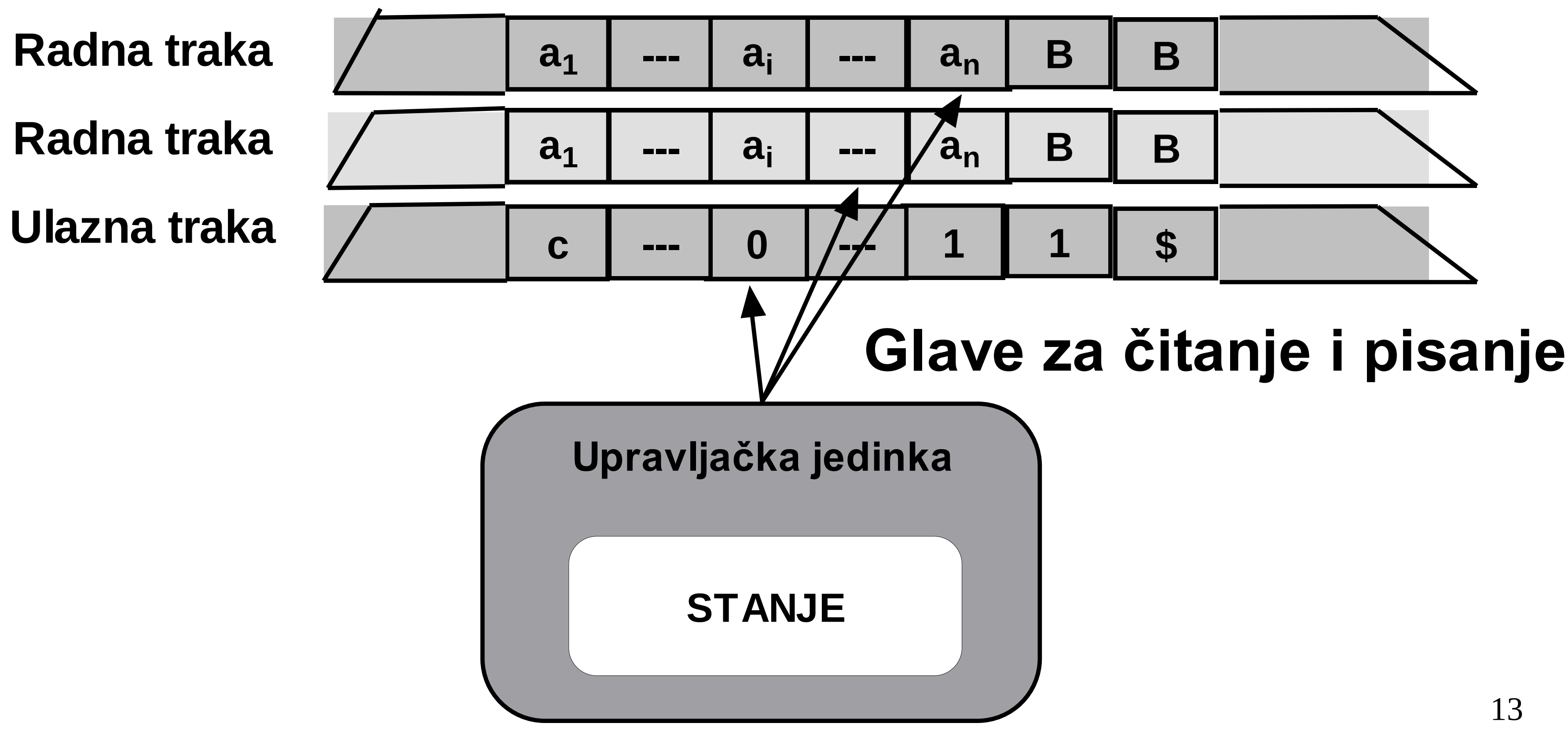


Prostorna složenost prihvaćanja jezika II

- ulaznu traku se samo čita
- duljina ulaznog niza je n
- k radnih traka je beskonačno na jednu stranu i na njih se čita i piše
- prostorna složenost S(n) određuje se na osnovu **samo jedne radne trake** i to one na kojoj je korišteno **najviše ćelija n**

Vremenska složenost prihvaćanja jezika

- deterministički TS s k dvostrano beskonačnih traka



Vremenska složenost prihvaćanja jezika II

- na sve trake (radne i ulazne) se čita i piše
- vremenska složenost T(n) određuje se pomoću broja pomaka glave za čitanje i pisanje n

Svojstva vremenske i prostorne složenosti

- broj traka ne utječe na prostornu ali utječe na vremensku složenost
- vremenska složenost povećava se povećanjem broja traka

Klase jezika

- ako jezik prihvaća nedeterministički TS jezik je nedeterminističke složenosti
- 4 klase jezika:
  - **DSPACE(S(n))** – jezici determinističke prostorne složenosti
  - **NSPACE(S(n))** – jezici nedeterminističke prostorne složenosti
  - **DTIME(T(n))** – jezici determinističke vremenske složenosti
  - **NTIME(T(n))** – jezici nedeterminističke vremenske složenosti

Primjena

- algoritmi koji se izvode na računalu
- imaju vremensku i prostornu složenost

Koji algoritam odabrati?

- mora biti:
  - razumljiv
  - jednostavna implementacija
  - jednostavno otklanjanje pogreški (debug)
  - efikasna iskorištenost računalnih resursa
  - brzina vs. prostor...
- jednokratna upotreba (troškovi razvoja)
- učestala upotreba (troškovi korištenja)

Cijena

- ukoliko algoritam radi često i s velikom količinom podataka isplati se potrošiti resurse (vrijeme i rad) na njegovo optimiranje
  - isplati se implementirati kompleksniji algoritam koji će raditi efikasnije (vremenski i prostorno)
- potrebno uvesti mjeru kompleksnosti algoritma, koja će ocijeniti njegove vremenske i prostorne potrebe

Vrijeme izvođenja programa

- ovisi od:
  - količine i vrste ulaznih podataka u program
  - kvalitete koda koju generira compiler
  - brzini i performansama računala (sklopovlja)
  - vremenskoj zahtjevnosti (kompleksnosti) algoritma

Primjer I

- sortiranje (najprije najmanji)

  2 1 3 1 5 8

  1 1 2 3 5 8

  – mjera kompleksnosti: broj elemenata koje sortiramo odnosno dužina liste
  – T(n) – vrijeme potrebno za izvođenje programa, koji na ulazu ima n podataka
  – T(n) = broj potrebnih instrukcija za izvršenje zadataka na idealnom računalu

Rješivost (izračunljivost) problema

- ako je problem izračunljiv *(decidable)*
  - moguće ga je riješiti (izračunati) - rješiv
- onda u praksi računalni algoritam za rješenje problema zahtijeva
  - prostor (memorijske kapacitete)
  - vrijeme (potrebno za izvođenje postupaka)
  - resurse (računalne resurse)

Vremenska kompleksnost

- određuje kompleksnost algoritma na osnovu potrebnog vremena za rješenje problema
- mjera vremenske kompleksnosti *(time complexity or runing time complexity)*
  - je maksimalan broj koraka M u kojima postupak obrađuje ulaz dužine n za rješenje problema
- računa se za
  - najgori slučaj *(worst-case, pesimistic)*
  - najbolji slučaj *(best-case, optimistic)*
  - prosjek

Ocjena vremenske kompleksnosti

- određivanje egzaktnog vremena izvođenja algoritma je kompleksno, pa i teško, zato se u praksi samo **ocjenjuje** vremenska kompleksnost izvođenja
- ocjena se izražava kao **O(n)** gdje n predstavlja gornju asimptotsku ocjenu reda funkcije f(n)
  - npr: ako ocijenimo vrijeme izvođenja programa s funkcijom $f(n)=6n^3+2n^2+2n+45$ onda je ocjena vremenske kompleksnosti algoritma $O(n^3)$

Primjeri vremenske kompleksnosti

- polinomska

  $f(n)=5n^4+2n^3+2n^2+22n+6$ **$O(n^4)$**

  $f(n)=6n^3+2n^2+2n+45$ **$O(n^3)$**

- logaritamska

  $f(n)=3n\log_2 n+ 3\log_2 n \log_2 5n$ **$O(n \log_2 n)$**

- eksponencijalna *(brut-force search)*
  - “neupotrebljivi” algoritmi

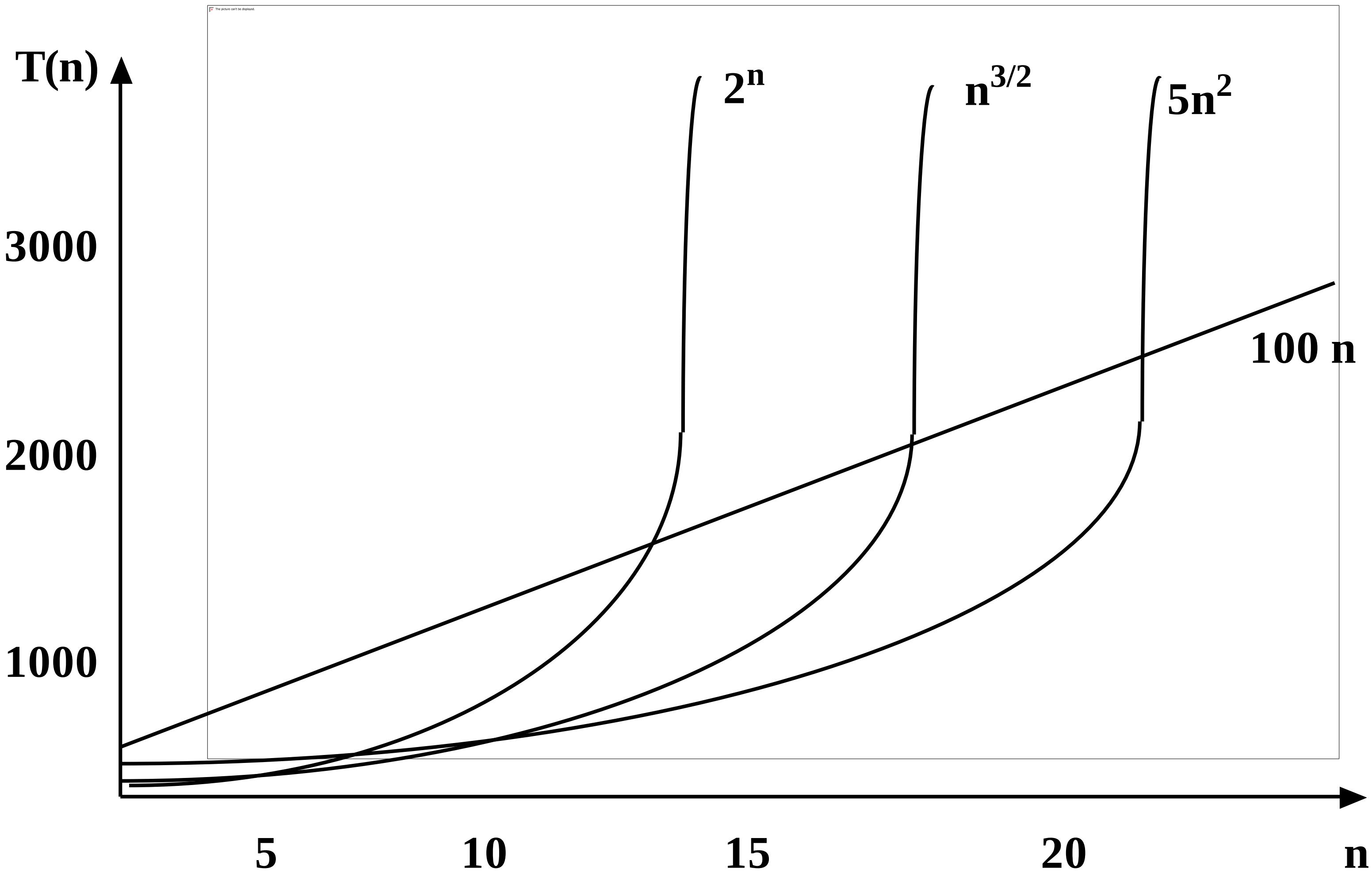
  $f(n)=2^n+ n^3$ **$O(2^n)$**

Primjer II

- $T(0) = 1$
- $T(1) = 4$
- $T(n) = (n+1)^2$
- $O(n^2)$

- $T(n) = 3n^3 + 2n^2$
- $O(n^3)$

- $T(n) = n \log(n) + n$
- $O(n \log(n))$



T(n)
3000
2000
1000
2n
n3/2
5n2
100 n
5
10
15
20
n

Dvije ocjene vremenske kompleksnosti

- **veliki O** - gornja asimptotska ocjena reda funkcije
  – ocjenjena funkcija nikad nije veća od ocjene O

- **mali o** – donja asimptotska ocjena reda funkcije
  – ocjenjena funkcija je veća od ocjene o

Klase

- **P klasa:** polinomska složenost algoritama
  - **odlučivi** u polinomskom vremenu na determinističkom TS s 1 trakom
- **NP klasa: ne mogu biti riješeni u** polinomskom vremenu (brut- force)
  - možda postoje bolji algoritmi ali ih još nismo pronašli
  - u polinomskom vremenu možemo provjeriti (verify) rješenje ali ga ne možemo odrediti (determine)
  - **provjerivi** u polinomskom vremenu

Intractable problems

- to su u principu rješivi problemi ali njihovo rješavanje zahtijeva toliko vremena i prostora da in ne koristimo u praksi

Prostorna kompleksnost

- određuje kompleksnost algoritma na osnovu potrebnog prostora (memorije) za rješenje problema
- mjera prostorne kompleksnosti *(space complexity)*
  - je maksimalan broj jedinica M dužine n koje postupak mora pročitati za rješenje problema

- prostorna kompleksnost određuje se pomoću Turingovog stroja (TS)

Praksa

- za svaki algoritam određuje se vremenska kompleksnost
  - brzina procesora ograničavajući faktor
  - **prihvatljivo:** logaritamska ili polinomska kompleksnost algoritma
  - **neprihvatljivo**: eksponencijalna kompleksnost algoritma
- rjeđe se određuje i prostorna kompleksnost
  - niske cijene memorijskih kapaciteta
  - jeftinije kupiti dodatnu brzu ili eksternu memoriju nego optimirati izvođenje algoritma

Umjesto zaključka

1. ukoliko će se program koristiti samo nekoliko puta: udio troškova pisanja i testiranja je značajan u ukupnom trošku
2. ukoliko različite osobe razvijaju i održavaju: algoritam efikasan ali kompleksan, raste trošak održavanja
3. poneki algoritmi rade brzo ali zahtijevaju puno prostora i zato koriste spore vanjske memorije i time postaju spori
4. kod numeričkih algoritama je točnost i stabilnost barem isto toliko važna kao i brzina

Literatura

- S. Srbljić: *Jezični procesori I + II*, Element, Zagreb, 2002.
- J.E. Hopcroft, J.D. Ullman: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, USA,1979.
- Michael Sipser, *Introduction to the Theory of Computation, second edition*, Course Technology, MIT, 2005.