



Milan Milosavljević

# VEŠTAČKA INTELIGENCIJA

# VEŠTAČKA INTELIGENCIJA

3.  
izdanje

**Milan Milosavljević**

UNIVERZITET SINGIDUNUM

Beograd, 2022.

# VEŠTAČKA INTELIGENCIJA

*Autor:* dr Milan Milosavljević

*Recenzenti:* dr Branko Kovačević  
dr Mladen Veinović

*Izdavač:* UNIVERZITET SINGIDUNUM  
Beograd, Danijelova 32  
[www.singidunum.ac.rs](http://www.singidunum.ac.rs)

*Za izdavača:* dr Milovan Stanišić

*Priprema za štampu:* Novak Njeguš

*Dizajn korica:* Aleksandar Mihajlović

*Godina izdanja:* 2022.

*Tiraž:* 450 primeraka

*Štampa:* Birograf, Beograd

*ISBN:* 978-86-7912-590-3

Copyright:

© 2022. Univerzitet Singidunum

Izdavač zadržava sva prava.

Reprodukcija pojedinih delova ili celine ove publikacije nije dozvoljena.

# Sadržaj

|                                                                           |    |
|---------------------------------------------------------------------------|----|
| Uvod                                                                      | 1  |
| <b>1. ŠTA JE VEŠTAČKA INTELIGENCIJA?</b>                                  | 3  |
| 1.1 Pristup zasnovan na Tjuringovom testu                                 | 4  |
| 1.2 Pristup zasnovan na kognitivnom modelovanju                           | 8  |
| 1.3 Pristup zasnovan na racionalnom mišljenju<br>uz pomoć formalne logike | 10 |
| 1.4 Pristup zasnovan na racionalnim agentima                              | 11 |
| 1.5 O nužnosti veštačke inteligencije                                     | 12 |
| <i>Rezime 1. poglavlja</i>                                                | 14 |
| <i>Pitanja i zadaci</i>                                                   | 14 |
| <b>2. OSNOVE VEŠTAČKE INTELIGENCIJE</b>                                   | 15 |
| 2.1 Filozofija                                                            | 15 |
| 2.2 Matematika                                                            | 17 |
| 2.3 Ekonomija                                                             | 20 |
| 2.4 Neuronauka                                                            | 22 |
| 2.5 Psihologija                                                           | 24 |
| 2.6 Računarstvo                                                           | 26 |
| 2.7 Teorija upravljanja i kibernetika                                     | 27 |
| 2.8 Lingvistika                                                           | 27 |
| <i>Rezime 2. poglavlja</i>                                                | 28 |
| <i>Pitanja i zadaci</i>                                                   | 29 |
| <b>3. KRATKA ISTORIJA VEŠTAČKE INTELIGENCIJE</b>                          | 31 |
| 3.1 Početak oblasti veštačke inteligencije i rani entuzijazam             | 31 |
| 3.2 Shvatanje principijelnih nedostataka i praktičnih dometa VI           | 34 |
| 3.3 Doba ekspertskih sistema                                              | 35 |
| 3.4 Povratak tehnologije neuronskih mreža                                 | 38 |
| 3.5 VI u doba Interneta i web-a                                           | 39 |
| 3.6 Neki najuspešniji VI sistemi                                          | 42 |
| <i>Rezime 3. poglavlja</i>                                                | 46 |
| <i>Pitanja i zadaci</i>                                                   | 46 |



|                                                                    |     |
|--------------------------------------------------------------------|-----|
| <b>4. ARHITEKTURE SISTEMA VEŠTAČKE INTELIGENCIJE</b>               | 47  |
| 4.1 Pojam produkcionih sistema                                     | 48  |
| 4.2 Odnos između vrste produkcionih sistema i tipova problema      | 53  |
| 4.3 Vrste produkcionih sistema u odnosu na smer pretrage           | 53  |
| 4.4 Arhitekture inteligentnih agenata                              | 55  |
| 4.4.1 Jedostavni refleksni agenti                                  | 55  |
| 4.4.2 Refleksni agenti zasnovani na modelu                         | 56  |
| 4.4.3 Agenti zasnovani na cilju                                    | 58  |
| 4.4.4 Agenti zasnovani na korisnosti                               | 59  |
| 4.4.5 Obučavajući agenti                                           | 60  |
| <i>Rezime 4. poglavlja</i>                                         | 62  |
| <i>Pitanja i zadaci</i>                                            | 52  |
| <b>5. STRATEGIJE PRETRAGE U SISTEMIMA VEŠTAČKE INTELIGENCIJE</b>   | 65  |
| 5.1 Osnovni pojmovi o gafovima                                     | 67  |
| 5.2 Opšti algoritam pretrage na grafu - Algoritam graphsearch      | 69  |
| 5.2.1 Razjašnjenje koraka 6                                        | 70  |
| 5.3 Merenje performanse strategija pretrage                        | 72  |
| 5.4 Neinformativne procedure pretrage na grafu                     | 73  |
| 5.4.1 Pretraga u širinu                                            | 73  |
| 5.4.2 Pretraga sa uniformnim težinama                              | 74  |
| 5.4.3 Pretraga po dubini                                           | 75  |
| 5.4.4 Dvosmerne pretrage                                           | 78  |
| 5.5 Poređenje performansi neinformativnih strategija               | 78  |
| <i>Rezime 5. poglavlja</i>                                         | 79  |
| <i>Pitanja i zadaci</i>                                            | 79  |
| <b>6. INFORMATIVNE PRETRAGE U SISTEMIMA VEŠTAČKE INTELIGENCIJE</b> | 81  |
| 6.1 Algoritam A i A*                                               | 84  |
| 6.2 Svojstva A* algoritma                                          | 87  |
| 6.3 Heuristička snaga evaluacionih funkcija                        | 88  |
| 6.4 Generisanje dopustivih heurističkih funkcija                   | 91  |
| 6.5 Mere efikasnosti pretraga                                      | 93  |
| <i>Rezime 6. poglavlja</i>                                         | 94  |
| <i>Pitanja i zadaci</i>                                            | 95  |
| <b>7. ALTERNATIVNI SISTEMI PRETRAGE</b>                            | 97  |
| 7.1 Pretraživanje usponom                                          | 98  |
| 7.2 Simulirano kaljenje                                            | 101 |
| 7.3 Lokalno pretraživanje po snopu                                 | 104 |
| 7.4 Genetski algoritmi                                             | 104 |
| 7.5 Lokalne pretrage u kontinualnim prostorima                     | 109 |
| 7.6 Hibridne strategije                                            | 110 |
| <i>Rezime 7. poglavlja</i>                                         | 113 |
| <i>Pitanja i zadaci</i>                                            | 114 |

|                                                                           |     |
|---------------------------------------------------------------------------|-----|
| <b>8. OSNOVE SIMBOLIČKIH SISTEMA VEŠTAČKE INTELIGENCIJE</b>               | 115 |
| 8.1 Deklarativna znanja                                                   | 117 |
| 8.2 Račun predikata                                                       | 118 |
| 8.2.1 Sintaksa računa predikata                                           | 119 |
| 8.2.2 Semantika računa predikata                                          | 121 |
| 8.3 Zaključivanje                                                         | 122 |
| 8.4 Semantička posledica                                                  | 125 |
| 8.5 Dokazivost                                                            | 125 |
| <i>Rezime 8. poglavlja</i>                                                | 126 |
| <i>Pitanja i zadaci</i>                                                   | 127 |
| <b>9. AUTOMATSKI REZONOVANJE</b>                                          | 129 |
| 9.1 Transformacija u klauzalnu formu                                      | 130 |
| 9.2 Unifikacija                                                           | 133 |
| 9.3 Princip rezolucije                                                    | 135 |
| 9.4 Primena rezolucije                                                    | 137 |
| 9.5 Sistemi za dobijanje konstruktivnih odgovora zasnovanih na rezoluciji | 140 |
| 9.6 Hornove rečenice                                                      | 141 |
| 9.7 Ekspertski sistemi                                                    | 143 |
| 9.7.1 Arhitektura ekspertnih sistema                                      | 143 |
| 9.7.2 Upravljanje ekspertnim sistemom (inerpreter)                        | 145 |
| <i>Rezime 9. poglavlja</i>                                                | 147 |
| <i>Pitanja i zadaci</i>                                                   | 148 |
| <b>10. UVOD U MAŠINSKO UČENJE</b>                                         | 149 |
| 10.1 Induktivno empirijsko učenje funkcionalnih preslikavanja             | 151 |
| 10.2 Induktivno učenje stabala odlučivanja                                | 155 |
| 10.3 Osnovni algoritmi učenja stabla odluke                               | 158 |
| 10.4 Traženje najboljeg atributa za klasifikaciju                         | 158 |
| 10.5 Učenje stabala odluke u prostoru hipoteza (version space)            | 163 |
| 10.6 Induktivni pomerač (bias)                                            | 165 |
| 10.7 Praktični problemi učenja stabala odluke                             | 167 |
| 10.7.1 Izbegavanje preprilagodjenosti (overfitting)                       | 167 |
| 10.7.2 Kresanje (pruning) u cilju smanjivanje greške klasifikacije        | 170 |
| 10.7.3 Uvođenje atributa sa kontinualnim vrednostima                      | 171 |
| 10.7.4 Alternativne mere za selektovanje atributa                         | 172 |
| 10.7.5 Obučavajući primeri kod kojih nedostaju vrednosti atributa         | 173 |
| 10.7.6 Atributi sa promenljivom cenom                                     | 173 |
| <i>Rezime 10. poglavlja</i>                                               | 174 |
| <i>Pitanja i zadaci</i>                                                   | 175 |
| <b>11. NEURONSKE MREŽE</b>                                                | 177 |
| 11.1 Definicija neuronske mreže                                           | 179 |
| 11.2 Svijsstva neuronskih mreža                                           | 179 |
| 11.3 Modeli neurona                                                       | 180 |

|                                                                                     |     |
|-------------------------------------------------------------------------------------|-----|
| 11.4 Tipovi aktivacionih funkcija                                                   | 182 |
| 11.5 Arhitekture neuronskih mreža                                                   | 183 |
| 11.6 Prezentacija znanja u neuronskim mrežama                                       | 186 |
| 11.7 Obučavanje neuronskih mreža                                                    | 186 |
| 11.7.1 Opšta forma pravila obučavanja                                               | 188 |
| 11.7.2 Hebovo učenje                                                                | 188 |
| 11.8 ADALINA (Adaptive Linear Element)                                              | 189 |
| 11.9 Jednoslojni perceptor                                                          | 190 |
| 11.10 Višeslojni perceptor                                                          | 192 |
| 11.10.1 Cibenkova teorema (1989)                                                    | 192 |
| 11.10.2 Algoritam propagacije greške unazad                                         | 193 |
| 11.10.3 Problem konvergencije                                                       | 197 |
| 11.10.4 Faktori koji utiču na obučavanje algoritma propagacije unazad               | 197 |
| 11.10.4.1 Inicijalizacija težina                                                    | 197 |
| 11.10.4.2 Koeficijent obučavanja (learning constant)                                | 197 |
| 11.10.4.3 Funkcija cilja                                                            | 198 |
| 11.10.4.4 Momentum                                                                  | 198 |
| 11.10.4.5 Pravila korekcije                                                         | 199 |
| 11.10.4.6 Obučavajući skup i generalizacija                                         | 199 |
| 11.10.4.7 Broj skrivenih neurona                                                    | 205 |
| 11.11 Upravljanje neuroračunarskim projektima                                       | 205 |
| 11.11.1 Osnovna svojstva neuroračunarskih projekata                                 | 206 |
| 11.11.2 Osnovne faze projekta                                                       | 206 |
| 11.11.3 Planiranje projekta                                                         | 208 |
| 11.11.3.1 Planiranje razvoja prototipa                                              | 208 |
| 11.11.3.2 Planiranje prikupljanja podataka                                          | 208 |
| 11.11.4 Analiza-revizija                                                            | 209 |
| 11.11.5 Upravljanje konfiguracijom                                                  | 209 |
| 11.11.6 Dokumentacija                                                               | 210 |
| 11.11.7 Implementacija konačnog sistema                                             | 210 |
| 11.11.8 Preuzimanje i održavanje sistema                                            | 210 |
| <i>Rezime 11. poglavlja</i>                                                         | 211 |
| <i>Pitanja i zadaci</i>                                                             | 211 |
| <b>12. DUBOKO OBUČAVANJE NEURONSKIM MREŽAMA</b>                                     | 213 |
| 12.1 Dubina arhitekture sistema za obučavanje                                       | 213 |
| 12.1.1 Motivacija za duboke arhitekture                                             | 214 |
| 12.1.2 Spektakularni proboj u obučavanju dubokih arhitektura                        | 216 |
| 12.2 Duboke neuronske mreže                                                         | 218 |
| 12.3 Filozofski pogled na učenje internih reprezentacija dubokim neuronskim mrežama | 225 |
| <i>Rezime 12. poglavlja</i>                                                         | 229 |
| <i>Pitanja i zadaci</i>                                                             | 229 |
| Literatura                                                                          | 231 |

# Uvod

Veštačka inteligencija (VI) je konstituisana kao naučna oblast u neposrednoj vezi sa uspostavljanjem računarstva kao naučno – tehničke discipline. Ozbiljno je počela da se razvija ubrzo posle Drugog svetskog rata, a samo ime je dobila tek 1956. godine. VI trenutno obuhvata mnoštvo podoblasti, od vrlo opštih, kao što su učenje i percepcija, do uskih zadataka kao što je igranje šaha, dokazivanje matematičkih teorema, pisanja poezije, medicinska dijagnostika, automatsko prevodjenje, prepoznavanje govora, robotika i sl. VI sistematizuje i automatizuje intelektualne zadatke i stoga je potencijalno značajna za bilo koju sferu ljudske intelektualne aktivnosti. U tom kontekstu VI je univerzalna oblast, koja sasvim opravdano predstavlja deo kurikuluma gotovo svih obrazovnih profila računarstva i informatike danas.

Postavlja se pitanje kada ćemo posegnuti za znanjima iz ove oblasti kada se susretnemo sa nekim konkretnim praktičnim problemom u praksi. U tom pogledu je uputno razlikovati dobro formalizovane, struktuirane probleme, od tzv. nestruktuiranih problema. Ovi prvi imaju preciznu formulu i rešavaju se u okviru formalnih teorija unutar kojih je i data njihova formula. Npr. ako je potrebno rešiti problem nalaženja korena kvadratne jednačine, poslužićemo se formulom koja je razvijena u okviru teorije rešavanja algebarskih jednačina. Takođe, svi praktični problemi, čija se reformulacija svodi na rešavanje kvadratne jednačine, rešavaće se na ovaj način. Za takav tip problema nam nisu potrebne metode veštačke inteligencije.

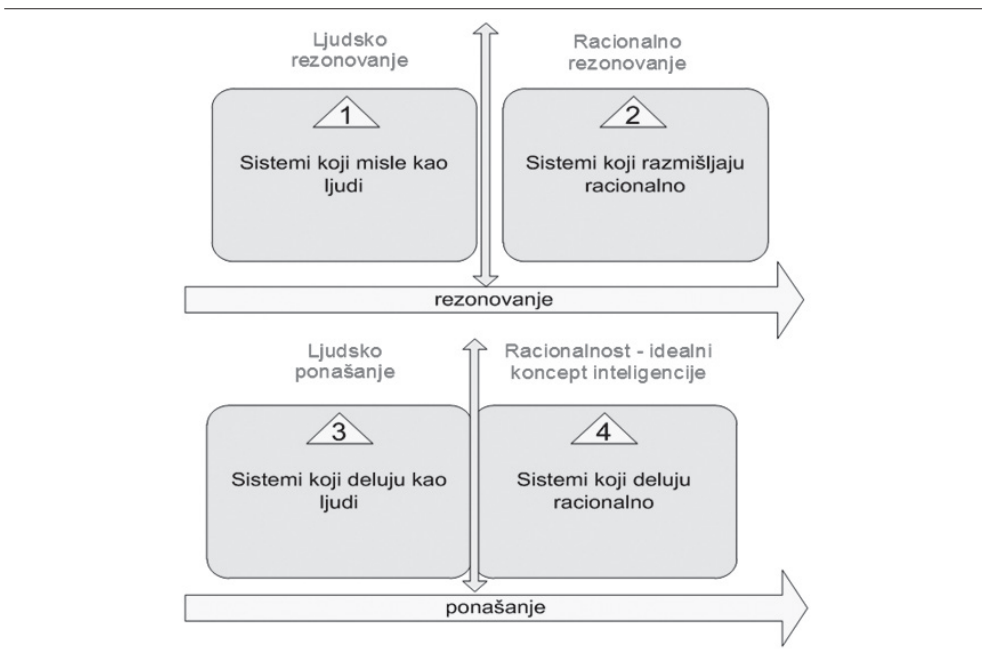
Razmotrimo sada problem transfera ekspertskih znanja jednog kardiologa dijagnostičara u automatizovani dijagnostički sistem. Ako bismo pitali eksperta kako on donosi dijagnostičke odluke, dobili bismo u najboljem slučaju jezičke iskaze tipa ako - onda, a u dosta slučajeva odgovore da se, pod određenim uslovima ove odluke donose intuitivno, naročito u hitnim slučajevima, ili u nedostatku dovoljne količine kliničkih testova i pokazatelja. Ne postoji, barem za sada, formalna teorija u okviru koje bi se ovaj problem rešavao na sličan način kao rešavanje kvadratne jednačine. Ovaj i slični problemi pripadaju klasi nestruktuiranih problema, koje dobar kardiolog dijagnostičar ipak rešava. Kako? Na osnovu nakupljenog ekspertskog znanja. Tehnologija ekspertskih sistema razvijena još osamdesetih godina prošlog veka u okviru VI prava je metoda za rešavanje ovog tipa nestruktuiranih zadataka. Da li se ekspertu moraju postavljati pitanja u cilju transfera znanja, ili je dovoljno pokupiti dobro vodjene elektronske kartone pacijenata u poslednjih 10 godina i na osnovu njih doći do sličnih rezultata. Ovaj pristup nas dovodi pred vrata oblasti mašinskog učenja, koja predstavlja front današnjeg razvoja veštačke inteligencije. Ovome je svakako doprinela poplava dostupnih podataka na internetu, čijom inteligentnom obradom možemo doći do dragocenih znanja, koja često odlučujuće utiče na uspešno poslovanje u novoj digitalnoj ekonomiji.

Dublja analiza mogućih problema sa kojima se susrećemo u sve bogatijoj praksi današnjeg složenog globalnog društveno ekonomskog sistema, oslanjenog na računarsko komunikacionu tehniku, pokazuje oštru podelu na klase problema, koji se lako rešavaju primenom računara i klase problema koje čovek lakše rešava od mašina. U te probleme spadaju različite igre, vizija, prepoznavanje oblika, razumevanje i klasifikacija dokumenata, semantička analiza tekstova, njihova sumarizacija, motorne sposobnosti, upravljanje vozilima, intuitivno i heurističko rezonovanje i donošenje odluka u uslovima neodređenosti, itd. Ova druga klasa problema je u vidokrugu oblasti veštačke inteligencije, upravo zato što je čovekov način rešavanja tih problema još uvek superioran u odnosu na automatizovane računarske sisteme. Otuda je oblast veštačke inteligencije dinamična i otvorena. Neki problemi iz nje se isključuju (npr. igra šaha u kojoj su dostignuti i prestignuti standardi ljudskih mogućnosti), a neki novi se uključuju, kao npr. data mining ili medicinska dijagnostika u hiperdimenzionim prostorima genskih ekspresija.

Sve ovo čini oblast veštačke inteligencije izazovnom i privlačnom, pa stoga ne čudi nesmanjeno interesovanje istraživača, uprkos brojnim problemima i sporadičnim uspesima. Nadamo se da će ovaj udžbenik dati povoda da nove generacije studenta računarstva i informatike vide sopstveni interes u poznavanju metoda veštačke inteligencije. Udžbenik bi postigao još veći cilj ako bi pomogao studentima u potsticanju novih ideja i razvoju sopstvenog pogleda na mesto i ulogu računarsko komunikacionih sistema u savremenom društvu.

# 1. Šta je veštačka inteligencija?

I pored poluvekovne istorije VI je i dalje oblast koju je teško precizno definisati. Primeri nekih definicija dobijenih dihotomijom u odnosu na dimenzije rezonovanja i ponašanja dati su na sl.1.1.1.



Sl.1.1.1 Kategorije o mogućim pogledima na sisteme veštačke inteligencije.

Gornje dve kategorije se bave procesima razmišljanja i rasudjivanja (rezonovanja), dok se donje dve bave ponašanjem. Kategorije sa leve strane mere uspešnost u kontekstu poklapanja sa ljudskim performansama, dok one sa desne strane mere uspešnost u poređenju sa idealnim konceptima inteligencije koje nazivamo racionalnost. Sistem je racionalan ako radi „pravu stvar“ s obzirom na ono što zna. Iza ove kolokvijalne tvrdnje stoji zahtev da sistem poseduje punu svest o cilju, odnosno formalno govoreći da je implicitno ili eksplicitno definisana kriterijumska funkcija, koja u nekom metričkom prostoru meri uspešnost delovanja inteligentnog sistema u svom radnom okruženju.

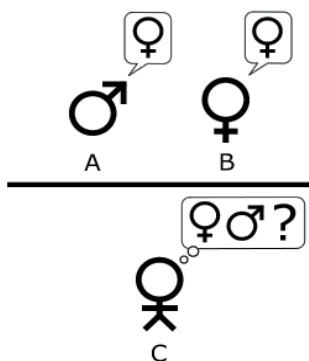
| <b>Sistemi koji razmišljaju kao ljudi</b>                                                                                                                                                                                                                                                                                | <b>Sistemi koji razmišljaju racionalno</b>                                                                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>„Uzbudljiv novi pokušaj da se kompjuteri nateraju da razmišljaju... <i>mašine sa umom</i>, u punom i bukvalnom smislu.“<br/>(Haugeland, 1985)</p> <p>[Automatizacija] aktivnosti koje asociramo sa ljudskim razmišljanjem, aktivnosti kao što su donošenje odluka, rešavanje problema, učenje...” (Bellman, 1987)</p> | <p>„Izučavanje mentalnih sposobnosti kroz korišćenje kompjuterskih modela“<br/>(Chamiak i McDermott, 1985)</p> <p>„Izučavanje računanja koje omogućava opažanje, rasudjivanje i delovanje.“<br/>(Winston, 1992)</p> |
| <b>Sistemi koji se ponašaju kao ljudi</b>                                                                                                                                                                                                                                                                                | <b>Sistemi koji se ponašaju racionalno</b>                                                                                                                                                                          |
| <p>„Veština pravljenja mašina koje izvršavaju funkcije koje zahtevaju inteligenciju kada se izvršavaju od strane ljudi“<br/>(Kurzweil, 1990)</p> <p>„Izučavanje kako naterati računare da rade stvari u kojima su, trenutno, ljudi bolji.“ (Rich i Knight, 1991)</p>                                                     | <p>„Računarska Inteligencija je nauka o dizajniranju inteligentnih agenata“<br/>(Poole, 1998)</p> <p>„VI ...se bavi inteligentnim ponašanjem veštačkih naprava“ (Nilsson, 1998)</p>                                 |

Tabela 1.1 – Neke definicije veštačke inteligencije organizovane u 4 kategorije sa Sl.1.1

Sva četiri tumačenja VI su bila aktuelna u pojedinim istorijskim fazama razvoja oblasti. Kao što se može očekivati, tenzija postoji između pristupa centriranih oko ljudi i pristupa centriranih oko racionalnosti. Ljudski-centrirani pristupi dominantno moraju biti empirijska nauka, koja uključuje hipotezu i eksperimentalnu potvrdu o ljudskom ponašanju. Racionalistički pristup koristi kombinaciju matematike i inženjerstva.

## 1.1. Pristup zasnovan na Tjuringovom testu

Alan Tjuring, engleski matematičar, sl.1.3, je u svom utemeljujućem radu iz 1950. godine „Computing Machinery and Intelligence“ predložio da umesto pitanja da li mašine mogu da misle, treba da se zapitamo da li mašina pretendent na inteligenciju može da prodje test inteligentnog ponašanja, koji je zatim nazvan Tjuringov test. Originalno je Tjuring ovaj test formulisao kao igru pogađanja ko je muškarac, a ko žena, na osnovu pitanja nezavisnog ispitivača, videti sl.1.2. Mašina prolazi Tjuringov test ako osoba ispitivač, posle postavljanja nekoliko pitanja u pisanoj formi, ne može da odredi da li je pisani odgovor dao čovek ili mašina. Mašina koja bi eventualno prošla Tjuringov test, morala bi da ima sledeće sposobnosti:



Igra imitacije kako je formulisana u Tjuringovom radu "Computing Machinery and Intelligence." Igrač C, na osnovu serije pisanih pitanja pokušava da odredi koji od druga dva igrača je muško a ko žensko. Igrač A, muškarac, teži da obmane igrača C i navede ga na pogrešan zaključak, dok igrač B pokušava da pomogne igraču C.

Sl.1.2 Imitaciona igra, kako je originalno formulisao Alan Tjuring. Modifikacija ove igre je zatim preformulisana kao čuveni Tjuringov test u domenu veštačke inteligencije.

- ♦ **obradu prirodnih jezika** - da bi mogla uspešno da komunicira u prirodnom jeziku,
- ♦ **reprezentaciju znanja** - da bi memorisala ono što zna i prima na osnovu senzora,
- ♦ **automatsko rezonovanje** - da bi koristila memorisane informacije za odgovaranje na pitanja i za donošenje novih zaključaka,
- ♦ **mašinsko učenje** - da bi se adaptirala novim okolnostima.



Sl.1.3 Alan Turing (1912-1954) utemeljivač računarskih nauka i veštačke inteligencije

Tjuringov test je namerno izbegavao direktnu fizičku interakciju između ispitivača i računara, zato što je fizička simulacija čoveka nepotrebna za inteligenciju. Međutim, takozvani potpuni Tjuringov test uključuje video signal, tako da ispitivač može da testira percepcijske i motorne sposobnosti ispitanika. Da bi prošao potpuni Turingov test računar mora da ovlada

- ♦ **računarskom vizijom** - u cilju vizuelene percepcije okoline i objekata u njoj, i
- ♦ **robotikom** – radi kretanja u prostoru i manipulacijom objekata u njemu.

Ovih 6 disciplina čine dominantan deo VI, a Tjuringov test u neizmenjenoj formi i danas predstavlja test dostignuća u ovoj oblasti. Kontraverzni bogataš Hju Lobner je 1990 godine ustanovio Lobnerovu nagradu i medaljon u vrednosti od 100 000 \$ za program koji se u tekućoj godini najviše približio prolazenju Tjuringovog testa, Sl.1.4.





Sl.1.4 Medalja koja se dobija uz Lobnerov unagradu, koju je ustanovio Hju Lobner uz dogovora Centrom za Bihevijoralne studije u Kembridžu 1990. god.,  
<http://www.loebner.net/Prizef/loebner-prize.html>

Ovih 6 disciplina čine dominantan deo VI, i Tjuring je zaslužan za dizajniranje testa koji ostaje još uvek relevantan, iako je od njegove formulacije prošlo znatno više od pola veka. Britanija je 24. decembra 2013. godine posthumno pomilovala Alana Tjuringa, koji je 1952. osuđen kao homoseksualac, što je u to vreme bilo ozbiljno krivično delo. Alan Tjuring je dve godine po izricanju presude izvršio samoubistvo u 41. godini života tako što je popio cijanid. Danas ga mnogi smatraju ocem računarske tehnologije i centralnom figurom za njen razvoj. Međutim, istraživači VI su posvetili malo napora u pravcu sistematskog približavanja krajnjem cilju: polaganju Tjuringovog testa, verujući da je bitnije izučavati osnovne principe inteligencije. Dobar primer, na koji se uvek vraćaju istraživači ovog uverenja je potraga za veštačkim letom. Ona je uspeła kada su braća Wright i ostali prestali da imitiraju ptice i počeli da uče o aerodinamici. Aeronautički inženjerski tekstovi ne definišu cilj njihove oblasti kao pravljenje mašina koje mogu da lete toliko slično pticama da mogu da zavaraju čak i druge ptice.

Primer uspeha avijacije nas vraća na fundamentalno pitanje ljudske kretivosti. Kao da nam se nameće princip po kome istinske prodore u domenu saznanja ostvarujemo ne u kopiranju i imitiranju, već u kreativnoj ili čak tzv. konkreativnoj interakciji sa predmetnim problemom. Danas je ovo pitanje u planetarnoj dominaciji liberalnog kapitalizma posebno zaoštreno, budući da već površna analiza najuspešnijih kompanija pokazuje, da je njihova diferencijalna prednost bazirana dominantno na inovativnosti i kreativnosti. U svojoj prirodi, istinska inovativnost znači prodore koji se malo ili uopšte ne odnose na nešto prethodno. Ovaj fenomen postao je predmet i filozofskih istraživanja dajući sasvim novu filozofsku disciplinu **hermetiku**. Hermetiku u noviju filozofiju uvodi Hajnrih Rombah, austrijski filozof, sl.1.5. Odnosi se na stvaralaštvo koje nema oslonca u onome što je do sada poznato.

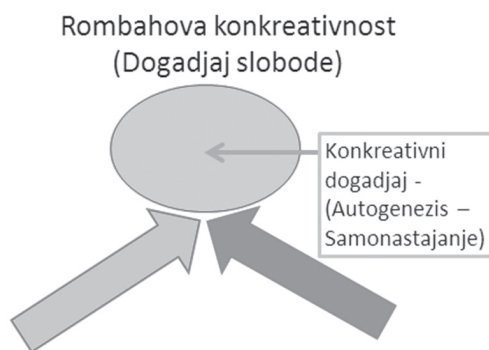
Ključni Rombahov pojam je **konkreativnost**, koja se odnosi na procese između čoveka i prirode, čoveka i situacije, čoveka i čoveka. Ovi procesi su hermetički, stoga što se ne izvode iz subjektivnosti niti jedne niti druge strane, a ne mogu se ni anticipirati niti uplanirati kao posledica nekog anonimnog mehanizma, ili nekog nadređenog sistema. Konkreativnost postaje ključni pojam mišljenja. Uključuje u sebe kako nešto nastaje iz zajedničkog događaja, na taj način da tek nakon toga postaje to što jeste, odnosno ono što jeste u potvrđenom obliku, polazeći od načina na koji ga shvatamo. Na primer u umetničkom delu, i umetničko



Sl.1.5 Heinrich Rombach (1923–2004), austrijski filozof, utemeljivač savremene hermetike

delo i umetnik postaju zajedno, prevazilazeći postojeće samokoncepte. Konkreativan pristup znači, npr. da se jedno pomera ka drugom na takav način da se u okviru tog susreta, odgovarajući novi svet (svet novih svojstava) aktivira u smeru pojavljivanja. Samo se u konkreativnom događaju dešava da se sami uslovi nastanka događaja samog, pojavljuju na prvom mestu, videti moguću interpretaciju na Sl.1.6.

Stoga je fokus da se razume drugo, ne samo u okviru sveta drugih, nego šta više i pre svega, pomažući drugom u ovom svetu i gledajući u njega. Tek tada nastaje novi svet, iz koga oboje izrastaju, umesto da samo započnu da participiraju u njemu.



Sl.1.6 Moguća strukturna interpretacija Rombahovog pojma konkreativnosti kao događaja slobode, kroz proces autogeneze, odnosno samonastajanja.

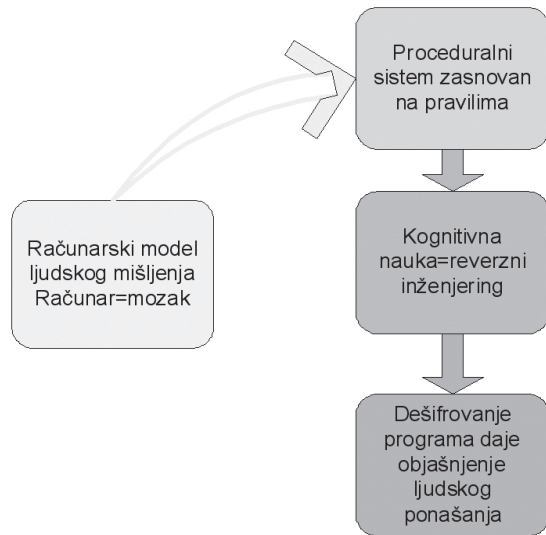
Čovek ne stoji nasuprot prirode, već razumeva sebe kao deo sveobuhvatnije prirode. Čovečanstvo i priroda uvek odgovaraju jedno drugom u konkreativnom događaju, istovremeno prevazilazeći trenutne mogućnosti. „Izuzetna pozicija čoveka“ se pokazuje kao metafizička konstrukcija kojoj nema mesta, budući da je priroda već nastala iz sebe kao proces samoradjanja i samokreacije.

Iako filozofski pristup ovom izuzetnom problemu predstavlja teškoću za stvaraoce u domenu prirodno tehničkih i ekonomskih nauka, već i sam filozofski diskurs daje dovoljnu impresiju o tome da se ovom problemu ne može pristupiti konvencionalnim mišljenjem zarobljenim u klišeima tehničko tehnološkog i matematizovanog mišljenja. Današnje vodeće kompanije u računarsko informatičkom sektoru, organizacijom rada, izborom i vođenjem kadrova, nesvakidašnjim radnim okruženjem, nekonvencionalnim pravilima ponašanja na radnom mestu, zapravo i nesvesno slede Rombahove duboke uvide zasnovane na konkreativnosti.

## 1.2 Pristup zasnovan na kognitivnom modelovanju

Ako pretendujemo da kažemo da dati računarski program razmišlja kao čovek, moramo prvo na neki način da odredimo kako ljudi razmišljaju. Postoje tri načina da se ovo uradi: kroz introspekciju – pokušavajući da uhvatimo sopstvene misli, pomoću psiholoških eksperimenata posmatrajući delovanje i reakcije čoveka u kontrolisanim uslovima, i konačno funkcionalnim mapiranjem mozga savremenim sistemima funkcionalne magnetne rezonance. Kada bi se izgradila dovoljno jaka teorija uma, naredni korak, implementacije ove teorije u vidu računarskog programa, predstavljala bi krupan praktičan korak u implementaciji inteligentnih veštačkih sistema. Ako se rezultati ovakvih računarskih programa poklapaju sa korespondirajućim ljudskim ponašanjima, to bi dokazivalo činjenicu da se neke ljudske aktivnosti odvijaju po programiranim šemama ponašanja i reagovanja, videti Sl.1.7. Na primer, Allen Newell i Herbert Simon, koji su razvili „Uopšteni Rešavač Problema“ (General Problem Solver, GPS), nisu imali za cilj samo to da njihovi programi rešavaju probleme tačno, već i da način rešavanja bude što sličniji ljudskom načinu dolaženja do rešenja. Interdisciplinarna oblast kognitivne nauke povezuje računarske modele iz VI i eksperimentalne tehnike iz psihologije, sa krajnjim ciljem pravljenja precizne teorije o načinu rada ljudskog mozga, koju je moguće praktično testirati. Kao rezultat uprošćenog shvatanja analogije između ljudskog mišljenja i računara, pojavila se interpretacija data na sl.1.8, po kojoj je naše mišljenje rezultat izvođenja ekvivalentnog programa na biološkom hardveru - mozgu, kao što je veštački inteligentni sistem rezultat izvršavanja računarskog programa na hardveru – računaru. Ovakve i slične analogije predstavljale su osnovu za dugogodišnje debate u naučnim krugovima oko njihovog stvarnog opravdanja, formirajući dva nepomirljiva tabora: zastupnike tzv. **jake i slabe hipoteze veštačke inteligencije**, o kojima će biti reč nešto kasnije.

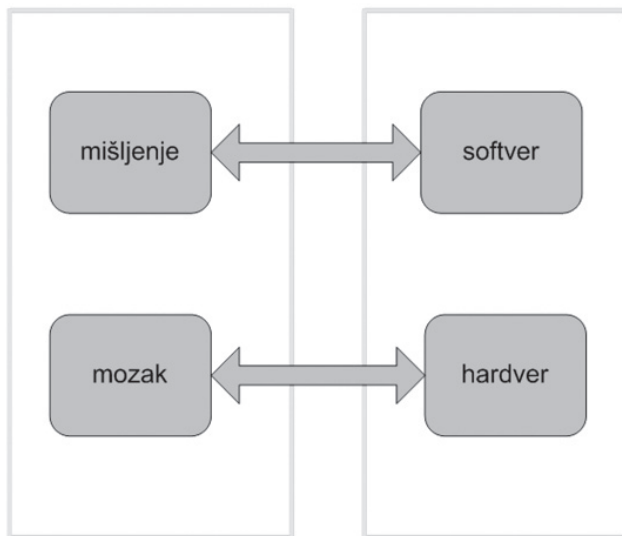
Veštačka inteligencija i kognitivne nauke se danas razvijaju nezavisno, upotpunjujući jedna drugu, posebno u oblastima vizije i prirodnih jezika. Vizija je posebno u skorije vreme napravila značajne pomake zbog integrisanog pristupa koji uzima u obzir i neuropsihološke dokaze i računarske modele.



---

Sl.1.7 Moguća interpretacija kognitivnih nauka kao reverznog inženjerstva, po kome se na osnovu teorije mišljenja, mogu identifikovati ekvivalentni programi koji se izvršavaju u našem mozgu.

---



---

Sl.1.8. Gruba ekvivalencija ljudskog mišljenja i softvera, odnosno biološkog mozga i hardvera u okviru identifikacije veze između veštačke inteligencije i kognitivnih nauka.

---

### 1.3 Pristup zasnovan na racionalnom mišljenju uz pomoć formalne logike

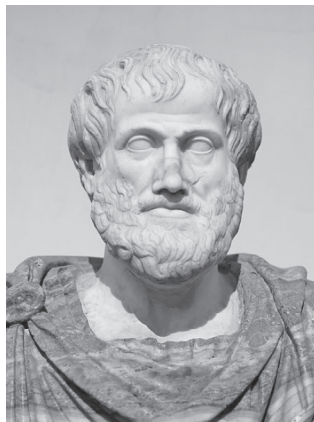
Aristotel je smatrao da ljudsko mišljenje ne može odražavati objektivnu stvarnost niti spoznati objektivnu istinu, ako samo nije postavljeno na sigurne principe. Pokazao je da se metodom indukcije, od pojedinačnog ka opštem, naš um uzdiže do pojmova koji su osnova racionalnog mišljenja.

Medutim, saznanje se ne zadržava samo na pojmu kao osnovnoj gradivnoj jedinici, već udruženi kroz jezičku formu iskaza ili sudova formiraju još oštije pojmove – kategorije. Da bi se ljudsko mišljenje moglo održati, ono mora imati osnovne principe, koji će biti opšte važeći i nije ih potrebno uvek ponovo dokazivati. Pronašao je tri takva principa:

- ♦ **princip identiteta** - sve što je istinito mora se potpuno samo sa sobom podudarati,
- ♦ **princip kontradikcije** - nemoguće je da se jednom i istom na isti način jedno i isto određenje dodaje i ne dodaje,
- ♦ **princip isključenja trećeg** - između kontradiktornih stavova ne može biti trećeg.

Ovo učenje predstavlja osnovu Aristotelove logike, koju je on sam nazivao analitika ili veština razdvajanja. Sudovi mogu biti pozitivni (istiniti) i negativni (lažni). Takođe je moguće iz opšteg, deduktivnim putem, doći do pojedinačnog, što se najbolje vidi u silogizmu: „Sokrat je čovek; svi ljudi su smrtni; dakle, Sokrat je smrtn.“ Silogističkom formom zaključivanja se iz dva stava - premisa izvodi treći stav - zaključak. Vrednost i značaj Aristotelove logike je jedinstven, i kada u narednim poglavljima budemo izučavali savremene sisteme automatskog rezonovanja, kao centralne oblasti klasične veštačke inteligencije, videćemo da su potpuno aristotelovski: na osnovu činjenica i pravila zaključivanja, deduktivnim metodama izvodimo zaključke i istinonosne sudove o činjenicama. Iako je današnja logika primenjena u sistemima veštačke inteligencije rezultat moderne matematičke logike Fregea, zasnovane krajem 19. veka, bez Aristotelovog temelja, ova impozantna matematička građevina ne bi bila moguća.

Već od sredine sedamdesetih godina 20. veka razvijeni su računarski programi koji su mogli da reše bilo koji rešiv problem opisan u logičkoj notaciji. Takozvani **logički pravac** u okviru veštačke inteligencije smatra da put do istinski inteligentnih sistema vodi preko usavršavanja ovakvih sistema automatskog rezonovanja.



Sl.1.9 Grčki filozof Aristotel (384–322 pre nove ere) utemeljivač logike. Smatrao je da ljudsko mišljenje može imati saznavnu snagu objektivne istine, samo ako je utemeljeno na čvrstim principima.

Da li ovakav pristup dozvoljava ozbiljne zamerke? Prva grupa problema proističe iz samog odredjenja logike. Za savremene logičare, to je učenje o valjanom zaključivanju, a ne učenje o ispravnom mišljenju. Ili drugačije rečeno, kakav je odnos između Fregeove matematičke logike i našeg mišljenja stvarnosti, posebno imajući u vidu pregnantnu Hajdegerovu tezu o sapripadnosti bića i mišljenja, ondosno stvarnosti i mišljenja. Videćemo iz narednih poglavlja da Godelova teorema nepotpunosti jednom za svagda stavlja tačku na raspravu o snazi formalnih sistema. Kada je u pitanju Fregeova matematička logika, to znači da je opis stvarnosti logičkim iskazima samo jedan od nepotpunih modela stvarnosti. Druga grupa problema proizilazi iz naše mogućnosti da pravimo efikasne programe za automatsko rezonovanje. Čak i problemi sa samo nekoliko desetina činjenica mogu da iscrpe resurse bilo kog savremenog računara. Donekle nije korektno da ove teškoće pripišemo samo logicističkom pravcu VI, budući da važe za bilo koji pokušaj izgradnje inteligentnog računarskog sistema.

## 1.4 Pristup zasnovan na racionalnim agentima

Agenti, koji su danas vrlo aktuelni na internetu, vode svoju etimologiju od latinske reči *agere* - raditi. U tom smislu, svaki program je agent, ali ono što se danas podrazumeva pod računarskim agentom su programi koji deluju i menjaju svoje okruženje, a pri tome poseduju autonomnu kontrolu, sposobnost percepcije i adaptacije. Racionalni agent je onaj agent koji se ponaša tako da ostvaruje najbolji mogući ishod za postavljeni cilj, ili kada postoji neki oblik neodređenosti, najbolji očekivani ishod.

Ova moderna paradigma VI potpuno odustaje od ideala da racionalni agent mora biti napravljen po uzoru na ljude i njihov način ostvarivanja ciljeva. S druge strane je jasno, da bi inteligentni agent koji prolazi Tjuringov test, zasigurno bio i racionalni agent. Opređeljivanje za kriterijumsku funkciju racionalnosti, koja je jasno definisana i potpuno uopštena, doprinosi da na razvoj inteligentnih sistema više utiče naučno tehnološki razvitak relevantnih disciplina od drugih pristupa baziranih na ljudskom ponašanju i ljudskom mišljanju. Ljudsko ponašanje, sa druge strane, je prilagođeno za specifično socio-bio-ekološko okruženje i rezultat je, delom komplikovanog i malo poznatog evolutivnog procesa koji je još uvek daleko od optimalnosti. Odnosno, princip optimalnosti jeste jedan od mogućih kriterijuma ove klase sistema, ali zasigurno nije jedini.

I pored svih dobrih strana ovog pristupa koji danas dominira oblašću VI, postoji snažan front unutar VI zajednice koji smatra da su na ovaj način izneverene izvorne ideje VI - prenošenje ljudskog načina delovanja i mišljenja na veštačke sisteme, odnosno stvaranje veštačkog humanoida koji bi prošao Tjuringov test. Odakle potiče ova snažna potreba čoveka za pravljenjem sopstvene kopije? Pokušaćemo da odgovorimo na to pitanje u narednom poglavlju.



## 1.5 O nužnosti veštačke inteligencije

Gotfrid Vilhelm Lajbnic, Sl.1.10, je bio jedan od retkih primera istinskih polihistora, odnosno rečeno običnim jezikom pravih sveznalica. Začetnik je ideje o univerzalnom jeziku nauke, odnosno logičkom računu koji bi obuhvatio sva znanja. Postavio je temelje savremene matematičke logike. Ono što je za nas bitno, jeste Lajbnicov uvid u uslove i nivoe ljudskog znanja.

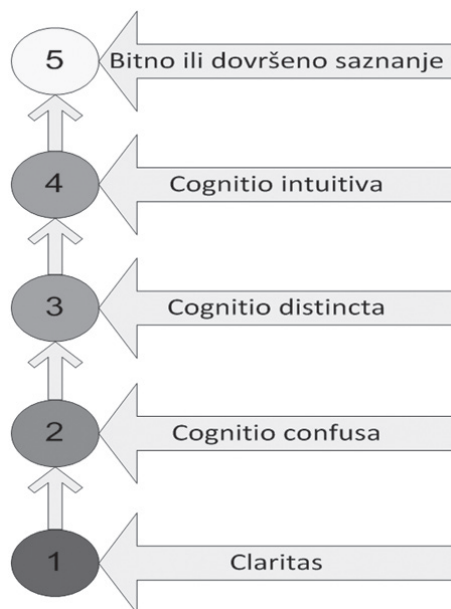
Stepeni saznanja po Lajbnicu, započinju najnižim nivoom, kada nam se predmet saznanja javlja uopšte kao prisutan. Tu prezentnost stvari Lajbnic prema Dekartu naziva **claritas**, Sl.1.11. Budući da nas sve pojedinačno susreće uvek iz sklopa povezanosti sa ostalim, da bi smo shvatili neku stvar moramo shvatiti šta je svojstveno samo za nju i šta je razlikuje od svega ostalog. Sledeći nivo saznanja mora da odgovori na to pitanje. Međutim, Lajbnic je pravilno uvideo, da precizno i tačno odgovoriti na ovaj zahtev, je zapravo naš ljudski put do pravog saznanja i da se on odvija u fazama, prolazeći kroz sve više novoe. Kada nam je samo maglovito jasno da se nešto razlikuje od ostalog, ali to ne znamo da izrazimo, Lajbnic naziva **cognitio confusa**, nerazdvojeno saznanje. U narednom stepenu napretka saznanja, odgovarajući na postavljeno pitanje o sopstvenim svojstvima i razlikama, ukoliko je odgovor obrazložen i jasan, takvo saznanje Lajbnic naziva **cognitio distincta**. Odredjenosti koje razlikuju jedno od drugoga su istovremeno one oznake jedne stvari po kojima možemo da je saznamo i prepoznamo. Dakle možemo reći da se saznanje sastoji u preciznim definicijama, koje na jedinstven način opisuju suštinska svojstva po kojima se neka stvar razlikuje od drugih. Ukoliko u formulaciji ove definicije koristimo samo one elementarne pojmove koji su već prethodno precizno definisani, dolazimo do **cognitio intuitiva** stepena saznanja, ili **adekvatnog saznanja**. Ovaj stepen saznanja se vrlo teško postiže. Najčešće se zadovoljavamo pojmovima koji su nam poznati, ali nisu razjašnjeni sve do njihovih pojmovnih elemenata. Ako nam takvo razjašnjenje i podje za rukom, jedva da smo u stanju da držimo prisutnim celokupni analitički sklop povezanosti elementarnih pojmova. Od svih nauka, jedino se matematika približila ovom idealu adekvatnog saznanja, budući da kao deduktivni formalni sistem zasnovan na početnim premisama – postulatima, generiše svoje objekte, u kojima su svi elementarni pojmovi već nasledjeni iz prethodnih objekata. Ukoliko su premise adekvatne, tada je i svo matematičko znanje adekvatno.

Tako se čini da je adekvatno intuitivno znanje najviši stepen saznanja. Međutim, znanje može da se podeli i prema tome da li u osnovi definicije predmeta saznanja imamo nominalnu ili realnu definiciju. Realna definicija dopušta dodatno da se sazna mogućnost stvari, odnosno mogućnost da egzistira, dok nominalna definicija ostavlja otvorenim to pitanje. Realna definicija može na dva načina da dopusti da se sazna mo-



Sl.1.10 Gotfrid Vilhelm Lajbnic  
(1646-1716)

gućnost definisane stvari. Prvi način dokazivanja je kroz iskustvo, jednostavno ukazivanjem na postojeću stvar. Realnost pojmova se dakle dokazuje empirijski, iskustveno. Međutim, ako bi takav dokaz sledio apriori, bez pozivanja iskustva u pomoć, nesumljivo bi realna definicija dostigla još jedan viši stepen saznanja. Kada je to uopšte moguće? Odgovor glasi: onda kada definicija u sebi istovremeno uključuje i saznanje mogućnog proizvodjenja stvari. Ako dakle, definicija daje način na koji stvar može da se proizvede, onda mi iz definicije saznajemo njenu realnu mogućnost i bez povratka ne činjenice iskustva – ako znamo da proizvedemo stvar onda znamo i da može da egzistira. Takva definicija je **kako realna tako i kauzalna**, i kao takva je najviši način saznanja uopšte. Ostaje još samo jedan stepen više, ako se i kod realnih i kauzalnih definicija možemo vratiti unazad do elementarnih pojmova. U tom slučaju to je kauzalno – realno, adekvatno saznanje, koje Lajbnic naziva **bitno ili dovršeno saznanje**.



Sl.1.11 Lajbnic svojoj *Raspravi o metafizici*, u 24. poglavlju, izlaže različite vrste i uslove saznanja, koje su važne za naše razumevanje potrebe za veštačkom inteligencijom.

U dve oblasti je uvek bilo saznanja koje u sebi samima sadrže pravilo proizvodjenja stvari: prvo u oblasti tehnike (techne – ono što je zgotovila ljudska ruka, staro grčki) i drugo u oblasti matematike. Npr. matematička definicija kruga sadrži zakon njegovog proizvodjenja, odnosno pravilo njegove konstrukcije.

Ako otvorimo prvu stranicu Aristotelove Metafizike, koja se smatra temeljnom knjigom evropske civilizacije, naći ćemo iskaz da čovek po prirodi teži saznanju. Ako je to saznanje usmereno ne samo na prirodu već i na čoveka samog, sledi da se u tom domenu najviši stepen saznanja može dostići bitnim ili dovršenim saznanjem o samom sebi, koje nužno uključuje i način na koji same sebe možemo proizvesti. Nije li to, zajedno sa Aristotelovim stavom o prirodnoj čovekovoj težnji, zapravo prirodna sila koja nas nužno vodi ka veštačkoj inteligenciji, koja u krajnjem cilju ima sintezu veštačkog humanoida. U tom krajnjem cilju sjedinjuje se zahtev dovršenog saznanja o nama samima. Stoga, možemo tvrditi da veštačka inteligencija nije samo jedna od disciplina računarskih nauka, ona je metodski put ka ostvarenju najvišeg stepena znanja o nama samima, a samim tim i o našem smislu i mestu u prirodi.



## Rezime 1. poglavlja

- ♦ Veštačka inteligencija je savremena oblast računarskih nauka čiji je osnovni cilj pravljenje sistema koji pokazuju inteligenciju.
- ♦ U podeli pristupa problemu veštačke inteligencije, posmatrane su dve nezavisne dimenzije: rezonovanje i ljudsko ponašanje, i to imajući u vidu kako to zaista rade ljudi, ili kako bi to bilo racionalno ostvariti, bez obzira na način ostvarivanja.
- ♦ Dominirajući pristup u današnje vreme je pristup racionalnog ponašanja veštačkih sistema, odnosno efikasna implementacija u odnosu na zacrtani cilj, pri čemu je ljudski način rešavanja istog problema u drugom planu. Ova tehnologija je danas poznata kao tehnologija racionalnih agenata.
- ♦ Ljudska kreativnost, dosledno primenjena na oblast VI, praktično znači izrazito inovativan pristup svim izazovima ove složene naučne discipline. Hermetika, razvijena u okviru filozofije i filozofske estetike, može nam biti od pomoći u sagledavanju osnovnih mehanizama konkreativnosti, koja leži u osnovi ljudskog napora da stvori nešto zaista radikalno novo.
- ♦ Lajbnicov pojam dovršenog znanja, daje nam jedno moguće objašnjenje nužnosti pojave veštačke inteligencije i njene vitalnosti, uprkos teškoćama i sporom napretku ka njenom konačnom cilju, veštačkom humanoidu.

## Pitanja i zadaci

1. Definišite svojim rečima (a) inteligenciju, (b) veštačku inteligenciju.
2. Koje primedbe imate na valjanost Tjuringovog testa?
3. Da li su refleksivna delovanja (recimo uzmicanja od vrele peći) racionalna? Da li su ona inteligentna?
4. U kojoj su meri sledeći sistemi primeri sistema VI:
  - a. skeneri bar kodova,
  - b. pretraživači veba,
  - c. telefonski meniji koji se aktiviraju glasom,
  - d. algoritmi rutiranja u računarskim mrežama koji uzimaju u obzir stanje mreže.
5. Zašto bi evolucija težila da daje sisteme koji deluju racionalno?
6. Da li je VI nauka ili tehnika?
7. Da li napredak VI zahteva razvoj novih metoda i pristupa kreativnim rešenjima?
8. Da li je pojam istine relevantan za sintezu racionalnih agenata?

## 2. Osnove veštačke inteligencije

Potraga za VI počinje sa snovima, kao što je sanjarenje početak svakog istinskog traganja. Ljudi su još od najstarijih vremena zamišljali mašine sa ljudskim sposobnostima, automate koji se kreću i naprave koje rezonuju kao ljudi. Humanoidne mašine su opisivane u mnogim bajkama, pesmama i romanima.

U ovom poglavlju daćemo kratak prikaz naučnih disciplina koje su imale najveći uticaj na razvoj oblasti VI. Cilj ovog prikaza je dvostruk. Prvo, stičemo uvid koliko je različitih naučnih disciplina participiralo u razvoju VI i koliko je vremena bilo potrebno za dostizanje kritičnog nivoa znanja neophodnog za ozbiljnije otvaranje osnovnih pitanja VI. Drugo, pokazaće se da jedan složeni poduhvat, kao što je sinteza veštačkih sistema sa svojstvima inteligencije, zahteva multidisciplinarnost, koja osim što menja poziciju u razmatranju nekog problema, istovremeno povratno doprinosi ubrzanom razvoju svake od pojedinačnih disciplina.

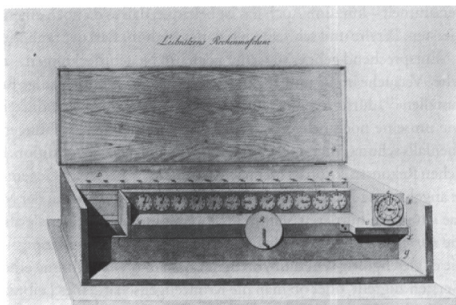
### 2.1 Filozofija

U Aristotelovoj Politici, paragraf 1254 a, nalazimo ovakav tekst: “Jer kad bi svako orudje samo moglo da radi svoj posao, bilo da mu je naredjeno, bilo da predoseti gospodarevu želju, kao što pričaju za Dajdalove statue i Hefajstove tronošce, za koje pesnik kaže da su sami išli na skupštinu bogova, i kada bi čunci mogli sami da tkaju a plektar sam da udara u kitaru, niti bi graditeljima bili potrebni pomoćnici niti gospodarima robovi.” Zadivljujuća je Aristotelova vizija automatizacije ljudskih sposobnosti uz istovremeno vrlo preciznu dijagnozu socijalnih posledica, koje kao da danas gubimo iz vida iako su Aristotelovi snovi višestruko realizovani i prevaziđeni. Možda nam ovaj tekst objašnjava i Aristotelovu centralnu misao da je čovek po prirodi sklon saznanju, a da je njegovo najjače orudje njegov razum. Valjano rezonovanje je praćeno preciznom formalnom strukturom koju je formulisao u svojim logičkim spisima Organon. Oni su i danas temelj propozicione logike.



Sl.2.1 Ramon Lull (1235-1316), Katalonski mistik i pesnik i njegova *Ars Magna* (Velika umetnost). Sastojala se od skupa papirnih diskova na koje su se upisivali na odgovarajući način pojmovi i atributi, a njihovom rotacijom bi se dobijali automatski odgovori na različita pitanja iz date predmetne oblasti.

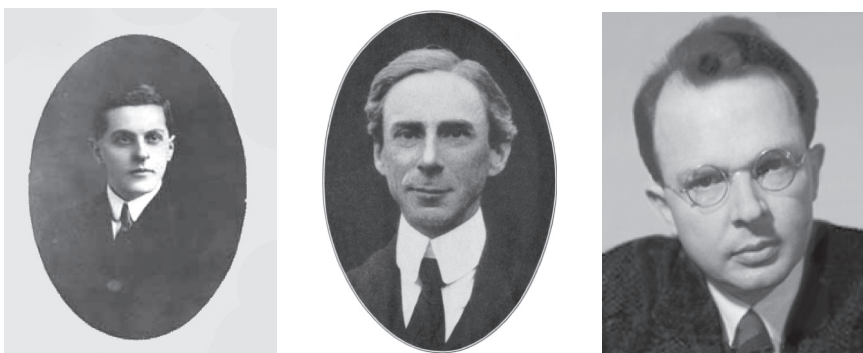
Ramon Lull je tek u XIII veku razvio ideju da se rasudjivanje može realizovati na mehaničkim napravama, Sl.2.1. Gotfried Vilhem Lajbnic je izradio mehanički, tzv univerzalni karakter, Sl.2.2, za koju je smatrao da ilustruje mogućnost obavljanja algebarskih operacija nad opštim konceptima. Lajbnicov san je bila zamisao da naš mozak upravo obavlja takav tip matematičkih operacija. Potsetimo se da je Lajbnicov doktorat u domenu pravnih nauka dokazivao tezu da se pravne procedure i odluke mogu ekvivalentirati logičkim operacijama.



Sl.2.2 Gotfried Vilhem Lajbnic (1646–1716) i njegov Univerzalni karakter.

Rudolf Karnap, Ludviga Vitgenštajna i Bernarda Rasela, koji su činili tzv. Bečki Krug, razvili su filozofiju logičkog pozitivizma, Sl.2.3. Na osnovu ove teorije sve ljudsko znanje može se opisati logičkim teorijama, koje obuhvataju i logičke iskaze opisa naših čulnih senzacija (tzv. opsevacione rečenice), i kao takve predstavljaju vezu sa svetom

koji nas okružuje. Teorija potvrđivanja Karnapa i Karla Hempela objašnjava kako se znanje stiče iz iskustva. Karnapovo delo *Logička Struktura Sveta* iz 1928 godine definisala je eksplicitnu računarsku proceduru za dobijanje znanja iz elementarnih iskustava. Ovim se zapravo logički pozitivizam Bečkog kruga transformisao u varijantu logičkog empirizma, koga možemo smatrati logičarskom varijantom Hjumobog empirizma. Za oblast VI ovo delo je značajno po tome što je u njemu prvi put pokušao ozbiljniji razvoj teorije razuma kao računarskog procesa.

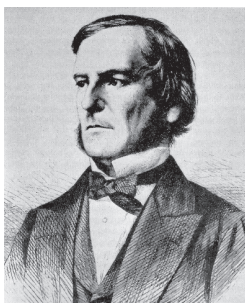


Sl.2.3 Bečki filozofski krug na čelu sa Rudolfom Karnapom je razvio doktrinu logičkog pozitivizma. Sa leva na desno: Ludwig Wittgenstein (1889-1951), Bertrand Russell (1872-1970) i Rudolf Carnap (1891-1970)

## 2.2 Matematika

Teorijski napredak u izučavanju pitanja pokrenutih u okviru VI, zahtevao je matematičke formalizacije u tri fundamentalne oblasti: u logici, računanju i verovatnoći.

Formalna logika započinje radovima grčkih filozofa, pre svega Aristotela, ali njen pravi matematički razvoj počinje radovima Džordža Bula koji je formulisao propoziciju ili Bulovu logiku, Sl.2.4. U osnovi ove ideje je zapravo davnašnji Lajbnicov san, da se formalizacija ljudskog mišljenja može realizovati odgovarajućom algebrom. Nemački matematičar Gotlib Frege je 1879. godine proširio Bulovu logiku tako da uključuje objekte i relacije, stvarajući logiku prvog reda koja se danas koristi u sistemima VI za reprezentaciju znanja i automatsko rezonovanje. U krajnjoj liniji programski jezik Prolog, omiljen u krugovima koji se bave inženjerijom znanja u okviru ekspertskih sistema u potpunosti sledi Fregeov formalizam, pa čak donekle i originalnu notaciju. Poljski matematičar i logičar Alfred Tarski uveo je teoriju reference koja pokazuje kako povezati objekat u logičkom sa objektima u realnom svetu. Autor je semantičke definicije istine u formalnim jezicima, vrlo bliske Aristotelovom stavu po ovom pitanju, naime stavu da je istina sadržana u sudu i da se odnosi na adekvatnost tvrdjenja (onoga šta se tvrdi) i same realne stvari na koju se to tvrdjenje odnosi u našem ljudskom svetu. Npr. Stav "Sneg je beo" je tačan ako i samo ako je sneg zaista beo. U formalnom smislu: "p" je istinito ako i samo ako je p, gde je p tvrdjenje iskazano u "p".



Sl.2.4 Matematička logika je tražila odgovore na pitanje: na osnovu kojih formalnih pravila se izvlače valjani zaključci. S leva na desno Džordž Bul (1815-1864), Gotlib Frege (1848-1925) i Alfred Tarski (1902-1983).

1900. godine, Dejvid Hilbert, Sl.2.5 predstavio je listu 23 problema za koja je tačno predvideo da će okupirati matematičare veći deo novog, XX veka. Poslednji na spisku Hilbertovih problema je pitanje da li postoji algoritam za odlučivanje istine o bilo kom logičkom iskazu koji uključuje prirodne brojeve – poznati *Entscheidung problem* ili problem odlučivanja. Hilbert je 1920. godine započeo istraživački projekt koji je postao poznat kao Hilbertov program. Cilj ovog programa je bila reformulacija matematike na čvrstim osnovama bez ikakvih logičkih kontradikcija. Smatrao je da se mogu dokazati dva stava:

- ♦ sva matematika proizlazi iz ispravno odabranog konačnog sistema aksioma,
- ♦ takav sistem aksioma je dokazivo konzistentan (neprotivrečan).

Ova dva stave bi bila osnova za preuredjeno matematičko zdanje bez kontradiktornih i nesigurnih teorija. Konsekvence bi bile dramatične: matematika bi se mogla generisati automatski kao čisti deduktivni sistem, i uz to u njoj ne bi bilo nikakve ljudske primese. To što matematičari veruju svojoj intuiciji prilikom dokazivanja matematičkih istina, pre bi se moglo sada nazvati slabošću a ne vrlinom. Srećom po ljudsku dimenziju matematike, Hilbertov program se pokazao neuspešnim. Ovaj udarac programu zadao je austrijsko – američki matematičar i logičar Kurt Gedel, koji je pokazao da svaki neprotivrečni formalni sistem koji bi bio dovoljno bogat da uključuje barem aritmetiku, ne može sam svojim aksiomima pokazati svoju potpunost. Godine 1931. njegova teorema nepotpunosti pokazala je da Hilbertov veliki plan od početka nije bio moguć. U tom radu je dokazao da za svaki izračunljiv aksiomatski sistem koji je dovoljno snažan da opiše aritmetiku prirodnih brojeva (na primer Peanove aksiome ili Zermelo-Frenkelova teorija skupova sa aksiomom izbora, tzv ZF sistem), važi:

- ♦ ako je sistem konzistentan, on ne može biti potpun
- ♦ konzistentnost aksioma ne može biti dokazana unutar sistema.

Teorema o nepotpunosti pokazuje da u bilo kom jeziku dovoljno ekspresivnom da opiše osobine prirodnih brojeva, postoje iskazi koje su neodlučivi, odnosno da njihova istinitost ne može da bude zaključena preko bilo kog algoritma odnosno u okviru date formalizacije. Ova teorema je okončala pola veka duge pokušaje da se pronade



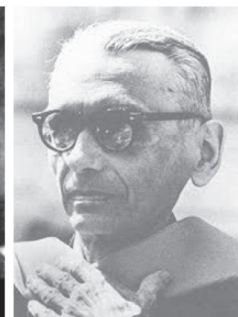
skup aksioma dovoljnih za zasnivanje celokupne matematike, koji su počeli radom Fregea, a završili Hilbertovim programom.

Ovaj fundamentalni za-  
ključak može biti protuma-  
čen i kao dokaz da postoje  
neke funkcije nad celim bro-  
jevima koje ne mogu biti izra-  
čunljive. Ovo je podstaklo  
Alena Tjuringa da pokuša  
da okarakterise izračunljive  
funkcije, iako ovaj pojam ne  
dozvoljava potpunu formal-  
nu definiciju. Generalno je  
prihvaćeno da je za ovo pita-  
nje relevantna tzv. Čerč-Tju-  
ringova teza, koja kaže da je  
Tjuringova mašina sposobna

da izračuna bilo koju izračunljivu funkciju. Drugim rečima ako je potrebno dokazati da je jedna funkcija izračunljiva, neophodno je napisati program za Tjuringovu mašinu, koji realizuje ovu funkciju, a zatim pratiti njen rad. Ako se pokaže ili dokaže da će program stati posle konačno ili beskonačno ciklusa rada mašine, funkcija je izračunljiva i obrnuto. Tjuring je odmah pokazao da postoje funkcije koje Tjuringova mašina ne može da izračuna. Na primer, ni jedna mašina ne može da kaže u principu da li će dati program dati odgovor na zadati ulaz ili će zauvek da radi (tzv. halting problem).



David Hilbert  
1862 - 1943



Kurt Gödel  
(1906-1978)

Sl.2.5 Hilbertov projekat je imao za cilj da dokaže da u matematici nema ničeg ljudskog, i da bi sjajna matematička građevina bila izgrađena na isti način i u nekoj vanzemaljskoj civilizaciji. Gedel je fundamentalnom teoremom o nepotpunosti pokazao da svaki formalni sistem generiše objekte za koje se ne može tvrditi ni da su istiniti ni da su neistiniti. Odluka o tome se donosi van sistema i to je tačka kroz koju u matematiku ulazi ljudki potpis i ljudski duh.

## Verovatnoća



Džerolamo Kardano  
(1501-1576)



Džejsms Bernuli  
(1654-1705)



Pjer Laplas  
(1749-1827)

Sl.2.6 Teorija verovatnoće je postala vodeća disciplina u modelovanju neodređenosti i nepotpunosti, koje su po pravilu prisutne u tipičnim problemima veštačke inteligencije

Osim logike i računanja, treći veliki doprinos matematike VI je teorija verovatnoće. Italijan Džerolamo Kardano je prvi dao ideju verovatnoće, opisujući je kao moguće ishode kockarskih događaja. Verovatnoća je uskoro postala važan deo kvantitativnih nauka, dajući formalan i operativni okvir za modelovanje raznovrsnih oblika neodređenosti, kako merenje tako i nepotpunih teorija i uverenja. Pijer Ferma, Blez Paskal, Džejms Bernuli, Pjer Laplas i drugi su razvili teoriju i uveli nove statističke metode, videti Sl.2.6. Tomas Bajes je predložio formulu (Bajesovo pravilo) koja opisuje način promene zadatih verovatnoća sa prispećem novih opservacija, Sl.2.7. Bajesovo pravilo i rezultujuća oblast nazvana Bajesova analiza formiraju osnove većine savremenih pristupa verovatnosnom zaključivanju u VI sistemima.

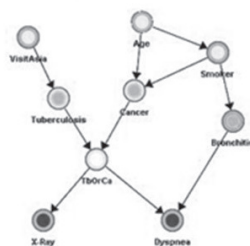
## Verovatnoća



Tomas Bajes  
(1702-1761)

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Bajesovo pravilo



Bajesovske mreže

Sl.2.7 Bajes je svojom čuvenom formulom rešio problem kako se menjaju naša prethodnih verovatnosna uverenja u skladu sa novim podacima. Moderna oblast Bajesovske mreže ili grafički verovatnosni modeli, predstavljaju front savremenog verovatnosnog modelovanja složenih sistema.

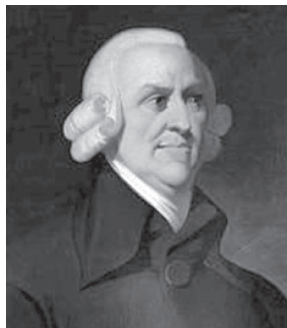
## 2.3 Ekonomija

Ekonomija kao naučna disciplina je započela 1776. godine radom *Ispitivanje Prirode i Uzroka Bogatstva Nacija* škotskog filozofa Adam Smita (1723-1790) u kome se ona posmatra kao da se sastoji od individualnih agenata koji maksimizuju svoje ekonomsko bogatstvo. Suština ekonomije se svodi na način donošenja odluka koje dovode do željenih ishoda, Sl.2.8.

Matematički opis „željenog ishoda” ili korisnosti je prvi put formalizovao Frank Ramzi (1931) i Džon fon Nojman sa Oskarom Morgensternom u knjizi *Torija Igara i Ekonomskog Ponašanja* (1944), Sl.2.9. Teorija odlučivanja, koja kombinuje teoriju verovatnoće sa teorijom korisnosti, daje formalan i kompletan okvir za donošenje ne samo ekonomskih odluka u uslovima nesigurnosti opisanih verovatnosnim zakonitostima.

Interakcija suprotstavljenih igrača povlači za sobom da akcije jednog igrača mogu značajno da utiču na korist drugog u pozitivnom ili negativnom smislu. Rezultati fon Nojmanove i Morgensternove teorije igara su često kontraintuitivni. Jedna od fundamentalnih posledica ove teorije, je rezultat poznatog matematičara Džona Neša, tzv. Nešov ekvilibrijum, Sl.2.10.

Nešov ekvilibrijum je koncept rešenja igre kod koga se podrazumeva da svaki igrač zna strategije ostalih igrača, i nijedan igrač ništa ne može da dobije tako što samo on jednostrano menja svoju strategiju. Ako je svaki igrač izabrao strategiju, i nijedan igrač ne može da profitira promenom svoje strategije, pod pretpostavkom da ostali igrači ne promene svoje strategije, onda trenutni skup izabranih strategija i odgovarajućih dobitaka predstavlja Nešov ekvilibrijum. Drugim rečima dva igrača se nalaze u Nešovom ekvilibrijumu, ako je svaki doneo najbolju moguću odluku uzevši u obzir odluku protivnika. Interesantno je da su ove odluke u suštini za svakog pojedinca suboptimalne, kada bi odlučivao bezuslovno. Razvojem teorije mehanizam dizajna, koja se bavi konstrukcijom igara (strategija) koje imaju zadate ekvilibrijume, značaj Nešovih ekvilibrijuma je značajno porastao. Dakle, ako želimo da jedan realan sistem dovedemo u želejno stanje, neophodno mu je nametnuti takvu stratešku igru, čiji su to Nešovi ekvilibrijumi. Nije potrebno detaljnije obrazložjenje koliko ovo može biti moćno oružje u savremenim globalnim socio ekonomskim konstelacija-



Sl.2.8 Adam Smit (1723-1790), škotski filozof, čijim radom *Ispitivanje Prirode i Uzroka Bogatstva Nacija*, započinje moderna ekonomska teorija, zasnovana na interakciji individualnih agenata koji maksimiziraju svoj dobitak.



Sl.2.9 Teorija korisnosti u okviru teorije odlučivanja izložena je prvi put u knjizi *Teorija igara i ekonomskog ponašanja* (1944) autora Fon Nojmana (1903-1957) i Oskara Morgensterna (1902-1977). Isti autori se smatraju rodonačelnicima savremene Teorije igara, koja matematički modeluje široki spektar praktičnih situacija u kojima se suočavaju nezavisni agenti sa suprotstavljenim ciljevima.



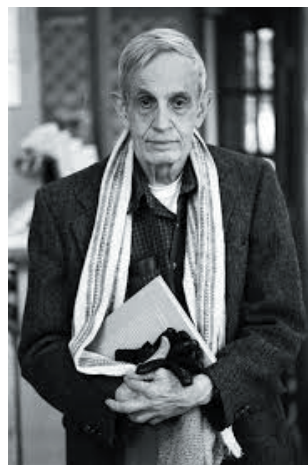
ma, za koje naivni posmatrač ima utisak da su usled svoje složenosti, neupravljive i podložne nepredvidivim slučajnim fluktuacijama.

Većinom ekonomisti se nisu doticali trećeg pitanja koje je navedeno, odnosno kako praviti racionalne odluke kada isplata akcije nije trenutna, već je umesto toga rezultat nekoliko akcija napravljenih u sekvenci. Ova tema je razrađena u oblasti operacionih istraživanja koja je nastala u Drugom svetskom ratu iz pokušaja engleza da optimizuju radarske instalacije, a kasnije je našla civilnu upotrebu u kompleksnim poslovnim odlučivanjima. Rad Richard Bellman-a (1957) formalizovao je klasu sekvencijalnih problema odlučivanja nazvanih Markovljevi procesi odlučivanja.

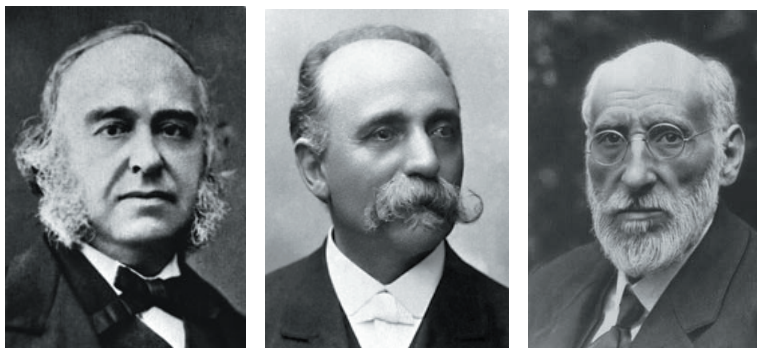
Rad u ekonomiji i operacionim istraživanjima je doprineo boljem razumevanju pojmova racionalnih agenata, iako se niz godina istraživanje ove teme u okviru VI razvijalo na drugačiji način. Jedan od razloga bila je kompleksnost donošenja racionalnih odluka. Herbert Simon (1916-2001), jedan od najpoznatijih istraživača VI, dobio je Nobelovu nagradu iz ekonomije 1978. godine za njegove rane radove koji pokazuju da modeli bazirani na zadovoljenju – donošenju odluka koje su “dovoljno dobre” umesto računanja optimalnih odluka – daju bolje opise stvarnog ljudskog ponašanja. Teorija odlučivanja zasnovana na agentima ponovo je aktuelizovana 1990-tih godina.

## 2.4 Neuronauka

Neuronauka je nauka o nervnom sistemu, posebno mozgu. Tačan način na koji mozak omogućava misao je jedna od najvećih misterija nauke. Hiljadama godina se smatralo da mozak na neki način utiče na misao, zbog dokaza da jaki udarci u glavu mogu da dovedu do gubitka mentalnih sposobnosti. Ipak, tek sredinom 18. veka mozak je postao priznat kao središte svesti. Pre toga, kandidati za tu ulogu su uključivali srce, slezinu i pinaelnu žlezdu.

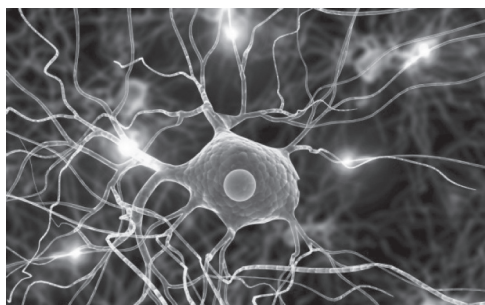
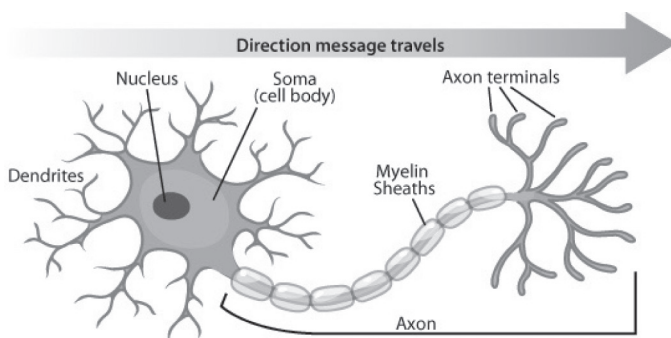


Sl. 2.10. Džon Neš (1928 - ), američki matematičar. Specifični ekvilibrijumi u simultanim igrama sa dva i više igrača, dobili su po njemu naziv Nešovi ekvilibrijumi. Za ovaj rezultat je dobio i Nobelovu nagradu za ekonomiju 1994. godine, mada ga lično, kao matematičar smatra svojim najtrivijalnijim rezultatom. Snaga i značaj Nešovih ekvilibrijuma leži u činjenici, da naše ljudsko odlučivanje po prirodi preferira odluke koje nas neminovno vode u ove ekvilibrije. Interesantno je da ove odluke nisu bezuslovno optimalne sa stanovišta svakog pojedinca. Da li je to znak da u našem urođenom ponašanju ima i nečega što može biti osnova solidarnosti i kolektizma?



Sl. 2.11 Paul Broca-ina (1824-1880), Camillo Golgi (1843-1926) i Santiago Ramon y Cajal (1852-1934), pioniri neuronauke.

Studija Paul Broca-ina (1824-1880) iz 1861. godine o afaziji, jednoj vrsti defekta govora kod pacijentata sa oštećenjem mozga, dovela je do saznanja da postoje lokalizovani delovi mozga odgovorni za specifične kognitivne funkcije, Sl.2.11. Pokazao je da je produkcija govora lokalizovana na deo leve hemisfere mozga, koja je po njemu nazvana Broca-ova oblast. U to vreme već je bilo poznato da se mozak sastoji od nervnih ćelija - neurona, ali je tek 1873. Camillo Golgi (1843-1926) razvio tehniku bojenja koja je omogućavala posmatranje pojedinih neurona, Sl. 2.12. Ovu tehniku je koristio Santiago Ramón y Cajal (1852-1934) u svojim radovima o neuralnoj strukturi mozga.



Sl. 2.12 – Nervna ćelija ili neuron

Merenje funkcija mozga je aktivno počelo 1929. godine sa Hans Berger-ovim izumom elektroencefalograma (EEG). Skoriji razvoj funkcionalne magnetne rezonance daje neuronaučnicima detaljne slike moždane aktivnosti, omogućavajući merenja koja korespondiraju sa tekućim kognitivnim procesima. Uprkos ovim napretcima, još uvek smo daleko od razumevanja kako bilo koji od ovih kognitivnih procesa stvarno funkcioniše. Ono što i danas zadivljuje istraživače je da skup jednostavnih nervnih ćelija predstavlja biološku osnovu svih naših mentalnih procesa i može da dovede do misli, akcija i svesti, ili drugim rečima, da mozak uzrokuje um, kako je to formulisao Searle 1992. god. Jedina realna alternativa ovoj teoriji je misticizam po kome naši umovi funkcionišu izvan poznatih fizičkih zakona.

Ljudski mozak i digitalni računari rade različite zadatke i imaju različite osobine. Tabela 2.1 pokazuje da postoji 1000 puta više neurona u tipičnom ljudskom mozgu nego što postoji kapija u centralnom procesoru (Central Processor Units - CPU) tipičnog brzog računara. Murov zakon predviđa da će broj CPU kapija biti jednak broju neurona u mozgu do 2020. Naravno, malo toga može biti zaključeno iz takvih predviđanja; šta više, razlika u kapacitetu memorisanja je mala u poređenju sa razlikom u prelaznoj brzini i paralelizmu. Računarski hardver može da izvrše instrukciju u okviru jedne nanosekunde, dok neuroni to rade milion puta sporije. Mozak međutim ovo nadoknađuje tako što su svi neuroni i sinapse aktivne istovremeno, dok većina savremenih računara ima samo jedan, ili najviše nekoliko procesora. Stoga, iako je računar milion puta brži u transferu podataka, mozak efektivno radi 100000 puta brže u odnosu na svaremeni računarski hardver.

|                             | Računar                        | Ljudski mozak             |
|-----------------------------|--------------------------------|---------------------------|
| <b>Procesorske jedinice</b> | 1 CPU, 10 <sup>8</sup> gejtova | 10 <sup>11</sup> neurona  |
| <b>Memorijske jedinice</b>  | 10 <sup>10</sup> bita RAM      | 10 <sup>11</sup> neurona  |
|                             | 10 <sup>11</sup> bita disk     | 10 <sup>14</sup> sinapsi  |
| <b>Vreme ciklusa</b>        | 10 <sup>-9</sup> sekundi       | 10 <sup>-3</sup> sekundi  |
| <b>Brzina obrade</b>        | 10 <sup>10</sup> bita/sec      | 10 <sup>14</sup> bita/sec |
| <b>Brzina memorisanja</b>   | 10 <sup>9</sup> bita/sec       | 10 <sup>14</sup> bita/sec |

Tabela 2.1 – Grubo poređenje računarskih resursa dostupnih računarima i ljudskom mozgu. Prošlost nas uči da tehnološki napredak u domenu računara dosta dobro prati stopu rasta sa faktorom 10 na svakih 5 godina. Kvantitativni pokazatelji kod ljudskog mozga se nisu promenile u zadnjih 10.000 godina.

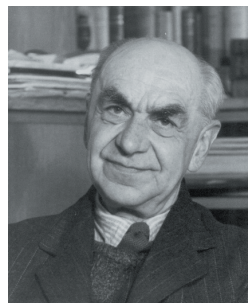
## 2.5 Psihologija

Poreklo naučne psihologije je uglavnom povezano sa radom nemačkog fizičara Hermann von Helmholtz-a (1821-1894) i njegovog studenta Wilhelm Wundt-a (1832-1920). Helmholtz je primenio naučne metode izučavanja ljudske vizije, a njegov *Priručnik Psihološke Optike* se i danas smatra kao najznačajnija rasprava o fizici

i psihologiji ljudske vizije. 1897. godine Wundt je otvorio prvu laboratoriju eksperimentalne psihologije na univerzitetu Leipzig. Insistirao je na pažljivo kontrolisanim eksperimentima u kojima ispitanici obavljaju perceptualne ili asocijativne zadatke prateći introspektivno odgovarajuće indukovane procese razmišljanja. Iako je psihologija već dobila status naučne discipline, neizbežna subjektivnost introspektivnih zapažanja uvek je ostavljala odškrinuta vrata sumnji u objektivnost izvedenih zaključaka. Koristeći takvo gledište, bihevioristički pokret, predvođen John Watson-om (1878-1958), odbijao je bilo kakvu teoriju koja uključuje mentalne procese budući da introspekcija ne može dati verodostojne dokaze. Mentalne konstrukcije kao što su znanje, vera, cilj i rezonovanje su odbačene kao nenaučna „narodna psihologija“. Bihevioristi insistiraju na izučavanju samo objektivnih merenja stimulusa i rezultujućih odziva, što je moglo biti primenjeno samo u eksperimentima sa životinjama. Stoga je biheviorizam otkrio mnoge istine o miševima i golubovima, a vrlo malo o ljudima. Uprkos tome, on je imao veliki uticaj na psihologiju, posebno u Sjedinjenim Državama, između 1920. i 1960. godine.

Stav da je mozak sistem koji obrađuje podatke, što je osnovna karakteristika kognitivne psihologije, može se naći još u radovima William James-a (1842-1910). Helmholtz je takođe insistirao da percepcija uključuje formu nesvesnog logičkog zaključivanja. Kognitivno modelovanje je dalje razvijano na odeljenju za primenjenu psihologiju univerziteta Cambridge koje je vodio Frederic Bartlett-a (1886-1969). Kenneth Craik, Bartlett-ov studenta i naslednik je 1943. godine objavio uticajnu knjigu *Priroda objašnjenja* kojom je obnovljena legitimnost mentalnih stanja i „mentalnih“ izraza kao što su verovanja i ciljevi, Sl.2.13. Craik je definisao tri ključna koraka kod agenata zasnovanih na znanju: (1) nadražaj mora biti preveden u internu reprezentaciju, (2) reprezentacija je manipulisana kognitivnim procesima čija je svrha nova interna reprezentacija, i (3) one su prevedene nazad u akciju. On je jasno objasnio zašto je ovo dobar dizajn za agenta.

Ako organizam nosi umanjeni model spoljašnje realnosti i svojih mogućih akcija u svojoj glavi, u mogućnosti je da isproba razne alternative, zaključuje koja je najbolja, reaguje na buduće situacije pre nego što nastanu, koristi znanje o prethodnim događajima u odlučivanju u sadašnjosti ili budućnosti, i u svakom pogledu reaguje na mnogo puniji, bezbedniji i kompetentniji način na hitne situacije sa kojima se susreće. (Craik, 1943)



Sl.2.13. Frederic Bartlett-a (1886-1969), Kenneth Craik (1914-1945) i Donald Broadbent (1926-1993), zastupnici kognitivnog modelovanja.

Posle Craik-ove smrti u biciklističkoj nezgodi 1945. godine, njegov rad je nastavio Donald Broadbent, čija je knjiga *Percepcija i Komunikacija* (1958.) uključivala neke od prvih modela informacionog obrađivanja psiholoških fenomena. U međuvremenu, u Sjedinjenim Američkim Državama, razvoj računarskog modelovanja je doveo do stvaranja kognitivne nauke. Za ovu oblast se smatra da je započela na radionici u septembru 1956. na MIT-u, dva meseca pre konferencije na kojoj je nastala VI. Na radionici George Miller je prezentovao *Magični Broj Sedam*, Naom Chomsky je prezentovao *Tri Modela Jezika* i Alan Newell i Herbert Simon su prezentovali *Logičku Teoriju Mašine*. Ova tri značajna rada su pokazala kako računarski modeli mogu biti korišćeni za psihologiju memorije, jezika i logičnog razmišljanja, respektivno. Među savremenim psiholozima danas prevladjuje stav da kognitivna teorija treba da bude kao računarski program, odnosno, treba da opisuje detaljni mehanizam obrade informacija na osnovu kojih je moguće realizovati odgovarajuće kognitivne funkcije.

## 2.6 Računarstvo

Fizičku osnovu na kojoj se implementira veštačka inteligencija predstavljaju savremeni digitalni računari. Naučnici su ih konstruisali prvi put, nezavisno i skoro istovremeno, u tri zemlje uključene u Drugi svetski rat. Prvi računar u operativnoj upotrebi je bio elektromehanički Heath Robinson, koga je 1940. konstruisao tim Alan Turing-a za potrebe dekrptiranja nemačkih šifrovanih poruka. Ista grupa je 1943. godine razvila Colossus, snažan računar opšte namene zasnovanu na vakumskim cevima. Prvi programibilni računar u operativnoj upotrebi je bio Z-3, izum Konrad Zuse-a u Nemačkoj 1941. godine. Zuse je takođe izumeo cifre sa pokretnim zarezom i prvi programski jezik visokog nivoa, PLANKALKIL. Prvi elektronski računar, ABC, je sastavio John Atanasoff-a i njegov student Clifford Berry između 1940. i 1942. godine na univerzitetu u Ajovi. Atanasoff-ovo istraživanje je dobilo malo podrške i priznanja, dok je ENIAC napravljen kao tajni vojni projekat na univerzitetu u Pensilveniji bio najuticajnija preteča savremenih računara. U timu koji je razvijao ENIAC su bili i John Mauchly i John Eckert.

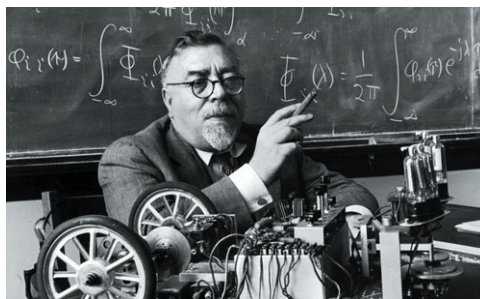
U toku poluvekovnog razvoja računarske tehnike, svaka generacija računarskog hardware-a je donela povećanje u brzini i kapacitetu i smanjenje u ceni. Performanse se dupliraju svakih 18 meseci. Ovakav trend treba očekivati i u naredne dve decenije, bez potrebe da se temeljno menja osnovna računarska tehnologija zasnovana na poluprovodničkim elementima.

Na razvoj VI je uticala i softverska strana računarskih nauka, odgovorna za operativne sisteme, programske jezike i alate potrebne za razvoj savremenih programa. Interesantno je da neki od osnovnih dostignuća u ovoj oblasti potiču od napora u domenu VI, kao što su: vremensko deljenje, interaktivni interpreteri, okruženja za brz razvoj softvera, tip podataka sa povezanim listama, automatska kontrola memorijskog prostora, kao i ključne koncepte simboličnog, funkcionalnog, dinamičkog i objektno orjentisanog programiranja.



## 2.7 Teorija upravljanja i kibernetika

Utemeljivač teorije upravljanja je Norbert Wiener (1894-1964) matematičar koji je radio sa Bertrand Russell-om, između ostalih, pre nego što je razvio interesovanje za biološke i mehaničke sisteme upravljanja i njihovu vezu sa saznanjem, Sl.2.14. Kao i Craik, koji je takođe koristio upravljanje procesima kao psihološke modele, Wiener i njegove kolege Arturo Rosenbluth i Julian Bigelow su osporavali biheviorističku filozofiju.



Sl.2.14 Norbert Wiener (1894-1964), tvorac moderne teorije upravljanja. Njegova knjiga Kibernetika iz 1948. godine postala je bestseller i najavila mogućnost veštačke inteligencije.

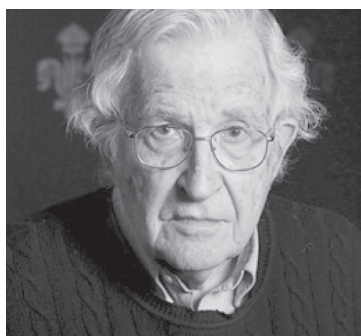
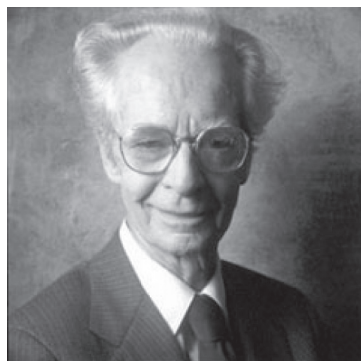
Smatrali su da hotimično ponašanje nastaje iz mehanizma upravljanja koji pokušava da minimizuje grešku – razliku između trenutnog i željenog stanja. Kasnih 1940ih, Wiener je zajedno sa Warren McCulloch-om, Walter Pitts-om i John von Neumann-om organizovao seriju konferencija koje su podsticale istraživanja novih matematičkih i računarskih modela mišljanja. Wiener-ova knjiga Kibernetika (1948.) postala je bestseller i najavila je mogućnost veštačkih inteligentnih mašina. Moderna teorija upravljanja, posebno grana poznata kao “stohastikčko optimalno upravljanje” ima za cilj sintezu sistema koji maksimizuju kriterijumsku funkciju tokom vremena. Ovo se grubo poklapa sa ciljevima VI: sinteza sistema koji se ponašaju optimalno. Ipak su se VI i teorija upravljanja razvijale sasvim nezavisno. Jedan od razloga je uprošćavanje matematičkih opisa sistema kojima se u praksi realno moglo upravljati. Za VI su upravo preostali oni praktični problemi koji se nisu mogli uspešno tretirati pojednostavljenim matematičkim modelima teorije upravljanja, ili se principijelno nisu uklapali ni u kakvu do tada poznatu formalizaciju, kao što su prirodni jezici, vizija, planiranje, igre i sl.

## 2.8 Lingvistika

Burrhus Fredric Skinner (1904-1990) je 1957. godine, objavio knjigu *Verbalno Ponašanje*, Sl.2.15. Ovo je bio sveobuhvatni prikaz biheviorističkog pristupa jeziku i učenju. Prikaz ove knjige široj publici napisao je lingvističar Noam Chomsky, koji je upravo objavio knjigu o svojoj novoj teoriji *Sintaksne Strukture*. Prikaz je bio sve drugo osim pohvale biheviorizmu, i ubrzo je postigao istu popularnost kao i sama knjiga na koju se prikaz odnosio. Chomsky je pokazao kako bihevioristička teorija ne uzima u obzir kreativnost jezika – ne objašnjava kako dete može da razume i izmišlja rečenice koje nikad ranije nije čulo. Chomsky-jeva teorija, zasnovana na sinaptičkom modelu koji datira još od Indijskog lingviste Panini-ja (oko 350. p.n.e), može ovo da objasni

i uz to još je i dovoljno formalna da bi u principu mogla biti isprogramirana.

Moderna lingvistika i VI su nastale gotovo u isto vreme, razvijale se istovremeno, susrećući se u hibridnom polju nazvanom računarska lingvistika ili obrada prirodnih jezika. Za problem razumevanja jezika se ubrzo shvatilo da je mnogo kompleksniji nego što se to činilo 1957. godine. Razumevanje jezika zahteva razumevanje suštine stvari i konteksta, a ne samo razumevanje strukture rečenica. Ovo možda deluje očigledno, ali nije bilo opšte poznato 1960-tih godina. Rani radovi na reprezentaciji znanja u okviru VI bili su u velikoj meri vezani za istraživanja i rezultate lingvističara. Budući da je zajednički koren ovim istraživanjima filozofija jezika, može se slobodno reći da pravog prodora u domenu efikasnog kodovanje znanja neće biti bez dubljeg razumevanja svih problema koje su pokrenuli tzv. filozofi jezika dvadestetog veka. Filozofija jezika je grana filozofije koja proučava prirodu jezika, njegovo poreklo i upotrebu.



Sl.2.15 Burrhus Fredric Skinner (1904-1990)  
i Noam Chomsky (1928 - )

Filozofija jezika unutar kontinentalne filozofije vodi poreklo od Ferdinanda de Saussure čiji projekt pokazuje veliku sličnost s Fregeovim. Filozofi kontinentalne filozofije Edmund Husserl, Martin Heidegger i Jacques Derrida smatraju se važnim predstavnicima ove filozofske grane.

## Rezime 2. poglavlja

- ♦ Današnji razvoj VI je rezultat viševjekovnih napora naučnika i istraživača u širokom spektru disciplina: od filozofije, preko matematike, ekonomije, neuronauka, psihologije, računarstva, teorije upravljanja pa do lingvistike.
- ♦ Filozofija je od početka svog nastanka postavljala pitanja o čoveku i njegovim intelektualnim sposobnostima, načinu percepcije spoljašnjeg sveta, njegovoj internoj reprezentaciji, racionalnom mišljenju i odlučivanju.
- ♦ Matematika je dala formalni aparat za logičko rasudjivanje, verovatnosno modelovanje i široki spektar optimizacionih procedura.
- ♦ Ekonomija je formalizovala procese donošenja odluka i uvela koncepte nezavisnih agenata koji interaguju tako da minimiziraju svoje lokalne kriterijumske funkcije i preferencijale.



- ♦ Neuronauka je svojim rezultatima doprinela boljem shvatanju kako radi čovekov biološki sistem nastao evolutivnim putem, a koji već unapred poseduje sva svojstva koja se zahtevaju od sistema veštačke inteligencije.
- ♦ Psiholozi su doprineli razvoju koncepta po kome se i ljudi i životinje mogu ekvivalentirati sistemima za procesiranje informacija.
- ♦ Računarska tehnika je obezbedila hardversko softversku osnovu za realizaciju VI sistema.
- ♦ Teorija upravljanja i kibernetika je doprinela boljem razumevanju sinteze optimalnih sistema, koji ekstremizuju unapred zadate kriterijumske funkcije.
- ♦ Budući da su jezički elementi nosioci znanja i da se u jeziku odigrava naš susret sa spoljnjim svetom indukujući inteligentno delovanje, lingvistika predstavlja naučnu disciplinu tesno povezanu sa VI.

## **Pitanja i zadaci**

1. Da li su filozofski napori u cilju formalizacije ispravnog mišljenja relevantni za savremenu VI?
2. Zašto je lingvistika kao nauka o jeziku važna za VI?
3. Da li Godelovi rezultati o nepotpunosti formalnih sistema imaju uticaja na razvoj VI?
4. U čemu je suština Bajesove teoreme (formule)?
5. Šta je smisao Nešovih ekvilibrijuma?
6. Kako biste svojim rečima opisali značaj Craik-ovih rezultata u vezi racionalnih agenata zasnovanih na znanju.
7. Svojin rečima opišite smisao apriornih Bajesovskih verovatnoća?
8. Koji bi budući rezultati u razumevanju funkcionisanja ljudskog mozga imali najveći uticaj na razvoj VI?
9. Ako pretpostavimo da odlučujućem razvoju VI nedostaje jedna ključna karika, kojoj naučnoj oblasti bi ona po vama pripadala?



## 3. Kratka istorija veštačke inteligencije

U ovom poglavlju biće dat kraći pregled istorijskog razvoja VI. Upoznavanje sa istorijom VI nije značajno samo sa stanovišta sticanja opšteg utiska o naporu više generacija istraživača u ovoj oblasti. Istorija nam daje uputstva kako ne ponavljati greške i kako se menjaju zahtevi i ciljevi jedne discipline u zavisnosti od tehnološkog razvoja i opštih ekonomskih prilika. Dok je rani razvoj VI bio dominantno diktiran vojnim programima i opštom klimom hladnog rata, kraj prošlog i početak ovog veka diktira zahteve proizašle iz internet i web tehnologije na čijoj osnovi niče nova digitalna ekonomija. Obuhvaćen je vremenski period od pojave prvih digitalnih računara pedesetih godina prošlog veka pa do danas. Zapaža se zatvaranje jednog čitavog ciklusa razvoja. Oblast je započela postavljanjem za cilj sinteze sistema koji pokazuju svojstva ljudske inteligencije, da bi smo danas ponovo imali oživljavanje ovog istog pristupa kroz zahtev za HLAI (Human Level Artificial Intelligence) sistemima.

### 3.1 Početak oblasti veštačke inteligencije i rani entuzijazam

Ako mašine pretenduju na inteligentno ponašanje, one moraju u najmanju ruku, biti u stanju da deluju i razmišljaju na ljudski način. Stoga je prvi korak u razvoju VI bio usmeren na identifikaciji nekih specifičnih zadataka koji zahtevaju inteligenciju prilikom rešavanja, sa ciljem transfera ovakvog stila rešavanja na mašine. Rešavanje zagonetki, igranje različitih igara kao što su šah i mica, dokazivanje teorema, odgovaranje na jednostavna pitanja i razvrstavanje vizuelnih slike su neki od takvih problema kojima su se bavili pioniri ove oblasti tokom 50-tih i 60-tih godina XX veka. Iako je većina ovih problema po svojoj složenosti odgovaraju „igračkama“ u odnosu na realne probleme, rana istraživanja u ovoj oblasti su dotakla i neke sasvim realne praktične probleme, kao što su automatsko prevodjenje sa jednog jezika na drugi ili prepoznavanje rukom pisanih znakova na bankovnim čekovima. Sejmur Papert je prvi upotrebio reč “problem-igračka“ 1967. godine razvrstavajući sve probleme unutar oblasti VI na tzv. tau (igračka), ro (realne) i teta (teorijske) probleme. Ova podela i danas ima sasvim zadovoljavajući smisao.

Pojava VI kao ravnopravne oblasti istraživanja koincidira sa tri značajne konferencije održane 1955, 1956. i 1958. godine. To su Sesija o mašinskom učenju u okviru Zajedničke zapadne računarske konferencije, održana u Los Andjelesu 1955. godine (Session on Learning Machine in Western Joint Computer Conference in Los Angeles), Letnji istraživački projekat iz veštačke inteligencije, održan 1955. god. na Datmut koledžu (Summer Research Project on Artificial Intelligence at Dartmouth College) i simpozijum pod nazivom Mehanizacija procesa razmišljanja održana 1958. godine u Engleskoj, pod pokroviteljstvom Nacionalne laboratorije za fiziku (Mechanization of Thought Processes, sponsored by National Physical Laboratory in the United Kingdom).

Letnja škola na Datmut koledžu nije u stručnom pogledu dala značajnije rezultate, ali predstavlja kulturni događaj iz dva razloga. Prvo na njoj se prvi put okupila grupa tada mladih istraživača, uglavnom Šenonovih bivših studenata, koji su odigrali presudnu ulogu u razvoju oblasti (John McCarthy, Marvin Minsky, Nathaniel Rochester, Arthur Samuel, Oliver Selfridge, Roy Solomonoff, Allen Newell, Herbert Simon), Sl.3.1. Drugo, Makarti je na ovom skupu prvi put upotrebio naziv Veštačka inteligencija, koji je zatim prihvaćen kao ime za celokupnu novu naučnu oblast.



Sl.3.1 Neki od utemeljivača VI, na četrdesetogodišnjici Datmut letnje škole, 2006. godine. Sa leva na desno: Trenchard More, John McCarthy, Marvin Minsky, Oliver Selfridge i Ray Solomonoff.

U ovom periodu razvoja VI, dominirala su dva pravca. Jedan, nošen prvobitnim entuzijazmom i idejom da u osnovi ljudske sposobnosti rešavanja problema leži mali broj opštih principa, čijim otkrivanjem bi se došlo do svetog grala oblasti, odnosno opšteg rešavača svih problema (General Problem Solver, GPS). Na problemu GPS radili su Newell i Simon, podstaknuti pionirskim uspehom u razvoju programa Logički Teoretičar (Logic Theorist), namenjenog dokazivanju teorema propozicione logike, polazeći od pet aksioma, kako je to izloženo u klasičnom radu *Principia Mathematica*, Vol.1, Russela i Whiteheada, Sl.3.2. U okviru ograničene klase slagalica koje može da reši, ispostavilo se da je red po kom program bira podciljeve i moguće akcije sličan načinu na koji čovek pristupa istom problemu. Zato se GPS smatra jednim od prvih programa koji je imitirao ljudskog razmišljanja. Uspeh GPS i kasnijih programa kao modela razmišljanja doveo je Newell-a i Simona-a do formulacije čuvene hipoteze

fizičkih simboličkih sistema, koja kaže da fizički simbolički sistem ima potrebnu i dovoljnu snagu da generiše inteligentnu akciju. U osnovi ove hipoteze je uverenje da suštinu inteligencije, bez obzira da li se ostvaruje u biološkim ili veštačkim sistemima, predstavlja ekvivalentna manipulacija strukturama simbola.



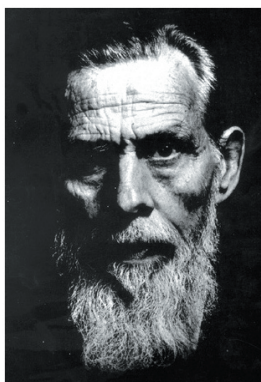
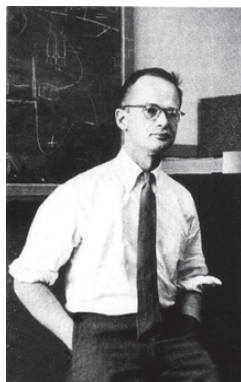
Sl. 3.2 Allan Newell (1927 - ) i Herbert Simon (1916-2001), tvorci prvog programa namenjenog opštem rešavanju problema (General Problem Solver, GPS).

Nathaniel Rochester i njegove kolege iz IBM-a su napravili neke od prvih VI programa. Herbert Gelernter (1959) je napravio Dokazivač Geometrijskih Teorema, koji je mogao da dokazuje teoreme čiji bi dokazi bili teški i za mnoge studente matematike. Počevši od 1952. godine, Arthur Samuel je napisao niz verzija programa za igru Dame, koji je na kraju igrao bolje od njegovih kreatora.

John McCarthy je dao niz važnih doprinosa samo u jednoj, 1958. godini. Definisao je jezik LISP, koji je postao dominantan VI programski jezik. LISP je drugi najstariji veliki jezik visokog nivoa koji se trenutno koristi, jednu godinu mlađi od FORTRAN-a. Iste godine objavio je rad pod nazivom *Programi sa Zdravim Razumom*, u kome je opisao Uzimač Saveta (Advice Taker), hipotetički program koji može da se posmatra kao kompletni VI sistem. Kao i Logički Teoretičar i Dokazivač Geometrijskih Teorema, McCarthy-jevi programi su pravljani tako da koriste znanje u potrazi za rešenjima problema. Uzimač Saveta je već sadržavao centralne principe reprezentacije znanja i automatskog rezonovanja: formalna eksplicitna reprezentacija sveta i načina na koji agentove akcije utiču na svet, kao i mehanizmi deduktivnog manipulisanja ovim reprezentacijama. Od 1958. godine se ništa nije promenilo u ovim jasnim i delotvornim principima organizacije i upotrebe znanja u deduktivnim sistemima VI.

Marvin Minsky je u tom istom periodu vodio grupu studenata koji su se bavili rešavanjem ograničenih problema za čije rešavanje je bila potrebna ljudska inteligencija. James Slagle-ov Saint program (1963.) je mogao da reši probleme nalaženja neodređenih integrala u zatvorenoj simboličkoj formi. Tom Evans-ov program Analogy (1968) rešavao je geometrijske probleme koji se pojavljuju na IQ testovima. Daniel Bobrow-ov program Student (1967.) je rešavao tekstualne algebarske probleme, tipične za niže i srednje razrede osnovne škole.

Paralelno sa klasičnim simboličkim pristupom rešavanju VI problema, počinje i razvoj novog pristupa zasnovanog na konceptu veštačkih neuronskih mreža. Prvobitni modeli su bili zasnovani na McCulloch i Pitts-ovom modelu neurona, Sl.3.3, i Hebb-ovom principu obučavanja. Bernie Widrow je reformulisao principe obučavanja i svoje neuronske strukture nazivao ADALIN-e, dok je Frank Rosenblatt razvijao svoje složenije strukture nazvane Perceptron. Rosenblatt je autor prve teoreme o konvergenciji algoritma obučavanja perceptrona, koja pokazuje da njegov algoritam obučavanja može da prilagodi pojačanje veza perceptrona tako da se na osnovu njih uvek može vršiti diskriminacija koncepata sadržanih u ulaznim podacima, pod uslovom da se ovi koncepti ne preklapaju u prostoru ulaznih opisa.



Sl.3.3 Amerikanci Walter Pitts (1923-1969), logičar, i Warren Sturgis McCulloch (1898-1969), neuropsiholog i kibernetičar, objavili su 1943 rad "A Logical Calculus of Ideas Immanent in Nervous Activity", kojim je predložen prvi matematički model neuronskih mreža. Osnovna jedinica ovog modela, jednostavni formalizovani neuron je i dalje standard u ovoj oblasti. U čast tvorca ovog modela, ovaj jednostavni model neurona se često naziva po njihovim imenima - McCulloch-Pitts neuron.

### 3.2 Shvatanje principijelnih nedostataka i praktičnih dometa VI

Početni uspesi sistema VI, ubrzo su pokazali i principijelne nedostatke. Prva poteškoća je nastala zato što je većina ranih programa sadržala malo ili nimalo znanja o domenu na koji se odnose. Njihov uspeh je poticao od jednostavne sintaksne manipulacije. Tipičan primer je razvoj sistema za automatsko prevodjenje, aktuelizovan šokom koji je u Americi izazvalo lansiranje ruskog satelita Sputnik 1957. godine. Amerikanci su shvatili da su podcenili ruska naučna dostignuća i da o njima vrlo malo znaju. Pojavila se urgentna potreba za pregledom ruske naučne literature, koja je usled jezičke barijere zahtevala dugogodišnji rad. Jedno moguće elegantno rešenje ove teškoće je dizajniranje sistema za automatsko prevodjenje sa ruskog na engleski jezik. Nacionalni istraživački savet je započeo finansiranje obimnog projekta automatskog prevodjenja. U početku se mislilo da jednostavne sintaksine transformacije zasnovane na gramatici ruskog i engleskog jezika uz zamenu reči korišćenjem elektronskog reč-



nika, mogu da budu dovoljne za očuvanje originalnog značenje rečenica u postupku prevodjenja. Međutim pokazalo se da prevod zahteva opšte poznavanje oblasti o kojoj se radi, da bi mogle da se razreše dvosmislenosti i uspostavi pravi kontekst rečenica. Projekat je neuspešno završen izveštajem savetodavnog komiteta da nema mašinskog prevoda opšteg naučnog teksta, i neće ga biti u neposrednoj budućnosti.

Druga vrsta poteškoća je bila eksponencijalna kompleksnost mnogih problema koje je VI pokušavala da reši. Većina ranih VI programa je rešavala date probleme ispitivanjem različitih kombinacije koraka rešavanja do postizanja konačnog rešenja. Ova strategija je u početku radila zato što su igračka-problemi sadržali veoma malo objekata i stoga posedovali malu kombinatornu kompleksnost. Na primer, optimizam koji je pratio razvoj automatskog dokazivanja teorema, je ubrzo splasnulo kada istraživači nisu uspeali da dokažu teoreme koje imaju više od nekoliko desetina činjenica. Činjenica da program može pronaći rešenje u principu, ne znači da ga može i praktično naći. Neuspeh da se savlada „kombinatorna eksplozija“ je bio jedan od glavnih razloga da britanska vlade 1973. godine donese odluku o prekidu finansiranja svih VI istraživanja, osim na dva univerziteta.

Treća poteškoća je nastala zbog fundamentalnih ograničenja osnovnih arhitektura inteligentnih sistema. Ovo je bilo naročito prisutno kod tehnologije neuronskih mreža. U čuvenoj knjizi *Perceptroni* Minsky i Papert-a iz 1969. godine, dokazana je teorema po kojoj jednoslojni perceptron sa dva ulaza i jednim izlazom ne može rešiti jednostavni XOR problem. Iako se ovaj rezultat ne odnosi na višeslojne perceptrone, usled nepostojanja efikasnog algoritma obučavanja višeslojnih perceptrona, kao da je predstavljao jedva očekivani razlog da se prestane sa finansiranjem gotovo svih istraživanja iz domena veštačkih neuronskih mreža. Istorijska je ironija, da su upravo te iste 1969. godine Bryston i Ho formulisali novi algoritam obučavanja višeslojnih neuronskih mreža koristeći princip propagacije greške unazad. Ovaj algoritam je mnogo kasnije ponovo pronadjen i direktno je omogućio novo zlatno doba razvoja tehnologije neuronskih mreža, koje sa manjim padovima traje sve do današnjih dana.

### 3.3 Doba ekspertskih sistema

Sistemi zasnovani na eksplicitno kodovanom znanju, počinju da se pojavljuju prvi put 70-tih i 80-tih godina XX veka. Više je razloga za pomeranje težišta razvoja oblasti VI ka ovoj tehnologiji. Prvi razlog leži u već pomenutoj kombinatorijalnoj eksploziji, koja se pojavljuje u iole složenijim VI sistemima zasnovanim isključivo na principu slepe pretrage u ekvivalentnom prostoru stanja, sa malo ili nimalo eksplicitno ugrađenog znanja o problemskom domenu. Drugi razlog je opšte sazrevanje oblasti VI, čime je omogućeno pomeranje ka realnijim praktičnim problemima. Ovo je vodilo specijalizaciji VI ka podoblastima kao što su ekspertski sistemi, računarska vizija i procesiranje prirodnih jezika. Treći i možda najvažniji razlog je donošenje tzv. Mansfildovog amandmana, koji je nametnuo ograničenje Američkom ministarstvu odbrane (U.S. Department of Defense – DoD) prilikom finansiranja osnovnih istraživanja. Naime ovaj amandman donet 1969. godine zahteva da svako osnovno istraživanje podržano

od DoD, mora imati direktnu vojnu praktičnu primenu. Budući da je i do 1969. godine DoD bio ekskluzivan finansijer VI istraživanja, Mansfieldov amandman je uticao na dodatnu aplikativnu crtu prilikom njihovog formulisanja. Osim togai druge velike kompanije na zapadu su ozbiljnije počele da istražuju komercijalne potencijale VI.



Sl.3.4 Edward Feigenbaum (levo), Joshua Lederberg (1925-2008), genetičar i dobitnik Nobelove nagrade (sredina), and Bruce Buchanan (1940 - ), (filozof i informatičar) (desno) su učesnici i autori kultnog ekspertskog sistema DENDRAL, koji na osnovu hemijske formule i podataka dobijenih iz mas-spektrometra daje strukturnu formulu analiziranog hemijskog jedinjenja.

Program DENDRAL (Buchanan, 1969) je rani primer ovog pristupa. Napravljen je na Stanfordu, gde su se Ed Feigenbaum, bivši student Herbert Simon-a, Bruce Buchanan, filozof i informatičar i Joshua Lederberg, genetičar i dobitnik Nobelove nagrade, udružili da bi rešili problem odredjivanje strukturne molekularne formule datog jedinjenja na osnovu informacija dobijenih iz mas-spektrometra i njegove hemijske formule, Sl.3.4. Liderberg je već razvio program koji može da generiše sve topološki moguće aciklične strukture na osnovu zadate hemijske formule. Uspeh ovog sistema potiče od ideje da se iskoristi znanje iskusnih hemičara prilikom interpretacije mas-spektralnih podataka. Ova znanja su iskorišćena da se ograniče moguće strukture generisane Liderbergovim algoritmom. Danas se pod imenom DENDRAL podrazumeva čitava kolekcija programa razvijana na ovom projektu sve do kraja 70-tih godina prošlog veka. Interesantan je podatak da mnoge od ovih programa hemičari i danas koriste upraksi. Značaj Dendral-a je bio u tome da je to prvi uspešni sistem koji je intenzivno koristio domensko znanja u formi IF-THEN pravila.

Sledeći značajan ekspertski sistem razvijan na Stanfordu je bio iz oblasti medicinske dijagnostike. Feigenbaum, Buchanan i Edward Shortliffe, u to vreme student medicine, Sl.3.5, razvili su sistem MYCIN koji je služio kao konsultantski sistem lekarima iz domena dijagnostike i terapije infekcija krvi. Prvo pitanje koje je razrešavano u okviru ovog projekta se odnosilo na način kodovanja ekspertskog znanja lekara dijagnostičara. Pokazalo se da je formalizam IF-THEN pravila, razvijen u okviru projekta DENDRAL idealan za medicinski domen. Naime, na prirodan način IF deo pravila obuhvata simptome, a THEN deo sadrži uzroke tih simptoma. Ova elementarna znanja su dobijana kroz postupak intervjua sa lekarima iz datog medicinskog domena.

Interesantno je napomenuti da IF-THEN tip rezonovanja ima vrlo dugu istoriju u medicini. U tzv Edwin Smith-ovom papirusu o hiruškim znanjima drevnih egipćana koji datira iz XVII veka pre nove ere, stoji sledeće:

### Slučaj 30

**Naziv:** Uputstva u vezi iščašenja pršljena vrata

**Pregled:** Ako ispituješ čoveka koji ima iščašenje vratnih pršljenova, potrebno je da mu kažeš: "Pogledaj prvo u svoja ramena, a zatim pogledaj svoje grudi". Saznaj da li pri tome oseća bol.

**Dijagnoza:** Ako ima bol, reci mu da ima iščašenje vratnih pršljenova i da ćeš izlečiti njegovu bolest.

**Tretman:** Prvog dana umotaj vrat svežim mesom. Nakon toga, svakog dana do oporavka stavljati med.

Sa oko 450 pravila, MYCIN je davao dijagnoze podjednako dobro kao ekspert, a znatno bolje od mladih doktora. Za razliku od prethodnih sistema, Shortliffe je u sistemu MYCIN uveo heurističku meru sigurnosti pravila, skaliranu na opseg [0,1], koju je nazvao faktorom uverenja.



Sl.3.5 Bruce Buchanan (levo) i Edward Ted Shortliffe (1947 - ), (desno) razvili su ekspertski sistem MYCIN za dijagnostiku infekcija krvi. Sa oko 450 pravila, MYCIN je davao dijagnoze podjednako dobro kao ekspert, a znatno bolje od mladih doktora.

Važnost ovog projekta proističe iz njegove inovativne arhitekture. Naime, sistem za rezonovanje je potpuno odvojen od baze medicinskih znanja, odnosno skupa kodovanih pravila. William van Melle je iskoristio ovu činjenicu i implementirao sistem EMYCIN, gde E potiče od Empty – prazan. U ovom sistemu baza znanja je prazna

i može se uz odgovarajuću interakciju sa ekspertima iz datog problemskog doma-  
na napuniti odgovarajućim pravilima. Ovim su stvoreni uslovi za savremene ljuske  
(shell) ekspertskih sistema, primenljivih na bilo koji domen. Postoji čitav niz razvijenih  
ekspertskih sistema u periodu 80-tih godina prošlog veka, koji su pokrenuli čitavu  
industriju. Ekspertski sistemi su služili za dobijanje odgovora na pitanje gde se nalaze  
najverovatnija nalazišta nafte, pa do odgovora na pitanje kako najbolje sastaviti jedan  
složeni računarski sistem od datih komponenti. VI industrija se razvila od početnih  
nekoliko miliona dolara 1980. do milijarde dolara 1988. godine. Ubrzo posle toga je  
došao period „VI zime“ i propasti mnogih naprednih kompanija iz ovog segmenta.  
Jedan od glavnih uzroka ovog pada je velika razlika između onog što su istraživači  
najavljivali kao moguće i ostvarivo i onoga što je zaista realizovano. Primitimo da je  
ova disproporcija prisutna od samog početka nastanka oblasti VI, pa sve do današnjih  
dana.

### 3.4 Povratak tehnologije neuronskih mreža

Pravi podsticaj za nastavak istraživanja i primene neuronskih mreža u okviru VI je  
nastao sredinom 1980ih kada su najmanje četiri različita tima ponovo otkrila algoritam  
obučavanja višeslojnih neuronskih mreža koristeći princip propagacije greške una-  
zad. Najstarija varijanta ovog temeljnog algoritma obučavanja pripisuje se algoritmu  
Bryson-a i Ho-a iz 1969. godine, Sl.3.6. Algoritam je primenjen na niz praktičnih pro-  
blema, probleme učenja u računarstvu i psihologiji. Uticajna kolekcija rezultata ovog  
novog početka primene neuronskih mreža objavljena je knjizi *Paralelno Distribuirano  
Procesiranje*, Rumelhart-a i McClelland-a iz 1986. godine, Sl.3.7.



Sl.3.6 Arthur Earl Bryson (1925 - ) i Yu-Chi Ho (1934 - ) u knjizi *Applied Optimal Control: Optimization, Estimation, and Control* iz 1969. godine su prvi formulisali algoritam optimiza-  
cije sa prostiranjem greške unazad, koji predstavlja centralni algoritam obučavanja savremenih  
višeslojnih neuronskih mreža.



Sl.3.7 David Rumelhart (1942-2011) i James McClelland (1948 - ) su u uticajnoj knjizi *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* iz 1986. godine, ponovo aktualizovali tehnologiju veštačkih neuronskih mreža.

Ovi takozvani konekcionistički modeli inteligentnih sistema su od mnogih smatrani kao direktna konkurencija dominirajućim simboličkim modelima predloženim od strane Newell-a i Simon-a i logičkom pristupu McCarthy-ja i ostalih. Šta je u osnovi ljudske inteligencije: konekcionizam ili logičko simboličko procesiranje, ostaje otvoreno pitanje. Ono što je racionalno pretpostaviti je da ovo nisu dva odvojena konkurentna pristupa, nego pre svega pristupi koji se međusobno dopunjuju i istovremeno razvijaju, osvetljavajući složene procese u biološkim nervnim sistemima i bolje razumevanje arhitektura, algoritama obučavanja i načina primene veštačkih neuronskih sistema. Napredak u razumevanju pojedinih podsistema mozga, naročito onih vezanih za procesiranje senzorskih signala, direktno utiču na sintezu savremenih efikasnih sistema u domenu računarske vizije, prepoznavanja oblika, sinteze i razumevanja govora, odlučivanja u uslovima velike količine sirovih podataka u uslovima neodređenosti i td. Osim tog, osnovni principi neuroračunarstva, kao što su adaptivnost, distribuiranost i masovni paralelizam, podstaknuti su i podržani konstantnim rastom hardverskih performansi savremene računarske tehnike.

### 3.5 VI u doba Interneta i web-a

Pojava interneta i weba je podsticajno uticala na razvoj VI. Uobičajena praksa u okviru oblasti je bio relativno nezavisan razvoj podoblasti usmerenih na specifične aspekte inteligencije, kao što su reprezentacija znanja, automatsko rezonovanje, planiranje, učenje, rešavanje problema ili razumevanje prirodnih jezika. Pojavom interneta i weba, nastalo je prirodno okruženje za razvoj softverskih agenata, koje možemo posmatrati i kao sisteme koji objedinjavaju neke ili sve aspekte inteligentnog ponašanja. VI sistemi su postali toliko rasprostranjeni u aplikacijama za internet da je sufiks „bot“ ušao u svakodnevni jezik. Štaviše, VI tehnologije se nalaze u mnogim alatima za interet kao što su algoritmi za pretragu, sistem za preporučivanje i sistem za pravljenje web sajtova.

U strateškom smislu naglasak je dat na razvoj autonomnih racionalnih agenata, a kao i višegentskih sistema. Uočena je njihova ključna uloga u daljem razvoju kompleksnih,

distribuiranih i otvorenih sistema, kao što je internet. Ovo je podstaklo stare napore u okviru VI, koje možemo nazvati rešavanje problema “potpunog agenta”, kako su to definisali John Laird, Allen Newell i Paul Rosenbloom u vezi sa projektom SOAR, Sl.3.8.



Sl.3.8 John Laird (1954 - ) i Paul Rosenbloom (1954 - ) su zajedno sa svojim PhD mentorom Allen Newell-om razvijali kognitivnu arhitekturu nazvanu SOAR (acronim od State, Operator And Result), koja se smatra jednim od prvih pokušaja sinteze potpunog agenta, koncepta koji se aktuelizovao pojavom interneta i weba. Arhitektura je kroz neprekidan proces usavršavanja evoluirala od početne SOAR1 iz 1982. godine do SOAR9 iz 2008. god.

Jedna od pozitivnih posledica pokušaja razvoja potpunog agenta je shvatanje neophodnosti redefinisavanja i objedinjavanja prethodno odvojenih podoblasti VI. Globalno je prihvaćen stav da senzorski sistemi kao što su vizija i govor, ne mogu davati potpuno pouzdane informacije narednim slojevima inteligentne arhitekture, tako da npr. podsistemi za rezonovanje i planiranje moraju biti osposobljeni za prihvataje nepouzdanih informacija. Osim toga, širi kontekst agenata obuhvata tradicionalno i teoriju upravljanja, ekonomiju i teoriju igara, dakle oblasti koje su bile udaljene od VI. Poznato je da iz susreta prethodno nepovezanih naučnih disciplina, po pravilu, kao posledicu imamo ubrzani razvoj svake od njih na račun komplementarnih ideja i dostignuća pojedinačnih oblasti. Jedan od najevidentnijih rezultata ove sinergije je napredak u razvoju autonomnog vozila (automobil bez vozača, ili drugačije rečeno automobil sa automatskim vozačem, npr. Google car). Bolja integracija senzorskih informacija, mapiranje, lokalizacija i planiranje je ostvarena na račun rezultata proisteklih iz teorije automatskog upravljanja.

Potpuni agent po definiciji mora da uči na osnovu primera, okruženja u kome se kreće i posmatranja rezultata svojih akcija. Ovo je pojavom interneta, na kome eksponencijalno rastu raspoloživi podaci, dovelo još jednu disciplinu na sam front današnje računarske tehnologije. To je mašinsko učenje. Razvoj teče u dva pravca: nove arhitekture i novi algoritmi obučavanja. U domenu novih arhitektura poseban napredak predstavlja tzv. Deep Learning, odnosno arhitekture obučavajućih struktura, čiji ekvivalentni graf procesiranja ulaznih signala ka izlaznim ima što veću dubinu. Često se ove arhitekture nazivaju i hijerarhijske, a postupak obučavanja se označava



kao učenje reprezentacije. Termin deep learning je nastao nakon utemeljujućeg rada Geoffrey Hinton-a "Learning multiple layers of representation," iz 2007. god., Sl.3.9.

Fenomen pred-obuke u samoobučavajućem režimu dubokih arhitektura je doveo do njihove izuzetne popularnosti, budući da je to izvanredno odgovaralo ogromnoj količini neoznačenih (nekategorisanih) podataka na internetu. Ovaj pristup se naročito pokazao efikasnim za tzv. prirodne senzorske signale, kao što je slika, video i govor. Mnogi spektakularni rezultati kojim nas danas zasipa kompanija Google se upravo baziraju na ovoj tehnologiji.

Konačno, projekat potpuni agent je podstakao stare rasprave šta je važno u današnjem trenutku razvoja VI. Mnogi uticajni autori, kao što su McCarthy, Minsky, Nils Nilsson i Patric Winston, izrazili su opšte nezadovoljstvo načinom na koji se VI danas razvija. Projektovanje VI sistema za partikularne probleme i njihovo potonje usavršavanje, ovi autori smatraju odustajanjem od prave ideje VI, a to su mašina koje misle, uče i kreiraju. Ova nastojanja su nazvali novim pravcem razvoja VI, tzv. Human-Level AI – HLAI, odnosno VI ljudskog nivoa, ili još opštije Artificial General Intelligence - AGI. Prva konferencija na ovu temu organizovana je 2004. godine, a od 2008. godine se održavaju redovne godišnje konferencije u organizaciji AGIS (Artificial General Intelligence Society) i AAAI (Association for the Advancement of Artificial Intelligence). AGI serija konferencija igrala je i igra važnu ulogu u preporodu istraživanja u oblasti VI, u dubljem i originalnijem smislu, podstičući interdisciplinarnost na osnovu različitih tumačenja i razumevanja pojma inteligencije i istraživanja različitih pristupa ovom problemu. Radi sticanja uvida u širinu ovih pristupa, navodimo podoblasti koje su bile zastupljene na poslednjoj AGI konferenciji održanoj u Quebec City-ju u Kanadi od 1. do 4. avgusta 2014. godine:

- ◆ Agent Architectures
- ◆ Autonomy
- ◆ Benchmarks and Evaluation
- ◆ Cognitive Modeling
- ◆ Collaborative Intelligence
- ◆ Creativity
- ◆ Distributed AI
- ◆ Formal Models of General Intelligence
- ◆ Implications of AGI for Society, Economy and Ecology
- ◆ Integration of Different Capabilities



Sl.3.9 Geoffrey Hinton (1947 - ) je prvi predložio efikasan algoritam obučavanja dubokih arhitektura, pod nazivom ograničena Bolcmanova mašina sa samoobučavanjem kao pred-obukom.



- ♦ Knowledge Representation for General Intelligence
- ♦ Languages, Specification Approaches and Toolkits
- ♦ Learning, and Learning Theory
- ♦ Motivation, Emotion and Affect
- ♦ Multi-Agent Interaction
- ♦ Natural Language Understanding
- ♦ Neural-Symbolic Processing
- ♦ Perception and Perceptual Modeling
- ♦ Philosophy of AGI
- ♦ Reasoning, Inference and Planning
- ♦ Reinforcement Learning
- ♦ Robotic and Virtual Embodiment
- ♦ Simulation and Emergent Behavior
- ♦ Solomonoff Induction.

### 3.6 Neki najuspešniji VI sistemi

Budući da se VI razvija u okviru mnogih podoblasti, nije lako odgovoriti na pitanje šta VI danas može da ponudi kao praktična rešenja i šta su njeni najveći dometi. U ilustrativne svrhe, navodimo neke od ostvarenih praktičnih rezultata VI u proteklom periodu.

**Autonomno planiranje i izvršavanje:** Robotski sistem „Remote Agent - RA“ je instaliran u NASA kosmički brod Deep Space 1- DS1, lansran oktobra 1998. godine u cilju evaluacije 12 naprednih kosmičkih tehnologija, Sl. 3.10. To je u stvari inteligentni agent koji služi kao interfejs između operatora na zemlji i senzora i efektora na DS1.

**Igranje šaha:** IBM-ov Deep Blue je postao prvi računarski program koji je pobedio svetskog šampiona u šahu, Gari Kasparova, sa rezultatom 3.5 : 2.5, 1997. godine Kasparov je rekao da je osetio „novu vrstu inteligencije“ preko puta table, Sl. 3.11.



Sl.3.10 Deep Space 1, NASA letilica, lansirana u oktobru 1998. godine nosi u sebi VI sistem Remote Agent, za automatsko planiranje i izvršavanje različitih operacija na kosmičkom brodu.



Sl.3.11 Gari Kasparov igra šah protiv IBM-ovog računara Deep Blue 1997. godine. Rezultat: 3.5:2.5 za Deep Blue. IBM-ov računar se sastojao od specijalizovanog hardvera i softvera koji se izvršavao na superračunaru IBM RS/6000 SP2.

**Igra Dame:** Profesor Jonathan Schaeffer sa svojim timom sa univerziteta Alberta u Edmontonu, Kanada, je 2007. godine objavio rad „Checkers is Solved“ u kome iznosi stav da perfektna igra oba igrača, bez obzira koje su boje, vodi nerešenom rezultatu, Sl. 3.12. Dakle, sistem za igranje Dame pod nazivom CHINOOK koji je ovaj tim je razvijao još od 1989. godine, igra najmanje nerešeno protiv bilo kog igrača, bez obzira da li igra sa crnim ili belim figurama. Ako imamo u vidu da igra Fama poseduje kompleksnost od 500,995,484,682,338,672,639 različitih pozicija, ovaj rezultat se može smatrati jednim od najvećih dostignuća VI do sada.



Sl.3.12 Jonathan Schaeffer (1957 - ) sa svojim timom sa univerziteta Alberta u Edmontonu, Kanada, je 2007. godine objavio rad „Checkers is Solved“. Sistem za igranje Dame pod nazivom CHINOOK ovaj tim je razvijao još od 1989. godine.

**Autonomno upravljanje vozilima:** Sistem ALVYNN (akronim od Autonomous Land Vehicle in a Neural Network) razvio je Ph.D. student Dean Pomerleau sa Carnegie Mellon University, USA, Sl.3.13. ALVYNN je ugradjen u CMU-ov NavLab računarski kontrolisan kombi i korišćen je za navigaciju kroz Sjedinjene Američke Države. Na putu dugom 2850 milja sistem je uspešno upravljao vozilom 98% vremena vožnje. U preostala 2% vremena vožnje, upravljanje je preuzimao čovek, uglavnom na izlaznim petljama autoputeva.



Sl.3.13 ALVINN - Autonomous Land Vehicle in a Neural Network, sistem za automatsko upravljanje vozilom, ugradjen u NavLab kombi sa Carnegie Mellon University, USA, gde je i razvijen. Ulaz u sistem je 30 x 32 matrica - crno bela slika dobijena sa kamere ugradjene na vrhu vozila. Ovih 960 ulaza se uvode u jednoslojnu neuronsku mrežu sa 30 izlaznih neurona koji koduju korespondentne zaokrete volana u toku vožnje.

**Medicinska dijagnostika i terapija:** Medicinski dijagnostički programi bazirani na verovatnosnim modelima dostigli su zavidnu tačnost u mnogim oblastima medicine. OpenClinical održava niz web sajtova na kojima se mogu naći podaci o medicinskim ekspertskim sistemima u praktičnoj upotrebi. Navedimo neke od njih: Athena DSS za dijagnostiku i lečenje povišenog krvnog pritiska, Gideon, za infektivne bolesti, Iliad za internu medicinu, TherapyEdge HIV za sprovođenje terapije obolelih od HIV, i niz drugih.

**Inteligentno rasporedjivanje:** Inteligentni softver za rasporedjivanje predstavlja značajnu primenu VI za industrijske potrebe. Navedimo nekoliko karakterističnih primera. Sistem AURORA kompanije Scottler Henke Associates Inc., koristi Boing kao pomoć prilikom sklapanja modela Boeing Dreamliner. Sistem TEMPORIS razvila je kompanija United Space Alliance, LLC, za pomoć kosmičkim posadama prilikom planiranja i upravljanja budućim kosmičkim letovima. TEMPORIS pomaže posadi da rasporedi gotovo sve dnevne aktivnosti, uključujući rutinske dnevne aktivnosti, održavanje kosmičkog broda u normalnom stanju, koa i izvodjenje planiranih naučnih eksperimenata.

**Robotika:** Mnogi hirurzi već koriste robotske pomoćnike u mikrohirurgiji. HipNav je sistem razvijen na Carnegie Mellon University, koji koristi računarsku viziju u cilju

formiranja trodimenzionalnog model pacijentove unutrašnje anatomije, a zatim asistira hirurgu robotsko upravljanu ugradnju veštačkog kuka, Sl.3.14.

**Automatsko preporučivanje:** Kada se logujete na Amazon.com više niste iznenađeni što vam ova kompanija lično preporučuje neke proizvode i sadržaje, za koje sa iznenađenjem konstatujete da su u dobroj meri u saglasnosti sa vašim preferencijama i ukusom. Jedan od tipičnih sistema za preporučivanje, zasnovan na tehnici mašinskog učenja, naziva se društveno ili kolaborativno filtriranje. Za svakog korisnika se vodi lista njegovih preferencija na osnovu prošlih aktivnosti na internetu i već obavljenih kupovina. Ukoliko su preferencijali korisnika B dovoljno korelisani sa preferencijalima korisnika A, tada sistem za kolaborativno filtriranje preporučuje korisniku A neke od kupovina korisnika B, koje još uvek nisu kupljene od strane korisnika A. Ovu tehniku koriste i mnoge druge kompanije, kao što su iTunes, TiVo, Netflix i sl.

**Računarske video igre:** VI počinje da igra značajnu ulogu u dizajniranju računarskih video igara u kojima čovek interaguje sa virtuelnim karakteima u virtuelnom svetu igre. Iako je naglasak u igrama na realističnoj i bogatoj grafici, primena VI tehnika čine ovaj virtuelni svet bogatijim i zanimljivijim. Npr. u uobičajenoj postavci igara igrač interaguje sa veštačkim agentima igre, koje trebaju smeravati od tačke A do tačke B sa ciljem da se izbegnu prepreke ili susrete sa drugim agentima, što se može ostvariti primenom, recimo A\* algoritama pretrage razvijenim u okviru VI. Igra *Black and White 2*, koju je razvio Lionhead Studios a distribuira Electronic Arts, koristi kombinaciju neuronskih mreža i stabala odlučivanja u cilju učenja optimalnih strategija za datu igru. Igra *F.E.A.R* (*First Encounter Assault Recon*) Jeff-a Orkina-a kompanije Monolith Productions koristi A\* algoritam za planiranje sekvence akcija agenata. Obe ove igre zauzimaju 1. i 2. mesto na listi prvih 10 najuticajnijih video igara koje koriste tehnike VI.



Sl.3.14 HipNav je robotski sistem koji asistira hirurzima prilikom ugradnje veštačkih kukova. Razvijen je na Carnegie Mellon University, USA.

### Rezime 3. poglavlja

- ♦ Formalni početak oblasti VI vezan je za *Summer Research Project on Artificial Intelligence at Dartmouth College*, održan 1956. godine, koji je okupila uglavnom mlade i ambiciozne Šenonove bivše studente.
- ♦ U toku svog gotovo 60 godišnjeg razvoja, VI je prošla niz faza, od entuzijazma praćenog nerealnim očekivanjima, pa do razočaranja i potpunog prekida finansiranja čitavih podoblasti.
- ♦ Razvoj VI pokazuje evidentnu prisutnost dva dominantna pristupa: simbolički i konekcionistački, prvi podržan idejom da je u osnovi inteligencije manipulacija strukturama simbola i drugi, koji se oslanja na modele inspirisane organizacijom i načinom rada ljudskog mozga.
- ♦ Pojava interneta i veba doprinela je naglom razvoju pravca VI zasnovanog na racionalnim agentima implementiranim kao širok spektar softboot-ova, koji danas dominiraju internetom obavljajući najrazličitije poslove: od pretraga relevantnih informacija do obavljanja kupovina, berzanskih transakcija, profilisanja korisnika za potrebe marketinga i sl.
- ♦ Usled izuzetne težine zadatka koji se postavlja pred oblast VI, ona se dugo godina razvijala kroz odvojene podoblasti, čime je izgubljena početna vizija pionira ove oblasti da je glavni cilj VI pravljenje mašina koje misle. Ovo je u poslednje vreme dovelo do novog pokreta koji je dobio naziv AGI (Artificial General Intelligence), a čiji je osnovni cilj upravo sinteza veštačkih sistema koji zaista misle na ljudski način.

### Pitanja i zadaci

1. Kako bi ste opisali GPS sistem Newell-a i Simona-a?
2. Zašto je GPS imao ograničene domete?
3. Defonišite ekspertne sisteme.
4. Šta su zajedničke karakteristike ranih ekspertskih sistema DENDRAL i MYCIN?
5. Šta su ograničenja ekspertskih sistema?
6. Koje su razlike između ekspertskih sistema i veštačkih neuronskih mreža?
7. Zašto je oblast neuronskih mreža ponovo aktuelizovana 80-tih godina prošlog veka?
8. Šta je to SOAR arhitektura?
9. Šta obuhvata koncept potpunog agenta?
10. Navedite neke primere primene VI na internetu.
11. Šta znače akronimi HLAI i AGI?
12. Zašto je AGI danas ponovo aktuelan?

## 4. Arhitekture sistema veštačke inteligencije

U ovom poglavlju izložit ćemo osnovne arhitekture pogodne za realizaciju sistema-VI. Neke od najznačajnijih su Pandemonium (Oliver Selfridge, 1959), produkcionni sistemi (Emil L. Post, 1943., Sl. 4.1) troslojna arhitektura, arhitektura školske table (Blackboard architectures, Lee D. Erman, i ostali, 1980), BDI architectures (Belief-Desire-Intention, Michael E. Bratman, 1987), SOAR (John Laird, Paul Rosenbloom, Allen Newell, SOAR - acronim od "State, Operator And Result"), ACT-R (John Robert Anderson, ACT-R je akronim od "Adaptive Control of Thought—Rational"), kao i čitava klasa arhitektura kojima se pokušava modelovati korteks.

Između svih pomenutih pristupa posebno se po opštosti i značaju izdvaja produkcionni sistem, koji je prvi put predložio još 1943. godine američki matematičar i logičar Emil Leon Post, ali ne u kontekstu moguće arhitekture VI sistema, već kao opšti model računanja. Pokazuje se da je ovaj sistem jednake snage kao i Turingova mašina. Prvi su Newell i Simon primenili produkcijske sisteme za modelovanje ljudskog mišljenja prilikom razvoja GPS sistema. Pokazalo se da je ovaj tip sistema vrlo pogodan za implementaciju pretraga koje su neizbežni centralni deo VI sistema. Stoga je ovaj računarski formalizam primenjivan u mnogim ekspertskim sistemima, kao i u sistemima koji su predstavljali osnovu za izučavanje fenomena učenja kako kod prirodnih tako i kod veštačkih sistema, npr. SOAR (Newell, 1990) i ACT\* (Anderson, 1983). Važna klasa jezika VI visokog nivoa, kao što je OPS (Official Production System), su direktno razvijeni na osnovu načina rada produkcijskih sistema. Osim toga, produkcionni sistemi su implementirani i u drugim, opštim jezicima, kao npr. CLIPS razvijen u jeziku C, a koji predstavlja objektno orijentisanu verziju produkcionnog sistema u širokoj upotrebi u NASA projektima. JESS je produkcionni sistem koga je razvila kompanija Sandia National Laboratories implementiran u jeziku Java (sajt: <http://www.jessrules.com/jess/download.shtml>).



Sl.4.1 Emil Leon Post (1897-1954), američki matematičar i logičar. 1936. godine razvio je nezavisno od Turinga matematičku teoriju računanja, koja je u svemu bila ekvivalentna Turingovoj. 1943. god. u radu Formal reductions of the general combinatorial problem. American Journal of Mathematics, 65:197-268., uveo je pojam produkcijskih sistema, koji je bio od esencijalnog značaja za razvoj VI.



## 4.1 POJAM PRODUKCIONIH SISTEMA

Produkcioni sistem predstavlja računarski formalizam, koji sadrži jasnu podelu na komponente:

- ♦ globalna baza podataka (GB)
- ♦ skup pravila produkcije
- ♦ strategija upravljanja.

**Globalna baza podataka** je centralna struktura podataka koju koristi sistem produkcije.

**Pravila produkcije** se primenjuju na GB i menjaju njeno stanje. Svako pravilo ima sledeći strukturni oblik:

**Pravilo: (Uslov primene, Akcija).**

Uslov primene je stanje koje zadovoljava GB da bi zadato pravilo bilo primenljivo. Akcija opisuje način na koji će primenjeno pravilo menjati stanje GB.

**Strategija upravljanja** određuje koje od primenljivih pravila se aktivira u svakom koraku rada produkcionog sistema i proverava da li GB zadovoljava terminalni uslov, nakon čijeg dostizanja zaustavlja rad sistema.

Razlike između produkcionih sistema i klasičnih programskih sistema sa hijerarhijskom strukturom se sastoji u sledećem.

- ♦ GB je dostupna svim pravilima produkcije
- ♦ Jedno pravilo produkcije ne indukuje neko drugo pravilo. Veza između pravila se ostvaruje samo preko GB.
- ♦ Sistem je u najvećem stepenu modularan, tako da se izmene u GB, produkcionim pravilima i sistemu upravljanja može vršiti relativno nezavisno.

Na Sl.4.2 prikazan je pseudo kod generičke procedure PRODUCTION, koja opisuje suštinu rada produkcionog sistema. Struktura DATA sadrži trenutno stanje globalne baze. U koraku 4. naredba **select** implementira strategiju upravljanja produkcionim sistemom. Produkcioni sistem završava rad kada GB prelazi u terminalno stanje.

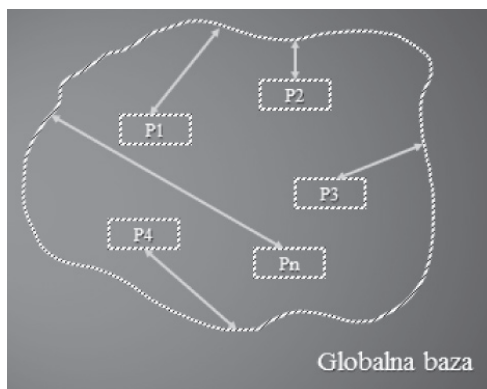
### Procedura PRODUCTION

1. DATA ← polazna GB
2. **Until** DATA zadovoljava terminalni uslov **Do:**
3. **begin**
4. **select** pravilo P iz skupa primenljivih pravila
5. DATA ← rezultat primene P na DATA
6. **end**

Sl.4.2 Pseudo kod generičkog produkcionog sistema. Struktura DATA sadrži trenutno stanje globalne baze. U koraku 4. naredba **select** implementira strategiju upravljanja produkcionim sistemom. Produkcioni sistem završava rad kada GB prelazi u terminalno stanje.

Na Sl.4.3 dat je šematski prikaz prirode interakcije pravila i GB. Pravila su međusobno nezavisna i nikada ne interaguju međusobno. Ovo svojstvo modularnosti predstavlja jaku stranu produkcionog sistema. U fazi razvoja, dodavanje ili ukidanje pravila ne remeti strukturu celokupnog sistema i ubrzava razvoj. Potsetimo se da u klasičnom računarskom formalizmu sa fiksom hijerarhijom glavni program – potprogrami, izmena u jednom modulu se mora pažljivo ispratiti u celokupnom sistemu. Nezavisnost strategije upravljanja od pravila produkcije i GB, daje narednu prednost produkcionom sistemu. Izmene u strategiji upravljanja su nezavisne od ostalih delova sistema.

Produkcioni sistem radi u ciklusima. Novi ciklus započinje poredjenjem preduslova svih pravila sa trenutnim stanjem GB. Sva pravila koja zadovoljavaju preduslov, formiraju tzv. **konfliktni skup**. U fazi **razrešavanja konfliktnog skupa** na osnovu primenjene strategije upravljanja bira se samo jedno aktivno pravilo, koje svojom akcijom prevodi GB u novo stanje. Ukoliko nije zadovoljen terminalni uslov, započinje novi ciklus rada produkcionog sistema, Sl.4.4.



Sl.4.3 Šematski prikaz interakcije pravila i GB u produkcionom sistemu. Pravila ne interaguju međusobno, već samo posredno preko GB.



Sl.4.4 Rad produkcionog sistema se obavlja u ciklusima. U bloku za poredjenje odredjuje se skup aktivnih pravila, odnosno pravila čiji preduslov zadovoljava trenutno stanje GB. Skup aktivnih pravila formira tzv. konfliktni skup. U bloku za razrešenje konfliktnog skupa, na osnovu primenjene strategije upravljanja bira se samo jedno pravilo, čija akcija izaziva prelazak GB u novo stanje, čime započinje novi ciklus rada.

## PRIMER 4.1

Ilustrujemo rad produkcionog sistema na primeru nalaženja najvećg zajedničkog delitelja. Poznato je da klasični Euklidov algoritam, čiji je pseudo kod dat na Sl.4.5 rešava ovaj problem.

### Euklidov klasičan algoritam

```
1. UNTIL A i B nisu jednaki
2. BEGIN
3.   IF A > B,  $A \leftarrow A - B$ 
4.   ELSE
5.      $B \leftarrow B - A$ 
6. END
7.  $D \leftarrow A$  (ILI B)
```

Sl.4.5 Pseudo kod Euklidovog algoritma za određivanje najmanjeg zajedničkog delitelja brojeva A i B. Algoritam se jednostavno uopštava na veći broj zadatih brojeva.

Da bi smo ovaj problem rešili u okviru formalizma produkcionih sistema, potrebno je definisati GB, sup pravila, početno stanje  $GB_0$ , terminalno stanje  $GB_t$  i odgovarajuću strategiju upravljanja.

- ♦  $GB_0$  – zadati skup brojeva
- ♦  $GB_t$  – skup u kome su svi brojevi jednaki
- ♦  $GB$  – zadati skup brojeva koji prolazi sve zahtevane transformacije u tranziciji od  $GB_0 \rightarrow GB_t$
- ♦ **Pravila:** P1 i P2.

**P1: preduslov** – u GB postoje dva različita broja A i B

**akcija:**  $A \leftarrow A - B$ , ako je  $A > B$ , ili  
 $B \leftarrow B - A$ , ako je  $B > A$

**P2: preduslov** – u GB postoje dva broja A i B

**akcija:**  $D \leftarrow A$  ili B; STOP  
D je konačni rezultat

Sl.4.6 Pravila P1 i P2 produkcionog sistema iz Primera 4.1.

- ♦ **Strategija upravljanja:** Pravilo P1 ima uvek prednost nad pravilom P2.

U Tabeli 4.1 data je ilustracija rada definisanog produkcionog sistema prilikom nalaženja najmanjeg zajedničkog delitelja za brojeve (20,10,15,5). Primetimo da sistem radi pet ciklusa i da strategija upravljanja u prva četiri ciklusa bira pravilo P1, dok u poslednjem petom ciklusu bira pravilo P2, koje po definiciji daje rezultat i zaustavlja rad produkcionog sistema.

|        | A   | B  | C  | D | primenjeno pravilo |
|--------|-----|----|----|---|--------------------|
| $GB_0$ | 20  | 10 | 15 | 5 |                    |
| $GB_1$ | 10  | 10 | 15 | 5 | $P1(A,B)$          |
| $GB_2$ | 10  | 10 | 5  | 5 | $P1(B,C)$          |
| $GB_3$ | 5   | 10 | 5  | 5 | $P1(A,C)$          |
| $GB_4$ | 5   | 5  | 5  | 5 | $P1(A,B)$          |
| $GB_t$ | D=5 |    |    |   | $P2(A,B)$          |

Tabela 4.1 Prikaz rada produkcionog sistema u slučaju nalaženja najvećeg zajedničkog delitelja za brojeve (20,10,15,5). Produkciono sistem završava rad u pet ciklusa, dajući rezultat D=5.

Prevodjenje jednog zadatog problema na opis pomoću produkcionog sistema se naziva rešavanje problema predstavljanja. Ovaj problem nema jednoznačno rešenje za dati zadatak, i u velikoj meri određuje efikasnost produkcionog sistema. Ilustrujemo ovo na sledećem primeru.

#### PRIMER 4.2

Slagalica sa klizećim pločicama se sastojio od table dimenzija 3x3 sa osam numerisanih i jednim praznim poljem (PP). Polja se mogu pomerati samo na mesto praznog polja. Cilj igre je prevodjenje početnog stanja u ciljno stanje. Na Sl.4.7 dat je jedan primer početne i ciljne konfiguracije. Ova igra se može prikazati kao produkciono sistem sa sledećim elementima:

- ♦ **GB** – matrica 3x3 sa odgovarajućim stanjem na tabli.
- ♦  **$GB_0$**  – matrica početnog stanja
- ♦  **$GB_t$**  – matrica ciljnog stanja

#### Pravila: Varijanta A

Mogli bi smo posmatrati pločicu sa jednom oznakom, recimo 5, i eksplicitno definisati četiri moguća pomeranja (gore, dole, levo, desno) za svaku od 9 pozicija na tabli 3x3. Ovo bi vodilo ka 72 pravila, pri čemu bi svako od njih imalo odgovarajuće uslove primene, npr. pločica sa oznakom 5 na poziciji (1,1) pomeri se dole, ako je pozicija (2,1) prazna.

#### Pravila: Varijanta B

Ubrzo uočavamo da je pomeranje bilo koje pločice uvek ekvivalentno odgovarajućem pomeranju prazne pozicije (PP). Stoga možemo definisati samo 4 jednostavna pravila:

**P1: preduslov** - PP se ne nalazi u 1. koloni  
akcija - pomeri PP levo

**P2: preduslov** - PP se ne nalazi u 3. koloni  
akcija - pomeri PP desno

**P3: preduslov** - PP se ne nalazi u 1. redu  
akcija - pomeri PP gore

**P4: preduslov** - PP se ne nalazi u 3. redu  
akcija - pomeri PP dole

- ♦ **Strategija upravljanja:** mogu biti primenjene različite klase upravljanja, odnosno pretrage, o čemu će biti više reči u narednom poglavlju.

|   |   |   |
|---|---|---|
| 7 | 2 | 4 |
| 5 |   | 6 |
| 8 | 3 | 1 |

Startno stanje

|   |   |   |
|---|---|---|
|   | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

Ciljno stanje

Sl.4.7 Slagalica sa klizećim pločicama sastoji se od table dimenzija 3x3 sa osam numerisanih i jednim praznim poljem. Cilj igre je prevodjenje početnog stanja u ciljno stanje.

U praksi susrećemo više vrsta produkcionih sistema, čija svojstva su u vezi sa tipovima problema kojima najviše odgovaraju. Navešćemo vrste nastale u odnosu na dimenziju komutativnosti.

**Monotoni produkциони sistemi** su oni u kojima primena nekog pravila ne sprečava kasniju primenu nekog drugog pravila, koje je moglo biti primenjeno u momentu primene prvog pravila.

**Parcijalno komutativni sistemi** poseduju svojstvo invarijantnosti u odnosu na permutovanje sekvence pravila. Naime ako jedna sekvenca pravila prevodi GB iz stanja X u stanje Y, tada to važi i za svaku permutaciju ovih pravila.

**Komutativni produkциони sistemi** su oni koji poseduju i svojstvo monotonosti i parcijalne komutativnosti.

Za jedan sistem je od značaja da poseduje svojstvo komutativnosti, budući da u tom slučaju strategije upravljanja mogu biti najjednostavnije.

## 4.2 ODNOS IZMEDJU VRSTE PRODUKCIONIH SISTEMA I TIPOVA PROBLEMA

Tipovi problema:

- ♦ **Ignorantni** – pojedini koraci u rešavanju se mogu u potpunosti ignorisati (dokazivanje teorema)
- ♦ **Povratni** – u kojima se koraci rešavanja mogu opovrgnuti (slagalice 3x3)
- ♦ **Nepovratni** – u kojima se koraci rešavanja ne mogu opovrgnuti (šah)

|                          | monoton             | nemonoton     |
|--------------------------|---------------------|---------------|
| parcijalno komutativan   | dokazivanje teorema | slagalice 3x3 |
| parcijalno nekomutativan | hemijska sinteza    | šah           |

Tabela 4.2 Primeri problema u odnosu na dimenzije monotonosti i komutativnosti produkci-  
onih sistema

**Ignorantni problemi** mogu biti rešavani jednostavnim kontrolnim strategijama, koje ne moraju imati mehanizam povratka na prethodne korake rešavanja problema.

**Povratni problemi** zahtevaju nešto kompleksnije strategije, koje moraju sadržati mehanizam povratka na prethodne korake rešavanja u cilju eliminacije učinjenih grešaka.

**Nepovratni problemi** zahtevaju najkompleksnije strategije, budući da je svaki korak u rešavanju problema konačan i neispravljiv.

## 4.3 VRSTE PRODUKCIONIH SISTEMA U ODNOSU NA SMER PRETRAGE

**Direktni sistemi:** ako se mogu izdvojiti stanja i ciljevi. GB opisuje stanja. Pravila se primenjuju na opise stanja i generišu nova stanja. To su tzv.  $\Pi$  – pravila produkcije.

**Reverzni sistemi:** ako se opisi ciljeva koriste kao GB. Tada se pravila primenjuju na opise ciljeva i generišu podciljeve. To su tzv.  $O$  –pravila.

**Dvostrani sistemi:** GB obuhvata i opise stanja i ciljeva zadatka. Na svakom koraku se primenjuju i  $O$  i  $\Pi$  pravila.

**Razloživi produkcionni sistemi** su oni kod kojih se GB i terminalni uslovi mogu razložiti na komponente koje dozvoljavaju nezavisnu primenu pravila.

U ovim sistemima je neophodno izraziti globalni terminalni uslov kroz parcijalne terminalne uslove komponenti. Za praksu je najvažniji slučaj kada se ovaj uslov može izraziti u vidu konjukcije terminalnih uslova komponenti.



#### Procedura SPLIT

1. DATA  $\leftarrow$  početna GB
2.  $\{Di\} \leftarrow$  dekompozicija GB . Svako  $\{Di\}$  se razmatra kao posebna GB
3. UNTIL svi  $\{Di\}$  zadovoljavaju terminalni uslov DO:
4. BEGIN
5. SELECT  $D^*$  iz  $\{Di\}$ , koje ne zadovoljava terminalni uslov
6. udaljiti  $D^*$  iz  $\{Di\}$
7. SELECT neko pravilo P iz skupa pravila koje se može primeniti na  $D^*$
8.  $D \leftarrow$  rezultat primene P na  $D^*$
9.  $\{di\} \leftarrow$  dekompozicija D
10. dodati  $\{di\}$  na  $\{Di\}$
11. END

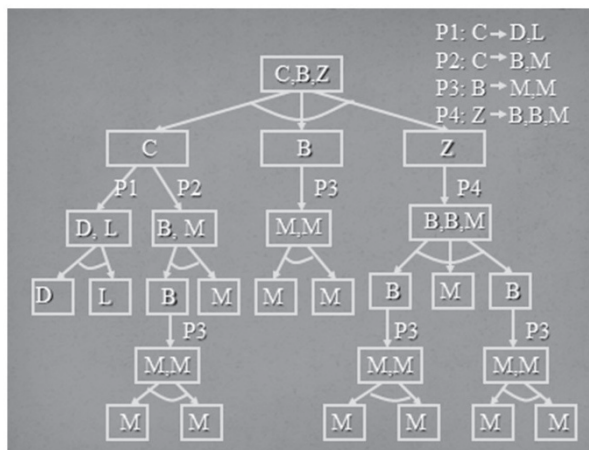
Sl.4.8 Opšta procedura razloživog produkcionog sistema.

Za opis rada razloživih sistema produkcije koriste se i-ili grafovi sa sledećim značenjem

- ♦ čvorovi su globalne baze i njihove komponente mogu biti i-ili tipa,
- ♦ čvorovi koji odgovaraju komponentama GB su i tipa jer sve moraju biti obrađene,
- ♦ čvorovi na koje se primenjuju pravila daju nove GB, koje su ili tipa,
- ♦ graf rešenja u i-ili grafovima je onaj čiji su krajnji čvorovi terminalni uslovi.

Na Sl.4.9 dat je jedan primer razloživog produkcionog sistema sa 4 pravila (P1, P2, P3, P4),  $GB_0 = (C, B, Z)$  i  $GB_t = M$ .

Interesantno je napomenuti da i-ili grafovske strukture imaju veliku primenu u modelovanju igara sa dva igrača u sistemima VI. Sa stanovišta prvog igrača koji je na potezu, njegovi mogući potezi su povezani logičkom operacijom ili, budući da je izbor konkretnog pravila u njrgovim rukama. Ovoj situaciji odgovaraju ili čvorovi. Kada prvi igrač razmatra moguće poteze protivnika (drugog igrača), njegovi mogući potezi su povezani logičkom operacijom i, budući da svaki od tih poteza mora biti uzet u razmatranje, jer nije pod njegovom kontrolom. Tako se redosled poteza prvog i drugog igrača na efikasan način mogu prikazati slojevitom grafovskom strukturom tipa stabla u kojoj se naizmenično smenjuju i i ili čvorovi.



Sl.4.9 Primer rada jednog razloživog produkcionog sistema

## 4.4 ARHITEKTURE INTELIGENTNIH AGENATA

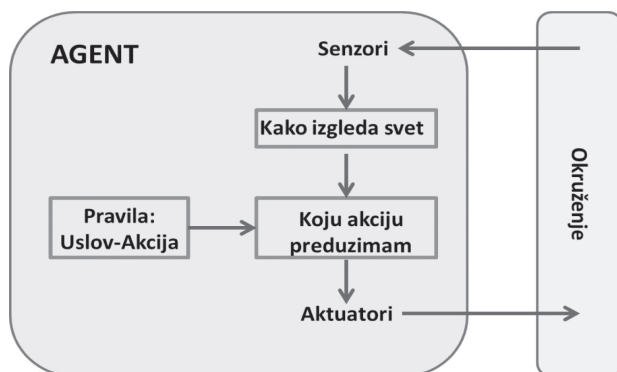
Pored produkcionih sistema, sledeća najopštija klasa arhitektura VI, koja je u poslednje vreme posebno aktuelizovana su inteligentni agenti. **Agent** je nešto što se može posmatrati kao percepcija svog **okruženja** na osnovu **senzora** i delovanje na srednu pomoću **aktuatora**. Ljudski agent ima oči, uši i ostale organe koji su senzori i ruke, noge, usta, i druge delove tela koji su aktuatori. Robotski agent može imati kamere i senzore u infracrvenom opsegu i razne motore koji su aktuatori. Softverski agent kao senzorski ulaz dobija ulaz sa tastature, sadržaje datoteka ili mrežne pakete i deluje na okolinu prikazom na ekranu, pisanjem datoteka ili slanjem mrežnih paketa. Uzećemo kao opštu pretpostavku da svaki agent zna za svoje akcije ali ne uvek i za efekte koje proizvodi na okruženje. Koristićemo termin opažaj koji će se odnositi na perceptivne ulaze agenta u svakom trenutku. Opažajna sekvenca agenata je kompletna istorija svega što je agent ikada percepirao. U principu, agentov izbor akcije u svakom trenutku može zavisiti od cele opažajne sekvence kojom raspolaže do tekućeg trenutka. Ako možemo odrediti izbor agentove akcije za svaku moguću opažajnu sekvencu, onda smo kompletirali pojam agenta. Ponašanje agenta se formalno može opisati **agentskom funkcijom** koja preslikava bilo koju opažajnu sekvencu u akciju.

U ostatku ovog odeljka, daćemo pregled četiri osnovne vrste agentskih arhitektura koje opisuju skoro sve savremene inteligentne sisteme zasnovane na ovom principu:

- ♦ Jednostavni refleksni agenti
- ♦ Refleksni agenti zasnovani na modelu
- ♦ Agenti zasnovani na cilju
- ♦ Agenti zasnovani na korisnosti
- ♦ Agenti koji uče.

### 4.4.1 Jednostavni refleksni agenti

Najjednostavnija vrsta agenta je jednostavni refleksni agent. Ovi agenti biraju akcije na osnovu trenutnog opažaja, ignorišući ostatak opažajne istorije.



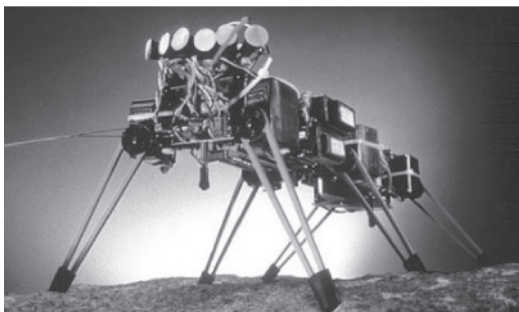
Sl. 4.10 Šematski dijagram jednostavnog refleksnog agenta

**funkcija** Jednostavni\_Refleksni\_Agent (percepcija) **returns** akcija  
**statični:** pravila, skup pravila tipa uslov-akcija  
 stanje←Interpretiraj\_ulaz(percepcija)  
 pravilo←Pravilo\_slaganje(stanje, pravila)  
 akcija←Pravilo\_Akcija[pravilo]  
**return** akcija

Sl.4.11 Jednostavni refleksni agent. On deluje na osnovu selektovanog pravila čiji se uslov delovanja poklapa sa trenutnim stanjem okruženja, koje je definisano percepcijom (okruženja). Funkcija Interpretiraj\_ulaz daje apstraktan opis trenutnog stanja okruženja. Funkcija Pravilo\_slaganje daje prvo pravilo iz aktuelnog konfliktnog skupa.

Jednostavni refleksni agenti su veoma ograničene inteligencije. Agent na slici 4.11 će raditi dobro samo ako ispravna odluka može biti doneta na osnovu trenutnog opažaja - to jest, samo ako je okruženje u potpunosti observabilno.

Refleksni agenti su se prvi put pojavili u radovima Skinnera 1953. godine, kao osnovni model biheviorističkog pristupa u psihologiji. Podsetimo se da bihevioristički pristup objašnjava psihološke fenomene na osnovu modela ulaz-izlaz, odnosno pobuda-odziv. Pomeranje od biheviorizma ka funkcionalizmu u psihologiji imalo je za posledicu uvođenje unutrašnjeg stanja agenta, što vodi ka složenijim arhitekturama koje ćemo izložiti u narednim odeljcima. Iako dominira stav da jednostavni refleksni agenti ne mogu rešavati složenije probleme u VI, radovi Rodneya Brooksa u domenu robotike, kao da ponovo aktuelizuju ovu najjednostavniju arhitekturu, Sl.4.12.



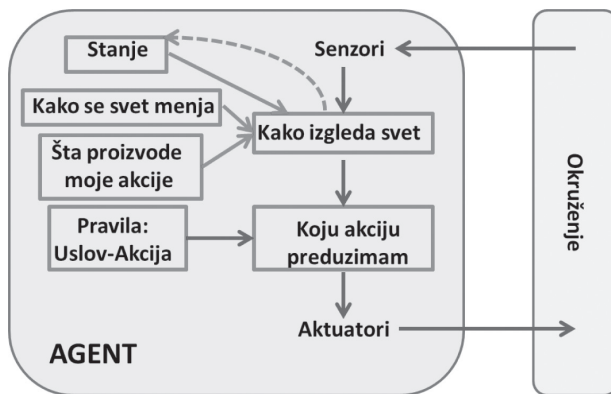
Sl.4.12 Rodney Brooks, australijski robotičar (1954 - ) je ponovo aktuelizovao arhitekture jednostavnih refleksnih agenata. Njegov šestonogi autonomni robot koji hoda (desno) izvršio je značajan uticaj na razvoj savremene robotike i novih složenijih tzv. arhitektura obuhvatanja koji u matematičkom smislu nisu ništa drugo nego prošireni konačni automati.

#### 4.4.2 Refleksni agenti zasnovani na modelu

Najefikasniji način snalaženja u delimično opservabilnom okruženju je osposobljavanje agenta da prati deo okruženja koji ne može da observira. Agent bi trebalo

da održi neku vrstu unutrašnjeg stanja koje zavisi od opažajne istorije i na taj način odražava bar neke neopažene aspekte trenutnog stanja. Ažuriranje ove informacije o unutrašnjem stanju, zahteva dve vrste znanja koje moraju biti kodovane u agentskom programu. Prvo, potrebne su neke informacije o tome kako se okruženje razvija nezavisno od agenta. Drugo, potrebne su neke informacije o tome kako agentove sopstvene akcije utiču na okruženje. Ovo znanje o tome kako svet funkcioniše - zove se model sveta. Agent koji koristi takav model naziva se agent zasnovan na modelu.

Slika 4.13. daje strukturu refleksnog agenta sa unutrašnjim stanjem, i pokazuje kako se aktuelni opažaj kombinuje sa starim unutrašnjim stanjem u cilju generisanja novih opisa trenutnog stanja. Agentski program je prikazan na slici 4.14. Funkcija Ažuriraj\_stanje formira nove opise unutrašnjih stanja na osnovu tumačenje novog opažaja, znanja o tome kako akcije agenta menjaju okruženje, kao i informacije o tome kako se svet menja, čime se delimično kompenzuje parcijalna neobservabilnost okruženja.



Sl. 4.13. Refleksni agent zasnovan na modelu

```
function Refleksni_agent_zasnovan_na_modelu(opažaj) returns akcija
```

**stalni:** stanje – agentovo trenutno shvatanje sveta

model – opis zavisnosti sledećeg stanja od trenutnog stanja i akcije

pravila – skup pravila (uslov,akcija)

akcije – najnovija akcija (u početku prazno)

```
stanje ← Ažuriraj_stanje(stanje,akcija,opažaj,model)
```

```
pravilo ← Podudaranje_pravila(stanje,pravila)
```

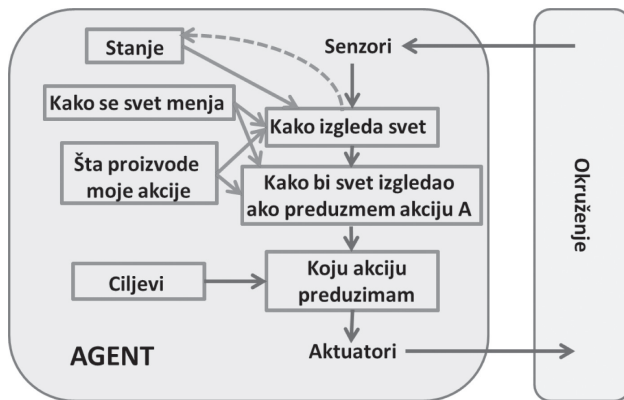
```
akcija ← pravilo.Akcija
```

```
return akcija
```

Slika 4.14. Refleksni agent zasnovan na modelu. Vodi računa o trenutnom stanju sveta koristeći unutrašnji model. Akciju bira na isti način kao i refleksni agent.

#### 4.4.3 Agenti zasnovani na cilju

Znanje o trenutnom stanju okruženja nije uvek dovoljno da bi se odlučilo šta dalje činiti. Drugim rečima, kao i trenutni opis stanja, agentu treba neka vrsta **ciljne** informacije koja opisuje situacije koje su poželjne. Agentski program može kombinovati ovo sa informacijama o rezultatima mogućih akcija da bi odabrao akciju kojom postiže cilj. Slika 4.15. prikazuje strukturu agenta zasnovanog na cilju.



Sl. 4.15. Model agenta zasnovanog na cilju. Agent vodi računa o stanju sveta i ciljevima koje želi da postigne birajući akcije koja će voditi popstizanju cilja.

Izbor akcije zasnovane na cilju je jednostavan ukoliko zadovoljenje cilja direktno sledi iz samo jedne akcije. Ukoliko je cilj moguće postići iz više koraka, neophodno je razmotriti čitave skupove sekvenci akcija da bi se pronašao način za postizanje postavljenog cilja. Ovim problemom se bave podoblasti VI, kao što su pretraživanje i planiranje. Iako se na prvi pogled čini da je agent zasnovan na cilju manje efikasan, poseduje važno svojstvo fleksibilnosti budući da je znanje koje podržava njegove odluke reprezentovano eksplicitno i može se menjati.

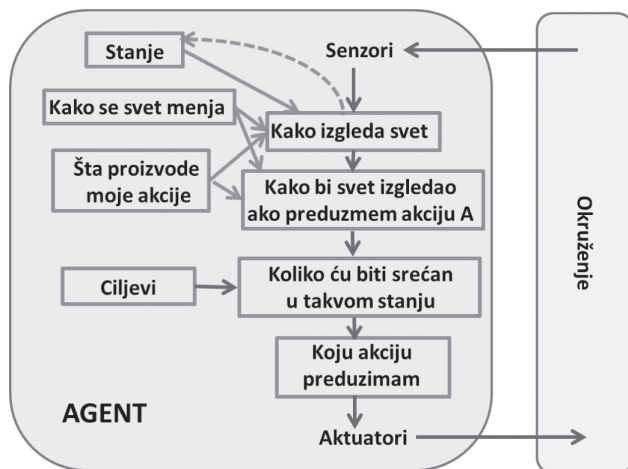
Tretiranje agenata kao agenata zasnovanih na cilju vodi poreklo iz kognitivne psihologije i oblasti ljudskog načina rešavanja problema. Newell i Simon su u uticajnom delu *Human Problem Solving*, iz 1972. godine primenili ovaj pristup u modelovanju i formalizovanju ljudskog načina rešavanja problema. Najraniji tragovi ove arhitekture mogu se naći u projektu Shakey, mobilnom robotu koji je razvijan na SRI (Stanford Research Institute), a vezana je za radove Nilsa Nillson-a, jednog od najplodnijih autora u okviru VI, Sl.4.16. Teorijsku logičku analizu agenta zasnovanog na cilju dao je Nillson u svojoj knjizi: Michael R. Genesereth and Nils J. Nilsson, *Logical Foundations of Artificial Intelligence*, Stanford University, 1987.



Sl.4.16. Charles Rosen (1917-2002) (levo), Nils Nilsson (1933 - ) (u sredini) i Milton Adams započeli su na SRI (Stanford Research Institute) rad na prvim mobilnim autonomnim robotima sredinom 60-tih godina XX veka. Ova istraživanja su rezultovala u robotu Shakey (desno), 1971. god., u kome je prvi put primenjena arhitektura agenta zasnovanog na cilju.

#### 4.4.4 Agenti zasnovani na korisnosti

Ako se ograničimo samo na ciljeve, imamo samo grubu binarnu razliku između dva krajnja stanja: uspeha pri postizanju cilja i neuspeha u suprotnom slučaju. Realna okruženja i zadaci, zahtevaju složeniju gradaciju uspeha koga postizemo odlučivanjem i sprovedenim akcijama. Prirodno se nameće kriterijum, čije vrednosti pokrivaju kontinuum vrednosti u unapred zadatom intervalu. Uobičajeno je da se u teoriji odlučivanja ovakva kriterijumska funkcija naziva funkcija korisnosti (engleski - utility). U opštem slučaju je potrebno razlikovati internu funkciju korisnosti od spoljašnjeg kriterijuma performansi agenta. Ukoliko su ove dve funkcije usaglašene, tada agent birajući sekvence akcija koje maksimiziraju korisnost za njega, istovremeno postiže i racionalnost u skladu sa spoljašnjim kriterijumom performansi. Kompletna specifikacija funkcija korisnosti omogućava racionalne odluke u dve vrste slučajeva u kojima su ciljevi neadekvatni. Prvo, kada postoje konfliktne ciljevi, gde samo neki od njih mogu biti postignuti (na primer, brzina i sigurnost u vožnji), funkcija korisnosti određuje odgovarajući kompromis. Drugo, kada postoji više ciljeva ka kojima agent može da teži, od kojih se ni jedan ne može sa sigurnošću postići, korisnost pruža način na koji se verovatnoći uspeha dodeljuju težinski faktori proistekli iz važnosti ciljeva. Struktura agenta zasnovanog na korisnosti prikazana je na slici 4.17.



Sl. 4.17. Model agenta zasnovanog na korisnosti. On koristi model sveta, zajedno sa funkcijom korisnosti koja meri njegove performanse među stanjima sveta. Zatim bira akciju koja dovodi do korisnosti sa najvećim očekivanjem, gde je očekivanje korisnosti usrednjavanjem svih mogućih stanja ishoda, odmerenim verovatnoćom ishoda.

Eric Horovitz i Judeja Pearl su u svojim radovima prvi predložili koncept racionalnog agenta zasnovanog na korisnosti, Sl.4.18.



Sl.4.18. Eric Horovitz i Judeja Pearl su u svojim radovima iz 1988. godine uveli pojam očekivane korisnosti kao osnovni kriterijum racionalnosti agenata. Pearlova iscrpna analiza verovatnosne strukture rasudjivanja i odlučivanja po principu korisnosti izneta u uticajnoj knjizi *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, presudno je uticala na popularnost agenata zasnovanih na cilju tokom 90-tih godina XX veka.

#### 4.4.5 Obučavajući agenti

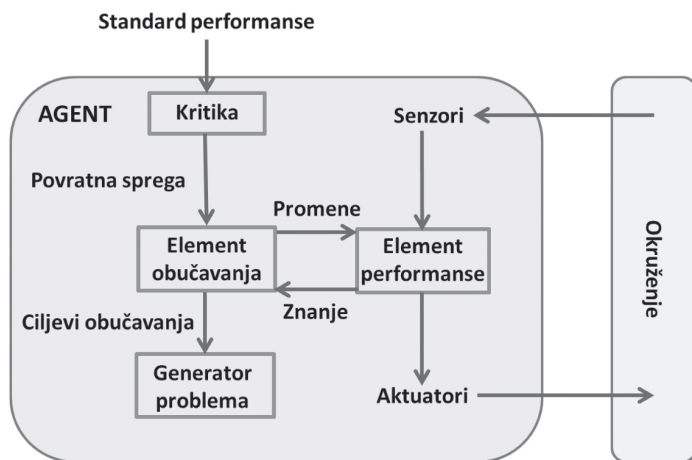
Stara ideja osnivača VI je bila da se u sintezi složenih sistema VI koristi prirodan princip učenja. Naime, ako nismo u stanju da napravimo sistem koji u samom startu poseduje sve željene performanse, cilj bi smo postigli ako bi smo ga snabdeli sposobno-



šću da uči. Kao i u biološkom svetu, kroz obučavanje, nesposobni i nemoćni mladunci posatju zrele jedinke sposobne za samostalan život u neprijateljskom okruženju. U mnogim oblastima veštačke inteligencije, ovo je sada metod kome se daje izrazita prednost. Obučavajući agent može biti podeljen u četiri konceptualne komponente, kao što je to prikazano na slici 4.19. Najvažnija je razlika između obučavajućeg elementa, koji je odgovoran za napredovanje u učenju, kao i elementa performansi, koji je odgovoran za izbor spoljnih aktivnosti. Element preformansi je ono što smo ranije smatrali da je ceo agent: on prihvata opažaje i odlučuje o akcijama. Obučavajući element koristi povratne informacije od kritike o tome kako agent radi i određuje kako treba izmeniti element performansi kako bi se bolje pokazao u budućnosti.

Kritika govori obučavajućem elementu koliko dobro agent radi u odnosu na standardnu fiksnu performansu. Kritika je neophodna jer opažaji sami po sebi ne daju indikaciju uspeha agenta. Poslednja komponenta obučavajućeg agenta je generator problema. On je odgovoran za predlaganje aktivnosti koje će dovesti do novih i informativnih iskustava.

Obučavajući element može da promeni bilo koju od komponenata “znanja” prikazanih u dijagramu agenata sa Sl.4.10, Sl.4.13, Sl.4.15 i Sl.4.17). Najjednostavniji slučajevi uključuju direktno učenje iz opažajne sekvence. Posmatranje parova uzastopnih stanja okruženja može dozvoliti agentu da nauči “Kako se svet menja”, a posmatranje rezultata sopstvenih aktivnosti može dozvoliti agentu da nauči “Šta proizvode moje akcije”. Ova dva zadatka obučavanja su teška ako je okruženje samo delimično observabilno.



Sl. 4.19. Opšti model obučavajućeg agenta

Nećemo pogrešiti ako kažemo opšti stav da je, unatoč mogućih raznovrsnosti obučavajućih agenata, njihova suština u modifikaciji svake komponente agenta kako bi se poboljšale ukupne performanse celog agenta.

Prvi program koji je imao sposobnost učenja napisao je pionir oblasti računarskih igara Artur Lee Samuel 1959. godine za igru Dame, Sl.4.20. To je ujedno i najraniji trag arhitekture agenta sa obučavanjem.



Sl.4.20. Arthur Lee Samuel (1901-1990) američki pionir u oblasti računarskih igara, veštačke inteligencije i mašinskog učenja. Samuelov program za igranje dame je prvi samoobučavajući sistem ikad isprogramiran na digitalnom računar. Možemo ga smatrati i prvim agentom u kome je ugrađena mogućnost obučavanja u cilju poboljšanja performansi.

## Rezime 4. poglavlja

- ♦ Sistemi VI zahtevaju specifične arhitekture, budući da su po pravilu orijentisani na intenzivnu pretragu u prostoru mogućih rešenja. Nezavisnost pojedinih komponenti arhitekture olakšava razvoj složenih sistema.
- ♦ Produkcioni sistemi predstavljaju jednu od osnovnih arhitektura i opštih računskih formalizama pogodnih za sisteme VI. Nezavisnost globalne baze, skupa pravila i kontrolnih strategija obezbeđuju efikasnu modularnost. U pogledu računarske snage ekvivalentni su Turingovoj mašini.
- ♦ Druga velika grupa savremenih arhitektura u VI se odnosi na agente. To su entiteti koji senzorima opserviraju svoje okruženje i na njega deluju preko efektor. Postali su posebno popularni sa pojavom interneta, na kome su poznati pod nazivom softbots.
- ♦ U hijerarhiji složenosti izučavali smo sledeće vrste agenata, počev od najjednostavnijih: jednostavni refleksni agenti, refleksni agenti zasnovani na modelu, agenti zasnovani na cilju, agenti zasnovani na korisnosti i obučavajući agenti.

## Pitanja i zadaci

1. Definišite svojim rečima produkcionni sistem.
2. Zašto je bitno kod produkcionnih sistema da su GB, skup pravila i kontrolna strategija nezavisne?
3. Odrediti GB, skup pravila i terminalne uslove produkcionnog sistema za rešavanje problema misionara i ljudoždera: Tri misionara i tri ljudoždera su došli do obale reke gde ih čeka čamac koji može da prevozi jednu ili dve osobe. Kako koristiti čamac za prelazak preko reke a da ni u jednom trenutku broj ljudoždera ne bude veći od broja ljudi ni na jednoj strani reke?

4. Odrediti GB, skup pravila i terminalne uslove za produkcionu sistem koji rešava sledeći problem: Dat je čup od 5 litara napunjen vodom do vrha i prazan čup od 2 litra. Kako doći do sadržaja od 1 litra u čupu od 2 litra. Voda se može prosipati ili prelivati iz jednog čupa u drugi.
5. GB nekog produkcionog sistema se sastoji od intedžera. GB se može transformisati dodavanjem proizvoda bilo koja dva broja iz GB. Pokazati da je ovaj produkcionu sistem komutativan.
6. Pokazati kako se produkcionu sistem može upotrebiti za konvertovanje decimalnih u binarne brojeve. Ilustrovati rad produkcionog sistema prilikom konverzije broja 242.
7. Napišite programe agenta u pseudokodu za agente zasnovane na cilju i na korisnosti.
8. Dajte primere nekih od navedenih vrsta agenata koji danas obavljaju neke od servisa na internetu.
9. Ako se stavite u ulogu dizajnera sofisticiranih virusa, koji tip arhitekture agenta bi obezbedjivao najveću malicioznost?
10. Ako jedan medicinski dijagnostički sistem posmatramo kao agenta, šta u tom slučaju predstavlja mera performanse, okruženje, aktuatori i senzori?
11. Ako robota za izdvajanje delova posmatramo kao agenta, šta u tom slučaju predstavlja meru performanse, okruženje, aktuatori i senzori?



## 5. Strategije pretrage u sistemima veštačke inteligencije

U ovom poglavlju prezentovaćemo najvažnije strategije upravljanja produkcionih sistema VI. Osnovni problemi koje strategije upravljanja rešavaju su izbor primenljivog pravila u svakom ciklusu rada sistema, a kod razloživih sistema se ovaj pojam proširuje i na izbor komponente GB koju treba obraditi u datom ciklusu. Ostali ne manje važni zadaci strategija upravljanja obuhvataju proveru uslova primenljivosti pravila, proveru terminalnog uslova, kao i pamćenje sekvence već oprobanih pravila.

U odnosu na mogućnost opovrgavanja razlikujemo dve klase strategija

- ♦ **probne (tentative)** – kod kojih se uvek možemo vratiti u prethodne tačke izbora i izabrati neko drugo pravilo umesto prethodno izabranog,
- ♦ **nepovratne (irrevocable)** – kod kojih je nemoguć povratak na prethodne tačke izbora i ponovno razmatranje izbora nekog drugog pravila.

U okviru probnih strategija možemo razlikovati dve podgrupe, saglasno istorijskom razvoju ove oblasti:

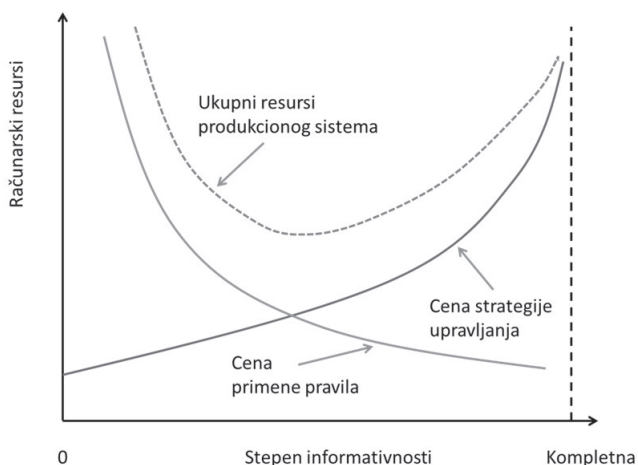
- ♦ **povratne (backtracking)** – kod kojih se prilikom izbora pravila ustanovljava tačka povratka, u koju se strategija upravljanja vraća birajući neko drugo pravilo, ukoliko tekući niz izabranih pravila ne dostiže cilj
- ♦ **strategije pretrage na grafu** – kod kojih se na sistematičan način memorišu efekti primene više sekvenci pravila istovremeno. Ova klasa strategija je vezana, u formalnom smislu, za grafovske strukture koje su pogodno za implementaciju i prikaz ovog tipa pretraga.

Važnu ulogu u ovom procesu igra znanje o problemu koji se rešava. U tom pogledu razlikujemo dve osnovne grupe strategija:

- ♦ **neinformativne**, koje ne koriste znanje o rešavanom problemu,
- ♦ **informativne**, koje intenzivno koriste raspoloživa znanja o rešavanom problemu.

U slučaju ekstremne neinformisanosti, pravila se biraju potpuno proizvoljno, recimo slučajno. U slučaju ekstremne informisanosti, strategija upravljanja je vodjena znanjem o problemu, koje je dovoljno da u svakom koraku bira „pravo“ pravilo koje nas na najdirektniji način vodi ka rešenju.

Ukupna računarska efikasnost produkcionog sistema zavisi u koji deo oblasti izmedju kompletne neinformisanosti-informisanosti pada konkretna izabrana strategija upravljanja. Ukupne računarske resurse koje tom prilikom trošimo možemo dekomponovati na dve glavne komponente: cenu primene pravila i cenu strategije upravljanja. U slučaju kompletne neinformisanosti, kontrolna strategija zahteva minimalne računarske resurse budući da proizvoljan izbor pravila u tom pogledu nije zahtevan. Međutim, ovakve strategije imaju velike računarske zahteve u pogledu primene pravila, budući da tada sa velikom verovatnoćom moramo oprobati veliki broj sekvenci pravila do postizanja konačnog rešenja. S druge strane, u slučaju informaivnih strategija, računarski resursi se troše u velikoj meri na kontrolne strategije, a u znatno manjoj meri na primenjena pravila, budući da očekujemo isprobavanje znatno manjeg broja sekvenci pravila do postizaja konačnog rešenja. Ove opšte tendencije su prikazane na Sl.5.1.



Sl.5.1 Računarski resursi koje troše produkциони sistemi VI u zavisnosti od stepena informativnosti primenjenih strategija.

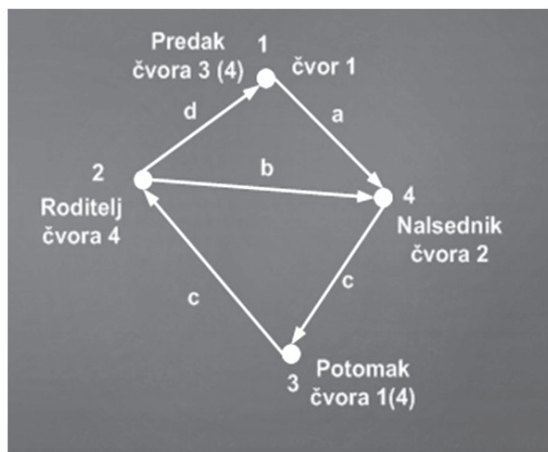
Ukupni utrošeni računarski resursi su zbir ove dve komponente. Dobro dizajnirani sistemi nalaze balans izmedju ove dve komponente, kao što se to vidi na Sl.5.1 Za svaki konkretan problem, optimalnost se po pravilu postiže strategijama upravljanja koje

nisu kompletno informativne, zato što se kompletna informativnost postiže na račun isuviše velike računarske kompleksnosti.

Ključna opservacija u vezi sa strategijama upravljanja produkcionih sistema se odnosi na činjenicu da ih možemo posmatrati kao proces pretrage sekvence pravila koje dovode do željenog cilja. Ova činjenica je uticala na konstituisanje problema pretrage kao samostalne podoblasti VI, a u matematičkom kontekstu kao dela diskretnih optimizacionih procedura za kojima postoji širi interes u gotovo svim savremenim naučnim disciplinama koje se oslanjaju na matematičke modele i njihovu identifikaciju i optimizaciju. Stoga ćemo poglavlje nastaviti pregledom osnovnih pojmova teorije grafova, neophodnih za praćenje i pravilno shvatanje savremenih algoritama pretrage.

## 5.1 OSNOVNI POJMOVI O GRAFOVIMA

**Graf** se sastoji od čvorova i grana koje spajaju čvorove. **Grane** mogu biti usmerene (digraf – usmereni graf) ili neusmerene (neusmereni graf). Ako od čvora  $n_i$  postoji direktna grana ka čvoru  $n_k$ , tada se  $n_i$  naziva roditeljski čvor, a  $n_k$  naslednik čvora  $n_i$ . Na Sl.5.2 prikazan je primer jednog digrafa. Uočava se da u opštem slučaju nasledna svojstva gube prirodni smisao, ukoliko graf sadrži zatvorene petlje. Npr. ako u grafu sa Sl.5.2 izaberemo da je čvor 2 roditeljski čvor, tada sledeći grane b i c, dobijamo da je on istovremeno i svoj sopstveni potomak.

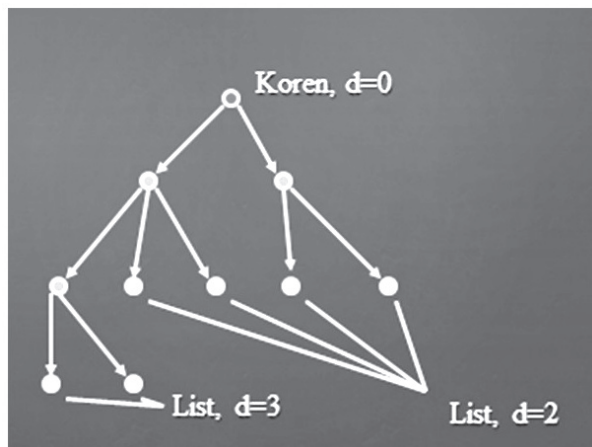


Sl.5.2 Digraf (usmereni graf) sa četiri čvorova i pet grana. Digrafi sa petljama (čvorovi: 2-4-5) nisu pogodni za kodovanje naslednih svojstava. Npr. ako čvor 2 izaberemo za roditeljski čvor, on istovremeno može biti i sopstveni potomak.

Poznato je u teoriji grafova da su grafovske strukture tipa stabla pogodne za jedinstveno prikazivanje naslednih svojstava. Reći ćemo da je graf **povezan** ako nema slobodnih nepovezanih čvorova. **Stablom** ćemo nazivati one povezane grafove koje

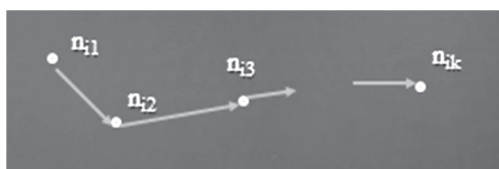


imaju  $n$  čvorova i  $n-1$  granu. Ekvivalentna definicija je da su stabla povezani grafovi u kome svaki čvor ima samo jednog roditelja izuzev samo jednog čvora koji nema roditelje. Čvor koji nema roditelje se naziva **koren**, dok su čvorovi bez naslednika – **listovi** (krajnji čvorovi stabla), Sl.5.3.



Sl.5.3 Graf tipa stabla sa šest listova, od čega su dva lista dubine 3, a preostala četiri dubine 2. Koren je jedinstveni čvor koji nema roditelje, dok su listovi oni čvorovi koji nemaju naslednike.

**Dubina čvora** u stablu je jednaka dubini roditeljskog čvora uvećano za jedan, pri čemu je dubina korena jednaka 0.



Sl.5.4 Put dužine  $k$  u jednom grafu

**Put dužine  $k$**  od čvora  $n_{i_1}$  do čvora  $n_{i_k}$  je niz čvorova

$$n_{i_1}, n_{i_2}, \dots, n_{i_k}$$

gde je svaki čvor  $n_{i_j}$  naslednik čvora  $n_{i_{j-1}}$ ,  $j=2,3,\dots,k$ , videti Sl.5.4.

Ako postoji put od čvora  $n_i$  do čvora  $n_j$  kažemo da je  $n_j$  **dohvatljiv** (dostižan) iz  $n_i$ . Ujedno je  $n_j$  **potomak** od  $n_i$ , a  $n_i$  je **predak** od  $n_j$ .

Zadatak nalaženja niza pravila koji transformišu GB od početnog do terminalnog stanja je ekvivalentno nalaženju jednog puta na grafu pretrage. Cena primene pravila se može izraziti uvođenjem težina grana. Stoga se nalaženje najefikasnijih (optimal-

nih) strategija svodi na nalaženje puta najmanje težine od početnog do ciljnog skupa čvorova. Težina jednog puta u grafu jednaka je zbiru težina svih grana koje spajaju čvorove posmatranog puta.

Grafovi se mogu prezentovati eksplicitno ili implicitno. U eksplicitnoj specifikaciji čvorovi, grane i njihove korespondentne težine su date tabelarno. Npr. tabeli se može sastojati od liste svih čvorova grafa, njihovih naslednika i težina asociranih grana. Jasno je da eksplicitna specifikacija nije praktična za grafove velikih dimenzija, a praktično je nemoguća za grafove sa beskonačnim brojem čvorova. Za velike grafove pretrage, pogodna je implicitna reprezentacija pomoću:

- ♦ početnog čvora,
- ♦ skupa pravila i
- ♦ tzv. operatora formiranja naslednika ili operatora otkrivanja.

Primenom operatora otkrivanja na dati čvor generišu se svi njegovi naslednici i asocirane težine grana. Operator otkrivanja direktno zavisi od prmenljivih pravila na GB koju prezentuje čvor koji se otkriva. U tom slučaju kažemo da je dati čvor otkriven. Svaka strategija u ovakvoj formulaciji, indukuje (otkriva) implicitno zadati graf koji sadrži ciljni čvor, Sl.5.5.



Sl.5.5 Strategija pretrage zasnovana na grafovima može se posmatrati kao proces eksplicitnog generisanja jednog dela implicitno zadatog grafa, tako da sadrži startni i ciljni čvor.

## 5.2 OPŠTI ALGORITAM PRETRAGE NA GRAFU – ALGORITAM GRAPHSEARCH

Proces eksplicitnog generisanja dela jednog implicitno definisanog grafa može se neformalno formulisati sledećom procedurom:

### Algoritam GRAPHSEARCH

1. Kreirati graf pretrage G, koje se u početku sastoji samo od startnog čvora s. Staviti s u uređenu listu OPEN.
2. Kreirati listu CLOSED, koja je u početku prazna.
3. Ako je OPEN prazna lista, neuspešan kraj procedure.
4. Selektovati prvi čvor n iz OPEN. Izbrisati ga iz OPEN i zapisati u listu CLOSED.
5. Ako je n ciljni čvor, procedura se završava sa uspehom. Ukupno rešenje se dobija preko ukazatelja iz G, unazad od n do s. (Ukazatelji se formiraju u koraku 6.)
6. Otvoriti čvor n generisanjem skupa M svih njegovih naslednika. Uvesti ukazatelje za sve elemente iz M ka n u G, u cilju identifikacije ovih čvorova kao naslednika čvora n. Za sve čvorove iz M, koji su već u CLOSED odlučiti da li treba promeniti ukazatelje njegovih potomaka iz G.
7. Sortirati listu OPEN, proizvoljno ili na osnovu neke heuristike.
8. Preći na korak 3.

Sl.5.6 Opšti algoritam pretrage na grafu – algoritam GRAPHSEARCH. Lista OPEN sadrži front pretrage, odnosno čvorove koji se potencijalno mogu otkriti u narednim koracima pretrage. Lista CLOSED sadrži čvorove preko kojih je proces pretrage prošao i koji su generisali svoje naslednike. Graf G predstavlja eksplicitno generisan graf pretrage.

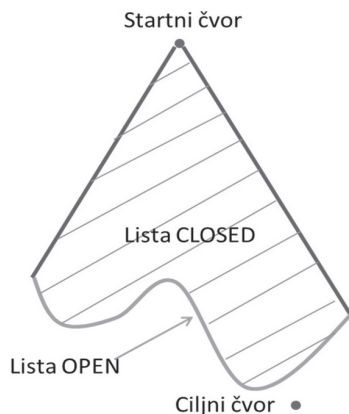
GRAPHSEARCH eksplicitno formira graf pretrage G i njegov podgraf - stablo pretrage Tr. Stablo Tr je određeno ukazateljima (korak 6). Ovim se obezbeđuje jedinstvenost roditelja. Čvorovi iz OPEN su krajnji čvorovi stabla pretrage, koji još nisu birani za otvaranje. Čvorovi iz CLOSED su ili krajnji čvorovi izabrani za otvaranje koji nemaju naslednike, ili neterminalni čvorovi ovog stabla.

Svi mogući putevi do nekog čvora, otkriveni ovim algoritmom su u G u eksplicitnom obliku, dok je jedan izabrani put do svakog čvora dat stablom Tr. Rešenje problema pretrage se dobija sledovanjem ukazatelja od terminalnog (ciljnog) do početnog (startnog) čvora.

#### 5.2.1 RAZJAŠNJENJE KORAKA 6

Kada bi G bilo stablo, bili bismo sigurni da ni jedan od potomaka generisan u koraku 6. nije već prethodno generisan. Pri ponovnom otkrivanju nekog čvora, do njega se otvara novi put, različit od onog koji je do tada fiksiran u stablu pretrage Tr. Ukoliko je ovaj put kraći (manje težine) od onog koji je do tada postojao, potrebno je korigovati stablo pretrage Tr.





Sl.5.8 Opšta fenomenologija procedure GRAPHSEARCH. Od startnog čvora napreduje front pretrage definisan listom OPEN. Čvorovi preko kojih je pretraga prešla nalaze se u listi CLOSED. Procedura se zaustavlja uspešno kada front pretrage obuhvati i ciljni čvor.

Na Sl.5.8 data je opšta fenomenologija procedure GRAPHSEARCH. Od startnog čvora napreduje front pretrage definisan listom OPEN. Čvorovi preko kojih je pretraga prešla nalaze se u listi CLOSED. Procedura se zaustavlja uspešno kada front pretrage obuhvati i ciljni čvor. Širenje fronta pretrage je u potpunosti određeno strategijom redosleda otvaranja čvorova iz OPEN. Prema koraku 7 algoritma, nakon svake promene u listi OPEN vrši se sortiranje čvorova shodno nekoj evaluacionoj funkciji, tako da na prvo mesto dolazi čvor sa najvećom perspektivnošću vođenja ka rešenju. Ova evaluaciona funkcija se u sistemima VI naziva **heuristička funkcija**. Njena konstrukcija se zasniva na heurističkim informacijama iz posmatranog problemskog domena, i kao što ćemo videti u narednim poglavljima igra ključnu ulogu u sintezi efikasnih procedura pretraga. Ukoliko bi heuristička funkcija bila idealna i odražavala ukupno suštinsko znanje o problemu koji se rešava, ona bi vršila idealno uređenje čvorova iz liste OPEN i na prvo mesto stavljala čvor koji se nalazi na direktnom (najkraćem) putu do ciljnog čvora.

### 5.3 MERENJE PERFORMANSE STRATEGIJA PRETRAGE

Rezultat algoritma pretrage je ili rešenje koje se dobija dostizanjem ciljnog čvora, ili neuspeh. Neki algoritmi se mogu zaglaviti u beskonačnu petlju i nikada ne završavaju rad. Svaki algoritam pretrage možemo karakterisati preko četiri pokazatelja performansi:

- ♦ **Kompletnost:** Da li algoritam garantuje da će naći rešenje kada ono postoji?
- ♦ **Optimalnost:** Da li algoritam nalazi optimalno rešenje najmanje težine?
- ♦ **Vremenska kompleksnost:** Koliko vremena treba da se nađe rešenje?
- ♦ **Prostorna kompleksnost:** Koliko je memorije potrebno da bi se izvršila pretraga?

U sistemima VI kompleksnost je izražena u funkciji tri veličine:

- ♦ b - faktora grananja ili maksimalni broj naslednika bilo kog čvora;
- ♦ d - dubina najplićeg ciljnog čvora;
- ♦ m - maksimalna dužina bilo koje putanje u prostoru stanja.

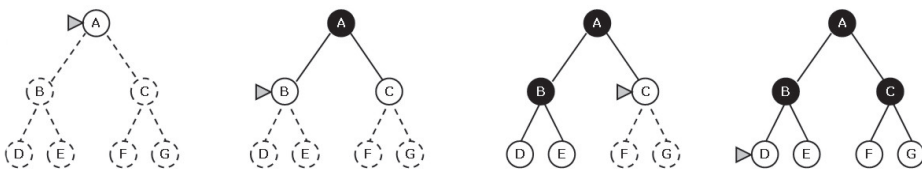
Vremenska kompleksnost se izražava u terminima broja generisanih čvorova tokom pretrage, a prostorna kompleksnost u terminima maksimalnog broja memorisanih čvorova. Težina rešenja se određuje kao zbir težina grana na putu od startnog do ciljnog čvora sledeći ukazatelje koje generiše algoritam GRAPHSEARCH. Težine pojedinih grana se određuju prema prirodi problema.

## 5.4 NEINFORMATIVNE PROCEDURE PRETRAGE NA GRAFU

Ukoliko ne raspolažemo heurističkim informacijama, možemo usvojiti proizvoljan postupak uredjenja čvorova u koraku 7. Rezultujuće procedure pretrage se nazivaju neinformativne pretrage. Budući da u njima nema ugrađenog znanja o problemskom domenu, obe su primenljive na svaki problem, ali je zato i za očekivati da ne poseduju efikasnost, često presudnu za probleme pretrage čiji je prostor pretrage velik. Ono što je bitno u teorijskom smislu je činjenica da je algoritam GRAPHSEARCH dovoljno opšti za obuhvatanje i informativnih i neinformativnih strategija pretrage.

### 5.4.1 PRETRAGA U ŠIRINU

Pretraga u širinu se dobija iz opšteg algoritma GRAPHSEARCH, kada se na prvo mesto u listi OPEN stavlja čvor čija je dubina najmanja. Ukoliko više čvorova ima istu dubinu, njihovo uredjenje je prouzvoljno. Kao rezultat dobijamo sistematsku pretragu po slojevima iste dubine, tako da se prvo otkriju svi čvorovi na datoj dubini pre nego što se predje na bilo koji čvor dubine veće za jedan. Ovakva pretraga implementira tzv. FIFO (First Input, First Output - Prvi unutra, prvi napolje) princip. Na Sl. 5.9 pokazani su koraci pretrage za primer u kome svaki čvor ima po dva naslednika.



Sl.5.9 Primer pretraga u širinu kada svaki čvor ima po dva naslednika. Na svakom koraku rada algoritma, čvor koji je određen za otkrivanje je obeležen znakom malog trougla.

Izvršimo evaluaciju ove strategije prema uvedenim kriterijumima iz prethodnog odeljka. Pretraga u širinu je kompletna. Ovo zaključujemo iz činjenice da ako je najplići ciljni čvor na dubini  $d$ , pretraživanje u širinu će ga naći nakon što otkrije sve pliće čvorove, pod uslovom da je faktor grananja  $b$  konačan. Najplići ciljni čvor nije neophodno i optimalan. Preciznije govoreći, pretraga u širinu je optimalna ako je cena putanje neopadajuća funkcija dubine čvora. Ovo je npr. uvek slučaj ako su sve grane iste težine.

Da bismo izračunali vremensku i prostornu kompleksnost zamislimo prostor pretrage u kome svaki čvor ima isti broj naslednika  $b$ . Koren stabla pretrage generiše  $b$  čvorova na prvom nivou, od kojih svaki generiše još  $b$  čvorova, tako da na drugom nivou ima ukupno  $b^2$  čvorova. Pretpostavimo sada da je rešenje na dubini  $d$ . U najgorem slučaju otkrićemo sve osim poslednjeg čvora na nivou  $d$ , generišući  $b^{d+1} - b$  čvorova na nivou  $d + 1$ . Tada je ukupan broj otkrivenih čvorova:

$$b + b^2 + b^3 + \dots + b^d + (b^{d+1} - b) = O(b^{d+1}).$$

Primetimo da je u ovom slučaju testiranje ciljnog čvora vršeno pri generisanju svakog pojedinačnog čvora. Ukoliko bi testiranje ciljnog čvora obavljano prilikom biranja čvora za otkrivanje, ukupan broj otkrivenih čvorova bio bi:

$$b + b^2 + b^3 + \dots + b^d = O(b^d).$$

Svaki generisani čvor mora ostati u memoriji pa je prostorna kompleksnost reda  $O(b^d)$ . Dakle našli smo da je pretraga u širinu eksponencijalne vremenske i prostorne složenosti, što je čini praktično primenljivom samo za vrlo jednostavne probleme.

## 5.4.2 PRETRAGA SA UNIFORMNIM TEŽINAMA

Pretraga u širinu je optimalna kada su težine svih koraka jednake, jer se uvek otkriva najplići neotkriveni čvor. Jednostavnim proširivanjem ovog algoritma, može se dobiti algoritam koji je optimalan za bilo koje težine grana. Umesto otkrivanja najplićeg čvora, pretraga sa uniformnim težinama otkriva čvor  $n$  sa najmanjom težinom puta  $g(n)$  od startnog čvora do čvora  $n$ . Ovaj mehanizam se postiže uredjenjem liste OPEN, tako da se na prvom mestu uvek nalazi čvor sa najmanjim  $g$ . Osim toga u ovom algoritmu se proverava da li je dostignut cilj obavlja onda kada je neki čvor izabran za otkrivanje, umesto kada se prvi put generiše.

Algoritam pretrage sa uniformnim težinama je kompletan ukoliko su težine svakog koraka veće ili jednake maloj pozitivnoj konstanti  $\epsilon$ . Dalje se pokazuje da je ovaj algoritam i optimalan u opštem slučaju. Naime, kad god je neki čvor izabran za otkrivanje, težina puta od startnog čvora do njega je optimalna (najkraća). Uz to uvek važi i svojstvo da težina puta nikad ne postaje kraća dodavanjem novog čvora (usled pozitivnosti svake težine grana). Stoga, prvi ciljni čvor izabran za otkrivanje mora biti optimalno rešenje.



Pretraga sa uniformnim težinama je vodjena težinama putanje, a ne dubinom čvorova pa se kompleksnost ne može direktno izraziti u funkciji veličina  $b$  i  $d$ . Stoga ćemo uzeti da je  $C^*$  težina optimalnog rešenja i pretpostaviti da svaka akcija ima težinu najmanje  $\varepsilon$ . Tada je vremenska i prostorna složenost algoritma reda  $O(b^{1+\lceil C^*/\varepsilon \rceil})$ , koja može biti znatno veća od  $b^d$ . Kada su sve težine koraka jednake, tada  $b^{1+\lceil C^*/\varepsilon \rceil}$  ne može biti mnogo veće od  $b^d$ . Stoga, kada su težine svih grana jednake, pretraživanje sa uniformnim težinama je vrlo slično pretrazi po širini izuzev razlike u kriterijumu za zaustavljanje. Podsetimo se da kod pretrage u širinu algoritam završava posao čim generiše ciljni čvor, dok kod pretrage sa uniformnim težinama se moraju ispitati svi čvorovi na istoj dubini cilja, da bi se proverilo da li među njima ima rešenja sa kraćom težinom puta. Prema tome, kada su sve grane iste težine, pretraživanje sa uniformnom težinom u opštem slučaju radi duže, nepotrebno otkrivajući sve čvorove na dubini  $d$ .

### 5.4.3 PRETRAGA PO DUBINI

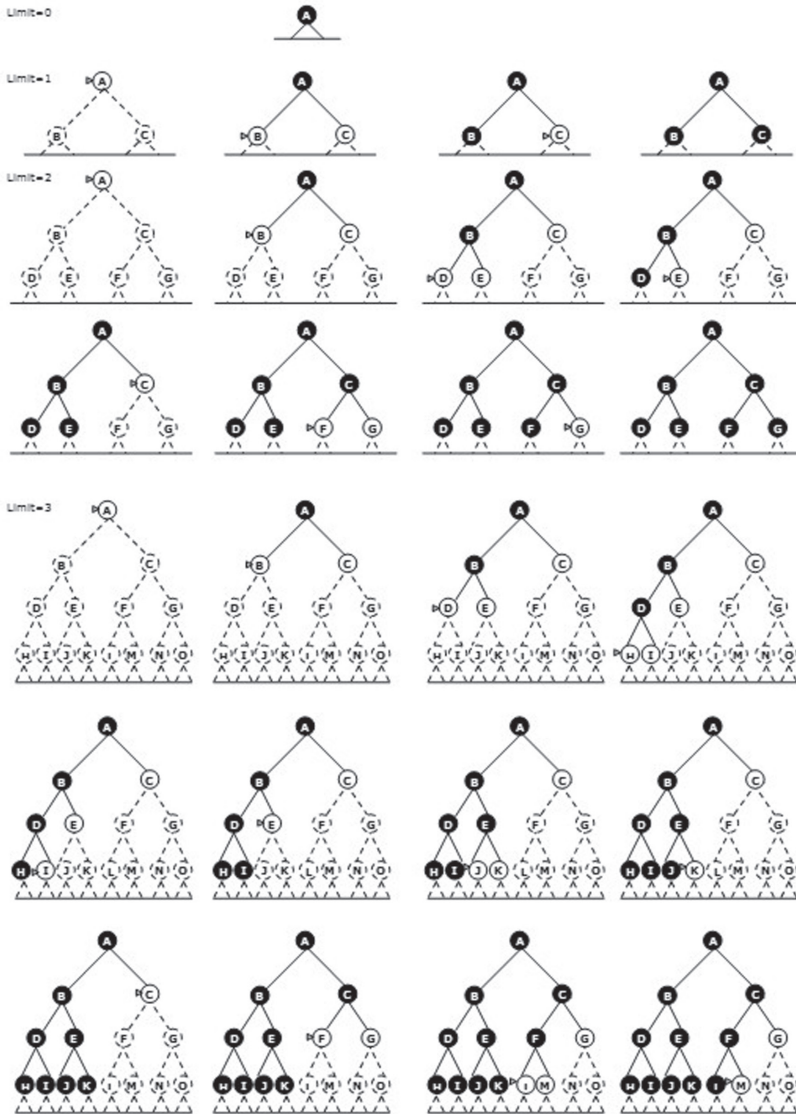
Pretraga po dubini uvek otkriva najdublji čvor iz liste OPEN. Na Sl. 5.10 dat je redosled pretraživanja po širini za primer u kome svi čvorovi imaju dva naslednika. Ovaj algoritam koristi tzv. LIFO (Least Input First Output – poslednji unutra, prvi napolje) princip u redu čekanja unutar liste OPEN. Algoritam radi tako štovrlo brzo silazi na najdublji nivo stabla pretraživanja na kome čvorovi nemaju naslednike. Nakon otkrivanja tih čvorova, pretraživanje se vraća na sledeći najdublji čvor koji nije do sada otkriven i koji ima naslednike.

Algoritam pretrage po dubini nije optimalan. U to se možemo uveriti posmatrajući primer sa Sl.5.10. Algoritam bi razvio levi deo stabla pretrage, čak ako je čvor  $C$  ciljni čvor. Ako bi npr. i čvor  $J$  bio ciljni čvor, algoritam bi njega izabrao za rešenje, iako je put od startnog čvora do njega duži od težine puta do pretpostavljenog drugog ciljnog čvora  $C$ .

Vremenska kompleksnost algoritma je reda  $O(b^m)$ , gde je  $m$  maksimalna dubina bilo kog čvora u stablu pretrage. Ova veličina može biti znatno veća od  $d$  (dubine najbližeg ciljnog čvora). Prednost ovog algoritma je u prostornoj složenosti. Kada se čvor jednom otkrije, on se može ukloniti iz OPEN kada se istraže svi njegovi naslednici, kako je to ilustrovano na Sl.5.10. Za prostor stanja čiji je faktor grananja  $b$  i maksimalna dubina  $m$ , zahtevani memorijski prostor je samo  $O(bm)$ . Ovo je dramatična razlika u odnosu na memorijske zahteve algoritma pretrage u širinu. Stoga je algoritam pretrage u dubinu bio jedan od osnovnih pokretača razvoja sistema VI.



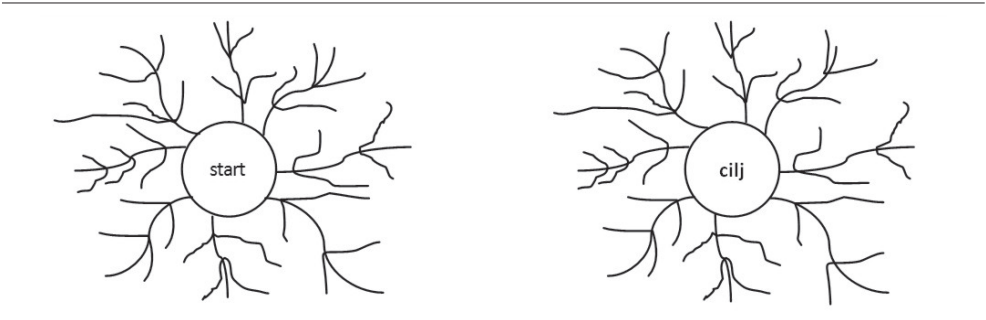
postepeno povećava granicu dubine - prvo 0, zatim 1, 2 i tako redom, sve do dubine  $d$  najbližeg ciljnog čvora. Ilustracija rada ovog algoritma je data na Sl.5.11. Iterativno produbljivanje kombinuje dobre osobine pretrage u širinu i dubinu. Kao i kod pretraga u dubinu, memorijski zahtevi su mali -  $O(bd)$ . Nasledjujući dobra svojstva algoritma pretrage u širinu, kompletnost je garantovana kada je faktor grananja konačan, a optimalnost kada je težina putanje neopadajuća funkcija od dubine čvora. Generalno, algoritam iterativnog produbljivanja je najbolja neinformativna metoda pretrage, kada je prostor pretraživanja veliki a nije poznata dubina rešenja.



Sl.5.11 Ilustrativan primer rada četiri iteracije algoritma pretrage po dubini sa iterativnim produbljivanjem kada svi čvorovi imaju samo dva naslednika.

5.4.4 DVOSMERNE PRETRAGE

Dvosmernost smo već pomenuli kod osnovnih tipova produkcionih sistema. Osnovna ideja dvosmerne pretrage je simultano korišćenje dva procesa pretrage: direktnog koji je započeo od startnog čvora i reverznog koji je starovao od ciljnog čvora. Algoritam završava rad kada se ova dva procesa susretnu, odnosno generišu isti čvor, Sl.5.12. Vremenska složenost dvosmerne pretrage  $b^{d/2} + b^{d/2}$  je značajno manja od vremenske složenosti jednosmerne pretrage  $b^d$ .



Sl.5.12 Opšta topologija dvosmerne pretrage. Vremenska kompleksnost je onoliko puta smanjena u odnosu na jednosmernu pretragu koliko je puta manji zbir površina krugova dvosmerne pretrage u odnosu na ekvivalentni krug jednosmerne pretrage čiji je poluprečnik jednak rastojanju između startnog i ciljnog čvora.

5.5. POREĐENJE PERFORMANSI NEINFORMATIVNIH STRATEGIJA

Slika 5.13 daje preglednu tabelu svojstava svih neinformativnih algoritama pretrage obradjenih u ovom poglavlju.

| Kriterijum  | U širinu        | Uniformne težine          | Po dubini | Ograničena dubina | Iterativno produbljenje | Dvosmerno         |
|-------------|-----------------|---------------------------|-----------|-------------------|-------------------------|-------------------|
| Kompletnost | Da <sup>a</sup> | Da <sup>a,b</sup>         | Ne        | Ne                | Da <sup>a</sup>         | Da <sup>a,b</sup> |
| Vreme       | $O(b^{d+1})$    | $O(b^{1+[C^*/\epsilon]})$ | $O(b^m)$  | $O(b^l)$          | $O(b^d)$                | $O(b^{d/2})$      |
| Prostor     | $O(b^{d+1})$    | $O(b^{1+[C^*/\epsilon]})$ | $O(bm)$   | $O(bl)$           | $O(bd)$                 | $O(b^{d/2})$      |
| Optimalnost | Da <sup>c</sup> | Da                        | Ne        | Ne                | Da <sup>c</sup>         | Da <sup>c,d</sup> |

Sl.5.13 Tabela prikazuje evaluaciju neinformativnih strategija pretrage, obradjenih u ovom poglavlju. Oznake:  $b$  – faktor grananja,  $d$  – dubina najplićeg ciljnog čvora,  $m$  – maksimalna dubina stabla pretrage,  $l$  – granica dubine,  $C^*$  – težina optimalnog rešenja. Oznake u eksponentu: <sup>a</sup> kompletan ako je  $b$  konačno; <sup>b</sup> kompletan ako su težina koraka  $\leq \epsilon$  za pozitivno  $\epsilon$ ; <sup>c</sup> optimalan ako su sve težine koraka jednake; <sup>d</sup> ako se u oba smera koriste pretrage u širinu.

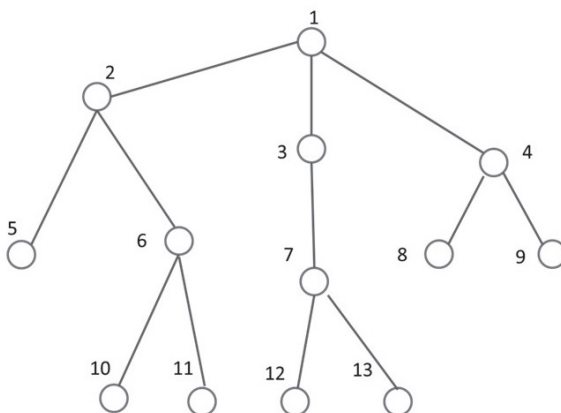
## Rezime 5. poglavlja

- ♦ Strategije upravljanja u produkcionim sistemima VI se mogu posmatrati kao ekvivalentne pretrage na implicitno zadatom grafu definisanom početnom i terminalnom globalnom bazom i operatorom otkrivanja čvorova, kojim se generišu svi naslednici, odnosno rezultatne GB dobijene primenom svih primenljivih pravila.
- ♦ Strategije upravljanja se dele na informativne i neinformativne, shodno tome da li se preference u redosledu otkrivanja čvorova grafa pretrage računaju na osnovu informacija o problemskom domenu ili ne.
- ♦ Jedan algoritam Pretrage Drveta se može koristiti da reši bilo koji problem; specifično varijante algoritma koji koristi različite strategije.
- ♦ Osnovna svojstva algoritama pretrage su kompletnost, optimalnost, vremenska i prostorna složenost.
- ♦ Opšta struktura svih algoritama pretrage data je algoritmom GRAPHSEARCH, koji na sistematičan način opisuje nastajanje eksplicitno zadatog grafa pretrage iz implicitno zadatog grafa pridruženog datom problemu koji se rešava. Lista OPEN sadrži čvorove iz aktuelnog fronta pretrage koji još nisu otkriveni, dok se u listi CLOSED vode čvorovi čiji su naslednici već generisani. Uredjenje čvorova u listi OPEN se vrši na osnovu heurističkih funkcija, koje na prvo mesto stavljaju čvorove koji se sa najvećom verovatnoćom nalaze na putu do rešenja.
- ♦ Generičke neinformativne procedure pretrage su: pretraga u širinu, pretraga u dubinu, pretraga sa ograničenom dubinom, pretraga sa iterativnim produbljavanjem, pretrage uniformne težine i dvosmerne pretrage.
- ♦ U odnosu na sve neinformativne pretrage, pokazuje se da najbolje performanse poseduje pretraga sa iterativnim produbljivanjem, ukoliko je prostor pretrage velik, a unapred nije poznata dubina najplićeg rešenja.

## Pitanja i zadaci

1. Opišite svojim rečima rad algoritma GRAPHSEARCH.
2. Zašto u ovom algoritmu postoji potreba za postavljanjem ukazatelja i kada se oni redefinišu?
3. Za igru 8 klizećih pločica i izabrano početno (2,8,3;1,6,4;7,■,5) i ciljno stanje (1,2,3;8,■,4;7,6,5) nacrtati rezultujući graf pretrage za algoritam pretrage u širinu.
4. Za problem dat u zadatku 3. nacrtati rezultujući graf pretrage za algoritam pretrage u dubinu.
5. Pokažite da sva stanja u igri sa 8 klizećih pločica podeljena u dva disjunktna skupa, unutar kojih je svako stanje dostižno iz bilo kog drugog stanja iz istog skupa, dok se ni jedno stanje ne može dostići iz bilo kog stanja u drugom skupu.

6. Za graf na slici napisati redosled otkrivanja čvorova algoritmima pretrage
- a) pretraga u širinu
  - b) pretraga u dubinu
  - c) iterativno produbljivanje
- Prilikom generisanja naslednika, uvek prvo razmatrati levog naslednika.



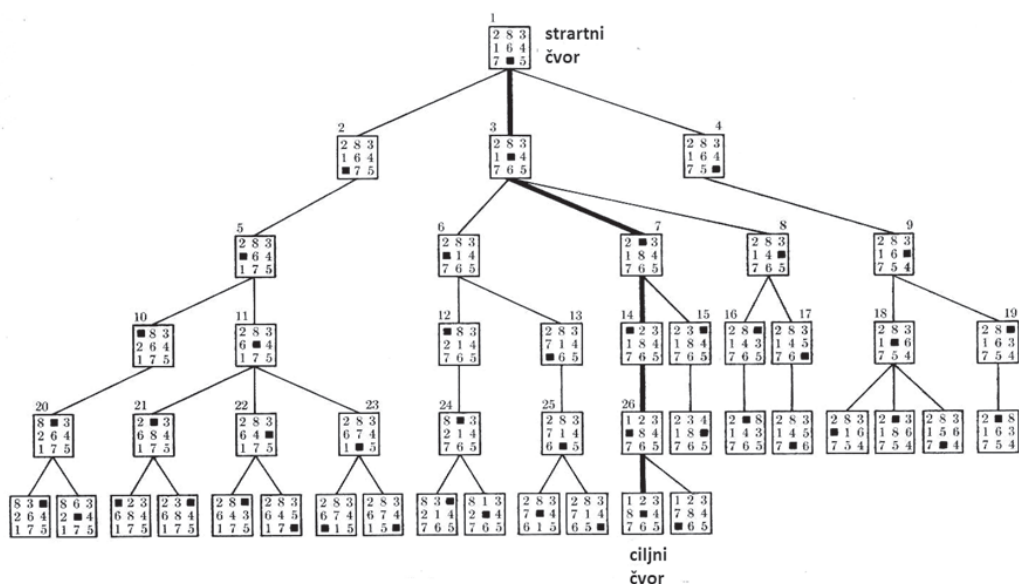
## 6. Informativne pretrage u sistemima veštačke inteligencije

Ukoliko za rešavanje datog problema posedujemo dodatne informacije vezane za prirodu rešavanog problema, a koje nam olakšavaju odluku o tome koji naredni čvor treba otkriti u grafu pretrage, u mogućnosti smo da koristeći opštu strukturu algoritma pretrage na grafu, optimalno iskoristimo ove informacije. U oblasti VI ova klasa informacija se naziva heurističke informacije. Na osnovu njih moguće je olakšati procenu perspektivnosti mogućih alternativnih akcija u toku rešavanja datog problema. Formalno se definišu kao evaluacione (heurističke) funkcije, koje se primenjuju na čvorove u grafu pretrage. Usvojeno je da malim vrednostima odgovaraju perspektivniji čvorovi, što odgovara ideji da procenjuju deo ili ukupnu dužinu puta od datog do ciljnog čvora.

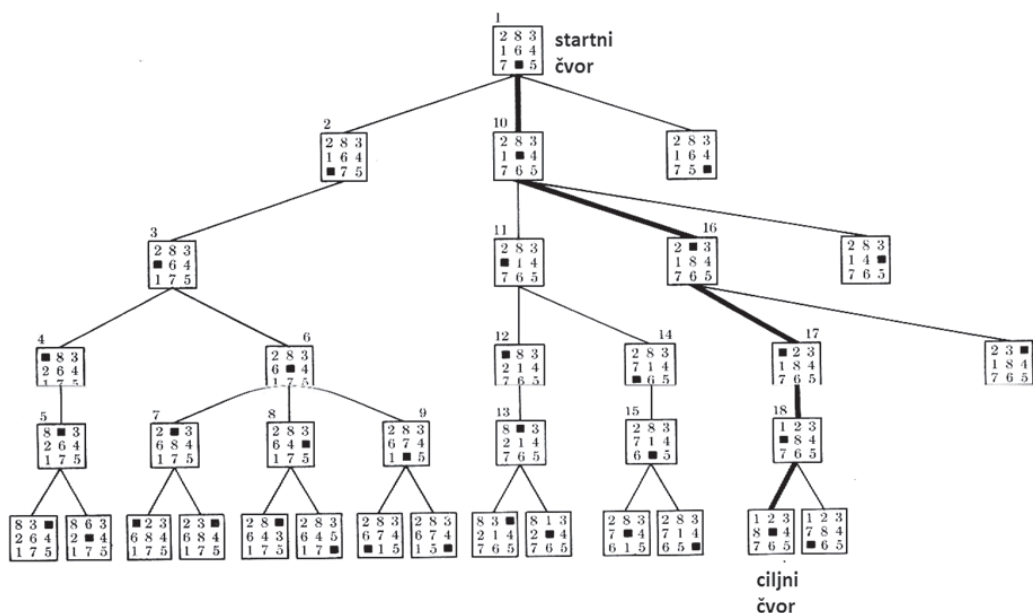
### PRIMER 6.1

Neka je zadat početni raspored (2,8,3;1,6,4;7,■,5) u igri 8 klizećih pločica i neka je ciljna konfiguracija (1,2,3;8,■,4;7,6,5). Prikazati graf pretrage za dve neinformativne strategije pretrage: u širinu i dubinu. Rešenja su data na Sl.6.1 i Sl.6.2 respektivno. Prilikom pretrage u širinu otkriveno je ukupno 26, a prilikom pretrage u dubinu 18 čvorova.





Sl.6.1 Pretraga u širinu prilikom rešavanja problema iz Primera 6.1. Ova neinformativna procedura pretrage nalazi rešenje nakon otkrivanja 26 čvorova. Primetimo da nikakva dodatna informacija o rešavanom problemu nije korišćena u postupku pretrage.

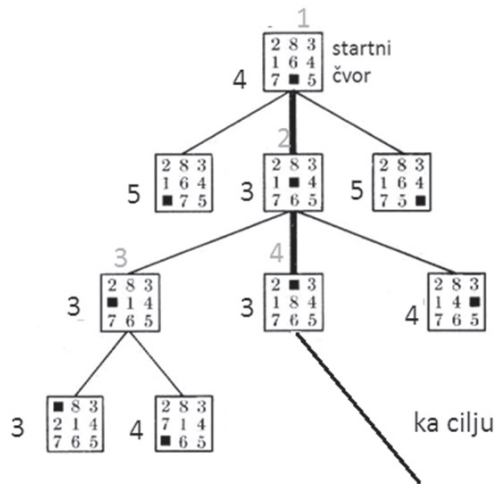


Sl.6.2 Pretraga u dubinu prilikom rešavanja problema iz Primera 6.1. Rešenje je nadjeno nakon otkrivanja 18 čvorova. Kao i kod pretrage u širinu, i kod pretrage u dubinu nije korišćena nikakva dodatna informacija o rešavanom problemu.

Ovi brojevi su dobri pokazatelji efikasnosti rada ovih procedura. U igri 8 klizećih pločica razvijeno je dosta dobrih heuristika. Npr. prilikom procene perspektivnosti nekog čvora, kada se pitamo da li da ga otvorimo ili ne u nekom skupu alternativnih mogućnosti, može nam pomoći broj pločica koje nisu na svom mestu u odnosu na ciljnu konfiguraciju. Dakle, definišimo evaluacionu funkciju sa

$$\hat{f}(n) = \text{broj klizećih pločica koje nisu na svom mestu u odnosu na cilj.}$$

Primetimo da ovako definisana evaluaciona funkcija ima vrednost nula za ciljno stanje i veliku vrednost za stanja koja su značajno udaljena od ciljnog, mereno brojem pločica koje treba pomeriti do poklapanja sa ciljem. Stoga zadovoljava intuitivnu predstavu o neophodnom svojstvu evaluacionih funkcija, kao mera perspektivnosti datog čvora. Što je njena vrednost manja za zadati čvor, on je perspektivniji, budući da je polazeći od njega potrebno napraviti mali broj koraka do cilja, i obrnuto.

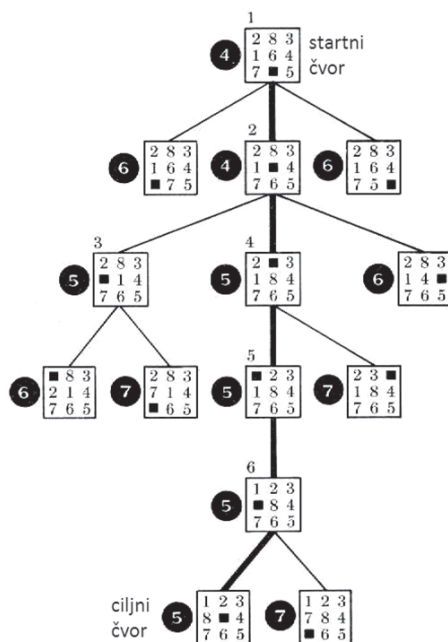


Sl.6.3 Heuristička pretraga sa evaluacionom funkcijom  $\hat{f}(n) = \hat{w}(n)$ , gde je  $\hat{w}(n)$  broj pločica koje se ne nalaze na pravom mestu u odnosu na ciljni čvor. Brojevi sa strane daju vrednosti evaluacione funkcije za dati čvor. Brojevi iznad čvorova daju redosled otvaranja čvorova.

Na Sl.6.3 prikazan je deo grafa pretrage otkriven na osnovu ove evaluacione funkcije. Uočava se da nas evaluaciona funkcija  $\hat{w}(n)$ , može odvesti u predeo čvorova velike dubine, a u kome ne leži ciljni čvor. Intuitivno se nameće da bi plićim čvorovima trebalo dati veću šansu. Ovo se može postići uvodenjem još jednog dodatnog člana, tj

$$\hat{f}(n) = \hat{g}(n) + \hat{w}(n),$$

gde je  $\hat{g}(n)$  dubina čvora  $n$  u grafu pretrage. Graf pretrage za ovu evaluacionu funkciju je dat na Sl.6.4 sa koje vidimo da se cilj postiže otkrivanjem svega 6 čvorova, što je višestruko manji broj od otvorenih čvorova pri pretrazi u širinu i dubinu. Ovaj primer pokazuje heurističku snagu dobro konstruisane evaluacione funkcije, koja znatno skraćuje pretragu.



Sl.6.4 Heuristička pretraga sa evaluacionom funkcijom  $\hat{f}(n) = \hat{g}(n) + \hat{w}(n)$ , gde je  $\hat{g}(n)$  dubina čvora  $n$ , a  $\hat{w}(n)$  broj pločica koje se ne nalaze na pravom mestu u odnosu na ciljni čvor. U zatam-njenim krugovima pored svakog čvora data je vrednost evaluacione funkcije. Brojevi iznad čvorova pokazuju redosled otvaranja čvorova. Uočavamo da algoritam otkriva svega 6 čvorova pre nego što završi rad, što je višestruko efikasnije od neinformativnih algoritama pretrage u širinu i dubinu.

## 6.1 ALGORITAM A i A\*

Neka je  $f(n)$  evaluaciona funkcija sa značenjem minimalne težine puta od pola-znog čvora  $s$  do ciljnog čvora, pod uslovom da put prolazi kroz čvor  $n$ . Neka je  $g(n)$  težina minimalnog puta od startnog čvora  $s$  do čvora  $n$ . Tada je  $f(n)=g(n)+h(n)$  težina minimalnog puta od  $s$  do ciljnog čvora, pod uslovom da prolazi kroz čvor  $n$ . Do iste formule možemo doći i na osnovu principa optimalnosti po kome je optimalan put između dva čvora i u delovima optimalan, Sl.6.5. Ako sa  $d(n, m)$  označimo minimalnu težinu puta između čvorova  $n$  i  $m$ , tada se aditivne komponente evaluacione funkcije mogu zapisati u formi

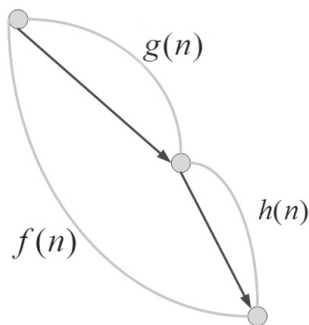
$$h(n)=\min_i\{d(n, t_i)\} \quad (6.1)$$

gde su  $t_i$  ciljni čvorovi, dok je

$$g(n)=d(s,n). \quad (6.2)$$

Očigledno je u startnom čvoru ispunjen uslov

$$f(s)=h(s). \quad (6.3)$$



Sl.6.5 Struktura evaluacione funkcije za A algoritme pretrage na grafu. Optimalan put (najmanje težine) između startnog i ciljnog čvora je jednak zbiru najkraćeg puta od ciljnog čvora do tekućeg čvora  $n$  (komponenta  $g(n)$ ) i najkraćeg puta od  $n$  do cilja (komponenta  $h(n)$ ).

Budući da nam u rešavanju praktičnih problema, istinite vrednosti za  $f$ ,  $g$  i  $h$  nisu poznate, upotrebićemo njihove procene  $\hat{f}, \hat{g}, \hat{h}$ , odnosno

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n), \quad (6.4)$$

pri čemu je uobičajeno da se  $\hat{g}(n)$  naziva faktor dubine, a  $\hat{h}(n)$  heuristička komponenta (faktor) evaluacione funkcije.

Algoritam GRAPHSEARCH u kome se za uređjivanje čvorova u listi OPEN koristi evaluaciona funkcija oblika (6.4) se naziva A algoritam. Istorijski gledano Nils Nilsson je 1968. godine, radeći na projektu robota Shakey razvio prvu ideju algoritma, koga je nazvao A1, a koja je zapravo bilabrža verzija Dijkstra algoritma za nalaženje najkraćih puteva na grafu. Zatim je Bertram Raphael dao značajne izmene koje su rezultovale u tzv. verziji A2. Na osnovu A2, Peter Hart je izvršio minimalne izmene i dao argumentaciju da je A2 najbolji mogući algoritam za nalaženje najkraćih puteva u grafu. U zajedničkom radu iz 1972. god. Hart, Nilson i Raphael su izveli dokaz da je modifikovani A2 optimalan algoritam za nalaženje najkraćih puteva, pod uslovima koje su vrlo precizno formulisali, Sl.6.6.

Procena  $\hat{g}(n)$  može biti tekući najkraći put od početnog čvora do  $n$  u toku izvršavanja algoritma. S obzirom da sve do poslednjeg koraka, u opštem slučaju, postoji mogućnost promene ukazatelja u algoritmu pretrage na grafu (korak 7), očigledno važi

$$\hat{g}(n) \geq g(n). \quad (6.5)$$

Heuristička komponenta  $\hat{h}(n)$  se mora zasnivati na heurističkim znanjima o datom problemu.



Sl.6.6 Nils Nillson (1933 - ), Bertram Raphael (1936 - ), Peter Hart (1940 - ) koautori konačne verzije algoritma A i A\*, koji su imali snažan uticaj na razvoj VI.

## PRIMER 6.2

Pretraga u širinu se može smatrati A algoritmom u kome je  $\hat{g}(n) = d$ , gde je  $d$  dubina čvora u stablu pretrage, dok je heuristička komponenta  $\hat{h}(n) = 0$ . Ukoliko zadržimo  $\hat{g}(n)$  kao procenu najkraćeg puta od startnog čvora do  $n$  i stavimo opet  $\hat{h}(n) = 0$ , dobijamo pretragu sa uniformnim težinama.

Algoritam u kome heuristička funkcija  $\hat{h}(n)$  zadovoljava nejednakost

$$\hat{h}(n) \leq h(n), \quad (6.6)$$

naziva se A\* algoritam. Ovakva heuristička funkcija se naziva **dopustiva**.

## PRIMER 6.3

Kao što smo videli u prethodnom primeru, stavljanjem  $\hat{h}(n) = 0$  i  $\hat{g}(n) = d$ , dobijamo algoritam pretrage u širinu, koji uvek nalazi optimalno rešenje ukoliko ono postoji. Kako je za ovu nultu heurističku funkciju uvek u važnosti nejednakost

$$h(n) \geq \hat{h}(n) = 0 \quad (6.7)$$

zaključujemo da je algoritam pretrage u širinu ujedno i A\* algoritam.

Ukoliko stavimo da je  $\hat{f}(n) = \hat{h}(n)$ , rezultujući algoritam se naziva **gramzivo pretraživanje (greedy search, greedy best-first search)**. Evaluacija svih čvorova se vrši samo na osnovu heurističke komponente evaluacione funkcije. Pokazuje se da ovaj algoritam nije kompletan čak i u konačnim prostorima stanja, slično kao i pretraga u dubinu. Vremenska i prostorna složenost algoritma je  $O(b^m)$ , gde je  $m$  maksimalna dubina grafa pretraživanja. Praksa pokazuje da se sa dobrim heurističkim funkcijama, kompleksnost pretrage može znatno smanjiti, zavisno od problemskog domena i izabrane heurističke funkcije.

## 6.2 SVOJSTVA A\* ALGORITMA

### Dopustivost (admissibility)

Algoritam pretrage je dopustiv ukoliko za svaki graf pretrage završava rad i nalazi optimalni put od početnog do ciljnog čvora, pod uslovom da taj put postoji.

---

#### Svojstvo 1

Algoritam A\* je dopustiv.

---

### Poredjenje A\* algoritama

Preciznost heuristike  $\hat{h}$  zavisi od količine ugrađenog heurističkog znanja o problemskom domenu. Neka su date dve verzije algoritam A\*:

$$A_1: \hat{f}_1(n) = \hat{g}_1(n) + \hat{h}_1(n),$$

$$A_2: \hat{f}_2(n) = \hat{g}_2(n) + \hat{h}_2(n).$$

Kažemo da je A<sub>2</sub> informativniji od A<sub>1</sub> ukoliko za sve neciljne čvorove n, važi

$$\hat{h}_2(n) > \hat{h}_1(n). \quad (6.8)$$

---

#### Svojstvo 2

Neka su A<sub>1</sub> i A<sub>2</sub> dve varijante algoritma A\*, takve da je A<sub>2</sub> informativnije od A<sub>1</sub>. Tada nakon okončanja pretrage na bilo kom grafu koji sadrži put od s do ciljnog čvora, svaki čvor otkriven algoritmom A<sub>2</sub> je otkriven i algoritmom A<sub>1</sub>. Stoga A<sub>1</sub> otkriva najmanje onoliko čvorova koliko otkriva i A<sub>2</sub>.

---

### PRIMER 6.4

Vratimo se na problem 8 klizećih kockica iz Primera 6.1. Neka su varijante A<sub>1</sub> i A<sub>2</sub> date sa:

$$A_1: \hat{h}_1(n) = w(n) \text{ gde je } w(n) \text{ broj nepoklapajućih pozicija tekuće i ciljne pozicije}$$

$$A_2: \hat{h}_2(n) = 0.$$

A<sub>1</sub> pripada klasi A\*, budući da je broj nepoklapajućih pozicija uvek manji ili jednak broju poteza koji vode do rešenja, dakle uvek je  $w(n) \leq f(n)$ . Već smo utvrdili na osnovu (6.7) da je A<sub>2</sub> algoritam pretraga u širinu i da takodje pripada klasi A\*. Na osnovu definicije, pošto je  $\hat{h}_1(n) \geq \hat{h}_2(n)$ , zaključujemo da je algoritam A<sub>1</sub> informativniji od A<sub>2</sub>, odnosno da će A<sub>1</sub> otkriti manji broj čvorova od A<sub>2</sub>. U Primeru 6.1 smo videli da A<sub>2</sub> otkriva 26 čvorova, dok A<sub>1</sub> otkriva 6 čvorova, što je u potpunosti saglasnosti sa intuitivnim pojmom stepena informativnosti i svojstva 2 algoritama A\*.

### Monotono ograničenje

Heuristička funkcija  $h(n)$  zadovoljava uslov monotonosti ukoliko za sve čvorove  $n_i$  i  $n_j$ , takve da je  $n_j$  naslednik čvora  $n_i$ , u važnosti je nejednakost

$$\ddot{h}(n_i) - \ddot{h}(n_j) \leq d(n_i, n_j), \quad h(t) = 0.$$

---

#### Svojstvo 3

Ako je zadovoljen uslov monotonosti, tada za svaki čvor  $n$  izabran za otkrivanje, važi  $\hat{g}(n) = g(n)$ .

---

Iz svojstva 3 sledi da  $A^*$  algoritam sa heurističkom funkcijom koja zadovoljava uslov monotonosti bira optimalan put do proizvoljnog čvora izabranog za otkrivanje. Drugim rečima, kada algoritam izabere čvor  $n$  za otvaranje, do njega je već pronađen optimalan put. Praktična posledica ovog svojstva je da tada nije potrebna dopunska korekcija stabla pretrage  $Tr$ , odnosno nema potrebe da se proverava da li su tek otvoreni čvorovi već bili prisutni u listi  $CLOSED$ . Stoga, nije potrebno proveravati da li se menjaju nasledni odnosi medju čvorovima na tekućem stablu pretrage fiksirani ukazateljima (korak 7 algoritma  $GRAPHSEARCH$ ).

---

#### Svojstvo 4

Ako su zadovoljeni uslovi monotonosti za heurističku funkciju, vrenosti evaluacione funkcije  $\hat{f}(n)$  ne opadaju na nizu čvorova otkrivenih algoritmom  $A^*$ .

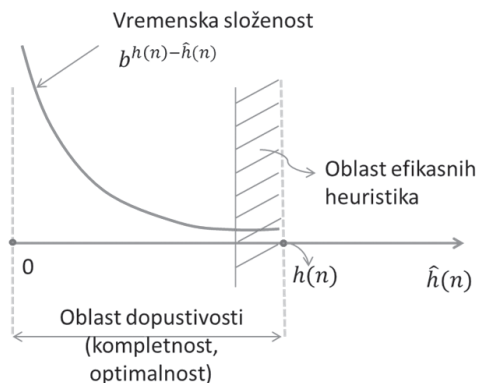
---

Mnoge heurističke funkcije zadovoljavaju uslov monotonosti, kao što je heuristička funkcija  $w(n)$  iz Primera 6.1.

## 6.3 HEURISTIČKA SNAGA EVALUACIONIH FUNKCIJA

Izbor heurističkih funkcija je od ključnog značaja prilikom određivanja heurističke snage algoritama pretrage iz klase  $A$ . Izborom  $\hat{h}(n) = 0$ , obezbedjena je dopustivost (kompletnost i optimalnost), ali je zato efikasnost rezultujuće pretrage u širinu mala. Stavljanjem da je  $\hat{h}(n)$  što bliže najvećoj mogućoj donjoj granici za  $h(n)$ , rezultovaće u algoritmu koji otvara najmanji mogući broj čvorova uz zadržavanje svojstva dopustivosti, Sl.6.7.





Sl.6.7 Slika prikazuje opsege i rezultujuću efikasnost algoritama iz klase A u zavisnosti od heurističke snage evaluacione funkcije. Prikazana je i oblast u kojoj se nalaze efikasne heurističke funkcije, za koje je garantovana dopustivost. Preko granice dopustivosti nalaze se heurističke funkcije koje mogu biti vrlo efikasne, ali je zato žrtvovana dopustivost, odnosno ne postoji garancija da će se sa takvim heurističkim funkcijama pronaći optimalno rešenje ukoliko ono postoji.

#### PRIMER 6.5

Heuristička funkcija  $w(n)$  je dopustiva, ali ne predstavlja naročiti dobru procenu težine pozicije, odnosno udaljenost od ciljne pozicije. Menhetn rastojanje  $P(n)$  dato sa

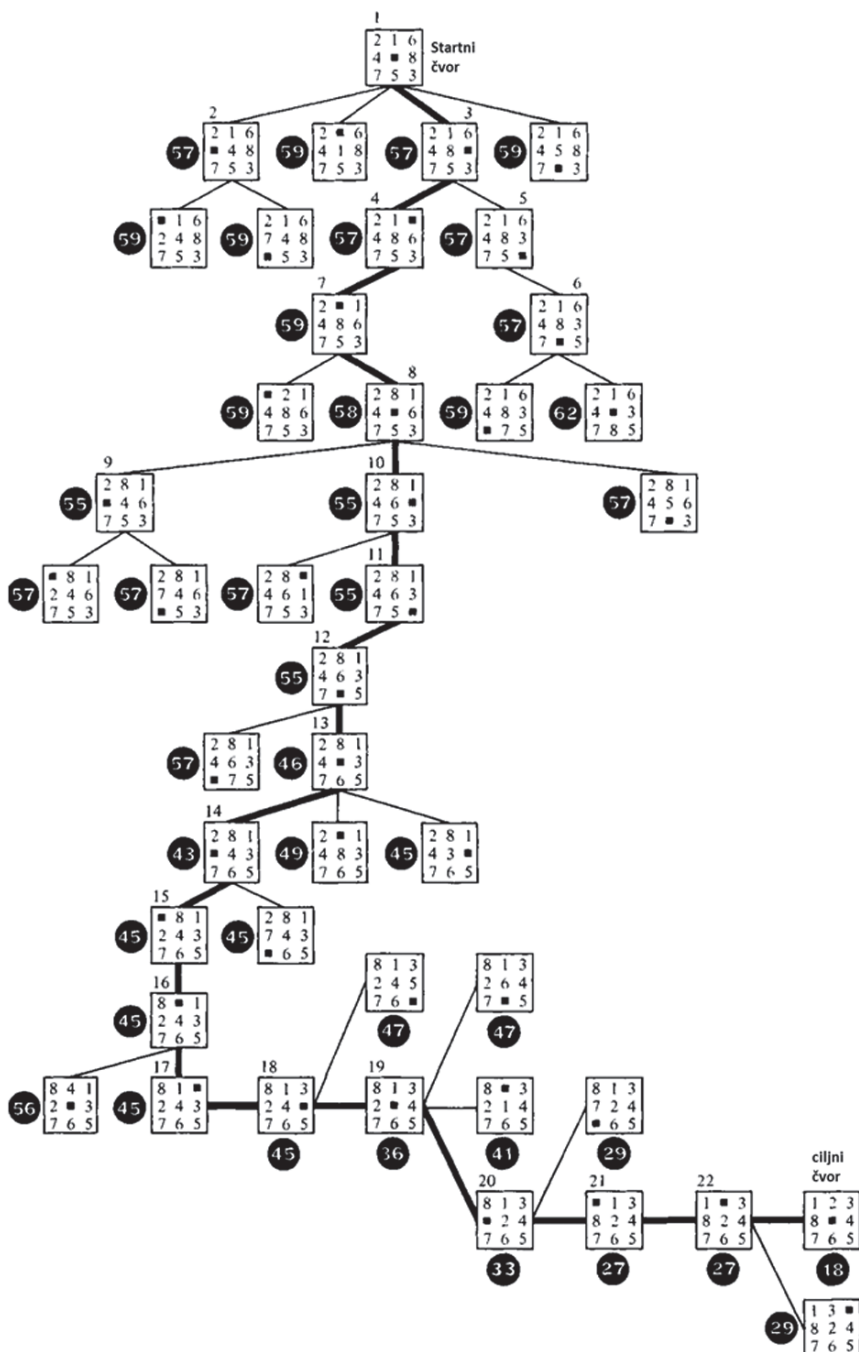
$$P(n) = \sum_{i=1}^8 |x_i(n) - x_i(t)| + |y_i(n) - y_i(t)|, \quad (6.9)$$

gde su  $(x_i(n), y_i(n))$  koordinate tekuće pozicija pločice, a  $(x_i(t), y_i(t))$  koordinate iste te pločice u ciljnoj poziciji, se pokazuje kao znatno bolja dopustiva heuristička funkcija.

Često se heuristička snaga dobija na račun žrtvovanja dopustivosti, odnosno izborom heurističke funkcije koja nije donja granica za  $h(n)$ . Pomoću ovakvih algoritama se mogu rešavati kompleksni problemi, ne garantujući optimalnost. Npr. za problem 8 klizećih pločica, heuristika koja se pokazala prilično dobro, a koja nije dopustiva, je kombinacija tzv. sekvencijalnog skora  $S(n)$  i Menhetn rastojanja

$$\hat{h}(n) = P(n) + 3 S(n), \quad (6.10)$$

pri čemu se sekvencijalni skor računa za svaku necentralnu pločicu dodavanjem 2, ako sledeća pločica nije na svom mestu, 0 ako jeste, dok se za centralnu pločicu dodaje 1. Na Sl.6.8 dat je graf pretrage za ovu heurističku funkciju i problem transformacije početne pozicije (2,1,6;4,8;7,5,3) u (1,2,3;8,4;7,6,5). Primećuje se svojstvo fokusiranosti ove heuristike ka cilju, budući da je algoritam otvorio relativno malo čvorova van direktnog puta od startnog do ciljnog čvora. Podebljane grane na Sl.6.8 prikazuju ovaj put. U ovom primeru, pronađeni put do cilja je istovremeno i optimalan - minimalne dužine 18, ako se uzme da je svaka grana u grafu pretrage jedinične težine. Napomenimo da usled žrtvovanja dopustivosti na račun efikasnosti, algoritam teorijski ne garantuje nalaženje ovog rešenja.



Sl.6.8 Graf pretrage za problem iz Primera 6.5. Primenjena heuristička funkcija 6.10 nije dopustiva, ali poseduje solidnu heurističku snagu, koja se vidi na grafu pretrage po tome što ima jako malo otvorenih čvorova koji nisu na direktnom putu od startne do ciljne konfiguracije. Kao i u prethodnim primerima, brojevi u crnim kružićima su vrednosti heurističke funkcije za dati čvor, dok broj iznad čvora predstavlja redne brojeve otvaranja čvorova u toku rada algoritma.

Heurističku snagu algoritma A možemo dobiti i množenjem izabrane heurističke funkcije nekom konstantom K, koja je veća od jedinice

$$\hat{f}(n) = \hat{g}(n) + K \cdot \hat{h}(n) . \quad (6.11)$$

Za malo K dobijamo pretrage koje podsećaju na pretrage u širinu, dok za veliko K dobijamo pretrage u kojima dominira heuristika. Praksa pokazuje da se na efikasnosti dobija ukoliko K nije konstantno u toku rada algoritma, već zavisi u obrnutoj srazmeri sa dubinom čvorova u grafu pretrage, odnosno

$$K \sim \frac{1}{d(n)} \quad (6.12)$$

gde je  $d(n)$  dubina čvora n u grafu pretrage. U početku rada algoritma, heuristička komponenta ima veliku vrednost, budući da na osnovu (6.12) K ima veliku vrednost. Kako proces rada algoritma napreduje, K postaje sve manje i heuristička komponenta evaluacione funkcije postaje zanemarljiva, omogućavajući dominaciju pretrage u širinu. Ova strategija podseća na zdravorazumsku strategiju da na početku rešavanja nekog kompleksnog problema, na osnovu intuicije usmeravamo našu pažnju na deo podprostora rešenja koji nam se čini najizglednijim, a zatim sistematično pretražujemo taj podprostor.

Iako je  $A^*$  pretraga kompletna, optimalna i efikasna, ona nije uvek najbolje rešenje jer broj čvorova po konturama pretraživanog prostora raste eksponencijalno na putu ka cilju. Takođe postoji i problem memorijskog prostora jer ovi algoritmi čuvaju u memoriju sve otkrivene čvorove. Zbog toga se oni ne koriste za velike probleme. Noviji algoritmi pokušavaju i uspevaju da donekle prevaziđu taj problem. Radi ilustracije navešćemo samo jedan primer, tzv.  $IDA^*$  algoritam.

**$IDA^*$  - (iterative-deepening  $A^*$ algorithm)** koristi princip iterativnog produblivanja. Kao granica odsecanja koristi se vrednost evaluacione funkcije, a ne dubina čvorova kao u standardnim neinformativnim algoritmima sa iterativnim produblivanjem. U svakoj iteraciji produblivanja vrednost odsecanja je najmanja vrednost evaluacione funkcije medju svim čvorovima koji su u prethodnoj iteraciji bili van granice odsecanja.

U literaturi se mogu naći i druge varijante, kao što su **rekurzivni prvi najbolji** (Recursive Best-First Search), **memorijski ograničeni  $A^*$** , **pojednostavljeni memorijski ograničeni  $A^*$** , i td.

## 6.4 GENERISANJE DOPUSTIVIH HEURISTIČKIH FUNKCIJA

Jedan od opštih principa generisanja dopustivih heuristika je zamenjivanje originalnog problema sa problemom koji ima manje ograničenja. Takav problem sa manjim brojem restrikcija se naziva **relaksirani problem**. Težina optimalnog rešenja relaksiranog

problema može biti manja od optimalnog rešenja originalnog problema budući da je nametnut manji broj ograničenja. Stoga je težina optimalnog rešenja relaksiranog problema ujedno i dopustiva heuristika za originalni problem. Kako je izvedena heuristika tačna težina za relaksirani problem ona zadovoljava i uslov monotonosti.

Relaksirani problemi se mogu automatski generisati iz formalnog opisa originalnog problema.

## PRIMER 6.6

Demonstrirajmo na primeru igre 8 klizećih pločica automatsko generisanje relaksiranih problema.

Originalni problem: Pločica može da se pomeri iz A u B ako je A horizontalno ili vertikalno susedno polju B i polje B je prazno.

Relaksirani problem: Pločica može da se premesti iz A u B:

$A \rightarrow B$ , ako su A i B susedni

$A \rightarrow B$ , ako je polje B prazno

$A \rightarrow B$ , bezuslovno.

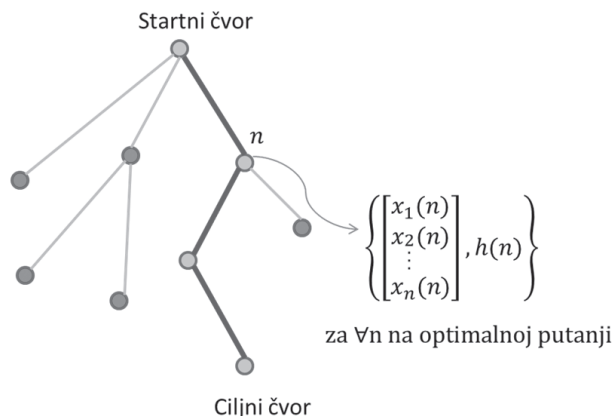
Iz varijante a) može da se izvede heuristika  $P(n)$  – Menhetn rastojanje (6.9). Iz varijante b) može da se izvede tzv. Gašingova heuristika, dok se iz varijante a) može izvesti heuristika  $w(n)$  – broj pločica koje nisu na svom mestu.

Program pod nazivom ABSOLVER može automatski da generiše heuristike na osnovu kombinacije različitih tehnika uključujući i tehniku relaksiranih problema. Za igru 8 klizećih pločica ovaj program je generisao novu do tada nepoznatu heuristiku. Napomenimo da ukoliko za jedan problem imamo na raspolaganju čitav skup heuristika  $\{h_1, h_2, \dots, h_n\}$  a ni jedna ne dominira izrazito po efikasnosti u odnosu na druge, možemo ih sve koristiti tako što će se za svaki čvor birati neka od njih na osnovu pravila

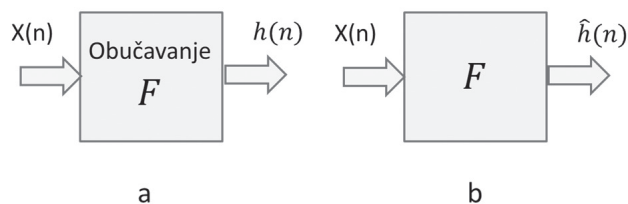
$$h(n) = \max_i \{h_1(n), h_2(n), \dots, h_n(n)\}.$$

Lako se pokazuje da je ovakva heuristika dopustiva, kada su njene komponente dopustive, monotona i efikasnija (preciznija) od pojedinačnih.

Heurističku funkciju možemo i automatski naučiti koristeći metode mašinskog učenja. Svako optimalno rešenje predstavlja jedan primer iz koga se može naučiti kako da formiramo  $h(n)$ . Svaki primer sastoji se od stanja koja se nalaze na putu koji vodi do rešenja i težine tog puta. Iz ovih primera algoritmom induktivnog učenja možemo naučiti funkciju  $h(n)$  koja će predvideti težinu rešenja za svako stanje koje se javlja duž puta, Sl.6.9 i Sl.6.10.



Sl.6.9 Način formiranja obučavajućih skupova u postupku automatskog učenja heuristika. Prethodno se odabere prostor obeležja  $X$ , kao vektorski prostor u kome svaka komponenta ima značenje odgovarajući odabраних аtribute описа stanja за задати problem. Nakon formiranja skupa rešenih problema, obučavajući skup se formira tako što se duž optimalnog puta od starta do cilja, за svaki čvor računa opis i odgovarajuća težina puta do cilja, što je defakto  $h(n)$ .



Sl.6.10 Sistem за obučavanje restauriše preslikavanje  $F: X(n) \rightarrow h(n)$ , (a). Preslikavanje  $F$  је zapravo novonaučena heuristička funkcija, koja за buduće rešavane probleme, за svaki čvor koji se otkriva из njegovog описа  $X$ , računa  $\hat{h}(n)$ , (b).

## 6.5 MERE EFIKASNOSTI PRETRAGA

Prilikom ocenjivanja efikasnosti rada nekog algoritma pretrage i poredjenja sa drugim algoritmima neophodno je uvesti neke kvantitativne pokazatelje. Navešćemo samo dve mere koje se često koriste u sistemima VI: usmerenost i efektivni faktor grananja.

### Usmerenost P

Usmerenost pretrage  $P$  se definiše odnosom

$$P = \frac{L}{T}, \quad (6.13)$$

gde je  $L$  težina nadjenog puta od starta do ciljне pozicije, dok je  $T$  ukupan broj čvorova formiranih u toku procesa pretrage, uključujući ciljni čvor a ne računajući startni čvor. Ukoliko bi operator otkrivanja čvorova bio tako efikasan da generiše samo

naslednike koji se nalaze na optimalnom putu ka rešenju, P dostiže svoju maksimalnu vrednost jednaku 1. Neinformativne pretrage karakteriše vrednost ovog pokazatelja, znatno veća od 1. Opisno govoreći, faktor usmerenosti meri koliko je konkretno stablo pretrage „razgranato“. Nedostatak ove mere je njena zavisnost od težine nadjenog puta.

### Efektivni faktor grananja B

B je konstantan faktor grananja stabla pretrage čija je dubina jednaka dužini optimalnog puta od starta do cilja i čiji je ukupan broj čvorova identičan broju čvorova posmatranog grafa pretrage. Dobro svojstvo je da B ne zavisi u velikoj meri od dužine puta od starta do cilja. Možemo ga dovesti u vezu sa veličinama L i T sledećim izrazima

$$B^1 + B^2 + \dots + B^L = T, \quad T = \frac{B}{B-1} (B^L - 1). \quad (6.14)$$

Vrednosti B bliske 1 odgovaraju izuzetno fokusiranoj pretrazi, koja otvara malo nepotrebnih čvorova. S druge strane, veliko B odgovara pretrazi sa vrlo razgranatim stablom, u kome je otvoreno znatno više čvorova nego što ih ima na optimalnoj putanji od starta do ciljnog čvora.

## Rezime 6. poglavlja

- ♦ Informativni A algoritmi pretrage imaju evaluacionu funkciju u aditivnoj formi sa značenjem najkraćeg puta (puta najmanje težine) od startnog do ciljnog čvora, pod uslovom da put prolazi preko čvora čija se evaluacija računa.
- ♦ Komponenta evaluacione funkcije od posmatranog čvora do cilja naziva se heuristička funkcija. Algoritmi kod kojih je ova komponenta uvek manja od istinite vrednosti najkraćeg puta od datog čvora do cilja, nazivaju se A\* algoritmi, a sama heuristika – dopustiva heuristika.
- ♦ Algoritmi A\* imaju niz poželjnih osobina: dopustivost (kompletnost i optimalnost), i strukturnu mogućnost kontrole efikasnosti preko heurističke komponente. Vremenska i prostorna složenost ovih algoritama zavisi eksponencijalno od odstupanja upotrebljene heurističke funkcije od njene istinite vrednosti, tj. od veličine  $\hat{h}(n) - h(n)$ .
- ♦ Ukoliko heuristička funkcija zadovoljava nejednakost trougla (osobina monotonosti) A\* algoritmi generišu uvek stablo pretrage umesto opštijeg grafa pretrage, čime se znatno uprošćava realizacija algoritma.
- ♦ Heurističke funkcije je moguće konstruisati na osnovu reformulisanog originalnog problema kroz slabljenje postavljenih restrikcija, tzv. relaksirani problem. Napredne metode podrazumevaju automatsko mašinsko učenje heuristika na osnovu velikog broja rešenih primera i odgovarajuće predstave čvorova pretrage na putu rešenja u vektorskim prostorima atributa.
- ♦ Uobičajene mere efikasnosti ovih algoritama su usmerenost P i faktor efektivnog grananja B.

## Pitanja i zadaci

1. U problemu 4 kraljice, potrebno je postaviti četiri kraljice na šahovsko polje  $4 \times 4$ , tako da se međusobno ne napadaju. Neka je problem rešavanja ovog problema formulisan sledećim produkcionim sistemom: Početni čvor je prazna  $4 \times 4$  tabla, operator otvaranja kreira novu  $4 \times 4$  tablu koja sadrži jednu kraljicu više postavljenu na legalan način bilo gde na tabli; uslov završetka: ako i samo ako na tabli postoje četiri kraljice koje se međusobno ne napadaju.
  - a. Osmisliti dopustivu heuristiku na osnovu broja kraljica preostalih do ciljnog stanja.
  - b. Na osnovu ove heuristike, pokrenuti  $A^*$  algoritam i nacrtati njegovo stablo pretrage uz označavanje pored svakog čvora vrednosti  $g(n)$  i  $\hat{h}(n)$ .
2. Algoritam  $A^*$  se ne završava dok se ciljni čvor ne izabere za otkrivanje. Međutim put do ciljnog čvora je često otkriven znatno ranije u odnosu na trenutak kada će biti izabran za otkrivanje. Zašto algoritam ne završava rad odmah čim generiše ciljni čvor? Ilustrujte vaš odgovor nekim upečatljivim primerom.
3. Pokazati da dopustiva verzija nekog  $A^*$  algoritma ostaje i dalje dopustiva, ukoliko iz liste OPEN isključimo sve čvorove  $n$ , za koje je  $\hat{f}(n) > F$ , gde je  $F$  gornja granica za  $f(n_0)$ , gde je  $n_0$  startni čvor.
4. Napišite program koji će kao ulaz prihvatati dva URL veb stranica i pronalaziti putanju veza između njih. Koja je strategija pretraživanja odgovarajuća?
5. Neka je evaluaciona funkcija jednog  $A$  algoritma data sa
$$f(n) = (2 - w) \cdot g(n) + w \cdot h(n).$$
  - a. Za koje vrednosti  $w$  je ovaj algoritam kompletan?
  - b. Za koje vrednosti je optimalan pretpostavljajući da je  $h$  dopustiva?
  - c. Koja vrsta pretraživanja se dobija ako se stavi da je
    - i)  $w=0$ ,
    - ii)  $w=1$ ,
    - iii)  $w=2$ .
6. Dokažite da svaka monotona heuristika mora biti dopustiva. Konstruišite dopustivu heuristiku koja nije monotona.
7. Kako bi se mogla naučiti objedinjujuća heuristička funkcija na osnovu parcijalnih heurističkih funkcija metodama mašinskog učenja. Razviti program za problem 8 klizećih pločica i izvršiti eksperimentalnu evaluaciju objedinjene heuristike kada su parcijalne heuristike  $w(n)$  i  $P(n)$ .
8. Komentarišite iskaz da je pretraživanje sa uniformnim težinama specijalan slučaj  $A^*$  pretraživanja.





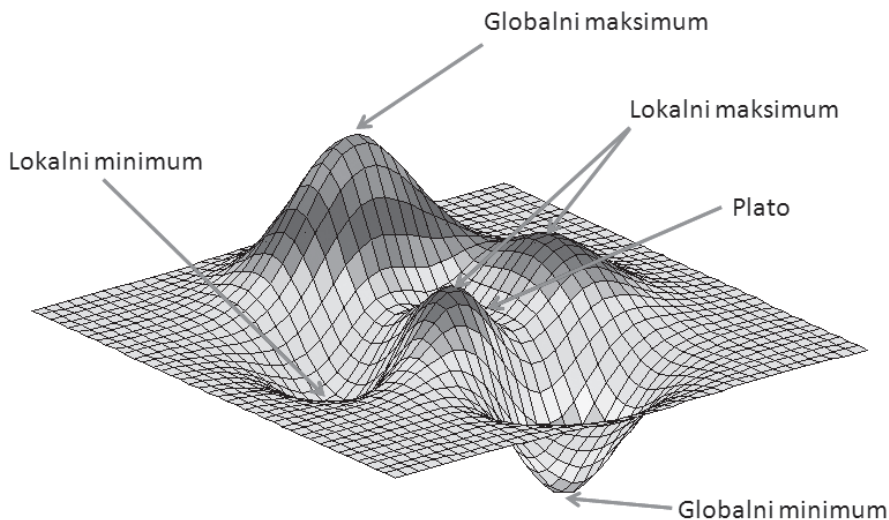
## 7. Alternativni sistemi pretrage

U ovom poglavlju obradićemo grupe algoritama pretrage koji se razlikuju u odnosu na zahteve nametnute u prethodnim poglavljima. Naime, algoritmi pretraživanja koje smo da sada upoznali su dizajnirani tako da sistematično istražuju prostor stanja, što zahteva čuvanje do tada isprobanih sekvenci poteza (akcija, pravila). Kada se dostigne ciljno stanje, put od početnog do ciljnog stanja konstituise rešenje datog problema. Međutim u mnogim problemima put ka cilju nije od značaja. Npr. u problemu 8 kraljica, nije važan put do rešenja, već samo ciljna konfiguracija u kojoj se 8 kraljica rasporedjenih na 8 x 8 šahovskoj tabli međusobno ne napadaju. Za ovakve probleme se koristi drugačija vrsta algoritama - **algoritmi lokalne pretrage**. Oni rade tako što vode računa samo o jednom čvoru - tekućem stanju i kreću se samo ka susedima tog stanja. Iako nisu sistematični, ova klasa algoritama ima dve prednosti:

- ♦ koriste malo memorije, po pravilu fiksnu malu vrednost
- ♦ često pronalaze prihvatljiva rešenja u velikim ili čak beskonačnim prostorima pretrage u kojima sistematični algoritmi nisu od pomoći usled enormne kompleksnosti.

Algoritmi lokalnog pretraživanja su podesni i za rešavanje problema optimizacije, u kojima je cilj naći najbolje stanje u skladu sa nekom kriterijumskom funkcijom. Npr. zamislimo nad dvodimenzionim prostorom stanja kriterijumsku funkciju kao na Sl.7.1, i neka većim vrednostima kriterijumske funkcije odgovaraju poželjnija stanja. U ovom slučaju optimalno rešenje odgovara stanju za koji se postiže apsolutni maksimum kriterijumske funkcije. Kao što je na Sl.7.1 ilustrovano, kriterijumska funkcija poseduje jedan globalni i više lokalnih maksimuma.

Podsetimo se svojstava kompletnosti i optimalnosti. Kompletan algoritam lokalne pretrage uvek pronalazi rešenje ako ono postoji, dok optimalni algoritam uvek pronalazi globalni minimum/ maksimum. Pretvaranje minimizacije u maksimizaciju i obrnuto, postiže se jednostavno dodavanjem znaka minus ispred kriterijumske funkcije.



Sl.7.1 Primer optimizacionog problema u dvodimenzionom prostoru stanja. Kriterijumska funkcija ima više lokalnih ekstremuma. Stoga nalaženje globalnog ekstremuma lokalnom pretragom ne garantuje uspeh, osim ako se unapred zna da je dati algoritam lokalne pretrage optimalan.

## 7.1 PRETRAŽIVANJE USPONOM

Medju pretragama koje su pogodne za optimizacione procedure spada pretraga najstrmijeg uspona (Hill Climbing – HC). Pretpostavljamo u ovom poglavlju da nam je cilj maksimizacija, a kao što smo već napomenuli, principijelno se ne razlikuje od minimizacije, do koje dolazimo samo promenom znaka kriterijumske funkcije.

### HILLCLIMB

1. Za tekući čvor  $n$  uzeti slučajno izabrani čvor  $n_0$
2. Generisati sve naslednike čvora  $n$  i izabrati naslednika  $n_b$  čija je vrednost  $v(n_b) = v_b$  najveća medju svim naslednicima
3. Ako je  $v_b < v(n)$ , završiti pretragu sa rešenjem  $n$
4. U suprotnom staviti da je  $n = n_b$ , a zatim ići na korak 2.

Sl.7.2 Algoritam najstrmijeg uspona (HC) na osnovu kog se pretraga vrši tako što se kretanje obavlja iz jednog u drugo susedno stanje, koje ima najveći pozitivnu promenu kriterijumske funkcije. Pretraga se završava tako što ni jedno susedno stanje nema veću vrednost kriterijumske funkcije u odnosu na tekuće – čime je dostignuta tačka lokalnog maksimuma.

Algoritam ovakvog tipa pretrage je petlja koja se stalno pomera u pravcu rastućih vrednosti, odnosno, slikovito rečeno uzbrdo, Sl.7.2. Završava se kada dostigne vrh gde ni jedan sused nema veću vrednost. Svaki čvor sadrži samo informaciju o korespondentnom stanju i vrednost kriterijumske funkcije za to stanje.

HC se ponekada naziva i gramzivo lokalno pretraživanje (greedy local search) jer grabi ka dobrom susednom stanju bez razmišljanja šta dalje. Ovaj algoritam često dospeva u ćorsokak i to iz sledećih razloga:

- ♦ **Lokalni maksimum** – vrh koji je viši od svih susednih ali ipak niži od globalnog maksimuma, Sl.7.1,
- ♦ **Greben** - sekvenca lokalnih maksimuma među kojima se HC algoritmi teško kreću,
- ♦ **Plato**-deo prostora stanja čija je kriterijumska funkcija konstantna. Ona pri tome može biti ravan lokalni maksimum odakle se ne može nigde naviše, ili stepenik sa koga je moguće napredovati dalje.

U slučaju platoa pogodno rešenje bi bilo da se omogući pomeranje u stranu (ne samo uzbrdo) kako bi se proverilo da li je moguć napredak naviše. Pri tome treba postaviti ograničenje u broju tih koraka kako algoritam ne bi ušao u beskonačnu petlju u slučaju neograničenog platoa.

Postoji nekoliko varijacija HC algoritma:

- ♦ **Stohastički HC** koji na slučajan način bira korake među onima koji vode naviše. Može se kombinovati sa verovatnoćom izbora koja je u vezi sa strminom poteza;
- ♦ **Uspom prvog izbora** – (First choice hill climbing) na slučajan način generiše naslednike sve dok se ne generiše prvi koji je bolji od tekućeg stanja. Može biti dobra strategija kada trenutno stanje ima veliki broj naslednika.
- ♦ **Uspom sa slučajnim ponovnim kretanjem** (Random-restart hill climbing) funkcioniše tako što se HC procedura startuje više puta sa različitim slučajno izabranim početnim stanjima, sve dok se ne dostigne cilj. Interesantno je da ova strategija teži kompletnoj pretrazi sa verovatnoćom 1, jer će jednom kao početno stanje izabrati ciljno stanje. Ako svaka pretraga ima verovatnoću uspeha  $p$ , onda je očekivani broj pokušaja  $1/p$ . Ovakvi algoritmi jako brzo rešavaju problem ukoliko je reljef koji formira kriterijumska funkcija nad prostorom stanja plato sa manjim brojem lokalnih maksimuma. Realni problemi ipak imaju mnogo više lokalnih maksimuma. Npr. NP teški problemi po pravilu imaju eksponencijalni broj lokalnih maksimuma.

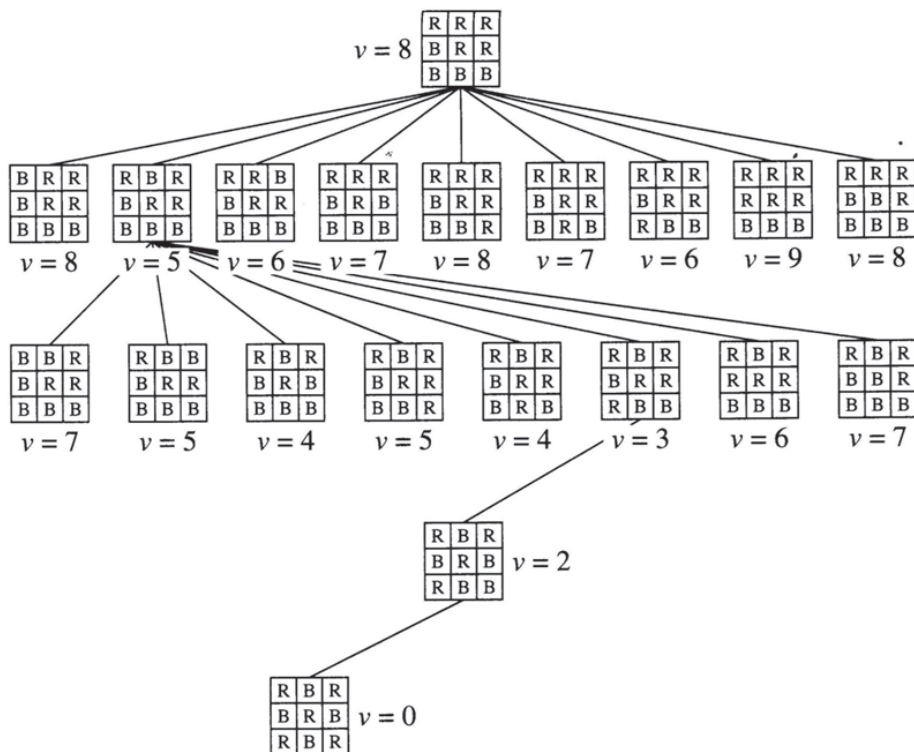
## PRIMER 7.1

Razmotrimo problem bojenja polja u jednoj matrici  $3 \times 3$ . Neka je dat početni proizvoljni raspored bojenja pomoću dve boje: plave (B) i crvene (R). Cilj je da se nadje takvo bojenje u kome će postojati minimalan broj susednih polja iste boje. Formalno, neka je:

$$v(n) = \text{broj parova susednih polja u čvoru } n \text{ koji imaju istu boju.}$$

Rešenje problema je nalaženje ciljnog čvora  $n^*$  u prostoru stanja, takvog da je  $v(n^*) \leq v(n)$ , za svako  $n$ .

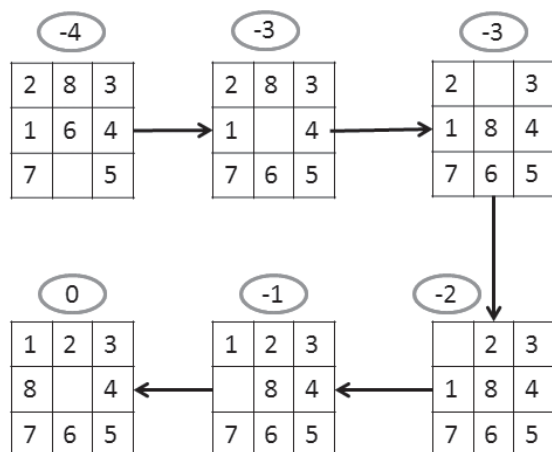
Primenljiva pravila za svaki čvor su ona koja menjaju boju svake ćelije sa B u R i obrnuto. Na Sl.7.3 prikazan je deo grafa pretrage nastao primenom algoritma HC, odnosno u ovom slučaju algoritma najstrmijeg spuštanja, s obzirom da se traži minimum kriterijuma  $v(n)$ .



Sl.7.3 Deo grafa pretrage za problem bojenja matrice 3x3 sa dve boje: R i B iz Primera 7.1. Primjenjen je algoritam najstrmijeg spusta, budući da se traži minimizacija funkcije  $v(n)$ .

## PRIMER 7.2

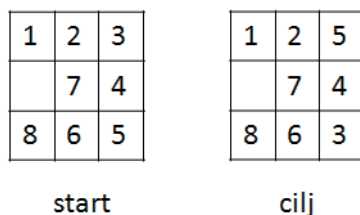
Neka je u igri 8 klizećih pločica početno stanje (2,8,3;1,6,4;7,■,5), a ciljno (1,2,3;8,■,4;7,6,5). Ako za kriterijumsku funkciju odaberemo broj pločica koje nisu na svom mestu i primenimo HC algoritam, dobija se graf pretrage prikazan na Sl.7.4.



Sl.7.4 Deo grafa HC pretrage sa vrednostima kriterijumske funkcije (- broj pločica koje nisu na svom mestu u odnosu na cilj) označenim iznad svakog čvora – stanja (pozicije).

Primenjena kriterijumska funkcija ima više lokalnih ekstremuma, odakle sledi da HC u tom slučaju nije kompletna pretraga. Npr. na Sl.7.5 dat je primer početnog i ciljnog stanja, u kome svaki potez smanjuje vrednost kriterijumske funkcije. Odavde sledi da je dato početno stanje lokalni, ali ne i globalni maksimum primenjene kriterijumske funkcije.

HC algoritam podseća na algoritam pretrage u dubinu, pri čemu nema povratka na prethodna stanja u cilju razmatranja alternativnih sekvenci rešenja. Stoga HC pripada klasi nepovratnih strategija.



Sl.7.5 Početna i ciljna konfiguracija u igri 8 klizećih pločica, za koju HC pretrage ne može da napravi ni jedan korak. Startna pozicija je zapravo u lokalnom maksimumu. Iz ove pozicije ovom strategijom nije moguće doći do ciljnog stanja u kome se ostvaruje globalni maksimum kriterijumske funkcije (vrednost 0).

## 7.2. SIMULIRANO KALJENJE

HC algoritam, koji nikada ne pravi poteze naniže u pravcu stanja sa manjim vrednostima kriterijumske funkcije u odnosu na trenutno stanje, je zgarantovano nekompletna, jer se u opštem slučaju može zaustaviti u bilo kom lokalnom maksimumu. Nasuprot tome, algoritam slučajnog izbora naslednika (Random Walk –RW) je kompletan ali krajnje neefikasan. Stoga se čini opravdanim pokušaj da se kombinuju HC i RW u cilju povećanja efikasnosti uz zadržavanje kompletnosti kombinovanog algoritma. Algoritam u kome je ovaj sinergizam ostvaren naziva se algoritam simuliranog kaljenja (Simulated Annealing – SA). Autorstvo se pripisuje Scott Kirkpatrick-u i koautorima, Sl.7.6.

Princip SA algoritma se odavno primenjuje u metalurgiji, odakle je i preuzeto ime algoritma. Naime da bi se u metalurgiji postigao dobar kvalitet nekog metala, često se vrši zagrevanje do vrlo visokih temperatura, a zatim kontrolisano hladjenje. Rezultat ovog postupka je dobijanje materijala s kristalnom rešetkom niskog energetskog nivoa, što materijalu daje niz dobrih osobina. Postupak se može posmatrati i kao minimizacija energije pomoću postepenog smanjivanja temperature. Da bi razumeli simulirano kaljenje, posmatrajmo HC algoritam minimizacije kriterijumske funkcije čiji je oblik kao reljef na Sl.7.1. Problem je ekvivalentan stavljanju klikera na početnu poziciju negde na ovom reljefu i puštanjem da se slobodno spusti. Jasnije je da će kliker u opštem slučaju završiti u nekom lokalnom udubljenju (minimumu) najbližem početnoj poziciji. Međutim, ako bismo ceo reljef zajedno sa klikerom, počeli da drmamo sa amplitudom oscilacija koje se tokom vremena polako smanjuju, kliker bi završio u najdubljem udubljenju, odnosno u globalnom minimumu. Upravo SA algoritam simulira ovu situaciju, tako što je ekvivalentno drmanje u početku jako (visoka temperatura), a onda se postepeno smanjuje (smanjivanje ka nižim temperaturama).



Sl.7.6 Scott Kirkpatrick je sa koautorima C. Gelatt-om i M. Vecchi-jem 1983. godine objavio rad u čuvenom časopisu Science pod nazivom „Optimization by Simulated Annealing”, u kome su prvi put izloženi osnovni principi SA algoritma i dokazana njegova asimptotska kompletnost.

Umesto da bira najbolji potez, ovaj algoritam bira slučajan potez. Ako potez poboljšava situaciju, uvek se prihvata sa verovatnoćom 1, u suprotnom, algoritam prihvata potez sa nekom verovatnoćom koja je manja od 1. Verovatnoća eksponencijalno opada sa pogoršanjem situacije koja je izazvana potezom. Verovatnoća takođe opada kako temperatura  $T$  opada. Stoga su loši potezi verovatniji na početku kada je temperatura visoka, i onda postaju sve manje verovatni kako temperatura opada. Pokazuje se formalno, da ako temperatura opada dovoljno sporo, SA nalazi globalni minimum sa verovatnoćom koja teži 1. Na Sl.7.7 je dat pseudo kod SA algoritma.

---

#### SIMULIRANO KALJENJE (SA)

$E$  – kriterijumska funkcija, ima značenje energije

$T$  - temperatura

- Generisati slučajno potez i izračunati razliku  $\Delta E = E_{\text{novi}} - E_{\text{stari}}$
- Ako je  $\Delta E \leq 0$ , prihvatiti potez
- Ako je  $\Delta E > 0$ , prihvatiti potez sa verovatnoćom:

$$P(\Delta E) = e^{-\frac{\Delta E}{kT}}$$

odnosno, porediti  $P(\Delta E)$  sa slučajno generisanim brojem  $rand$  između 0 i 1.

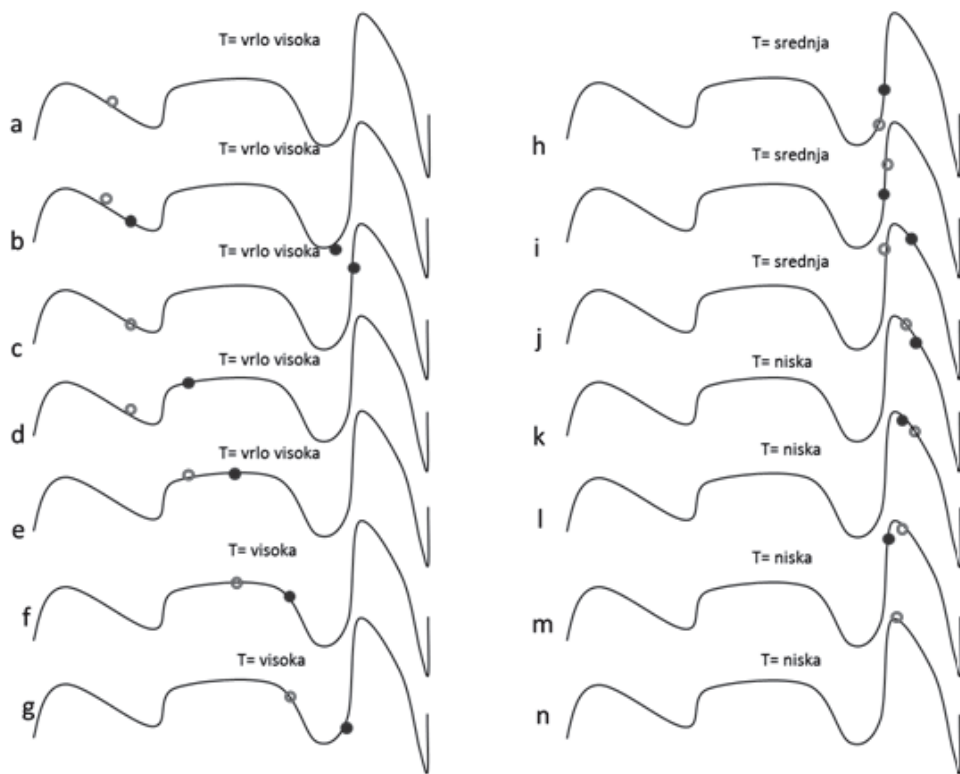
Ako je  $P(\Delta E) < rand$ , prihvatiti potez, u suprotnom odbiti.

---

Sl.7.7 Algoritam simuliranog kaljenja za potrebe minimizacije kriterijumske funkcije.



Očigledno je da efikasnost algoritma zavisi od pažljivo izabranog profila promene temperature kroz iteracije, tzv. rasporeda hladjenja (Cooling Schedule). Ako temperatura suviše brzo opada, SA može da promaši globalni ekstremum, a ako se suviše sporo menja, konvergencija je spora i algoritam je neefikasan. Na Sl.7.8 dat je primer rada SA algoritma iz iteracije u iteraciju prilikom rešavanja problema maksimizacije kriterijumske funkcije.



Sl.7.8 Primer rada SA algoritma iz koraka u korak prilikom maksimizacije kriterijumske funkcije. a) startna pozicija, b) novo stanje se prihvata iako je E manje, d) novo stanje se prihvata budući da je E veće, e) novo stanje se prihvata jer je E veće, f) novo stanje se prihvata iako je E manje, g) novo stanje se prihvata iako je E manje, h) novo stanje se prihvata pošto je E veće, i) ovaj korak se odbacuje, budući da se usled pada temperature smanjila verovatnoća prihvatanja smanjenja E, j) novo stanje se prihvata budući da je E veće, k) novo stanje se prihvata pošto je promena mala, l) novo stanje se prihvata pošto je E veće, m) novo stanje se odbacuje, pošto je za dato E i nisku temperaturu, verovatnoća prihvatanja mala, n) asimptotska konvergencija algoritma ka globalnom maksimumu.

## 7.3 LOKALNO PRETRAŽIVANJE PO SNOPU

Ukoliko zahtevi za memorijom nisu krajnje restriktivni, HC pretraga se može poboljšati uvođenjem k čvorova koji generišu svoje naslednike, umesto samo jednog. Tako dolazimo do koncepta lokalne pretrage po snopu (Local Beam Search – LBS), Sl.7.9. Algoritam započinje slučajnim izborom k početnih stanja. Zatim se generišu svi njihovi naslednici. Ukoliko među njima nije ciljni čvor, sledi izbor k najboljih iz celokupne liste naslednika i ponavljanje procedure.

---

### LOCALBEAMSEARCH (LBS)

Generisati k slučajno izabranih stanja

**do for**

- Generisati sve naslednike svih k stanja
- Ako je bilo koji od njih rešenje, završiti pretragu
- U suprotnom selektovati k najboljih

**end**

---

Sl.7.9 Algoritam lokalne pretrage po snopu (Local Beam Search – LBS)

Na prvi pogled, LBS sa k stanja izgleda kao paralelno pokretanje k HC nezavisnih pretraga. Međutim, izbor k čvorova za narednu iteraciju iz skupa svih naslednika se može interpretirati i kao svojevrsna razmena informacija između ovih paralelnih procesa, obezbeđujući maksimalno dobar izvor u datim uslovima. random-restarta u paraleli umesto u sekvenci (na red). Međutim, ova dva algoritma su veoma Na primer, ako jedno stanje generiše nekoliko dobrih sukcesora a ostala k-1 stanja generišu loše sukcesore, algoritam ubrzo napušta neperspektivne pretrage i premešta svoje resurse na mesto gde je načinjen najveći napredak. Osnovni LBC algoritam može patiti od nedostatka različitosti među k stanja, kao posledica koncentrisanja pretrage u malom regionu prostora stanja. Varijanta ovog algoritma koja se zove stohastička lokalna pretraga po snopu (Stochastic Local Beam Search – SLBS), analogna sa stohastičkim HC-om, pomaže u prevazilaženju ovog problema. Umesto da bira k naslednika iz skupa svih naslednika, SLBS slučajno bira k naslednika. SLBS podseća na proces prirodne selekcije, gde naslednici stvaraju sledeću generaciju u skladu sa svojim sposobnostima (prilagodjenošću), odnosno vrednostima kriterijumske funkcije.

## 7.4 GENETSKI ALGORITMI

Cilj genetskih algoritama (GA) je računarska imitacija procesa na kojima počiva prirodna selekcija, i njihova primena u rešavanju poslovnih i istraživačkih problema. John Hollanda je 60-ih i 70-ih godina prošlog veka prvi formulisao osnovnu strukturu genetskih algoritama, Sl. 7.10. Genetski algoritmi pružaju okvir za izučavanje efekta-

ta biološki inspirisanih procesa poput odabira partnera, reprodukcije, mutacije i ukrštanja genetskih informacija. U prirodi, ograničenja i opterećenja određenog okruženja primoravaju različite vrste i različite jedinke u okviru vrste da se takmiče u ostavljanju najbolje prilagođenih potomaka. U svetu genetskih algoritama, pogodnost raznih potencijalnih rešenja se poredi, i najpogodnija rešenja evoluiraju da bi proizvela još bolja rešenja.



Oblast genetskih algoritama je pozajmila dobar deo terminologije iz oblasti genetike. Svaka ćelija u našem telu sadrži isti skup hromozoma, nizova DNK koji predstavljaju šemu po kojoj smo stvoreni. Svaki hromozom se može podeliti u gene, koji predstavljaju blokove DNK dizajnirane da koduju jednu karakteristiku, poput boje očiju. Konkretna instanca jednog gena je alel. Svaki gen se nalazi na određenom lokusu u hromozomu. Rekombinacija, ili krossover, javlja se tokom reprodukcije, kada se formira novi hromozom kombinovanjem karakteristika oba roditelja. Mutacija, promena jednog gena u hromozomu potomka, javlja se slučajno i relativno retko. Potom se procenjuje prilagođenost (pogodnost) potomka – ili u smislu sposobnosti za život (koji treba da bude dovoljno dugačak da se jedinka i sama reprodukuje), ili u smislu plodnosti potomka.

Sl.7.10 John Henry Holland (1929 - ) američki naučnik, profesor psihologije, elektrotehnike i računarstva, koji je prvi dao osnovnu strukturu genetskih algoritama. Smatra se pioninom ove oblasti, začete 60-tih godina XX veka.

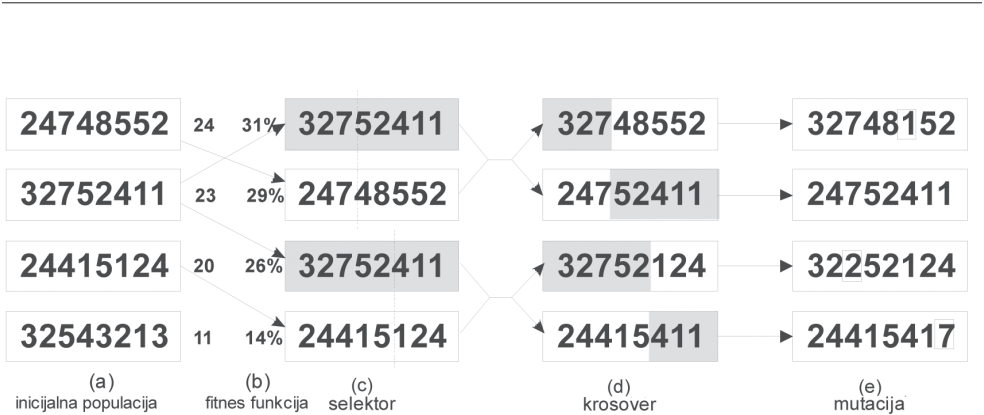
U polju genetskih algoritama, hromozom se odnosi na kandidata za rešenje, gen je jedan bit ili cifra u kandidatu, a alel je konkretna instanca bita ili cifre. Genetski algoritmi koriste tri operatora:

1. **Selekcija.** Operator selekcije se odnosi na metod odabira hromozoma koji će učestvovati u reprodukciji. Funkcija prilagođenosti evaluiira svaki od hromozoma (kandidata za rešenje), i što je hromozom prilagođeniji, veća je verovatnoća da će biti izabran za reprodukciju.
2. **Rekombinacija (krossover).** Operator rekombinacije kreira dva nova potomka nasumično birajući lokus u dva hromozoma izabrana u postupku selekcije i razmenjujući podsekvence tih hromozoma levo i desno od datog lokusa. Na primer, u binarnom obliku, dva niza 11111111 i 00000000 mogu se rekombinovati na šestom lokusu i generisati dva nova potomka 11111000 i 00000111.
3. **Mutacija.** Operator mutacije nasumično menja bitove ili cifre na određenom lokusu u hromozomu – po pravilu sa jako malom verovatnoćom. Na primer, posle rekombinacije, potomak 11111000 bi mogao mutirati na lokusu 2 i postati 10111000. Mutacija unosi nove informacije i sprečava prebrzu konvergenciju ka lokalnom optimumu.

Većina genetskih algoritama funkcioniše tako što iterativno ažurira skup potencijalnih rešenja koji se naziva populacija. Prilagođenost svakog člana populacije se procenjuje u svakoj iteraciji. Nova populacija zamenjuje staru korišćenjem gore navedenih operatora, pri čemu se najprilagođeniji članovi biraju za reprodukciju ili kloniranje. Mera prilagođenosti  $f(x)$  je realna funkcija koja operiše nad hromozomima (potencijalnim rešenjima), a ne genima, tako da se njen argument  $x$  odnosi na numeričku vrednost koju hromozom uzima u trenutku evaluacije prilagođenosti.

Genetski algoritmi (GA) se mogu smatrati varijantom SLBS pretraživanja u kojima se naslednici generišu kombinovanjem po 2 roditeljska, umesto modifikovanjem jednog jedinog stanja.

Kao i LBS pretraživanje, genetski algoritmi počinju sa skupom od  $k$  slučajno generisanih stanja, koja se nazivaju populacija. Svako stanje, ili individua, se predstavlja kao string (niz) nad konačnim alfabetom. Najčešće je niz binaran, sačinjen od nula i jedinica. Na primer, stanje 8-kraljica treba da reprezentuje stanje 8 kraljica, svaku u koloni od 8 pola, i na taj način zahteva  $8 \cdot \log_2 8 = 24$  bita. Alternativno, stanje se može predstaviti pomoću 8 cifara, svaka u opsegu od 1 do 8. Sl.7.11 prikazuje populaciju od četiri 8-cifarska niza koja predstavljaju jedno stanje u problemu 8-kraljica.



Sl.7.11 Genetski algoritam. Inicijalna populacija (a) je rangirana pomoću mere performanse (fitesa) (b). Na osnovu ovog uredjenog skupa se biraju slučajno dva para za reprodukciju (c). Dobijeni potomci (d), kroz proces mutacija daju novu populaciju (e).

Stvaranje nove generacije stanja je prikazano na Sl.7.11 (b) do (e). Na Sl 7.11 (b), svako stanje se procenjuje pomoću evaluacione funkcije ili mere performanse (fitesa). Mera performanse daje i veće vrednosti za bolja stanja. Vrednosti fitesa za prikazana 4 stanja su 24, 23,20 i 11, respektivno. U ovoj varijanti genetskog algoritma, verovatnoća da neko bude izabran za reprodukciju je direktno proporcionalna vrednosti mere performanse. Verovatnoće su prikazane u formi procenata desno od odgovarajućih vrednsoti za fites.

Na slici 7.11 (c), 2 para su slučajno izabrana za reprodukciju, u saglasnosti sa verovatnoćama prikazanim u Sl.7.11 (b). Primetimo da je jedan pojedinac izabran dva puta, a da jedan nije izabran nijednom. Za svaki par koji se uparuje, tačka ukrštanja (krossover) se slučajno bira u odnosu na poziciju u nizu. Na slici 7.11, tačke ukrštanja su nakon treće cifre u prvom paru i nakon 5 cifre u drugom paru.

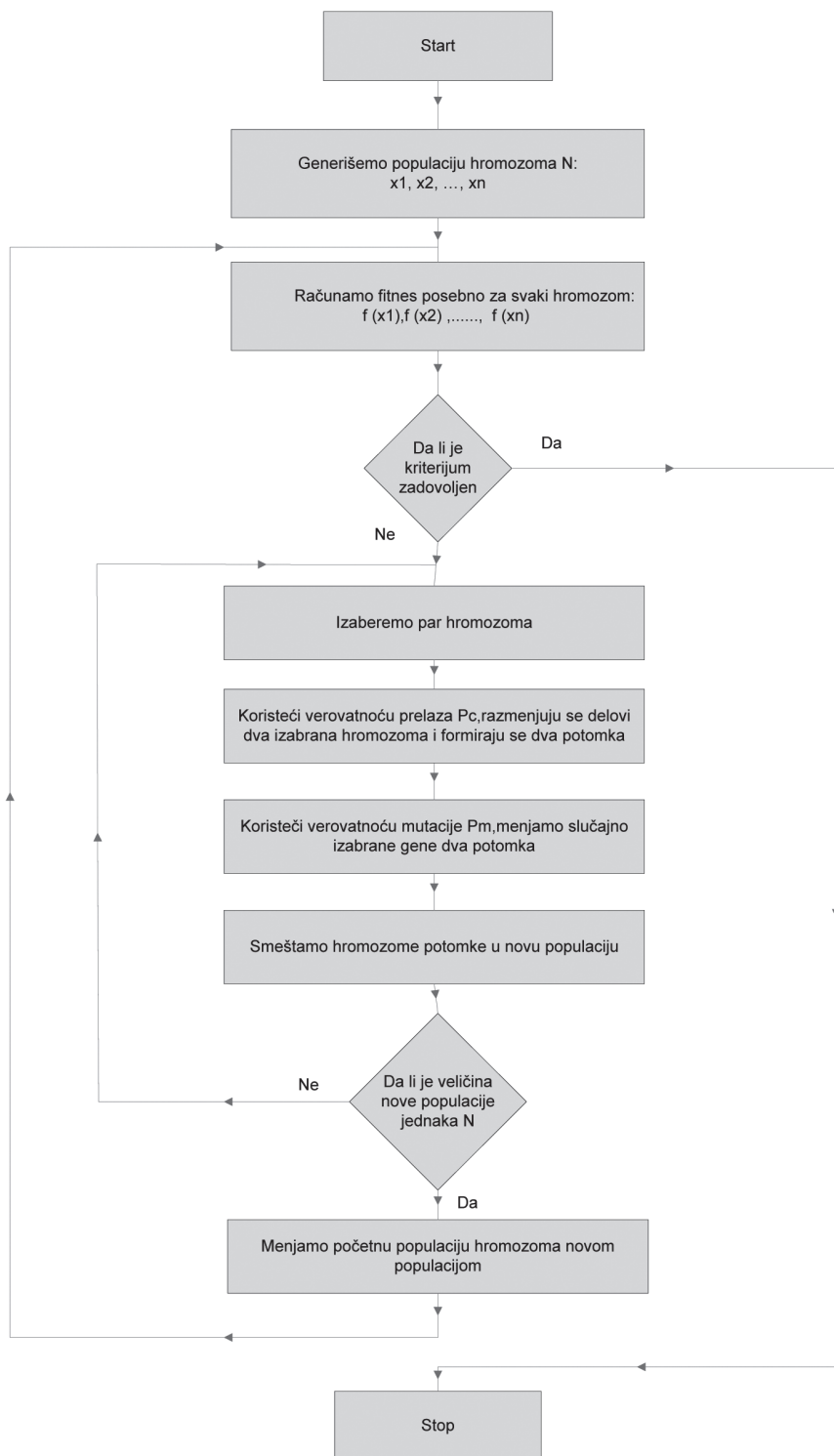
Na Sl 7.11 (d), potomci su kreirani ukrštanjem roditeljskih nizova u tački ukrštanja. Na primer, prvo dete prvog para uzima prve tri cifre od prvog roditelja i preostale cifre od drugog roditelja, dok drugo dete uzima prve tri cifre od drugog roditelja i preostale cifre od prvog roditelja. Čest slučaj je da je populacija veoma raznovrsna na početku procesa, tako da proces ukrštanja često pravi velike korake u prostoru stanja na početku procesa pretraživanja i manje korake kasnije, kada je većina individua vrlo slična, što u izvesnom smislu podseća na algoritam simuliranog kaljenja SA.

Konačno, na Sl 7.11 (e), svaka lokacija je podložna slučajnim mutacijama sa malom nezavisno odabranom verovatnoćom. Jedna cifra je mutirana u prvom, trećem i četvrtom potomku.

Genetski algoritmi, kao klasa algoritama stohastičkih pretraga baziranih na biološkoj evoluciji imaju sledeću opštu strukturu, Sl.7.12:

1. Predstaviti problem u domenu promenljivih kao hromosome fiksirane dužine; izabrati veličinu populacije hromozoma  $N$ , verovatnoću prelaza  $P_c$  i verovatnoću mutacije  $P_m$ .
2. Definirati funkciju fitnesa koja meri performanse svakog individualnog hromozoma. Funkcija fitnesa uspostavlja osnovu za selekciju hromozoma izabranih za reprodukciju.
3. Nasumično generišemo početnu populaciju hromozoma  $N$ :  $x_1, x_2, \dots, x_N$ .
4. Izračunamo fitnes posebno za svaki hromozom:  $f(x_1), f(x_2), \dots, f(x_N)$ .
5. Izaberemo par hromozoma iz postojeće populacije. Roditelji hromozomi su izabrani na osnovu verovatnoće povezane sa njihovim fitnesom. Hromozomi koji imaju veću vrednost fitnesa imaju veće šanse da budu izabrani.
6. Kreira se par potomaka hromozoma korišćenjem genetskih operatora krossovera i mutacije.
7. Kreirane potomke hromozoma stavljamo u novu populaciju.
8. Ponavljamo korake 5 - 7 dok veličina nove populacije ne dostigne veličinu početne populacije.
9. Zamenjujemo početnu roditeljsku populaciju sa novom populacijom - populacijom potomaka.
10. Vraćamo se na korak 4. sve dok ne zadovoljimo zadati kriterijum.

Kao što vidimo, GA predstavlja iterativni proces. Svaka iteracija se zove generacija. Tipičan broj generacija za jednostavnije GA se kreće od nekoliko desetina do nekoliko stotina jedinki.



Sl.7.12 Opšta struktura genetskih algoritama.

Kao i kod algoritma SLBS pretraživanja, genetski algoritmi kombinuju tendenciju uspona sa slučajnim istraživanjem i razmenu informacija putem paralelnih niti pretraživanja. Osnovna prednost genetskih algoritama potiče od operacije ukrštanja. Osim toga, može se pokazati da ako je pozicija genetskog koda na početku permutovana u slučajni poredak, ukrštanjem se ne dobija nikakvo poboljšanje.

Teorija genetskih algoritama objašnjava kako se ovo radi kada se koriste **šeme**, koje predstavljaju podnizove u kojima se neke pozicije ostavljene nespecificirane. Na primer, šema 246\*\*\*\* opisuje sva stanja 8-kraljica u kojima su prve tri kraljice na pozicijama 2, 4 i 6 respektivno. Nizovi koji se poklapaju sa šemom (kao što je niz 24613578) se nazivaju instance šeme. Pokazuje se da ako je srednja vrednost mere performanse pojedinih instanci jednog šablona iznad prosečne, tada će broj instanci šablona unutar populacije rasti sa vremenom. Jasno je da je malo verovatno da ovaj efekat bude značajan ako su susedni bitovi potpuno međusobno nekorelisani, jer će tada postojati nekoliko susednih blokova koji obezbeđuju konzistentnu korist. Genetski algoritmi najbolje rade kada šeme odgovaraju značajnim komponentama rešenja. Na primer, ako niz predstavlja antenu, tada šeme mogu predstavljati komponente antene, kao što su reflektori i deflektori. Ovo ukazuje na to da korišćenje genetskih algoritama zahteva pažljivo projektovanje reprezentacije.

## 7.5 LOKALNE PRETRAGE U KONTINUALNIM PROSTORIMA

Do sada nijedan od opisanih algoritama ne može da se izbori sa kontinualnim prostorima stanja. Operator generisanja naslednika će u većini slučajeva vratiti beskonačno mnogo stanja.

Postoji više metoda koje pokušavaju da koriste gradijent kriterijumske funkcije u cilju nalaženja maksimuma. Gradijent kriterijumske funkcije je vektor  $\nabla f$  koji daje veličinu i pravac najstrmijeg nagiba, i dat je vektorom čije su komponente parcijalni izvodi po svim argumentima

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial y_3} \right).$$

U nekim slučajevima možemo naći maksimum rešavanjem jednačine  $\nabla f = 0$ , ali je nalaženje rešenja u zatvorenoj analitičkoj formi više izuzetak nego pravilo. Ako se ograničimo na traženje lokalnog ekstremuma, možemo sprovesti prilagođenu varijantu HC algoritma najstrmijeg uspona, po kojoj se ažuriranje narednog stanja vrši pomoću izraza

$$x \leftarrow x + \alpha \nabla f(x)$$

gde je  $\alpha$  mala konstanta, nazvana veličina koraka, budući da ona određuje veličinu skoka u naredno stanje. Često nam u praksi nije dostupna kriterijumska funkcija u formi koja je pogodna za računanje gradijenta. U tim slučajevima, može se izračunati tzv. empirijski gradijent, tako što se računaju promene kriterijumske funkcije za male promene argumenata.



Odredjivanje najpovoljnije vrednosti za  $\alpha$  je ključno za efikasnost ove klase algoritama. Ukoliko je  $\alpha$  suviše malo, potreban je veliki broj koraka; ukoliko je  $\alpha$  suviše veliko, pretraga može premašiti maksimum. Tehnika linijskog pretraživanja pokušava da prevaziđe ovaj problem proširujući pravac gradijentna – najčešće ponavljenim dupliranja  $\alpha$  sve dok  $f$  ne počne ponovo da opada. Tačka u kojoj se ovo događa postaje novo tekuće stanje.

Za mnoge probleme, najefektivniji algoritam je poznati Njutn-Rafsonov metod. Ovo je opšta tehnika za pronalaženje korena funkcije – odnosno rešavanje jednačina forme  $g(x) = 0$ . On radi tako što se računa nova estimacija za koren  $x$  u saglasnosti sa Njutnovom formulom

$$x \leftarrow x - \frac{g(x)}{g'(x)}.$$

Da bi pronašli maksimum ili minimum od  $f$ , moramo naći  $x$  takvo da je gradijent jednak nuli, tj.  $\nabla f = 0$ . Drugim rečima,  $g(x)$  u Njutnovoj formuli postaje  $\nabla f(x)$ , i jedinačina može biti zapisana u matricno-vektorskoj formi

$$x \leftarrow x - H_f^{-1}(x) \nabla f(x)$$

gde je  $H_f(x)$  Hesijan matrica drugih izvoda, čiji su elementi  $H_{ij}$  dati pomoću  $\partial^2 f / \partial x_i \partial x_j$ . Dok god Hesijan ima  $n^2$  članova, Njutn-Rafson postaje skup u prostorima velikih dimenzija, pa su razvijene mnoge aproksimacije ovog postupka.

Metode lokalnog pretraživanja imaju nedostatak zaustavljanja u lokalnim ekstremima, ivicama, i platoima, slično kao i u slučaju diskretnih prostora stanja. Metode slučajnih restarta ili simulirano kaljenje mogu biti od velike praktične koristi za izbegavanje ranog zaustavljanja u lokalnim ekstremumima.

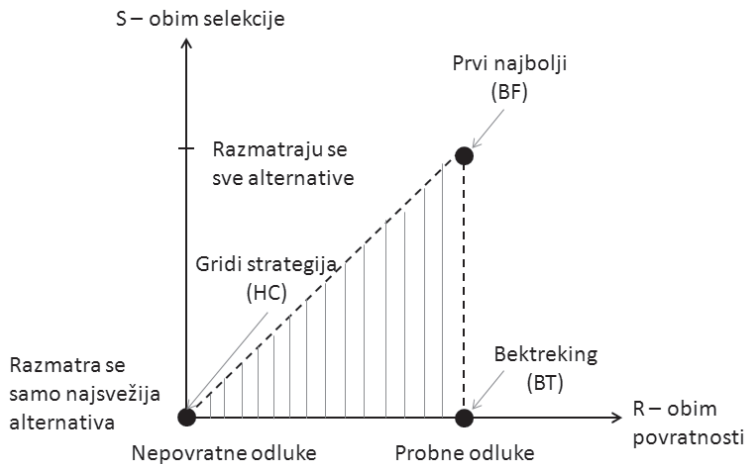
Osim optimizacionih problema koja smo do sada razmatrali, postoji i važna klasa tzv. optimizacionih problema pod ograničenjem (constrained optimization). To su oni problemi optimizacije u kojima rešenje mora da zadovolji neka dodatna ograničenja nad vrednostima argumenata - promenljivih. Težina problema ograničene optimizacije zavisi od prirode ograničenja i kriterijumske funkcije. Najpoznatija klase ovih algoritama su linearno programiranje, u kojima je kriterijumska funkcija linearna, a ograničenja predstavljaju linearne nejednakosti koje formiraju konveksni region u prostoru argumenata.

## 7.6 HIBRIDNE STRATEGIJE

Tri osnovne strategije gridi ili penjanje uzbrdo (HC), bektreking (BT) i prvi najbolji (BF-Best First) mogu se posmatrati kao tri ekstremne tačke u kontinualnom prostoru strategija pretraga. Da bi smo pojasnili ovaj koncept, uvedimo dve karakteristične dimenzije strategija pretraga:

1. Sposobnost povratka (R-Recovery of pursuit): sposobnost strategije da se povrati iz dela prostora pretrage koji ne vodi izglednom rešenju u cilju povratka na prethodno suspendovane alternative,
2. Zahvat evaluacije (S-Scope of evaluation): broj alternativa koje se razmatraju pri svakoj odluci.

U okvirima dimenzije R nalazimo da je HC jedan ekstrem u kome nema nikakve mogućnosti povratka na suspendovane alternative, dok su BT i BF drugi ekstrem u kome su sve odluke probne i povratak na suspendovane alternative je uvek moguć. Po dimenziji S nalazimo da su strategije HC i BT usko fokusirane na skup najnovijih raspoloživih alternativa, dok BF strategija razmatra pri svakoj odluci skup svih raspoloživih alternativa, kako onih najnovijih, tako i svih suspendovanih u prošlim koracima odlučivanja, videti Sl.7.13.

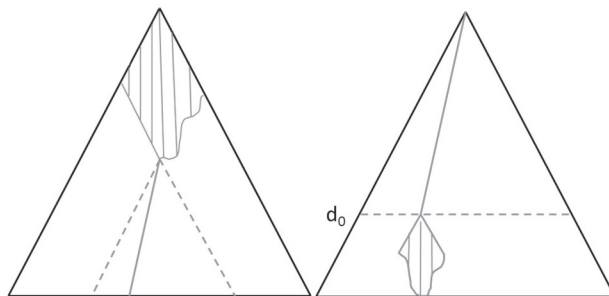


Sl.7.13 Hibridne strategije prikazane u prostoru (R,S). Šrafirani trougao označava oblast hibridnih strategija čije komponente su čiste strategije BF, BT i HC, prikazane kao vrhovi ovog trougla.

Šrafirana oblast prikazuje kontinuum strategija koje imaju delimično svojstva svake od navedena tri prototipa, u cilju postizanja bolje kombinacije njihovih karakteristika. Podsetimo se da HC troši najmanje računskih resursa rizikujući promašaj rešenja, BF radi oprezno i mudro nalazeći rešenje u najmanjem broju koraka, ukoliko ono postoji, ali plaćajući veliku cenu računarskih resursa. BT zahteva ograničenu memoriju, koja se troši za čuvanje samo onih alternativa koje direktno vode do tekuće tačke odlučivanja. Ova strategija u cilju minimiziranja memorije razmatra samo uzak skup najnovijih alternativa, trošeći po pravilu veliku količinu računarskog vremena u cilju nalaženja konačnog rešenja.

## BF-BT kombinacija

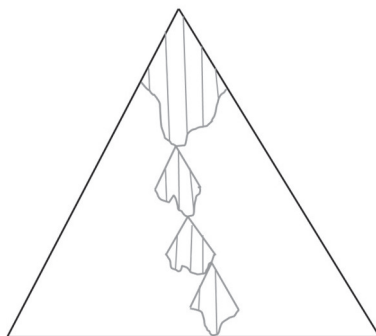
Ovom strategijom se pravi kompromis u pogledu memorijskih zahteva BF strategije. Na Sl.7.14 prikazana je jedna hibridna kombinacija ovih strategija. U prvoj varijanti, prikazanoj levo, prvo se započinje sa BF strategijom dok se ne potroše svi memorijski resursi. Zatim se iz zatečenog stanja liste OPEN bira najbolji čvor i iz njega startuje BT pretraga malih memorijskih zahteva. Ukoliko se ne dostigne rešenje, ovaj čvor se označava kao nerešiv i bira se sledeći najperspektivniji, itd.



Sl.7.14 Šematski prikaz dve BF-BT hibridne strategije. Levo: strategija započinje BF pretragom (šrafirana oblast), a zatim startuje BT. Desno: prvo startuje BT strategija dok se ne dostigne unapred definisana dubina  $d_0$ , a zatim nastavlja BF pretraga do nalaženja konačnog rešenja.

Na desnoj strani Sl.7.15 prikazana je suprotna kombinacija. Prvo se startuje BT dok se ne dostigne unapred definisana dubina pretrage  $d_0$ . Iz dostignute tačke, umesto da se vraćamo nazad, startuje se BF pretraga dok se ne dostigne rešenje ili se pretraga završava neuspehom. Dubina  $d_0$  se bira u skladu sa raspoloživim memorijskim resursima.

Na Sl.7.15 data je još jedna mogućnost kombinovanja ovih strategija, u kojoj se BF koristi lokalno, a BT globalno. Procedura započinje BF pretragom dok se ne iskoriste unapred zadati memorijski resursi  $M_0$ .



Sl.7.15 Hibridna strategija BF-BT. Lokalna BF pretraga (šrafirana oblast) je ulančana u globalnu BT pretragu.

Svi zatečeni čvorovi u listi OPEN se posmatraju kao direktni naslednici startnog čvora i na njih se primenjuje BT pretraga. BT bira najbolji čvor među naslednicima i iz njega startuje novu BF pretragu sve dok se ponovo ne potroše zadati memorijski resursi. Postupak se ponavlja dok se ne dostigne cilj ili proglašeni neuspeh. Ovu strategiju možemo posmatrati i kao informativnu strategiju pretrage u dubinu kod koje se svaki čvor otvara memorijski ograničenom BF pretragom, čiji se čvorovi u listi OPEN smatraju direktnim naslednicima startnog čvora. Memorijski zahtevi ove kombinovane strategije rastu linearno sa dubinom rešenja, dok su memorijski zahtevi originalne BF strategije eksponencijalno zavisni od dubine rešenja.

## Rezime 7. poglavlja

- ♦ U ovom poglavlju smo razmotrili alternativne strategije pretrage u kojima nisu od važnosti putevi do rešenja, već postizanje samog rešenja. Ove strategije imaju lokalni karakter i ne zahtevaju pamćenje svih oprobanih sekvenci akcija (pravila).
- ♦ Strategija najstrmijeg uspona (HC), se zasniva na pamćenju samo tekućeg čvora i izbora narednog čvora iz skupa naslednika kod koga je najveća pozitivna promena kriterijumske funkcije (najstrmiji uspon), ukoliko se radi o maksimizaciji kriterijumske funkcije.
- ♦ Simulirano kaljenje (SA) je algoritam novijeg datuma, koji uvođenjem stohastičke komponente omogućava prevazilaženje problema zaglavljivanja u lokalne ekstremume HC algoritma.
- ♦ Lokalna pretraga po snopu (LBS) omogućava kombinovanje dobrih svojstava istovremenih paralelnih HC pretraga sa međusobnom razmenom informacija o dostignutim vrednostima kriterijumske funkcije u paralelnim procesima.
- ♦ Genetski algoritmi (GA) koriste evolutivne principe ukrštanja i mutacija, koji su se u prirodnim sistemima pokazali uspešni u ekstremizaciji funkcije prilagodjenosti – fitnesa.
- ♦ U kontinualnim prostorima stanja nije moguće direktno primenjivati strategije pretrage diskretnih prostora. Ključna veličina je lokalni gradijent (ili empirijski ako ovaj nije dostupan) na osnovu koga je moguće formulisati različite gradijentne postupke ekstremizacije zadate kriterijumske funkcije. Svi algoritmi ove klase pate od problema lokalnih ekstremuma, koji se mogu prevazići složenijim kombinovanjem sa pretragama tipa simuliranog kaljenja
- ♦ Dat je ilustrativan primer kombinovanja HC i BF strategija u cilju dobijanja hibridnih strategija manjih memorijskih zahteva.

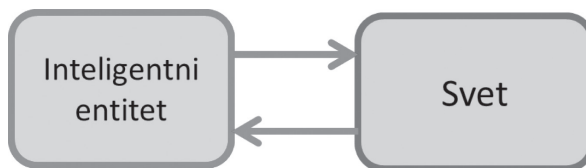
## Pitanja i zadaci

1. Dodelite odgovarajuća imena algoritmima koji se dobijaju u sledećim specijalnim slučajevima:
  - a. Lokalno pretraživanje po snopu za  $k=1$ .
  - b. Lokalno pretraživanje po snopu sa jednim početnim stanjem i bez ograničenja na broj stanja.
  - c. Simulirano kaljenje u kome je uvek  $T=0$ .
  - d. Simulirano kaljenje u kome je uvek  $T=\infty$ .
  - e. Genetski algoritmi sa veličinom populacije  $N=1$ .
2. Dati primere u kojima lokalne pretrage garantuju nalaženje globalnog optimalnog rešenja.
3. Izvršiti simulaciju problema 8 kraljica. Generisati veliki broj primera i rešenja metodama najstrmijeg penjanja -HC, lokalne pretrage po snopu - LBS i simuliranog kaljenja- SA. Rangirati algoritme na osnovu empirijski dobijene efikasnosti. Obrazložiti ovako dobijen redosled.
4. Obrazložite svojim rečima zašto su evolutivni principi u prirodnim sistemima efikasni.
5. Ukoliko bi u genetskim algoritmima zabranili mutacije, kakve bi efekte očekivali?
6. Ukoliko se u genetskim algoritmima zabrani ukrštanje, koju klasu algoritama dobijamo?
7. Da li je moguće iskombinovati genetske algoritme i algoritme simuliranog kaljenja? Da li u prirodi imamo primere ove kombinacije?

## 8. Osnove simboličkih sistema veštačke inteligencije

Dve glavne konkurentske paradigme VI su **simbolički i konekcionistački pristup**. O konekcionistačkom pristupu će biti više reči u 15. poglavlju.

**Tradicionalni simbolički pristup**, još nazvan i GOF AI (Good Old Fashioned Artificial Intelligence) zasnovan je na pretpostavci da osnovu naše inteligencije predstavlja naša sposobnost transformacije problema u simboličku formu (apstraktno govoreći u nizove simbola nekog formalnog jezika) i manipulisanje tim simbolima. Ovaj stav potiče iz čuvene hipoteze Allen Nevell-a i Herbert Sajmon-a, nazvane **hipoteza sistema fizičkih simbola**, u kojoj se tvrdi da formalni sistem koji manipuliše simbolima ima potrebnu i dovoljnu snagu za kreiranje opšte inteligentne akcije. Najuspešniji oblik simboličkog pristupa VI su ekspertski sistemi, koji se zasnivaju na bazama znanja sastavljenim od elementarnih pravila dobijenih od domenskih eksperata. Formalno gledano, pitanja koja postavljamo i odgovori koje dobijamo od jednog ekspertskog sistema su nizovi simbola. Postupak automatskog rezonovanja, kao centralni algoritamski deo ekspertskih sistema se takodje može posmatrati kao manipulacija nad nizovima simbola baze znanja.



Sl. 8.1 Situiranost inteligentnog entiteta u okruženju – svetu, nameće nužno dvosmernu komunikaciju: od okruženja ka inteligentnom entitetu i obrnuto, od inteligentnog entiteta ka svetu.

Vratimo se jedan korak unazad i osvetlimo na jedan opštiji način nužnost pojave simboličkog pristupa u VI. Ako govorimo o inteligentnom ponašanju neke jedinke, onda je to neodvojivo od njene situiranosti u nekom okruženju – svetu. Odvojenost ova dva entiteta nužno nameće potrebu za dvosmernom komunikacijom, Sl.8.1. Iskustvo

pokazuje da delovanje inteligentnih entitija u okruženju zahteva odgovarajuće unutrašnje struktuiranje inteligentnog entitija, koje će podržati i omogućiti ovu komunikaciju. Jedan od centralnih koncepata ove unutrašnje strukture je znanje. Inteligentni entiteti kao da predviđaju stanje okoline i posledice sopstvenih akcija, što nas navodi na zaključak da poseduju znanje o svom svetu. U kom obliku je to znanje? Kako se do njega dolazi? Kako se koristi? Šta su njegova ograničenja? Ovo su pitanja na koje za sada, kada su u pitanju biološki sistemi, posedujemo mala ili nikakva znanja. Međutim, situacija se menja ako usmerimo našu pažnju na veštačke sisteme, koji poseduju makar rudimentirana inteligentna svojstva. Budući da smo mi kreatori tih sistema, mi odlučujemo šta znači da ti sistemi imaju znanje o svetu koji ih okružuje. Uslovno rečeno, ova znanja se mogu podeliti u dve kategorije:

- ♦ implicitna
- ♦ eksplicitna.

Primer implicitnih znanja su znanja ukodovana u program za rešavanje linearnih jednačina. Teško ih možemo eksplicitno izvući iz računarskog koda, koji rešava sistem linearnih jednačina, i koristiti za druge potrebe. Ovakva znanja se nazivaju **proceduralna znanja**, jer su neodvojiva od procedura koje ih koriste. Poseban oblik eksplicitnih znanja, koja se mogu interpretirati kao iskazi o nečemu, nazivaju se **deklarativna znanja**. Ovakvi iskazi su po pravilu kodovani u simboličke strukture, dostupne procedurama koje to znanje treba da koriste. Postoji više razloga zašto su deklarativna znanja poželjna prilikom sinteze inteligentnih sistema:

- ♦ Lako se menjaju, za razliku od proceduralnih znanja.
- ♦ Može služiti različitim namenama, čak i onim koje uopšte nisu anticipirane u fazi njihovog kodovanja. Baze znanja se ne moraju ponavljati ili posebno prilagođavati za svaku aplikaciju.
- ♦ Mogu se proširivati bez eksplicitne naznake, na osnovu procesa rezonovanja koje dovodi do dodatnih znanja.
- ♦ Ova znanja su dostupna kroz tzv. introspektivne programe, dobijanjem odgovara na postavljena pitanja, o tome šta se zaista zna.

Cena koja se plaća za sve ove prednosti deklarativnog znanja su sporiji i skuplji sistemi za njihovo prikupljanje i korišćenje, u poredjenju sa proceduralnim znanjima. Dakle ovde je na delu žrtvovanje efikasnosti za račun fleksibilnosti.

Interesantno je razmišljati o tome kako stoji stvar sa ova dva tipa znanja kod bioloških sistema. Kod nižih vrsta, kakvi su insekti, skloni smo da tvrdimo da dominiraju proceduralna znanja. Npr. pauk se kroz proces evolucije prilagodio svom svetu u kome plete mrežu a da pri tome ne poseduje eksplicitna znanja o materijalima i mogućim strukturama svoje mreže. S druge strane, kad građevinski inženjer razmišlja o projektu gradnje novog mosta, izgleda nam razumno da se mora osloniti na deklarativna znanja o materijalima i strukturama primerenim za gradnju mostova. Ovo nikako ne znači da ljudi ne koriste i proceduralna znanja. Recimo znanja koja koristi jedan teniski šampion su proceduralna, dok su znanja kojima nas uči jedan dobar profesor deklarativna.



## 8.1 DEKLARATIVNA ZNANJA

Kao što smo videli, inteligentno ponašanje zavisi od znanja koje entitet poseduje o svom okruženju. Dominantan deo ovog znanja je deskriptivan i može se izraziti u deklarativnoj formi. Kako se ovo znanje može formalizovati i uneti u računar? To je pitanje koje nas primarno interesuje u ovoj glavi.

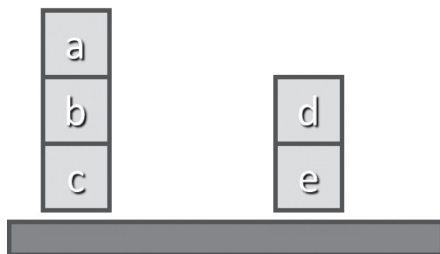
Formalizacija znanja datog u deklarativnoj formi započinje sa konceptualizacijom. **Konceptualizacija** daje opis osnovnih objekata i njihove relevantne međusobne odnose.

**Pojam objekat** se mora shvatiti u najširem smislu, od konkretnih (knjiga, učionica), pa do apstraktnih i fiktivnih (pravda, skup celih brojeva, biće, Homer Simpson). Objekat može biti bilo šta o čemu znamo ili želimo da sopštimo nešto. Formalizacija i reprezentacija znanja ne zahteva da razmatramo sve moguće objekte, već samo one koji su za nas relevantni.

Skup objekata od interesa o kojima se izražava neko znanje se **naziva domen razmatranja**.

### PRIMER 8.1

Na Sl.8.2 dat je primer tzv Sveta blokova, uobičajenog uprošćenog sveta koji se provlači kao primer kroz celokupnu istoriju VI. Za ovaj svet, skup objekata = {a,b,c,d,e}.



Sl.8.2 Primer jednostavnog blokovskog sveta

Drugi element konceptualizacije je opis međuzavisnosti u skupu razmatranih objekata. Uobičajeno je da se one opisuju **funkcijama i relacijama**.

### PRIMER 8.2

U posmatranom svetu blokova iz Primera 8.1 možemo definisati funkcija d koja dodeljuje svakom objektu iz domena samo jedan objekat koji je iznad njega, datu sa

{(b,a), (c,b), (e,d)}

Relacija je mnogo generalniji oblik međuzavisnosti. U blokovskom svetu možemo definisati relaciju **na**, koja postoji između dva bloka ako i samo ako je jedan blok neposredno iznad drugog bloka. Ova relacija je data sa

$$\{(a,b), (b,c), (d,e)\}.$$

Relacija **iznad** važi između dva bloka ako i samo ako je jedan bilo gde iznad drugog. Ova relacija je data sa

$$\{(a,b), (b,c), (a,c), (d,e)\}.$$

**Funkcija** je specijalan tip relacije u kojoj svakom elementu domena korespondiran tačno jedan element.

**Relacija** nad  $n$  objekata ( $n > 1$ ) se zadaje skupom uređenih  $n -$  torki.

**Unarna relacija** odgovara pojmu **atributa**. Dakle objekti domena imaju attribute – svojstva.

Formalno, konceptualizacija se definiše kao trojka koja se sastoji od objekata, skupa funkcija i relacija.

Za primer blokovskog sveta, jedna konceptualizacija je data sa

$$[\{a,b,s,d,e\}, \{d\}, \{na, iznad\}].$$

Kada je jedna konceptualizacija bolja od druge?

Svaka konceptualizacija je podložna promeni i menjajući je, mi se odlučujemo za onu koja će najbolje poslužiti našim namerama i potrebama.

## 8.2 RAČUN PREDIKATA

Kada je zadata jedna konceptualizacija sveta, formalizacija znanja započinje formulisanjem rečenica u nekom izabranom jeziku koji je pogodan za datu konceptualizaciju. U ovom poglavlju daćemo osnove formalnog jezika koji se zove **račun predikata**. Sve rečenice u računu predikata su nizovi simbola uređeni shodno sintaksnim (gramatičkim) pravilima ovog računa.

Znanja se izražavaju posebnim tipom rečenica koje imaju neki smisao i koje se pod određenim uslovima nazivaju sudovima i predikatima.

**Sud ili iskaz** je afirmativna (izjavna) rečenica koja ima smisao i koja je ili istinita ili lažna.

**Predikat** je afirmativna rečenica koja ima smisla, koja sadrži jedan ili više promenljivih parametara i koja postaje sud kada parametri iz rečenice dobiju konkretne vrednosti. Predikat dužine  $n$  na skupu  $D$  je svaka  $n$ -arna relacija skupa  $D$ . Broj parametara koji se pojavljuju u predikatu naziva se dužina (arnost) predikata

### PRIMER 8.4

Navedena su dva predikata  $Q$  i  $R$ , prvi binarni i drugi  $n$ -arni:

$$Q(x,y), R(x_1, x_2, \dots, x_n).$$

## 8.2.1 SINTAKSA RAČUNA PREDIKATA

U računu predikata susrećemo dva tipa simbola (terma):

- ♦ promenljive
- ♦ konstante

Konstante se dele na:

- ♦ funkcijske,
- ♦ relacijske i
- ♦ objektno,

koje se koriste za označavanje funkcija, relacija i objekata iz datog domena.

### Definicija terma:

1. Svaka promenljiva i konstanta se naziva **term**.
2. Ako su  $t_1, t_2, \dots, t_n$  termi i  $p$  funkcijska konstanta, tada je i niz simbola  $p(t_1, t_2, \dots, t_n)$  term.
3. Termi se dobijaju samo na osnovu konačno mnogo puta primenjenih pravila 1. i 2. ove definicije.

### Formule

Ako su  $t_1, t_2, \dots, t_n$  termi i  $r$  relacijska konstanta, onda je niz simbola  $r(t_1, t_2, \dots, t_n)$  **elementarna formula (atom)**.

Pod logičkim formulama podrazumevaćemo formule koje se formiraju kombinacijom elementarnih formula sa nekim od logičkih simbola

### Definicija logičkih formula

1. Svaka elementarna formula je logička formula.
2. Ako su  $s$  i  $r$  logičke formule, onda je i svaki od nizova simbola

$$\sim s, s \wedge r, s \vee r, s \Rightarrow r, s \Leftrightarrow r$$

logička formula i naziva se redom negacija, konjunkcija, disjunkcija, implikacija i ekvivalencija.

3. Logičke formule se dobijaju samo pomoću konačno mnogo primena pravila 1. i 2. ove definicije.

**Kvantifikovane formule (rečenice)**, omogućavaju da se govori o svim objektima iz domena razmatranja ili da se ustanove osobine individualnih objekata bez njihovog eksplicitnog navođenja.

**Univerzalno kvantifikovanje** -  $\forall, \forall x f(x)$

Rečenica kvantifikovana na ovaj način je istinita bez obzira kojim objektima iz domena je instancirana (konkretizovana) promenljiva  $x$ .

## PRIMER 8.5

Date su dve kvantifikovane logičke formule u računu predikata

$$a. (\forall x(Jabuka(x) \Rightarrow Crvena(x)))$$

$$b. (\forall x(Jabuka(x) \wedge Crvena(x)))$$

Rečenice pod a) i b) imaju isti smisao, naime “Svaka jabuka je crvena”, odnosno “Sve su jabuke crvene”, izraženo pomoću različitih logičkih formula.

**Egzistencijalno kvantifikovanje**  $\exists, \exists x g(x)$

Postoji barem jedan objekat iz domena razmatranja za koje važi  $g(x)$ , se izražava formulom  $\exists x g(x)$ .

## PRIMER 8.6

$$(\exists x(Jabuka(x) \Rightarrow Crvena(x)))$$

Ova logička formula ima značenje “Postoji jedna crvena jabuka”.

**Kvantifikovana rečenica** (formula) je rečenica u kojoj je prisutan ili egzistencijalni ili univerzalni kvantifikator.

## PRIMER 8.7

$$((\forall x(Jabuka(x) \vee (\exists x Kruška(x)))$$

$$(\forall x(\exists y Voli(x,y))) \quad (\exists x(\forall x Voli(x,y)))$$

Prva rečenica obuhvata sve jabuke ili jednu krušku. Budući da je delovanje kvantifikatora ograničeno na pojedinačne formule, redosled nije od važnosti. Međutim u drugoj i trećoj rečenici, budući da jedan kvantifikator deluje na domen delovanja drugog kvantifikatora, njihov redosled je od ključnog značaja. Npr. drugom rečenicom se saopštava da svaka osoba nekog voli, pri čemu se ne iznosi nikakva tvrdnja o tome da li je ta voljena osoba ista ili različita za različite osobe koje ih vole. U trećoj rečenici se tvrdi da postoji osoba koju svi vole, što je zaista sasvim različito od tvrdnje druge rečenice, iako su formalno samo kvantifikatori zamenili svoja mesta.

Budući da kvantifikovanju ne podležu funkcijske i relacijske konstante – ovaj račun se naziva **kvantifikatorski račun prvog reda** ili **račun predikata prvog reda**.

Sva pojavljivanja neke promenljive mogu biti:

- ♦ slobodna
- ♦ vezana

Pojavljivanje promenljive  $x$  je **vezano** ako je ono locirano iza kvantifikatora ili ako se nalazi u zoni dejstva nekog kvantifikatora

Sva pojavljivanja promenljive  $x$  koja nisu vezana, nazivamo **slobodnim** pojavljivanjem.

## PRIMER 8.8

$$\forall xP(x,y) \Rightarrow \exists yP(x,y)$$

Prvo i drugo pojavljivanje promenljive  $x$  i drugo pojavljivanje promenljive  $y$  su vezane promenljive. Ostala pojavljivanja promenljivih su slobodna.

Formula u kojoj su sve promenljive vezane naziva se **zatvorena formula**.

Formula u kojoj nema ni slobodnih ni vezanih promenljivih naziva se **osnovna formula (ground)**.

Vezana promenljiva se u svim svojim pojavljivanjima može zameniti simbolom neke druge promenljive koja se ne pojavljuje u formuli. Pri tome se značenje formule ne menja.

## PRIMER 8.9

$$\forall zP(z,y), \quad \forall xP(x,y)$$

Obe formule imaju isto značenje.

Negacija ispred kvantifikatora, menja ga u suprotan, naime:

$$\sim (\forall x)f(x) \equiv (\exists x) \sim f(x)$$

$$\sim (\exists x)f(x) \equiv (\forall x) \sim f(x)$$

## 8.2.2 SEMANTIKA RAČUNA PREDIKATA

Neformalno, semantika je posebno preslikavanje između elemenata konceptualizacije i simbola prirodnog jezika. Time se uvodi narativna komponenta kojom se povezuju elementi konceptualizacije sa rečenicama računa predikata, omogućavajući evaluaciju njihove istinosne vrednosti.

Istinitost neke rečenice zavisi od njene interpretacije, koja je određena ako je navedeno značenje njenih konstanti, funkcijskih i relacijskih simbola u odnosu na domen razmatranja, odnosno interpretaciju.

Interpretacija rečenice  $p$ , označava se sa  $I(p)$ .

Ako je neka pravilno definisana formula (pdf), tačna pri barem jednoj interpretaciji  $I$  i za neku vrednost promenljivih  $U$ , kazaćemo da je **ispunljiva ili zadovoljiva**. Ta interpretacija se naziva **model pdf**. Činjenica da je rečenica  $f$  zadovoljiva za  $I$  i  $U$  se zapisuje sa

$$\models_I f[U]$$

$I$  kaže se da je pdf  $f$  tačna u odnosu na interpretaciju  $I$  i vrednost promenljivih  $U$ . Pdf koja je netačna pri svim interpretacijama je **neispunljiva (kontradiktorna, nezadovoljiva)**.

Pdf je **valjana** ako je istinita za sve moguće interpretacije i vrednosti promenljivih.

## PRIMER 8.10

Primer jedne valjane pdf

$$P(A) \vee \sim P(A)$$

Zadovoljivost logičkih rečenica zavisi od logičkih operatora i zadovoljivosti komponentnih rečenica.

Navedene definicije zadovoljivosti, modela i nezadovoljivosti mogu se proširiti i na skup rečenica. U slučaju da je skup rečenica nezadovoljiv ravnopravno se koristi i termin **nekonzistentan**.

Rezimirajmo osnovne ideje reprezentacije znanja u računarima:

1. korak: konceptualizacija domena razmatranja
2. korak: definišu se simboli za konstante, kao i funkcijske i relacijske konstante
3. korak: “oživljavanje” definisanih konstanti preslikavanjem u elemente, funkcije i relacije uvedene konceptualizacije
4. korak: formalizacija deklarativnog znanja pisanjem rečenica koristeći definisane konstante i sintaksu jezika predikata.

## 8.3 ZAKLJUČIVANJE

Zaključivanje je proces izvodjenja zaključaka na osnovu premisa. Sposobnost da se izvedu zaključci je esencijalni deo inteligencije. Zaključivanje je proces koji se izvodi u više koraka. Svaki korak u zaključivanju je karakterisan nekim pravilom zaključivanja. Pravilo zaključivanja se sastoji od

- ♦ skupa rečenica koji se nazivaju **uslovi**
- ♦ drugog skupa rečenica koji se nazivaju **zaključci**

Navešćemo neka osnovna pravila zaključivanja koja vode poreklo još od Aristotelovih radova u vezi utemeljenja logičkog mišljenja. Rečenice iznad linije su uslovi, dok je rečenica ispod linije zaključak.

### Modus ponenes (MP)

$$\begin{array}{c} f \Rightarrow g \\ f \\ \hline g \end{array}$$

### Modus tolens (MT)

$$\begin{array}{c} f \Rightarrow g \\ \sim f \\ \hline \sim g \end{array}$$

**Pravilo i eliminacije (AE)**

$$\frac{f \wedge g}{f}$$

$$\frac{f}{g}$$

**Pravilo i formiranja (AI)**

$$\frac{f}{g}$$

$$\frac{g}{f \wedge g}$$

**Univerzalno instanciranje (UI)**

$$\frac{\forall x f}{f_{x/t}}$$

Term  $t$  je slobodan za promenljivu  $x$  u  $f$ . Omogućava zaključivanje od opšteg ka posebnom.

**PRIMER 8.11**

$$\text{iz } (\forall x)P(x, f(x), B) \text{ sledi } P(A, f(A), B)$$

**Pravilo egzistencijalnog instanciranja (EI)**

$$\frac{\exists x f}{f_{x/s(y_1, y_2, \dots, y_k)}}$$

Omogućava eliminaciju egzistencijalnog kvantifikatora. Term  $s$  je nova funkcijska konstanta, a  $y_1, y_2, \dots, y_k$  su slobodne promenljive u rečenici  $f$ .

**PRIMER 8.12**

$$\text{iz } (\exists z)Hates(y, z) \text{ sledi } Hates(y, Foe(y))$$

gde je  $Foe$  nova funkcijska konstanta. Term  $Foe(y)$  označava osobu koju  $y$  mrzi.

**PRIMER 8.13**

Kao ilustraciju navedenih koncepata razmotrimo sledeći primer. Zna se da su konji brži od pasa i da postoji hrt koji je brži od svakog zeca. Zna se da je Soko konj, a Ralf zec. Izvesti da je Soko brži od Ralfa.



|                                                                                                          |           |
|----------------------------------------------------------------------------------------------------------|-----------|
| $\forall x \forall y \text{Konj}(x) \wedge \text{Pas}(y) \Rightarrow \text{Brzi}(x, y)$                  |           |
| $\exists y \text{Hrt}(y) \wedge (\forall z \text{Zec}(z) \Rightarrow \text{Brzi}(y, z))$                 | Premise   |
| $\forall y \text{Hrt}(y) \Rightarrow \text{Pas}(y)$                                                      |           |
| $\forall x \forall y \forall z \text{Brzi}(x, y) \wedge \text{Brzi}(y, z) \Rightarrow \text{Brzi}(x, z)$ |           |
| $\text{Konj}(\text{Soko})$                                                                               |           |
| $\text{Zec}(\text{Ralf})$                                                                                |           |
| <hr/>                                                                                                    |           |
| $\text{Brzi}(\text{Soko}, \text{Ralf})$                                                                  | Zaključak |

Prvo se formalizuju premise.

Primetimo da su dodate dve činjenice koje nisu eksplicitno sadržane u formulaciji problema:

- ♦ Da su hrtovi psi
- ♦ Da je relacija Brži tranzitivna

Izvođenje zaključka obavićemo nizom od 19 rečenica.

|     |                                                                                                                                        |           |
|-----|----------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 1.  | $\forall x \forall y \text{Konj}(x) \wedge \text{Pas}(y) \Rightarrow \text{Brzi}(x, y)$                                                | Premise   |
| 2.  | $\exists y \text{Hrt}(y) \wedge (\forall z \text{Zec}(z) \Rightarrow \text{Brzi}(y, z))$                                               |           |
| 3.  | $\exists y \text{Hrt}(y) \Rightarrow \text{Pas}(y)$                                                                                    |           |
| 4.  | $\forall x \forall y \forall z \text{Brzi}(x, y) \wedge \text{Brzi}(y, z) \Rightarrow \text{Brzi}(x, z)$                               |           |
| 5.  | $\text{Konj}(\text{Soko})$                                                                                                             |           |
| 6.  | $\text{Zec}(\text{Ralf})$                                                                                                              |           |
| 7.  | $\text{Hrt}(y) \wedge (\forall z \text{Zec}(z) \Rightarrow \text{Brzi}(\text{Greg}, z))$                                               | 2,EI      |
| 8.  | $\text{Hrt}(\text{Greg})$                                                                                                              | 7,E       |
| 9.  | $\forall z \text{Zec}(z) \Rightarrow \text{Brzi}(\text{Greg}, z)$                                                                      | 7,E       |
| 10. | $\text{Zec}(\text{Ralf}) \Rightarrow \text{Brzi}(\text{Greg}, \text{Ralf})$                                                            | 9,UI      |
| 11. | $\text{Brzi}(\text{Greg}, \text{Ralf})$                                                                                                | 10,6, MP  |
| 12. | $\text{Hrt}(\text{Greg}) \Rightarrow \text{Pas}(\text{Greg})$                                                                          | 3,UI      |
| 13. | $\text{Pas}(\text{Greg})$                                                                                                              | 12,8 MP   |
| 14. | $\text{Konj}(\text{Soko}) \wedge \text{Pas}(\text{Greg}) \Rightarrow \text{Brzi}(\text{Soko}, \text{Greg})$                            | 1,UI      |
| 15. | $\text{Konj}(\text{Soko}) \wedge \text{Pas}(\text{Greg})$                                                                              | 15,13,IF  |
| 16. | $\text{Brzi}(\text{Soko}, \text{Greg})$                                                                                                | 14,15, MP |
| 17. | $\text{Brzi}(\text{Soko}, \text{Greg}) \wedge \text{Brzi}(\text{Greg}, \text{Ralf}) \Rightarrow \text{Brzi}(\text{Soko}, \text{Ralf})$ | 4,UI      |
| 18. | $\text{Brzi}(\text{Soko}, \text{Greg}) \wedge \text{Brzi}(\text{Greg}, \text{Ralf})$                                                   | 16,11,IF  |
| 19. | $\text{Brzi}(\text{Soko}, \text{Ralf})$                                                                                                | 17,19,MP  |

Proces izvođenja je u potpunosti mehanički. Svaki zaključak sledi iz prethodnih zaključaka mehaničkom primenom nekog pravila zaključivanja. U toku ovog procesa se odbacuju mnoga alternativna rešenja. Selekcija alternativa koja vode ka rešenju su glavni problem automatizacije procesa zaključivanja.

## 8.4 SEMANTIČKA POSLEDICA

Skup rečenica  $G$  **logički implicira** rečenicu  $f$  (rečenica  $f$  je **semantička posledica** skupa  $G$ ) ako i samo ako pri svakoj interpretaciji i vrednosti promenljivih za koju su sve rečenice skupa  $G$  tačne je i rečenica  $f$  tačna.

$$G \models f \Leftrightarrow \models_i G[U] \text{ implicira } \models_i f[U]$$

Procedura (pravilo) zaključivanja je valjana – **logična (sound)** ako i samo ako je svaka rečenica izvedena ovom procedurom iz baze znanja (KB) ujedno i njena semantička posledica. Procedura zaključivanja je **kompletna (potpuna)** ako i samo ako se svaka rečenica koja je semantička posledica KB može dobiti primenom ove procedure zaključivanja.

**Teorija** je skup rečenica zatvoren u odnosu na semantičke posledice.

**Teorija T je kompletna (potpuna)** ako i samo ako svaka rečenica ili njena negacija pripada T.

### PRIMER 8.14

$$\begin{aligned} \{P\} &\models P \\ \{P, P \Rightarrow Q\} &\models Q \\ P \Rightarrow Q &\models P \end{aligned}$$

Značaj logičkog impliciranja u VI leži u tome što na osnovu ovog principa, na osnovu nekog skupa tačnih stavova, možemo generisati neke druge tačne stavove od interesa, koji nisu direktno dostupni u skupu polaznih stavova.

## 8.5 DOKAZIVOST

Problem provere za neku rečenicu da li je semantička posledica zadatog skupa premisa, leži u nužnosti provere za sve moguće interpretacije, što praktično znači beskonačan broj mogućih interpretacija.

Prema jednoj teoremi matematičke logike, kad god je  $f$  semantička posledica skupa premisa  $G$ , postoji konačan dokaz za rečenicu  $f$ , koji počinje rečenicama iz  $G$ .

Ukoliko su pravila zaključivanja logična i kompletna, možemo principijelno odrediti da li neka pdf sledi iz nekog zadatog skupa, iznalaženjem dokaza, umesto korišćenja tablica istinosnih vrednosti.

Ukoliko je neko pravilo zaključivanja logično i ukoliko nadjemo dokaz da  $f$  sledi iz  $G$  na osnovu primene ovog pravila, istovremeno znamo da  $f$  logički sledi iz  $G$ .

Ukoliko je neko pravilo zaključivanja kompletno, znamo da ćemo biti u stanju da potvrdimo da  $f$  sledi iz  $G$  (ukoliko je to zaista tako) koristeći kompletnu pretragu svih mogućih dokaza.

Zamena metode tablica istinosnih vrednosti metodom dokaza, po pravilu daje veliku računarsku uštedu. U opštem slučaju problem da li neka pdf logički sledi iz zadatog skupa pdf, ili se može dokazati na osnovu tog skupa pdf, predstavlja NP težak problem.

**Logički aksiom** je rečenica koja je tačna za sve interpretacije (valjana formula). Proširivanjem polaznih premisa sa logičkim aksiomima omogućava odbacivanje svih pravila zaključivanja i zadržavanje samo modus ponensa, kao logičnog pravila zaključivanja. Ako postoji dokaz rečenice  $f$  iz premisa  $G$ , pri čemu se koriste samo modus ponens i logičke aksiome, tada se kaže da je rečenica  $f$  **dokaziva** iz  $G$  ili da je teorema skupa  $G$ , u oznaci

$$G \mid -$$

Ekvivalentnost dokazivosti i logičke posledice prvi je dokazao Gödel 1930. godine i može se zapisati sa

$$G \mid - f \equiv G \models f$$

Koncept dokazivosti je važan u VI, pošto sugerise kako možemo automatizovati određivanje logičkih implikacija. Polazeći od skupa premisa  $\Delta$ , možemo nabrojati sve zaključke koji slede iz tog skupa. Ako se među zaključcima nadje neka rečenica  $\phi$ , tada je ona dokaziva iz  $\Delta$ , pa prema tome je i logička posledica. Ukoliko se u zaključcima pojavi negacija neke rečenice, npr.  $\sim\phi$ , tada je ona logička posledica skupa premisa  $\Delta$ , a rečenica  $\phi$  logički ne sledi iz  $\Delta$ , dok god je skup  $\Delta$  konzistentan (nekontradiktoran).

## Rezime 8. poglavlja

- ♦ Simbolički pristup VI zasniva se na uverenju da simbolička predstava realnosti i manipulacija nizovima simbola predstavlja dovoljnu osnovu za opštu inteligentnu akciju.
- ♦ Čovek operiše sa dve vrste znanja: proceduralnim i deklarativnim. Ono što predstavlja suštinu inteligencije oslanja se na deklarativna znanja u formi istinitih iskaza o svetu koji nas okružuje.
- ♦ Formalizacija i kodifikacija znanja se u domenu VI obavlja u nekoliko koraka: konceptualizacija, identifikacija objekata i veza, njihovo povezivanje sa elementima konceptualizacije, pisanje rečenica koristeći definisane konstante i sintaksu jezika predikata prve vrste.
- ♦ Pravila zaključivanja na osnovu početnog skupa rečenica, premisa, izvode nove rečenice, zaključke. Posebno su značajna logička pravila zaključivanja, kod kojih za svaku interpretaciju za koju su premise istinite su i zaključci istiniti.

- ♦ Značaj logičkog impliciranja u VI leži u tome što na osnovu ovog principa, na osnovu nekog skupa tačnih stavova, možemo generisati neke druge tačne stavove od interesa, koji nisu direktno dostupni u skupu polaznih stavova.
- ♦ Ako postoji dokaz rečenice  $f$  iz premisa  $G$ , pri čemu se koriste samo modus ponens i logičke aksiome, tada se kaže da je rečenica  $f$  dokaziva iz  $G$  ili da je teorema skupa  $G$ .
- ♦ Gedel je prvi dokazao stav o ekvivalenciji dokazivosti i logičke posledice.

## Pitanja i zadaci

1. Dajte jedan realan primer konceptualizacije.
2. Korišćenjem predikata  
 $\text{Male}(x)$  –  $x$  je muško  
 $\text{Female}(x)$  –  $x$  je žensko  
 $\text{Vegetarian}(x)$  –  $x$  je vegetarijanac  
 $\text{Butcher}(x)$  –  $x$  je mesar,  
 izrazite u računu predikata prve vrste sledeće iskaze:
  - a) Nijedan čovek nije istovremeno i mesar i vegetarijanac
  - b) Svi ljudi osim mesara vole vegetarijance
  - c) Samo su žene mesari vegetarijanci
  - d) Ni jedan čovek ne voli žene vegetarijance
  - e) Ni jedna žena ne voli muškarce koji ne vole sve vegetarijance.
3. Prevedite sledeće rečenice računa predikata prve vrste na svakodnevni srpski jezik:
  - a)  $\forall x \text{ Hesitates}(x) \Rightarrow \text{Lost}(x)$
  - b)  $\sim \exists x \text{ Business}(x) \wedge \text{Like}(x, \text{Showbusiness})$
  - c)  $\sim \forall x \text{ Glitters}(x) \Rightarrow \text{Gold}(x)$
  - d)  $\exists x \forall t \text{ Person}(x) \wedge \text{Time}(t) \wedge \text{Canfool}(x, t)$
4. Za svaku od navedenih rečenica odredite da li je nezadovoljiva, zadovoljiva ili validna
  - a.  $P \Rightarrow P$
  - b.  $P \Rightarrow \sim P$
  - c.  $\sim P \Rightarrow P$
  - d.  $P \Rightarrow \sim P$
  - e.  $P \Rightarrow (Q \Rightarrow P)$ .
5. Po zakonu je označeno kao kriminalna radnja prodaja neregistrovanih pištolja. Pera poseduje više neregistrovanih pištolja koje je kupio od Djoke. Koristeći pravila zaključivanja data u ovoj glavi, izvedite zaključak da je Djoka kriminalac.



## 9. Automatsko rezonovanje

U ovom poglavlju bavićemo se sistemima za automatsko dokazivanje teorema zasnovanim na računu predikata prve vrste i posebnom pravilu zaključivanja – rezoluciji. Osnovni okvir automatskog dokazivanja teorema, daleko prevazilazi sam naziv ove podoblasti VI. Naime, to je istovremeno metodološki okvir za svako automatsko rezonovanje, sisteme zasnovane na znanju, sisteme za dobijanje odgovora, automatsko programiranje i td. Prvi modeli automatskog rezonovanja u VI se mogu naći već u Logic Theorist-u i General Problem Solver-u sistemima razvijenim na Carnegie institutu 60-tih i 70-tih godina prošlog veka.

Postoje tri važne teorijska svojstva logičkih sistema automatskog rezonovanja

- ♦ valjanost (sound)
- ♦ kompletnost
- ♦ traktabilnost.

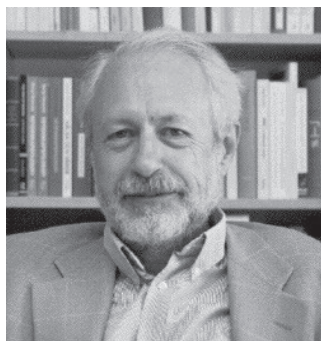
Valjanost je potrebna da bi smo bili sigurni da su izvedeni zaključci tačni.

Kompletnost je potrebna da bi smo bili sigurni da sistem može da izvede bilo koji istinit stav.

Traktabilnost nam obezbedjuje praktičnu izvodljivost automatskog rezonovanja.

Kao što smo videli u prethodnom poglavlju na primeru 8.13, iz zadatih premisa, nizom sukcesivnih primena pravila zaključivanja, njih 5 u ovom slučaju: E, EI, UI, MP, IF, izveli smo željeni zaključak. Da li se ova mukotrpa procedura može pojednostaviti? Da li se upotrebljenih 5 ili u nekom drugom slučaju svih 6 navedeni pravila zaključivanja može redukovati na manji broj, a da se pri tome ne gubi na valjanosti i kompletnosti?

John Alan Robinson, američki matematičar i filozof, Sl.9.1 je dao suštinske doprinose razvoju automatskog dokazivanja teorema. Uveo je princip rezolucije na osnovu prethodnih radova Prawitz-a i dokazao njenu valjanost i kompletnost. Njegov algoritam unifikacije je eliminisao jedan izvor kombinatorne eksplozije u postupku dokazivanja rezolucijom.



Sl.9.1 Dag Prawitz (1936 - ) švedski filozof i logičar poznat po svojim radovima iz teorije dokazivanja i prirodne dedukcije (levo). John Alan Robinson (1928 - ) američki matematičar i filozof je dao suštinske doprinose razvoju automatskog dokazivanja teorema. Uveo je princip rezolucije na osnovu prethodnih radova Prawitz-a i dokazao njenu valjanost i kompletnost. Njegov algoritam unifikacije je eliminisao jedan izvor kombinatorne eksplozije u postupku dokazivanja rezolucijom. Svojim radovima je pripremio teren za osnovni okvir logičkog programiranja, posebno za pojavu jezika PROLOG (desno).

**Rezolucija** je procedura zaključivanja zasnovana na samo jednom efikasnom pravilu izvodjenja. Pored jednostavnosti njena dobra osobina je i

- ♦ valjanost i
- ♦ u širem smislu kompletnost.

Ulazni argument procedure rezolucije je skup rečenica koje se javljaju u pojednostavljenom obliku **klauzalnim formama**, koje su podskup skupa formula predikatskog računa.

**Literal** je elementarna rečenica ili njena negacija. **Elementarna rečenica** se naziva **pozitivni literal**, a njena negacija **negativni literal**.

**Klauzula** je disjunkcija literala. Prikazuju se kao skup literala.

#### PRIMER 9.1

Skupovi

$$\{Na(A,B)\} \text{ i } \{\sim Na(A,B), Iznad(A,B)\}$$

su klauzule, gde drugi označava disjunkciju

$$\sim Na(A,B) \vee Iznad(A,B)$$

**Hornove klauzule** su klauzule sa najviše jednim pozitivnim literalom.

## 9.1 TRANSFORMACIJA U KLAUZALNU FORMU

Na prvi pogled, klauzalna forma može da izgleda kao vrlo restriktivna. Međutim, bilo koja rečenica u računu predikata prve vrste se kroz proceduru od devet koraka

može transformisati u njenu ekvivalentnu klauzalnu formu. Ekvivalencija ima značenje u tom smislu da je polazna rečenica zadovoljiva ako i samo ako je zadovoljiv njoj korespondirajući skup klauzula. Na Sl.9.2 dat je pseudo-kod procedure koja vrši transformaciju proizvoljne zatvorene rečenice u njoj odgovarajuću klauzalnu formu.

| <b>Procedure Convert(x)</b> |                                                 |
|-----------------------------|-------------------------------------------------|
| 1. Begin                    | $x \leftarrow \text{Implication\_out}(x)$       |
| 2.                          | $x \leftarrow \text{Negations\_in}(x)$          |
| 3.                          | $x \leftarrow \text{Standardize\_variables}(x)$ |
| 4.                          | $x \leftarrow \text{Existentials\_out}(x)$      |
| 5.                          | $x \leftarrow \text{Ubiversals\_out}(x)$        |
| 6.                          | $x \leftarrow \text{Disjunctions\_in}(x)$       |
| 7.                          | $x \leftarrow \text{Operators\_out}(x)$         |
| 8.                          | $x \leftarrow \text{Rename\_variables}(x)$      |

Sl.9.2 Pseudo-kod procedure za konverziju proizvoljne rečenice računa predikata prve vrste u klauzalnu formu.

**1. Korak.** Eliminacija logičkih operatora  $\Rightarrow, \Leftarrow, \Leftrightarrow$  i njihova zamena operatorima  $\wedge, \vee, \sim$ .

$f \Rightarrow g$  se zamenjuje sa  $\sim f \vee g$

$f \Leftarrow g$  se zamenjuje sa  $f \vee \sim g$

$f \Leftrightarrow g$  se zamenjuje sa  $(\sim f \vee g) \wedge (f \vee \sim g)$

**2.Korak.** Distribucija operatora negacije  $\sim$ .

$\sim \sim f$  se zamenjuje sa  $f$

$\sim (f \wedge g)$  se zamenjuje sa  $\sim f \vee \sim g$

$\sim (f \vee g)$  se zamenjuje sa  $\sim f \wedge \sim g$

$\sim \forall x f$  se zamenjuje sa  $\exists x \sim f$

$\sim \exists x f$  se zamenjuje sa  $\forall x \sim f$

**3.Korak.** Preimenovanje promenljivih tako da svaki kvantifikator ima jedinstvenu promenljivu, odnosno jedna promenljiva se kvantifikuje najviše jednom u okviru rečenice.

## PRIMER 9.2

$(\forall x P(x,x) \wedge (\exists x Q(x)))$  se prevodi u formulu

$(\forall x P(x,x) \wedge (\exists y Q(y)))$



**4.Korak.** Eliminacija egzistencijalnog kvantifikatora, tzv. Skolemizacija prvo se vrši izvlačenje kvantifikatora u prefiks formule pomoću

$$Qx A(x) \wedge B \text{ se zamenjuje sa } Qx (A(x) \wedge B)$$

$$Qx A(x) \vee B \text{ se zamenjuje sa } Qx (A(x) \vee B)$$

gde formula B ne sadrži promenljivu x a Q označava egzistencijalni ili univerzalni kvantifikator.

Neka je formula zadata u preneksnom obliku

$$\forall x_1 \forall x_2 \dots x_n \exists y R(y)$$

gde R(y) može da sadrži kvantifikatore. Tada se gornja formula zamenjuje sa

$$\forall x_1 \forall x_2 \dots x_n \exists y R(f(x_1, x_2, \dots, x_n))$$

Funkcija  $f(x_1, x_2, \dots, x_n)$  se naziva **Skolemova funkcija**. Njeni argumenti su promenljive vezane univerzalnim kvantifikatorom. Kada je prisutan samo egzistencijalni kvantifikator, Skolemova funkcija se svodi na konstantu, koja se naziva **Skolemova konstanta**.

### PRIMER 9.3

$\exists y R(y)$  se svodi na  $R(c)$ , gde je  $R(c)$  Skolemova konstanta.

Nakon ovog koraka sve formule sadrže samo univerzalni kvantifikator.

**5.Korak.** Konverzija u preneks formu. U ovom koraku svaki univerzalni kvantifikator ima za svoje varijable sopstvene simbole, dok su svi egzistencijalni kvantifikatori eliminisani. Stoga se sada svi univerzalni kvantifikatori mogu premestiti ispred formule, tako da je polje njihovog dejstva cela formula. Ova forma se naziva **preneks forma**. Sastoji se od niza kvantifikatora opštosti - **prefikasa** i formule bez kvantifikatora - **matrica**. Stoga se svi univerzalni kvantifikatori mogu eliminisati, a da se pri tome ne izgubi nikakva informacija.

**6.Korak.** Rečenice se prevode u konjuktivnu normalnu formu, odnosno u konjukciju disjunkcije literala. Svaka matrica se može dovesti u ovu formu višestrukom primenom jednog od distributivnih pravila disjunkcije prema konjukciji, naime zamenom sledećeg oblika

$$w_1 \vee (w_2 \wedge w_3) \rightarrow (w_1 \vee w_2) \wedge (w_1 \vee w_3).$$

**7.Korak.** Eliminacija operatora pisanjem konjukcija, dobijenih u koraku 6, kao skupa klauzula. Npr. rečenica  $P \wedge (Q \vee R)$  se zamenjuje sa skupom koji se sastoji od klauzula  $\{P\}, \{Q, R\}$ .

**8.Korak.** Preimenovanje promenljivih, tako da se svaka promenljiva javlja najviše jednom u svakoj od dobijenih klauzula. Ovaj korak se naziva i pojedinačna standardizacija promenljivih.

#### PRIMER 9.4

Transformisati formulu u klauzalni oblik

$$\forall x(\forall yP(x,y)) \Rightarrow \sim(\forall yQ(x,y) \Rightarrow R(x,y))$$

Sledeći opisani postupak, dat procedurom prikazanom na Sl.9.2, dobijamo sledeći niz rečenica:

1.  $\forall x \sim(\forall yP(x,y)) \vee \sim(\sim\forall yQ(x,y) \vee R(x,y))$
2.  $\forall x(\exists y \sim P(x,y)) \vee \sim(\exists yQ(x,y) \wedge \sim R(x,y))$
3.  $\forall x(\exists y \sim P(x,y)) \vee \sim(\exists zQ(x,z) \wedge \sim R(x,z))$
4.  $\forall x \exists y \exists z \sim P(x,y) \vee (\exists zQ(x,z) \wedge \sim R(x,z))$
5.  $\forall x \sim P(x, F1(x)) \vee \sim(Q(x, F2(x)) \wedge \sim R(x, F2(x)))$
6.  $(\sim P(x, F1(x)) \vee (Q(x, F2(x)) \wedge (\sim P(x, F1(x)) \vee \sim R(x, F2(x))))$
7.  $\{(\sim P(x, F1(x)), (Q(x, F2(x))), \{(\sim P(x, F1(x)), \sim R(x, F2(x)))\}$
8.  $\{(\sim P(x1, F1(x1)), (Q(x1, F2(x1))))\},$   
 $\{(\sim P(x2, F1(x2)), \sim R(x2, F2(x2)))\}$

## 9.2 UNIFIKACIJA

Prilikom dokazivanja teorema i uopšte kod sistema automatskog rezonovanja, neophodno je u toku procesa dokazivanja dovesti do slaganja pojedine podizraze koji sadrže promenljive. Jasno je da izjednačavanje ovih podizraza može biti postignuto samo određenim zamenama promenljivih.

**Unifikacija** je postupak zamene promenljivih u rečenicama u cilju njihovog ekvivalentiranja. Zamena je svako konačno preslikavanje između promenljivih i terma.

Ako je  $x$  promenljiva a  $t$  term koji ne sadrži promenljivu  $x$ , onda se zapis  $x \rightarrow t$ , ili  $x/t$  naziva **zamenska komponenta**.

Konačan skup zamenskih komponenti

$$s = \{x_1 \rightarrow t_1, x_2 \rightarrow t_2, \dots, x_n \rightarrow t_n\} \text{ odnosno}$$

$$s = \{x_1 / t_1, x_1 / t_2, \dots, x_n / t_n\}$$

naziva se zamena ili supstitucija, pod uslovom da se nijedna promenljiva kojoj je dodeljen term se ne javlja u nekom od preostalih terma koji učestvuju u zameni.

## PRIMER 9.5

$$P(x, f(y), B)$$

- |                       |                                        |
|-----------------------|----------------------------------------|
| 1. $P(z, f(w), B)$    | $s_1 = \{z/x, w/y\}$ alfabetska zamena |
| 2. $P(x, f(A), B)$    | $s_2 = \{A/y\}$                        |
| 3. $P(g(z), f(A), B)$ | $s_3 = \{g(z)/x, A/y\}$                |
| 4. $P(C, f(A), B)$    | $s_4 = \{C/x, A/y\}$ osnovna zamena    |

**Alfabetska zamena** je ona u kojoj su jedne varijable zamenjene drugim varijablama. **Osnovna (ground) zamena** je ona u kojoj ni jedan term u literalima ne sadrži promenljive. Da bi smo označili jednu zamenu izraza  $w$ , primenom zamene  $s$ , pišaćemo  $ws$ . Stoga je za dati primer

$$P[z, f(w), B] = P[P(x), f(y), B]s_1.$$

**Kompozicija zamena**  $s_1$  i  $s_2$ , u oznaci  $s_1s_2$  se dobija tako što se

- ♦ primeni  $s_2$  na termine iz  $s_1$ ,
- ♦ dodaju se parovi iz  $s_2$  koji sadrže promenljive koje nisu medju promenljivama u  $s_1$ .

## PRIMER 9.6

$$s_1s_2 = \{g(x, y)/z\} \{A/x, B/y, C/w, D/z\} = \{g(A, B)/z, A/x, B/y, C/w\}$$

Osobine kompozicije zamena:

$$(Ls_1)s_2 = L(s_1s_2)$$

$$(s_1s_2)s_3 = s_1(s_2s_3), \quad \text{asocijativnost}$$

$$s_1s_2 \neq s_2s_1, \quad \text{nekomutativnost}$$

## PRIMER 9.7

$$L = P(x, y), \quad s_1 = \{f(y)/x\}, \quad s_2 = \{A/y\}$$

$$(Ls_1)s_2 = [P(f(y), y)] \{A/y\} = P(f(A), A)$$

$$L(s_1s_2) = [P(x, y)] \{f(A)/x, A/y\} = P(f(A), A)$$

## PRIMER 9.8

$$L(s_1s_2) = [P(x, y)] \{f(A)/x, A/y\} = P(f(A), A)$$

$$L(s_2s_1) = [P(x, y)] \{A/y, f(y)/x\} = P(f(y), A)$$

Ako se zamena  $s$  primenjuje na svaki član skupa  $\{E_i\}$ , dobijene zamene označavamo sa  $\{E_i\}s$ .

Skup  $\{E_i\}$  je **unificibilan** ukoliko postoji zamena  $s$ , takva da je

$$E_1s = E_2s = E_3s = \dots$$

.....

Tada se s naziva **unifikatorom skupa**  $\{E_i\}$ . Unificiran skup se svodi na jednu jedinu vrednost.

#### PRIMER 9.9

Skup  $\{P(x, f(y), B), P(x, f(B), B)\}$   
unificira zamena  $s = \{A/x, B/y\}$ , dajući  $\{P(A, f(B), B)\}$ .

**Najopštiji ili najprostiji unifikator** g skupa  $\{E_i\}$  je ona zamena koja ima sledeće svojstvo:

za proizvoljni unifikator s skupa  $\{E_i\}$ , postoji zamena  $s'$ , takva da je  
 $\{E_i\}_s = \{E_i\}_{s'}$ .

Primenom najopštijeg unifikatora na dati skup dobija se najopštija parcijalna zamena. Najznačajnija osobina najopštijeg unifikatora je njegova jedinstvenost.

#### PRIMER 9.10

$\{x/A\}$  je najopštiji unifikator za  $P(A, y, z)$  i  $P(x, y, z)$ .  
 $\{x/A, y/B, z/C\}$  je manje opšti unifikator i može se dobiti kompozicijom najopštijeg unifikatora i zamene  $\{y/B, z/C\}$ .

#### PRIMER 9.11

| Skup literala                                  | Najopštiji unifikator       |
|------------------------------------------------|-----------------------------|
| $\{P(x), P(A)\}$                               | $P(A)$                      |
| $\{P[f(x), y, g(y)], P[f(x), z, g(x)]\}$       | $P[f(x), x, g(x)]$          |
| $\{P[f(x, g(A, y)), g(A, y)], P[f(x, z), z]\}$ | $P[f(x, g(A, y)), g(A, y)]$ |

### 9.3 PRINCIP REZOLUCIJE

#### Rezolucija propozicionih rečenica

Ako je istinita klauzula  $F$  koja sadrži literal  $f$  i klauzula  $G$  koja sadrži literal  $\sim f$ , može se zaključiti da je istinita klauzula koja je unija polaznih klauzula bez komplementarnog para.

$$\frac{\begin{array}{ll} F & f \in F \\ G & \sim f \in G \end{array}}{\{F - \{f\}\} \cup \{G - \{\sim f\}\}}$$

Rezultujuća klauzula je **rezolventa**.

### PRIMER 9.12

|           |         |
|-----------|---------|
| 1. {P,Q}  | 1. {P}  |
| 2. {¬P,R} | 2. {¬P} |
| 3. {Q,R}  | 3. {0}  |

Rezolucija je jeadn oblik generalizacije modus ponensa. U to se možemo uveriti na osnovu narednog primera.

### PRIMER 9.13

|           |           |
|-----------|-----------|
| 1. {¬P,Q} | 1. P => Q |
| 2. {P}    | 2. P      |
| 3. {Q}    | 3. Q      |

rezolucija modus ponens

### Rezolucija u opštem slučaju

Neka su F i G dve klauzule. Ako postoji literal f u F i ¬g u G, takvi da f i g imaju najopštiji unifikator s, tada možemo da izvedemo (zaključimo) klauzulu koja se dobija primenom zamene s na uniju F i G bez komplementarnog para, odnosno

|                        |        |
|------------------------|--------|
| F                      | f ∈ F  |
| G                      | ¬g ∈ G |
| {F - {f}} ∪ {G - {¬g}} |        |

gde je f s = g s.

### PRIMER 9.14

|                    |                                           |
|--------------------|-------------------------------------------|
| 1. {P(x),Q(x,y)}   | primenjuje se najopštiji unifikator {A/x} |
| 2. {¬P(A),R(B,z)}  |                                           |
| 3. {Q(A,y),R(B,z)} |                                           |

Primena rezolucije može dovesti do različitih rezolvenata.

### PRIMER 9.15

|                                                          |  |
|----------------------------------------------------------|--|
| 1. {P(x,x),Q(x),R(x)}                                    |  |
| 2. {¬P(A,z),¬Q(B)}                                       |  |
| 3. {Q(A),R(A),¬P(A,z)}, f=P(x,x), g=P(A,z), s= {A/x,A/z} |  |
| 4. {P(B,B),R(B),¬P(A,z)}, f=Q(x), g=Q(A,B), s= {B/x}     |  |

## 9.4 PRIMENA REZOLUCIJE

Najjednostavnija upotreba rezolucije je za dokazivanje neispunjivosti (nezadovoljivosti). Znak neispunjivosti je pojava rezolvente – predstavljene praznim skupom, tj. skupom bez literala.

Problem određivanja da li je neka rečenica semantička posledica (logički sledi iz) zadatog skupa formula, može se svesti na problem određivanja neispunjivosti klauzula.

Neka se želi dokazati da  $G$  logički implicira formulu  $g$ . Ovo postizemo ako pronadjemo dokaz za  $g$  iz  $G$ , tj da važi

$$G \vdash g$$

Po **teoremi opovrgavanja**, za ovaj dokaz je potrebno pokazati da je

$$G \cup \{\sim g\}$$

neispunljivo.

### Teorema opovrgavanja (kontradikcije)

Ako je  $G \cup \{g\}$  kontradiktorno (neispunljivo), tada važi  $G \vdash \sim g$ .

Rezimirajmo postupak tzv. opovrgavanja rezolucijom:

- 1. Korak** - negira se ciljna formula  $g$  i dodaje polaznom skupu premisa  $G$ , čime se dobija novi skup  $G'$ .
- 2. Kopak** -  $G'$  se prevodi u klauzalnu formu.
- 3. Korak** - Primenjuje se postupak rezolucije na dobijeni skup klauzulai iz  $G'$ .
- 4. Korak** - Ako se dobije kao rezolventa prazan skup, tada je  $G'$  neispunljivo, odakle proizilazi da  $G$  logički implicira  $g$ .

### PRIMER 9.16

Dokazati pomoću rezolucije, koristeći princip opovrgavanja sledeći stav:

1. Oni koji čitaju su pismeni.

$$\forall x [C(x) \Rightarrow P(x)]$$

2. Delfini nisu pismeni.

$$\forall x [D(x) \Rightarrow \sim P(x)]$$

3. Neki delfini su inteligentni.

$$\exists x [D(x) \wedge I(x)]$$

Želimo da dokažemo da li je ili ne tačan iskaz: Neko ko je inteligentan ne zna da čita.

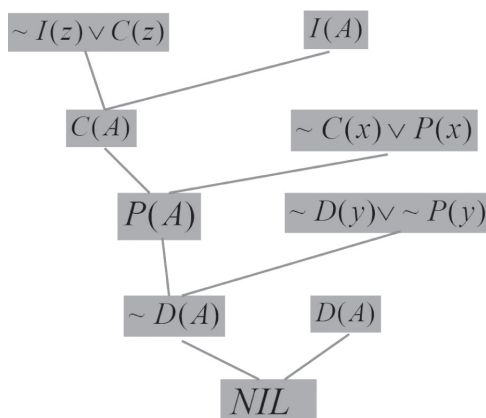
Skup rečenica premisa:

1.  $\sim C(x) \vee P(x)$
2.  $\sim D(y) \vee \sim P(y)$
3.  $D(A), \quad 3a. I(A)$

Negirana ciljna rečenica:

4.  $\sim I(z) \vee C(z)$

Niz rezolucija koje dovode konačno do prazne rezolvente, što je znak protivrečnosti, može se pregledno prikazati tzv. grafom zaključivanja zaključivanja, odnosno **stablom protivrečnosti**, Sl.9.3.



Sl.9.3 Stablo protivrečnosti nastalo primenom postupka opovrgavanja rezolucijom u Primeru 9.16. Primetimo da je u korenu ovog stabla znak protivrečnosti NIL.

Sistem opovrgavanja rezolucijom možemo realizovati i kao produkcionni sistem, čiji je pseudokod dat na Sl.9.4. Globalna baza ovog produkcionnog sistema su klauzule, dok primena pravila na dve odabrane rečenice, implementira postupak rezolucije. Novo stanje globalne baze je prethodno stanje uvećano za novodobijenu rezolventu. Kriterijum zaustavljanja procedure se definiše pojavom znaka protivrečnosti (NIL) u globalnoj bazi. Interesantno je primetiti da je ovaj sistem komutativan sistem produkcije, što znači da strategija upravljanja principijelno može biti i bespovratna, odnosno da se nikada ne moramo vraćati na prethodne alternative.

#### **procedura RESOLUTION**

CLAUSES  $\leftarrow$  S

**until** NIL ne bude u listi CLAUSES, **DO**:

**begin**

**select** dve različite razrešive rečenice

$c_i$  i  $c_j$  iz CLAUSES

izračunati rezolventu  $r_{ij}$  za  $c_i$  i  $c_j$

CLAUSES  $\leftarrow$  skup dobijen unijom  $r_{ij}$  i CLAUSES

**end**

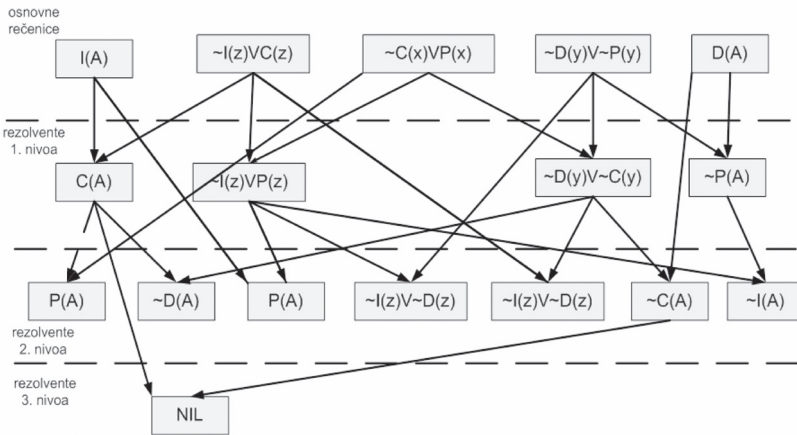
Sl.9.4 Pseudokod procedure RESOLUTION kojom se realizuje osnovni produkcionni sistem za implementaciju postupka opovrgavanja na osnovu rezolucije.

Razmotrićemo neke tipične strategije pretrage.

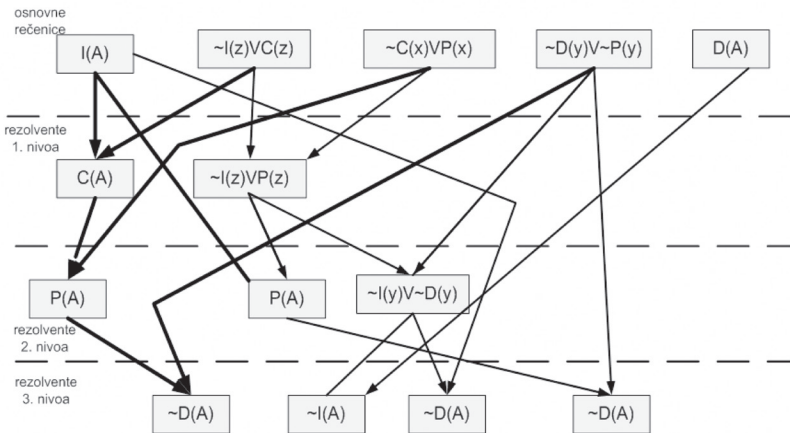
1. Strategije pretrage u širinu: prvo se računaju sve rezolvente prvog nivoa, zatim rezolvente drugog nivo itd., Sl.9.5.

Rezolvente prvog nivoa nastaju od rečenica osnovnog skupa, rezolventa i-tog nivoa je ona čiji je najdublji roditelj rezolventa i-1 nivoa.

2. Strategije osnovnog skupa: osnovni skup – potomci negirane ciljne pdf. Sl.9.6. U svakoj rezoluciji barem jedan roditelj je iz osnovnog skupa. Ukoliko protivrečnost postoji , ona se može dokazati ovom strategijom i efikasnija je od pretrage u širinu. Posедуje elemente reverznog sistema produkcije, pošto rezolventa čiji je barem jedan roditelj iz osnovnog skupa nosi u sebi elemente cilja, odnosno predstavlja neki podcilj.



Sl.9.5 Ilustracija strategije pretrage u širinu produkcionog sistema RESOLUTION pri radu na PRIMERU 9.16.



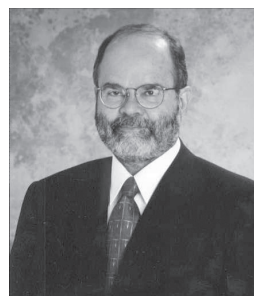
Sl.9.6 Ilustracija strategije pretrage osnovnog skupa produkcionog sistema RESOLUTION pri radu na PRIMERU 9.16.



Osim navedenih, postoje i mnoge druge strategije pretraga za ovaj produkcijski sistem, budući da se on nalazi u osnovi gotovo svih savremenih sistema zasnovanih na znanju, kao i drugih sistema gde je komponenta automatskog rezonovanja od važnosti.

## 9.5 SISTEMI ZA DOBIJANJE KONSTRUKTIVNIH ODGOVORA ZASNOVANI NA REZOLUCIJI

Da bismo upotreabili rezoluciju za dobijanje odgovora na pitanje korišćenjem znanja reprezentovanog rečenicama u računu predikata prve vrste, moramo uraditi više od dokazivanja teorema. Npr. pretpostavimo da treba da dokažemo teoremu  $(\exists x)W(x)$ . Dokaz teoreme bi nas uverio da postoji neko  $x$  za koje važi  $W(x)$ , ali koje je to konkretno  $x$ , na to ne bi smo dobili odgovor. Brzo se možemo uveriti, da nam je za ovaj konkretan odgovor potrebno da sačuvamo zamene promenljivih koje smo koristili tokom postupka dokazivanja. Čuvanje zamena se može obaviti uvođenjem tzv **literala odgovora**  $Ans(x_1, x_2, \dots, x_n)$  uz svaku klauzulu koja se pojavljuje od negacije ciljne rečenice koju treba dokazati. Varijable u  $Ans$  literalu će biti sve one koje se pojavljuju u negiranoj ciljnoj klauzuli. Termovi koji se koriste za zamenu varijabli iz literala  $Ans$  će upravo biti instance egzistencijalno kvantifikovanih promenljivih iz ciljne rečenice. Ovaj postupak je prvi predložio Cordell Green 1969. godine, Sl.9.7, američki naučnik sa Stanford Research Institute.



Sl.9.7 Cordell Green sa Stanford Research Institute, je prvi u svojim radovima iz 70-tih godina prošlog veka predložio metod za dobijanje odgovora na osnovu automatskog dokazivanja teorema metodom opovrgavanja rezolucijom.

Odgovor se dobija sledećom procedurom.

Formira se literal odgovora na otvoreno pitanje  $f$  u obliku  $Ans(x_1, x_2, \dots, x_n)$ , gde su promenljive  $x_1, x_2, \dots, x_n$  slobodne promenljive u  $f$ .

Formira se disjunkcija od  $\sim f$  i literala odgovora.

Ova konstrukcija se prevede u klauzalnu formu i doda početnim premisama (bazi).

Primeni se rezolucija, ali sa kriterijumom zaustavljanja pojave rezolvente koja sadrži samo literal odgovora.

### PRIMER 9.17

$R(x, \text{Sloba})$  je rečenica kojom se postavlja pitanje ko je Slobin roditelj, ukoliko predikat  $R(x, y)$  označava relaciju (roditelj, deca).

Aca je Slobin otac. Boba je Milanov otac. Očevi su roditelji. Da li je Aca Slobin roditelj.

|                              |               |
|------------------------------|---------------|
| 1. $\{O(Aca, Sloba)\}$       |               |
| 2. $\{O(Boba, Milan)\}$      |               |
| 3. $\{\sim O(x,y), R(x,y)\}$ |               |
| 4. $\{\sim O(Aca, Sloba)\}$  | negirani cilj |
| 5. $\{R(Aca, Sloba)\}$       | 1,3           |
| 6. $\{R(Boba, Milan)\}$      | 2,3           |
| 7. $\{\sim O(Aca, Sloba)\}$  | 3,4           |
| 8. $\{ \}$                   | 4,5           |
| 9. $\{ \}$                   | 1,7           |

### PRIMER 9.19

Aca je Slobin otac. Boba je Milanov otac. Očevi su roditelji. Ko je Slobin otac.

|                                   |               |
|-----------------------------------|---------------|
| 1. $\{O(Aca, Sloba)\}$            |               |
| 2. $\{O(Boba, Milan)\}$           |               |
| 3. $\{\sim O(x,y), R(x,y)\}$      |               |
| 4. $\{\sim R(z, Sloba), Ans(z)\}$ | negirani cilj |
| 5. $\{R(Aca, Sloba)\}$            | 1,3           |
| 6. $\{R(Boba, Milan)\}$           | 2,3           |
| 7. $\{\sim O(w, Sloba), Ans(w)\}$ | 3,4           |
| 8. $\{ Ans(Aca) \}$               | 4,5           |
| 9. $\{ Ans(Aca) \}$               | 1,7           |

Odgovor se dobija u 7.i 8. koraku.

## 9.6 HORNOVE REČENICE

Opovrgavanje rezolucijom je i valjana i kompletna procedura zaključivanja. Usled kompletnosti smo sigurni, da ukoliko neki stav  $\omega$  logički sledi iz  $\Delta$ , primenom procedure opovrgavanja rezolucijom, taj stav možemo i da dokažemo. Međutim, ako  $\omega$  ne sledi logički iz  $\Delta$ , procedura opovrgavanja rezolucijom može raditi beskonačno dugo bez zaustavljanja i donošenja konačnog rešenja. Stoga ne možemo koristiti rezoluciju kao sveobuhvatnu proceduru odlučivanja. Šta više, pokazuje se da ne postoji ni neka druga procedura koja će nam uvek potvrditi da neka pdf  $\omega$  logički ne sledi iz  $\Delta$ , kada  $\omega$  zaista poseduje to svojstvo. Usled ove činjenice, kažemo da je račun predikata prve vrste polu odlučiv ili semi odlučiv, što ga čini potencijalno netraktabilnim.

Međutim, u praksi je situacija još gora. Čak i za situacije u kojima će se procedura opovrgavanja rezolucijom završiti u konačnom broju koraka, ona je po kompleksnosti NP kompletna, kao i bilo koja druga valjana i kompletna procedura zaključivanja u

računu predikata prve vrste. Stoga se praktično može reći da je ovaj metod zaključivanja netraktabilan za vrlo kompleksne probleme.

Poboljšanja ovih principijelnih nedostataka idu u sledećim pravcima:

1. Stvaranje efikasnijih sistema rezonovanja žrtvovanjem valjanosti (tj. dopuštaju se i neistiniti zaključci u retkim situacijama)
2. Žrtvovanje kompletnosti za račun efikasnijih procedura rezonovanja (znajući unapred da se za neke istinite stavove neće pronaći dokaz).
3. Ograničavanjem jezika predikata u kome se radi, npr. radi samo sa tzv. Horno-vim rečenicama.

**Hornove rečenice** su one koje imaju najviše jedan pozitivan literal.

#### PRIMERI 9.20

1.  $P$
2.  $\sim P \vee Q$
3.  $\sim P \vee \sim Q$
4.  $\sim p \vee \sim R$ .

Logičar Alfred Horn (1951) je prvi istraživao ovaj tip rečenica, Sl.9.8. Postoje tri tipa ovih rečenica

1. Atomi – često nazivani *činjenice* ( $P$ )
2. Implikacije često nazivane *pravila*. Antecedenta se sastoji od konjukcije pozitivnih literala, a konsekvanta od jednog pozitivnog literala ( $P \wedge Q \Rightarrow R$ )
3. Skup negativnih literala napisanih u formi impikacija sa antecedentom u formi konjukcije pozitivnih literala i prazne konsekvente. Dobija se po pravilu kada se negira ciljna pdf koja je u formi konjukcija pozitivnih literala. Takve rečenice se nazivaju *ciljne* ( $P \wedge Q \Rightarrow$ ).

Važan rezultat za propozicione Hornove rečenice je da za njih postoji algoritam za ključivanja linearne kompleksnosti.

Rezonovanje u kome se polazi od ciljne rečenice, ulančavanjem preko pravila, i dolazi do činjenica se naziva rezonovanje sa ulančavanjem unazad (backwar chaining).

U sistemima sa ulančavanjem unapred (forward chaining), pravila su primenljiva ako se literali iz antecedenti mogu unificirati sa činjenicama, dajući kao konsekvente nove činjenice. Ulančavanje se nastavlja dok se ne postigne ciljna pdf.



Sl.9.8 Alfred Horn (1918-2001), američki matematičar, pionir logičkog programiranja.

Proces rezonovanja sa ulančavanjem unapred je sličan našem prirodnom rezonovanju u kome ulogu činjenica igra tzv. kratkotrajna memorija ili radna memorija, a pravilima odgovara tzv. dugotrajna memorija.

## 9.7 EKSPERTSKI SISTEMI

### 9.7.1 ARHITEKTURA EKSPERTNIH SISTEMA

Cilj istraživanja u oblasti ekspertnih sistema (ES) sastoji se u razradi programa (sistema) koji pri rešavanju teško rešivih zadataka, koji tipično zahtevaju eksperte, daju rezultate po kvalitetu i efikasnosti bliske rezultatima eksperata.

U većini slučajeva ES rešavaju teško formalizovane zadatke, koji nemaju algoritamsko rešenje.

Fajgenbaumova definicija ove oblasti (1977): *Rešavanje metoda veštačke inteligencije teških praktičnih problema koji zahtevaju znanje eksperata*. Stoga se ova oblast često naziva i inženjerija znanja (knowledge engineering).

Razlozi priličnog praktičnog uspeha ES proizilaze iz činjenice da su pri sintezi ovih sistema korišćeni mnogi rezultati postignuti u domenu veštačke inteligencije (VI). Navešćemo tri osnovna rezultata:

1. Snaga ES je prevashodno određena snagom baze znanja i mogućnošću njene dopune, a tek na drugom mestu korišćenim metodama ili procedurama. Ovo je u suprotnosti sa ranijim stavovima u VI po kojima je izvor intelektualnosti uzak sklop opštih snažnih procedura zaključivanja. Praksa je međutim pokazala da je važnije imati raznovrsna specijalizovana znanja, a ne opšte procedure zaključivanja.
2. Znanja koja u najvećoj meri doprinose efikasnom i kvalitetnom rešenju nekog problema su u osnovi
  - ♦ heuristička
  - ♦ eksperimentalna
  - ♦ neodređena
  - ♦ verovatnosna.

Uzrok ovome je neformalizovanost ili salaba formalizovanost zadataka koji se rešavaju. Osim toga, znanja eksperata imaju individualni karakter.

3. Imajući u vidu neformalizovanost problema, kao i heuristički i lični karakter korišćenih znanja, korisnik (odnosno ekspert) mora da poseduje mogućnost neposredne interakcije sa ES u formi dijaloga.

Neposredna posledica činjenice da osnovnu snagu ES čine znanja, ES nužno mora da poseduje sposobnost ovladavanja znanjem (knowledge acquisition).

Proces ovladavanja znanjem može se podeliti na

- ♦ dobijanje znanja od eksperata
- ♦ organizaciju znanja koja obezbeđuje efikasan rad sistema
- ♦ predstavljanje znanja u formi koja je razumljiva sistemu.

Dobijanje znanja se ostvaruje analizom rada eksperata koji rašavaju zadate realne probleme od strane tzv. inženjera znanja (knowledge engineer). Heuristički karakter ovih znanja čini postupak dobijanja ovih znanja veoma teškim. Stoga, ovaj postupak je tipično usko grlo u sintezi jednog ES.

Napomenimo vrlo važnu činjenicu da se u sistemima VI i ES se rešavaju, po pravilu, neformalizovani problemi, odnosno VI i ES ne odbacuju i ne zanemaruju tradicionalne metode u sintezi programa orjentisanih na rešavanje formalizovanih problema.

Parafrizirajući Njuela i Sajmona u neformalizovane (ill-structured) zadatke svrstamo one koji imaju jednu od sledećih karakteristika:

- ♦ Problem ne može biti zadat u numeričkoj formi.
- ♦ Ciljevi ne mogu biti formulisani preko neke precizno definisane ciljne funkcije.
- ♦ Ne postoji algoritamsko rešenje zadatka.
- ♦ Algoritamsko rešenje postoji, ali ga nije moguće postići usled ograničenih resursa (vreme, memorija).

Neformalizovani problemi poseduju tipično, sledeća svojstva:

- ♦ Pogrešni, nejednoznačni, nepotpuni i protivrečni polazni podaci.
- ♦ Pogrešna, nejednoznačna, nepotpuna i protivrečna znanja o problemskoj oblasti i o zadatku koji se rešava.
- ♦ Velika dimenzija prostora rešenja, odnosno, pretraga pri rešavanju zadatka je veoma velika.
- ♦ Dinamički promenljivi podaci i znanja.

Imajući u vidu prva dva osnovna teorijska principa, ES obuhvataju dve komponente:

- ♦ Sistem zaključivanja,
- ♦ Bazu znanja.

Treći praktični princip, nameće ES sledeće zadatke:

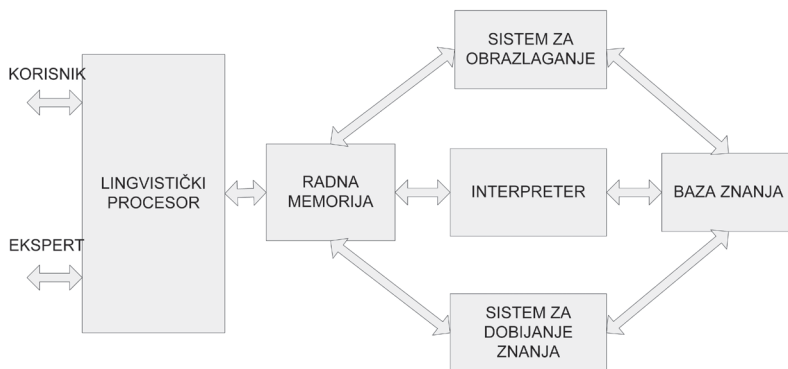
1. Mogućnost vođenja dijaloga o zadatku koji se rešava na jeziku koji je prilagodjen korisniku (ili ekspertu) i posebno dobijanju novih znanja u toku dijaloga.
2. Mogućnost praćenja, u toku rešavanja zadatka, linije rasudjivanja jasne korisniku (ekspertu).
3. Sposobnost objašnjavanja rasudjivanja na jeziku pogodnom za korišćenje od strane korisnika (eksperta).

Prvi zahtev se realizuje lingvističkim procesorom ES i komponentom dobijanja znanja. Drugi i treći zahtev se realizuju uvođenjem komponente za obrazlaganje.

Tipičan ES ima sledeće komponente, Sl.9.9:

1. Baza znanja – memoriše skup pravila.
2. Radna memorija – memoriše podatke.
3. Interpreter – rešava zadati problem na osnovu znanja kodovanih u ES.
4. Lingvistički procesor, kojim se ostvaruje dijalog sa korisnikom i ekspertom na prirodnom jeziku.

5. Komponenta za dobijanje znanja.
6. Komponenta za obrazlaganje, koja daje objašnjenja za preduzete korake i odgovore na pitanja zašto je neki zaključak izveden ili opovrgnut.



Sl.9.9 Arhitektura tipičnog ekspertnog sistema

ES radi u dva režima:

- ♦ izvalačenje znanja iz eksperata
- ♦ rešavanje konkretnih problema.

U tradicionalnom pristupu, prvoj fazi odgovara algoritimizacija i programiranje od strane programera, dok u ES izradu programa ne vrši programer, nego ekspert koji u opštem slučaju ne mora da zna programiranje.

Lingvistički procesor – transformiše ulazne podatke predstavljene na ograničenom prirodnom jeziku u predstavu na unutrašnjem jeziku sistema i obrnuto, transformiše poruke sistema izražene na unutrašnjem jeziku u saopštenja na ograničenom prirodnom jeziku.

Interpreter - na osnovu ulaznih podataka, produkcionih pravila i opštih činjenica o problemskoj oblasti, formiraju rešenje zadatka.

Sistem za obrazlaganje – saopštava:

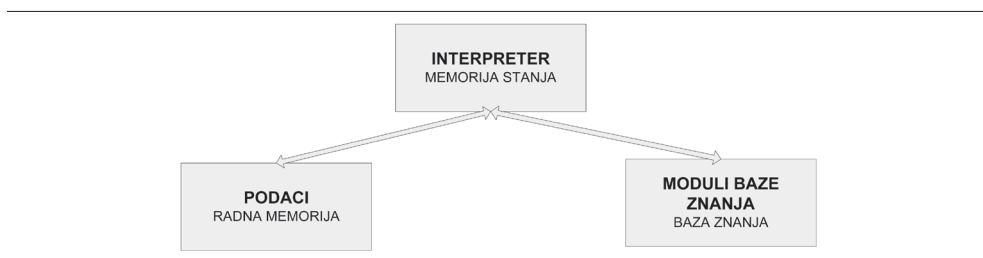
- ♦ kako pravila koriste informacije korisnika,
- ♦ zašto su upotrebljena (neupotrebljena) data pravila,
- ♦ koji su zaključci izvedeni.

Sva saopštenja se daju u ograničenom prirodnom jeziku.

## 9.7.2 UPRAVLJANJE EKSPERTNIM SISTEMOM (INTERPRETER)

Osnovna razlika u upravljanju tradicionalnim programskim sistemima i ES:

- ♦ različiti moduli se pozivaju ne po imenu, nego na osnovu opisa situacije
- ♦ međusobna interakcija i redosled modula se formira u toku rešavanja zadatka i ne može biti formulisana unapred.



Sl.9.10 Klasična šema upravljanja ES

Zadatak interpretera – da se na osnovu tekućeg stanja radne memorije odredi koji modul i sa kojim podacima će biti aktivan u tekućem ciklusu rada sistema, Sl.9.10.

Nakon završetka rada tekućeg modula, interpreter proverava uslove završetka zadatka, i ukoliko oni nisu ispunjeni, izvršava se naredni ciklus.

Moduli baze znanja ES su obično realizovani u obliku programa i pravila.

Svaki modul poseduje opis koji ukazuje pri kojim uslovima ovaj modul može pristupiti izvršenju.

U svakom ciklusu interpreter izvršava četiri operacije:

- ♦ formiranje uzoraka modula (odabiranje)
- ♦ poredjenje (provera uslova izvršavanja)
- ♦ razrešavanje konflikta
- ♦ izvršenje

Često se ove funkcije grupišu u prepoznavanje (odabiranje, poredjenje, razrešavanje konflikta) i izvršavanje.

Sa teorijske tačke gledišta rad interpretera zavisi samo od sadržaja radne memorije i baze znanja. Informacija o radu interpretera se pamti u tzv. memoriji stanja interpretera.

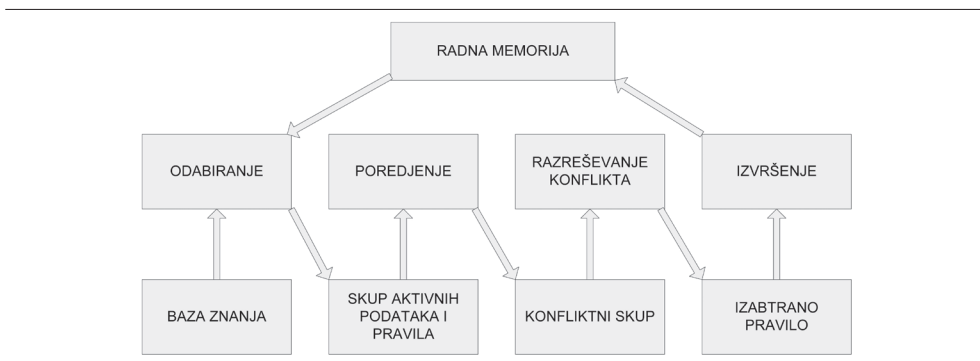
U opštem slučaju, na svakom koraku rada, interpreter koristi tri izvora znanja:

- ♦ radna memorija
- ♦ baza znanja
- ♦ memorija stanja interpretera.

Odabiranje – određuje se podskup baze znanja i radne memorije, koji mogu biti korišćeni u datom ciklusu. Često se jedno odabiranje vrši za više narednih ciklusa.

Pri realizaciji odabiranja se uglavnom koriste dva pristupa:

- ♦ sinksično odabiranje- kojim se vrši grubi odbir znanja, koja će se koristiti u datom ciklusu. Osnovu znanja pri odabiranju čine formalna (sintaktička) znanja ugrađena u fazi razvoja sistema.
- ♦ semantičko odabiranje – kojim se vrši odabir znanja na osnovu podataka kao što su: model predmetne oblasti, razbijanje zadatka na podzadatke, tekući ciljevi itd. Semantička znanja koja se koriste u ovom pristupu, se unose u sistem od strane eksperta putem metapravila.



Sl.9.11 Ciklusi rada interpretera

Uobičajeno je da se u toku svakog ciklusa vrši odabiranje i podataka i pravila. Odabiranjem podataka sistem fokusira svoju pažnju na ciljeve (hipoteze), a odabiranjem pravila određuje način obrade podataka.

Razlikuju se dva tipa odabiranja:

- ♦ prosto (podaci i pravila se razmatraju kao jednonivovske strukture)
- ♦ hijerarhijsko

Kod hijerarhijskog odabiranja objekti (pravila, podaci) se razbijaju na hijerarhijske podskupove – klase. Odabiranje se u ovom slučaju sastoji u izboru jedne od klasa na osnovu metapravila. Pri tome klase mogu biti disjunktne ili preklapajuće.

## Rezime 9. poglavlja

- ♦ Osnovni cilj ovog poglavlja je prikaz metodološke osnove sistema automatskog rezonovanja, kao osnove šire klase sistema VI, počevši od automatskog dokazivanja teorema, dobijanja odgovora na pitanja upućena sistemu koji ima konstituisanu bazu znanja iz neke predmetne oblasti, automatsko programiranje, ekspertski sistemi i td.
- ♦ Rezolucija je procedura zaključivanja zasnovana na samo jednom efikasnom pravilu izvođenja. Pored jednostavnosti njena dobra osobina je i valjanost i u širem smislu kompletnost.
- ♦ Budući da rezolucija operiše nad klauzalnom formom rečenica računa predikata prve vrste prikazan je opšti postupak svodjenja bilo kog skupa rečenica na ovu formu.
- ♦ Postupak unifikacije omogućava poredjenje i uparivanje delova rečeničkih iskaza koji sadrže promenljive, tako da se one izjednačuju nakon uvođenja odgovarajućih smena. Unifikacija pretstavlja centralni deo procedure automatskog rezonovanja zasnovan na rezoluciji.
- ♦ Izložen je osnovni postupak automatskog rezonovanja primenom rezolucije. Zasniva se na jednom od bazičnih rezultata logike, po kome negirana ciljna rečenica (zaključak) nakon negacije i priključenja premisama, ukoliko je tačna



(logički sledi iz premisa) mora dati nakon sukcesivne primene rezolucije, rezolventu – prazan skup, ili što je ekvivalentno (NIL).

- ♦ Osim ove primene rezolucije, prikazan je i postupak dobijanja tzv. konstruktivnih odgovora, kod kojih nije bitno samo to da li neka rečenica logički sledi iz datih premisa, već se zahteva i za koju konkretnu zamenu promenljivih se dobija ova logička posledica. Ova zamena promenljivih je ujedno i traženi konkretni odgovor na postavljeno pitanje.
- ♦ Hornove rečenice predstavljaju strukturno uprošćavanje rečenica koje nam stoje na raspolaganju u modelovanju datog problema i rezonovanju nad njim. Dobra osobina Hornovih rečenica je znatno smanjena kompleksnost procedure automatskog zaključivanja zasnovanog na rezoluciji. Hornove rečenice su našle direktnu primenu u konstrukciji jezika PROLOG, jednog od osnovnih programskih jezika tzv. logičkog programiranja.
- ♦ Poglavlje se završava kratkim pregledom ekspertskih sistema, podoblasti VI direktno oslonjene na automatsko rezonovanje. Ekspertski sistemi su dostigli zrelost industrijske primene i kao takvi su jedan od najvećih dometa simboličkog pristupa veštačkoj inteligenciji.

## Pitanja i zadaci

1. Koji od navedenih parova izraza se mogu unificirati? Nadjite najopštiji unifikator za svaki od takvih parova.
  - a)  $P(x, B, B), P(A, y, z)$
  - b)  $P(g(f(v)), g(u)), P(x, x)$
  - c)  $P(x, f(x)), P(y, y)$
  - d)  $P(y, y, B), P(z, x, z)$
  - e)  $2+3=x, x=3+3$
2. Konvertovati sledeće izraze u klauzalnu formu:
  - a)  $((\exists x)[P(x)] \vee (\exists x)[Q(x)] \Rightarrow (\exists x)[P(x) \vee Q(x)])$
  - b)  $(\forall x)[P(x) \Rightarrow (\forall y)[(\forall z)[Q(x, y)] \Rightarrow \sim(\forall z)[R(y, x)]]]$
  - c)  $(\forall x)[P(x)] \Rightarrow (\exists x)[(\forall z)[Q(x, z)] \vee (\forall z)[R(y, x, z)]]]$
3. Dokazati postupkom opovrgavanja rezolucijom da  $(\exists x)w(x)$  logički sledi iz  $[P(A1) \wedge P(A2)]$ .
4. Mika, Pera i Laza su slonovi. O njima znamo sledeće činjenice:
  - a) Mika je ružičast
  - b) Pera je siv i voli Lazu
  - c) Laza je ili ružičast ili siv (ali ne i jedno i drugo) i voli Miku.
 Upotrebom opovrgavanja rezolucijom dokazati da sivi slon voli ružičastog slona, odnosno dokazati  $(\exists x, y)[Siv(x, y) \wedge Ružičast(y) \wedge Voli(x, y)]$ .

# 10. Uvod u mašinsko učenje

Učenje je proces koji se kod ljudi odvija gotovo neprekidno. Postoje različite vrste učenja, od čistog memorisanja podataka, preko učenja motornih vještina, do sticanja sposobnosti analitičkog i kreativnog mišljenja. Kako se sposobnost učenja smatra osnovnom odlikom inteligentnih bića, to nije iznenađujuće što je mašinsko učenje jedna od centralnih oblasti istraživanja u veštačkoj inteligenciji. Veštački sistemi koji su sposobni da uče evidentno poboljšavaju svoje performanse, a biološki sistemi povećavaju verovatnoću svog opstanka i produžetka vrste.

Mašinsko učenje je oblast koja proučava procese na kojima se zasniva učenje kod ljudi i kod veštačkih sistema. U cilju obuhvatanja svih relevantnih aspekata, oslanja se na veliki broj drugih disciplina, uključujući veštačku inteligenciju, verovatnoću i statistiku, teoriju informacija, psihologiju i neurobiologiju, teoriju upravljanja, filozofiju itd.

## Definicija 10.1

Za jedan sistem VI (računarski program) kažemo da uči zadatu klasu zadataka  $T$  na osnovu iskustva  $E$  i zadate mere performansi  $P$ , ako se njegove performanse za rešavanje zadataka iz klase  $T$ , poboljšavaju sa iskustvom  $E$ .

Da bi jedan problem učenja bio dobro definisan neophodno je identifikovati tri komponente:

- ♦ Klasu zadataka ( $T$ )
- ♦ Meru performanse koju treba poboljšati ( $P$ )
- ♦ Izvor iskustva ( $E$ )

U Tabeli 10.1 dat je pregled nekih uspešnih sistema mašinskog učenja.

|                                                                                      |
|--------------------------------------------------------------------------------------|
| ♦ Prepoznavanje izgovorenih reči<br>Sphinx-II, Aplov Siri, Google-411, Deep Learning |
| ♦ Učenje vožnje autonomnog vozila<br>ALVINN, Google Car, BRAIVE                      |
| ♦ Učenje klasifikacije novih astronomskih struktura                                  |
| ♦ Učenje igara (dame, šah, tavlja, poker)                                            |

Tabela 10.1 Neke od oblasti u kojima je mašinsko učenje dalodobre rezultate

## PRIMER 10.1

### Problem učenja igre dame:

- ♦ Zadatak T: igra dame
- ♦ Mera performanse P: procenat dobijenih igara
- ♦ Izvor iskustva E: igranje protiv samog sebe.

### Problem učenja prepoznavanja rukom pisanih znakova:

- ♦ Zadatak T: prepoznavanje i klasifikacija rukom pisanih reči u zadatim slikama
- ♦ Mera performanse P: procenta tačno klasifikovanih reči
- ♦ Izvor iskustva E: baza rukom pisanih reči sa tačnom klasifikacijom (obučavajući skup)

### Obučavanje robota vožnji automobila:

- ♦ Zadatak T: vožnja automobila na javnim autoputevima sa četiri trake korišćenjem vizuelnih senzora
- ♦ Mera performanse P: prosečna dužina vožnje bez greške
- ♦ Izvor iskustva E: sekvenca slika i primenjenih komandi, snimljenih tokom vožnje ljudskog vozača.

U savremenoj oblasti mašinskog učenja dominiraju sledeći pristupi:

- ♦ induktivno učenje;
- ♦ analiticko učenje (analogija sa logikom);
- ♦ učenje na slučajevima (case-based learnig – analogija sa ljudskim pamćenjem);
- ♦ neuralne mreže (analogija sa neurobiologijom);
- ♦ genetski algoritmi (analogija sa evolucijom);
- ♦ hibridni modeli (kombinacija više pristupa).

Nezavisno od pristupa i primenjenih tehnika učenja, važno pitanje je i šta se uči. U okviru VI dominiraju sledeće računске strukture, za koji postoji poseban interes kao objekta učenja:

- ♦ funkcije (funkcionalna preslikavanja, izlazno-ulazna preslikavanja)
- ♦ logički programi i skupovi pravila

- ♦ automati sa konačnim unutrašnjim stanjima
- ♦ sistemi za rešavanje problema.

Od svih navedenih oblika učenja, od najvećeg značaja za tekuću praksu u domenu računarstva i veštačke inteligencije predstavlja induktivno učenje, koje se često proširuje u nazivu na induktivno empirijsko učenje. Suština ovog tipa učenja je učenje na osnovu raspoloživih primera, ili kako bi to rekli u svakodnevnom jeziku, učenje iz iskustva (drugih). S obzirom na objekat učenja najopštije je učenje funkcionalnih, odnosno ulazno-izlaznih preslikavanja. U cilju razjašnjenja osnovnih koncepata mašinskog učenja, zadržaćemo se isključivo na induktivnom učenju funkcionalnih preslikavanja na osnovu zadatih primera. Razumevanje ove uže klase problema mašinskog učenja pruža čitaocu mogućnost za razumevanje i ostalih oblika učenja, u kojima se varira kako objekat učenja, tako i same tehnike, izvori iskustva i ciljevi učenja.

## 10.1 INDUKTIVNO EMPIRIJSKO UČENJE FUNKCIONALNIH PRESLIKAVANJA

Na Sl.10.1 dat je prikaz osnovnih elemenata sistema induktivnog učenja nepoznatog preslikavanja. Ulazi  $X$  su po pravilu  $n$  dimenzioni vektori, koji se susreću u literaturi iz domena mašinskog učenja pod različitim nazivima: **ulazni vektori**, **vektori oblika**, **vektori obeležja**, **uzorci**, **primeri** i **instance**. Komponente  $x_i$  ulaznih vektora se nazivaju **obeležja**, **atributi**, **ulazne varijable** i **komponente**. Ove komponente mogu biti tri različite prirode: realni brojevi npr.  $x_i = 0.34$ , diskretni brojevi, npr.  $x_i \in \{0, 1, 2, 3\}$  i kategorijalne varijable, npr. komponenta  $x_i$  znači boju, koja može uzeti vrednosti  $\{\text{crven}, \text{plav}, \text{zelen}\}$ . Kategorijalne varijable mogu biti **uredjene** kao npr.  $\{\text{mali}, \text{srednji}, \text{veliki}\}$ , ili **neuredjene**, kao prethodni primer sa bojama.

Izlazni prostor može biti:

- ♦ kategorijalan sa  $K$  distiktnih vrednosti, kada obučeni sistem obavlja **klasifikaciju**, **prepoznavanje** ili **kategorizaciju**. Sam izlaz se naziva **oznaka**, **klasa**, **kategorija** ili **odluka**,
- ♦ realan,  $Y=R$ , kada obučeni sistem realizuje **regresiju** (**funkcionalni estimator**).
- ♦ vektorski, sa komponentama koje mogu biti realne ili kategorijalne varijable.

Obučavanje se vrši na osnovu obučavajućeg skupa  $D$ , koji se sastoji od  $N$  ulazno-izlaznih parova. Celokupan proces obučavanja možemo posmatrati kao pretragu u prostoru hipoteza  $H$ , u cilju izbora najpogodnije hipoteze  $g$ , koja je u saglasnosti sa obučavajućim skupom. Postoji više načina na koji se obučavajući skup može koristiti:

- ♦ **batch metod** – celokupan obučavajući skup je na raspolaganju i kao celina se koristi u izračunavanju hipoteza  $h$ ,
- ♦ **inkrementalni metod** – koristi se sekvencijalno, po jedna instanca obučavajućeg skupa, koja modifikuje tekuću hipotezu  $h$ ,
- ♦ **on line metod** – korišćenje jedne instance obučavajućeg skupa u trenutku kada ona postane dostupna.

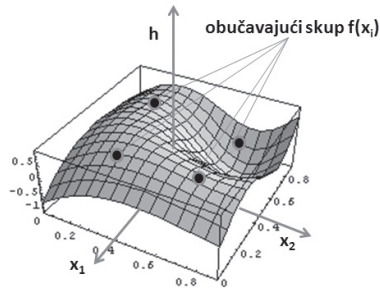


Sl.10.1 Principijelni dijagram ključnih elemenata u induktivnom učenju funkcionalnih preslikavanja:  $f$  – nepoznato preslikavanje,  $D$  – obučavajući skup ulazno – izlaznih parova,  $H = \{h\}$  – skup hipoteza unutar koga biramo finalnu hipotezu  $g \approx h$  posredstvom algoritma obučavanja.

Pitanje merenja kvaliteta obučenog sistema je jedno od ključnih pitanja mašinskog učenja. Lako možemo zamisliti da je izabrana hipoteza  $g$  najbolja moguća u odnosu na dati obučavajući skup, ali da se pokazuje vrlo lošom na novim primerima koji nisu učestvovali u obučavanju. Stoga je opšteprihvaćena mera kvaliteta jednog sistema mašinskog učenja njegova **spособnost generalizacije**, odnosno ponašanje sistema na instancama koje nisu vidjene u fazi obučavanja. Stoga tipično sintezu prate dva skupa, obučavajući -trening i test skup. Test skup se koristi samo za procenu generalizacionih sposobnosti sintetisanog sistema i ne sme se koristiti u fazi obučavanja. Uobičajene numeričke mere performansi su ili srednjekvadratna greška između izlaza sistema i ciljnih vrednosti ili ukupan broj grešaka.

Sada možemo postaviti suštinsko pitanje: kako je nešto uopšte moguće naučiti? Na Sl.10.2 je prikazana hiperpovršina koja prolazi kroz četiri zadate dvodimenzione tačke. Zašto bi neka procedura obučavanja odabrala ovu hiperpovršinu kao rezultat obučavanja na osnovu obučavajućeg skupa od zadate četiri tačke? Takvih površina ima beskonačno mnogo. Vrlo rano se u teorijskom razvoju mašinskog učenja shvatilo da je neophodno apriori uvesti neka ograničenja na prostor hipoteza u kome se traži finalno rešenje. Ova vrsta apriornih informacija se zove **pomeraj ili bias**. Obučavanje od nekog praktičnog značaja nije moguće bez biasa. Postoje dve vrste biasa:

- ♦ **apsolutni bias** (bias kojim se uvodi neka restrikcija u prostoru hipoteza) – npr. ograničavanjem na manji podskup,
- ♦ **preferencijalni bias** – hipoteze se uređuju u odnosu na neki kriterijum uređjenja, a zatim se u procesu obučavanja bira ona hipoteza koja minimizira ovaj kriterijum. Npr. ako je kriterijum uređjenja kompleksnost hipoteza, u procesu obučavanja će biti izabrana hipoteza minimalne kompleksnosti koja se dobro ponaša na obučavajućem skupu (ima najmanju grešku). U ovom svetlu, **princip Okamovog rezača**, po kome se u nauci preferiraju jednostavnija objašnjenja u odnosu na kompleksnija, može se smatrati preferencijalnim biasom.



Sl.10.2 Hiperpovršina  $f(x_1, x_2)$  koja prolazi kroz četiri tačke.

U praktičnim primenama, susreću se različiti modeli mašinskog učenja u zavisnosti od toga koje informacije su prisutne u obučavajućim skupovima, na koji način se kontroliše proces obučavanja i da li sistem za obučavanje može da menja svet koji ga okružuje. U cilju sticanja opštije slike o oblasti, navešćemo neke najvažnije podklase sistema mašinskog učenja.

U pogledu informacija sadržanih u obučavajućem skupu razlikujemo sledeće vrste sistema obučavanja.

**Obučavanje sa učiteljem (supervised learning)** – obučavajući skup je oblika  $\{x_i, f(x_i)\}$ , gde je  $f(x_i)$  ciljna vrednost vektora obeležja  $x_i$ . Zadatak obučavanja je nalaženje aproksimacije za  $f$ , a mera performanse je kvalitet aproksimacije u tačkama koje ne pripadaju obučavajućem skupu.

**Obučavanje bez učitelja, smooučavanje (unsupervised learning)** – u obučavajućem skupu su prisutne samo ulazne instance  $\{x_i\}$ . Tipičan pristup je **klasterovanje**, odnosno grupisanje raspoloživih podataka u manji broj grupa unutar kojih su podaci sličniji u poredjenju sa podacima iz ostalih grupa. Imenovanjem klastera se dolazi posrednim putem do označenih uzoraka, pošto se podrazumeva da sve instance jednog klastera pripadaju istom konceptu.

**Obučavanje sa podsticanjem (reinforcement learning)** – je složeniji koncept od prethodna dva. Poreklo vodi iz teorije upravljanja u kome je dinamičko okruženje opisano trojkom (stanje, akcija, nagrada). U ovom tipu učenja potrebno je naučiti kako vršiti preslikavanja situacija u akcije a da se pri tome maksimizira nagrada. Za razliku od obučavanja sa učiteljem algoritmu obučavanja nije rečeno koje akcije da preduzima u datoj situaciji. Dobar primer da se shvati ovaj scenario je učenje igranja šaha. Pozicije na tabli su stanja, akcije su mogući potezi za datu poziciju. Nagrada za izabrani potez je pobeda, i suprotno – kazna je gubitak igre. I nagrada i kazna kasne u odnosu na trenutak izbora akcije, što je tipično za ovu vrstu obučavanja.

Obučavanje sa učiteljem se dalje može podeliti na:

- ♦ induktivno i
- ♦ transduktivno.
- ♦ Kod **induktivnog učenja**, cilj je da se sintetisani sistem dobro ponaša na svim primerima dobijenim iz iste raspodele kao i obučavajući skup.

Kod **transduktivnog učenja** test primeri (bez ciljne klasifikacije) se znaju već u trenutku obučavanja. Cilj sintetisanog sistema je da se dobro ponaša samo na tim test primercima.

## PRIMER 10.2

Zamislamo da nam je cilj sinteza klasifikatora budućih email poruka koje ćemo primati, na osnovu obučavajućeg skupa koga čine ručno klasifikovane do sada primljene email poruke. Ovaj scenario odgovara induktivnom učenju, budući da ne posedujemo buduće email poruke u trenutku obučavanja sistema.

Pretpostavimo da posedujemo 5 miliona dokumenata u personalnom računaru, a od toga je samo njih hiljadu ručno klasifikovano u kategorije koje su za nas lično interesantne. Zamislamo da želimo da napravimo sistem za automatsku klasifikaciju i arhiviranje svih 5 miliona dokumenata, u što je moguće boljem skladu sa hiljadu ručno klasifikovanih primera. Ovaj scenario potpuno odgovara transduktivnom pristupu, budući da su nam sva dokumenta koja treba da arhiviramo i klasifikujemo, već prisutna u fazi obučavanja.

U pogledu na mogućnost uticaja na okruženje, obučavanje može biti

- ♦ pasivno i
- ♦ aktivno.

U scenariju **pasivnog obučavanja**, na raspolaganju su samo uzorci iz obučavajućeg skupa.

U scenariju **aktivnog obučavanja**, obučavajući sistem ima do izvesne mere mogućnost usmeravanja učenja. Npr. sistem može samostalno da generiše uzorke i da zahteva od učitelja njihovo označavanje (membership query model). U modelu tzv. selektivnog odabiranja, obučavajući sistem observira primere date od učitelja, a zatim odlučuje za koje od njih će zahtevati označavanje (tačnu klasifikaciju). Ovaj model ima velikog praktičnog smisla u situacijama gde je jeftino doći do primera, ali je skupo njihovo označavanje.

Pokazuje se da pod određenim uslovima aktivno obučavanje, za isti nivo tačnosti klasifikacije, zahteva eksponencijalno manje označenih primera u odnosu na pasivno obučavanje.

## Induktivni transfer

Osobina ljudskog načina učenja je da ne učimo izolovane zadatke, već istovremeno ili sekvencijalno učimo više zadataka. Npr. u dečijem uzrastu istovremeno učimo i motoriku i govor, socijalne odnose, vizuelnu scenu i td.

Induktivni transfer je model obučavanja koji pokušava da zahvati ovu vrstu ljudskog učenja. U sintezi obučavajućih sistema iz ove klase, znanja dobijena prilikom učenja jednog zadatka se pokušavaju iskoristiti prilikom učenja nekog drugog zadat-



ka. Stoga je glavna tačka ovih sistema pitanje tipa znanja koji se može transferisati sa jednog zadatka obučavanja na drugi. Da ovo nije trivijalan zadatak, možemo videti iz sledećeg primera transfer učenja. Zamislimo obučeni sistem koji dobro klasifikuje ulazne slike na dve kategorije: na slici je mačka ili na slici nema mačke. Kako se ovo znanje može iskoristiti (transferovati) za sintezu sistema koji treba da kategoriše ulazne slike na kategorije: na slici je sto ili na slici nema stola?

Interesantno je da je jedan mogući način transfera znanja učenje zajedničke reprezentacije ulaznih primera za sve zadatke koji nas interesuju. Ovo je u izvesnom smislu uticalo na potpuno novu i najatraktivniju oblast mašinskog učenja danas: **duboko obučavanje** (Deep Learning) praćeno odgovarajućim arhitekturama obučavajućih sistema, kao što su **duboke neuronske mreže** (Deep Neural Networks) ili **duboke mreže uverenja** (Deep Belief Networks). Ovim sistemima se danas postižu spektakularni rezultati u domenu prepoznavanja govora, slike i videa i bukvalno ih čitamo kao vesti sa prvih stranica svetskih medija.

## 10.2 INDUKTIVNO UČENJE STABALA ODLUČIVANJA

Ukoliko pretpostavimo da se prostor hipoteza sastoji od objekata strukture stabla dolazimo do koncepta induktivnog empirijskog učenja stabala odlučivanja, kao jedne od veoma važnih oblasti mašinskog učenja.

Stabla odlučivanja se sastoje od čvorova i grana, koje spajaju čvorove roditelja sa čvorovima naslednika - dece. Čvor koji se nalazi na vrhu stabla (koren) nema roditelja, dok svi ostali tzv. unutrašnji čvorovi imaju samo jednog roditelja. Čvorovi koji nemaju naslednike se nazivaju listovi. Listovi predstavljaju sva moguća rešenja koja se mogu dobiti iz datog stabla. To su čvorovi odgovora, a ostali se nazivaju čvorovi odluke.

Postoji veliki broj algoritama induktivnog učenja, od kojih su najpoznatiji: CLS (Hunt et al.), ID3 (Quinlan), ID4 (Schlimmer i Fisher), ID5 (Utgoft), CN2 (Clark i Boswell), BCT (Chan), C4.5 (Quinlan), AQ (Michalski i Larson) i RULES (Aksoy), Sl.10.3.

Stabla odluke klasifikuju primere u dve ili više klasa na osnovu vrednosti atributa kojima su primeri opisani, propuštajući ih niz stablo od korena ka listovima. Na početku se bira atribut čija vrednost najbolje deli raspoložive primere. Svaki čvor u stablu predstavlja test nekog atributa, a svaka grana koja polazi iz tog čvora odgovara jednoj od mogućih vrednosti tog atributa. Na taj način se primeri dele



Sl.10.3 Dvojica najuticajnijih istraživača u oblasti induktivnog učenja na osnovu stabala odlučivanja: Ross Quinlan (ID3, C4.5) (levo) i Ryszard S. Michalski (1937 - 2007) (AQ) (desno).

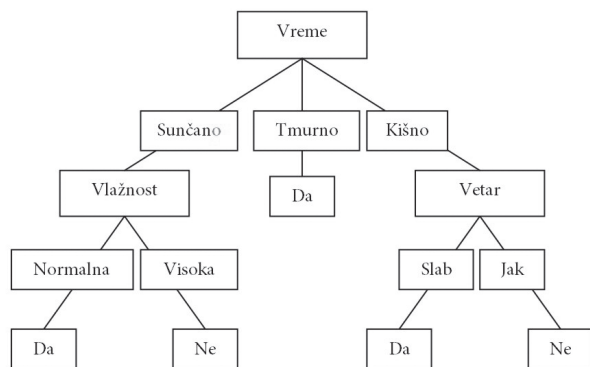


u podskupove, zavisno od vrednosti izabranog atributa. Ako svi atributi u podskupu pripadaju istoj klasi, stablu se dodaje list. Svaki put kroz stablo predstavlja jedno klasifikaciono pravilo, tj. konjunkciju testova atributa, a samo stablo je disjunkcija ovih konjunkcija.

Stabla odluke su značajna za oblast mašinskog učenja, jer su zbog svoje sistematske strukture razumljiva za ljude i predstavljaju odličnu tehnologiju obrazlaganja u automatizovanim dijagnostičkim sistemima. Obrazloženje zašto je doneta neka konkretna odluka se dobija izdvajanjem puta od pripadajućeg lista do korena, pri čemu se iz svakog čvora iščitava razlog parcijalnog izbora. Osim toga, svakom stablu odlučivanja se jednoznačno može pripisati skup pravila tipa: ako – onda (if – then), koja su osnovni gradivni blokovi baza znanja ekspertskih i drugih sistema zasnovanih na znanju. Ovim se otvara mogućnost direktnog punjenja baze znanje pravilima dobijenim induktivnim empirijskim putem na osnovu zadatih primera, bez uticaja eksperta i njegove direktne uključenosti u proces sinteze jednog ekspertskog sistema. Dovoljno je imati reprezentativan skup podataka koji odslikavaju njegovu ekspertizu. Pojava interneta, weba i javno dostupnih baza podataka, značajno su aktualizovali ovu oblast mašinskog učenja, čineći je jednom od najperspektivnijih profesija danas.

### PRIMER 10.3

Na Sl.10.4 prikazan je jedan primer stabla odluke. Ovo stablo klasifikuje subotnja jutra na osnovu toga da li je vreme pogodno za igranje tenisa ili ne. Čvorovima odgovaraju atributi (vreme, vlažnost, vetar), dok listovima odgovaraju klasifikacije (Da – vreme pogodno za igranje tenisa i Ne- vreme nije pogodno za igranje tenisa).



Slika 10.4. Primer stabla odlučivanja za koncept *Igrati Tenis*. Svaka situacija je opisana sa tro-dimenzionim vektorom atributa (vreme, vlažnost, vetar). Broj klasa (koncepta) o kojima se odlučuje je dva (Da, Ne).

Ono što stabla odlučivanja čini tehnikom široko prihvaćenom u praksi je i njihova veza sa pravilima odlučivanja koja dominiraju u simboličkom pristupi veštačkoj inte-

ligenciji. Stablu sa sl.10.4 odgovaraju ekvivalentna pravila vezana za listove, koje ćemo numerisati sa leva na desno:

List 1: **Ako** je: Vreme: *Sunčano* i Vlažnost: *Normalna* **Onda**: IgratiTennis Da

List 2: **Ako** je: Vreme: *Sunčano* i Vlažnost: *Visoka* **Onda**: IgratiTennis Ne

List 3: **Ako** je: Vreme: *Tmurno* **Onda**: IgratiTennis Da

List 4: **Ako** je: Vreme: *Kišno* i Vetar: *Slab* **Onda**: IgratiTennis Da

List 4: **Ako** je: Vreme: *Kišno* i Vetar: *Jak* **Onda**: IgratiTennis Ne.

Postoji više dimenzija na osnovu kojih se stabla odluke mogu razlikovati:

1. Testovi mogu biti multivarijabilni (testiraju više atributa ulaznih oblika odjednom) ili monovarijabilni (testiraju samo jedan atribut).
2. Testovi mogu imati dva ili više ishoda. Ako svi testovi u stablu odluke imaju po dva ishoda, onda je to binarno stablo odluke.
3. Atributi mogu biti kategorički ili numerički. Binarni atributi se formalno mogu trezirati i kao numerički i kao kategorički.
4. Možemo imati dve ili više klasa. Ako imamo dve klase i binarne ulaze, stablo reprezentuje jednu Bulovu funkciju i naziva se Bulovo stablo odluke.

Učenje stabla odluke je jedna od najviše korišćenih praktičnih metoda induktivnog zaključivanja. Najpogodnija su za primenu na problemima sa sledećim karakteristikama:

- ♦ Primeri su predstavljeni vektorima, tj. parovima atribut-vrednost. Opisani su fiksnim skupom atributa, npr. *Temperatura*, i njihovim vrednostima, npr. *Vruće*. Za stabla odlučivanja je najjednostavnija situacija kada svaki atribut obuhvata mali broj disjunktih vrednosti (npr. *Vruće*, *Toplo*, *Hladno*), ali je dozvoljeno i da atributi imaju realne vrednosti, npr. numeričko predstavljanje atributa *Temperatura*.
- ♦ Ciljne funkcije imaju diskretne izlazne vrednosti. U slučaju kontinualnih izlaznih vrednosti moguće je izvršiti transformaciju u diskretne (nominalne) promenljive kroz proces diskretizacije. Vrednosti ciljnih funkcija su linearne logičke kombinacije vrednosti atributa.
- ♦ Podaci za obuku mogu da sadrže greške - šum. Metodi učenja stabala odluke su robusni na greške u klasifikaciji obučavajućih primera, kao i na greške u vrednostima atributa koje opisuju ove primere.
- ♦ Obučavajućim podacima mogu da nedostaju vrednosti atributa. Metodi stabala odluke mogu da se koriste čak i kada neki obučavajući primeri imaju nepoznate vrednosti, npr. ako je *Vlažnost* određenog dana poznata samo za neke obučavajuće primere.

Postoje mnogi praktični problemi koji odgovaraju ovim karakteristikama. Učenje stabala odluke je našlo primenu u rešavanju problema kao što su medicinska dijagnostika, detekcija kvarova na različitim uređajima, procena kreditne sposobnosti podnosioca zahteva za kredit i sl. Takvi problemi, kod kojih je zadatak da se ulazni podaci klasifikuju u jednu od unapred zadatih disjunktih klasa, nazivaju se **klasifikacioni problemi**.

## 10.3 OSNOVNI ALGORITMI UČENJA STABALA ODLUKE

Većina algoritama koji su razvijeni za učenje stabala odluke su varijacije bazičnog algoritma koji koristi od-vrha-naniže (*top-down*), pohlepnu (*greedy*) pretragu kroz prostor mogućih stabala odluke. Ovaj pristup je korišćen kod ID3 algoritma i njegovog naslednika C4.5.

Osnovni algoritam ID3 uči stabla odluke konstruišući ih od vrha naniže, počevši pitanjem: koji atribut treba testirati u korenu stabla?. Da bi se odgovorilo na ovo pitanje, atribut svakog primera se procenjuje pomoću statističkog testa, da bi se odredilo koliko dobro klasifikuju obučavajuće uzorke. Odabere se najbolji atribut i koristi kao test u korenu stabla. Potom se generišu čvorovi naslednici za svaku moguću vrednost ovog atributa, a obučavajući primeri se propuštaju kroz grane i sortiraju na dostignutim čvorovima naslednicima. Proces određivanja najboljeg atributa za testiranje u svakom čvoru nasledniku se ponavlja, koristeći obučavajuće primere koji su dospeli do tog čvora. Na taj način se ustari vrši pohlepna potraga za odgovarajućim stablom odluke. Shodno svojstvu pohlepne pretrage, algoritam se nikada ne vraća u prethodno stanje da bi eventualno preispitao ranije izbore.

## 10.4 TRAŽENJE NAJBOLJEG ATRIBUTA ZA KLASIFIKACIJU

Glavni izbor u ID3 algoritmu je određivanje atributa koji se testirati u svakom čvoru stabla. Stoga nam je potrebna kvantitativna mera koja govori o vrednosti atributa. Definišemo statističku veličinu, informacioni dobitak, koja meri koliko dobro može dati atribut da razdvoji obučavajuće primere na osnovu njihove ciljne klasifikacije. ID3 koristi veličinu informacioni dobitak da među kandidatima odabere najbolji atribut u svakom koraku dok stablo raste.

Da bismo precizno definisali informacioni dobitak, podsetimo se smisla i definicije entropija kao mere informacija i neodređenosti jednog sistema. Za datu klasu S, koja sadrži pozitivne (p) i negativne (n) primere nekog ciljnog koncepta, entropija od S u odnosu na ovu binarnu klasifikaciju je:

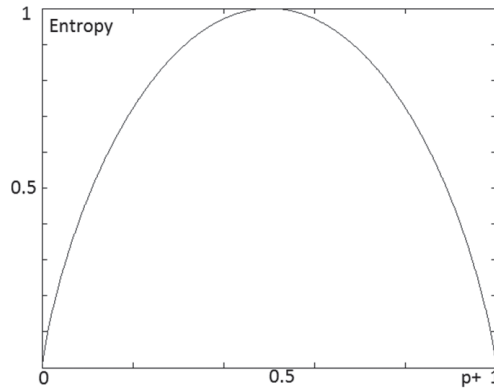
$$Entropy(S) = Entropy([p, n]) = -p_+ \log_2 p_+ - p_- \log_2 p_- ,$$

gde je

$$p_+ = \frac{p}{p+n} \text{ proporcija pozitivnih primera u S, a}$$

$$p_- = \frac{n}{p+n} \text{ proporcija negativnih primera u S, Sl.10.5.}$$

U svim računanjima u vezi sa entropijom uzimamo da je  $0 \cdot \log 0 = 0$ .



Slika 10.5. Funkcija entropije u zavisnosti od  $p_+$ .

#### PRIMER 10.4

Pretpostavimo da je  $S$  klasa od 14 primera nekog binarnog koncepta i sadrži  $p=9$  pozitivnih i  $n=5$  negativnih primera. Usvajamo notaciju  $[9+,5-]$ . Tada je entropija u odnosu na ovu binarnu klasifikaciju

$$Entropy([9+,5-]) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

Primetimo da je entropija 0 ako svi članovi  $S$  pripadaju istoj klasi, bilo pozitivnoj ili negativnoj, kao i da je jednaka 1 ako pozitivna i negativna klasa imaju isti broj primera. Na slici 3. je prikazana zavisnost funkcije entropije kada se  $p_+$  kreće u intervalu od 0 do 1.

Do sada smo diskutovali o entropiji u specijalnom slučaju kada je ciljna klasifikacija binarna. Generalno, ako ciljni atribut može uzeti jednu od  $c$  različitih vrednosti, onda entropiju definišemo kao:

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

gde je  $p_i$  proporcija primera u  $S$  koji pripadaju klasi  $i$ .

Pošto je entropija mera nehomogenosti u klasama obučavajućih primera, sada možemo da definišemo meru efektivnosti atributa pri klasifikaciji obučavajućih primera. Mera koju ćemo koristiti, **informacioni dobitak (information gain)**, je očekivana redukcija entropije koju izaziva particija primera na osnovu ovog atributa. Preciznije, informacioni dobitak za slučaj da se atribut  $A$  koristi za razvrstavanje primera iz  $S$ , tj. kao koren stabla, je:

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

gde je  $Values(S)$  skup svih mogućih vrednosti atributa  $A$ , a  $S_v$  podskup od  $S$  za koji atribut  $A$  ima vrednost  $v$ , tj.  $S_v = \{s \in S \mid A(s)=v\}$ . Prvi izraz na desnoj strani jednačine

je entropija originalne klase primera  $S$ , a drugi izraz je očekivana vrednost entropije nakon što je  $S$  podeljeno uz pomoć atributa  $A$ , tj. suma entropija svakog podskupa  $S_v$  pomnožena sa udelom  $|S_v|/|S|$  primera koji priradaju  $S_v$ .  $\text{Gain}(S,A)$  je u stvari informacija o vrednosti ciljne funkcije za poznatu vrednost atributa  $A$ . Vrednost  $\text{Gain}(S,A)$  se može interpretirati i kao broj bita potrebnih za kodovanje ciljne vrednosti određenog elemeta iz  $S$ , kada znamo vrednost atributa  $A$ .

### ID3(*Primeri*, *Ciljni\_atribut*, *Atributi*)

*Primeri* su obučavajući primeri. *Ciljni\_atribut* je atribut čiju vrednost stablo odluke treba da predvidi. *Atributi* je lista ostalih atributa koje može da testira stablo odluke. Vraća stablo odluke koje korektno klasifikuje *Primeri*.

- ♦ Kreiraj čvor *Koren*
- ♦ Ako su svi *Primeri* pozitivni, vrati stablo *Koren* sa jednim čvorom i *oznaka* = +
- ♦ Ako su svi *Primeri* negativni, vrati stablo *Koren* sa jednim čvorom i *oznaka* = -
- ♦ Ako je *Atributi* prazno, vrati stablo *Koren* sa jednim čvorom i *oznaka* = najčešća vrednost *Ciljni\_atribut* u *Primeri*
- ♦ Inače **počni**
  - $A \leftarrow$  atribut iz *Atributi* koji najbolje\* klasifikuje *Primeri*
  - Atribut odluke za *Koren*  $\leftarrow A$
  - Za svaku moguću vrednost  $v_i$  od  $A$ 
    - Dodaj novu granu stablu ispod *Koren*, koja odgovara testu  $A=v_i$
    - Neka *Primeri*  $v_i$  bude podskup od *Primeri* koji za  $A$  ima vrednost  $v_i$
    - Ako je *Primeri*  $v_i$  prazno
      - Onda ispod ove nove grane dodaj list i *oznaka* = najčešća vrednost *Ciljni\_atribut* u *Primeri*
      - Inače ispod ove nove grane dodaj podstablo
- ♦ ID3(*Primeri*  $v_p$ , *Ciljni\_atribut*, *Atributi*\_{ $A$ })
- ♦ **Kraj**
- ♦ Vraća *Koren*

\*Najbolji atribut je onaj koji ima najveći informacioni dobitak

Sl.10.6 Sažeti ID3 algoritam specijalizovan za funkcije sa binarnim vrednostima atributa. Ovaj proces se nastavlja sve dok stablo ne klasifikuje primere idealno, ili dok ne iskoristi sve attribute.

## PRIMER 10.5

Pretpostavimo da je  $S$  skup obučavajućih primera - dana, opisanih atributima među kojima je i *Vetar* (uzima vrednosti *Slab* ili *Jak*). Kao i ranije, pretpostavimo da je  $S$  klasa od 14 primera, [9+,5-]. Od ovih 14 primera, pretpostavimo da je kod 6 pozitivnih i 2 negativna *Vetar*=*Slab*, a kod ostalih je *Vetar*=*Jak*. Informacioni dobitak na osnovu sortiranja originalnih 14 primera po atributu *Vetar* se može izračunati na sledeći način:

$Vrednosti(Vetar) = Slab, Jak$

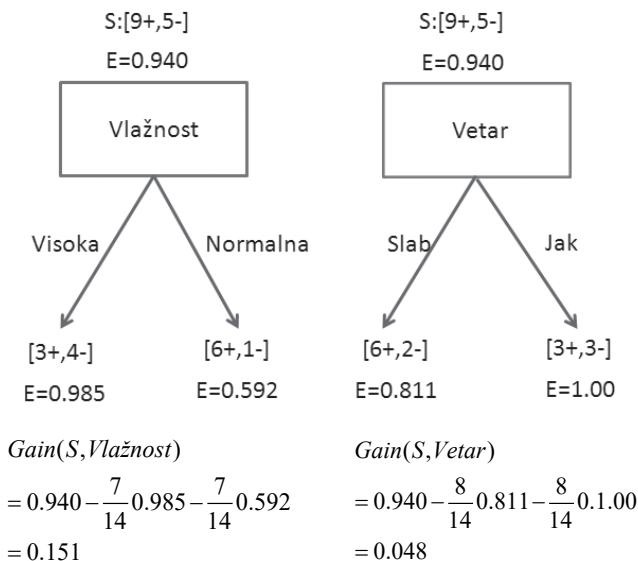
$S = [9+, 5-]$

$S_{Slab} \leftarrow [6+, 2-]$

$S_{Jak} \leftarrow [3+, 3-]$

$$\begin{aligned}
 Gain(S, Vetar) &= Entropy(S) - \sum_{v \in \{Slab, Jak\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
 &= Entropy(S) - \frac{8}{14} Entropy(S_{Slab}) \\
 &\quad - \frac{6}{14} Entropy(S_{Jak}) \\
 &= 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1.00 \\
 &= 0.048
 \end{aligned}$$

Informacioni dobitak je veličina koju ID3 algoritam koristi da odredi najbolji atribut u rastućem stablu. Na Sl.10.7 je prikazano kako se koristi informacioni dobitak da bi se odredila relevantnost atributa. Informacioni dobitak dva različita atributa *Vlažnost* i *Vetar* se računa da bismo odredili koji od njih je bolji za klasifikaciju obučavajućih primera prikazanih u Tabeli 10.2.



Slika 10.7 Koji atribut daje bolji klasifikator?

*Vlažnost* pruža veći informacioni dobitak od *Vetar* u odnosu na ciljnu klasifikaciju. Na Sl 10.5  $E$  predstavlja  $Entropy(S)$ . Za datu klasu  $S$  od 9 pozitivnih i 5 negativnih primera, nakon sortiranja po atributu *Vlažnost* dobijamo klase  $[3+, 4-]$  (*Vlažnost*=*Visoka*) i  $[6+, 1-]$  (*Vlažnost*=*Normalna*). Informacioni dobitak prilikom ovakve podele je 0.151, u odnosu na informacioni dobitak kod atributa *Vetar* od samo 0.048.

Da bismo ilustrovali rad algoritma ID3, razmatraćemo zadatak prikazan u Tabeli 10.2. Ovde je ciljni atribut *IgratiTenis*, koji može imati vrednosti *Da* i *Ne* za različita subotnja jutra.

| Dan | Vreme   | Temperatura | Vlažnost | Vetar | IgratiTenis |
|-----|---------|-------------|----------|-------|-------------|
| D1  | Sunčano | Vruće       | Visoka   | Slab  | Ne          |
| D2  | Sunčano | Vruće       | Visoka   | Jak   | Ne          |
| D3  | Tmurno  | Vruće       | Visoka   | Slab  | Da          |
| D4  | Kišno   | Toplo       | Visoka   | Slab  | Da          |
| D5  | Kišno   | Hladno      | Normalna | Slab  | Da          |
| D6  | Kišno   | Hladno      | Normalna | Jak   | Ne          |
| D7  | Tmurno  | Hladno      | Normalna | Jak   | Da          |
| D8  | Sunčano | Toplo       | Visoka   | Slab  | Ne          |
| D9  | Sunčano | Hladno      | Normalna | Slab  | Da          |
| D10 | Kišno   | Toplo       | Normalna | Slab  | Da          |
| D11 | Sunčano | Toplo       | Normalna | Jak   | Da          |
| D12 | Tmurno  | Toplo       | Visoka   | Jak   | Da          |
| D13 | Tmurno  | Vruće       | Normalna | Slab  | Da          |
| D14 | Kišno   | Toplo       | Visoka   | Jak   | Ne          |

Tabela 10.2 Obučavajući primeri za ciljni koncept *IgratiTenis*.

Pogledajmo prvi korak algoritma, u kome se kreira najviši čvor stabla. Postavlja se pitanje koji je to atribut koji treba da testiramo prvi u stablu odluke. ID3 određuje informacioni dobitak za svaki kandidovani atribut, tj. za *Vreme*, *Temperatura*, *Vlažnost* i *Vetar*, potom selektuje onaj sa najvećim informacionim dobitkom. Računanje informacionog dobitka za dva od ovih atributa je prikazano na slici 10.7. Vrednosti informacionog dobitka za sve attribute iznose:

$$Gain(S, Vreme) = 0.246$$

$$Gain(S, Vlažnost) = 0.151$$

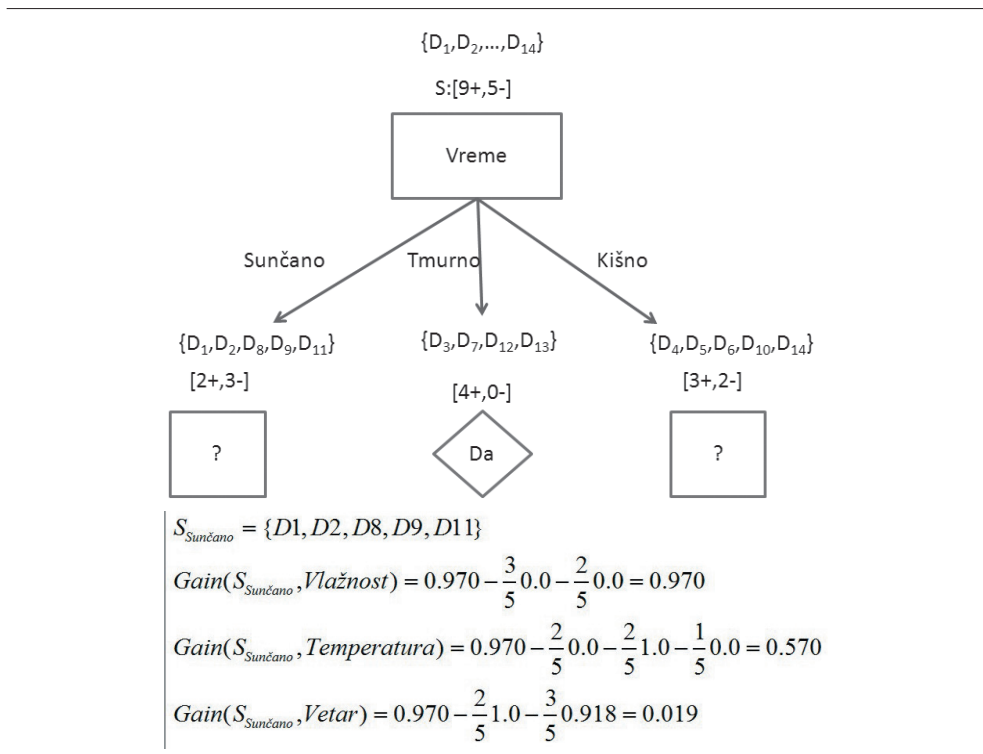
$$Gain(S, Vetar) = 0.048$$

$$Gain(S, Temperatura) = 0.029$$

gde *S* predstavlja klasu obučavajućih primera iz Tabele 10.2.

Atribut *Vreme* za zadate obučavajuće primere daje najbolju predikciju ciljnog atributa *IgratiTenis*. Stoga je *Vreme* selektovano kao odlučujući atribut za koren, a grane se postavljaju ispod korena za svaku od njegovih mogućih vrednosti, tj. *Sunčano*, *Tmurno* i *Kišno*. Rezultujuće nedovršeno stablo odluke je dato na Sl.10.8. zajedno sa obučavajućim primerima sortiranim u svakom sledećem novom čvoru. Primetimo da je svaki primer za koji je *Vreme=Tmurno* takođe pozitivan primer za *IgratiTenis*. Stoga ovaj čvor stabla postaje list sa klasifikacijom *IgratiTenis=Da*. Nasuprot tome, sledejući čvorovi od *Vreme=Sunčano* i *Vreme=Kišno* još uvek nemaju nultu entropiju, usled čega stablo odluke mora i dalje da se razvija ispod ovih čvorova.

Proces selektovanja novog atributa i podela klase obučavajućih primera se ponavlja za svaki sledeći neterminalni čvor, ovaj put koristeći samo obučavajuće primere koji se odnose na taj čvor. Atributi koji su već iskorišćeni u stablu se isključuju, tako da se svaki atribut može pojaviti samo jednom u stablu odluke duž bilo kog puta. Ovaj proces se nastavlja za svaki novi list dok se ne ispuni jedan od sledećih uslova: (1) svaki atribut je već iskorišćen duž ovog puta kroz stablo, ili (2) svi obučavajući primeri koji se odnose na ovaj list imaju istu vrednost ciljnog atributa, tj. entropija je nula. Slika 10.6 ilustruje izračunavanje informacionog dobitka za sledeći korak u rastućem stablu odluke. Krajnje stablo odluke koje je ID3 algoritam naučio iz 14 obučavajućih primera iz Tabele 10.1. je dato na Sl.10.4.



Slika 10.8 Delimično naučeno stablo odluke

## 10.5 UČENJE STABALA ODLUKE U PROSTORU HIPOTEZA (VERSION SPACE)

Posmatrajmo proces nalaženja stabla odluke za zadate obučavajuće primere kao pretragu u prostoru svih hipoteza (stabala). Kako se može pronaći traženo stablo odluke? Jedan način bi bio da se pronađu sva moguća stabla za određeni skup atributa i njihove vrednosti, a zatim da se izabere najbolje stablo. Ovaj postupak nije praktičan, jer broj mogućih stabala može biti jako veliki. Drugi mogući pristup bio bi:



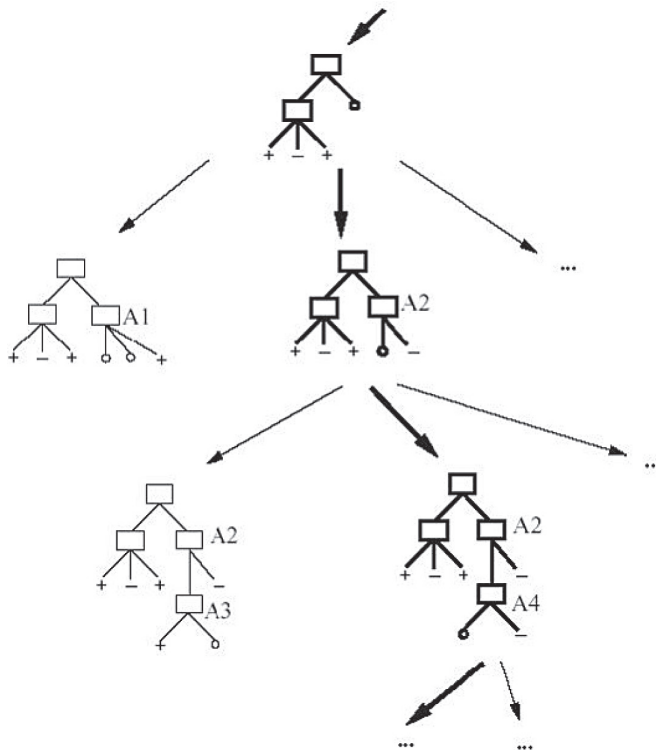
1. Nađi sva moguća stabla koja klasifikuju zadati skup primera tačno.
2. Izabрати najjednostavnije stablo.

I ovaj pristup je velike računarske kompleksnosti.

Rekapitulirajmo sada kako ID3 algoritam za dve klase, radi:

1. Ako svi primeri pripadaju istoj klasi, kreiraj list.
2. Inače, nađi najbolji atribut A, dodaj granu za svaku vrednost atributa A, rasporedi primere u podskupove.
3. Ako su svi primeri klasifikovani tačno, stop.
4. Inače, primeni korake 1-3 za listove.

Kao i za druge induktivne metode, za ID3 se može reći da pretražuje prostor hipoteza da bi došao do one koja odgovara obučavajućim primerima. Prostor hipoteza koji ID3 pretražuje je skup mogućih stabala odluke. ID3 izvodi pretragu krećući se od jednostavnijih ka složenijim stablima kroz ovaj prostor hipoteza. Startuje se od praznog stabla, a nastavlja sa razmatranjem soženijih hipoteza u potrazi za stablom odluke koje tačno klasifikuje sve obučavajuće primere. Kriterijumska funkcija performanse koja usmerava ovu pretragu je informacioni dobitak. Ilustrativni primer ove vrste pretrage je prikazana na slici 10.9.



Sl.10.9 Pretraga prostora hipoteza - ID3 algoritam pretražuje prostor mogućih stabala odluke od najjednostavnijih do rastuće složenih, vođen heuristikom informacionog dobitka.

Posmatranjem ID3 algoritma u pogledu strategije pretrage prostora hipoteza, možemo steći neke uvide u njegove sposobnosti i odgraničenja:

- ♦ Prostor hipoteza svih stabala odluke je potpun prostor konačnih funkcija diskretnih vrednosti. Zato što svaku diskretnu funkciju može da predstavi stablom odluke, ID3 izbegava jedan od najvećih rizika kod metoda koje pretražuju nekompletne prostore hipoteza - mogućnost da prostor hipoteza ne sadrži ciljnu funkciju.
- ♦ ID3 održava samo jednu trenutnu hipotezu u toku pretraživanja prostora stabala odluke. Održavajući samo jednu hipotezu, ID3 gubi mogućnosti koje proizilaze iz eksplicitnog predstavljanja svih konzistentnih hipoteza.
- ♦ ID3 u svojoj izvornoj formi ne može da se vrati na prethodne alternative u pretrazi. Kada jednom selektuje atribut koji se testira na određenom nivou stabla, nikada se ne vraća da preispita taj izbor. Stoga je podložan uobičajenim rizicima HC pretraga bez vraćanja - konvergiranje ka lokalno optimalnim rešenjima, koja nisu i globalno optimalna.
- ♦ Da bi doneo odluke, ID3 koristi sve obučavajuće primere u svakom koraku pretrage. Ovo je potpuno suprotno metodama koje odluke donose inkrementalno na bazi individualnih obučavajućih primera. Prednost korišćenja statističkih osobina svih primera (npr. informacionog dobitka) je ta što su rezultati pretrage mnogo manje osetljivi na greške u individualnim obučavajućim primerima. ID3 se lako može modifikovati režimu rada sa zašumljenim obučavajućim podacima, tako što se modifikuje njegov kriterijum završetka rada po kome se prihvataju i hipoteze koje ne moraju biti idealno konzistentne sa obučavajućim podacima.

## 10.6 INDUKTIVNI POMERAJ (BIAS)

Induktivna pomeraj (*bias*) je skup pretpostavki koje zajedno sa obučavajućim primerima opravdavaju klasifikacije budućih primera, koji nisu učestvovali u fazi obučavanja. Ili preciznije, induktivni pomeraj su restrikcije koje namećemo prostoru hipoteza, smatrajući zadati algoritam obučavanja uspešnim ukoliko je pronašao najbolju hipotezu u prostoru hipoteza uz istovremeno zadovoljavanje nametnutih restrikcija.

Za skup obučavajućih primera postoji veliki broj stabala odluke koja su konzistentna sa ovim primerima. Opis induktivnog pomeraja algoritma ID3 se stoga sastoji u opisu pomeraja na osnovu koga on bira jednu od konzistentnih hipoteza među svim ostalima. ID3 bira prvo prihvatljivo stablo odluke na koje naiđe u svojoj pretrazi. Grubo govoreći, strategija pretraživanja metode ID3 (a) bira kraća stabla pre nego duža, i (b) bira stabla kod kojih su atributi sa najvećim informacionim dobitkom postavljani najbliže korenu. Zbog malih, ali važnih interakcija između heuristike za selekciju atributa koju ID3 koristi i konkretnih obučavajućih primera sa kojima se susreće, teško je precizno definisati njegov induktivnu pomeraj. Pomeraj možemo približno definisati kao davanje prednosti kratkim stablima odluke nad kompleksnim stablima.

Zamislamo algoritam koji počinje prazim stablom i pretražuje po širini (*breadth-first*) progresivno ka kompleksnijim stablima, razmatrajući prvo sva stabla dubine 1, zatim 2 i tako redom. Kada pronađe stablo odluke konzistentno sa obučavajućim podacima, vraća najmanje stablo koje je konzistentno na toj dubini pretrage (npr. stablo sa najmanjim brojem čvorova). Nazovimo BFS-ID3 ovakav algoritam koji pretražuje po širini. BFS-ID3 pronalazi najkraće stablo odluke i time precizno primenjuje pomeraj “kraća stabla imaju prednost nad većima”. ID3 se može posmatrati kao efikasna aproksimacija algoritma BFS-ID3, koja koristi pohlepnu heurističku pretragu u pokušaju da pronađe najkraće stablo bez sprovođenja čitave pretrage po širini kroz prostor hipoteza.

Pošto ID3 koristi heuristiku informacionog dobitka i HC strategiju pretrage, on pokazuje složeniji pomeraj nego BFS-ID3. Konkretno, on ne pronalazi najkraće stablo odluke svaki put, i naklonjen je stablima kod kojih su atributi sa najvećim informacionim dobitkom postavljeni što bliže korenu.

Induktivna pomeraj ID3 algoritma je prednost koja se daje nekim hipotezama u odnosu na druge (npr. kraće hipoteze), bez striktnih ograničenja na hipoteze koje kasnije mogu biti generisane. Ova vrsta pomeraja se naziva **pomeraj prednosti** ili, alternativno, **pomeraj pretrage**. Nasuprot tome, postoje algoritmi čiji je pomeraj u vidu kategoričke restrikcije na skup razmatranih hipoteza. Takva vrsta pomeraji se naziva **restriktivni pomeraj**.

Obično je pomeraj prednosti poželjniji od restriktivnog pomeraja, jer dozvoljava obučavajućem sistemu da radi u celokupnom prostoru hipoteza, koji sigurno sadrži ciljnu funkciju. Restriktivni pomeraj, sa druge strane, striktno ograničava skup potencijalnih hipoteza, pa se javlja mogućnost da nepoznata ciljna funkcija nije u domašaju obučavajućeg sistema.

Da li je činjenica, da induktivna pomeraj algoritma ID3 daje prednost kraćim stablima odluke, razumna osnova za generalizaciju van obučavajućih primera? Filozofi i naučnici koji se bave tom oblašću su diskutovali o ovom pitanju vekovima, ali je za sada debata ostala nerešena. Vilijem Okam je jedan od prvih koji su razmatrali ovo pitanje oko 1320. godine, pa se ovaj pomeraj često naziva i Okamov rezač. Okamova rezač znači da za objašnjenje nekog fenomena treba uvesti što je moguće manje pretpostavki, eliminišući, tj. odsecajući kao rezačem, one pretpostavke koje ne doprinose predviđanjima hipoteze ili teorije. Kada više različitih teorija ima jednaku mogućnost predviđanja, princip preporučuje da se uvede što je moguće manje pretpostavki.

Jedan od argumenata za primenu ovog principa je da postoji manje kratkih nego dugačkih hipoteza, pa je manja verovatnoća da će biti pronađena kratka hipoteza koja slučajno odgovara obučavajućim podacima. Na taj način se izbegavaju koincidencije i greške do kojih one dovode. Nasuprot tome, postoji veliki broj kompleksnih hipoteza koje odgovaraju trenutnim obučavajućim podacima, ali ne mogu da se generalizuju za naredne podatke. Problem kod primene Okamovog rezača je taj što je veličina hipoteze određena konkretnom unutrašnjom reprezentacijom obučavajućeg sistema. Dva sistema sa različitim unutrašnjim reprezentacijama mogu doći do različitih hipoteza, a da oba mogu da opravdaju svoje kontradiktorne zaključke Okamovim rezačem. Stoga možemo razmišljati i o odbacivanju Okamovog rezača. Medjutim razmotrimo sledeći

scenario gde se ispituje koje unutrašnje reprezentacije mogu proizaći iz procesa evolucije i prirodne selekcije. Zamislamo populaciju veštačkih agenata kreiranih u simuliranom procesu evolucije, koji obuhvata reprodukciju, mutaciju i prirodnu selekciju agenata. Pretpostavimo da ovi procesi evolucije mogu da menjaju sistem opažanja kod agenata iz generacije u generaciju, čime je omogućena promena unutrašnjih atributa preko kojih opažaju svoj svet. Pretpostavimo da agenti koriste fiksni algoritam učenja, recimo ID3, koji se ne menja tokom evolucije. Razumno je pretpostaviti da će tokom vremena evolucija proizvesti unutrašnje reprezentacije koje će ove agente činiti sve uspešnijim u njihovoj okolini. Ako uspeh agenta značajno zavisi od njegove sposobnosti da precizno generalizuje, očekujemo da će evolucija razviti unutrašnje reprezentacije koje dobro rade sa bilo kojim algoritmom učenja i induktivnim pomerajem. Ako neka vrsta agenata primenjuje algoritam učenja čiji je induktivni pomeraj Okamov rezač, onda očekujemo da evolucija stvori unutrašnje reprezentacije za koje je Okamov rezač uspešna strategija. Suština argumenta ovde je da će evolucija stvoriti unutrašnje reprezentacije koje induktivni pomeraj algoritma učenja čine neizbežnim, jer evolucija lakše menja reprezentaciju nego što bi menjala algoritam učenja.

## 10.7 PRAKTIČNI PROBLEMI UČENJA STABALA ODLUKE

Praktični problemi koji se javljaju kod stabala odluke obuhvataju:

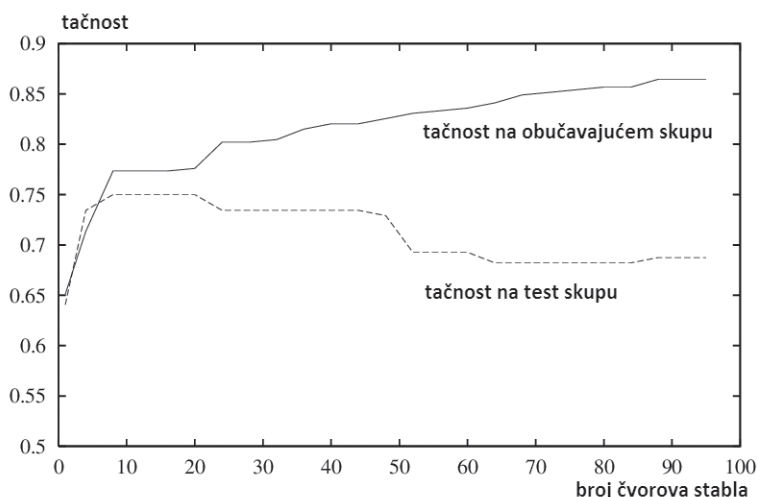
- ♦ određivanje koliko duboko treba razviti stablo odluke,
- ♦ obrađivanje kontinualnih atributa,
- ♦ odabir odgovarajuće mere za selekciju atributa,
- ♦ rad sa obučavajućim podacima kojima nedostaju vrednosti atributa,
- ♦ rad sa atributima koji imaju promenljive troškove,
- ♦ unapredjenje efikasnosti računanja.

Posebno ćemo razmatrati svako od ovih pitanja kao i modifikacije osnovnog ID3 algoritma kojim se ovi problemi rešavaju ili ublažuju. Tako dobijen rezultujući sistem preimenovan je u algoritam poznat pod nazivom C4.5.

### 10.7.1 Izbegavanje preprilagodjenosti (overfitting)

Algoritam prikazan na Sl.10.6 pušta svaku granu stabla da raste upravo toliko duboko da idealno klasifikuje obučavajuće primere. Iako je ovo ponekad razumna strategija, ona u stvari može da dovede do poteškoća kada su podaci zašumljeni, ili kada je broj obučavajućih primera suviše mali da bi predstavljao reprezentativni uzorak stvarne ciljne funkcije. U bilo kom od ova dva slučaja, jednostavni algoritam može stvoriti stablo koje ima osobinu **preprilagodjenosti (overfitting)** datom obučavajućem skupu. Kaže se da je neka hipoteza preprilagodjena obučavajućem skupu, ako postoji neka druga hipoteza, koja je manje tačne na obučavajućem skupu, ali daju veću tačnost na primerima van obučavajućeg skupa.

Slika 10.10 prikazuje uticaj prilagodjenosti u jednoj tipičnoj primeni učenja stabala odluke. Horizontalna osa prikazuje ukupan broj čvorova u stablu odluke tokom njegovog formiranja. Vertikalna osa predstavlja tačnost predikcije stabla. Punom linijom je predstavljena tačnost stabla odluke na obučavajućim primerima, a isprekidanom tačnost na nezavisnom test skupu primera. Kao što se može predvideti, tačnost stabla na obučavajućim primerima se monotono povećava kako stablo raste. Nasuprot tome, tačnost merena na test skupu se prvo povećava, a potom opada.



Sl.10.10 Efekat prilagodjenosti pri obučavanju stabala odluke. Apscisa označava veličinu stabla odluke izraženu u broju čvorova. Ordinata označava verovatnoću tačne klasifikacije.

Kako je moguće da je stablo  $h$  bolje na obučavajućem skupu nego  $h'$ , a da se lošije ponaša od  $h'$  na test primerima? Jedan razlog za to je što obučavajući primeri mogu imati slučajne greške ili šum. Na primer, razmotrimo efekat koji dodavanje sledećeg pozitivnog primera, koji je greškom obeležen kao negativan, ima na inače ispravne primere u tabeli 2.

{ *Vreme* = *Sunčano*, *Temperatura* = *Vruće*, *Vlažnost* = *Normalna*,  
*Vetar* = *Jak*, *IratiTenis* = *Ne* }

Ako koristi tačne podatke, ID3 pravi stablo dato na Sl.10.4. Međutim, dodavanje ovog netačnog primera će navesti ID3 da napravi složenije stablo odluke. Novi primer će biti svrstan u drugi list s leva naučenog stabla sa Sl.10.4 zajedno sa prethodnim pozitivnim primerima D9 i D11. Pošto je novi primer obeležen kao negativan, ID3 će nastaviti da traži dalje grananje stabla ispod ovog čvora. Sve dok se novi pogrešni primer razlikuje na neki procenjiv način od ostalih primera svrstanih u isti čvor, ID3 će nastavljati narastanje stabla odluke. Kao rezultat, ID3 će dati stablo odluke  $h$  koje je složenije od originalnog stabla  $h'$  sa Sl.10.4. Naravno,  $h$  će bez problema odgovarati

skupu obučavajućih primera, dok jednostavnije stablo  $h'$  neće. Međutim, pošto je novi čvor odluke posledica praćenja obučavajućeg primera koji je nastao zbog šuma, očekujemo da će se  $h$  bolje pokazati nego  $h'$  na novim nevidjenim primera.

Nesistemske greške ove vrste ili u vrednostima atributa ili u informacijama o klasi se obično nazivaju šumom. Gornji primer ilustruje kako slučajni šum u obučavajućim primerima može dovesti do preprilagodjavanja. U stvari, do preprilagodjavanja može doći i kada u obučavajućim podacima nema šuma, naročito ako je za listove stabla vezan mali broj primera. U takvim slučajevima su moguće slučajne regularnosti zbog kojih neki atributi mogu dobro da separišu primere iako nisu povezani sa ciljnom funkcijom. Kad postoje takve slučajne regularnosti, postoji šansa da se javi preprilagodjenost.

Preprilagodjenost je značajan praktičan problem za učenje stabala odluke, ali i gotovo sve algoritme mašinskog učenja. Postoji više pristupa rešavanju problema preprilagodjenosti kod učenja stabala odluke, a ovde ćemo navesti samo dva:

- ♦ pristupi **ranog zaustavljanja** rasta stabla, pre nego što dostigne tačku gde idealno klasifikuje obučavajuće podatke;
- ♦ pristupi koji dozvoljavaju da se desi preprilagodjenje, a onda se naknadno vrši tzv. **kresanje** preprilagodjenog stabla.

Iako prvi pristup može delovati direktniji, drugi pristup je uspešniji u praksi, usled nemogućnosti tačne procene konačne veličine stabla kod ranog zaustavljanja.

Nezavisno da li se adekvatna veličina stabla postiže na osnovu prvog ili drugog pristupa, ključno pitanje je koji kriterijum treba koristiti za njeno određivanje. Pristupi su sledeći:

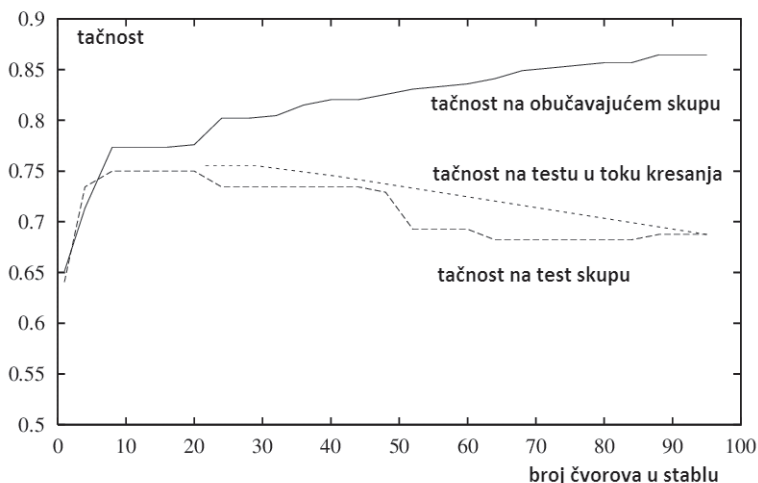
1. Koristiti odvojen skup primera, različit od obučavajućeg, za procenu koristi od naknadnog kresanja čvorova sa stabla.
2. Koristiti sve dostupne podatke za obuku, ali primeniti statistički test za procenu da li proširivanje ili kresanje određenog čvora dovodi do poboljšanja za celokupnu distribuciju primera ili samo za obučavajuće podatke.
3. Koristiti eksplicitne mere kompleksnosti za kodovanje obučavajućih primera i stabla odluke, zaustavljajući rast stabla kada je kompleksnost kodovanja minimizirana.

Prvi od gore navedenih pristupa se najčešće sreće i često se naziva pristup obučavajućeg i **validacionog skupa**. Kod njega se podaci na raspolaganju dele u dva skupa: obučavajući skup, koji se koristi da se određena hipoteza nauči, i validacioni skup, koji se koristi za procenu preciznosti ove hipoteze nad narednim podacima i efekata kresanja ove hipoteze. Opravdanje za ovaj pristup leži u činjenici da je ponavljanje slučajnih grešaka i regularnosti u obučavajućem skupu, malo verovatno i u validacionom skupu. Stoga se očekuje od validacionog skupa da izvrši proveru da li se preklapaju slučajne karakteristike obučavajućeg skupa. Stoga je važno da validacioni skup bude dovoljno veliki da može da obezbedi statistički značajan uzorak primera. Uobičajena heuristika je da se za validacioni skup zadrži jedna trećina primera koji su na raspolaganju, a da se preostale dve trećine iskoriste za obuku.

### 10.7.2 Kresanje (pruning) u cilju smanjivanje greške klasifikacije

Na koji način možemo da koristimo validacioni skup da sprečimo preprilagodjenost? Pristup koji se naziva kresanje sa umanjenom greškom razmatra svaki čvor odluke u stablu kao kandidata za odbacivanje. Kresanje čvora odluke se sastoji od uklanjanja podstabla čiji je koren taj čvor, pravljenja lista i dodeljivanja tom listu oznake većinske klasifikacije obučavajućih primera koji su bili povezani sa čvorom koji krešemo. Čvor se uklanja samo ako se rezultujuće okresano stablo ne pokaže lošije na validacionom skupu od polaznog stabla. Ovo je korisno, jer će bilo koji list koji je dodat zbog slučajnih regularnosti u obučavajućem skupu biti najverovatnije uklonjen zbog toga što je malo verovatno da će se ove koincidencije ponoviti u validacionom skupu. Čvorovi se krešu iterativno, uvek uzimajući onaj čvor čijim se uklanjanjem najviše povećava tačnost stabla odluke na validacionom skupu. Kresanje čvorova se nastavlja dok ne postane štetno, tj. dok ne počne da se smanjuje tačnost stabla na validacionom skupu.

Uticaj kresanja sa umanjenom greškom je dat na Sl.10.11. Kao i na Sl.10.10 prikazana je tačnost stabla na obučavajućim i test primerima. Dodatna linija na grafiku prikazuje tačnost na test primerima nakon kresanja. Kada kresanje počne, stablo ima svoju maksimalnu veličinu i najmanju tačnost na test skupu. Kako se kresanje nastavlja, broj čvorova se smanjuje i tačnost na test skupu raste. U ovom primeru su dostupni podaci podaljeni u tri podskupa: obučavajuće primere, validacione primere koji se koriste za kresanje stabla i skup test primera, koji treba da obezbedi što bolju procenu tačnosti za buduće nepoznate primere – **tzv. tačnost generalizacije**.



Sl.10.11 Efekat kresanja u toku obučavanja stabala odluke. Apscisa označava veličinu stabla odluke izraženu u broju čvorova. Ordinata označava verovatnoću tačne klasifikacije.

Korišćenje zasebnog skupa podataka za vođenje kresanja je efektan pristup, pod uslovom da je dostupna velika količina podataka. Glavna mana ovog pristupa je što,



kada su podaci ograničeni, zadržavanje jednog njihovog dela za validacioni skup još više smanjuje broj primera koji se koriste za obuku.

### 10.7.3 Uvođenje atributa sa kontinualnim vrednostima

Početna definicija ID3 algoritma je ograničena na diskretne attribute koji uzimaju vrednosti iz konačnih skupova. Prvo, ciljni atribut čija vrednost se predviđa stablom odluke mora biti diskretan. Drugo, atributi koji se testiraju u čvorovima stabla odluke moraju takođe biti diskretni. Ovo drugo ograničenje se lako može ukloniti, tako da stabla odluke podržavaju rad sa kontinualnim atributima. To se rešava pomoću dinamičke diskretizacije atributa, tj. podele opsega kontinualnog atributa u disjunktne intervale kojima se zatim dodeljuju diskretne vrednosti. Npr. za atribut  $A$  koji je kontinualan, algoritam može dinamički da formira novi Bulov atribut  $A_c$ , koji je tačan ako je  $A < c$  i netačan u ostalim slučajevima. Pitanje je kako odrediti vrednost praga  $c$ ?

Kao primer, pretpostavimo da želimo da uključimo novi kontinualni atribut *Temperatura* u vektore obeležja obučavajućeg skupa datog u Tabeli 10.2. Pretpostavimo da obučavajući primeri u vezi sa nekim određenim čvorom u stablu odluke imaju sledeće vrednosti za *Temperatura* i ciljni atribut *IgratiTenis*:

|                    |           |           |           |           |           |           |
|--------------------|-----------|-----------|-----------|-----------|-----------|-----------|
| <i>Temperatura</i> | 4.5       | 9         | 15.5      | 22        | 26.5      | 32        |
| <i>IgratiTenis</i> | <i>Ne</i> | <i>Ne</i> | <i>Da</i> | <i>Da</i> | <i>Da</i> | <i>Ne</i> |

Tabela 10.3 Primer kontinualnog atributa *Temperatura* i korespondentnih vrednosti ciljne klasifikacije. Postupak optimalne dinamičke diskretizacije započinje sortiranjem vrednosti kontinualnog atributa i detektovanjem mesta promene ciljne klasifikacije, što je označeno uspravnim isprekidanim linijama.

Poželjno bi bilo da odaberemo prag  $c$  takav da se dobije najveći informacioni dobitak. Sortiranjem vrednosti kontinualnog atributa  $A$ , a potom identifikovanjem mesta promene ciljne klasifikacije (isprekidane uspravne linije u Tabeli 10.3), možemo generisati skup vrednosti koje su kandidati za prag. Pragove računamo kao sredinu intervala između odgovarajućih vrednosti atributa  $A$  na mestima promene ciljne klasifikacije. U gornjem primeru su to kandidati:

$$\frac{9 + 15,5}{2} = 12,5 \text{ i } \frac{26,5 + 32}{2} = 29,25.$$

Potom računamo informacioni dobitak za svaki od atributa kandidata: *Temperatura* > 12.25 i *Temperatura* > 29.25, a zatim biramo veći, koji je u ovom primeru *Temperatura* > 12.25.

Ovakav dinamički kreiran Bulov atribut se priključuje drugim diskretnim atributima u procesu formiranja stabla odluke. Izloženi pristup se može generalizovati sa binarne na opštu,  $n$ -arnu diskretizaciju.



#### 10.7.4 Alternativne mere za selektovanje atributa

Jedan od nedostataka mere informacionog dobitka je favorizovanje atributa sa više vrednosti nad onima sa manjim brojem vrednosti. Kao ekstreman primer, razmotrimo atribut *Datum*, koji ima veliki broj mogućih vrednosti. Kada bismo ovaj atribut dodali u tabelu 2. on bi imao najveći informacioni dobitak među svim ostalim atributima. To sledi iz činjenice da *Datum* savršeno predviđa ciljni atribut za obučavajuće podatke. Stoga bi *Datum* bio odabran za atribut odluke za koreni čvor stabla i na taj način bismo dobili veoma široko stablo dubine 1, koje idealno klasifikuje obučavajuće primere. Ovakvo stablo odluke bi se loše pokazalo na sledećim primerima, jer nije koristan prediktor uprkos tome što odlično razdvaja obučavajuće podatke.

Jedan način da se ova prepreka prebrodi je da se selektuju atributi bazirani na meri koja nije informacioni dobitak. Alternativna mera koja se uspešno koristi u oblasti mašinskog učenja je **stepen dobitka (*gain ratio*)**. Mera stepen dobitka kažnjava attribute kao što je *Datum* tako što uključuje pojam podeljene informacije (*split information*), koji je osetljiv na to koliko uniformno i široko atribut deli podatke:

$$SplitInformation(S, A) = - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|},$$

gde su  $S_1$  do  $S_c$  podskupovi primera koji nastaju particijom skupa  $S$  atributom  $A$ , čiji je broj vrednosti jednak  $c$ . Primetimo da *SplitInformation* ustvari predstavlja entropiju od  $S$ , u odnosu na vrednosti atributa  $A$ . Ovo je potpuno suprotno našim prethodnim upotrebama entropije, kada smo razmatrali samo entropiju od  $S$  u odnosu na ciljni atribut čiju vrednost treba predvideti naučenim stablom.

Mera *GainRatio* se može izraziti preko ranije definisanih mera *Gain* i *SplitInformation*:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}.$$

Primetimo da *SplitInformation* zabranjuje selekciju atributa sa verlikim brojem uniformno raspoređenih vrednosti. Pogledajmo, na primer, kolekciju od  $n$  primera koji su potpuno razdvojeni atributom  $A$  (npr. *Datum*). U ovom slučaju, vrednost *SplitInformation* bi bila  $\log_2 n$ . Nasuprot tome, Bulov atribut  $B$  koji razdvaja istih  $n$  primera tačno na pola bi imao *SplitInformation*=1. Ako atributi  $A$  i  $B$  daju isti informacioni dobitak, onda je očigledno da će  $B$  imati veći stepen dobitka.

Jedan praktičan problem koji dovodi u pitanje korišćenje mere stepen dobitka je to što imenilac u izrazu može biti za neko  $S_i$  jednak nuli ili veoma mali kada je  $|S_i| \approx |S|$ . Zbog toga vrednost *GainRatio* može biti ili nedefinisana ili veoma velika kod atributa za koje se dogodi da imaju skoro iste vrednosti za sve članove skupa  $S$ . Da bismo izbegli biranje atributa samo po ovoj osnovi, možemo usvojiti neke heuristike, kao što je prvo računati *Gain* svakog atributa ponaosob, a onda primeniti *GainRatio* test, razmatrajući samo one attribute sa vrednošću *Gain* iznad prosečne vrednosti.

### 10.7.5 Obučavajući primeri kod kojih nedostaju vrednosti atributa

U nekim slučajevima kod obučavajućih podataka mogu da nedostaju vrednosti nekih atributa. Na primer, u domenu medicine kada želimo da predvidimo ishod za nekog pacijenta na bazi raznih iscrpnih testova, može se desiti da je test *RezultatAnalizeKrv*i dostupan samo za podskup pacijenata. U takvim slučajevima, uobičajeno je da se atributi koji nedostaju estimiraju na osnovu ostalih primera čiji atributi imaju poznate vrednosti.

Posmatrajmo situaciju u kojoj treba izračunati  $Gain(S,A)$  u  $n$ -tom čvoru stabla odluke, da bi se procenilo da li je atribut  $A$  najbolji atribut za testiranje u ovom čvoru. Pretpostavimo da je  $\langle x, c(x) \rangle$  jedan od obučavajućih primera u skupu  $S$  i da vrednost  $A(x)$  nije poznata.

Jedna strategija rešavanja problema je da mu se dodeli vrednost koja se najčešće javlja među obučavajućim primerima u  $n$ -tom čvoru. Alternativno, možemo mu dati vrednost koja se najčešće pojavljuje među primerima u  $n$ -tom čvoru i koja istovremeno ima klasifikaciju  $c(x)$ . Potom, algoritam učenja stabla odluke direktno može da koristi ovakav obučavajući primer koji je dopunjen procenjenom vrednošću  $A(x)$ .

Druga, kompleksnija procedura, koju koristi C4.5, je da se dodeli verovatnoća svakoj od mogućih vrednosti atributa  $A$ , a ne jednostavno biranje najčešće vrednosti  $A(x)$ . Ove verovatnoće se procenjuju na osnovu učestanosti pojavljivanja svake vrednosti od  $A$  za  $n$ -ti čvor. Na primer, za Bulov atribut  $A$ , ako  $n$ -ti čvor ima šest poznatih vrednosti za  $A=1$  i četiri za  $A=0$ , rekli bismo da su verovatnoće  $P(A(x)=1)=0.6$  i  $P(A(x)=0)=0.4$ . Deo od 0.6 primera  $x$  se raspoređuje duž grane za  $A=1$ , a preostalih 0.4 niz drugu granu. Ovi frakcioni primeri se koriste radi računanja informacionog dobitka i mogu se dalje deliti na nove grane ako dođe do testiranja još nekog atributa čije vrednosti nedostaju. Isto frakcionisanje primera može biti primenjeno i nakon učenja, da bi se klasifikovali novi primeri nepoznatih vrednosti atributa. U ovom slučaju, klasifikacija novih primera je najverovatnija klasifikacija, a računa se sumiranjem težina fragmenata klasifikovanih na različite načine u listovima stabla.

### 10.7.6 Atributi sa promenljivom cenom

U nekim zadacima učenja primeri se mogu vezivati i za cenu (*cost*). Prilikom učenje automatske dijagnostike u medicinskom domenu, pacijente možemo opisati atributima kao što su *Temperatura*, *RezultatiBiopsije*, *Puls*, *RezultatAnalizeKrv*i i slično. Cene ovih atributa značajno variraju, kako u pogledu novčanih troškova takao i uticaja na zdravlje. Kod takvih zadataka, više odgovaraju stabla odluke koja koriste attribute sa niskom cenom gde god je moguće, a da se oslanjaju na attribute sa visokom cenom samo gde je to neophodno da bi se došlo do pouzdane klasifikacije.

ID3 se može modifikovati da uzima u obzir cene atributa tako što se uvodi cena u meru selekcije atributa. Na primer, možemo podeliti informacioni dobitak cenom atributa, tako da se više koriste atributi sa nižom cenom.

## Rezime 10. poglavlja

- ♦ Za jedan sistem VI (računarski program) kažemo da uči zadatu klasu zadataka T na osnovu iskustva E i zadate mere performansi P, ako se njegove performanse za rešavanje zadataka iz klase T, poboljšavaju sa iskustvom E.
- ♦ Od mnogih oblika učenja, od najvećeg značaja za tekuću praksu u domenu računarstva i veštačke inteligencije predstavlja induktivno učenje, koje se često proširuje u nazivu na induktivno empirijsko učenje.
- ♦ Induktivno empirijsko učenje funkcionalnih preslikavanja se odnosi na najtradicionalniju klasu mašinskog učenja. Obučavanje se vrši na osnovu obučavajućeg skupa D, koji se sastoji od N ulazno-izlaznih parova.
- ♦ Celokupan proces obučavanja možemo posmatrati kao pretragu u prostoru hipoteza H, u cilju izbora najpogodnije hipoteze g, koja je u saglasnosti sa obučavajućim skupom.
- ♦ Opšteprihvaćena mera kvaliteta jednog sistema mašinskog učenja je njegova sposobnost generalizacije, odnosno ponašanje sistema na instancama koje nisu vidjene u fazi obučavanja.
- ♦ Tipično sintezu prate dva skupa, obučavajući (trening) i test skup. Test skup se koristi samo za procenu generalizacionih sposobnosti sintetisanog sistema i ne sme se koristiti u fazi obučavanja. Uobičajene numeričke mere performansi su ili srednjekvadratna greška između izlaza sistema i ciljnih vrednosti ili ukupan broj grešaka (ili prosečna verovatnoća greške)
- ♦ Induktivni pomeraj (bias) je skup pretpostavki koje zajedno sa obučavajućim primerima opravdavaju klasifikacije budućih primera, koji nisu učestvovali u fazi obučavanja. Induktivni pomeraj su restrikcije koje namećemo prostoru hipoteza, smatrajući zadati algoritam obučavanja uspešnim ukoliko je pronašao najbolju hipotezu u prostoru hipoteza uz istovremeno zadovoljavanje nametnutih restrikcija.
- ♦ Bez induktivnog pomeraja obučavanje u pravom smislu nije moguće.
- ♦ Kaže se da je neka hipoteza prilagodjena obučavajućem skupu, ako postoji neka druga hipoteza, koja je manje tačne na obučavajućem skupu, ali daju veću tačnost na primerima van obučavajućeg skupa.
- ♦ Ukoliko pretpostavimo da se prostor hipoteza sastoji od objekata strukture stabla dolazimo do koncepta induktivnog empirijskog učenja stabala odlučivanja, kao jedne od veoma važnih oblasti mašinskog učenja.
- ♦ Svaki put kroz stablo predstavlja jedno klasifikaciono pravilo, tj. konjunkciju testova atributa, a samo stablo je disjunkcija ovih konjunkcija.
- ♦ Većina algoritama koji su razvijeni za učenje stabala odluke su varijacije bazičnog algoritma koji koristi od-vrha-naniže (top-down), pohlepnu (greedy) pretragu kroz prostor mogućih stabala odluke. Ovaj pristup je korišćen kod ID3 algoritma i njegovog naslednika C4.5.

- ♦ Grananja u unutrašnjim čvorovima stabla odluke se vrše na osnovu testova čistoće paricije pripadajućih primera na što čistije podskupove u pogledu vrednosti ciljne funkcije. Najpozantiji testovi su informacioni dobitak i stepen dobitka.
- ♦ Generalizaciona svojstva indukovanih stabala odlučivanja se mogu poboljšati primenom metoda kresanja i metodom ranog zaustavljanja.

## Pitanja i zadaci

1. Da li bi sistem mašinskog učenja mogao da nauči neki koncept bez uvođenja induktivnog pomeraja?
2. Zašto je induktivni transfer značajan u savremenim sistemima veštačke inteligencije?
3. Da li je učenje interne reprezentacije jedini način ostvarivanja induktivnog transfera?
4. Objasnite svojim rečima zašto manje složene hipoteze imaju veću šansu da dobro generališu?
5. Objasnite razliku između regresije i klasifikacije.
6. Objasnite svojim rečima kako je uopšte moguće učenje bez učitelja (samoo-učavanje)?
7. Zašto su kratki obučavajući skupovi prepreka sintezi uspešnih sistema mašinskog učenja?
8. Objasnite svojim rečima zašto transdukcija može dati bolje rezultate od induktivnog učenja za isti broj obeleženih primera?
9. Zašto aktivno obučavanje može dati bolje rezultate od pasivnog?
10. Opišite svojim rečima induktivni pomeraj ID3 algoritma.
11. Zašto se prilagodjenost može ublažiti metodama ranog zaustavljanja obučavanja?
12. Zašto se prilagodjenost može ublažiti procesom kresanja stabala odlučivanja nakon faze obučavanja?
13. Kakva je veza između tehnike stabala odlučivanja i sistema zasnovanih na bazama znanja?
14. Dajte primere aktivnog i pasivnog učenja u domenu unapredjenja korišćenja mobilnih telefona.
15. Dajte primer induktivnog i transduktivnog obučavanja u domenu poboljšanja web servisa.
16. Zašto pojava velike količine podataka na Internetu pogoduje tehnologiji mašinskog učenja?



# 11. Neuronske mreže

U ovom poglavlju biće izložena osnovna svojstva, arhitekture i algoritmi obučavanja veštačkih neuronskih mreža. Kao što je već napomenuto u poglavljima 2 i 3, neuronske mreže predstavljaju osnovu tzv. konekcionistačkog pristupa VI. Ovaj pristup, za razliku od komplementarnog simboličkog pristupa, podržava princip od dole ka gore (botom-up), po kome se, polazeći od senzorskih signala ili sirovih podataka, induktivno kroz proces obučavanja, formiraju koncepti, pojmovi, kategorije i sl. Ove apstrakcije formirane na izlaznim slojevima neuronskih sistema, mogu predstavljati osnovu za prelazak na procesiranje simboličkih struktura i povezivanje sa sistemima dizajniranim na osnovu simboličkog pristupa. Tipičan primer ovakve sinergije dva komplementarna pristupa je najnoviji Google-ov sistem, koji automatski tekstualno opisuje šta se vidi na jednoj slici.

Neuronske mreže pobudjuju velika očekivanja u oblasti VI, ne samo kao samostalni blokovi modelovanja pojedinih preslikavanja, neophodnih u sintezi sistema VI, već i kao alat pomoću koga se mogu modelovati biološki neuronski sistemi, uključujući i čovekov centralni nervni sistem i mozak. Ukoliko bi se u nekom trenutku tehnološkog razvoja mogao simulirati sistem koji ima kapacite ljudskog mozga, očekuje se da bi se time ostvarila najbolja moguća računarska podloga za izučavanje svih kognitivnih procesa prisutnih u našem mozgu, što bi u svakom slučaju dalo snažan podsticaj ostvarivanju većine ciljeva VI. Današnji najsloženiji simulatori pokrivaju samo procenite ili delove procenata ukupnog broja neurona i sinaptičkih veza ljudskog mozga. Stoga su tzv. emergentna svojstva neuronskih mreža (osobine nastale kao posledica interakcije ogromnog broja jednostavnih elemenata) još uvek van domašaja istraživača.

U velikoj meri se oblast veštačkih neuronskih mreža danas razvija kao saostalna oblast, u kojoj je VI samo jedna od oblasti primene. Neuronskih mreža su u okviru mašinskog učenja već stekle status dobro teorijski obrađene oblasti, koja je stavlja u istu ravan sa ostalim top tehnikama, kao što su SVM (Support Vector Machine) metode ili grafički verovatnosni modeli. Poseban vetar u jedra je oblast dobila aktuelizovanjem tzv. dubokih arhitektura za obučavanje, čime nastaje čitava nova oblast DNN (Deep Neural Networks), za koju se ispostavlja da je danas najefikasnije sredstvo za obradu senzorskih signala ljudskog perceptivnog sistema, kao što su vizuelni i audio signali.

Ako neuronske mreže posmatramo kao novu računarsku paradigmu, možemo napraviti interesantna poređenja sa klasičnom računarskom tehnikom zasnovanoj na Fon Nojmanovoj arhitekturi, Tabela 11.1. i 11.2. Kao što se može zapaziti, distinktivna razlika neuronskih mreža u odnosu na klasično računarstvo potiče od masovnog paralelizma i brze adaptacije ogromnog broja konekcija – sinapsi. U pogledu ekvivalentnog programiranja, dok klasično računarstvo zahteva eksplicitno pisanje programa u nekom izabranom programskom jeziku, neuronske mreže obavljaju namenjeni zadatak implicitno, preko faze obučavanja podržane primerima struktuiranim u tzv. obučavajuće skupove, što u izvesnom smislu odgovara konceptu automatskog adaptivnog programiranja, potpuno stranom za Fon Nojmanove arhitekture.

|   | Neuronske mreže                                                                                             | Fon Nojmanov digitalni računar                                          |
|---|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| 1 | Obučavanje (učenje na osnovu primera) zasnovano na podešavanju jačine konekcionih veza, pragova i strukture | Programiranje kroz instrukcije (ako-onda analiza zasnovana na logici)   |
| 2 | Memorijski i procesni elementi su kolocirani                                                                | Memorija i procesiranje su separisani                                   |
| 3 | Paralelni i asinhroni rad (kontinualni ili diskretni)                                                       | Sekvencijalni ili serijski rad, sinhronisan zajedničkim taktom          |
| 4 | Otporan na greške usled distribuirane reprezentacije i velike redundanse                                    | Nije otporan na greške                                                  |
| 5 | Smooorganizovanje u toku obučavanja                                                                         | Zavisan od programa                                                     |
| 6 | Kodovano znanje je adaptibilno. Prezentovano je interkonekcijama između neurona                             | Znanje je memorisano u adresibilnoj memoriji i striktno je replikabilno |
| 7 | Procesiranje je anarhično                                                                                   | Procesiranje je autokratično                                            |
| 8 | Osnovni vremenski ciklus obrade je reda milisekundi                                                         | Osnovni vremenski ciklus obrade je reda nanosekundi                     |

Tabela 11.1 Sličnosti i razlike između neuronskih mreža i Fon Nojmanovog računara.

|   |                         | Neuronske mreže                            | Fon Nojmanov digitalni računar         |
|---|-------------------------|--------------------------------------------|----------------------------------------|
| 1 | Broj procesnih jedinica | 1CPU, $10^8$ gejtova                       | $10^{11}$ neurona                      |
| 2 | Memorijske jedinice     | $10^{10}$ bita RAM,<br>$10^{11}$ bita disk | $10^{11}$ neurona<br>$10^{14}$ sinapsi |
| 3 | Vreme jednog ciklusa    | $10^{-9}$ sec                              | $10^{-3}$ sec                          |
| 4 | Propusni opseg          | $10^{10}$ b/sec                            | $10^{14}$ b/sec                        |
| 5 | Promene u vremenu       | $10^{10}$                                  | $10^{14}$                              |

Tabela 11. 2. Sličnosti i razlike između neuronskih mreža i Fon Nojmanovog računara, kvantitativni pokazatelji

## 11.1 DEFINICIJA NEURONSKIH MREŽA

Neuronske mreže pretenduju da simuliraju način rada ljudskog mozga pri obavljanju datog zadatka ili neke funkcije. Mogu se posmatrati kao masovno paralelizovani distribuirani procesori sa prirodnom sposobnošću memorisanja iskustvenog znanja i obezbeđivanja njegovog korišćenja. Veštačke neuronske mreže podsećaju na ljudski mozak u dva pogleda:

1. neuronska mreža zahvata znanje kroz proces obučavanja,
2. težine medjuneuronskih veza (jačina sinaptičkih veza) služe za memorisanje znanja.

Procedura kojom se obavlja obučavanje naziva se algoritam obučavanja. Kroz ovu proceduru se na algoritamski (sistematičan) način menjaju sinaptičke težine u cilju dostizanja željenih performansi mreže. Osnovnu računarsku snagu neuronskih mreža čini masovni paralelizam, sposobnost obučavanja i generalizacija. Generalizacija predstavlja sposobnost proizvodjenja zadovoljavajućeg izlaza neuronske mreže i za ulaze koji nisu bili prisutni u toku obučavanja.

## 11.2 SVOJSTVA NEURONSKIH MREŽA

Nabrajaćemo neka distinktna svojstva neuronskih mreža.

1. Nelinearnost, koja je u osnovi distribuirana.
2. Ulazno-izlazno preslikavanje, koje se restauriše kroz proces obučavanja.
3. Adaptivnost-sposobnost menjanja jačine sinaptičkih veza.
4. Evidencionalni odziv. Neuronska mreža kao izlaz može da produkuje i stepen uverenja o datoj odluci.
5. Kontekstualna informacija. Svaki neuron u neuronskoj mreži je pod uticajem globalne aktivnosti ostalih neurona. Stoga je kontekstualna informacija prirodno imanentna ovim strukturama



6. Otpornost na otkaz.
7. Mogućnost realizacije u VLSI (Very Large Scale Integration) tehnologiji.
8. Uniformnost analize i sinteze. Neuron je zajednički element za sve tipove neuronskih mreže. Modularne neuronske mreže se mogu formirati integracijom pojedinih celina-modula. Za rešavanje različitih praktičnih problema koriste se iste teorijske postavke i algoritmi obučavanja.
9. Neurobiološke analogije. Neurobiolozi gledaju na neuronske mreže kao istraživački alat za interpretaciju neurobioloških fenomena, i obrnuto, inženjeri gledaju na neurobiologiju kao oblast iz koje mogu da izvlače nove ideje za rešavanje kompleksnijih problema od onih koji se mogu rešiti klasičnim hardversko-sofverskim tehnikama.

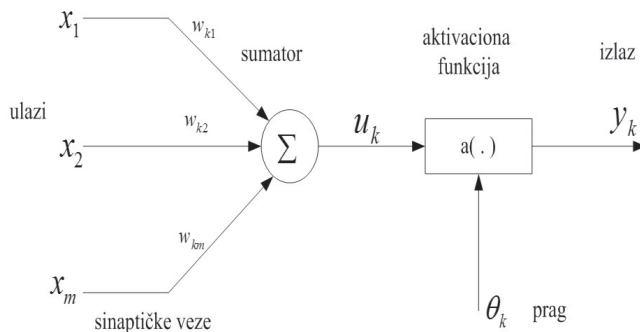
### 11.3 MODELI NEURONA

Model neurona čine tri bazična elementa:

- ♦ Skup sinaptičkih težina  $\{w_{ij}\}$ . Pozitivne težine odgovaraju ekscitirajućim sinaptičkim vezama, a negativne inhibitornim.
- ♦ Sumator (linearni kombajner) – formira težinsku sumu ulaza.
- ♦ Aktivaciona funkcija – limitira amplitudu izlaznog signala neurona. Tipično se uzima normalizacija izlaza na interval  $[0,1]$  ili  $[-1,1]$ .

Jednačine modela sa Slike 11.1

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (11.1)$$

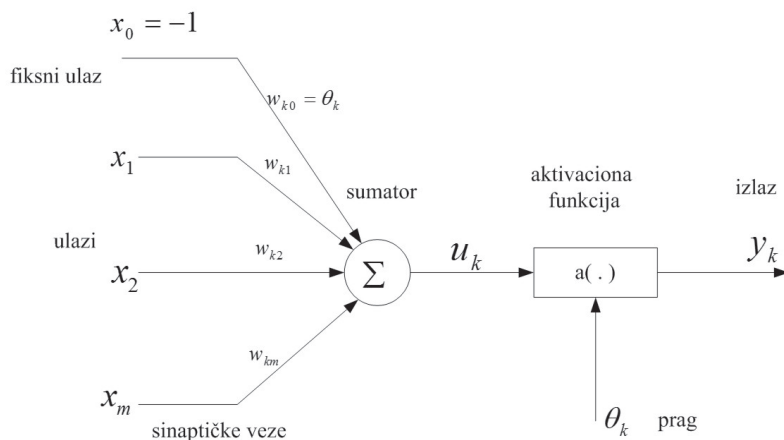


Sl.11.1. Nelinearni model neurona

$$y_k = a(u_k - \theta_k) \quad (11.2)$$

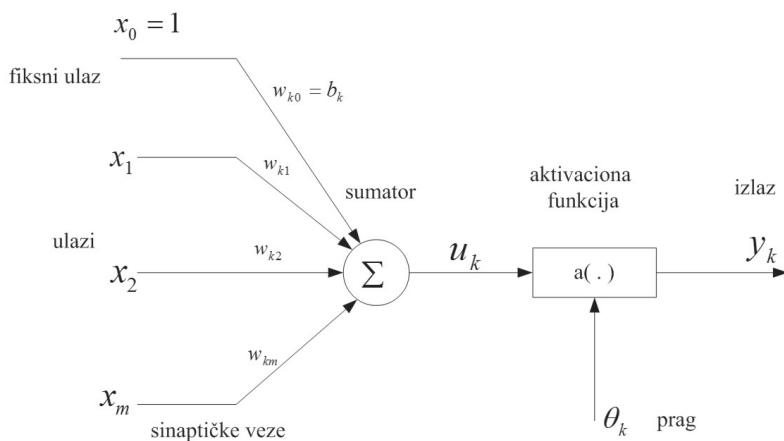
Alternativni zapis

$$v_k = \sum w_{kj} x_j, \quad y_k = a(v_k), \quad x_0 = -1, \quad w_{k0} = \theta_k. \quad (11.3)$$



Sl.11.2. Nelinearni model neurona sa proširenim ulazom i prenosom praga u sinaptičku težinu.

Ako stavimo da je  $x_0 = 1$ , a  $w_{k0} = b_k$ , tada se  $b_k$  naziva bajas, videti sl.11.3.



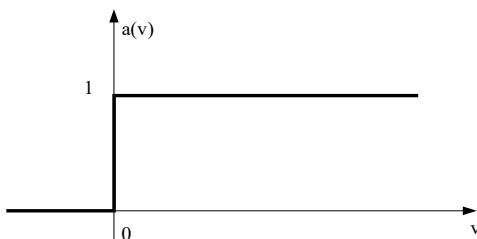
Sl.11.3. Nelinearni model neurona sa proširenim ulazom i bajasom u obliku sinaptičke težine.

## 11.4 TIPOVI AKTIVACIONIH FUNKCIJA

Razlikujemo sledeće najčešće tipove aktivacionih funkcija.

### Funkcija praga

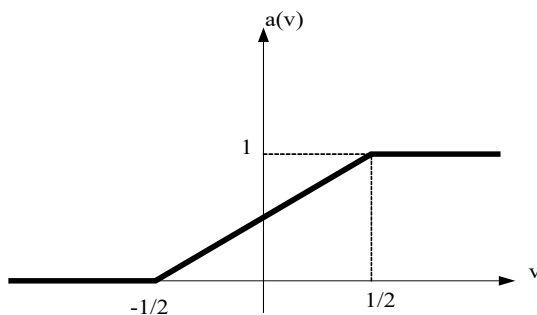
$$a(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$



Sl.11.4. Aktivaciona funkcija tipa praga. Neuron sa ovom aktivacionom funkcijom je poznat kao McCulloch – Pitts-ov model neurona (1943)

### U delovima linearna

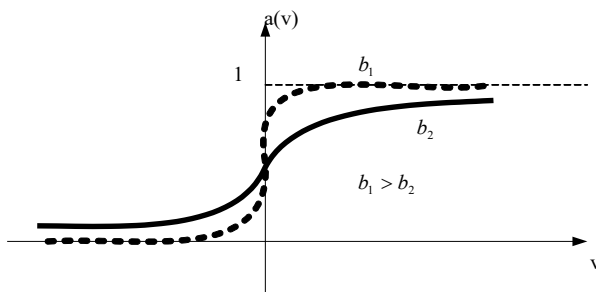
$$a(v) = \begin{cases} 1, & v \geq 1/2 \\ 1/2 + v, & -1/2 < v < 1/2 \\ 0, & v \leq -1/2 \end{cases}$$



Sl.11.5. U delovima linearna aktivaciona funkcija

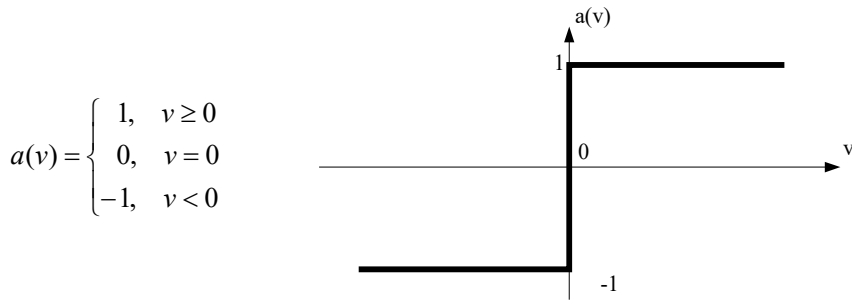
### Sigmoidalna (logistička)

$$a(v) = \frac{1}{1 + \exp(-bv)}$$

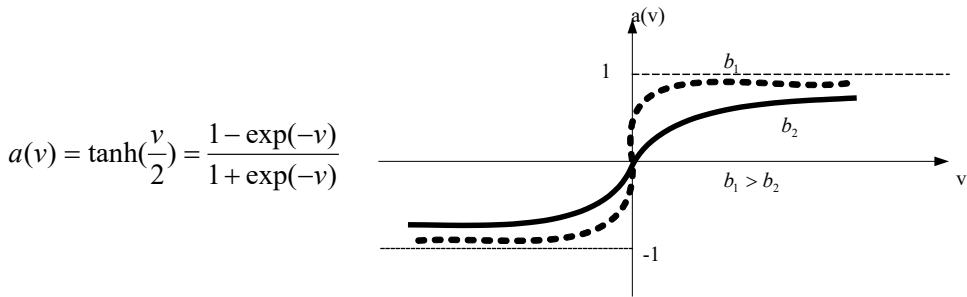


Sl.11.6. Sigmoidalna (logistička) aktivaciona funkcija. Parametar b je parametar nagiba.

Ukoliko se izlaz neurona normira na interval  $[-1,1]$ , tada dobijamo



Sl.11.7. Bipolarna aktivaciona funkcija tipa znaka ( $\text{sgn}(v)$ )

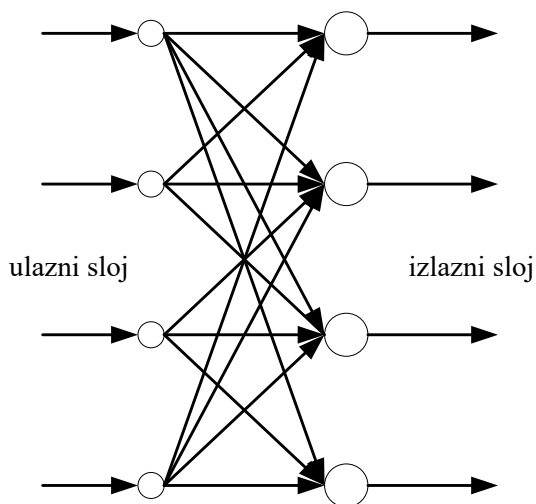


Sl.11.8. Bipolarna aktivaciona funkcija tipa tangensa hiperboličnog  
(sigmoidalna aktivaciona funkcija)

## 11.5 ARHITEKTURE NEURONSKIH MREŽA

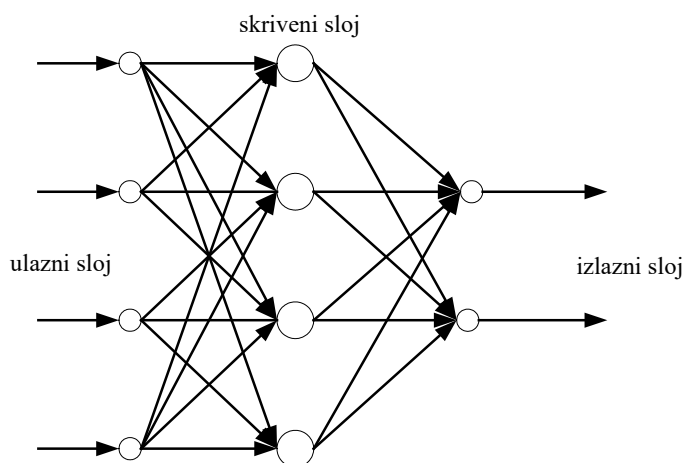
Različite arhitekture neuronskih mreža dobijamo variranjem rasporeda neurona i sinapsi, kao i smera prostiranja signala kroz ove strukture, od ulaza do izlaza. Po analogiji sa ljudskim mozgom, neuroni se po pravilu slažu u slojeve (layers), koji se zatim povezuju na specifičan način. Navešćemo neke nejprimjenjivnije arhitekture.

**Jednoslojne neuronske mreže sa prostiranjem signala unapred (feed forward single layer neural network)**



Sl.11.9. Jednoslojni perceptron sa prostiranjem unapred

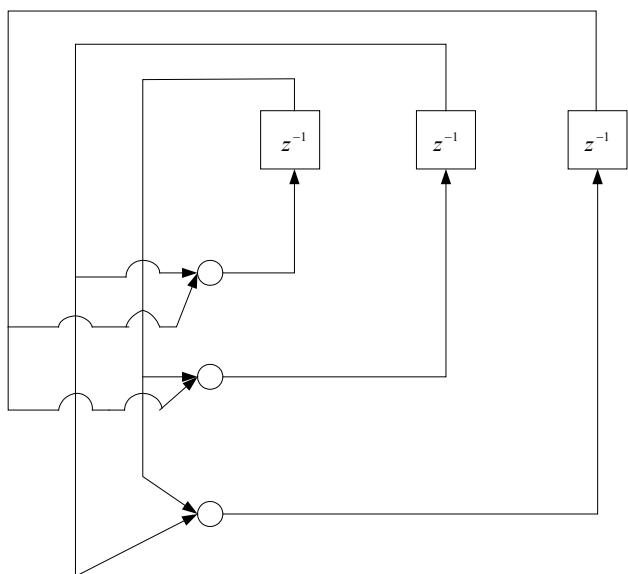
**Višeslojne neuronske mreže sa prostiranjem signala unapred (feedforward multilayer neural network)**



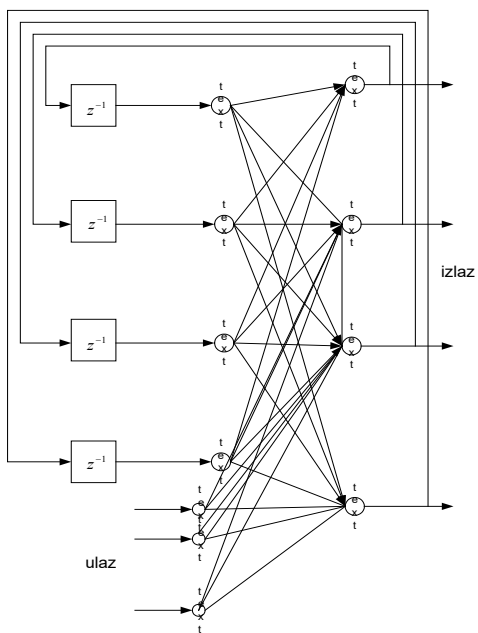
Sl.11.10. Višeslojna neuronska mreža sa prostiranjem unapred

**Rekurentske neuronske mreže**

Za razliku od višeslojnih neuronskih mreža, rekurentne neuronske mreže poseduju zatvorene petlje povratnih sprega.



Sl.11.11. Rekurentna neuronska mreža bez sopstvenih povratnih sprega i skrivenih slojeva.  
Operator  $z^{-1}$  ima značenje jediničnog vremenskog kašnjenja.



Sl.11.12. Rekurentna neuronska mreže sa skrivenim slojem

## 11.6 PREZENTACIJA ZNANJA U NEURONSKIM MREŽAMA

Znanje o okruženju je generalno dvojako.

1. Poznata znanja o okruženju, izražena kroz činjenice o tome šta je poznato – apriorno znanje.
2. Observacije (merenja) – dobijena od različitih senzora kao odraz stanja okruženja. Na osnovu ovih observacija se kreiraju obučavajući skupovi za obučavanje neuronskih mreža. Svaki primer u njemu se sastoji od parova (ulaz, izlaz).

Obučavajući skupovi predstavljaju znanje o okruženju od interesa. U klasičnom procesiranju, prirodno je prvo kreirati matematički model observacija, a zatim izvršiti validaciju ovog modela na realnim podacima. Neuronske mreže su direktno bazirane na podacima i daju implicitni model okruženja uz istovremeno obavljanje željenog procesiranja. Znanje o okruženju u neuronskim mrežama je kodovano kroz konkretne vrednosti slobodnih parametara dobijenih kroz obučavanje. Teško je bilo šta konkretno reći o reprezentaciji samog znanja unutar neuronske mreže. Postoje četiri pravila o reprezentaciji znanja u neuronskim mrežama, koji su opšte prirode.

Pravilo 1. Slični ulazi sličnih klasa prouzrokuju sličnu unutrašnju reprezentaciju.

Pravilo 2. Primeri koji pripadaju različitim klasama treba da budu predstavljeni različitim unutrašnjim reprezentacijama.

Pravilo 3. Apriorne informacije se ugrađuju direktno u neuronsku mrežu bez procesa obučavanja (specijalizacija strukture). Ovo se postiže ili

- ♦ restrikcijom arhitekture (lokalne konekcije)
- ♦ restrikcijom izbora sinaptičkih težina (weight shearing – metoda zajedničkih sinaptičkih težina).

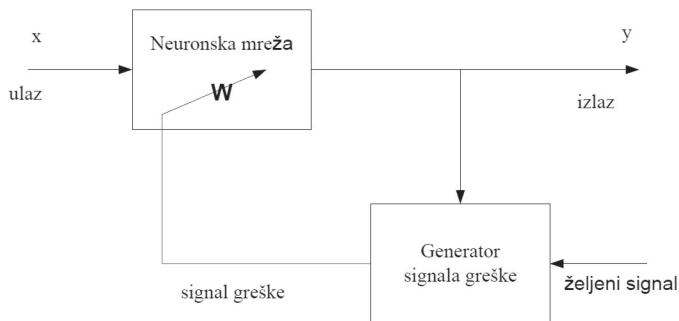
Specijalizacijom strukture se postiže:

- ♦ manji broj slobodnih parametara
- ♦ manji potrebni obučavajući skupovi
- ♦ brže obučavanje
- ♦ bolja generalizacija
- ♦ ubrzana je prenos signala kroz restriktovanu neuronsku mrežu
- ♦ cena realizacije je manja.

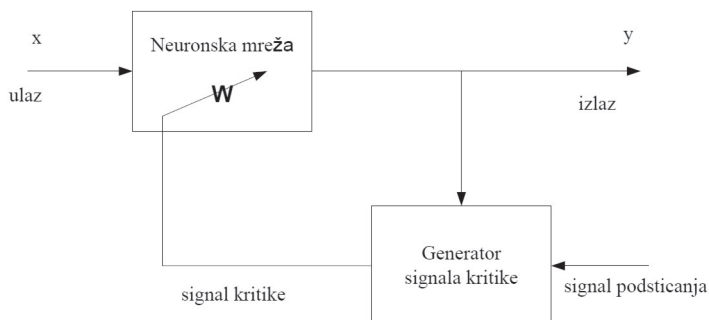
## 11.7 OBUČAVANJE NEURONSKIH MREŽA

Obučavanje je proces adaptiranja slobodnih parametara neuronske mreže, koji se obavlja kroz stimulaciju okruženja u kome se neuronska mreža nalazi. Proces obučavanja je klasifikovan u tri kategorije:

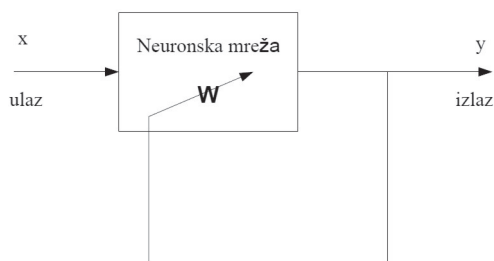
1. obučavanje sa učiteljem (nadzorom), (supervised learning)
2. obučavanje sa podsticanjem (reinforcement learning)
3. samoobučavanje (obučavanje bez učitelja), (unsupervised learning)



Sl.11.13 Obučavanje sa učiteljem



Sl.11.14 Obučavanje sa podsticanjem



Sl.11.15 Samoobučavanje

Kod obučavanja sa učiteljem prisutan je obučavajući skup u formi parova  $\{X^{(i)}, d^{(i)}\}$ ,  
gde je  $X^{(i)}$  ulaz, a  $d^{(i)}$  željeni izlaz.

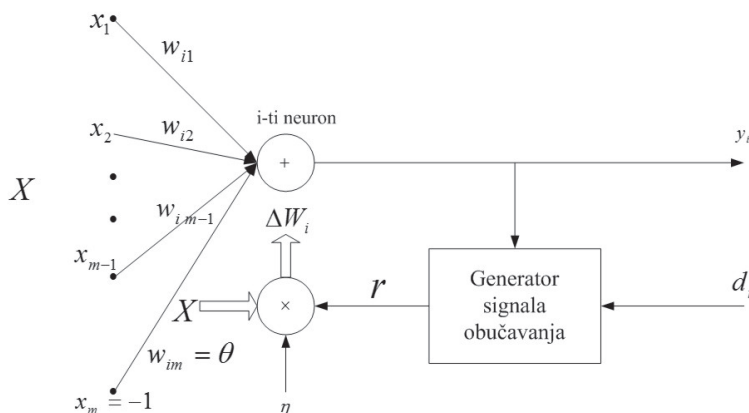
Kod obučavanja sa podsticanjem, neuronska mreža dobija rudimentirane informacije o tome kakav izlaz produkuje, najčešće samo u formi jednog bita informacije tipa {dobar, loš}. Analogno obučavanju sa učiteljem, ova forma obučavanja se može



treirati na isti način s tim što umesto učitelja, koji egzaktno ukazuje kakav odziv neuronske mreže treba da bude, u ovom slučaju imamo “kritičara” koji daje grublju ocenu odziva neuronske mreže.

Samooučavanje je karakterisano odsustvom bilo kakve povratne sprege od okruženja.

### 11.7.1. OPŠTA FORMA PRAVILA OBUČAVANJA



Sl.11.16 Opšta šema obučavanja i-tog neurona

$w_i = (w_{i1}, w_{i2}, \dots, w_{im})^T$ ,  $i = 1, 2, \dots, n$  - vektor sinaptičkih težina i-tog neurona

$$\Delta w_i(t) = \eta r x(t), \quad (11.4)$$

$\eta$  - koeficijent obučavanja – pozitivna konstanta.

$r$  – signal obučavanja, u opštem slučaju funkcija oblika

$$r = f_r(w_i, x, d_i), \quad (11.5)$$

$$w_i(t+1) = w_i(t) + \eta f_r(w_i(t), x(t), d_i(t)) x(t), \quad (11.6)$$

Na osnovu opšte jednačine (11.6), generisani su mnogi zakoni obučavanja, dominantno variranjem načina generisanja signala obučavanja  $r$ .

### 11.7.2 HEBOVO UČENJE

Hebov princip učenja je jedan od najstarijih i najpoznatijih. Zasniva se na Hebovom postulat:

*Kada je akson neurona A dovoljno blizu neurona B, tako da ga može eksitovati, i ako se to ponavlja dovoljno često, dešavaju se takve promene i metabolički procesi u obe ćelije da je efikasnost uticaja neurona A na neuron B povećana.*

$$r = y_i, \Rightarrow \Delta w_i = \eta y_i x. \quad (11.7)$$

Hebovo učenje je u osnovi samoobučavajuće, budući da nije prisutan signal željenog izlaza. U skalarnoj formi (11.7), ima formu

$$\Delta w_{ij} = \eta y_i x_j, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, m \quad (11.8)$$

Ako je ulazno-izlazni korelacioni član  $y_i x_i$  pozitivan,  $w_{ij}$  se povećava (u suprotnom se smanjuje), usled čega se povećava izlaz. Stoga će ulaz koji se najčešće pojavljuje, imati najveći uticaj na promenu težina, i na kraju će produkovati najveći izlaz, što i jeste ideja Hebovog postulata.

## 11.8 ADALINA (Adaptive Linear Element)

Neuron sa linearnom aktivacionom funkcijom se naziva linearni neuron. Neka je na raspolaganju obučavajući skup

$$\{(x^{(1)}, d^{(1)}), \dots, (x^{(p)}, d^{(p)})\}. \quad (11.9)$$

Cilj obučavanja je izračunavanje težina  $w_p$  koje zadovoljavaju relaciju

$$\sum_{j=1}^m w_j x_j^{(k)} = d^{(k)}, \quad k = 1, 2, \dots, p \quad (11.10)$$

i pri tome se minimizira kriterijum performansi

$$E(w) = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - y^{(k)})^2 = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - W^T x^{(k)})^2 = \frac{1}{2} \sum_{k=1}^p (d^{(k)} - \sum_{j=1}^m w_j x_j^{(k)})^2. \quad (11.11)$$

Ekstremizaciju kriterijuma (7.3) možemo obaviti gradijentnom metodom. datom sa

$$\Delta w = \eta \nabla_w E(w), \quad (11.12)$$

odnosno

$$\Delta w_j = \eta \frac{\partial E}{\partial w_j} = \eta \sum_{k=1}^p (d^{(k)} - W^T x^{(k)}) x_j^{(k)}, \quad j = 1, 2, \dots, m \quad (11.13)$$

Ukoliko se ove promene obavljaju individualno za svaki ulazni signal  $x^{(k)}$ , nalazimo da je

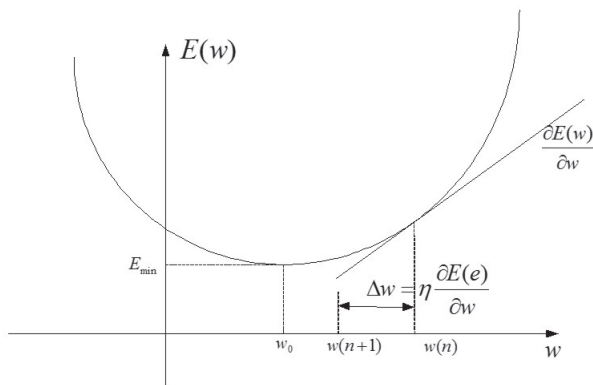
$$\Delta w_j = \eta (d^{(k)} - W^T x^{(k)}) x_j^{(k)}, \quad (11.14)$$

što je poznato Vidrov-Hofovo pravilo obučavanja. Ono se susreće i pod nazivom LMS pravilo (pravilo najmanjih kvadrata, Least Mean Square).

Ako želimo da Vidrov-Hofovo pravilo obučavanja izvedemo iz opšte jednačine obučavanja, neophodno je staviti za signal učenja

$$r = d - y = d - W^T x. \quad (11.15)$$

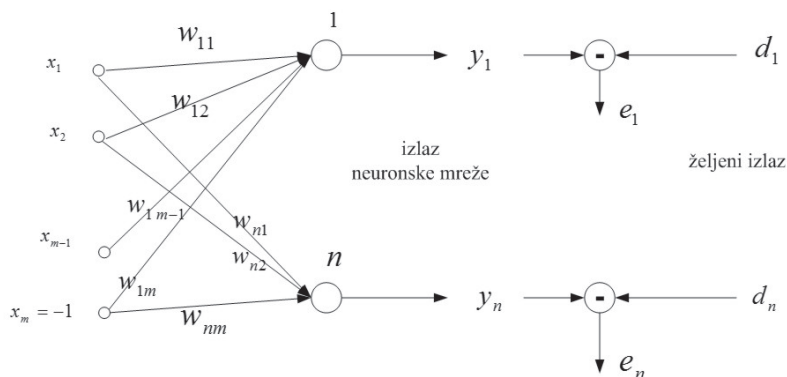
Budući da je  $E(w)$  hiperparabolična površina u prostoru sinaptičkih težina  $w$ , sa jedinstvenim globalnim ekstremumom (minimumom), postupak konvergira ka njemu bez obzira na početne uslove, pod uslovom da je  $\eta$  dovoljno malo.



Sl.11.17 Ilustracija Vidrov-Hofovog pravila obučavanja za jedan koeficijent sinaptičkih težina  $w$ .

## 11.9 JEDNOSLOJNI PERCEPTRON

Prethodni rezultat se lako može generalisati na slučaj opšte nelinearne diferencijabilne aktivacione funkcije  $a(\cdot)$ . Razmotrimo strukturu jednoslojnog perceptrona.



Sl.11.18 Jednoslojni perceptron

$$(w_{1m} = \theta_1, w_{2m} = \theta_2, \dots, w_{pm} = \theta_n)$$

$m$  – broj ulaza

$n$  – broj izlaza

$p$  – dužina obučavajućeg skupa

$$y_i^{(k)} = a(W_i^T x^{(k)}) = a\left(\sum_{j=1}^m w_j x_j^{(k)}\right) = d_i^{(k)}, \quad i = 1, 2, \dots, n, \quad k = 1, 2, \dots, p. \quad (11.16)$$

gde je

$$W_i^T = [w_{i1}, w_{i2}, \dots, w_{im}]^T \quad (11.17)$$

vektor težina pridružen neuronu i. Ako definišemo kriterijumsku funkciju kao matematičko očekivanje greške na izlazu neuronske mreže, odnosno u slučaju konačnih obučavajućih skupova u obliku ukupne kvadratne greške na obučavajućem skupu, dobijamo

$$E(w) = \frac{1}{2} \sum_{k=1}^p \sum_{i=1}^n (d_i^{(k)} - y_i^{(k)})^2 = \frac{1}{2} \sum_{k=1}^p \sum_{i=1}^n [d_i^{(k)} - a(w_i^T x^{(k)})]^2 = \frac{1}{2} \sum_{k=1}^p \sum_{i=1}^n \left[ d_i^{(k)} - a\left(\sum_{j=1}^m w_j x_j^{(k)}\right) \right]^2.$$

$$\frac{\partial E}{\partial w_j} = - \sum_{k=1}^p [d_i^{(k)} - a(net_i^{(k)})] a'(net_i^{(k)}) x_j^{(k)}, \quad (11.18)$$

$net_i^{(k)} = W_i^T x^{(k)}$  - ulaz u i-ti neuron kada je k-ti ulazni vektor prisutan.

$$a'(net_i^{(k)}) = \frac{\partial a(net_i^{(k)})}{\partial net_i^{(k)}}. \quad (11.19)$$

Korekcija  $w_{ij}$  nakon prezentacije k-tog obučavajućeg uzorka je

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_j} = \eta [d_i^{(k)} - a(net_i^{(k)})] a'(net_i^{(k)}) x_j^{(k)}, \quad (11.20)$$

i naziva se Delta pravilo obučavanja (delta learning rule), koje se iz opšteg pravila obučavanja dobija stavljanjem

$$r = [d_i - a(w_i^T x)] a'(w_i^T x). \quad (11.21)$$

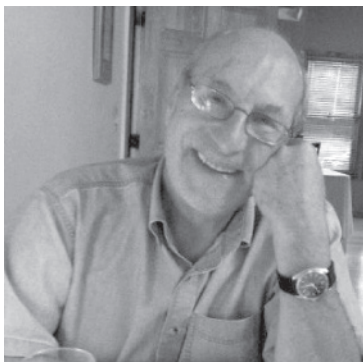
Opisana procedura konvergira ka nekom od lokalnih ekstremuma. Budući da kriterijum obučavanja poseduje više lokalnih ekstremuma, gradijentna procedura (11.20) ne garantuje globalni, već samo neki od lokalnih ekstremuma, zavisno od početnih uslova i parametara obučavanja.

## 11.10 VIŠESLOJNI PERCEPTRON

Višeslojni perceptron (Feed Forward Artificial Neural Networks - FFANN), predstavlja jednu od najvažnijih neuronskih struktura, kako zbog opštosti preslikavanja koju potencijalno može restaurirati, tako i zbog efikasnog algoritma obučavanja poznato pod nazivom algoritam propagacije greške unazad (Backpropagation Algorithm).

### 11.10.1 CIBENKOVA TEOREMA (1989)

Jedna od najvažnijih karakteristika višeslojnih neuronskih mreža je njihova sposobnost restauracije široke klase ulazno izlaznih preslikavanja, pod relativno opštim uslovima nametnutim aktivacionim funkcijama. Prvi teorijski dokaz ovog važnog stava dao je George Cybenko 1989. god. Kurt Hornik je 1991. godine pokazao da dobra aproksimirajuća svojstva višeslojnih perceptrona ne potiču od posebnih svojstava aktivacionih gunkcika, već od same arhitekture neuronske mreže.



Sl.11.19 George Cybenko (levo) i Kurt Hornik (desno). Cybenko je prvi pokazao 1989. godine da je jednoslojna neuronska mreža univerzalni aproksimator. Hornik je 1991. godine pokazao da svojstvo univerzalnog aproksimatora ne potiče od posebnih osobina aktivacionih funkcija, već od same arhitekture neuronskih mreža.

### Cibenkova Teorema

*Višeslojna neuronska mreža sa najmanje jednim skrivenim slojem i aktivacionom funkcijom koja poseduje sledeća svojstva*

1.  $\lim_{\lambda \rightarrow \infty} a(\lambda) = 1$
2.  $\lim_{\lambda \rightarrow \infty} a(\lambda) = 0 \quad (-1)$
3.  $a(\lambda)$  je neopadajuća funkcija

aproksimira bilo koju Borel merljivu funkciju na kompaktnim skupovima, sa proizvoljnom tačnošću, pod uslovom da je na raspolaganju dovoljan broj neurona u skrivenom sloju.

Borel merljive funkcije na kompaktnim skupovima obuhvataju sve neprekidne i u delovima neprekidne funkcije (sa konačno ili prebrojivo mnogo diskontinuiteta na skupovima mere nula).

Oдавde sledi da je FFANN univerzalni aproksimator. Stoga neuspeh FFANN da u nekom konkretnom slučaju restauriše preslikavanje implicitno zadato obučavajućim skupom, potiče ili od neadekvatnog izbora arhitekture, parametara obučavanja, obučavajućih skupova i drugih faktora, ali ne i od samog osnovnog restauratorskog principa FFANN.

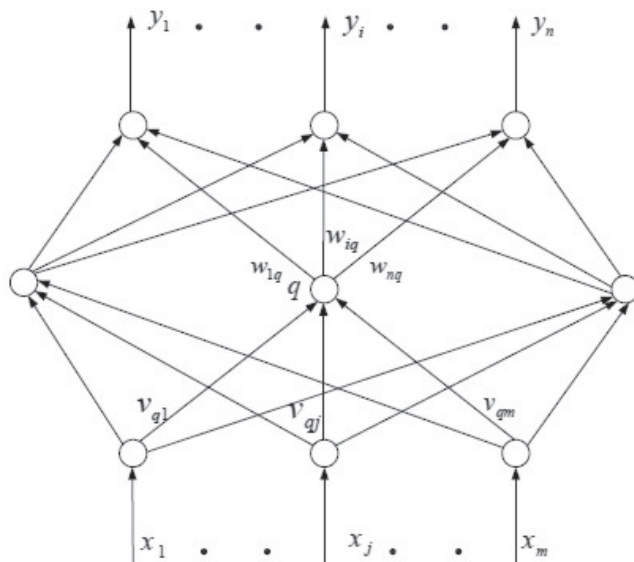
Za mnoge praktične probleme, pokazuje se da uprkos Cibenkovoj teoremi jedan skriveni sloj nije dovoljan, budući da zahteva neprihvatljivo velik broj neurona. raktično bolji rezultati se često dobijaju razmeštanjem manjeg broja neurona u dva ili više skrivenih slojeva.

### 11.10.2 ALGORITAM PROPAGACIJE GREŠKE UNAZAD

Ovaj algoritam obuhvata dve faze:

1. ulazni vektor  $x^{(k)}$  propagira od ulaznog ka izlaznom sloju, produkujući izlaz  $y^{(k)}$ .
2. sinal greške, zatim u drugoj fazi propagira unazad od izlaznog ka ulaznom sloju u cilju korigovanja težina  $w_{ij}$ .

U cilju ilustracije rada algoritma propagacije greške unazad (BP algoritam) razmotrimo višeslojni perceptron tipa  $m - l - n$  sa jednim skrivenim slojem.



Sl.11.20 Višeslojni perceptron sa jednim skrivenim slojem

Neka je neuronskoj mreži sa Sl.11.21 prezentovan par  $(x,d)$  iz zadatog obučavajućeg skupa.

Uvedimo sledeće oznake:

$net_q$  - ulazni signal u neuron q u skrivenom sloju,

$$net_q = \sum_{j=1}^m v_{qj} x_j ,$$

$z_q$  - izlazni signal neurona q

$$z_q = a(net_q) = a\left(\sum_{j=1}^m v_{qj} x_j\right)$$

Ulaz u i-ti neuron u izlaznom sloju dat je sa

$$net_i = \sum_{q=1}^l w_{iq} z_q = \sum_{q=1}^l w_{iq} a\left(\sum_{j=1}^m v_{qj} x_j\right).$$

Izlazi neurona u izlaznom sloju dati su sa

$$y_i = a(net_i) = a\left(\sum_{q=1}^l w_{iq} z_q\right) = a\left(\sum_{q=1}^l w_{iq} a\left(\sum_{j=1}^m v_{qj} x_j\right)\right).$$

Ovim je opisana prva faza, propagacija ulaznog signala. Kriterijumska funkcija obučavanja ima oblik

$$E(w) = \frac{1}{2} \sum_{i=1}^n (d_i - y_i)^2 = \frac{1}{2} \sum_{k=1}^n [d_i - a(net_i)]^2 = \frac{1}{2} \sum_{i=1}^n \left[ d_i - a\left(\sum_{q=1}^l w_{iq} z_q\right) \right]^2 .$$

U skladu sa gradijentnim postupkom ekstremizacije, korekcija težina između skrivenog i izlaznog sloja je data sa

$$\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}},$$

odnosno uzimajući u obzir relaciju o prostiranju unapred i lančano pravilo parcijalnih izvoda za  $\partial E / \partial w_{iq}$ , imamo

$$\Delta w_{iq} = -\eta \left[ \frac{\partial E}{\partial y_i} \right] \left[ \frac{\partial y_i}{\partial net_i} \right] \left[ \frac{\partial net_i}{\partial w_{iq}} \right] = \eta [d_i - y_i] [a'(net_i)] [z_q] \stackrel{\Delta}{=} \eta \delta_{0i} z_q ,$$

gde je sa  $\delta_{0i}$  označen signal greške

$$\delta_{0i} = -\frac{\partial E}{\partial net_i} = -\left[ \frac{\partial E}{\partial y_i} \right] \left[ \frac{\partial y_i}{\partial net_i} \right] = [d_i - y_i] [a'(net_i)] ,$$

gde je  $net_i$  ulaz u neuron i u izlaznom sloju, dok je

$$a'(net_i) = \frac{\partial a(net_i)}{\partial net_i} .$$

Rezultat je u potpunosti identičan Delta pravilu za jednoslojni perceptron čiji je ulaz  $z_q$  jednak izlazu neurona iz skrivenog sloja.

Korekcija težina između neurona  $j$  u ulaznom i neurona  $q$  u skrivenom sloju je data sa

$$\begin{aligned} v_{qj} &= \eta \left[ \frac{\partial E}{\partial v_{qj}} \right] = -\eta \left[ \frac{\partial E}{\partial net_q} \right] \left[ \frac{\partial net_q}{\partial v_{qj}} \right] = -\eta \left[ \frac{\partial E}{\partial z_q} \right] \left[ \frac{\partial z_q}{\partial net_q} \right] \left[ \frac{\partial net_q}{\partial v_{qj}} \right] = \\ &= \eta \sum_{i=1}^n [(d_i - y_i) a'(net_i) w_{iq}] a'(net_q) x_j. \end{aligned}$$

Korišćenjem izraza za signal greške  $\delta_{oi}$ , dobijamo

$$\Delta v_{qj} = \eta \sum_{i=1}^n [\delta_{oi} w_{iq}] a'(net_q) x_j = \eta \delta_{hq} x_j,$$

gde je  $\delta_{hq}$  signal greške za neuron  $q$  u skrivenom sloju i definiše se kao

$$\delta_{hq} = -\frac{\partial E}{\partial net_q} = -\left[ \frac{\partial E}{\partial z_q} \right] \left[ \frac{\partial z_q}{\partial net_q} \right] = a'(net_q) \sum_{i=1}^n \delta_{oi} w_{iq},$$

gde je  $net_q$  ulaz u neuron  $q$ .

Izraz za  $\delta_{hq}$  pokazuje da se ovaj signal greške za neuron  $q$  u skrivenom sloju dobija propagiranjem unazad od izlaznog sloja signala greške  $\delta_{oi}$  pridruženih izlaznim neuronima. Ovo svojstvo pokazuje važnu lokalnu osobinu algoritma, naime, da bi se izračunala korekcija koeficijenata zadate grane potrebne su samo veličine (signali) na oba kraja ove grane.

Ova razmatranja se lako mogu proširiti na perceptron sa proizvoljnim brojem slojeva, sukcesivnom primenom pravilom ulančavanja za diferenciranje. U opštem slučaju, za proizvoljan broj slojeva, pravilo korekcije težina u algoritmu propagacije greške unazad ima formu

$$\Delta w_{ij} = \eta \delta_i x_j = \eta \delta_{output-i} x_{input-j},$$

gde se „output- $i$ “ i „input- $j$ “ odnose na dva kraja konekcije neurona  $j$  ka neuronu  $i$ . Sumarno, algoritam propagacije greške unazad se može opisati kroz sledeće korake.

Neka višeslojni perceptron ima  $Q$  slojeva,  $q=1,2,...,Q$  i neka je

${}^q net_i$  - net ulaz za  $i$ -ti neuron u  $q$ -tom sloju,

${}^q y_i$  - izlaz neurona  $i$  u  $q$ -tom sloju.

Neka postoji  $m$  ulaznih i  $n$  izlaznih čvorova. Neka  ${}^q w_{ij}$  označava težinu veze između  ${}^{q-1} y_i$  i  ${}^q y_i$ .



**ULAZ:** Skup parova  $\{(x^{(k)}, d^{(k)}), k=1, 2, \dots, p\}$

**KORAK 0:** (Inicijalizacija) Izabrati  $\eta > 0$  i  $E_{max}$  (maksimalna prihvatljiva greška). Inicijalizovati sve sinaptičke težine malim slučajnim vrednostima.  $E=0, k=1$ .

**KORAK 1:** Primeniti k-ti obučavajući vektor na ulaz ( $q=1$ ):

$${}^q y_i = {}^1 y_i = x_i^{(k)}, \text{ za sve } i.$$

**KORAK 2:** (Propagacija unapred). Propagirati signal unapred do izlaza po formuli

$${}^q y_i = a({}^q net_i) = a\left(\sum_j {}^q w_j {}^{q-1} y_j\right) \quad \forall i, q$$

sve dok se ne dobije izlaz  ${}^q y_i$ .

**KORAK 3:** (Računanje izlazne greške  ${}^Q \delta_i$ )

$$E = \frac{1}{2} \sum_{i=1}^n (d_i^{(k)} - {}^Q y_i)^2 + E,$$

$${}^Q y_i = (d_i^{(k)} - {}^Q y_i) a'({}^Q net_i).$$

**KORAK 4:** (Propagacija greške unazad). Propagacija greške unazad u cilju korigovanja težina i sračunavanja greške  ${}^{q-1} \delta_i$  za prethodni sloj:

$$\Delta {}^q w_j = \eta {}^q \delta_i {}^{q-1} y_j, \quad {}^q w_j^{new} = {}^q w_j^{old} + \Delta {}^q w_j,$$

$${}^{q-1} \delta_i = a'({}^{q-1} net_i) \sum_j {}^q w_{ji} {}^q \delta_j, \text{ za } q = Q, Q-1, \dots, 2.$$

**KORAK 5:** Provera da li su svi uzorci iz obučavajućeg skupa jednom prošli proceduru. Ako je  $k < p$ , tada je  $k=k+1$  i prelazi se na korak 1. U suprotnom prelazi se na korak 6.

**KORAK 6:** (Provera ukupne greške). Da li je ukupna akumulirana greška prihvatljiva? Ako je  $E < E_{max}$ , prekida se proces obuke, u suprotnom  $E=0, k=1$ , preći na korak 1.

**END**

Ova varijanta algoritma propagacije greške unazad je tzv. inkrementalna, tj. težine se koriguju nakon predstavljanja svakog uzorka iz obučavajućeg skupa. Alternativni pristup je tzv. blokovski (batch – mod training) algoritam, po kome se težine menjaju nakon što su svi uzorci u obučavajućem skupu prezentovani.

### 11.10.3 PROBLEM KONVERGENCIJE

Površina na kojoj se traži ekstremum (error surface – površina greške) nije deterministička. Algoritam ustvari pripada klasi algoritama stohastičke aproksimacije. Za površinu greške se znaju tri bazična svojstva:

- ♦ veliki broj lokalnih minimuma, budući da postoji veliki broj kombinatornih permutacija težina koje daju isti izlaz mreže.
- ♦ postojanje lokalnih minimuma iznad nivoa globalnog minimuma
- ♦ postojanje višestrukih platoa sa malim nagibima. Ovo je direktna posledica zasićenja aktivacionih funkcija u domenu velikih signala, kada su izlazi neosetljivi na male promene težina. Postojanje ovakvih delova površine greške prouzrokuje sporu konvergenciju algoritma propagacije greške unazad.

### 11.10.4 FAKTORI KOJI UTIČU NA OBUČAVANJE ALGORITMA PROPAGACIJE GREŠKE UNAZAD

#### 11.10.4.1 Inicijalizacija težina

Početne vrednosti izuzetno utiču na krajnji rezultat obučavanja. Tipična inicijalizacija je malim slučajnim vrednostima. Velike vrednosti vode u zasićenje i zaglavljivanje u lokalnim ekstremumima bliskim startnoj poziciji. Praktična preporuka za inicijalizaciju je izbor početnih težina u opsegu:

$$\left[ -\frac{3}{\sqrt{k_i}}, \frac{3}{\sqrt{k_i}} \right],$$

gde je  $k_i$  broj ulaznih konekcija u neuron  $i$ .

#### 11.10.4.2 Koeficijent obučavanja (**learning constant**)

Velike vrednost za  $\eta$  mogu da ubrzaju konvergenciju, ali i da dovedu do premašaja cilja, dok isuviše male vrednosti imaju suprotan efekat. Dosadašnja praksa pokazuje da se  $\eta$  može kretati, zavisno od konkretnog problema u opsegu od 0.001 do 10. Dobra strategija je adaptivna promena za  $\eta$ , npr. po sledećem zakonu

$$\Delta\eta = \begin{cases} a, & \Delta E < 0, \text{ konzistentno} \\ -b\mu, & \Delta E > 0 \\ 0, & \text{u ostalim slučajevima} \end{cases}, \quad a, b > 0$$

Konzistentno, može da ima značenje ili npr. K uzastopnih koraka ili težinsko pokretno usrednjavanje  $\Delta E$ .

### 11.10.4.3 Funkcija cilja

Kvadratna funkcija nije jedini mogući izbor. promenom ove funkcije menja se samo signal greške  $\delta_{oi}$  u izlaznom sloju, dok ostale jednačine ostaju nepromenjene. Mogući izbori funkcije greške su  $L_p$  norma

$$E = \frac{1}{p} \sum_i (d_i - y_i)^p, \quad 1 \leq p < \infty,$$

Čebiševljeva norma

$$L_\infty = \sup_i |d_i - y_i|.$$

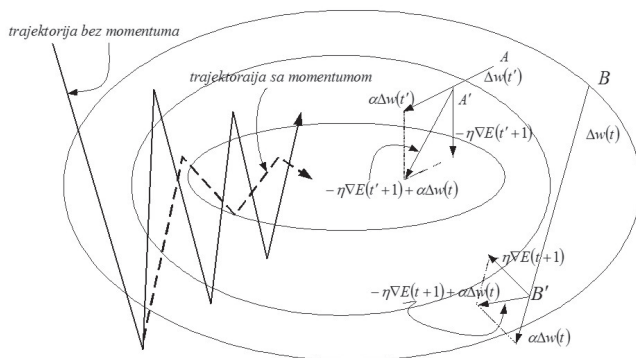
### 11.10.4.4 Momentum

Jedan od načina da se konstanta obučavanja poveća, a da ne dodje do divergentnog oscilovanja je dodavanje tzv. momentum člana. Momentum je u stvari dodatni inercijalni član koji omogućava odvijanje procesa obučavanja u pravcu „srednje sile na dole“. Ovo se može ostvariti uključivanjem prethodnih promena težina u trenutnu promenu, npr. na sledeći način

$$\Delta w(t) = -\eta \nabla E(t) + \alpha \Delta w(t-1), \quad \alpha \in [0,1],$$

gde je  $\alpha$  momentum parametar (uobičajena praktična vrednost je 0.9).

Na Sl.11.21 prikazana je analiza uticaja momentuma na proces korekcije težina neuronske mreže u toku obučavanja. Elipse prikazuju izohipse hipotetičke površine greške (error surface) kvadratnog tipa. Primer  $AA'$  ilustruje slučaj dobrog usmerenja vektora korekcije težina. Korekcija momentumom u ovom slučaju poboljšava usmerenje korekcije težina. Primer  $BB'$  ilustruje slučaj pogrešno usmerenog vektora korekcije težina (prebačaj). Korekcija momentumom preusmerava ovaj vektor u dobrom pravcu. Ovi primeri pokazuju da momentum tipično ubrzava konvergenciju.



Sl.11.21 Uticaj momentuma na konvergenciju težina u toku obučavanja

#### 11.10.4.5 Pravila korekcije

Do sada analizirano pravilo korekcije težina se zasnivalo na najjednostavnijem postupku gradijentnog spusta. Dobro razvijena teorija optimizacije nudi niz daleko razvijenijih i efikasnijih tehnika. Prvo poboljšanje se može učiniti uključivanjem viših redova kriterijumske funkcije. Ako  $E(w)$  razvijemo u tajlorov red dobijamo

$$E(w) = E(w_0) + (w - w_0)^T \nabla E(w_0) + \frac{1}{2} (w - w_0)^T H(w) (w - w_0) + \dots$$

gde je

$$H(w) = \nabla^2 E(w) \quad , \quad H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} .$$

Da bi smo našli minimum od  $E(w)$ , stavljamo  $\nabla E(w)=0$ , odnosno

$$\nabla E(w) = \nabla E(w_0) + H(w_0)(w - w_0) + \dots = 0 .$$

Ako zanemarimo članove reda većeg od dva u gornjem razvoju, dobijamo

$$w = w_0 - H^{-1}(w) \nabla E(w_0) ,$$

ili u iterativnoj proceduri

$$w^{(k+1)} = w^{(k)} - H^{-1}(w^{(k)}) \nabla E(w^{(k)}) ,$$

što je poznat Njutnov metod korekcije težina, za koga se dokazuje da u slučaju konveksnih kriterijumskih funkcija  $E$ , konvergira kvadratno ka rešenju. Međutim i dalje procedura ima niz nedostataka:

- ♦ računarska kompleksnost
- ♦ zahteva dobro početno pogadjanje
- ♦ za ne konveksne kriterijumske funkcije može da konvergira ka lokalnom ekstremumu i sedlastim tačkama.

Računarski relaksirana metoda pogodna za implementaciju je npr. kvazi Njutnova metoda ili algoritam konjugovanih pravaca.

#### 11.10.4.6 Obučavajući skup i generalizacija.

Algoritam propagacije greške umazad ima dobra svojstva generalizacije. Neuronska mreža dobro generalizuje ukoliko daje dobre interpolacije za nove ulaze, koji nisu bili prisutni u postupku obučavanja. Neuronska mreža sa isuviše slobodnih parametara za zadati obučavajući skup može biti dobro obučena, sa velikom verovatnoćom loše generalizacije. Ovaj fenomen se naziva overfitting. Ukoliko međutim mreža ima isuviše malo slobodnih parametara, nije u stanju da se obuči na obučavajućem skupu, a samim tim ima loše performanse i na test skupu (skup za testiranje obuhvata primere koji ne pripadaju obučavajućem skupu). Budući da generalizacija predstavlja važno svojstvo, razvijeno je više procedura za njeno poboljšanje.

- a. **Smanjivanje osetljivosti mreže.** Da bi neuronska mreža posedovala dobra svojstva generalizacije, potrebno je da male promene ulaznih signala ne izazivaju velike promene na izlazu neuronske mreže. Jedan od mogućih načina za poboljšanje generalizacije direktnom primenom ovog principa je proširivanje obučavajućeg skupa varijacijama ulaznih signala, recimo dodavanjem šuma niskog nivoa oko svakog elementa obučavajućeg skupa. Formalno, obučavajući skup

$$\{ (x_i, d_i) \} \text{ se zamenjuje sa } \{ (x_i + \xi_j), d_i \}, j = 1, \dots, m, \quad i = 1, \dots, p.$$

Druga mogućnost postizanja sličnog efekta smanjivanja osetljivosti mreže u odnosu na ulazne signale je dodavanje novog člana standardnoj kriterijumskoj funkciji oblika

$$E_b = \frac{1}{2} \left[ \left( \frac{\partial E_f}{\partial x_1} \right)^2 + \left( \frac{\partial E_f}{\partial x_2} \right)^2 + \dots + \left( \frac{\partial E_f}{\partial x_n} \right)^2 \right],$$

gde je  $E_f$  funkcional greške u standardnom algoritmu obučavanja. Razlog za uključivanje  $E_b$  je da njegova minimizacija u stvari znači, prema gornjoj definiciji, malu osetljivost  $E_f$  na varijacije ulaza, što je i bio cilj.

- b. **Regularizacija.** Ovaj metod se svodi na proširivanje kriterijumske funkcije tzv. regularizacionim članom

$$\tilde{E} = E + v\Omega$$

gde je  $E$  standardni kriterijum,  $v$  je parametar kojim se kontroliše uticaj dodatnog člana  $\Omega$ , koji je u direktnoj vezi sa kompleksnošću neuronske mreže. Na taj način, minimizacijom ukupnog kriterijuma  $\tilde{E}$  postiže se uslovna ekstremizacija standardnog kriterijuma  $E$  uz uslov minimalne kompleksnosti neuronske mreže, koja je osnovni uzrok overfittinga. Najčešće korišćen oblik regularizacionog člana je

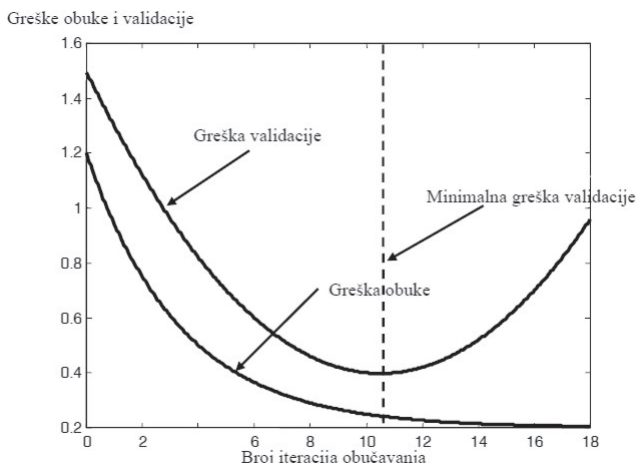
$$\Omega = \frac{1}{2} \sum_i w_i^2,$$

poznat pod nazivom weight decay – smanjivanje težina, pri čemu se suma odnosi na sve težine i bajase u mreži. Praksa pokazuje da se na ovaj način postiže značajno poboljšanje generalizacije. Moguće heurističko objašnjenje ovog efekta se svodi na sledeće rezonovanje. Ukoliko su težine mreže velike, aktivacije neurona su u predelu zasićenja, dakle nelinearnih oblasti koje upravo prouzrokuju kompleksna preslikavanja mreže, i obrnuto, za male vrednosti težina aktivacije su u predelu linearnosti aktivacionih funkcija i kompleksnost mreže je mala, čime se smanjuje verovatnoća overfittinga za fiksiranu dužinu obučavajućeg skupa.

- c. **Rano zaustavljanje (Early stopping).**

Tokom tipične procedure obučavanja greška obučavanja (vrednost kriterijumske funkcije) po pravilu opada sa brojem iteracija obučavanja. Međutim greška merena na skupu podataka nezavisnih od obučavajućeg skupa (test ili validacioni skup) po pravilu opada do jedne određene vrednosti iteracija, a zatim počinje da raste,

Sl.11.21. Ovaj rast je vezan za pojavu overfittinga, pa je stoga celishodno proceduru obučavanja zaustaviti u toj tački, iako kriterijumska funkcija na obučavajućem skupu i dalje opada. Otuda naziv ove metode – rano zaustavljanje.



Sl.11.22 Tipični oblik zavisnosti greške obučavanja i validacije

#### d. Kresanje (Prunning)

Budući da je generalizacija vezana za adekvatan odnos izmedju broja slobodnih parametara (bogatstvo arhitekture) i složenosti modelovane pojave (struktura i dužina obučavajućeg skupa), ideja kresanja se temelji na principu ostvarivanja što boljih performansi sa što siromašnijom arhitekturom. Operativno se ovo može postići sledećom klasom procedura:

**Korak.1.** Izabrati početnu bogatu arhitekturu neuronske mreže. Obučiti zatim. neuronsku mrežu na obučavajućem skupu i testirati na validacionom skupu. Neka su vrednosti ovih kriterijuma  $E_{obuka}$  i  $E_{valid}$ .

**Korak.2.** Saobrazno nekom od unapred usvojenih kriterijuma značajnosti parametra, iz računati značajnost svih parametara obučene neuronske mreže i sortirati ih po rastućim vrednostima, tako da se na prvom mestu nalazi najneznačajniji parametar.

**Korak.3.** Izbaciti granu lil bajas koji odgovara najneznačajnijem parametru. Na ovaj način smo smanjili složenost arhitekture i broj slobodnih parametara

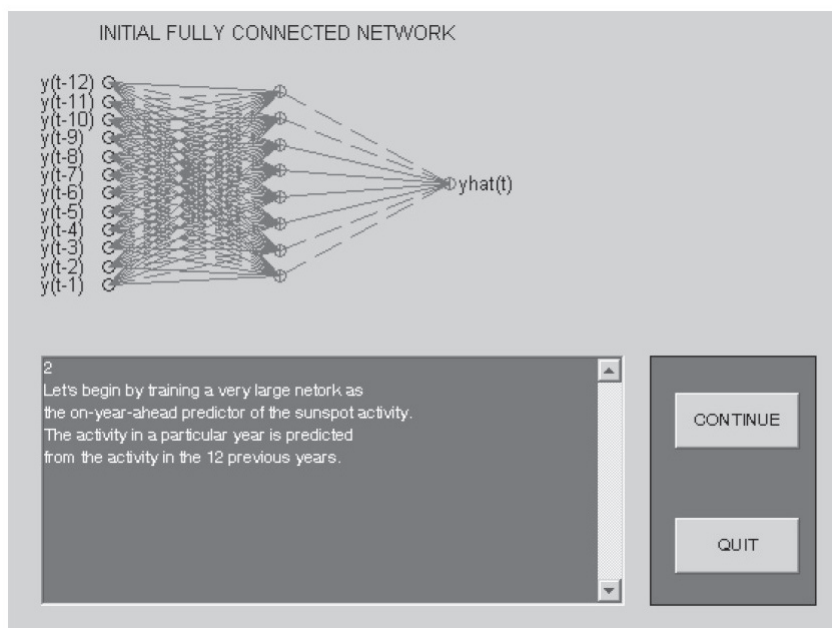
**Korak.4.** Za novu arhitekturu izvršiti novu slučajnu inicijalizaciju mreže i novo obučavanje. Izračunati ponovo  $E'_{valid}$ . Ako je  $E'_{valid} \leq E_{valid}$ , staviti da je  $E_{valid} = E'_{valid}$  i preći na korak 2., u suprotnom zaustaviti proceduru.

Nakon završetka rada ovog algoritma, posedovaćemo neuronsku mrežu najsiromašnije arhitekture i minimalne vrednosti kriterijumske funkcije na validacionom skupu, što i jeste bio cilj.

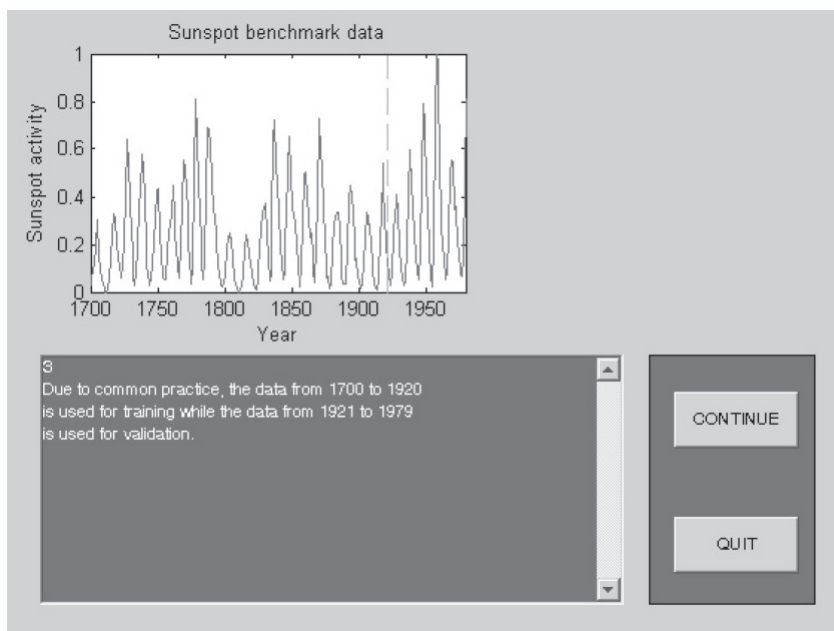
Postavlja se pitanje izbora kriterijuma značajnosti. Jedan od češće korišćenih je tzv. kriterijum oštećenja mozga (brain damage), po kome je značajniji onaj parametar za koga je vezana veća promena kriterijumske funkcije nakon njegovog uklanjanja. Pored ovog kriterijuma i njegovih različitih varijanti, razvijeni su i drugi, vezani prevashodno za različite statističke testove značajnosti nelinearnih parametarskih modela.

## PRIMER 11.1

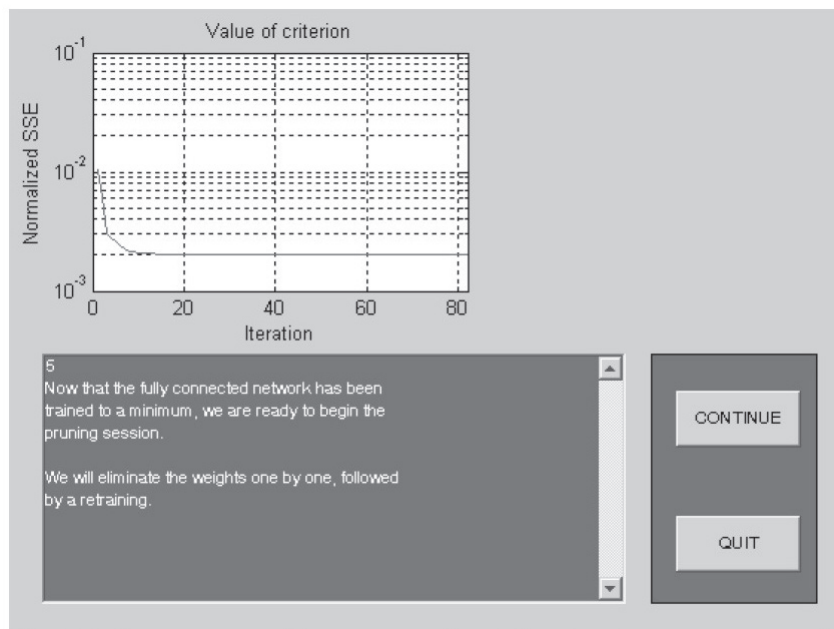
U narednoj seriji slika dat je primer izbora arhitekture sa dobrim generalizacionim svojstvima, u problemu predviđanja broja sunčevih pega. Neka nam je na raspolaganju vremenska serija broja sunčevih pega od 1700. god. do 1799. god. Neka se vremenska serija u periodu (1700-1920) koristi za trening neuronskog prediktora, a deo vremenske serije (1921 - 1979) za validaciju i testiranje. Eksperimenti su izvedeni sa NNSYSID MATLAB Toolbox-om, <http://www.iau.dtu.dk/research/control/nnsysid.html>, razvijenog uz knjigu: *Neural Networks for Modelling and Control of Dynamic Systems* by Magnus Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen Springer-Verlag, London, 2000.



Sl.11.23 Početna arhitektura neuronskog prediktora 12. reda, budući da koristi 12 prošlih godina za predviđanje sunčevih pega u tekućoj godini. Primenjena neuronska mreža ima 12 ulaza, 8 neurona u skrivenom sloju i jedan izlazni neuron. Po svojoj prirodi to je jednoslojni perceptron sa prostiranjem signala unapred.

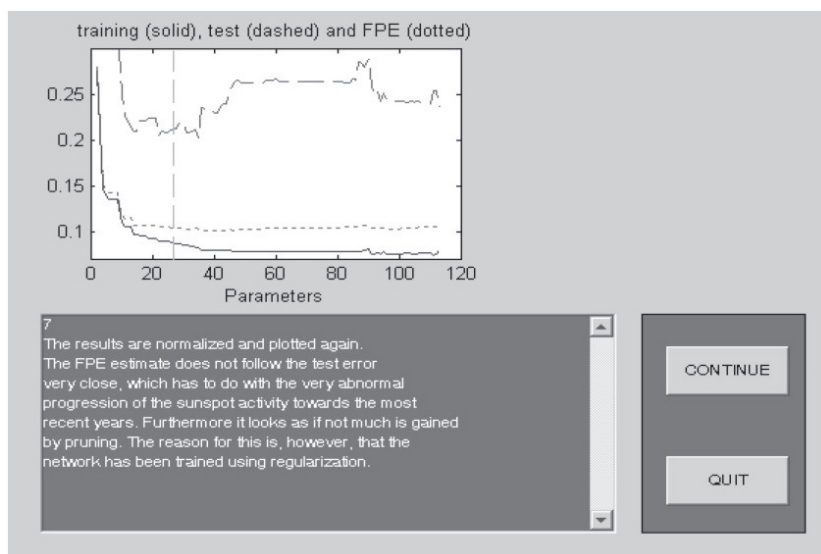


Sl.11.24 Prikaz analizirane vremenske serije broja sunčanih pega u periodu (1700-1979). Posebno je označen interval treninga (1700-1920) i validacije (1921-1979).

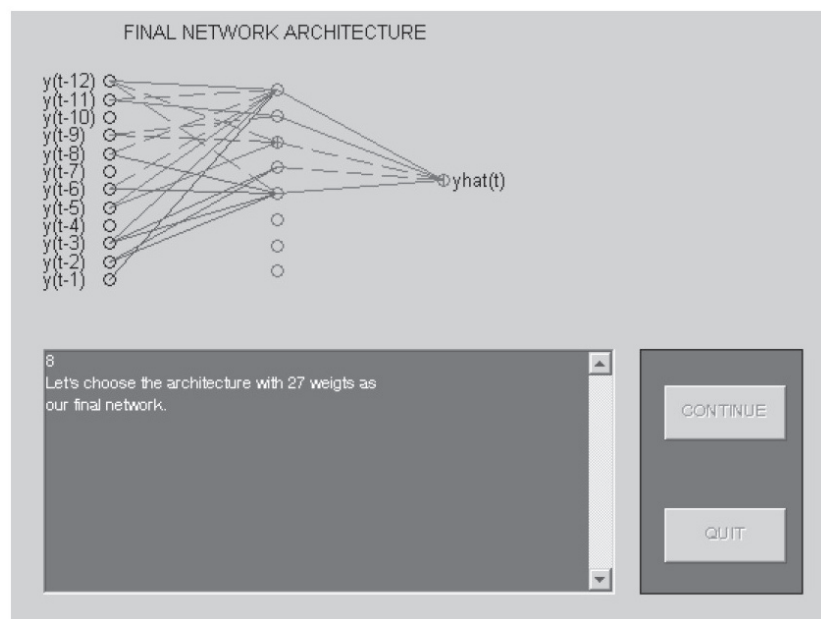


Sl.11.25 Prikaz promene greške treninga u toku obučavanja početne neuronske mreže sa maksimalnim brojem sinaptičkih veza.





Sl.11.26 Prikaz tri vrste greške u toku procesa kresanja pune arhitekture: greška na treningu (puna zelena linija), greška na test skupu (isprekidana crvena linija), procena finalne greške predikcije (Final Prediction Error –FPE) u funkciji broja parametara neuronske mreže. Procena FPE daje u stvari procenu greške generalizacije.



Sl. 11.27 Na osnovu analize zavisnosti FPE od broja parametara, Sl. 11.25, zaključujemo da se minimum FPE postiže za arhitekturu od 27 parametara, koja je i prikazana na ovoj slici. Uočavamo da je za dobru predikciju dovoljno imati 4 neurona u skrivenom sloju. Primetimo da sinaptičke veze nacrtane punom linijom odgovaraju ekscitirajućim, a isprekidanom linijom inhibirajućim vezama.

Primer jasno pokazuje da siromašna arhitektura od 27 slobodnih parametara daje značajno bolje rezultate od potpune arhitekture sa 113 slobodnih parametara (104 sinaptičke veze i 9 bajasa), potvrđujući princip da složenost arhitekture mora biti prilagođena složenosti fenomena koga opisuje.

#### 11.10.4.7 Broj skrivenih neurona

Pitanje broja neurona u skrivenom sloju je fundamentalno pitanje koje se nezaobilazno javlja gotovo u svakoj primeni višeslojnog perceptrona. Egzaktanu analizu je teško sprovesti, budući da je preslikavanje koje ostvaruje višeslojni perceptron veoma kompleksno, kao i usled stohastičke prirode većine algoritama obučavanja. Praktične preporuke se svode na princip: probaj sa početnim brojem neurona u skrivenom sloju znatno manjim od dimenzije ulaznog sloja. Ako je obučavanje zadovoljavajuće, pokušati sa daljnjim smanjivanjem, u suprotnom, inkrementalno povećavati njihov broj. postoje i određene analitički zasnovane analize o ovom broju. Ekvivalentirajmo pitanje odgovarajućeg broja neurona u skrivenom sloju sa pitanjem koliko je tih neurona potrebno da bi se u  $m$  dimenzionom ulaznom prostoru formiralo  $M$  disjunktne oblasti na kojima mreža ima konstantne izlaze, razdvojene međusobno hiperravnima. Ako taj broj obeležimo sa  $N_m$ , tada je u važnosti

$$N_m + 1 \leq M \leq \sum_{j=0}^m \binom{N_m}{j}, \quad \text{gde je } \binom{N_m}{j} = 0, \text{ za } N_m < j.$$

Maksimalan broj linearne separabilne oblasti upotrebom  $N_m$  skrivenih neurona u  $m$ -dimenzionom prostoru,  $M_{max}$  je dat sa

$$M_{max} = \sum_{j=0}^m \binom{N_m}{j} = 1 + N_m + \frac{N_m(N_m-1)}{2!} + \dots + \frac{N_m(N_m-1)\dots(N_m-j+1)}{m!}, \text{ za } N_m > m.$$

U slučaju  $N_m \leq m$ , važi  $M_{max} = 2^{N_m}$ , odnosno  $N_m = \log_2 M_{max}$ . Podsetimo se da je broj disjunktne particije u izlaznom prostoru neuronske mreže u direktnoj vezi sa maksimalnim brojem koncepata (klasa) koje ta mreža može da prepozna.

### 11.11 UPRAVLJANJE NEURO RAČUNARSKIM PROJEKTIMA

Upravljanje projektima koji koriste informacione tehnologije danas predstavlja zasebnu, dobro zasnovanu i razvijenu disciplinu. Pored mnogih sličnosti, upravljanje projektima iz domena neuroračunarstva zahteva i posebne specifičnosti. Njih moraju imati u vidu ne samo rukovodioci projekta, već i projektanti i izvršioci. Stoga je cilj ovog kraćeg pregleda osobenosti sinteze sistema zasnovanih na neuronskim mrežama olakšavanje upravljanja ovim tipom projekata uz istovremeno ukazivanje projektantima na kritične tačke realizacije i moguće teškoće, koje se blagovremenim uočavanjem mogu ublažiti ili otkloniti.

### 11.11.1.OSNOVNA SVOJSTVA NEURORAČUNARSKIH PROJEKATA

1. Projekat je uslovljen podacima. Naime, analiza i prikupljanje podataka za obučavanje, validaciju i testiranje neuronskih mreža je integralni deo procedure projektovanja. Prikupljanje podataka u ove svrhe je po pravilu osetljiv i vremenski zahtevan postupak, čije se trajanje najčešće podcenjuje.
2. Po pravilu, nije moguće specificirati konačno rešenje sistema u fazi projektovanja. Ovo nameće neophodnost sinteze prototipa, pomoću koga se, kroz odgovarajuće organizovane eksperimente razrešavaju otvoreni problemi i dolazi do konačnog rešenja koje zadovoljava postavljene zahteve.
3. Ključna karakteristika ove klase sistema je tačnost, a ne brzina obrade. Stoga je neophodno posvetiti punu pažnju ovom kriterijumu kroz sve faze realizacije projekta: od analize zahteva, projektovanja, pa do testiranja konačnog rešenja. Posebnu teškoću predstavlja praktično pokazivanje da ponudjeno rešenje zadovoljava postavljene zahteve u pogledu tačnosti.

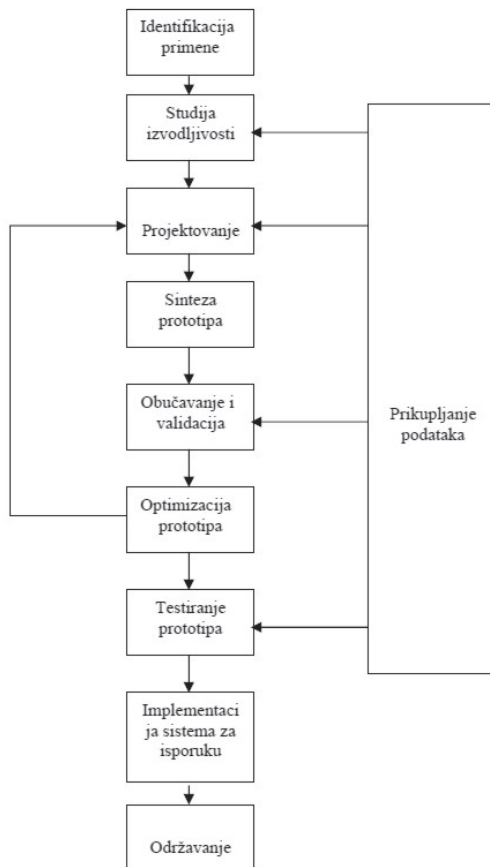
Ove specifičnosti posebno utiču na sledeće domene upravljanja projektom:

- ♦ **Planiranje projekta** - mora obuhvatiti sve specifične faze neuroračunarskog projekta, kao što su
  - prikupljanje podataka
  - sinteza prototipa
  - obučavanje, validacija i testiranje sistema
  - revizija ciklusa projektovanja i validacije sistema.
- ♦ **Upravljanje projektom** - Iterativna priroda sinteze sistema zasnovanih na neuronskim mrežama zahteva specifične tehnike planiranja, kontrole i revizije.
- ♦ **Upravljanje konfiguracijom** - Podaci koji formiraju obučavajuće skupove, kao i podaci koji opisuju rezultate sprovedenih eksperimenata moraju biti tako integrisani, da pružaju jedinstvenu celinu prilikom razmatranja različitih varijanti rešenja.
- ♦ **Dokumentacija** - Svi eksperimentalni rezultati moraju biti tako dokumentovani da obezbede preglednost obimnog eksperimentisanja uz istovremenu potpunost, tako da se bilo koji eksperiment može ponoviti pod egzaktno istim uslovima.

### 11.11.2. OSNOVNE FAZE PROJEKTA

Na sl.1 date su osnovne faze tipičnog neuroračunarskog projekta, od faze formulacije do implementacije. Ciklus razvoja obuhvata tri osnovne faze:

1. Identifikacija moguće primene i studija ostvarljivosti
2. Razvoj i validacija prototipa
3. Konverzija prototipa u konačno rešenje, uvođenje sistema i održavanje.



Sl.11.28 Tipičan životni ciklus neuroračunarskog projekta

Cilj prve faze je istovetan kao i kod projekata zasnovanih na informacionim tehnologijama, tj. identifikacija potencijalne primene i studija ostvarljivosti uz odgovarajuću tehnoeкономsku analizu razvoja i uvođenja rešenja. Studija ostvarljivosti je praćena posebnim dokumentom - **specifikacijom zahteva**. Uobičajeno je da kod ovog tipa projekata osnovni zahtev bude usmeren ka tačnosti sistema, mada mogu biti prisutni i zahtevi u pogledu ograničenja arhitekture, npr. broja ulaznih čvorova.

Razvoj prototipa je iterativni postupak sukcesivnog poboljšanja i optimizacije neuronskog sistema, sl.1. čime se omogućava razrešavanje problema sinteze kroz eksperimentisanje. Evidentne prednosti ovakvog pristupa su:

- ♦ eksperimenti se mogu obavljati i na pogodno odabranom podsistemu celovitog rešenja,
- ♦ prototip i konačno rešenje mogu biti implementirani na različitim hardversko - softverskim platformama, čime se za svaku fazu razvoja mogu izabrati najpogodnije varijante,
- ♦ prototip može biti strogo testiran i verifikovan pre nego što se pristupi njegovoj konverziji u konačno rešenje.

Finalna faza konverzije prototipa u konačno rešenje, obuhvata eventualno reprogramiranje u operativno softversko rešenje, kao i implementaciju dodatnih funkcija, npr. specifičnih korisničkih interfejsa. Ovu fazu implementiraju softver - inženjeri i kao takva se malo razlikuje od ekvivalentnih faza implementacije klasičnih informacionih sistema.

### 11.11.3 PLANIRANJE PROJEKTA

Planiranje i upravljanje projektom se može posmatrati kao aktivnost u okviru tri faze: planiranje (u cilju ostvarivanja postavljenih zahteva), realizacija (ostvarivanje plana uz nadgledanje i kontrolu aktivnosti) i analiza - revizija (određivanje šta je uradjeno a šta ne, uz donošenje zaključaka koji služe kao ulaz u naredni ciklus planiranih aktivnosti).

U odnosu na konvencionalne informacione projekte, neuroračunarski projekti zahtevaju posebno planiranje faze razvoja prototipa i kolekcije podataka.

#### 11.11.3.1 PLANIRANJE RAZVOJA PROTOTIPA

Prototipski razvoj unosi dodatne elemente neodređenosti, kao što su:

- ♦ broj ciklusa neophodnih za razvoj prototipa
- ♦ vreme i resursi neophodni za razvoj prototipa
- ♦ vreme i resursi neophodni za kolekciju neophodnih podataka

Posebno je važno odrediti ciljeve svakog ciklusa razvoja prototipa. Praksa je pokazala da je bolje usvojiti veći broj kraćih ciklusa iako to zahteva više napora za upravljanje i kontrolu.

#### 11.11.3.2. PLANIRANJE PRIKUPLJANJA PODATAKA

Praksa pokazuje da je ovaj deo planiranja neuroračunarskih projekata jedan od najkritičnijih. Istaknimo neke od tipičnih teškoća:

- ♦ prikupljanje podataka može da obuhvata personal i organizacije koje nisu pod direktnom kontrolom rukovodioca projekta
- ♦ neophodnost korišćenja kompleksne i nedovoljno poznate akvizicione opreme
- ♦ prikupljanje podataka je isprekidano svakodnevnim radnim aktivnostima date radne organizacije.

Budući da ova aktivnost dominantno utiče na vreme realizacije celokupnog projekta, posebno se moraju razmotriti sledeće podaktivnosti:

- ♦ provera celokupne glavne procedure kolekcije podataka
- ♦ konverzija podataka, ukoliko je neophodna,
- ♦ provera kvaliteta podataka.

Poželjno je da procedura prikupljanja podataka bude praćena odgovarajućom dokumentacijom, naročito ako se obavlja od strane personala koji nije uključen u projekt.

Provera kvaliteta podataka je ključna, budući da pogrešno obeležavanje kategorijalnih promenljivih ili pogrešno očitavanje numeričkih vrednosti standardnih promenljivih, ima ozbiljne reperkusije na celokupni sintetisani prototip obučen na pogrešnim obučavajućim skupovima.

#### 11.11.4 ANALIZA-REVIZIJA

Revizija neuroračunarskih projekata obuhvata:

- ♦ reviziju podataka
- ♦ reviziju na kraju svakog ciklusa razvoja prototipa
- ♦ reviziju na kraju celokupne faze razvoja prototipa.

Reviziju podataka treba izvršiti na kraju faze prikupljanja podataka. Cilj revizije je utvrđivanje konačnog značenja svake varijable i procena da li su raspoloživi podaci dovoljni za postizanje ciljeva projekta. Osim toga potrebno je izvršiti podelu raspoloživih podataka na obučavajući, validacioni i test skup.

Na kraju svakog ciklusa razvoja prototipa potrebno je analizirati sledeće:

- ♦ šta je postignuto u cilju ostvarivanja postavljenih zahteva
- ♦ da li su postignute željene performanse
- ♦ da li je razvoj prototipa kompletiran
- ♦ ako nije, koje promene je neophodno implementirati.

Rezultat ovih analiza predstavlja osnovu za planiranje narednog ciklusa razvoja prototipa.

Revizija na kraju faze razvoja prototipa je vitalna za narednu fazu implementacije konačnog rešenja. Između ostalog potrebno je odlučiti:

- ♦ da li je potrebno izvršiti minorne izmene u cilju konačne implementacije
- ♦ kada i kako implementirati konačan sistem.

#### 11.11.5 UPRAVLJANJE KONFIGURACIJOM

Neuroračunarski projekti nameću posebne zahteve u pogledu organizovanja i upravljanja podacima, eksperimentalnim uslovima i rezultatima. U fazi razvoja prototipa po pravilu se generiše velika količina eksperimentalnih rezultata, neophodnih za sintezu konačnog rešenja. Cilj ove komponente upravljanja projektom je obezbeđivanje adekvatnog načina zapisivanja ovih podataka. Opisi treba da obuhvate takav nivo detaljnosti svih eksperimentalnih uslova kojim je omogućeno egzaktno ponavljanje svakog relevantnog eksperimenta. U tom smislu, tipično se vodi evidencija o sledećem:

- ♦ verzija softvera koji se koristi za obučavanje neuronske mreže, pre i post procesiranje podataka
- ♦ arhitektura neuronske mreže,
- ♦ svi parametri koji određuju transformacije ulaznih podataka u fazi predobrade i postobrade

- ♦ skupovi podataka koji se koriste za obučavanje, validaciju i testiranje
- ♦ parametri procedure obučavanja, npr. broj ciklusa, kriterijum zaustavljanja procedure obučavanja, vrednosti koeficijenata brzine obučavanja i momentuma, i td.
- ♦ koeficijenti neuronske mreže - inicijalne i krajnje vrednosti.

#### 11.11.6 DOKUMENTACIJA

Poseban deo deokumentacije predstavlja sumarni izveštaj o performansama svih algoritama isprobanih u fazi projektovanja prototipa. Obuhvata pored kratkog opisa svakog od upotrebljenih i testiranih algoritama i opseg promena parametara u toku testiranja, kao i sumarne statističke karakteristike ostvarenih performansi.

#### 11.11.7 IMPLEMENTACIJA KONAČNOG SISTEMA

Nakon faze prototipskog razvoja, sledi implementacija konačnog rešenja u operativno okruženje. Neophodna intervencija na prototipu može da varira u širokom opsegu - od minornih promena pa do potpunog redizajna uključujući i eventualnu implementaciju na hardveru posebne namene. Postoji niz razloga za kompletnim redizajnom, npr.:

- ♦ radno okruženje u koje će se ugraditi sistem se razlikuje od okruženje u kome je razvijan prototip
- ♦ konačan sistem zahteva posebne performanse, npr. rad u realnom vremenu
- ♦ konačan sistem zahteva posebne interfejs, npr. prema drugim radnim podsystemima ili operaterima
- ♦ konačan sistem zahteva poseban nivo pouzdanosti rada.

Nakon konačne implementacije, sistem u radnom težimu treba da demonstrira zahtevane performanse. U toku pripreme plana testiranja sistema, treba imati u vidu sledeće:

- ♦ usvajanje jednog ili više kriterijuma kvaliteta, npr. procenat tačne klasifikacije, relativna ili apsolutna tačnost u slučaju regresione primene
- ♦ garantovani i željeni nivo performansi
- ♦ statistički testovi i intervali poverenja, koji defakto određuju broj neophodnih testiranja
- ♦ adekvatnost test skupova, posebno u graničnim režimima rada sistema.

#### 11.11.8 PREUZIMANJE I ODRŽAVANJE SISTEMA

Krajnji korisnik sistema treba biti pripremljen na sva ograničenja sistema zasnovanog na tehnologiji neuronskih mreža, posebno u uslovima odstupanja radnog režima od režima u toku koga su prikupljeni podaci za obučavanje, validaciju i testiranje

sistema u fazi razvoja. Stoga je i pojam održavanja neuroračunarskog sistema širi od uobičajenog pojma održavanja klasičnih softverskih rešenja. Naime održavanje sada podrazumeva i održavanje kako performansi sistema tako i njegove funkcionalnosti. Svaka izmena radnog režima koja prevazilazi normalno odstupanje od režima za koji je neuronski sistem obučen, zahteva preobučavanje sistema, koje se u slučaju konstantnog menjanja radnih uslova može prevesti u redovnu proceduru sa unapred zadatom učestanošću ponavljanja.

## Rezime 11. poglavlja

- ♦ Neuronske mreže predstavljaju adaptivne distribuirane sisteme zasnovane na jednostavnim procesnim elementima - neuronima.
- ♦ Neuronska mreža je određena svojom arhitekturom, smerom prostiranja signala i algoritmom obučavanja. Mogu da rade u režimu obučavanja, samoobučavanja ili podsticanja.
- ♦ Algoritam obučavanja u višeslojnim perceptronima sa prostiranjem signala unapred je čuveni BP-algoritam. To je stohastička gradijentna procedura koja garantuje dostizanje lokalnih ekstremuma kriterijumske funkcije.
- ♦ Generalizaciona svojstva obučених neuronskih mreža mogu biti poboljšana postupcima dodavanja šuma u fazi obučavanja, smanjivanja osetljivosti mreže, uvođenjem regularizacije u kriterijumsku funkciju ili postupkom kresanja, kojim se iterativno osiromašuje arhitektura uz očuvanje tačnosti.
- ♦ Vodjenje neuronskih projekata se u znatnoj meri razlikuje od vodjenja informatičkih projekata zasnovanih na klasičnim tehnologijama.

## Pitanja i zadaci

1. Objasnite svojim rečima razliku između ljudskog mozga i savremenih digitalnih računara.
2. Zašto je potrebno da aktivacione funkcije u modelu neurona budu neprkidne?
3. Šta je smisao Hebovog učenja?
4. Ako je prema Cibenkovoj teoremi dovoljno imati samo jedan skriveni sloj neurona, da li uopšte i zašto imamo praktične potrebe za neuronskim mrežama sa više skrivenih slojeva?
5. Ako BP algoritam puštamo da radi više puta za isti obučavajući skup, da li će i rezultujuća obučena neuronska mreža biti ista ili različita?
6. Kako bi ste eksperimentalno odredili značaj pojedinih ulaza u obučenu neuronsku mrežu u odnosu na zadatak koji obavlja?
7. Kako bi ste dizajnirali sistem za nabolje predviđanje sledećeg izvlačenja kombinacija u igri Loto, ako su vam na raspolaganju dobitne kombinacije iz prošlih izvlačenja?



8. Otidite na web lokaciju <https://archive.ics.uci.edu/ml/datasets.html>, odaberite jedan set podataka za problem klasifikacije i izvršite sintezu jednog neuronskog klasifikatora. Procenite njegovu klasifikacionu tačnost. Sugestija: koristite neki od javno dostupnih paketa za neuronske mreže (NNSYSID, WEKA, R, CLOP, SPIDER,...)
9. Opišite korake u sintezi sistema za određivanje najbolje strategije poslovanja na berzi, ako su vam dostupne vremenske serije kretanja pojedinih akcija.
10. Kako bi se mogao napraviti sistem za automatsko dekriptiranje algoritma DES ili AES, ukoliko su vam na raspolaganju proizvoljne količine parova otvoreni tekst, šifrat, pod uslovom da se unutrašnji tajni ključ ne menja?
11. Kako bi se mogao napraviti sistem za automatsku medicinsku dijagnostiku, ukoliko su vam na raspolaganju dovoljne količine podataka (simptomi, dijagnoza)?

## 12. Duboko obučavanje neuronskim mrežama

**Duboko obučavanje (deep learning)** je nova podoblast mašinskog učenja koja se oslanja na algoritme učenja višenivovskih reprezentacija u cilju modelovanja kompleksnih relacija između ulaznih podataka u cilju njihove analize i prepoznavanja-klasifikacije. Obeležja (features) i koncepti na višim hijerarhijskim nivoima se formiraju na osnovu obeležja i koncepata sa nižih hijerarhijskih nivoa. Struktura sistema mašinskog učenja koja obezbeđuje ovakvu hijerarhiju obeležja se naziva **duboka arhitektura**.

Duboko obučavanje je deo šire oblasti mašinskog učenja koje se odnosi na **učenje reprezentacija**. Jedna observacija, npr. slika, se može reprezentovati na bezbroj načina: vektorima piksela, koeficijentima ortogonalnih transformacija, kodnim sekvencama različitih kompresionih kodova i sl. Zapaženo je da neke klase reprezentacija značajno olakšavaju složene zadatke kao što je npr. dobijanje odgovora na pitanje “ko je na prikazanoj slici?”. Da li se ove reprezentacije mogu automatski naučiti? Da li su one invarijantne na partikularne zadatke? Ovo su samo neka od pitanja kojima se bave istraživači ove nove discipline, čiji je širi cilj pomeranje mašinskog učenja bliže ka originalnom cilju: veštačkoj inteligenciji. Uspešnost ovog pristupa pokazala se u značajnom unapređenju performansi sistema za prepoznavanje audio signala, slika i teksta.

### 12.1 DUBINA ARHITEKTURE SISTEMA ZA OBUČAVANJE

Da bi smo formalizovati pojam dubine arhitekture, moramo uvesti skup računskih elemenata. Primer takvog skupa je skup operacija koje mogu biti predstavljene logičkim kolima. Drugi primer je skup veštačkih neurona. Ulazno-izlazna funkcija nekog sistema može biti izražena kompozicijom računskih elemenata iz datog skupa. Ova kompozicija se predstavlja **grafom toka** kod koga svaki čvor odgovara jednom računskom elementu.

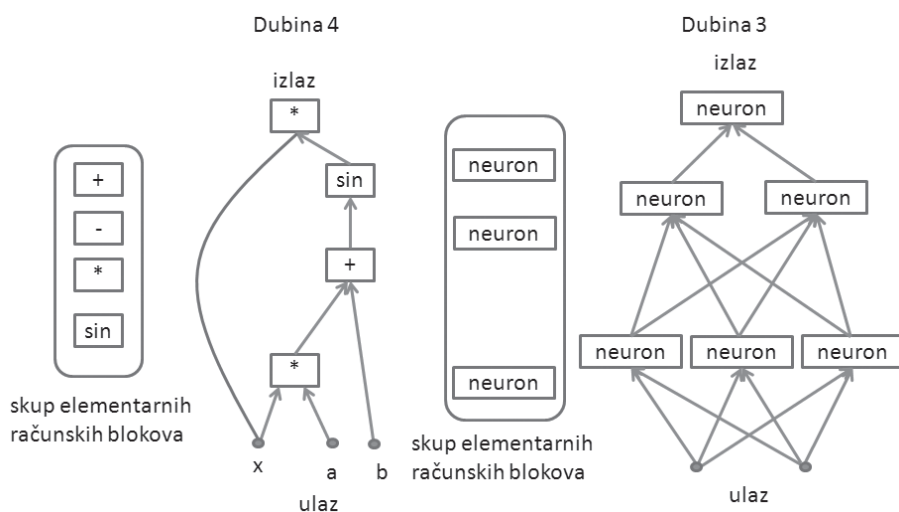
**Dubina arhitekture** jednaka je najdužem putu od ulaznog do izlaznog čvora pridruženog grafa toka.

Kada skup računskih elemenata predstavlja skup veštačkih neuron, dubina odgovara broju slojeva u neuronskoj mreži.

Ilustrirajmo koncept dubine na sledećem primeru.

#### PRIMER 12.1

Posmatraćemo funkciju  $f(x)=x \cdot \sin(a \cdot x + b)$ . Izračunavanje ove funkcije se može izraziti preko skupa jednostavnih operacija kao što su sabiranje, oduzimanje, množenje i sinus funkcija, videti Sl.12.1.



Sl.12.1 Primeri funkcija predstavljenih preko grafa toka, gde je svaki čvor uzet iz nekog skupa elemenata dozvoljenih računanja. Levo: elementi računanja su operacije sabiranja, množenja, oduzimanja i funkcija sin. Desno: elementi računanja su veštački neuroni; svaki element u skupu ima različit (w,b) parameter, odnosno sinaptičke veze i bajase neurona. Leva arhitektura ima dubinu 4, a desna 3.

#### 12.1.1 MOTIVACIJA ZA DUBOKE ARHITEKTURE

Osnovne motivacije za izučavanje i primenu dubokih arhitektura potiču od sledeća tri razloga.

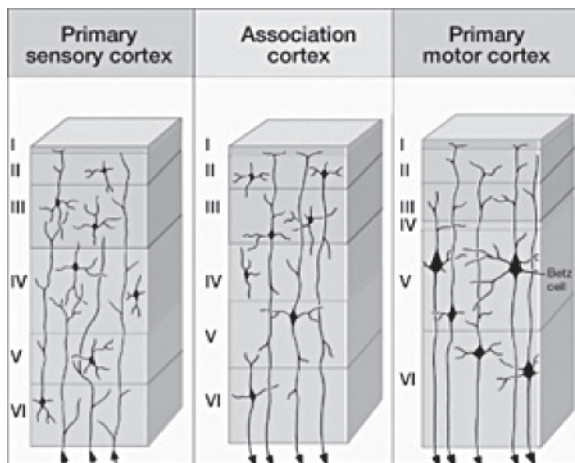
1. Nedovoljna dubina arhitekture može sprečiti dobro obučavanje
2. Naš mozak poseduje duboku arhitekturu
3. Ljudski kognitivni procesi su duboki

## Nedovoljna dubina arhitekture može sprečiti dobro obučavanje

Arhitekture koje susrećemo u tradicionalnom mašinskom učenju su po pravilu dubine 2 (logički gejtovi, sigmoidalni neuroni, Support Vector Machine -SVM) i u stanju su da dovoljno tačno reprezentuju ciljnu funkciju. Međutim, za ovako plitku arhitekturu se mora platiti odgovarajuća cena. Zahtevani broj čvorova u grafu toka, a samim tim i broj operacija i broj slobodnih parametara, može biti veoma velik. Duboke arhitekture se mogu posmatrati i kao jedan vid faktorizacije. Većina slučajno izabranih funkcija se ne mogu reprezentovati efikasno ni sa dubokom ni sa plitkom arhitekturom. Međutim mnoge funkcije koje se mogu efikasno predstaviti dubokom arhitekturom ne mogu se predstaviti pomoću plitke arhitekture. Postojanje kompaktne duboke reprezentacije indicira postojanje određene strukture u funkciji koju reprezentuju. Ukoliko nema nikakve strukture, generalizacija nije moguća. Stoga su duboke arhitekture jedan od potrebnih uslova efikasnog obučavanja sa dobrim generalizacionim svojstvima.

## Naš mozak poseduje duboku arhitekturu

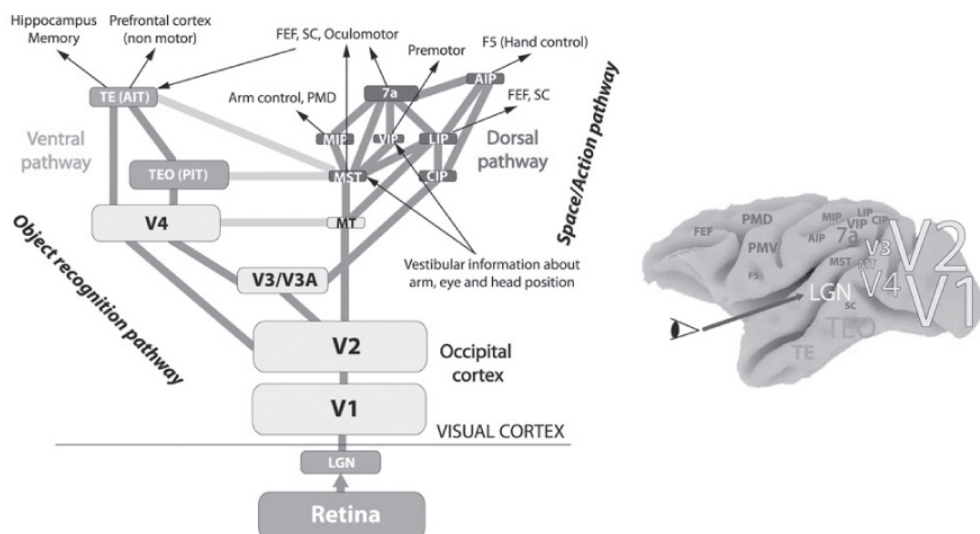
Neokorteks ne vrši eksplicitno procesiranje senzorskih signala, već ih pušta da propagiraju kroz kompleksnu hijerarhiju modula, koji u toku vremena uče reprezentaciju observacija na osnovu regularnosti koje pokazuju, Sl.12.2. Modelovanje vremenske komponente observacija igra ključnu ulogu u efektivnoj reprezentaciji informacija o okruženju u ljudskom mozgu.



Sl.12.2 Slojevita struktura korteksa.

Npr. vizuelni korteks je dobro proučen i u njemu je pronađen niz slojeva od kojih svaki sadrži svoju internu reprezentaciju ulaza. Svaki nivo ove hijerarhijske reprezentacije obeležja vizuelnog senzorskog ulaza, predstavlja ulaz za sledeći apstraktniji nivo,

Sl.12.3. Napomenimo da je reprezentacija u našem mozgu između visoko distribuirane i čisto lokalne. Oko 1% neurona u našem mozgu je simultano aktivno u svakom momentu. Ovo sugerira da se radi o poredjenoj (sparse) i vrlo efikasnoj reprezentaciji, kada se ima u vidu ukupan broj neurona.



Sl.12.3 Slojevita struktura vizuelnog korteksa.

Tok vizuelnih informacija: retina – LGN – V1 – V2 – V3/V3A – V4 – PIT – AIT.

## Ljudski kognitivni procesi su duboki

Ljudi organizuju koncepte i ideje hijerarhijski. Uobičajeno je da prvo naučimo jednostavnije koncepte, a zatim ih komponujemo u apstraktnije. Inženjerska praksa je često uspešna zbog uobičajenog pristupa rastavljanja datog problema na više nivoa apstrakcije i procesiranja.

Introspekcija lingvistički izrazivih koncepata takodje sugerira poredjenu reprezentaciju: samo mali broj reči/konceptata od ukupnog mogućeg broja u jednom jeziku je primenljiv za određeni ulaz, npr. opis date vizuelne scene.

## 12.1.2 SPEKTAKULARNI PROBOJ U OBUČAVANJU DUBOKIH ARHITEKTURA

Pre 2006. godine svi pokušaji obučavanja dubokih arhitektura su bili neuspešni. Obučavanje dubokih neuronskih mreža tipa višeslojnih perceptrona sa propagacijom signala unapred davalo je na istim problemima gore rezultate od plitkih arhitektura sa jednim ili dva sloja neurona, i to kako u pogledu greške pri obučavanju, tako i greške na test skupovima.

2006. godine pojavila su se tri rada, predvodjena Hintonovim revolucionarnim radom o treniranju dubokih mreža uverenja (Deep Belief Networks – DBN), koja su potpuno izmenila stanje ove oblasti mašinskog učenja, videti Sl.12.4:

1. Hinton, G. E., Osindero, S. and Teh, Y., *A fast learning algorithm for deep belief nets* Neural Computation 18:1527-1554, 2006
2. Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, *Greedy Layer-Wise Training of Deep Networks*, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007
3. Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun, *Efficient Learning of Sparse Representations with an Energy-Based Model*, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007



Sl.12.4 Geoffrey Hinton, Yoshua Bengio i Marc'Aurelio Ranzato prvi koautori tri ključna rada iz 2006. godine u kojima je napravljen suštinski napredak u obučavanju dubokih arhitektura.

U sva tri rada ukazano je na sledeće ključne principe:

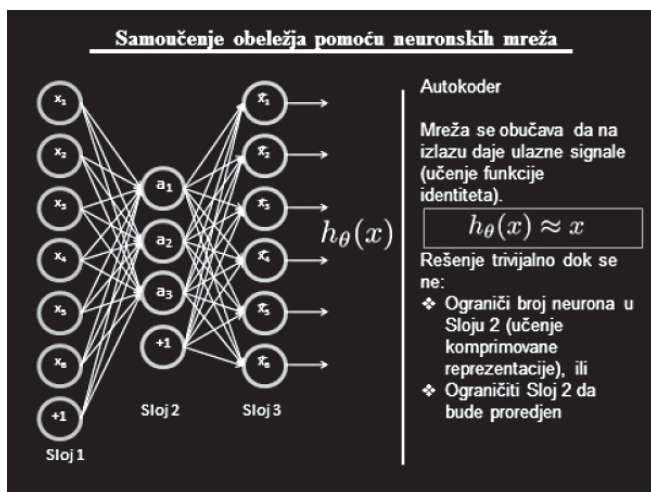
1. Samoobučavanje reprezentacije se koristi kao pretrening za svaki sloj duboke arhitekture.
2. Sekvencijalno samoobučavanje svakog sloja posebno, nakon završenog samoobučavanja prethodnog sloja. Reprezentacija naučena u svakom sloju predstavlja ulaz za naredni sloj.
3. Finalno obučavanje sa učiteljem svih slojeva u cilju finog podešavanja uz dodatak jednog ili više slojeva na vrhu arhitekture u cilju formiranja izlaznih odluka (predikcije, klasifikacije i sl.)

Predložene su dve arhitekture dubokog obučavanja

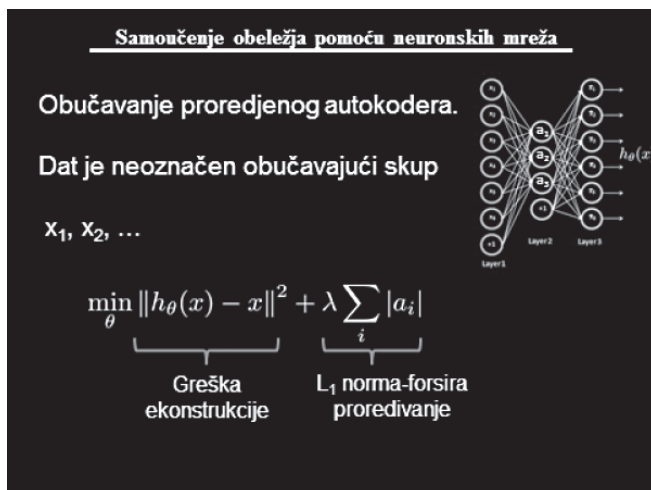
1. **Duboke mreže uverenja** (Deep Belief Networks - DBM) koje koriste ograničene Bolcmanove mašine (Restricted Boltzman Machine – RBM) u samoobučavajućem režimu u cilju učenja reprezentacije svakog sloja arhitekture
2. **Duboke neuronske mreže** koje koriste autokodere u svakom sloju. Autokodери su neuronske mreže koje vrše predikciju ulaza na osnovu restriktivne interne reprezentacije.

## 12.3 DUBOKE NEURONSKE MREŽE

Osnovni gradivni element dubokih neuronskih mreža je autokoder prikazan na Sl.12.5. Autokoder ima tri sloja: ulazni, srednji i izlazni (Sloj 1, Sloj 2 i Sloj 3). Da bi se izbeglo trivijalno rešenje direktnog preslikavanja sa ulaza na izlaz, neophodno je uvesti dodatna ograničenja prilikom obučavanja. Uobičajena su dva postupka: ograničavanje broja neurona u srednjem sloju i nametanje proredjene reprezentacije izlaznih signala srednjeg sloja, Sl.12.6.

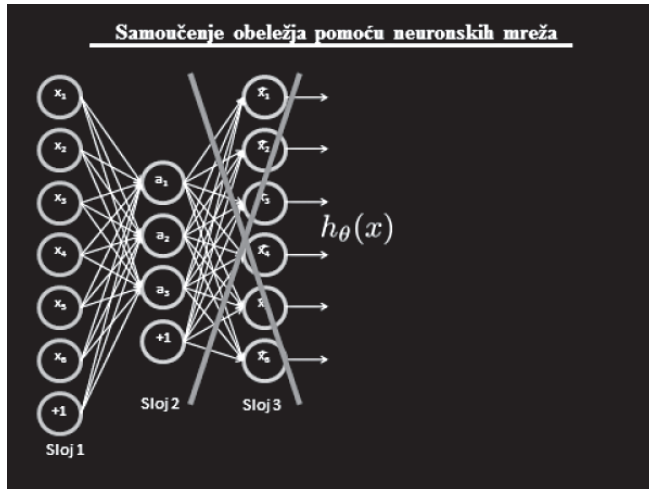


Sl.12.5 Autokoder je glavni gradivni element dubokih neuronskih mreža. Posедуje tri sloja – ulazni (Sloj 1), srednji (Sloj 2) i izlazni (Sloj 3). Cilj obučavanja je što bolja predikcija ulaza.



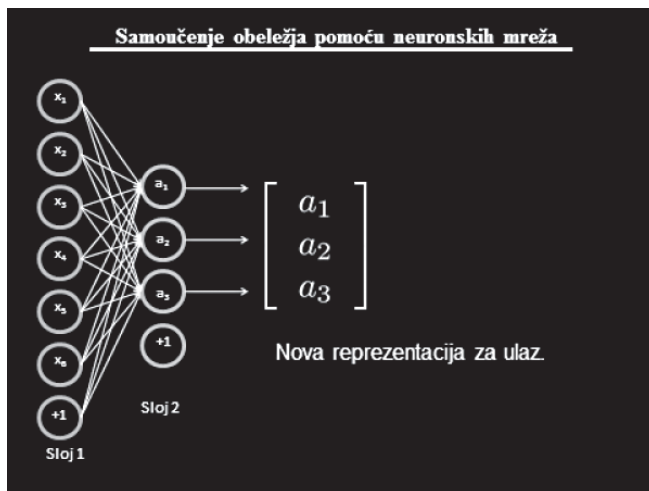
Sl.12.6 Kriterijumska funkcija samoobučavanja autokodera ima dva člana. Prvi minimizira opštu grešku rekonstrukcije ulaza, dok drugi, u formi  $L_1$  norme, forsira proredjenu reprezentaciju ulaza u srednjem sloju autokodera.

Na nizu slika od 12.7 do 12.13 prilazan je genrički postupak obučavanja neuronske mreže duboke arhitekture, sloj po sloj, sa ciljem formiranja hijerarhije reprezentacija kroz postupak samoobučavanja odgovarajućih autokodera. Algoritam započinje obučavanjem autokodera koga čine Sloj 1, 2 i 3, Sl.12.7. Primetimo da su izlazi Sloja 3, aproksimacije ulaza u Sloj 1. Nakon obučavanja, uklanja se Sloj 3, a izlazi Sloja 2 predstavljaju prvi nivo interne reprezentacije ulaza, Sl.12.8.



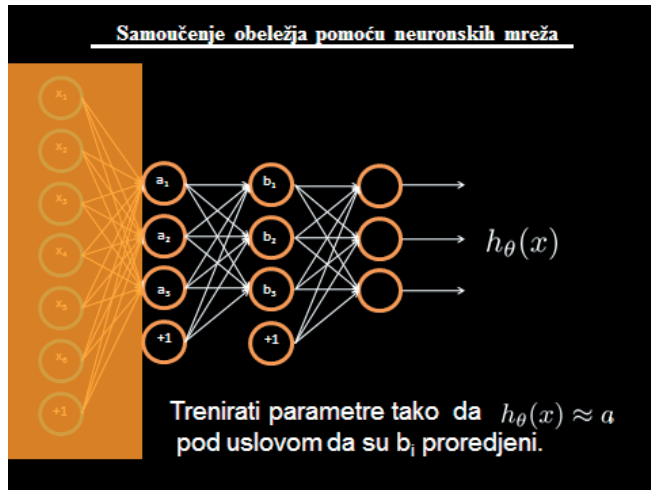
Sl.12.7 Nakon samoobučavanja autokodera koga čine Sloj 1, 2 i 3, uklanja se Sloj 3, koji predstavlja aproksimaciju ulaza u Sloj 1.

Zatim se formira novi autokoder čiji je ulaz dobijena reprezentacija iz prethodne iteracije algoritma obučavanja, Sl.12.9.



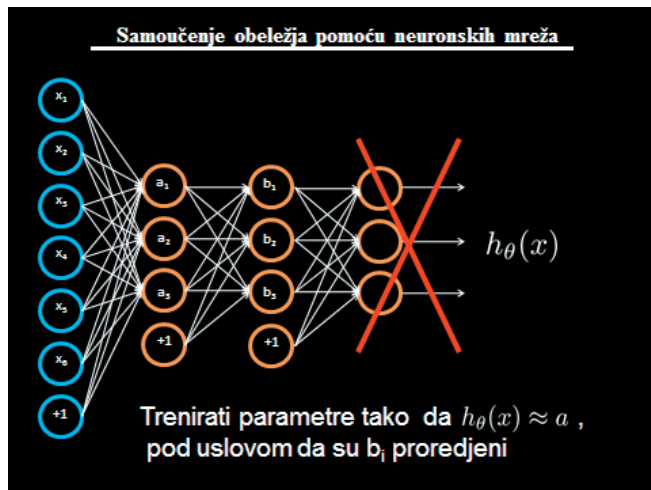
Sl.12.8 Prvi nivo interne reprezentacije  $[a_1, a_2, a_3]$ , dobijen obučavanjem autokodera, čiji je srednji sloj jednak Sloju 2.



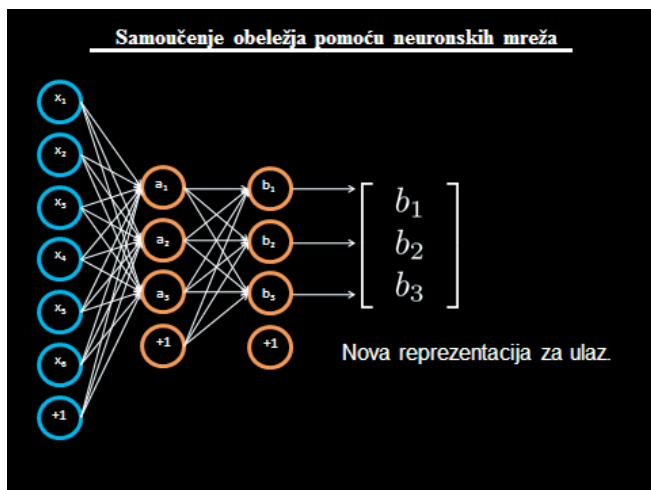


Sl.12.9 Autokoder za učenje sledećeg nivoa interne reprezentacije. Cilj obučavanja je da se na izlazu dobije što bolja procena interne reprezentacije prvog nivoa ( $[a_1 a_2 a_3]'$ ), pod uslovom da su izlazi srednjeg sloja autokodera ( $b_i$ ) poredjeni.

Nakon izvršenog obučavanja, uklanja se izlazni sloj autokodera, Sl.12.10, čime se dobija drugi nivo interne reprezentacije  $[b_1 b_2 b_3]'$ , koja je formirana na osnovu interne reprezentacije sa prethodnog hijerarhijskog nivoa (reprezentacija  $[a_1 a_2 a_3]'$ ), Sl.2.11.

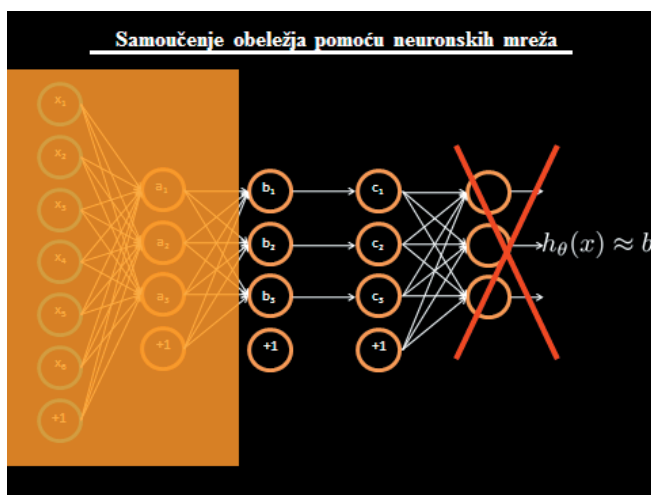


Sl.12.10 Nakon samoobučavanja autokodera koga čine Sloj 2, 3 i 4, uklanja se Sloj 4, koji predstavlja aproksimaciju ulaza u Sloj 2.

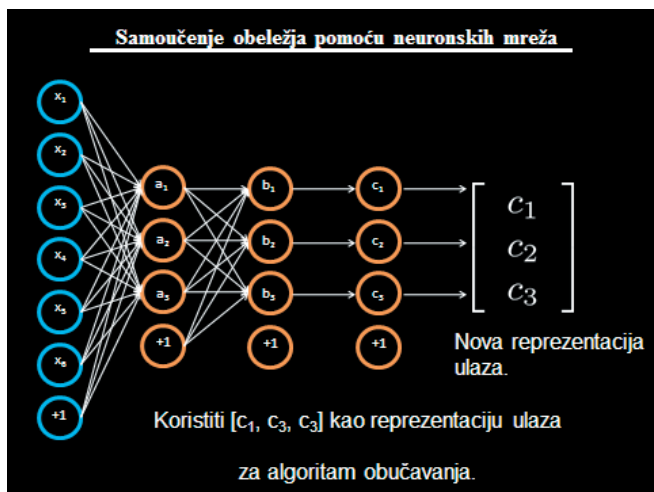


Sl.12.11 Drugi nivo interne reprezentacije  $[b_1 \ b_2 \ b_3]'$ , koja je formirana na osnovu interne reprezentacije sa prethodnog hijerarhijskog nivoa (reprezentacija  $[a_1 \ a_2 \ a_3]'$ ).

Na Sl.12.12 i Sl.12.13 prikazana je naredna iteracija algoritma pri formiranju trećeg nivoa interne reprezentacije  $[c_1 \ c_2 \ c_3]'$ , na osnovu interne reprezentacije sa drugog nivoa  $[b_1 \ b_2 \ b_3]'$ .



Sl.12.12 Nakon samoobučavanja autokodera koga čine Sloj 3, 4 i 5, uklanja se Sloj 5, koji predstavlja aproksimaciju ulaza u Sloj 3.



Sl.12.13 Treći nivo interne reprezentacije  $[c_1, c_2, c_3]'$ , koja je formirana na osnovu interne reprezentacije sa prethodnog hijerarhijskog nivoa (reprezentacija  $[b_1, b_2, b_3]'$ ).

Obučena duboka arhitektura sa Sl.12.13 je dobijena na osnovu neoznačenih primera, što predstavlja ogromnu prednost ovog postupka u današnjim uslovima obilja neoznačenih podataka na internetu. Često se ova faza obučavanja dubokih arhitektura naziva **samoobučavajući pre-trening**.

Nakon ove faze se na ovu arhitekturu dodaju još dva sloja, jedan skriveni i jedan izlazni, čije parametre treniramo u klasičnom obučavanju sa učiteljem, stim što su označeni podaci iz obučavajućeg skupa, prethodno transformisani u prostor interne reprezentacije poslednjeg sloja dobijenog u fazi pre-treninga. U primeru koji je ilustrisan prethodnim nizom slika, to bi bio prostor vektora reprezentacija  $[c_1, c_2, c_3]'$ . Mnogi obavljeni eksperimenti pokazuju da se na osnovu izuzetno malog obučavajućeg skupa mogu dobiti značajno bolji rezultati nego u klasičnim sistemima mašinskog učenja zasnovanim na obučavanju sa učiteljem.

## PRIMER 12.2

Ilustrirajmo ovaj pristup na jednom konkretnom primeru klasifikacije slika sa interneta u dve kategorije: automobili i motocikli. (primer preuzet sa <http://ufldl.stanford.edu/eccv10-tutorial/>, Part 4: Learning Feature Hierarchies and Deep Learning (by Andrew Ng). Na Sl.12.14 prikazan je opšti scenario obučavanja sa učiteljem, na Sl.12.15 scenario polunadgedanog (semisupervised) obučavanja, a na Sl. 12.16 i Sl.12.17 scenario rešavanja ovog problema primenom samoobučavajućeg pred-treninga dubokih neuronskih mreža.



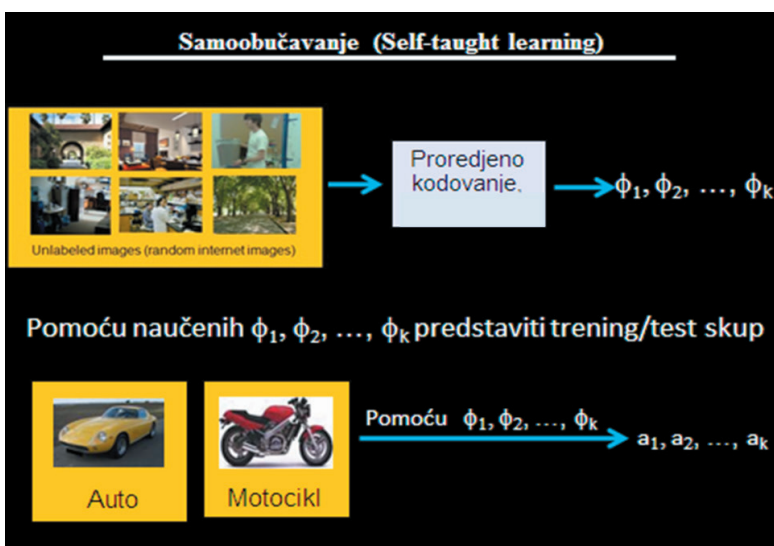
Sl.12.14 Klasična postavka problema prepoznavanja automobila i motocikla. Neophodna su dva skupa označenih primera. Što su ti skupovi veći, sistemi prepoznavanja su tačniji.



Sl.12.15 U scenariju polunadgledanog obučavanja na raspolaganju je obučavajući skup u kome su samo neoznačene slike automobila i motocikla.



Sl.12.16 Samoobučavajući sistem: na raspolaganju su slučajno izabrane neoznačene slike sa interneta.



Sl.12.17 Savremeno rešenje samoobučavajućeg sistema zasnovanog na automatskom učenju hijerarhije internih reprezentacija u čistom nenadgledanom režimu koji ne zahteva označavanje slika. Rezultat ove faze je interna reprezentacija  $[\phi_1, \phi_2, \dots, \phi_k]$ . Na osnovu nje i malog obučavajućeg skupa, uz dodavanje još dva sloja na izlazu duboke neuronske mreže, moguće je sa velikom tačnošću prepoznavati automobile i motocikle. Poseban kuriozitet predstavlja činjenica da u neoznačenom skupu koji se koristi u pre-treningu ne mora postojati ni jedna slika automobila ili motocikla.

## 12.3 FILOZOFSKI POGLED NA UČENJE INTERNIH REPREZENTACIJA DUBOKIM NEURONSKIM MREŽAMA

Učenje hijerarhije internih reprezentacija je dominantno inspirisano strukturom i načinom funkcionisanja našeg mozga, a naročito vizuelnog korteksa. Veliku pomoć u pravilnom tumačenju načina funkcionisanja našeg mozga pruža složena oprema za snimanje moždanih aktivnosti, kao i opšti prodori u razumevanju funkcionisanja neuronskih ćelija i njihovih sinaptičkih veza. Stoga je fascinantno kako su veliki filozofi iz bliže i dalje istorije imali neverovatno dobru intuiciju o načinu obrade senzorskih signala i njihovom uzdizanju do pojmovnog nivoa i viših oblika mišljenja.

Kao ilustraciju ovog fenomena predstavimo razvoj jednog filozofskog pojma tzv. **transcendentalne uobrazilje** i njegove tesne veze sa idejama koje upravo sada okupiraju istraživače iz domena dubokog učenja. Pokazaćemo na kraju da pravilno tumačenje onoga šta su značajni filozofi rekli o transcendentalnoj uobrazilji (Aristotel, Kant, Hegel, Hajdeger), može biti veoma inspirativno za usmeravanje najnovijih istraživanja u oblasti dubokog mašinskog učenja.

Navedimo prvo jedno lucidno Aristotelovo zapažanje:

“Duša nikad ne misli bez fantazije.”, Aristotel [De Anima, III, 7 i 8],

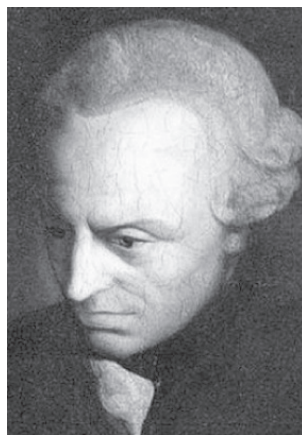
Svaki apstraktni pojam je u našem mišljenju praćen konkretnom slikom - fantastičnom predodžbom - uobraziljom. Kada mislimo na trougao na umu imamo sliku konkretnog trougla. Odakle dolazi ova fantazija? Kako se konstituiše? Koji delovi mozga su odgovorni za ovaj fenomen.

Tako je u filozofiju ušlo pitanje uobrazilje ili fantazije, koju će mnogo kasnije Kant nazvati transcendentalnom uobraziljom, Sl.12.18.

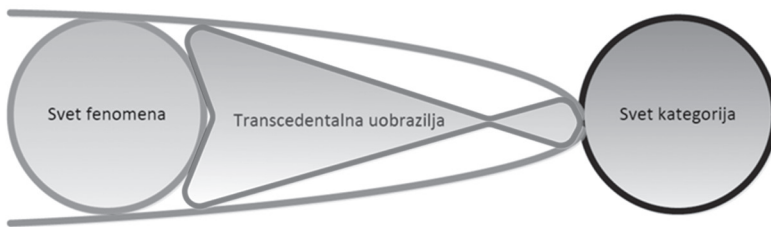
Kant postulira trostepeni proces koji dovodi do saznanja

1. Raznolikost čistog opažanja.
2. Sinteza te raznolikosti pomoću uobrazilje (transcendentalna sinteza uobrazilje).
3. Zasnovano na razumu, davanje jedinstva ovoj sintezi pomoću pojmova.

Jedna moguća predstava ovog procesa je data na Sl.12.19.



Sl.12.18 Imanuel Kant (1724 – 1804) pruski filozof, tvorac nemačkog idealizma. Njegovo interesovanje za fenomene obrade čulnih datosti i načina na koji se transformišu u više oblike saznanja dovele su ga između ostalog i do toga da rasvetli problem fantazije i uobrazilje. Budući da ona nadilazi neposrednu čulnu datost, nazvao je transcendentalnom uobraziljom.



Sl.12.19 Šta je ta uobrazilja i gde je smeštamo? Kant joj je dao vrlo značajnomesto – na direktnom putu od sveta fenomena ka svetu kategorija. Njena uloga je čak nezamenljiva, jer se samo kroz transcendentalnu sintezu uobrazilje naše mišljenje može uzdići do pojmova i kategorija.

Kant je transcendentalnoj uobrazilji dao vrlo značajno mesto – na direktnom putu od sveta fenomena ka svetu kategorija. Sinteza je jednostavno delovanje uobrazilje, slepe, mada neophodne funkcije duše bez koje mi uopšte ne bi smo imali nikakvo saznanje, ali koje smo retko kada svesni.

Dve su moguće interpretacije transcendentalne sinteze uobrazilje:

1. Sinteza je jednostavno delovanje uobrazilje, bez uplitanja razuma
2. Sinteza je direktna primena sintetičke moći razuma na niži, primitivniji pred saznažni nivo.

Drugim rečima, da li je transcendentalna uobrazilja koren subjektiviteta i obuhvatni rod iz koga niče razum, ili je razum obuhvatni rod pri čemu je uobrazilja samo neka vrsta senke koju razum baca na niže nivoe opažaja.

Nemački filozof Hajdeger, koga smatraju najvećim filozofom XX veka, bio je takodje okupiran fenomenom transcendentalne uobrazilje, mnogo više nego što je to javno priznavao ili razradjivao u svom obimnom filozofskom delu, Sl.12.20. Najnovije analize pokazuju da ono što je našao u svom istraživanju prikazanom u kulturnom delu “Biće i vreme” je bezdan radikalnog subjektiviteta najvaljen već u kantovskoj transcendentalnoj uobrazilji. Od tog bezdana je uzmakao prema misli o povesnosti bića, i ostavio nedovršenim najavljeni II i III tom Bića i vremena.

Hajdeger tvrdi da je transcendentalna sinteza uobrazilje temeljna dimenzija u korenu diskurzivnog razuma. Stoga je treba analizirati nezavisno od kategorija razuma.



Sl.12.20 Martin Heidegger (1889 – 1976) nemački filozof koga smatraju najvećim filozofom XX veka. Najnovije analize pokazuju da odustajanje od pisanja II i III dela njegove kulturne knjige “Biće i vreme” je zapravo posledica uzmicanja pred bezdanom koji stvara transcendentalna uobrazilja kod svakog pokušaja da se ozbiljnije zahvati jedna filozofska ontologija i odgovarajuća pozicija čoveka u svetu.



Kant je prema Hajdegerovoj analizi, uzmao kao pred ovim radikalnim korakom, zadržavajući transcendentalnu uobrazilju kao puko posredujuće između opažajne raznolikosti i sazajnog sintetičkog delovanja razuma. Kao što se pokazuje u najnovijim analizama, Hajdegerovo uzimicanje je bilo mnogo radikalnije od Kantovog.

Možda je najinspirativniji pogled na transcendentalnu uobrazilju imao veliki Hegel, Sl.12.21.

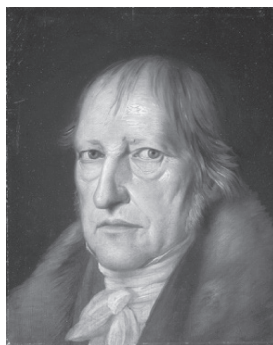
Hegel ističe suprotnu moć uobrazilje – uobrazilja kao delatnost rastvaranja. Ta negativna moć obuhvata i razum i uobrazilju. Ovo sledi iz dva ključna fragmenta o „noći sveta“, pri čemu ćemo za naše potrebe navesti samo prvi:

„Ljudsko biće je ta noć, to prazno ništa, koje sadrži sve u svojoj jednostavnosti – beskrajno bogatstvo mnogih predstava, slika, od kojih ni jedna ne pripada njemu ili koje nisu prisutne. U toj noći niče krvava glava, tamo još jedna bela, sablasna prikaza, odjednom ovde pred njim, i jednako tako isčezava. Ta se noć vidi kada se pogleda čoveku u oči – u noć koja postaje strašna..“

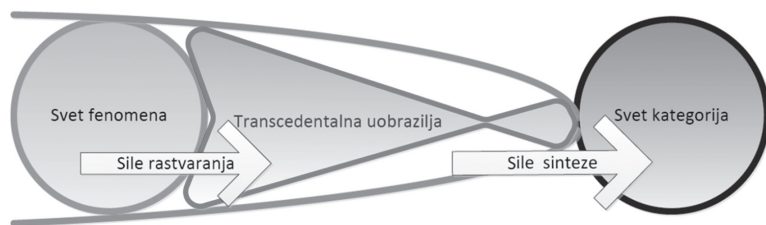
(G.W.F.Hegel, „Jenaer Realphilosophie“ u Fruhe politische Systeme)

Dakle, dat je opis uobrazilje u njenom negativnom, razarajućem, rastvarajućem obliku. To je moć koja raspršuje kontinuiranu stvarnost u konfuzno mnoštvo parcijalnih predmeta, koji u realnosti postoje samo kao delovi većih entiteta.

Transcendentalna uobrazilja se ovde pokazuje kao sposobnost našeg uma da rastavi ono što neposredno opažanje sastavlja, da apstrahuje ne na nivou pojma, nego određeno obeležje od drugih obeležja, Sl.12.22.



Sl.12.21 Georg Wilhelm Friedrich Hegel (1770 – 1831) nemački filozof, glavna ličnost nemačkog idealizma. Njegova zapažanja o transcendentalnoj uobrazilji sa frapantnom podudarnošću opisuju postupak samoobučavajućeg pre-treninga dubokih neuronskih arhitektura, ako ih stavimo u kontekst funkcionisanja vizuelnog korteksa.



Sl.12.22 Sile rastvaranja u Hegelovom konceptu transcendentalne uobrazilje

Sada smo dovoljno upućeni da napravimo ključnu analogiju:

**Transcendentalna uobrazilja odgovara svetu internih reprezentacija unutar sistema dubokog učenja.**



Kada se identifikuje neka plodna analogija, potrebno je pustiti da veliki umovi koji nam kroz te analogije šalju korisne poruke, progovore o našem problemu sa kojim je analogija uspostavljenja. Evo kako bi izgledao razgovor sa Hajdegerom i Kantom o dubokom mašinskom učenju.

Hajdeger u svom vremenu (original): Transcedentalna sinteza uobrazilje je temeljna dimenzija u korenu diskurzivnog razuma. Stoga je treba analizirati nezavisno od kategorija razuma.

Hajdeger danas: Interna reperezentacija u deep modelima je osnova svakog ljudskog koncepta. Stoga je nužno nastala kao rezultat samoobučavanja (nezavisno od kategorija razuma).

Kant u svom vremenu (original):

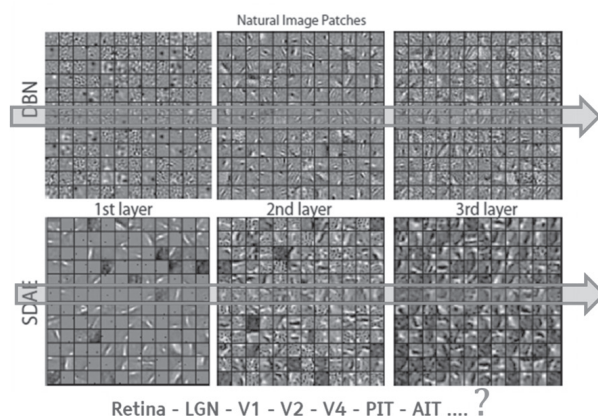
1. Sinteza je jednostavno delovanje uobrazilje, bez uplitanja razuma.
2. Sinteza je direktna primena sintetičke moći razuma na niži, primitivniji predstajni nivo.

Kant danas:

1. Interna reprezentacija u deep modelima je čisto samoobučavanje bez uplitanja supervizora.
2. Interna reprezentacija u deep modelima je delom formirana složenim uticajem povratne sprege sa izlaznog konceptualnog nivoa na niže hijerarhijske nivoe, prethodno formirane kroz čisto samoobučavanje.

Završimo ovu analizu ilustracijom prave Hegelovske “noći sveta” - simbola rastvorene stvarnosti, koja je dobijena 200 godina kasnije kao hijerarhija internih reprezentacija duboke neuronske mreže obučene na slikama ljudskih lica, Sl.12.23.

## Ka Hegelovskoj “noći sveta”



Sl.12.23 Interne reprezentacije na izlazu 1., 2. i 3. sloja duboke neuronske mreže obučene na primerima slika lica. Parcijalne slike u ovim internim reprezentacijama zaista odgovaraju Hegelovom opisu “noći sveta”, citiramo “... bogatstvo mnogih predstava, slika, od kojih ni jedna ne pripada njemu ili koje nisu prisutne...”.

## Rezime 12. poglavlja

- ♦ Svaka šema računanja, a samim tim i šema računanja hipoteza u mašinskom učenju, se može predstaviti grafom toka u kome svaki čvor predstavlja elementarne računске blokove.
- ♦ Najduži put od ulaza do izlaza grafa toka naziva se dubina šeme.
- ♦ Oblast dubokog učenja je nova podoblast mašinskog učenja koja se odnosi na računске šeme hipoteza velike dubine.
- ♦ Pokazuje se da šeme mašinskog učenja velike dubine imaju veću šansu za dobrom generalizacijom.
- ♦ Algoritmi obučavanja dubokih arhitektura neuronskih mreža se obavljaju u dve faze: samoobučavanje internih reprezentacija, sloj po sloj, uz pomoć autokodera, a zatim dodavanje završnih slojeva u cilju realizacije završne klasifikacije.
- ♦ Algoritmi obučavanja dubokih neuronskih mreža se odlično uklapaju u današnji scenario masovnog dostupa podataka sa interneta, od kojih je samo manji broj označen i pogodan za tradicionalno obučavanje sa učiteljem.
- ♦ Na kraju poglavlja data je analiza jednog filozofskog pojma – transcendentne uobrazilje. Kroz govor najvećih mislilaca prošlih epoha Aristotela, Kanta, Hegela i Hajdegera o tom pojmu, dobijamo inspirativne poruke o pravoj prirodi dubokog učenja i mogućim pravcima njegovog budućeg razvoja.

## Pitanja i zadaci

1. Zašto su duboke šeme bolje za generalizaciju?
2. Šta je smisao proredjenog kodovanja?
3. Zašto je hijerarhija internih reprezentacija značajna za probleme VI?
4. Šta su to autokoderi?
5. Kako bi ste objasnili da su plitke neuronske mreže lošije od dubokih u tipičnim problemima mašinskog učenja?
6. Zašto je algoritam propagacije greške unazad daje loše rezultate za duboke arhitekture neuronskih mreža?
7. Skicirajte jedan projekat sa dubokim neuronskim mrežama koji bi imao komercijalnog značaja, kao poželjan servis na web-u.



# Literatura

- Jan Romportl, Eva Zackova, Jozef Kelemen (Editors), *Beyond Artificial Intelligence The Disappearing Human-Machine Divide*, Springer, 2015.
- Shai Shalev-Shwartz, Shai Ben-David, *Understanding Machine Learning - From Theory to Algorithms*, Cambridge University Press, 2014
- Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella (Eds.) *Machine Learning for Computer Vision*, Springer-Verlag Berlin Heidelberg 2013.
- Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Je Dean, Andrew Y. Ng, *Building High-level Features Using Large Scale Unsupervised Learning*, arXiv:1112.6209v5 [cs.LG] 12 Jul 2012.
- Nils J. Nilsson, *The Quest for Artificial Intelligence - a history of ideas and achievements*, Cambridge University Press, 2010.
- Stuart J. Russell, Peter Norvig, *Artificial Intelligence A Modern Approach*, 3rd ed, Prentice Hall, 2010
- Elain Rich, Kevin Knight, Shivashankar B. Nair, *Artificial Intelligence*, 3rd ed, McGraw Hill, 2009
- Nils J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann Pub.Inc., 1980
- Michael R. Genesereth, Nils J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Pub.Inc., 1987.

- Nils J. Nilsson, *Artificial Intelligence – A New Synthesis*, Morgan Kaufmann Pub.Inc., 1998.
- Vincent C. Müller (Ed.), *Philosophy and Theory of Artificial Intelligence*, Springer, 2013.
- Judea Pearl, *Heuristics – Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley Pub.Co., 1984.
- George F Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th ed., Pearson Education, Inc., 2009.
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). *Greedy layer-wise training of deep networks*. In NIPS'2006.
- Hinton, G., Deng, L., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012), *Deep neural networks for acoustic modeling in speech recognition*, IEEE Signal Processing Magazine , 29(6), 82–97.
- Hinton, G. E., Osindero, S., and Teh, Y. (2006), *A fast learning algorithm for deep belief nets*. *Neural Computation*, 18, 1527–1554.
- Li Deng, Dong Yu, *Deep Learning Methods and Applications, Foundations and Trends in Signal Processing* Vol. 7, Nos. 3–4 (2013) 197–387.
- Tom.M.Mitchell, *Machine Learning*, McGraw Hill, 1997.
- Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.
- Ethem Alpaydın, *Introduction to Machine Learning*, 3rd ed. MIT Press, 2014
- Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford University Press. 1995.
- Duda, R. O., P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
- Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2011.
- Webb, A., and K. D. Copsey, *Statistical Pattern Recognition*, 3rd ed. New York: Wiley. 2011.
- G. Cybenkot, *Approximation by Superpositions of a Sigmoidal Function*, Math. Control Signals Systems (1989) 2:303-314
- Armand E. Prieditis, *Machine Discovery of Effective Admissible Heuristics*, Machine Learning, 12, 117-141 (1993)
- Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun, *Efficient Learning of Sparse Representations with an Energy-Based Model*, in J. Platt et al. (Eds), *Advances in Neural Information Processing Systems (NIPS 2006)*, MIT Press, 2007

CIP - Каталогизација у публикацији  
Народна библиотека Србије, Београд

004.8(075.8)

МИЛОСАВЉЕВИЋ, Милан, 1952-

Veštačka inteligencija / Milan Milosavljević. - 3. izd. - Beograd : Univerzitet Singidunum, 2022 (Beograd : Birograf). - VI, 232 str. : ilustr. ; 24 cm

Tiraž 450. - Bibliografija: str. 231-232.

ISBN 978-86-7912-590-3

а) Вештачка интелигенција

COBISS.SR-ID 57498121

© 2022.

Sva prava zadržana. Nijedan deo ove publikacije ne može biti reprodukovan u bilo kom vidu i putem bilo kog medija, u delovima ili celini bez prethodne pismene saglasnosti izdavača.



Milan Milosavljević

# VEŠTAČKA INTELIGENCIJA

Veštačka inteligencija (VI) je konstituisana kao naučna oblast u neposrednoj vezi sa uspostavljanjem računarstva kao naučno – tehničke discipline. Ozbiljno je počela da se razvija ubrzo posle Drugog svetskog rata, a samo ime je dobila tek 1956. godine. VI trenutno obuhvata mnoštvo podoblasti, od vrlo opštih, kao što su učenje i percepcija, do uskih zadataka kao što je igranje šaha, dokazivanje matematičkih teorema, pisanja poezije, medicinska dijagnostika, automatsko prevodjenje, prepoznavanje govora, robotika i sl.

VI sistematizuje i automatizuje intelektualne zadatke i stoga je potencijalno značajna za bilo koju sferu ljudske intelektualne aktivnosti. U tom kontekstu VI je univerzalna oblast, koja sasvim opravdano predstavlja deo kurikuluma gotovo svih obrazovnih profila računarstva i informatike danas.