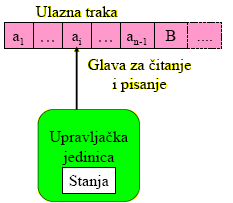
**21. TURINGOV STROJ**

**21.1. Formalna definicija Turingovog stroja**

* Rekurzivno prebrojivi jezici i Turingov stroj
* Osnovni modeli Turingovog stroja
* Donošenje odluka Turingovog stroja
* Formalna definicija Turingovog stroja

Rekurzivno prebrojivi jezici se zasnivaju na Turingovom stroju. Jezik jest rekurzivno prebrojiv ako i samo ako postoji Turingov stroj koji ga prihvaća. Time je definirana istovjetnost RPJ i TS: za bilo koji rekurzivno prebrojiv jezik moguće je izgraditi Turingov stroj koji ga prihvaća.

Osnovna primjena TS je prihvaćanje jezika. Obzirom da može pisati po traci, TS se koristi za generiranje jezika i za računanje cjelobrojnih funkcija.



TS nakon čitanja piše novi znak na traku koja je neograničena s desne strane. Glava se pomiče lijevo i desno. Na početku, niz w je upisan krajnje lijevo, a ostatak je popunjen s B.

Upravljačka jedinica donosi odluku o:

* promjeni stanja
* upisu znaka na traku
* pomaku glave za čitanje

Odluku donosi na osnovu stanja i znaka na traci.

Turingov stroj formalno zadajemo sedmorkom TS = (Q, , , , q0, B, F) gdje su:

* **Q** je konačan skup stanja s početnim stanjem **q0**
* ****je konačan skup (ulaznih) znakova trake bez praznog znaka **B**
* ****je konačan skup znakova trake s praznim znakom **B**
* ****funkcija prijelaza Q Q {L,R}
* **F** Q podskup prihvatljivih stanja

Funkcija **(q, v) = (p, z, W)** na osnovu stanja **q** i pročitanog znaka **v** određuje sljedeće stanje **p**, znak za upis na traku **z** i pomak **W**.

**21.2. Prihvaćanje jezika Turingovim strojem**

* Konfiguracija Turingovog stroja
* Prihvaćanje niza
* Prihvaćanje jezika
* Rekurzivni i rekurzivno prebrojivi jezici

Rad TS-a opisujemo konfiguracijom **α1qα2**sa značenjem „lijevi dio trake – stanje – desni dio trake“. TS čita krajnje lijevi znak niza **α2**. Oznake za q se moraju razlikovati od oznaka za α.

Pozicija glave se numerira:

X1 X2 … Xi-1 q Xi Xi+1 … Xn-1 Xn

Lijevo od X1 nema trake, nema lijevog pomaka. Desno od Xn je sve prazno, samo B.

Za **i>1** i **(q, Xi) = (p, Y, L)** TS promijeni konfiguraciju:

X1 X2 … Xi-1 q Xi Xi+1 … Xn-1 Xn  X1 X2 … Xi-2 p Xi-1 Y Xi+1 … Xn

Za **(q, Xi) = (p, Y, R)** TS promijeni konfiguraciju:

X1 X2 … Xi-1 q Xi Xi+1 … Xn-1 Xn  X1 X2 … Xi-1 Y p Xi+1 … Xn

Izraz  znači da TS iz konfiguracije J prelazi u K u **m** koraka.

Automat M prihvaća jezik koji ga dovodi u prihvatljivo stanje:

L(M) = { w| w\* i  ; pF }

Ako niz nije prihvatljiv, TS ne mora stati.

TS prihvaća klasu **rekurzivno prebrojivih** jezika (ime dano na osnovu svojstva da TS ispisuje (nabraja) sve nizove, a termin „rekurzivno“ u današnjem značenju). Kontekstno neovisni jezici su pravi podskup **rekurzivno prebrojivih jezika**. Za niz koji nije član jezika, TS ne mora stati. Za jezike gdje TS uvijek stane, kaže se da su samo **rekurzivni**. Klasa rekurzivnih jezika je podskup klase prebrojivih jezika.

**21.3. Cjelobrojna aritmetika Turingovim strojem**

* Osnovna sintaksa zapisa
* Rekurzivne funkcije i jezici
* Primjer

Turingov stroj se koristi za računanje cjelobrojnih funkcija. Na traci se koristi notacija **0n10m1**…

Niz **n** nula ima vrijednost **n**, niz **m** nula vrijednost **m**. Nizovi nula odijeljeni su znakom 1. To je sustav s bazom s = 1 (rimska notacija).

Turingov stroj izračunava **parcijalno rekurzivne** funkcije. ParRF i RPJ su ekvivalentni, Turingov stroj se ne mora zaustaviti. Ako se TS uvijek zaustavi, radi se o **potpuno rekurzivnim funkcijama.** PotRF i RekJ su ekvivalentni, TS se uvijek zaustavi.

Primjer: TS računa **m-n**, ako je m<n razlika je **0**.

Za niz 0m10n:

* TS skida 0 lijevog broja i 0 desnog broja (upisuje 1)
* Ako je m > n ostane lijevo 0m-n
* Ako je m = n ne ostane niti jedna nula
* Ako je m < n ostanu nule desno, pa ih TS mora ukloniti

Gradi se TS ({q0,...,q6}, {0, 1}, {0, 1, B}, , q0, B, )

(q0, 0) = (q1, B, R) početna zamjena

(q1, 0) = (q1, 0, R) (q1, 1) = (q2, 1, R) traži desni broj

(q2, 1) = (q2, 1, R) (q2, 0) = (q3, 1, L) zamijeni i idi natrag

(q3, 0) = (q3, 0, L) (q3, 1) = (q3, 1, L) (q3, B) = (q0, B, R) idi natrag

Ako je **m>n**, briši jedinice i vrati nulu:

(q2, B) = (q4, B, L) okreni natrag!

(q4, 1) = (q4, B, L) briši jedinice

(q4, 0) = (q4, 0, L) (q4, B) = (q6, 0, R) idi lijevo i vrati nulu, STOP

Ako je **m<n**, briši:

(q0, 1) = (q5, B, R) briši jedinice desno

(q5, 1) = (q5, B, R) (q5, 0) = (q5, B, R) briši nule i jedinice desno

(q5, B) = (q6, B, R) sve obrisano, STOP

Za niz 0100 vrijedi 1-2 = 0.

q00100 Bq1100 B1q200 Bq3110 q3B110 Bq0110 BBq510 BBBq50 BBBBq5 BBBBBq6

**22. SVOJSTVA TURINGOVOG STROJA**

**22.1. Višekomponentna stanja i znakovi trake**

* Višekomponentne oznake stanja
* Višekomponentni znakovi trake
* Primjeri

Višekomponentne oznake stanja olakšavaju izradu TS-a. Uz osnovno stanje bilježi se informacija koju ono nosi. Koriste se uglate zagrade: [q, a, b, …] – znači da u stanju q nosimo parametre a, b, …

Stanje je **upravljačka komponenta** – može biti samo jedna upravljačka komponenta.

Informacija je **radna komponenta** – radnih komponenti može biti više.

Primjer višekomponentnih stanja: želimo izgraditi TS koji prihvaća nizove kod kojih se prvi znak ne ponavlja unutar niza.

Koristimo višekomponentno stanje [q, a]. Upravljačka komponenta ima dva stanja:

q0 – prije čitanja prvog znaka

q1 – prvi znak je pročitan

Radna komponenta koristi se za pamćenje prvog znaka i može poprimiti vrijednost 0, 1 i B. Imamo svega šest stanja: [q0, B], [q0, 0], [q0, 1], [q1, B], [q1, 0], [q1, 1].

Gradimo TS M = (Q, {0, 1}, {0, 1, B}, , [q0, B], B, [q1, B]}).

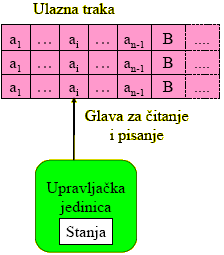
Funkcija prijelaza je:

([q0, B], 0) = ([q1, 0], 0, R) ([q0, B], 1) = ([q1, 1], 1, R)

([q1, 0], 1) = ([q1, 0], 1, R) ([q1, 1], 0) = ([q1, 1], 0, R)

([q1, 0], B) = ([q1, B], B, L) ([q1, 1], B) = ([q1, B], B, L)

Drugi primjer: pomak znakova za n mjesta u desno: koristimo n radnih komponenti. U svakom koraku prelazimo u novo stanje tako da rotiramo radne komponente.



Znakovi mogu imati više komponenti.

Za konačni skup komponenti, rad je ekvivalentan

Turingovom stroju.

Komponente zapišemo u posebne tragove trake.

Ukupni znak trake je kombinacija komponenti.

Tragove možemo koristiti za posebne namjene, npr.

pomoćni trag može se koristiti za označivanje ranije

ispitanih znakova, što je korisno za TS koji prihvaćaju jezike gdje se dijelovi niza ponavljaju ili gdje se dijelovi niza uspoređuju.

**22.2. Proširenja i pojednostavljenja Turingovog stroja**

* Proširenja trake, neizravni TS
* Stogovni stroj i stroj s brojilima
* Ograničenja stanja i trake
* Univerzalni TS

Proširenja osnovnog modela:

* TS s dvostranom beskonačnom trakom
* TS s višestrukim trakama
* Nedeterministički TS
* TS s višedimenzionalnim ulaznim poljem
* TS s više glava za čitanje i pisanje
* Neizravni TS

Neizravni TS je model automata koji se koristi u istraživanju prostorne složenosti prihvaćanja jezika i prostorne složenosti računanja cjelobrojnih funkcija. Neizravni TS ima više radnih traka i jednu ulaznu traku. Ulaznu traku moguće je samo čitati, te je nije moguće pomaknuti izvan područja omeđenog graničnicima.

Neizravni TS je istovjetan TS-u s višestrukim trakama. Rad glava za čitanje i pisanje TS s višestrukim trakama nije ograničen, što omogućuje da TS s višestrukim trakama simulira rad neizravnog TS-a. Dodavanjem jedne trake neizravnom TS-u i prijepisom niza s ulazne trake na dodatnu traku, te uporabom dodatne trake kao ulazne, omogućuje da neizravni TS simulira rad TS-a s višestrukim trakama.

Pojednostavljenja osnovnog modela:

* Stogovni stroj
* Stroj s brojilima
* TS s ograničenim brojem stanja i znakova trake
* Univerzalni TS Mu

Stogovni stroj jest deterministički Turingov stroj sa jednom ulaznom trakom i više stogova. Ograničenje mu je da je ulaznu traku moguće samo čitati. Stog je posebna radna traka pojednostavljenih funkcija prijelaza: ako se glava pomakne u lijevo, u ćeliju se upisuje oznaka prazne ćelije (zato će sve ćelije desno od glave za čitanje i pisanje biti prazne).

Kod stroja s brojilima umjesto dva stoga se koriste četiri brojila. Skup znakova stoga se ograniči na dva znaka: oznaku dna stoga X i oznaku prazne ćelije B.

Ako istodobno ograničimo broj znakova trake, broj traka i broj stanja, onda je ograničen broj Turingovih strojeva koje možemo izgraditi.

Ako ne ograničimo broj znakova trake, onda su nam dovoljna tri stanja i jedna traka za prihvaćanje bilo kojeg rekurzivno prebrojivog jezika.

Ako ne ograničimo broj stanja, onda je dovoljno ograničiti broj znakova trake na {0, 1, B}.

Primjenom univerzalnog TS Mu moguće je simulirati rad bilo kojeg TS M s jednom trakom. Univerzalni TS ima 3 trake:

* Prva traka: zapisuju se funkcije prijelaza TS M i niz w (označavanje na način <M, w>)
* Druga traka: sadržaj druge trake Mu-a simulira sadržaj trake proizvoljnog TS M.
* Treća traka: zapisuje se stanje TS M.

Univerzalni TS Mu prepiše niz *w* s prve trake na drugu traku, simulira rad TS M primjenom prijelaza zapisanih na prvoj traci i zapisuje stanje TS M na treću traku. Ukoliko za stanje zapisano na trećoj traci i za znak zapisan na drugoj traci nema daljnjih prijelaza zapisanih na prvoj traci, zaustavlja se rad univerzalnog TS Mu. Ukoliko je stanje zapisano na trećoj traci prihvatljivo, tada univerzalni TS Mu prelazi u prihvatljivo stanje.

Univerzalni TS Mu prihvaća niz <M, w> **ako i samo ako** TS M prihvaća niz *w*.

**22.3. Generiranje jezika Turingovim strojem**

* Struktura TS za generiranje jezika
* Prihvaćanje jezika generiranog TS
* Generiranje jezika prihvaćenog TS
* Jednostavni i složeni TS

Koristi se TS s višestrukim trakama. Jedna traka je izlazna, jednom upisani znak se ne može mijenjati. Glava za čitanje i pisanje izlazne trake miče se u desno. Nizovi jezika G(M) ispisuju se na izlaznu traku, kraj je #. Ako TS stane, G je konačan, inače je beskonačan. TS generira klasu rekurzivno prebrojivih jezika. Jezik je rekurzivno prebrojiv samo ako postoji TS koji ga generira.

**Prihvaćanje jezika generiranog TS:**

Za TS M1 koji generira G(M1) moguće je izgraditi TS M2 koji prihvaća L(M2) = G(M1). Gradi se TS M2 koji ima jednu traku više od TS M1 – to je ulazna traka.

TS M2 generira neki niz: Ako je niz istovjetan zadanom, niz se prihvaća. Inače, generira se sljedeći niz.

**Generiranje jezika prihvaćenog TS:**

Za TS M2 koji prihvaća L(M2) moguće je izgraditi TS M1 koji generira G(M1) = L(M2). Gradi se TS M1:

* **Jednostavan**, generira rekurzivni jezik, uvijek stane
* **Složen**, generira rekurzivno prebrojiv jezik, nekad ne stane

Za rekurzivne jezike **jednostavni TS** ispisuje sve nizove na radnu traku. Simulira rad M2 i provjeri novi niz. Ako je niz prihvatljiv, kopira ga sa radne trake na izlaznu.

Za rekurzivno prebrojive jezike **jednostavni TS** nekad ne stane, stoga ne može pokrenuti ispitivanje sljedećeg niza. Zato se koristi **složeni TS**.

**Složeni TS** generira par i,j rastućim redoslijedom i+j, i. Nakon toga generira niz duljine *i* na radnu traku. Na kraju se provjerava prihvatljivost niza u *j* koraka. Tako se izbjegne situacija da se TS ne zaustavi.

**22.4. Istovjetnost rekurzivnog jezika i kanonskog slijeda**

* Definicija kanonskog slijeda
* Istovjetnost RekJ i kanonskog slijeda

U kanonskom slijedu kraći nizovi su ispred dužih. Redoslijed nizova iste duljine određuje se po numeričkoj vrijednosti kao broj s bazom jednakom broju znakova.

Rekurzivne jezike moguće je generirati kanonskim slijedom. Za generiranje se može koristiti jednostavan TS. Neka sve nizove generira kanonskim slijedom na radnu traku – tada će na izlaznoj traci prihvaćeni nizovi biti isto poredani.

Jezik G(M1) kojeg je moguće generirati kanonskim slijedom jest rekurzivni jezik.

Neka TS M1 generira jezik G(M1) kanonskim slijedom, i neka TS M2 prihvaća jezik

G(M2) = G(M1) i koji uvijek stane.

TS M2 simulira rad TS M1:

* M2 generira niz kanonskim slijedom sve do wi ili wj.
* Generira li wi – niz je prihvatljiv i TS stane.
* Generira li wj, j>i, wi sigurno nije prihvatljiv i automat stane.

Ako je jezik G(M1) beskonačan, TS M2 uvijek stane. Ako je jezik G(M1) konačan i ako se ispituju nizovi iza svih nizova od L(M2), TS M2 nikad ne stane.

Nije moguće izgraditi TS za opći slučaj konačnog jezika. Moguće je izgraditi pojedinačne TS.

**23. GRAMATIKA NEOGRANIČENIH PRODUKCIJA**

**23.1. Formalna specifikacija gramatike neograničenih produkcija**

* Oblik produkcije i vrsta gramatike
* Definiranje jezika generiranog gramatikom

Za kontekstno neovisne gramatike na lijevoj strani imaju samo jedan nezavršni znak. Za regularne gramatike, dodatno, još su lijevo linearne i desno linearne.

Za gramatike neograničenih produkcija 

* i su nizovi završnih i nezavršnih znakova
* niz ne smije biti prazan, 

Gramatika neograničenih produkcija zove se **gramatika tipa 0.**

Gramatika neograničenih produkcija je četvorka: G = (V, T, P, S).

Za produkciju definira se relacija  : .

Refleksivno i tranzitivno okruženje relacije  je  .

G = (V, T, P, S) generira jezik: L(G) = { w| w\* ; S  w }koji pripada klasi rekurzivno prebrojivih jezika.

**23.2. Konstrukcija TS iz GNP**

* Gramatika, jezik i TS
* Postupak rada automata
* Simulacija gramatike G

G = (V, T, P, S) - ako je gramatika neograničenih produkcija i generira L(G) tada je L(G) rekurzivno prebrojiv jezik. Po definiciji, L(G) je rekurzivno prebrojiv ako postoji TS koji ga prihvaća: L(M) = L(G).

Gradi se nedeterministički TS (NTS) s dvije trake koji simulira rad gramatike G.

Postupak:

* Na prvu traku zapiše se niz w
* Na drugu traku zapiše se početni nezavršni znak gramatike S
* Tijekom simulacije, na drugu traku ispisuju se nizovi α
* TS uspoređuje nizove α s nizom w i eventualno ga prihvati (ako je niz α jednak w)

Simulacija gramatike G se izvodi:

1. TS nedeterministički izabere mjesto *i* u nizu α druge trake, za sve vrijednosti od *i*
2. Po svim *i*, TS nedeterministički izabere produkciju , dakle sve produkcije
3. Po svim *i*, , ako je na mjestu *i* zamijeni se s
4. Niz generiran na drugoj traci uspoređuje se s nizom w koji je zapisan na prvoj traci:

* Ako su nizovi jednaki, TS se zaustavlja i prihvaća w
* Ako nisu jednaki, TS nastavlja od koraka 1

**23.3. Konstrukcija GNP iz TS**

* Pristup definiranju gramatike
* Veza TS i gramatike\*

Ako TS prihvaća RPJ L(M) postoji gramatika neograničenih produkcija GNP koja generira L(G) = L(M).

Gradi se G = (V, T, P, S) koja simulira rad TS M = (Q, , , , q0, B, F). Gramatika G generira redom međunizove znakova koji predstavljaju konfiguracije TS M.

\*Nezavršni znak q u generiranom međunizu je oznaka stanja TS M. Početna konfiguracija TS M se simulira međunizom u kojem su nezavršni znakovi oblika [ai,ai] (ai∈Σ). Prva komponenta čuva znakove niza tijekom simulacije na temelju kojih G generira niz nezavršnih znakova ako i samo ako TS prihvaća taj isti niz. Druga komponenta predstavlja znakove trake. \*

**24. SVOJSTVA REKURZIVNIH JEZIKA**

**24.1. Zatvorenost RekJ i RPJ obzirom na uniju**

* Unija rekurzivnih jezika
* Unija rekurzivno prebrojivih jezika

Unija rekurzivnih jezika jest rekurzivni jezik.

TS M kao serijski spoj TS M1 i TS M2:

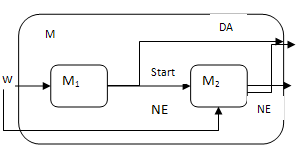
* TS M1 prihvaća RekJ L1
* TS M2 prihvaća RekJ L2

Ako M1 stane i prihvati niz, M stane i prihvati niz.

Ako M1 stane i prihvati niz, starta se M2.

Ako M2 stane i prihvati niz, M stane i prihvati niz.

Ako M2 stane i ne prihvati niz, M ne prihvati niz.



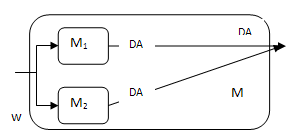
Unija rekurzivno prebrojivih jezika jest rekurzivno prebrojivi jezik.

TS M kao paralelni spoj TS M1 i TS M2:

* TS M1 prihvaća RPJ L1
* TS M2 prihvaća RPJ L2

Ako M1 stane i prihvati niz, M stane i prihvati niz.

Ako M2 stane i prihvati niz, M stane i prihvati niz.



**24.2. Zatvorenost RekJ i RPJ obzirom na komplement**

* Komplement rekurzivnih jezika
* Komplement rekurzivno prebrojivih jezika

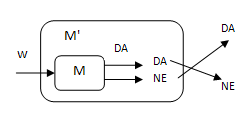
Komplement rekurzivnog jezika jest rekurzivni jezik.

TS M kao komplement TS M1 (okrenuti izlazi):

* TS M1 prihvaća RekJ L1

Ako M1 stane i prihvati niz, M stane i **ne** prihvati niz.

Ako M1 stane i **ne** prihvati niz, M stane i prihvati niz.



Ako su jezik L i njegov komplement rekurzivno prebrojivi, tada su oba jezika **rekurzivni**.

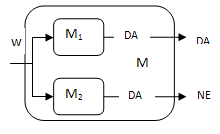
TS M kao paralelni spoj TS M1 i TS M2:

* TS M1 prihvaća RPJ L1
* TS M2 prihvaća RPJ L2 = L1c

Ako M1 stane i prihvati niz, M stane i prihvati niz.

Ako M2 stane i prihvati niz, M stane i **ne** prihvati niz.

M uvijek stane, dakle L i Lc su rekurzivni.



Za L i Lc vrijedi jedno od tri svojstva:

1. Oba jezika L i Lc su rekurzivna
2. Oba jezika L i Lc nisu rekurzivno prebrojivi
3. Ako je L rekurzivno prebrojiv, ali ne i rekurzivan, Lc sigurno nije rekurzivno prebrojiv

**25. IZRAČUNLJIVOST I ODLUČIVOST**

**25.1. Izračunljivost**

* Definicija izračunljivosti
* Church-Turingova hipoteza
* Problem kodiranja: RAM, dijagonalni jezik

Problem jest izračunljiv

* Prihvaćanje jezika
* Računanje cjelobrojnih funkcija
* Generiranje jezika

ako postoji automat koji korak po korak (mehanički) rješava zadani problem.

Ne nameću se nikakva ograničenja na broj koraka niti na veličinu spremnika. Ne traži se da postupak ikad stane.

Church-Turingova hipoteza:

Na temelju intuitivne definicije izračunljivosti, izračunljive funkcije su parcijalno rekurzivne funkcije. Pokazuje se samo prikladnost ove hipoteze. Parcijalno rekurzivne funkcije su izračunljive jer je za njihovo računanje moguće izgraditi TS. TS računa parcijalno rekurzivne funkcije „mehaničkim“ putem korak po korak primjenom zadanih funkcija prijelaza. Budući da su poznati svi fomalizmi (kao što su λ- račun, Post sustavi, opće rekurzivne funkcije, itd.) istovjetni klasi parcijalno rekurzivnih funkcija, sve izračunljive funkcije su ujedno parcijalno rekurzivne. Primjena TS omogućuje ugradnju bilo kojeg algoritma, te simulaciju bilo kojeg programskog jezika suvremenih računala.

**Simulacija RAM računala s TS**: računalo zasnovano na memoriji sa slobodnim pristupom moguće je simulirati primjenom Turingova stroja (TS). Primjena TS omogućuje ugradnju bilo kojeg algoritma, te simulaciju bilo kojeg programskog jezika suvremenih računala.

* Memorija sa slobodnim pristupom
* Registri

Za simuliranje koristimo TS s višestrukim trakama. Adrese i sadržaji mem. Ćelija spremaju se na traku u obliku: # 0 \* v0 # 1 \* v1 # 10 \* v2 # \_ \_ \_ # k \* vk, gdje je *i* adresa mem. Ćelije, a *vi* sadržaj ćelije. Vrijednosti ćelija se zapisuju u binarnom obliku. Dva su načina simuliranja izvođenja naredbi:

* Korištenje funkcije prijelaza TS
* Korištenje mikrokoda

**Neizračunljivost dijagonalnog jezika**: ako je moguće izgraditi TS M koj prihvaća L, onda je na temelju definicije izračunljivosti i Church-Turingove hipoteze jezik L izračunljiv (rekurzivno prebrojivi jezici). Ako nije moguće izgraditi, jezik L nije izračunljiv (dijagonalni jezik Ld).

**25.2. Odlučivost**

* Definicija odlučivosti
* Univerzalni TS i jezik\*

Rekurzivni jezici su **odlučivi** jer ih prihvaćaju TS koji uvijek stanu. Rekurzivni jezici su **izračunljivi** i **odlučivi**.

Rekurzivno prebrojivi nisu odlučivi jer ne postoji TS koji će uvijek stati. Ako w nije u jeziku, moguće je da TS nikad ne stane. Ako TS ne stane, nema odluke da w ne pripada jeziku.

Rekurzivno prebrojivi jezici su izračunljivi i **neodlučivi**.

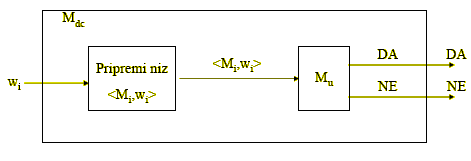
**Univerzalni** jezik Lu je primjer rekurzivno prebrojivog jezika, dakle izračunljivog jezika koji **nije** odlučiv.

Univerzalni TS ima tri trake. Prva traka je ulazna traka i na nju se zapiše niz <M,w> koji se kodira na način: 111kod111kod2…111w (kodp 🡪 kod funkcije prijelaza TS M). Sadržaj druge trake simulira sadržaj trake TS M, dok se na treću traku zapisuje stanje qi TS M nizom 0i. Univerzalni TS Mu prihvaća **univerzalni jezik** Lu koji se definira:

Lu= { <M,w> | TS M prihvaća niz w}.

\*Budući da je pokazano da je za univerzalni jezik Lu moguće izgraditi TS Mu, jezik Lu je rekurzivno prebrojiv, odnosno izračunljiv. Univerzalni jezik Lu nije rekurzivan, odnosno nije odlučiv. Na temelju svojstava dijagonalnog jezika Ld i njegova komplementa Lcd pokazuje se svojstvo nerekurzivnosti, odnosno neodlučnosti univerzalnog jezika Lu.

Komplement dijagonalnog jezika Ld ={wi|TS Mi ne prihvaća niz wi} jednak je Lcd={wi|TS Mi prihvaća niz wi}. Pošto dijagonalni jezik Ld nije rekurzivno prebrojiv, na temelju svojstva komplementa zaključuje se da njegov komplement Lcd nije rekurzivan.



Pretpostavimo da je jezik Lu rekurzivan, odnosno pretpostavimo da univerzalni TS Mu koji prihvaća jezik Lu uvijek stane za bilo koji ulazni niz. Slika pokazuje konstrukciju TS Mdc koji prihvaća jezik Lcd:

1. Ako je na ulazu TS Mdc niz wi, onda na temelju znakova niza wi TS Mdc pripremi niz

<Mi, wi>

2. TS Mdc zapiše niz <Mi, wi> na prvu traku univerzalnog TS Mu i pokrene njegovo izvođenje.

Ako je Lu rekurzivan i ako TS Mu uvijek stane, onda TS Mdc uvijek stane za bilo koji ulazni niz. Na temelju činjenice da TS Mdc uvijek stane za bilo koji ulazni niz, zaključuje se da je jezik Lcd rekurzivan, što je suprotno činjenici da jezik Lcd nije rekurzivan. Budući da pretpostavka da je jezik Lu rekurzivan dovodi do suprotnosti, zaključuje se da jezik Lu nije rekurzivan.\*

**26. KONTEKSTNO OVISNI JEZICI**

**26.1. Definicija kontekstno ovisnog jezika i gramatike**

* Kontekstno ovisni jezik i gramatika
* Oblik produkcija
* Primjer

Jezik je **kontekstno ovisan** ako i samo ako postoji **kontekstno ovisna** gramatika koja ga generira. Time je definirana istovjetnost KOJ i KOG. Za bilo koji kontekstno ovisni jezik moguće je izgraditi kontekstno ovisnu gramatiku koja ga generira i obrnuto.

Klasa KOJ je pravi podskup RekJ.

G = (V, T, P, S) ima oblik produkcija ograničenih:

* Broj znakova s desne strane je veći ili jednak broju znakova s lijeve strane
* Niz mora biti neprazan niz

Tada je G kontekstno ovisna gramatika. Naziv dolazi zbog oblika produkcija: 1A212, neprazan. Zamjena A moguća je samo ako je ispunjen „kontekst“ 1/2.

Primjer: Zadana je gramatika G = ({A, B, C, D, E}, {a}, P, S) neograničenih produkcija:

1. S CaB 2) Ca  aaC 3) CB DB 4) CB E

5) aD Da 6) AD  AC 7) aE  Ea 8) AE ε

Produkcije 4) i 8) ne zadovoljavaju zahtjeve KOG jer je na desnim stranama tih produkcija manji broj znakova nego na lijevim. Za zadani jezik L = {a 2i | i>0} moguće je izgraditi KOG G' za koju vrijedi L(G') = L(G).

Budući da desne strane produkcija KOG moraju biti dulje ili jednako dugačke kao lijeve strane, znakove grupiramo u složene nezavisne znakove umjesto da ih označavamo jedinstvenim nezavršnim znakovima. U skupu nezavisnih znakova su: [ACab], [Aa], [ACa], [ADa], [AEa], [Ca], [Da], [Ea], [aCB], [CaB], [aDB], [aE], [DaB] i [aB].

Dobijemo sljedeće produkcije:

1. S → [ACaB]

2. [Ca]a → aa[Ca]

[Ca][aB] → aa[CaB]

[ACa]a → [Aa]a[Ca]

[ACa][aB] → [Aa]a[CaB]

[ACaB] → [Aa][aCB]

[CaB] → a[aCB]

3. [aCB] → [aDB]

4. [aCB] →[aE]

5. A[Da] →[Da]a

[aDB] →[DaB]

[Aa][Da] →[ADa]a

a[DaB] →[Da][aB]

[Aa][DaB] →[ADa][aB]

6. [ADa] →[ACa]

7. A[Ea] →[Ea]a

[aE] →[Ea]

[Aa][Ea] →[AEa]a

8. [AEa] →a

Gramatika G' generira niz *aa* na sljedeći način:

S => [ACaB] => [Aa][aCB] => [Aa][aE] => [Aa][Ea] => [AEa]a => aa

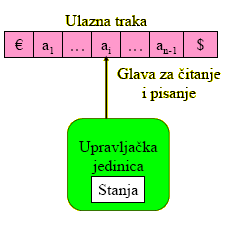
Gramatika G generira niz *aa* na sljedeći način:

S => ACaB => AaaCB => AaaE => AaEa => AEaa => aa

**26.2. Linearno ovisni automat**

* Model LOA
* Formalna definicija LOA
* Naziv LOA

Posebnim znakovama € i $ ograničava se traka. Zabranjuje se pomak glave izvan označenog dijela. To je nedeterministički TS koji koristi samo dio trake s nizom w.



Formalno se LOA zadaje osmorkom: LOA = (Q, , , , q0, €, $, F).

LOA M prihvaća jezik L(M) = { w| w\ {€,$})\* i q0€w$qqF }

Zahtjeva se da LOA stane kad prihvati w (slično TS). € i $ nisu dio ulaznog niza. Ako je moguće izgraditi deterministički LOA, DLOA kaže se da je L deterministički kontekstno ovisni jezik. Pod LOA podrazumijeva se nedeterministički LOA.

Naziv LOA nastao je na osnovi svojstva: Ako je duljina radne trake TS M ograničena za bilo koji w linearnom funkcijom f(w), moguće je izgraditi istovjetni TS M' koji koristi samo onaj dio trake koji je napisan w.

Vrijedi svojstvo sažimanja radne trake TS za konstantni faktor.

**26.3. Konstrukcija linearno ovisnog automata iz KOG**

* Postupak izgradnje
* Konstrukcija LOA iz KOG

Ako KOG G = (V, T, P, S) generira KOJ L(G), L, moguće je izgraditi LOA M tako da vrijedi L(M) = L(G). Postupak je sličan gradnji TS za GNP:

* LOA M koristi dva traga ulazne trake
* U gornji trag LOA napiše se niz €w$
* Na početak drugog traga LOA M zapiše se početni nezavršni znak S
* Za prazni niz  LOA stane i ne prihvati niz

Za neprazni w LOA simulira gramatiku G: na donjem tragu ispisuje nizove gramatike G i uspoređuje generirani niz sa zadanim nizom w.

Obzirom da su produkcije iz KOG s lijevom stranom kraćom od desne, LOA nikad neće koristiti dio trake duži od |w|. Ako se generira niz |w|<||, rad LOA se prekida jer se sigurno ne može dobiti traženi niz w.

**27. SVOJSTVA KONTEKSTNO OVISNIH JEZIKA**

**27.1. Unija, nadovezivanje i Kleenee**

* Zatvorenost KOJ po uniji
* Zatvorenost KOJ po nadovezivanju
* Zatvorenost KOJ po Kleenee

**Unija** KOJ jest KOJ. Neka KOG G1 = (V1, T1, P1, S1) i G2 = (V2, T2, P2, S2) generiraju KOJ L(G1) i L(G2), V1V2 = 

Gradimo G3 = (V3, T3, P3, S3) koja generira L(G3) = L(G1) L(G2):

* V3 = V1V2{S3}; S3 V1V2
* T3 = T1 T2
* P3 = P1 P2 {S3 S1 | S2}

Postupak je sličan gradnji KNG za uniju: nakon prijelaza u S1|S2 dalji rad primjenjuje produkcije samo jedne gramatike.

**Nadovezivanje** KOJ jest KOJ. Neka KOG G1 = (V1, T1, P1, S1) i G2 = (V2, T2, P2, S2) generiraju KOJ L(G1) i L(G2), V1V2 = 

Gradimo G4 = (V4, T4, P4, S4) koja generira L(G4) = L(G1)L(G2):

* V4 = V1V2{S4}; S4 V1V2
* T4 = T1 T2
* P4 = P1 P2 {S4 S1S2}

Postupak je sličan gradnji KNG za nadovezivanje: nakon prijelaza u S1S2 po dijelovima se primjenjuju produkcije samo jedne gramatike.

Problem: T1 T2 pa u nizu 

Postoji mogućnost da je sufiks od γ α1 i prefiks od δ Aα2 te da postoji produkcija

1A2 12. Primjenom te produkcije dobije se niz koji **nije** član jezika L(G4). Uvodimo ograničenje: na lijevoj strani produkcija su isključivo nezavršni znakovi čime se dobije gramatika kojoj su sa lijeve strane samo nezavršni znakovi i V1V2 = Slijedi da spajanjem međunizova ne može nastati lijeva strana neke od produkcija.

**Kleene L+:** KOJ su zatvoreni obzirom na Kleene L+. Neka KOG G1 = (V1, T1, P1, S1) generira KOJ L(G1). Gradimo G5 = (V5, T5, P5, S5) koja generira L(G5) = L(G1)+. Konstruiramo pomoćnu gramatiku G' V1V’ = zamjenom nezavršnih znakova gramatike G1. Slijedi:

* + V5 = V1V’ {S5, S’5}; S5, S’5V1; S5 , S’5V’
  + T5 = T1
  + P5 = P1 P’ {S5 S1S’5| S, S’5 S’S5|S’}

**27.2. Presjek i komplement**

* Zatvorenost KOJ po presjeku
* Zatvorenost KOJ po komplementu

**Presjek** KOJ jest KOJ. Neka LOA M1 prihvaća jezik L(M1), a LOA M2 prihvaća jezik L(M2). LOA M3 prihvaća jezik L(M3) = L(M1) ∩ L(M2) i ima dva traga ulazne trake.

U oba traga zapisuje se isti niz završnih znakova w. LOA M3 simulira rad LOA M1 na gornjem tragu ulazne trake, dok se rad LOA M2 simulira na donjem tragu ulazne trake. Niz se prihvaća ako i samo ako tijekom simulacije LOA M1 i LOA M2 prihvate niz.

**Komplement** determinističkog KOJ jest DKOJ. Neka DLOA M prihvaća DKOJ L(M). Za bilo koji DLOA moguće je izgraditi istovjetni DLOA M' koji uvijek stane za bilo koji ulazni niz.

Budući da DLOA koristi isključivo ćelije trake na kojima je zapisan ulazni niz w, moguće je izračunati maksimalan broj različitih konfiguracija DLOA na sljedeći način:

s(n+2)tn. s = |Q|, n = |w|; t = |Γ|.

**27.3. Odlučivost kontekstno ovisnih jezika**

* Svojstvo odlučivosti KOJ

Bilo koji KOJ jest RekJ. Algoritam prihvaćanje jezika zasniva se na gradnji usmjerenog grafa:

* Čvorovi grafa su nizovi završnih i nezavršnih znakova gramatike α koji su jednako dugački ili kraći od ulaznog niza *w*, tj. |α|≤*|w*|.
* Vrijedi li relacija α → β, čvorovi grafa α i β povezuju se granom.
* Put usmjerenog grafa predstavlja postupak generiranja niza primjenom produkcija gramatike G.
* Put od čvora S do čvora w postoji ako i samo ako gramatika G generira niz w.

Budući da je konačni broj nizova α koji su jednako dugački ili kraći od ulaznog niza *w*:

* TS u konačnom broju koraka izgradi usmjereni graf i pretraži sve putove usmjerenog grafa
* TS se uvijek zaustavi za bilo koji ulazni niz i odluči o prihvaćanju niza, što dokazuje da su kontekstno ovisni jezici rekurzivni, odnosno **odlučivi**.

**28. STRUKTURNA SLOŽENOST JEZIKA**

**28.1. Klase i hijerarhija jezika**

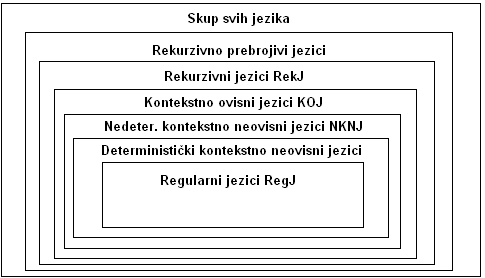
* Klase jezika
* Hijerarhija Chomskog
* Definicije istovjetnosti

Ako je klasa A pravi podskup klase B:

* Automat koji prihvaća jezike klase A jednostavnije je strukture od automata koji prihvaća jezike klase B.
* Produkcije gramatike koja generira jezike klase A jednostavnije su od produkcija gramatike koja generira jezike klase B.

Jezici klase A su jednostavnije strukturne složenosti (npr. RegJ su jednostavniji od KNJ).

Koristi se hijerarhija Chomskog:



Definicije:

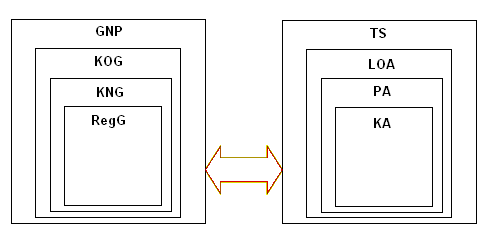
* Klasa RegJ je pravi podskup klase DKNJ.
* Klasa DKNJ je pravi podskup klase NKNJ. (jezik wcwR)
* Klasa NKNJ je pravi podskup klase KOJ. (jezik wwR)
* Klasa KOJ je pravi podskup klase RekJ. (jezik ww)
* Klasa RekJ (vrlo rijetki) je pravi podskup klase RPJ.
* Klasa RPJ (Lu) je pravi podskup klase svih jezika (Ld).

**28.2. Hijerarhija gramatika i automata**

* Jezici i istovjetnosti
* Hijerarhija gramatika i automata
* Oblici produkcija i definicije automata

Hijerarhija se temelji na hijerarhiji jezika. Temelji se na istovjetnostima:

* RegJ, konačni automati i regularne gramatike
* KNJ, kontekstno neovisne gramatike i potisni automati
* KOJ, kontekstno ovisne gramatike i linearno ograničeni automati
* RPJ, gramatike neograničenih produkcija i Turingovog stroja



Definicije:

* GNP G0 , odgovara TS M0 = (Q, , , , q0, B, F)
* KOG G1 , , || || odgovara LOA M1 = (Q, , , , q0, €, $, F)
* KNG G2 Aodgovara PA M2 = (Q, , , , q0, Z0, F)
* RegG G3 (AwB, Aw) ili (ABw, Aw) odgovara KA M3 = (Q, , , q0, F)

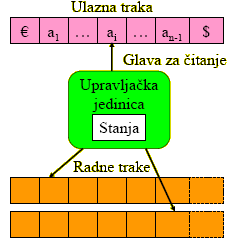
**29. SLOŽENOST PRIHVAĆANJA JEZIKA**

**29.1. Prostorna složenost**

* Korištenje modela TS
* Model za prostornu složenost
* Definicija prostorne složenosti

Koristi se neizravni deterministički TS s *k* polubeskonačnih traka.

Ulazna traka sadrži niz *w* koji ispituje duljine *n*. Ulazna traka se samo čita. TS ima *k* radnih traka beskonačnih s desna na koje čita i piše.



Definicija prostorne složenosti određuje se na osnovu samo jedne trake i to one na kojoj se koristi najviše ćelija. Kada TS koristi S(n) ćelija na jednoj od radnih traka, TS je prostorne složenosti S(n) za niz dužine n. Jezik L = L(M) je prostorne složenosti S(n).

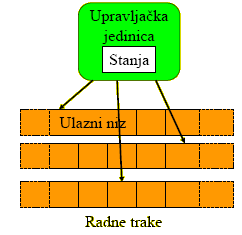
Osnova: za bilo koji TS s *k* traka moguće je izgraditi istovjetni TS s jednom trakom i iste je prostorne složenosti.

Uzimaju se u obzir samo radne trake, prostorna složenost može biti manja od n. Uzima se da je najmanja složenost 1. Dozvolimo li pisanje po ulaznoj traci, u obzir se uzima i ulazna traka. Tada je složenost najmanje n.

**29.2. Vremenska složenost**

* Model za vremensku složenost
* Definicija vremenske složenosti

Koristi se deterministički TS s *k* beskonačnih traka. Jedna traka je ulazna i sadrži niz koji se ispituje. Sve trake su za čitanje i pisanje. Ulazni niz *w* je duljine *n*.



Definicija vremenske složenosti računa se na osnovu broja pomaka glave. Izvede li TS M najviše T(n) pomaka glave, TS M je vremenske složenosti T(n). Jezik L = L(M) je vremenske složenosti T(n). TS treba pročitati svih *n* znakova + znak iza niza. Najmanja vremenska složenost je *n+1*. Pri tome se ignorira situacija kada se TS zaustavi prije čitanja cijelog niza.

**29.3. Broj traka i složenost**

* Svojstva složenosti
* Broj traka i prostorna složenost
* Broj traka i vremenska složenost

Broj traka nema utjecaja na prostornu složenost.

Broj traka ima utjecaja na vremensku složenost: vremenska složenost povećava se sa smanjenjem broja traka.

Konstantne faktore kod složenosti moguće je zanemariti. U razmatranje uzimamo linearne funkcije, polinomne funkcije, eksponencijalne funkcije, itd.

Ako TS M1 s *k* radnih traka **prostorne** **složenosti** S(n) prihvaća jezik L(M1), tada postoji TS M2 s jednom radnom trakom jednake prostorne složenosti S(n) i koji prihvaća jezik

L(M2) = L(M1).

Neka TS M1 na jednoj od traka koristi S(n) ćelija. Gradi se istovjetni TS M2 samo s jednom trakom i 2k tragova. Prvih *k* tragova koristi se za spremanje sadržaja *k* traka TS M1. Drugih *k* tragova koristi se za bilježenje položaja glava TS M1. Prostorna složenost i dalje je jednaka S(n).

Izgradimo istovjetni TS M2 s jednom radnom trakom koji prihvaća jezik L(M2) = L(M1). Povećanje broja pomaka glave opisuje se kvadratnom funkcijom:

Ako je TS M1 s *k* traka **vremenske** **složenosti** T(n), tada je TS M2 s jednom trakom vremenske složenosti T2(n). Pri tome su zanemareni konstantni faktori.

Smanji li se *k* na dvije trake (umjesto na jednu) izgradi se TS vremenske složenosti T(n)log2T(n).

**29.4. Sažimanje prostora i ubrzanje vremena**

* Sažimanje prostora
* Ubrzanje vremena

**Sažimamo prostor** za konstantni faktor.

Ako TS M1 sa *k* radnih traka ima prostornu složenost S(n) i prihvaća L(M1):

za bilo koju konstantu c>0 postoji TS M2 prostorne složenosti cS(n), L(M1) = L(M2).

Znakovi u *r* susjednih ćelija TS M1 kodiraju se jednim znakom TS M2. Taj znak od *r* komponenti predstavlja *r* ćelija TS M1.

Želi li se sažeti prostor za faktor *c*, odredi se *r* tako da je *rc*≥2, a M2 koristi najviše S(n)/r ćelija.

Ako je S(n)≥r i ako je rc≥2 onda broj ćelija nije veći od cS(n).

Ako je S(n)<r za kodiranje sadržaja radne trake koristi se samo jedna ćelija.

**Ubrzavamo vrijeme** za konstantni faktor.

Ako TS M1 sa *k* radnih traka ima vremensku složenost T(n) i prihvaća L(M1):

za bilo koju konstantu c>0 postoji TS M2 s *k* traka vremenske složenosti cT(n), tako da je L(M1) = L(M2). Vrijedi: k>1 i infn∞T(n)/n = ∞ (najveća donja granica kada *n* teži ∞).

TS M2 ima najmanje dvije trake (k>1):

TS čita ulazni niz i *m* znakova pretvara u jedan na radnu traku. Prepisivanje i sažimanje se nastavlja za cijeli ulazni niz. Sada je sažeti zapis ulazni niz.

TS M2 simulira rad TS M1 tako da po grupama od po 3 komprimirana znaka obavi obradu u najviše 8 koraka, 4 za čitanje i 4 za pisanje. Dakle ukupno treba 8T(n)/m koraka.

Dodamo još prevođenje trake: n koraka za čitanje ulaznog niza i n/m koraka za povratak na početak komprimiranog niza.

TS M2 ukupno treba: n + n/m + 8T(n)/m koraka. Biramo cm≥16.

Postigne se n + n/m + 8T(n)/m koraka < cT(n).

**30. KLASE JEZIKA PO SLOŽENOSTI**

**30.1. Model složenosti i odnosi među klasama**

* Model složenosti i TS
* Osnovne klase složenosti
* Odnosi među klasama složenosti

Prethodne definicije bazirane su na determinističkom TS. Jezik je nedeteminističke složenosti ako ga prihvaća nedeterministički TS.

NTS je prostorne složenosti S(n) ako za niti jedan niz duljine *n*, niti jedan slijed prijelaza i na niti jednoj radnoj traci ne koristi više od S(n) ćelija.

Jezik L je nedeterminističke prostorne složenosti S(n) ako i samo ako postoji NTS M:

L(M) = L, prostorne složenosti S(n).

NTS je vremenske složenosti T(n) ako za niti jedan niz duljine *n* i niti jedan slijed prijelaza ne pomakne glavu više od T(n) puta.

Jezik L je nedeterminističke vremenske složenosti T(n) ako i samo ako postoji NTS M:

L(M) = L, vremenske složenosti T(n).

Jezici su podijeljeni u četiri osnovne klase složenosti:

* DSPACE(S(n)) čine jezici determinističke prostorne složenosti S(n)
* NSPACE(S(n)) …
* DTIME(T(n))
* NTIME(T(n))

Ako je jezik L u klasi DTIME(f(n)), onda je u klasi DSPACE(f(n)).

Ako je jezik L u klasi DSPACE(f(n)), f(n)≥log2n, onda je u klasi DTIME(cf(n)), c ovisi o L.

Ako je jezik L u klasi NTIME(f(n)), f(n)≥log2n, onda je u klasi DTIME (cf(n)), c ovisi o L.

Ako je jezik L u klasi NSPACE(f(n)), f(n)≥log2n, onda je u klasi DSPACE(f2(n)).

**30.2. Vremenska složenost DTS i NTS i prostorna izgradivost**

* Opis NTS stablom
* Vremenska složenost i stablo
* Prostorna izgradivost S(n)
* Svojstva hijerarhije

Vremensku složenost DTS i NTS opisujemo stablom NTS. Čvorovi su konfiguracije NTS, korijen je početna konfiguracija. Grane definiraju prijelaze u skup konfiguracija nedeterminističke funkcije prijelaza.

Vremenska složenost NTS proporcionalna je dubini stabla. Vremenska složenost DTS proporcionalna je vremenu slijednog pretraživanja stabla. U općem slučaju DTS treba eksponencijalno vrijeme u odnosu na NTS.

Neka je TS M prostorne složenosti S(n). S(n) je **prostorno izgradiva** ako za bilo koji *n* postoji barem jedan niz *w* za koji TS koristi svih S(n) ćelija.

Prostorno su izgradive funkcije log(n), nk, 2n, n!

Ako su prostorno izgradive S1(n) i S2(n), tada je prostorno izgradiva S1(n)S2(n), 2S1(n) i S1(n)S2(n).

S(n) je **potpuno prostorno izgradiva** ako za bilo koji niz duljine *n* TS koristi svih S(n) ćelija. Slično definiramo **vremenski izgradive** i **potpuno vremenski izgradive** funkcije.

Svojstva hijerarhije:

* Hijerarhija je beskonačna
* Može se pokazati prostorna hijerarhija
* Može se pokazati vremenska hijerarhija
* U vremenskoj i prostornoj hijerarhiji postoje praznine
* Za neke jezike ne postoji optimalni TS
* Moguće je graditi složene klase jezika

**30.3. Klasa jezika polinomne složenosti**

* Odlučivost i jezici polinomne složenosti
* Označivanje
* Odnosi determinističke i nedeterminističke polinomne složenosti

Ne garantira se efikasnost rada TS. Može se koristiti isključivo DTS eksponencijalne složenosti. Ako se nađe posebno rješenje koje je jednostavnije, to bi bilo jako korisno.

Značajna je klasa determinističke polinomne složenosti. Za većinu funkcija polinomi su niskog stupnja.

Koristimo oznake P i NP:

* P:
  + P = , i ≥ 1
* NP:
  + NP = 

PSPACE = 

NPSPACE = 

Na temelju svojstva unije klasa jezika, u klasi jezika P su svi oni jezici za koje postoji deterministički TS polinomne vremenske složenosti n, n2, n3, n4, ...

Mnogi značajni jezici nisu u klasi P, ali za njih postoji nedeterministički TS polinomne vremenske složenosti. Klasa jezika nedeterminističke polinomne složenosti označava se oznakom NP.