
**VISOKA POSLOVNA ŠKOLA**
**STRUKOVNIH STUDIJA**
**ČAČAK**

**SEMINARSKI RAD**

# Ekspertni sistemi

Mentor:_____
Profesor: _____

Student: _____
Br.Indeksa: _____

# Sadržaj

# 1 Poglavlje 1 - Ekspertni sistemi, uvod

## 1.1 Inteligentne mašine

Odavno postoji želja („sveti gral") da se napravi mašina koja bi bila sposobna da obavlja posao koji zahteva intelektualne sposobnosti čoveka, npr. da igra šah (Charles Babbage, 1834. konstruiše mehaničku „analitičku mašinu" koja računa i štampa neke matematičke proračune, imao je nameru da napravi i mašinu za igranje šaha). Tek napretkom informatičke tehnologije od sredine 20. veka taj san postaje ostvariv. Nekoliko istraživača na Dartmut koledžu 1956. g. (Dartmouth College, USA) učestvuje u konferenciji koji organizuje McCarthy na temu VI (koji je prvi predložio upravo taj naziv za tu oblast, a poznat je i kao otac LISP-a). Pored pregleda postojećih dostignuća u oblasti automatskih dokazivača teorema i programskih jezika, raspravlja se i o mogućnosti razvoja računara koji bi bio u stanju da simulira ljudsko razmišljanje. Ovo okupljanje istraživača označava rođenje veštačke inteligencije kao oblasti računarstva.

## 1.2 Definicija i kratka istorija VI

**Definicija 1.1** *(jedna od definicija)*
*Veštačka inteligencija (VI) je oblast računarstva čiji je cilj rezonovanje na računaru na način koji je sličan ljudskom.*

Veštačka inteligencija beleži prve uspehe akademske prirode kao što su prvi program za igranje šaha (Shannonn, 1955) ili dama (Samuel, 1963), automatsko dokazivanje teorema („Logic Theorist", Simon i Newell, 1972), kao i ambiciozan pokušaj ostvarivanja opšteg sistema za rešavanje problema GPS (General Problem Solver - Newell, 1960) baziran na traženju razlika i operatora između ciljnog i trenutnog stanja u bazi činjenica i operatora koji se pokazao ipak slabim za složenije probleme. VI je nakon početnih uspeha i popularnosti došla u krizu početkom sedamdesetih godina 20. veka (Lajthilov preterano kritičan izveštaj 1971.) kada je shvaćeno da nisu dovoljni algoritmi pretraživanja i simbolički programski jezici sa simboličkom reprezentacijom (baze) znanja da bi se rešili kompleksniji problemi (čuveni primer prevođenja sa jednog prirodnog jezika na drugi i nazad, tj. da za automasko prevođenje nije dovoljna samo sintaksna analiza i rečnik - Dreyfys, 1972. i Lighthill 1973). Prvi uspešni ekspertni sistemi kao što je to bio DENDRAL označili su izlazak iz te krize.

## 1.3 Ekspertni sistemi - pojam

Sistemi bazirani na znanju (knowledge-based systems) ili ekspertni sistemi stavljaju naglasak na znanje (bazu znanja) pre nego na način pretraživanja i zaključivanja. Uopštene metode zaključivanja i sistemi čija oblast nije dovoljno specificirana se pokazuju nedovoljno sposobnim za mnoge realne probleme (i njihovo rešavanje u realnom vremenu). Ekspert je osoba koja izuzetno dobro poznaje neku oblast ljudskog znanja kao i načine rešavanja problema vezanih za tu oblast.

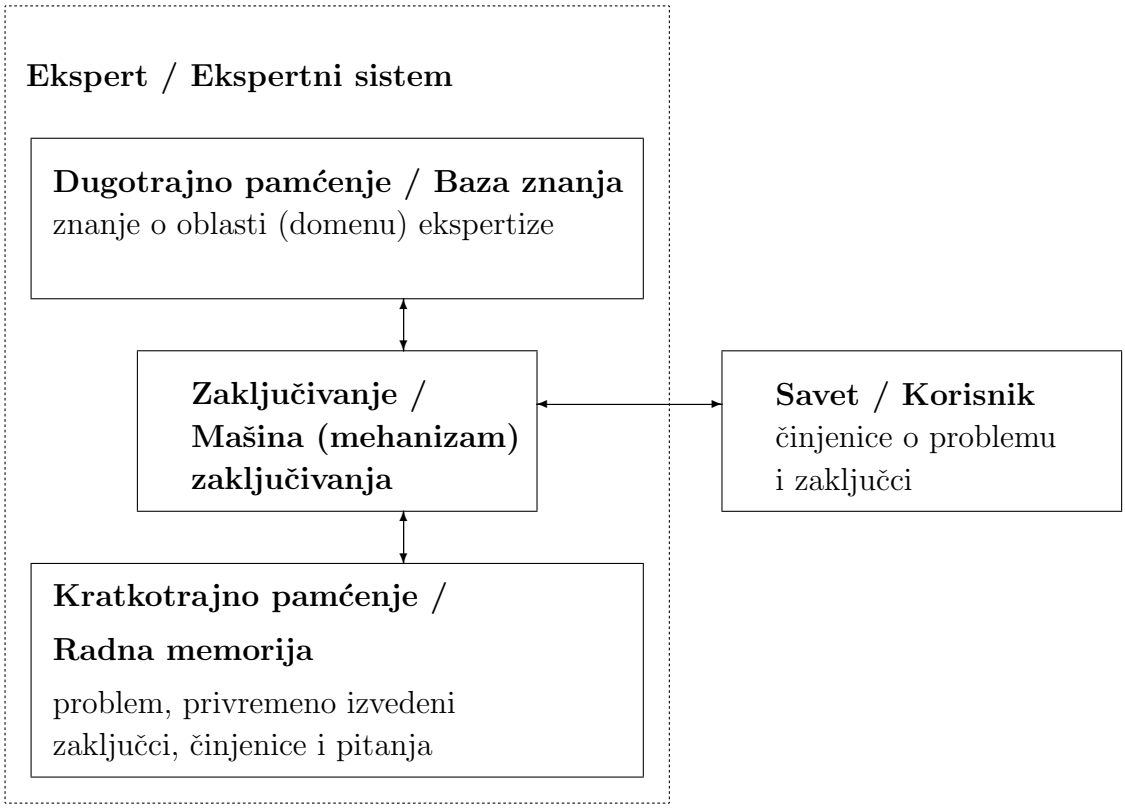
**Definicija 1.2** *Ekspertni sistem je program koji je projektovan da modelira sposobnosti rešavanja problema ljudskog eksperta u nekoj oblasti.*

Dve stvari se pre svega modeliraju u ekspertnom sistemu (ES): znanje eksperta i njegovo zaključivanje. Zato se ES sastoji iz baze znanja i mašine zaključivanja (inference engine). Znanje koje nam ekspert pruža može se predstavljati činjenicama, pravilima, konceptima ili relacijama. Način i problem njegovog predstavljanja jeste reprezentovanje znanja, dok način i problem zaključivanja na osnovu baze znanja i zadatih upita je pitanje tehnika zaključivanja.

Ekspert je često veoma neophodna osoba u mnogim organizacijama, i postavlja se onda pitanje zašto ga zamenjivati mašinom ? Pre svega, donekle slično zameni nekih ljudi i njihovih poslova mašinom tokom industrijske revolucije, ekspertni sistem može biti koristan kao pomoć kada čovek nije raspoloživ, ili prosto kao alat koji olakšava posao i omogućava automatizaciju nekih postupaka, radi sa većim stepenom formalizma koji isključuje mogućnost greške ili slabosti zbog ljudskog faktora i sl. Postoji i jedna dodatna prednost - znanjem kao opštim dobrom ili kapitalom neke organizacije se lakše upravlja, lakše se prenosi i primenjuje jer ne zavisi u toj meri od pojedinca-eksperta ako se koristi ekspertni sistem. Gotovo da nema oblasti ljudskih delatnosti gde nije napravljen neki ES i uspešno primenjivan i njihov broj geometrijski raste.

# 2 Poglavlje 2 - Osnovne osobine ES

## 2.1 Struktura i osobine




Na dijagramu iznad je prikazan odnos i poređenje modela rada jednog ljudskog eksperta i strukture ekspertnog sistema. Osnovna struktura ekspertnog sistema se tako sastoji iz osnovnih komponenti: baze znanja, radne memorije (koja se menja tokom sesije u radu sa ES dobijenim novim činjenicama i zaključcima) i mašine zaključivanja (procesor koji povezuje činjenice iz baze znanja sa dobijenim zaključcima i činjenicama iz radne memorije i izvlači nove zaključke o problemu). Pored ovih osnovnih komponenti postoje dodatno i neke osnovne osobine ekspertnih sistema:

- **objašnjenja (kako, zašto)** - u svakom trenutku u toku sesije sa ES može se dobiti objašnjenje kako je došao do nekog zaključka ili pitanja (ugrađeno u sve komponente)

- **interfejs (prema korisniku)** - često se traži komunikacija prirodnim jezikom kroz interaktivni dijalog s korisnikom (mada mogu biti zastupljeni i multimedijalni i drugi sadržaji i vidovi komunikacije što zavisi od prirode same oblasti i namene ES)
- **razdvojeno znanje od kontrole** - veoma važna karakteristika ES nasuprot klasičnim proceduralnim programima gde se upravljačka kontrola u programu meša sa raspoloživim znanjem (kontrola tj. zaključivanje kod ES se može menjati nezavisno od baze znanja)
- **poseduje ekspertsku bazu** - baza znanja koja potiče pre svega od ljudksih eksperata, gde se pod ekspertizom podrazumeva sposobnost efikasnog rešavanja problema u nekoj oblasti
- **naglasak na ekspertizi** - svako širenje oblasti znanja obavezno nosi rizik da ES postane neefikasan i nepotrebno složen (bilo je pokušaja ograničenog uspeha, GPS ili Ham 1984.) - uvek je lak'še rešavati problem kroz potprobleme odnosno probleme u podoblastima
- **koristi simbole (činjenice, koncepte, pravila)** - predstavljanje znanja u različitim strukturama (od deklarativnih oblika kao što je prirodni jezik do potpuno proceduralnih) čini ES obrađivačima znanja nasuprot konvencionalnim proceduralnim programima kao procesorima podataka
- **zaključianje heuristikama** - da bi mogao da efikasno zaključuje mora da vodi računa o najboljem načinu rešavanje problema ne samo kroz algoritme pretraživanja s heuristikama, nekakvim poznatim prečama
- **često se koristi neegzaktno zaključivanje** - zaključivanje sa verovatnoćom, mogućnostima ili uverenjima
- **ograničenje na probleme koji su rešivi ekspertnim sistemom** - ne treba očekivati nemoguće ...
- **borba sa kompleksnošću** - rešavanje veoma jednostavninh problema ES se još i može opravdati, ali ako je problem previše kompleksan nemože se opravdati predugačko vreme potrebno za njegovo rešavanje
- **može napraviti grešku** - kao i pravi (ljudski) ekspert

## 2.2 Programiranje naspram inženjerstva znanja

**Konvencionalno programiranje** ima dobro poznate i proučene metode razvoja i odvija se sekvencijalno (iterativno), grubo rečeno u tri faze redom (životni ciklus): projektovanje, kodiranje i provera / otklanjanje grešaka (debug). Konvencionalni program je završen tek nakon prolaska kroz sve tri faze do poslednje. **Inženjerstvo znanja** u odnosu na konvencionalno programiranje nema još uvek tako dobro ustanovljene metodologije razvoja. Ovde nije naglasak na obradi podataka (algoritmom, programom) već na oblikovanju (baze) znanja - deklarativno-proceduralna kontroverza: kontrolna struktura ES kod kojih je naglasak na deklarativnom znanju je potpuno odvojena od podataka za razliku od proceduralnih programa, paradigma deklarativnih jezika naspram proceduralnih, i sl.. Grubo rečeno, faze razvoja su:

**Faza procene:** Tokom ove faze se identifikuju važni problemi, izvodivost njihovog rešenja u odnosu na ciljeve, i drugi važni činioci projekta kao što su ciljevi, oblast i potrebni resursi - podaci i ljudi (eksperti). Tako se formiraju osnovni zahtevi projekta.

**Faza prikupljanja znanja:** Znanje se obično preuzima od eksperta, organizuje i proučava kako radi boljeg uvida u problem, tako i zbog potreba daljeg razvoja. Može biti usko grlo projekta i zato se smatra važnom fazom.

**Faza projektovanja:** Definišu se ukupna struktura i organizacija znanja sistema, metodi obrade znanja - bira se (npr.) alat kojim se znanje predstavlja i obrađuje (način zaključivanja u upravljanja znanjem) tako da bude što sličniji ljudskom ekspertu.

**Faza testiranja:** Ova faza se često odvija zapravo tokom svih prethodnih faza, i nakon nje se po potrebi iterativno vraća na neku od prethodnih.

**Faza dokumentacije:** Specifičnost u odnosu na konvencionalnu dokumentaciju je rečnik znanja - daje inženjeru uvid u organizovano predstavljeno znanje i procedure rešavanja sistema.

**Faza održavanja:** ES je po pravilu sistem koji živi jer se baza znanja menja vremenom.

## 2.3 Ljudi uključeni u ES razvoj

Projekat ES obavezno uključuje:

- Domenskog eksperta (Domain Expert)
  - poseduje ekspertsko znanje, ima sposobnosti za efikasno rešavanje problema
  - može da prenese (objasni) znanje, komunikacione sposobnosti
  - raspoloživost - ima vremena koje može da posveti razvoju ES
  - nije neprijateljski raspoložen (prema razvoju ES)
- Inženjera znanja
  - poznaje inženjerstvo znanja
  - ima dobre komunikacione sposobnosti
  - može da problem pretvori u softver
  - ima programerske sposobnosti za ES (jedan od glavnih ciljeva ove knjige)
- Krajnjeg korisnika
  - u stanju je da pomogne u specificiranju korisničkog interfejsa
  - može da pomogne u sakupljanju znanja
  - može da pomogne u razvoju sistema

Ekspert u odnosu na ne-eksperta poseduje znanje (to je njegova ,,apsolutna vrednost"). Svaki od ovih učesnika poseduje određene praktične osobine i znanja potrebne da se projekat uspešno završi.

# 3 Predstavljanje znanja (knowledge representation)

U ovom poglavlju se pravi pregled nekoliko najpopularnijih metoda za predstavljanje znanja u simboličkom obliku. Pre svega **znanje** se definiše kao *razumevanje predmetne oblasti* analogno znanju ljudskog eksperta, a **domen** kao deo oblasti znanja koji je od značaja za ES je *dobro usredsređena oblast znanja* gde se pod tim podrazumeva oblast znanja vezana za jednu specifičnu temu a ne više njih (npr. tema ,,zarazne bolesti krvi"). Tada je domen znanja ono čime se ES bavi, i potrebno je nekako njegove elemente kodirati i smestiti u ES - **predstavljanje (reprezentovanje) znanja** je *metod kodiranja znanja u bazi znanja nekog ES.*

## 3.1 Tipovi znanja

Različiti tipovi predstavljanja znanja prema različitim vrstama problema naglašava jednu vrstu informacija o problemu a drugu ignoriše i ne postoji idealan tip predstavljanja znanja koji za sve primene.

Ljudsko znanje se sastoji iz pojedinih činjenica kao osnovnih elemenata. U mnogim formalnim sistemima se pod činjenicom kao oblikom deklarativnog znanja obično podrazumeva iskaz (bilo kao logički iskaz ili kao iskaz nekog drugog formalnog jezika - u većini slučajeva se pokazuje da je izražajnost ista tj. da postoje rečenice u oba jezika ekvivalentne semantike - problem u tom slučaju može da bude npr. proceduralni deo okvira) - rečenica koja je tačna ili nije tačna. Činjenice mogu biti reprezentovane na razlčite načine - mogu biti zadate i tabelama ili grafovima, itd. Neki ES koriste i prirodni jezik i druge oblike informacija (multimedijalnog karaktera na primer), ali to su obično elementi interfejsa prema korisniku koji se prevode u internu reprezentaciju i obratno.

Sledi kratak pregled tipova znanja prema suštinskom značaju i upotrebi:

| | |
|---|---|
| **proceduralno znanje** - *kako* se nešto rešava | pravila<br>strategije<br>agende<br>procedure |
| **deklarativno znanje** - činjenice, *šta* se zna | koncepti<br>objekti<br>činjenice |
| **meta-znjanje** - znanje kako upotrebiti i upravljati znanjem | znanje o drugim tipovima znanja |
| **heurstičko znanje** - ,,preko palca", *plitko* empirijsko znanje | ,,pravilo preko palca" |
| **strukturno znanje** - opisuje strukture znanja, mentalni model eksperta | skupovi pravila<br>koncepti i relacije |

Pet osnovnih tehnika za predstavljanje znanja:

- **trojke objekat-atribut-vrednost (O-A-V)** - atribut može imati jednu ili više vrednosti, iskaz može imati vrednost i sa verovatnoćom (zadatim koeficijentom), fuzzy pravila

- **pravila** - proces zaključivanja u ES je određen je pravilima (po uzoru na modus ponens) sa levom i desnom stranom (uslov - zaključak), gde desna strana može osim nove činjenice značiti i izvršenje neke procedure. Meta pravila su posebna pravila kojima se upravlja domenskim znanjem - o upotrebi pravila. Pravila mogu biti šeme pravila (promenljiva), grupisana u strukture koje odgovaraju podsistemima koji komuniciraju preko deljenih resursa (table)

- **semantičke mreže** - graf kao struktura korisna i kao prikaz znanja na način blizak ljudima, i kao struktura koja nudi neke korisne osobine pretraživanja i formalno predstavljanja znanja (npr. šetanjem od

zadatog čvora se može tražiti odgovor na upit uz osobine nasleđivanja (luk "jeste") i tranzitivnosti, organizovanje izuzetaka od nasleđenih osobina)

- **okviri (frames)** - šema kao proširenje semantičkih mreža (Barlett, 1932) sadrži i deklarativno i proceduralno tipično znanje o nekom objektu ili konceptu. Struktura podataka kojom se ovo opisuje je okvir (frame, Minsky, 1975). Objekat tj. okvir ima slotove koji predstavljaju osobine sa vrednostima (O-A-V), mogu imati pretpostavljene (default) vrednosti, mogu biti statičke ili dinamičke, mogu biti niska, broj, Boolean ili objekat - okvir. Ovo poslednje ukazuje na objektnu strukturu okvira (zapravo klasa) sa osobinom nasleđivanja i instanciranja, a metodima se opisuje proceduralno znanje. Faceti su metodi kojima se daju ili menjaju vrednosti, a koriste se i ako finije ograničenje mogućih vrednosti pored tipa, ili utiču na izbor i okidanje pravila prema uslovu.

- **logika** - iskazna logika i PR1 - u mnogim sistemima je potrebno makar imati implicitno zadat model znanja u obliku formula i teorija PR1, a u nekima se koristi upravo kao reprezentovanje znanja u nekom vidu (PROLOG, Hornove šeme, rezolucija) i kao način zaključivanja.

# 4 Tehnike zaključivanja

Postavlja se najpre granica između procesa ljudskog razmišljanja i modela ljudskog razmišljanja kojeg srećemo u VI tj. ES:

**Definicija 4.1** *Razmišljanje (reasoning) je proces rada sa znanjem, činjenicama i strategijama rešavanja problema (pravilima) radi dolaženja do zaključaka o istim.*

- **Deduktivno zaključivanje** - klasičan oblik razmišljanja i zaključivanja (npr. modus ponens) u kome se na osnovu postojećeg skupa činjenica (aksioma) izvode novi zaključci ili međuzaključci.

- **Induktivno zaključivanje** - Ljudi često zaključuju uopštavanjem pojedinih slučajeva (generalizacijom) u sve slučajeve određenog tipa. Suština takvog induktivnog razmišljanja je taj prelazak od pojedinog ka svemu (Firebaugh, 1988. - ako važi $P(x)$ na nekom podskupu vrednosti X onda važi $(\forall x)P(x)$).

- **Abduktivno zaključivanje** - abduktivno zaključivanje je oblik deduktivnog zaključivanja u kome se dozvoljavaju i mogući zaključci (generalizacija nad predikatom ili podizrazom a ne samo nad promenljivom, npr. ako je $B$ i $A \Rightarrow B$ onda je *moguće da je tačno* i $A$).

- **Analogično zaključivanje** - okvirom se mogu uhvatiti stereotipne informacije, i ako se kaže da je neki objekat predstavljen okvirom sličan nekom drugom objektu predstavljenim okvirom s nekim uhvaćenim razlikama onda je to analogično zaključivanje o novom objektu koji preuzima osobine starog objekta uz neke razlike.

- **Zdravorazumsko zaključivanje** - ili zaključivanje uz upotrebu heuristika, ,,plitkog" ali praktičnog znanja o nekim stvarima i najboljim postupcima zaključivanja o njima (kao kod algoritma pretraživanja s izborom najbooljeg, best-first, kao i sa upotrebama heuristika - korisnih prečica u pretrazi), što je čest slučaj kod ljudi eksperata.

- **Nemonotono zaključivanje** - često se razmišlja o problemu tako da mnoge činjenice u bazi i njihove posledice imaju konstantnu istinitosnu vrednost nakon dodavanja nove činjenice u bazu (ili se istinitost dobrog dela baze znanja ne menja) i ako to važi za sve činjenice u bazi

znanja onda je to monotono zaključivanje, inače je nemonotono i postoje činjenice koje se menjaju nakon dodavanja nove činjenice u bazu. Praktično, obično se činjenice ili dodaju ili oduzimaju iz baze znanja pored međuzaključaka (RETRACT - povlačenje činjenice, CYCLE - pokretanje pretrage ponovo sa novim stanjem), nakon čega se promene mogu reflektovati na ostatak ili deo baze znanja.

Zaključivanjem se naziva model ljudskog razmišljanja u ES.

**Definicija 4.2** *Zaključivanje (inference) je proces ekspertnog sistema kojim se dobijaju nove informacije na osnovu postojećih.*

Mehanizam zaključivanja uzima prema nekom izboru iz baze znanja neke činjenice u vidu okvira i pravila, kao i instance činjenica iz radne memorije, i onda prema zadatom načinu zaključivanja dobija nove instance činjenica i smešta ih u radnu memoriju ili dobija ciljeve. Pred mehanizam zaključivanja se postavljaju sledeća pitanja pored samog načina tj. pravila zaključivanja za date premise i pravila:

- Kada i koja pitanja postavljati korisniku ?
- Kako pretraživati bazu podataka ?
- Kako izabrati pravilo koje treba primeniti ako ih postoji više u datom trenutku ?
- Kako zaključena informacija utiče na dalju pretragu ?

Dve osnovne vrste zaključivanja bile bi:

- **Nizanje unapred (Forward Chaining)** - od baze znanja stiže se ka cilju (podaci utiču na rezultat, karakteristično za sisteme bazirane na pravilima), koristi se modus ponens ili rezolucija kao način zaključivanja - dokle god se ne zaključi cilj ili ne iscrpu primenjiva pravila.
- **Nizanje unazad (Backward Chaining)** - polazeći od cilja stiže se do činjenica u bazi znanja tako da se utvrdi istinitost cilja (npr. nerezolucija, gde premise postaju podciljevi) - primenjivo pravilo (cilj je sa desne strane pravila) čije premise nisu u bazi znanja ili radnoj memoriji postaju podciljevi, a podcilj jeste primitiva (traži se vrednost od korisnika) ako nema primenjivog pravila. Agenda je spisak ciljeva koje treba

rešiti kao problem (neki put je bitno ne držati se strogo redosleda). Korisno je imati pravila kojima se bira cilj (nizanjem unapred) i to se može postići posebnim pravilima (meta-pravilima) koja utiču na biranje cilja.

Može biti korisno imati mrežu zaključivanja (u toku razvoja ili eksploatacije sistema) - strukturu zaključivanja prikazanu grafom na osnovu činjenica (čvorovi) i pravila (lukovi) koja učestvuju (npr. polazeći od cilja kod nizanja unazad).

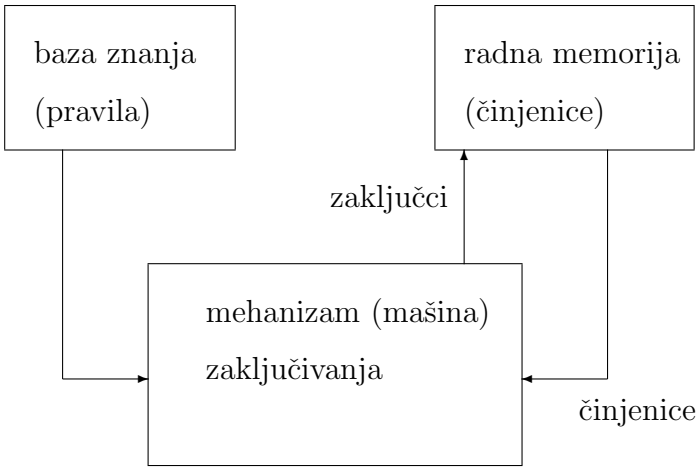
Pojam razrešavanja sukoba (conflict resolution) je strategija odabira pravila koje se okida kada ima više primenjivih pravila. Sastoji se iz 3 faze: prepoznavanja (svih premisa pravila u radnoj memoriji i skupa primenjivih pravila), razrešenja (biranje pravila prema nekoj strategiji), i primene (nakon čega se zaključak dodaje u radnu memoriju). Tipične strategije su: prvo pravilo po redu u memoriji, po zadatom prioritetu (salience), pravilo najveće specifičnosti, pravilo koje se odnosi na poslednji dodat element radne memorije, ne okidaj već upotrebljeno pravilo, pravila iz druge linije zaključivanja.

| | **Dobre strane** | **Loše strane** |
|---|---|---|
| **Nizanje unapred** | dobar za probleme vođene podacima (planiranje, monitoring, interpretacija) | nema predstavu o važnosti pitanja (postavlja i nepotrebna pitanja, sva zadata) |
| **Nizanje unazad** | dobro za potvrdu hipoteze usredsređen je na problem (dijagnoza, savet, debugging) | pratiće liniju zaključivanja iako ne treba (postoje poboljšanja, meta-pravila i faktori uverenja) |

Kombinovanje jednog i drugog se često koristi pored samog izbora između ova dva načina (smera) nizanja tokom projektovanja ES (npr. posmatranjem eksperta). To se može postići komunikacijom između različitih podsistema ili posebnim demon pravilima (koja nizanjem unapred rešavaju specifične slučajeve). Nemonotono zaključivanje može podrazumevati i uklanjanje činjenice (kao promenu u bazi znanja) kao i svih činjenica koje zavise od nje prema nekom pravilu.

# 5 ES bazirani na pravilima (rule-based)

ES bazirani na pravilima su zasnovani na strukturi u kojoj bazu znanja čine pravila (kao dugotrajna memorija), radnu memoriju unete i zaključene činjenice (kao kratkotrajna memorija), kao i mehanizam zaključivanja. Pored ovih osnovnih komponenti strukture tu su i korisnički interfejs, razvojni interfejs (koji može biti isti kao i prethodni, ista školjka (shell) - okruženje, ali predstavlja drugačiji pogled na sistem i možda dozvoljava promenu izvornog koda), komponenta za objanjenja (obično proima sve ostale) i spoljni programi (koji dodatno podravaju ES). Osnovna vrsta ovakvih sistema su produkcioni sistemi gde se pod pravilima podrazumevaju produkcije - relacije između stanja i akcija (leva strana produkcije je stanje određeno radnom memorijom (pattern), desna akcija nad radnom memorijom, tj. nova činjenica). Prvi uspešni primeri su DENDRAL (60-tih), MYCIN i PROSPECTOR (70-tih).




## 5.1 Prednosti ES baziranih na pravilima

- **Prirodno izražavanje** - ljudima je blisko iskazivanje pravila u obliku ako - onda

- **Razdvojeno znanje od kontrole** - kao i kod svih ES, Baza znanja i mehanizam zaključivaja (sa meta-pravilima npr.) su razdvojeni

- **Modularnost znanja** - pravila su nezavisni delovi znanja i tako mogu biti organizovani
- **Lako proširivanje** - dodavanje pravila i činjenica je lako omogućeno razdvojenošću kontrole od znanja
- **Proporcionalni rast inteligencije** - dodavanjem znanja i činjenica raste i sposobnost ES da rešava problem bolje
- **Upotreba relevantnog znanja** - sistem izdvaja bitna pravila za zadati problem (i u tom smislu je dodatno fokusiran)
- **Izdvajanje objašnjenja iz formalne sintakse** - pravila nude mogićnost davanja objašnjenja i praćenja celog traga objašnjenja datog zaključka
- **Provera konzistentnosi** - sama školjka može nuditi mogućnost provere unetog znanja s obzirom na formalnu prirodu pravila i sintakse
- **Heurističko znanje** - moguće je koristiti ,,zdravorazumsko" iskustvo eksperta i zaključivanje (meta-pravilima npr.)
- **Znanje s verovatnoćom** - može se implementirati zaključivanje (pravila i činjenice) s verovatnoćom
- **Upotreba promenljivih** - mogu se koristiti šeme pravila upotrebom promenljivih

## 5.2 Mane ES baziranih na pravilima

- **Potreba za tačnim uparivanjem** - da bi se pravilo okinulo neophodno je da postoji činjenica koja formalno potpuno i tačno odgovara uslovu
- **Nejasni odnosi među pravilima** - među pravilima postoji često međuzavisnost koju nije lako uočiti u toku razvijanja sistema a to može biti važno
- **Performanse** - sistem sa velikom bazom znanja može biti spor ili bar nedovoljno brz za neke primene

- **Odgovaraju samo određenim problemima** - znanje se teško ,,hvata" odnosno reprezentovanje znanja pravilima nije uvek odgovarajuća za, što se i očekuje, i onda se mogu koristiti drugi oblici reprezentovanja znanja (okviri, semantičke mreže, tabele odluka - tabele u kojima su kolone činioci tj. preduslovi i (neposredni) zaključci tj. odluke, itd.) ili rešenja

## 5.3 ES bazirani na pravilima s nizanjem unazad

Tipični koraci u razvoju:

1. Definisanje problema
2. Definisanje ciljeva
3. Definisanje ciljnih pravila (koja zaključuju cilj)
4. Širenje sistema
5. Prečišćavanje sistema (uopštavanje pravila, sigurnosna mreža, optimizacija, itd.)
6. Projektovanje korisničkog interfejsa
7. Evaluacija sistema

Preporuke u razvoju ovakvog sistema bi bile:

- **Razdvojiti problem u potprobleme i njima dodeljene podsisteme** - npr. (MYCIN) - posebno se vrši dijagnoza, zatim ,,ispravka problema" odnosno prepisivanje recepta, ili kontrola i novo rešenje ako prethodno ne pomogne
- **Inteligentan korisnik** - pretpostavlja se da korisnik zna korisne informacije koje mogu da reše problem i treba omogućiti njihov unos (korisniku se može ,,ugađati" dodatnim objašnjenjima i međurezultatima, pogotovu o bitnim fazama i stvarima)
- **Sigurnosna mreža (Safety Net)** - treba uvek imati ,,ELSE" granu ili odg. pravilo u zaključivanju sa nekim pretpostavljenim zaključcima da bi sistem uvek mogao dođe do nekakvog zaključka (makar pretpostavljenog)

- **Dokumentacija** - treba uredno dokumentovati sva pravila i njihovo značenje i efekat
- **Promenljivi cilj** - npr. prepisani recept je promenljiva tako da je podržano više mogućnosti prepisivanja i zaključivanja, kao i kasnije lakše proširivanje sistema
- **Testiranje** - nakon svake izmene baze znanja treba testirati rad sistema, ako se sistem řiri poželjno je to raditi iterativno slučaj po slučaj

## 5.4 ES bazirani na pravilima s nizanjem unapred

Za razliku od prethodnih ES gde su mogući ciljevi poznati i dobro utvrđeni, ovde se rešenje traži i nemora biti poznato unapred. Tipični koraci u razvoju:

1. Definisanje problema
2. Definisanje ulaznih podataka (pravila kojima se zadaje korisnički unos)
3. Definisanje strukture vođene podacima (pravila)
4. Pisanje inicijalnog koda (ne samo pravila koja daju ispravan rezultat, već niz koji daje šablon po kome sva ostala mogu da se napišu)
5. Testiranje sistema
6. Projektovanje korisničkog interfejsa
7. Širenje sistema
8. Evaluacija sistema

Preporuke u razvoju ovakvog sistema bi bile:

- **Sigurnosna mreža, dokumentacija, testiranje** - takođe su primenjivi kao i kod prethodne vrste ES
- **Grupisanje pravila** - poželjno je grupisati pravila, pogotovu prema fazama rada - npr. sistem za dijagnozu ima obično 4 faze: detekciju, izolaciju, dijagnozu i preporučen postupak rešavanja problema.

- **Promenljive** - koristiti šeme pravila upotrebom promenljivih, mada se ovo odnosi i na prethodnu vrstu ES
- **Unos podataka** - obično ako podaci nisu već u radnoj memoriji (ASSERTion) unose se okidanje inicijalnog pravila kojim se traže od korisnika
- **Izuzeci i zaustavljanje rada** - koriste se posebna pravila za specijalne slučajeve koji se obično okidaju pre svih ostalih, kao i mogućnost zaustavljanja rada (STOP) iako postoje još neka primenjiva pravila

# 6 Bajesov pristup neegzaktnom zaključivanju

## 6.1 Teorija verovatnoće i zaključivanje, PROSPECTOR

Bez posebnog definisanja osnovnih pojmova teorije verovatnoće (polje događaja i njihovi svetovi kao skupovi, verovatnoća, slučajna promenljiva i njena raspodela, itd.), osnova ovakvog zaključivanja je Bajesova teorema o uslovnoj verovatnoći:

$$p(H|E) = \frac{p(H)p(E|H)}{p(E)}$$

gde je $H$ neki događaj - hipoteza, a $E$ dokaz - odnosno uslov td. je $H|E$ uslovni događaj ,,H je tačno kad god je E (tačno)" i tada važi pomenuto Bajesovo pravilo koje se može zapisati i kao:

$$p(E) = p(E|H)p(H) + p(E|\neg H)p(\neg H)$$

Ako se definiše odnos $O(E) =_{def} \frac{p(E)}{p(\neg E)} = \frac{p(E)}{1-p(E)}$ kao ,,izgledi za E" (odds), tada su uslovni izgledi za $H$ dati sa $O(H|E) = \frac{p(H|E)}{1-p(H|E)}$. Ako je:

(faktor dovoljnosti) $\lambda =_{def} \frac{p(E|H)}{p(E|\neg H)}$, (faktor potrebnosti) $\overline{\lambda} =_{def} \frac{p(\neg E|H)}{p(\neg E|\neg H)}$

onda je $O(H|E) = \lambda O(H)$, $O(H|\neg E) = \overline{\lambda} O(H)$ i umesto običnog pravila sa premisama i posledicama imamo i dva faktora koja vezujemo za dato pravilo: $E \rightarrow H(\lambda, \overline{\lambda})$. Vrednosti ova dva faktora su, naravno, povezane (npr. $\overline{\lambda} = (1 - \lambda\ p(H|\neg E))/(1 - p(H|\neg E))$, te ako je $\lambda < 1$ onda je $\overline{\lambda} > 1$ i obratno, a važi i $\lambda = 1$ akko $\overline{\lambda} = 1$. O tome treba voditi računa prilikom građenja baze znanja ili nekako prevazići subjektivno zadate vrednosti (koje obično zadaje ekspert koji nije dovoljno svestan teorije verovatnoće i ove povezanosti). Vrednosti faktora šetaju od 0 do $\infty$ pri čemu se $\lambda$ tumači ovako:

| | |
|---|---|
| **0** | $H$ je $\perp$ kada $E$ je $\top$, ili<br>$\neg E$ je neophodno da bude $H$ |
| **malo** | $E$ je nepoželjno da bi bilo $H$ |
| **1** | $E$ ne utiče na $H$ |
| **veliko** | $E$ je poželjno da bi bilo $H$ |
| $\infty$ | $E$ je dovoljno da bi bilo $H$ |

dok se $\overline{\lambda}$ tumači na sledeći način:

| | |
|---|---|
| **0** | $H$ je $\perp$ kada $E$ je $\perp$, ili<br>$E$ je neophodno da bude $H$ |
| **malo** | $\neg E$ je nepoželjno da bi bilo $H$ |
| **1** | $\neg E$ ne utiče na $H$ |
| **veliko** | $\neg E$ je poželjno da bi bilo $H$ |
| $\infty$ | $\neg E$ je dovoljno da bi bilo $H$ |

Ako sa $E'$ obeležimo događaj koji predstavlja dokaz, simptom ili uverenje da je $E$, tada se može zapisati

$$p(H|E') = p(H|E)p(E|E') + p(H|\neg E)p(\neg E|E')$$

i definiše linearnu zavisnost $p(H|E')$ i $p(E|E')$ td. je $p(H|E') = p(H)$ ako je $p(E|E') = p(E)$ tj. kada $E'$ potpuno podržava $E$, ako je $E$ tačno onda je $p(E|E') = 1$, $p(H|E') = p(H|E)$, a ako je $E$ netačno onda je $p(\neg E|E') = 1$, $p(H|E') = p(H|\neg E)$. Ovaj odnos može biti narušen ako se unose nekonzistentne vrednosti pomenutih faktora $\lambda$ i $\overline{\lambda}$. Pored ispravljanja i provere u toku samog unosa, postoji i metod korekcije (Duda, 1976) koji nije sasvim u skladu sa teorijom verovatnoće ali se dobro pokazao u PROSPECTOR

sistemu:

$$p(H|E') = \begin{cases} p(H|\neg E) + \frac{p(E|E')}{p(E)}(p(H) - p(H|\neg E)), & 0 \leq p(E|E') \leq p(E); \\ \frac{p(H) - p(H|E)p(E)}{1-p(E)} + p(E|E')\frac{p(H|E)-p(H)}{1-p(E)}, & p(E) \leq p(E|E') \leq 1. \end{cases}$$

(linearna aproksimacija). Ako imamo konjunkcija više premisa u pravilu $E_1, E_2, ... \rightarrow H$ uz dokaze $E'_1 \wedge E'_2 \wedge ...$ i pretpostavku da su međusobno (uslovno) nezavisni kao i premise, onda se može uzeti da je $p(E|E') = \min_i E_i|E'$ u prethodnoj formuli, odnosno $p(E|E') = \max_i E_i|E'$ ako je disjunkcija $(E'_1 \vee E'_2 \vee ...)$ premisa u pitanju. Ovakvu fuzzy aproksimaciju koristi PROSPECTOR, s tim da se verovatnoća uverenja u neku od premisa unosi kao vrednost između -5 i 5. Domenski ekspert zadaje početne verovatnoće svih premisa, a korisnik unosi verovatnoće uverenja.

Nakon svake nove unete premise preračunati koeficijente izgleda hipoteze (posledice). Za sve složene premise, prema prethodnom računu, pravila sa tom hipotezom (uz pomenutu pretpostavku o nezavisnosti premisa) važi za datu hipotezu:

$$p(H|E'_1, ..., E'_n) = O(H) \prod_{i=1}^{n} \lambda_i$$

gde je $\lambda_i = \frac{O(H|E'_i)}{O(H)} = \frac{p(E_i)H}{p(E_i|\neg H)}$. To npr. ako postoji samo jedno pravilo znači $O(H|E') = \frac{p(H|E')}{1-p(H|E')}$. Takođe važi:

$$p(H|\neg E'_1, ..., \neg E'_n) = O(H) \prod_{i=1}^{n} \overline{\lambda_i}$$

gde je $\overline{\lambda_i} = \frac{O(H|\neg E'_i)}{O(H)} = \frac{p(\neg E_i)H}{p(\neg E_i|\neg H)}$.

## 6.2 Osobine ovakvog pristupa

- **Verovatnoće uslova moraju biti poznate** - verovatnoća svih premisa mora biti zadata pre pokretanja mehanizma zaključivanja, a to nije uvek lako ni moguće, a subjektivne i time neprecizno zadate vrednosti (greške) uzrokuju greške u rezultatu s obzirom na strogu zasnovanost računa

- **Ažuriranje verovatnoća i faktora** - nakon svake promene i unosa verovatnoća uverenja moraju biti ažurirane i sve zavisne vrednosti izgleda, što može biti veoma zahtevno

- **Ukupna verovatnoća** - mora biti $p(H|E) + p(\neg H|E) = 1$ na osnovu teorije verovatnoće, ali u praksi ekspert može biti nezadovoljan takvim odnosom hipoteze i njene negacije (zato je možda interesantnija teorija uverenja)

- **Nezavisnost premisa** - je preduslov, što ako nije izraženo nemora biti problem, i mora se prilikom razvoja o tome voditi računa. Pomenuta aproksimacija konjunkcije minimumom (i disjunkcije maksimumom) ima tu manu da male promene ostalih uslova koji nisu minimum ne utiču na hipotezu. Složenijom formulom se to može rešiti ali se tako račun čini dodatno složenijim.

# 7 Teorija uverenja

## 7.1 Zaključivanje s uverenjem

Ovde se se događaj ne posmatra formalno kao u teoriji verovatnoće iako ima sličnosti i dodira, već se značenje uverenja naspram verovatnoće vezuje za praktično iskustvo, intuitivno ljudsko znanje i drugačiji formalni aparat. Uverenje predstavlja meru da je nešto moguće odnosno verovanja da je tako (*moguće* naspram *verovatno*). Pravilo $E_1 \wedge E_2 ... \rightarrow H\beta$ tako ima faktor uverenja $\beta$ (certainity factor) koji ima vrednost od $-1$ (potpuno netačno) do 1 (potpuno tačno). MYCIN koristi takav pristup, s tim da je naglasak bio na mehanizmu i formuli koja bi imala osobine: komutativna (da bi se izbegla zavisnost rezultata od redosleda primene pravila) i asimptotna (svako pravilo koje dodatno podupre uverenje ga povećava asimptotski ka 1).

## 7.2 Mere verovanja i neverovanja i ukupno uverenje

Uvode se mere verovanja $\mu_B$ (belief) i neverovanja $\mu_D$ (disbelief) takođe tako da budu komutativne i asimptotne. Nakon prikupljanja svih podataka (za i protiv) i računanja ovih mera za datu hipotezu $H$ se određuje ukupno uverenje (net belief): $\beta = \mu_B - \mu_D$. Na osnovu dokaza $E$ uverenje u hipotezu se može uvećati ako je $p(H|E) > p(H)$ ili smanjiti ako je $p(H|E) < p(H)$ (odnosno, povećava se neverovanje u potonjem slučaju):

$$\mu_B(H,E) = \begin{cases} 1, & \text{ako je } p(H) = 1 \\ \frac{\max[p(H|E), p(H) - p(H)}{1 - p(H)]}, & \text{inače.} \end{cases}$$

$$\mu_D(H,E) = \begin{cases} 1, & \text{ako je } p(H) = 0 \\ \frac{\min[p(H|E), p(H) - p(H)]}{-p(H)}, & \text{inače.} \end{cases}$$

... i odavde se vidi da je $0 \leq \mu_B(H,E) \leq 1$ i $0 \leq \mu_D(H,E) \leq 1$. Dalje, $\beta(H,E) = \mu_B(H,E) - \mu_D(H,E)$ ima vrednost -1 ako $E$ potpuno opovrgava $H$, 0 ako $E$ nije dokaz (nedostatak dokaza - $E$ je nezavisan od $H$ tj.

$p(H|E) = p(H)$ pa su obe mere i uverenje onda jednake 0), ili 1 ako $E$ potpuno potvrđuje $H$. Zbir $\beta(H,E) + \beta(\neg H,E)$ uopšte ne mora biti 1.

## 7.3 Propagiranje uverenja

Za dato pravilo

$$E \rightarrow H \ \beta(PRAVILO)$$

računa se uverenje kao $\beta(H,E) = \beta(E)\beta(PRAVILO)$. Ako je u pitanju konjunkcija

$$E_1 \wedge E_2... \rightarrow H \ \beta(PRAVILO)$$

onda je:

$$\beta(H, E_1 \wedge E_2...) = \min_i \beta(E_i)\beta(PRAVILO)$$

a ako je disjunkcija

$$E_1 \vee E_2... \rightarrow H \ \beta(PRAVILO)$$

u pitanju onda je:

$$\beta(H, E_1 \vee E_2...) = \max_i \beta(E_i)\beta(PRAVILO)$$

Ako dva pravila zaključuju o istoj hipotezi, onda se ,,akumulira" uverenje prema (Shortliffe, Buchanan, 1975):

$$\mu_B(H, E_1 \& E_2) = \begin{cases} 0, & \mu_D(H, E_1 \& E_2) = 1 \\ \mu_B(H,E_1) + \mu_B(H,E_2)(1 - \mu_B(H,E_1)), & \text{inače.} \end{cases}$$

$$\mu_D(H, E_1 \& E_2) = \begin{cases} 0, & \mu_B(H, E_1 \& E_2) = 1 \\ \mu_D(H,E_1) + \mu_D(H,E_2)(1 - \mu_D(H,E_1)), & \text{inače.} \end{cases}$$

... odnosno, ako se odmah računa uverenje ($\beta_1$ i $\beta_2$ su izračunata uverenja dvaju pravila):

$$\beta(\beta_1, \beta_2) = \begin{cases} \beta_1 + \beta_2(1 - \beta_1), & \text{ako je } \beta_1, \beta_2 > 0 \\ \frac{\beta_1 + \beta_2}{1 - \min[|\beta_1|, |\beta_2|]}, & \text{jedan od } \beta_1, \beta_2 < 0 \\ \beta_1 + \beta_2(1 + \beta_1) & \text{inače.} \end{cases}$$

## 7.4 Preporuke i zaključci

- **Heuristika** - ako treba odabrati koje pravilo okinuti, ima smisla izabrati ono sa najvećim koeficijentom uverenja (best-first)
- **Meta-pravila s uverenjem** - ako npr. vrednost koeficijenta uverenja po trenutnoj grani padne ispod nekog praga odbaciti trenutni i postaviti drugi cilj
- **Čišćenje pretrage** - ako npr. koeficijent uverenja upadne u interval $[-0.2, 0.2]$ onda se odbacuje trenutni cilj
- **Unos vrednosti** - neka vrsta primitive može biti koeficijent uverenja
- **Atributi sa više vrednosti** - pravila s uverenjima mogu imati i premise i zaključke sa više vrednosti (u smislu O-A-V)
- **Biranje zaključivanja** - mogu se kombinovati egzaktno i neegzaktno zaključivanje s uverenjem
- **Sortiranje rezultata po uverenju** - prednost ovakvog zaključivanja
- **Problem s dubokim lancem pretrage** - treba ih izbegavati (zbog prirode računa se ,,gubi uverenje", povećava greška)
- **Mnogo pravila s istom hipotezom** - može biti problem ako se pokaže da daje precenjeno jako uverenje
- **Problem s disjunkcijom** - složene premise mogu biti problem - onda je bolje složeno pravilo razbiti u više manjih ali konzistentnih pravila

# 8 Fuzzy logika

Fuzzy sistemi potiču još od 1930. (Lukasiewicz, poznat i po ,,poljskoj" tj. prefiksnoj notaciji, daje samo načelnu ideju koju ne sprovodi dosledno) a Zadeh (1965) to formalizuje do kraja i nastaje formalna teorija fuzzy logike.

## 8.1 Osnovni pojmovi

Fuzzy logika se zasniva na skupovima i elementima čija se pripadnost meri pre nego da egzaktno pripradaju ili ne pripadaju skupu (naspram Aristotelovske logike, ,,fuzzy" - nejasan). Lingvističke promenljive su iskazi prirodnog jezika koji imaju tako nejasno značenje kao npr. visina sa vrednostima nizak, srednje, visok. Onda svaka od vrednosti daje fuzzy podskup vrednosti iz domena skupa visina.

**Definicija 8.1** *Ako je $X$ domen sa elementima $x$, fuzzy skup $A$ od $X$ domena je karakterisan funkcijom pripadnosti*

$$\mu_A(x) : X \to [0, 1]$$

*koja dodeljuje elementu $x$ stepen pripadnosti skupu $A$.*

Tako se može zapisati $\mu_A(x) = Degree(x \in A)$ gde je $0 \leq \mu_A(x) \leq 1$. Fuzzy teorija je tako proširenje klasične teorije skupova - ako je $\mu_A(x) = 1$ onda je $x \in A$, odnosno ako je $\mu_A(x) = 0$ onda je $x \notin A$ u klasičnom smislu.

## 8.2 Reprezentovanje

Fuzzy skup s diskretnim (,,crisp") vrednostima se može jednostavno prikazati kao vektor karakterističnih vrednosti $A = (a_1, ..., a_n)$ ili određenije kao niz uređenih parova $A = ((a_1, x_1), ..., (a_n, x_n))$ gde je $a_i = \mu_A(x_i)$. Uz konvenciju zapisa uređenih parova sa ,,/" i unije kao ,,+" to se može zapisati i kao $A = \sum_{i=1}^{n} \mu_A(x_i)/x_i$ ako je $X$ diskretan, odnosno $A = \int_i \mu_A(x_i)/x_i$ ako nije. Skraćeni zapis koji se najčešće koristi je samo $\mu_A(x)$ uz podrazumevane vrednosti domena $X$.

U praksi se često koriste dodante operacije nad karakterističnom funkcijom kojom se određuju (odnosno modifikuju) dodatno granice tj. **ograničenja ili odredbe (hedges)** pripadnosti skupu (proširuju je ili skupljaju - u narednim primerima se pretpostavlja da je $A$ skup visokih osoba):

- **Koncentracja (VEOMA)** - $\mu_{CON(A)}(x) = (\mu_A(x))^2$ npr. koncetracija daje skup VEOMA visokih osoba
- **Dilatacija (DONEKLE)** - $\mu_{DIL(A)}(x) = (\mu_A(x))^{1/2}$ npr. dilatacija daje skup DONEKLE (MANJE ILI VIŠE) visokih osoba
- **Intenziviranje (ZAISTA)** -
  $\mu_{INT(A)}(x) = \begin{cases} 2(\mu_A(x))^2, & \text{ako je } 0 \leq \mu_A(x) \leq 1/2 \\ 1 - 2(1 - \mu_A(x))^2, & \text{ako je } 1/2 < \mu_A(x) \leq 1 \end{cases}$
  npr. intenziviranje daje skup zaista visokih osoba (intenzivira pripadnost izraženo visokih, a smanjuje pripadnost ostalih)
- **Snažno (VEOMA VEOMA)** - $\mu_{POW(A,n)}(x) = (\mu_A(x))^n$ pojačanje $\mu_{POW}$ za n=3 ili veće ...

Operacije nad fuzzy skupovima:

- **Presek** -
  $\mu_{A \wedge B}(x) =_{def} min[\mu_A(x), \mu_B(x)] = \mu_A(x) \wedge \mu_B(x) = \mu_A(x) \cap \mu_B(x)$ za sve $x \in X$ ... osim što je ovim definisan presek fuzzy skupova, ovo takođe tretira i kao logička operacija ,,i"
- **Unija** -
  $\mu_{A \vee B}(x) =_{def} max[\mu_A(x), \mu_B(x)] = \mu_A(x) \vee \mu_B(x) = \mu_A(x) \cup \mu_B(x)$ za sve $x \in X$ ... osim što je ovim definisana unija fuzzy skupova, ovo takođe tretira i kao logička operacija ,,ili"
- **Komplement** - $\mu_{\neg A}(x) =_{def} 1 - \mu_A(x)$ ... tretira se i kao logička negacija

Operatorima i ograničenjima se prave derivati fuzzy skupova.

## 8.3 Fuzzy zaključivanje

Fuzzy logika tretira fuzzy skup kao *fuzzy iskaz*. Uopšte, fuzzy iskaz se može predstaviti kao ,,X je A" (gde je $X$ domen, A fuzzy skup), a fuzzy pravilo kao

$$X \; je \; A \rightarrow Y \; je \; B$$

Ovakvo pravilo uspostavlja relaciju među fuzzy iskazima i obično se takvo pravilo zapisuje u obliku matrice. Ovakva se fuzzy asocijativna matrica $M$

koja mapira fuzzy skup $A$ u fuzzy skup $B$ zove još i Fuzzy Associative Memory (FAM - Kosko, 1992). Umesto običnog linearnog mapiranja $\mathbf{b} = \mathrm{M}\ \mathbf{a}$ zadatog matričnim množenjem $b_j = \sum_{i=1}^{n} a_i m_{ij},\ j = \overline{1,n}$ obično se koristi operator $\circ$ max-min kompozicije $\mathbf{b} = \mathrm{M} \circ \mathbf{a}$ zadat sa $b_j = max_{i=\overline{1,n}} min[a_i, m_{ij}],\ j = \overline{1,n}$. Zadeh daje jedan opštiji pristup u kome se računanjem uslovnih verovatnoća (koristi termin distribucija mogućnosti, possibility distribution, umesto karakteristične funkcije) dobija pomenuta matrica $\prod_{B|A}$ sa klasičnim množenjem matrica, td. važi $\prod_B = \prod_{B|A} \prod_A$. Ovakav pristup naziva *kompozicionim pravilom zaključivanja.* Daje uopšteni postupak dobijanja matrice $\prod_{B|A}$:

$$\prod_{B|A} = \begin{pmatrix} a_1 \to b_1 & a_1 \to b_2 & \cdots \\ a_2 \to b_1 & \ddots & \\ \vdots & & \end{pmatrix} = (m_{ij}) = M$$

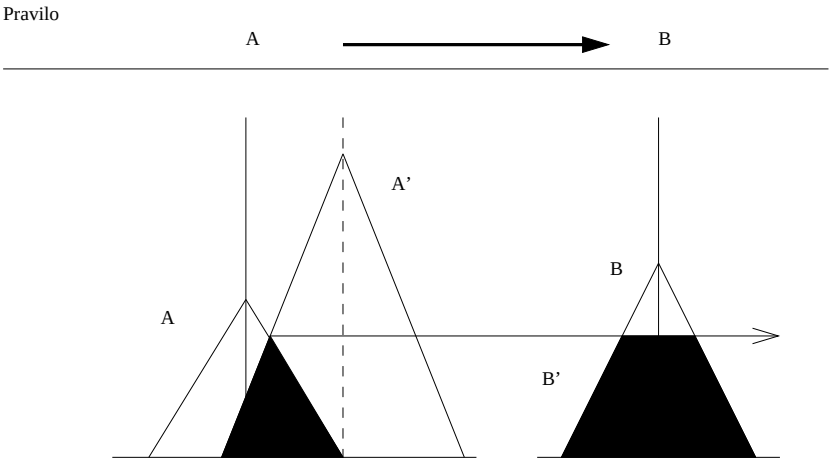
Zavisno od definisanja operatora implikacije definiše se i matrica $M$.

## 8.4 Max-Min zaključivanje

Ako se minimumom definiše operator implikacije $m_{ij} = Val(a_i \to b_j) = min(a_i, b_j)$ onda se za data dva fuzzy skupa na osnovu ove formule definiše matrica $M$.

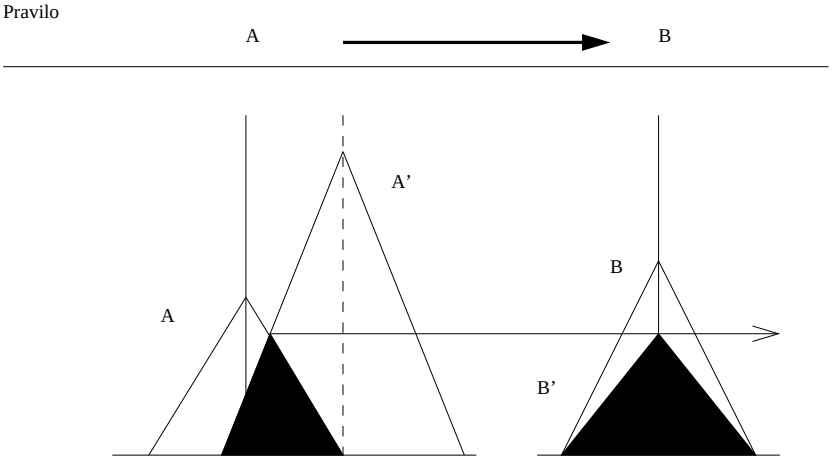
Ako su u pitanju ,,trougani" fuzzy skupovi (granica linearna), onda se slikanje nekog skupa $A'$ svodi na odsečak $B$ na visini vrha preseka $A$ i $A'$ na niže, što proizilazi iz definicije i osobina ovakvog preslikavanja. Npr. ako je $\mu_A(x_k)$ pomenuti vrh ili jedna diskretna izmerena vrednost, što se najčešće koristi kao ulaz (kao da su ostale vrednosti ulaznog vektora 0), važi za diskretne vrednosti $y \in X$:

$$b(y) = \mu_A(x_k) \wedge \mu_B(y),\ y \in X$$




## 8.5 Max-Proizvod zaključivanje

Ovaj način zaključivanja se dobija se proizvodom definiše operator implikacije $m_{ij} = a_i b_j$.




Preslikavanje trouglova ima osobine $b(y) = \mu_A(x_k) \cdot \mu_B(y),\ y \in X$ ako je $\mu_A(x_k)$ pomenuti vrh) slične prethodnom, ali se ovde dobija ,,sniženi" ali ceo trougao umesto odsečka. Za jednu ulaznu diskretnu vrednost onda važi:

$$b(y) = \mu_A(x_k)\mu_B(y),\ y \in X$$

## 8.6 Pravila sa više premisa

Ako imamo dve premise $A$ i $B$ (može ih biti i više, analogno) razrešenje možemo naći (Kosko, 1992) pošavši od toga kao da imamo dva pravila $A \rightarrow C$ i $B \rightarrow C$ sa svojim matricama $M_{AC}$ i $M_{BC}$ td. važi:

$$A' \circ M_{AC} = C_{A'}$$

$$B' \circ M_{BC} = C_{B'}$$

Tada se definiše $C' = C_{A'} \wedge C_{B'}$ ako je u pitanju konjunkcija $A \wedge B \rightarrow C$, odnosno $C' = C_{A'} \vee C_{B'}$ ako je u pitanju disjunkcija $A \vee B \rightarrow C$. U ranije pomenutom specijalnom slučaju trouglastih skupova, minimum odnosno maksimum respektivno konjunkcija odnosno disjunkcija vrhova trouglova određuje prag odsecanja odnosno sabijanja (zavisno od toga da li se koristi max-min ili max-proizvod zaključivanje) skupa $B$. Za date diskretne ulazne vrednosti $a_i = \mu_A(x_i)$ i $b_j = \mu_B(y_j)$ i diskretne vrednosti $z \in X$ domena važi onda:

| $C'$ | Spajanje | Zaključivanje |
|---|---|---|
| $min(a_i, b_j) \wedge \mu_C(z)$ | I | Max-Min |
| $max(a_i, b_j) \wedge \mu_C(z)$ | ILI | Max-Min |
| $min(a_i, b_j)\mu_C(z)$ | I | Max-Proizvod |
| $max(a_i, b_j)\mu_C(z)$ | ILI | Max-Proizvod |

## 8.7 Defazifikacija (Defuzzification)

Ako imamo fuzzy zaključak, njegovo tumačenje je defazifikacija i obično se nekim postupkom izdvoji jedna diskretna vrednost fuzzy skupa kao reprezent. Najčešće koristi **fuzzy centroid** odnosno nekakva sredina u odnosu na pripadnost kao težinu (najbliža vrednosti u $X$):

$$y' = \frac{\sum_{i=1}^{n} y_i \mu_{B'}(y_i)}{\sum_{i=1}^{n} y_i}$$

Ako se traži zaključak na osnovu više fuzzy pravila $A_1 \rightarrow B_1$, ..., $A_n \rightarrow A_n$ onda se uzima da je ukupni fuzzy zaključak $B' = \bigcup_{i=1}^{n} B'_i$ tj. $\mu'_B(x) = max_{i=\overline{1,n}}[\mu_{B'_i}(x)]$ i njegov centroid se tumači kao diskretna vrednost zaključka na osnovu svih polaznih premisa $A'$.

## 8.8 Razvoj fuzzy sistema, preporuke

Uobičajeni koraci u razvoju fuzzy sistema su:

1. **Definisanje problema** - slično kao i kod ostalih ES
2. **Definisanje lingvističkih promenljivih** -
3. **Definisanje fuzzy skupova** - obično se formira tabela čije su kolone raspoložive lingvističke promenljive a redovi pridevi koje koristi ekspert (diskretne vrednosti na domenu). Poželjno je da se susednu skupovi preklapaju (npr. ako se koriste trouglovi da se dodiruju parni odnosno neparni redom na domenu).
4. **Definisanje fuzzy pravila** - pravila se mogu definisati na nivou prethodne tabele kojom je dobar deo posla urađen (u CubiCalc školjki zapis pravila podseća sintaksno na LEVEL5 jer se zadaju simbolički)
5. **Izgradnja sistema** - može se razvijati posebno ili koristiti gotovo okruženje, slično ostalim ES
6. **Testiranje** - osim ispravljanja grubih grešaka, korisno je i za naredni korak
7. **Dodatno podešavanje (tuning)** - pravila se mogu preispitati, tamo gde je potrebna veća osetljivost povećava se broj podskupova i sužavaju trouglovi, dodaju se odredbe, itd.

# 9 Sistemi zasnovani na okvirima

Okvir i instanca se u mnogo čemu mogu uporediti (i poistovetiti bez veće greške) sa pojmom klase i objekta u objektnom programiranju od kojeg je vremenom dosta toga ,,pozajmljeno" iako pojam okvira potiče još od Minskog (1975). Tako se klasa sastoji iz (opisa) osobina i metoda (šta objekat može), dok se u ES terminologiji kaže da se okvir sastoji iz deklarativnog (osobine, slotovi) i proceduralnog znanja (metodi). Prisutni su npr. tipični odnosi među klasama bazirani na osnovnoj osobiini nasleđivanja (Rumbaugh, 1988):

- **Generalizacija** - odnos oblika ,,vrsta-od"
- **Agregacija** - odnos oblika ,,Deo-od"
- **Asocijacija** - semantički odnos

Moguće je vištruko nasleđivanje (od više različitih klasa) i upravljanje izuzecima (nasleđena osobina se menja, specijalan slučaj). Specifičnost okvira su **faceti** - pružaju dodatnu kontrolu nad slotovima:

- Tip
- Default (pretpostavljena vrednost)
- Dokumentacija
- Constraint (dozvoljene vrednosti)
- Minimalna, maksimalna kardinalnost vrednosti (u smislu A-O-V)
- if-needed (za čitanje) metod
- if-changed (za upis) metod

Metodi tipa if-needed i if-changed se mogu se koristiti za komunikaciju među instancama, a negde se koristi mehanizam slanja poruka (poziv metoda po nazivu i opciono zadatim argumentima).

Pored ranijih preporuka o razvoju ES i metodologija objektno-orijentisanog programiranja mogu se izdvojiti sledeći koraci u razvoju sistema baziranog na okvirima:

- **Definisanje problema**
- **Analiziranje domena** - definisanje objekata sa osobinama, događajima i arhitekturom
- **Definisanje klasa** - na osnovu prethodnog uz npr. facete i druge dorade
- **Definisanje instanci**
- **Definisanje pravila**
- **Definisanje komunikacije među objektima**
- **Razvoj interfejsa**
- **Evaluacija sistema**
- **Širenje sistema**

Primeri školjki: Kappa, CLIPS.

# 10 Indukcioni sistemi

## 10.1 ID3 algoritam

Kroz primer ID3 algoritma - ID3 je potomak CLS algoritma (Hunt et al. 1966) koji se koristi kao podalgoritam. CLS je u stanju da klasifikuje zadate primere u dve grupe - pozitivnu (DA, pripada klasi) i negativnu (NE, ne pripada klasi). Počinje s praznim drvetom odlučivanja koje gradi i zaustavlja se kad drvo odlučivanja (koncept, pravilo) može pravilno da klasifikuje sve zadate probne primere:

1. Ako su svi primeri u $C$ pozitivni, generiši DA čvor i stani.
   Ako su svi primeri u $C$ negativni, generiši NE čvor i stani.
   Inače, (koristeći neku heuristiku) izdvoj neki atribut $A$ s vrednostima $V_1$, $V_2$, ..., $V_n$ i kreiraj čvor odlučivanja

2. Particioniši $C$ u $C_1$, $C_2$, ..., $C_n$ prema vrednostima $A$

3. Primeni algoritam rekurzivno na svaki od podskupova $C_i$

Heuristika koju ID3 koristi da bi odabrao atribut vezuje se za očekivanu količinu informacija $M(C) = -p^+ \log_2 p^+ - p^- \log_2 p^-$ td. se bira atribut koji maksimizuje $M(C) - B(C, A)$ gde je $B(C, A) = p(A = V_i)M(C_i)$.
ID3 algoritam dalje sledi:

1. Izaberi proizvoljan podskup $W$ primera iz $C$ zadate velčine (prozor, window)

2. Primeni CLS na $W$ (rezultat je drvo odlučivanja tj. pravilo)

3. Proveri sve primere u $C$ koji su izuzeci trenutno dobijenom pravilu

4. Ako postoje izuzeci, uvrsti ih u $W$ i pokreni ponovo korak 2. i prikaži dobijeno pravilo

ID3 je u stanju da klasifikuje mnogo veći broj primera nego CLS koji je ograničen početnom pretragom svih primera.

## 10.2 Prednosti i mane ID3

**Prednosti** su:

- **Biranje skupa primera** - ID3 se zasniva na *filterisanju vođenom izuzecima* što se smatra prednošću jer algoritam može da se usresredi na izuzetke ako se tako postavi prozor. CLS i ID3 su veoma dobri za probleme s **jednim konceptom**.
- **Heuristika** - bira se atribut koji je najviše diskriminatoran (najviše utiče na klasifikaciju) i zato je posebno efikasan

**Mane** su:

- **Pravila nisu verovatnosna**
- **Više identičnih primera** - ne utiče više nego jedan
- **Kontradiktorni primeri** - ne snalazi se sa kontradiktornim primerima
- **Velika osetljivost na male promene primera**

## 10.3 Razvoj indukcionih sistema

- **Ustanoviti cilj**
- **Utvrditi činioce odlučivanja** - atribute
- **Utvrditi vrednosti činioca odlučivanja**
- **Utvrditi rešenja** - koje vrednosti mogu krajnji čvorovi drveta odlučivanja da uzmu
- **Formiranje skupa primera**
- **Formiranje drveta pretrage** - upotreba neke školjke kao što je ID3
- **Testiranje**
- **Revizija sistema**

Indukcija nudi mogućnost da se izbegne otkrivanje znanja kao jer se znanje dobija iz ranije unetih primera, a iz tih primera donosi otkriva i novo znanje, otkriva kritične činioce odlučivanja, može da eliminiše nebitne, može da eliminiše kontraditorne primere (još prilikom unosa kod nekih školjki). Međutim, često je teško odabrati i izdvojiti činioce odlučivanja (može loše da utiče na rad sistema), teško je razumeti drvo odlučivanja složenijih problema, i na kraju, indukcioni sistemi se primenjuju samo na probleme klasifikovanja. Primeri su: Meta-DENDRAL, AQ11, WILLARD.

# 11 Prikupljanje znanja

Kako je pomenuto, ovo je usko grlo razvoja ES jer je često veoma teško izvesti ovu fazu. Postoje različite tehnike otkrivanja znanja prilagođene različitim tipovima sistema, ali osnova je razgovor (struktuirani i nestruktuirani intervju) sa ekspertom za šta opet postoje standardni praktični postupci. Nakon prikupljanja informacija od eksperta sledi analiza sakupljenog znanja na osnovu čega se izdvajaju važne činjenice i strukture koje pomažu i utiču na dalji tok razvoja sistema. Prikaz gotovih primera (case studies, retrospektivno proučavanje i posmatračko - uživo) kao šablon kroz koji se može vršiti procena, demonstracija, i sl. Protokol (pojam kognitivne psihologije) kao beleška postupaka koje lice ili sistem čini da bi rešio problem. Otkrivanje znanja:

- Skupljanje
- Interpretacija
- Analiza
- Razvoj sledeće sesije

# 12 Inženjerstvo znanja

Neki aspekti inženjerstva znanja, sličnosti i razlike u odnosnu na uobčajene postupke projektovanja informacionih sistema (software engineering) su već pomenuti. Razlike pre svega potiču od strukture i osobina ES - što se može uporediti sa deklarativno-proceduralnom kontroverzom. Osnovni tok upravljanja projektom razvoja ES:

- **Utvrđivanje problema**
  - Motiv organizacije (upravljen problemom ili rešenjem pre nego problemom)
  - Kandidati problema (Identifikacija, lista, demonstracija)
  - Studija izvodljivosti (Zahtevi, rizici i problemi, ljudi (pored ranije pomenutih i menadžment je od presudnog značaja), implementacija)
  - Analiza cene i dobijenog
  - Odabir projekata
  - Pisanje predloga projekta (cilj, pregled, problem, rešenje, plan i raspored, dobijeno, ljudi, cena)
- **Prikupljanje znanja** item **Razvoj sistema**
  - Biranje reprezentovanja
  - Mehanizam kontrole
  - Biranje sistema za razvoj
  - Prototip
  - Korisnički interfejs
  - Razvoj proizvoda (greška u ranijim fazama manje košta nego u kasnijim)
- **Testiranje i evaluacija**
  - Validacija sistema (rezultata i zaključivanja)
  - Validacija rezultata (kriterijum, slučajevi i ljudi koji vrš evaluaciju)

- **Dokumentovanje**
- **Održavanje**
  - Dokumentacija
  - O održavanju treba razmišljati još tokom razvoja
  - Modularnost, meta-pravila
  - Razdvojeno znanje (kao kontrola) od informacija
  - Školjka, portabilnost
  - Važno je ko održava sistem i pod kojim uslovima, dnevnik promena

# 13 Primer - sistem za generisanje i klasifikaciju muzičkog zapisa

## 13.1 Opis i arhitektura sistema

Kao školjku i okruženje sam koristio CLIPS - sistem koji po sintaksi veoma podseća na LISP, podržava upotrebu pravila nizanjem unapred i okvire, nastao po ugledu na komercijalnu varijatnu CLIPS/R3 koja podržava i nizanje unazad (mada postoji i ,,nezvanična" koja to može). Postoje i mnoge druge implementacije i dijalekti kao što je to JESS realizovan na Java platformi. Kao razvojno okruženje dovoljan je tekst editor, mada sam probno koristio i Protégé kao GUI koji podržava i CLIPS. Korisnički interfejs se u prvom sloju oslanja na tekstuelni ulaz i izlaz koji se svodi na abc format namenjen tekstuelnom obliku notnog zapisa koji se lako dalje transformiše u ili iz MIDI formata ili nekog drugog. Kada sistem dostigne neki stadijum zrelosti sve ove komponente mogu se objediniti u GUI-u napisanom na Java platformi.

## 13.2 CLIPS

Ukratko, CLIPS (autor Joseph C. Giarratano) je sistem baziran na pravilima (koristi RETE algoritam koji veoma efikasno uparuje pravila - pattern matching, algoritam koji podseća na algoritam unifikacije samo mnogo jednostavniji ali opet dovoljno moćan), omogućava i upotrebu klasa i faceta. Sintaksno je veoma blizak LISP-u sa svim prednostima uz to osim što ne podržava listu kao strukturu, niti podržava direktno enkapsulaciju druge klase kao osobine tj. slota (postoji, doduše, mogućnost indirektnog referenciranja instance na drugu instancu).

Primer sesiju u CLIPS-u:

```
CLIPS> (defclass DUCK (is-a USER) (multislot sound (default quack quack))
 (message-handler silence))
CLIPS> (defmessage-handler DUCK silence () (send ?self put-sound nil)
 (printout t (instance-name ?self) " can say now " ?self:sound crlf))
CLIPS> (make-instance [Dorky1] of DUCK)
[Dorky1] of DUCK
CLIPS> (make-instance [Dorky2] of DUCK (sound quack))
[Dorky2] of DUCK
```

```
CLIPS> (make-instance [Dorky3] of DUCK (sound cough))
[Dorky3] of DUCK
CLIPS> (make-instance [Dorky] of DUCK (sound cough cough))
[Dorky4] of DUCK
CLIPS> (deffacts dummy (a b) (c d))
CLIPS> (defrule find-sound ?duck <- (object (is-a DUCK)(sound
\$?find)(name [Dorky])) =>
 (printout t "Duck " (instance-name ?duck) " says " ?find crlf)
 (assert (says ?find)))
CLIPS> (defrule remove-sound ?fact <- (says \$?X) => (retract ?fact)
 (printout t "Removed " ?X "!" crlf))
CLIPS> (run)
Duck [Dorky] says (cough cough)
Removed (cough cough)!
CLIPS> (send [Dorky1] silence)
[Dorky1] can say now (nil)
CLIPS> (defrule delcd (a b) ?fact<-(c d) => (retract ?fact))
CLIPS> (agenda)
0   delcd: f-1,f-2
For a total of 1 activation
CLIPS> (facts)
f-0 (initial-fact)
f-1 (a b)
f-2 (c d)
For a total of 3 facts.
CLIPS> (run)
CLIPS> (facts)
f-0 (initial-fact)
f-1 (a b)
For a total of 2 facts.
```

Osnovna verzija CLIPS-a je besplatna i Open Source, dok verzije sa naprednijim mogućnostima kao što su backward chaining i neegzaktno zaključivanje nisu takve (NASA-in projekat CLIPS/R3, zatim ECLIPSE, itd. - mada postoje nezvanične modifikacije Open Source verzije koje to podržavaju). Postoji ugrađena osnovna hijerarhija objekata COOL (CLIPS Object-Oriented Language framework), mogućnost uticanja na rad sistema (options) i pre svega bogatstvo opisa levih strana pravila koje čini ovaj sistem veoma upotrebljivim.

## 13.3 Protégé

Grafičko okruženje i sistem koji je nastao kao otvoreni projekat na Stanford univerzitetu, u kome se mogu organizovati i razvijati pre svega ontologije (recimo neke vrste uopštenja okvira uz forme kao dodatnu specifikaciju klase vezanu npr. za korisnički interfejs) pa time i okviri i objekti. Okruženje je modularno tako da se npr. može dodati neki od postojećih ili sopstveni ,,TAB" (plug-in) kao proširenje. Postoji CLIPSTab koji je predstavlja potpunu implementaciju standardnog CLIPS-a i u kojem se može importovati Protégé ontologija (uz neka ograničenja). Veoma je korisno ako se razvija složen sistem sa dovoljno velikim brojem okvira i instanci. Više o tome na stranici [PROTEGE].

## 13.4 Jess

Jess (autor Ernest J. Friedman-Hill) nastao po uzoru na CLIPS je realizovan na Java platformi, dobar deo Jess koda je kompatibilan sa CLIPS-om i obratno. Nema klasa i objekata kao u standardnoj CLIPS implementaciji (već se oslanja na svoju sintaksu i Java mrežu klasa), ali ima ugrađen backward chaining (npr. ciljno pravilo stvara podciljeve oblika need-podcilj gde je podcilj deo šablona s leve stranje pravila tj. činjenica dela leve strane ciljnog pravila) i može npr. da koristi regularne izraze.

Primer sesije u Jess-u:

```
Jess> (clear)
TRUE
Jess> (assert (a b))
<Fact-0>
Jess> (defrule x (a ?X) => (assert (c)))
TRUE
Jess> (defrule y (c) => (printout t "Hej !"))
TRUE
Jess> (rules)
MAIN::x
MAIN::y
For a total of 2 rules in module MAIN.
Jess> (reset)
TRUE
Jess> (deffacts s (a c) (d c) (a d) (a f))
TRUE
```

```
Jess> (run)
0
Jess> (facts)
f-0   (MAIN::initial-fact)
For a total of 1 facts in module MAIN.
Jess> (reset)
TRUE
Jess> (facts)
f-0   (MAIN::initial-fact)
f-1   (MAIN::a c)
f-2   (MAIN::d c)
f-3   (MAIN::a d)
f-4   (MAIN::a f)
For a total of 5 facts in module MAIN.
Jess> (agenda)
[Activation: MAIN::x  f-4 ; time=5 ; salience=0]
[Activation: MAIN::x  f-3 ; time=4 ; salience=0]
[Activation: MAIN::x  f-1 ; time=2 ; salience=0]
For a total of 3 activations in module MAIN.
Jess> (run)
Hej !f!Hej !d!Hej !c!6
Jess> (run)
0
Jess> (facts)
f-0   (MAIN::initial-fact)
f-1   (MAIN::a c)
f-2   (MAIN::d c)
f-3   (MAIN::a d)
f-4   (MAIN::a f)
f-5   (MAIN::c f)
f-6   (MAIN::c d)
f-7   (MAIN::c c)
For a total of 8 facts in module MAIN.
```

Jess se posebno licencira (vlasništvo Sandia National Laboratories za internu i akademsku upotrebu jeste besplatan) i nije ni u kom obliku open source. Zvanični sajt je [JESS].

## 13.5 abc

Ovaj način zapisivanja muzike je nastao početkom 90-tih (autor Chris Walshaw) motivisan željom da formalan notni zapis bude u ascii formatu i koristio se u početku za prikupljanje zapisa folklora i etno muzike (Steve Allen ga kasnije prijavljuje kao standardni MIME format). Najpre je napravljen alat *abc2mtex* koji takav zapis pretvara u TEX kod (uz *mtex* biblioteku), kasnije *abc2ps* i *abcMIDI* za konverziju u *PostScript* i *MIDI* (može i obratno, iz MIDI-ja u abc, MIDI je preko 20 godina standardni format notnog muzičkog zapisa koji se koristi pre svega za reprodukciju dok su TEXi PostScript namenjeni štampi). Više se može naći na stranicama [CW], BNF sintaksa na [NB], FAQ na [ABC]. Sistem koji opisujem koristi abc2midi iz abcMIDI paketa James Allwright-a (ili implementacija koju je napravio Guido Gonzato) napravljenog pod otvorenom GNU GPL licencom koji se može naći na [Seymour] odnosno na
[abcPlus] ili [sourceforge] (Win32 binaries). Pomenutu komandu koristim da bih generisani abc kod pretvorio u MIDI i pustio tako nastalu pesmu. Koristim samo deo abc 1.6 standarda po RFC-u za abc, i u budućnosti planiram da koristim deo 1.7 abc standarda koji implementira abc2midi a koji se odnosi na implicitne `%%MIDI` komande u abc kodu.

Ukratko o abc kodu koji se ovde koristi - struktura abc zapisa: zaglavlje se sastoji iz polja X:n (koje se koristi ako je u zapisu više pesama od jedne), polja T:... ili C:... za naziv pesme i kompozitora. Nakon zaglavlja mogu slediti sledeća polja bilo gde u pesmi - M:... za oznaku metrike, L:... za predefinisanu dužinu note, Q:... za tempo. Pored ovih polja mogao bih iskoristiti i P:... i V:... polja kojima se veoma jednostavno mogu definisati delovi pesme i njihovo nizanje (u zaglavlju npr. P:A.B(AB)6C.A a u telu zapisa P:A, P:B, ... dok V: daje mogućnost polifonije unutar jednog dela), kao i abc2midi proširenja oblika ,,`%%MIDI komanda`"(sa `%` počinje linija sa komentarom u pesmi) gde komanda može biti channel n ili program n (postavljanje midi kanala i programa) ili neki MIDI kontroler. Veoma je korisna opcija ,,`%% MIDI gchord niz`"koja je se već koristi po default-u (generiše akorde u zapisu prema nizu i metrici, niz se sastoji od f - osnovni (fundamentalni) stupanj, c - akord, b - ton akorda, z - pauza, sve sa mogućim dužinama, može se isključiti sa gchordoff), dok se drum komanda mora uključiti (drumon) i definisati sekvcenca (npr. `%%MIDI drum d2zdd 35 38 38 100 50 50` - d2zdd je niz kojim se definiše ritmička sekvenca, 35 i 38 su programi (GM

standard) tj. ,,note"bubnjeva a 100 50 50 je niz koji se odnosi na jačinu (velocity) udaraca d). Same note su određene visinom i visinom koja se zadaje razlomkom iza visine u trenutnoj jedinici dužine, z je pauza. Ovo je samo kratak opis koda koji koristim ili planiram da koristim, abc nudi daleko veće mogućnosti.

## 13.6 Primer - MUST.CLP

Sistem koji nizanjem unapred uz neke zadate parametre i na osnovu postojeće baze znanja generiše muziku zadatog raspoloženja ili stila. Osnovni elementi i pojmovi su:

1. **n-shema** - predstavlja zapis muzičke fraze u zadatoj rezoluciji s ocenama raspoloženja i stila. Sam zapis fraze je izražen nizom parova (ton položaj) gde je *ton* broj polustepena u odnosu na C-1 a *položaj* je broj osnovnih jedinica od početka fraze, s tim da je -101 ,,specijalni" ton - pauza. Svaka n-shema se vezuje za 3 raspoloženja i 3 stila, a jačina veze se izražava koeficijentom u istom multislotu.

2. **STANJE** - objekat koji sadrži trenutno stanje - trenutna skala, tonalitet, rezolucija, raspoloženje, stil, itd. (tempo i razmak kao opseg tonova se pomalo nedosledno drže kao posebna činjenica u bazi, ali to se lako može ispraviti)

3. **skala** - skala je niz tonova - stupnjeva skale (od mogućih 12 u jednoj oktavi) i povezuje se sa 3 stila i 3 raspoloženja

4. **tonalitet, akordi i drugi muzički pojmovi** - pod akordom se podrazumeva više tonova (u uglastim zagradama u abc notaciji) koji se reproduku u isto vreme, transponovanje je uvećanje / smanjenje za zadati broj polutonova svih nota u pesmi ili delu pesme, tonalitet bi se mogao definisati kao zadato transponovanje u visini neke note ako se uzme u obzir da su sve činjenice u bazi vezane za osnovni tonalitet 0 (iz ,,C"), oktavu čini 12 polutonova i moglo bi se reći da je *nota* klasa tonova koji su kongruentni po modulu 12, tako da svaki ton čini njegova nota i oktava. Tonalitet u muzičkoj teoriji podrazumeva i skalu i pomenutu visinu (odraz u odnosu na C), i akord se obično gradi po nekim pravilima vezanim za stupanj skale i opet ima funkciju i vezu ka raspoloženjima i stilovima.

5. **kvant, rezolucija i drugi pojmovi** - ovo su pojmovi koji nisu deo standardne muzičke teorije (mada su bitni deo MIDI formata i MIDI sveta), gde je rezolucija broj najmanjih (atomskih) vremenskih jedinica zapisa trajanja cele note a kvant je broj tih jedinica sa kojim se trenutno radi (kao L: polje u abc zapisu) i prema kome se obično vremenski poravnavaju elementi (trenutno je sistem vezan za jednu rezoluciju ali se lako može uopštiti npr. uz primenu funkcije transliraj)

6. **struktura** - trenutno je struktura određena nizom trojki `(stupanj tonalitet tip)` koji predstavljaju delove zadate dužine gde stupanj (aktuelne skale) ukazuje na akord u tom delu a tonalitet na promenu transpozicije. Tip se može iskoristiti kao veza ka n-shemama (zadnje polje id slota) npr. uvod i završnica ili neki drugi oblik klasifikacije (0 u n-shemi znači proizvoljan tip). Struktura na početku iza identifikatora ima jedan string koji ako nije prazan označava abc kod koji se ubacuje ne početku tog dela (npr. ranije pomenute komande). To je samo najgrublji opis, delovi mogu imati i drugačiji tonalitet (skalu), metriku, tempo, itd (trenutno se to postavlja odgovarajućom komandom u n-shemi).

Prvo polje id slota n-sheme i prvo polje strukture (činjenice `duzxa` i `kracya`) su jedinstveni brojevi od 1 do broja zadatog činjenicama `koliko-n-shema`, odnosno `koliko-duzxih` i `koliko-kracyih` - time se omogućava nsumično biranje jedne od tih činjenica (ili prelazak na pravilo bez n-sheme, isto nasumice) za razliku od pravila u početnoj fazi kojima se sve činjenice obrađuju (ali se bira jedna od njih na osnovu nekog pravila) - tu i vidim nekakav začetak neegzaktnog zaključivanja.

Navedene definicije muzičkih pojmova odgovaraju onima u klasičnoj muzičkoj teoriji ali su formulisani na način koji omogućava lakšu (matematičku) formalizaciju. Pravila su grupisana prema fazi izvođenja u tri faze:

**faza 0** - u ,,nultoj"fazi se najpre prikupljaju parametri ako nedostaju (raspoloženje ili stil, tempo, trajanje nije u jedinicama rezolucije već je pre približan broj nota pesme koji grubo određuje kasnije strukturu pesme) uz osnovnu proveru njihove validnosti, a onda se na osnovu činjenica u bazi formiraju osobine objekta početnog stanja i činjenice koje čine strukturu pesme. Sistem trenutno raspolaže činjenicama: ako je traženo raspoloženje srećno onda koristi dursku skalu, ako je tužno onda neku molsku, ako je besno onda smanji kvant, itd.

**faza 1** - u prvoj fazi se na osnovu činjenica (n-shema) u bazi ako i ima ili nasumice ako ih nema generiše detaljnija struktura (note) koje čine pesmu u vidu novih činjenica. Trenutno sistem raspolaže samo dvema činjenicama (što je zaista malo) i to o ,,srećnim frazama" (C, D, E, F, G, F, E, D, C i varijanta), sve ostale nasumično generiše *formiraj* funkcijom. U ovoj fazi se implicitno koristi i jedna transformacija - uzeti note do sada generisane fraze kao nekakve tonove skale i njima prepraviti naredni deo koji se dodaje (bilo da je iz n-sheme ili slučajno generisan).

**faza 2** - u drugoj fazi se transformacijama koje zavise od zadatih parametara i činjenica u bazi formira konačan oblike pesme tj. fraze. Praktično se koristi samo nasumično (po nekim pravilima) variranje fraze u zadatom broju iteracija.

Na kraju, skup pravila koji je zadužen za početni i konačan ispis pozivom funkcije daj-abc formira konačan rezultat zapisan u abc kodu i ispisuje ga na konzolu i u output.abc datoteku. Veze između činjenica su donekle već ,,labave" (od svih činjenica o skalama se bira ona koja najviše po koeficijentu odgovara raspoloženju ili stilu, a ako ih ima više iste ,,jačine" bira se neka nasumice, struktura i n-sheme se biraju nasumično) ali nije sasvim implementirano neegzaktno zaključivanje. Cilj je vremenom pre svega obogatiti bazu znanja novim pravilima o čemu sam vodio računa - npr. pravilima su opisane samo delom osobine harmonija i akorda ali nisu ni do kraja implementirane, kao ni upotreba zapisa različitih rezolucija. Trudio sam se da pratim metodologiju o projektovanju forward-chaining rule-based sistema odnosno njene delove iz Durkinove knjige [JD] shodno trenutnim potrebama razvoja ovakvog jednog sistema. Nakon potpune implementacije i testiranja bih onda prema ranije opisanoj metodologiji proširio bazu znanja npr. pravilima i složenijim akordima (septakordima pored kvintakorda), dodatao bih pravila kojima se opisuju paralelni tonaliteti, kvintni i kvartni krug, itd. Takođe, ograničenje da se prati do 3 stila i raspoloženja po skali ili n-shemi je po nekom iskustvu opravdano, ali se može uopštiti i na veći broj. Odnos akorda i drugih elemenata se zato svakako mora uopštiti (nema opravdanja ni u nekom iskustvu ograničenje na tri). Uz veoma malo truda, implementacijom delova i ritma se može postići veliki efekat, dok implementacija polifonije zahteva složeniju formalizaciju ali ne težu od postojećih. Veoma bi bilo korisno uvesti dinamiku (glasnoću, jačinu udarca), što bi bila samo jedna nova komanda i pravila o njoj. Jednostavno se može dodati stakato kao stil. Takođe, mislim da bi bilo korisno uvesti inkrementalno generisanje u kojem bi koris-

nik mogao da utiče nekim parametrima i usmerava svaki naredni rezultat do nekog konačno zadovoljavajućeg rezultata. Dokumentacija o bazi znanja se trenutno nalazi u samom kodu ovakvog sistema ali je treba izdvojiti i voditi posebno uz detaljnij obrazloženja (npr. kratko sam opisao nekoliko pravila i činjenica u bazi u vezi faza izvršenja).

Kasnije bih pokušao sa klasifikacijom zadate melodije nizanjem unazad u odnosu na zadate atributa, kao i nekim drugim (statističkim metodama, GA, i sl.) metodama. Sistem koji sam uči i prepoznaje takvo znanje bi bio neki krajnji izazov. Koje su moguće primene ovakvog sistema - kao pomoć u aranžiranju u nekom paketu za MIDI obradu i sekvenciranje, varijacije i improvizacije u realnom vremenu, ali i ,,hvatanje" nekog muzičkog korpusa kao opšteg znanja o nekoj vrsti muzike (nizanjem unapred i unazad se može potvrditi uspešnost ovakve baze znanja) ili primena u edukaciji (možda bi to bila osnova za jedan takav sistem, što bih voleo da vidim). Što većim brojem činjenica o strukturi, skalama i primerima u vidu n-shema može se dobiti baza koja bolje opisuje znanje, ali i transformacijama koje bi bolje objasnile vezu između primera i znanja (korisne bi bile npr. transformacije kombinatornog karaktera nad frazama kao nizovima). Praktično, sistem bi na kraju trebalo poboljšavati samo dodavanjem primera, skala i transformacija i tu očekujem najveće bogatstvo u opisivanju ovakvog znanja (npr. dodati modalne skale) - trenutno sistem raspolaže jako malim brojem činjenica (tek nekoliko). Neki primeri transformacija koje sam implementirao: transponovanje, vremenska translacija, nasumično menjanje ili dodavanje neke slučajno izabrane note ili nasumično oduzimanje note iz pesme (uz poštovanje pravila o redosledu, tonalitetu i sl.), itd. Takođe, abc nudi i mogućnost zapisa ponavljanja i strukture, više polifonih redova (tj. MIDI kanala) itd. što može doprineti jednostavnosti i kvalitetu rezultata.

Da bi se ovakav CLIPS kod portovao u JESS morala bi se sva pravila prepraviti da ne koriste objekat koji sam koristio da beleži trenutno stanje. Takođe, bilo bi veoma korisno implementirati vid neegzaktnog zaključivanja (što je nagovešteno u samom kodu) kojim bi se finije opisivale veze osobina i transformacija. Korisnički interfejs trenutno predstavljaju funkcije konverzije između abc i internog zapisa (n-shema), kao i poziv eksternog alata, kasnije bi moglo da se to poboljša - pre svega konverzija abc u interni zapis znanja, a mogu da zamislim i ulaz u vidu MIDI sekvenci, pretraga različitih zapisa na internetu, ulaz iz skeniranog i automatski prepoznatog notnog zapisa ili još bolje automatski prepoznat zvučni zapis (što je dovoljan problem za sebe, teži nego OCR ali postoje neki ohrabrujući rezultati i primeri). Pri-

mena takvog sisema bi onda mogla biti i neka pretraga audio materijala na internetu. Trenutno se koristi samo najosnovniji oblik abc sintakse dovoljan za reprodukciju / konverziju u MIDI bez dodatnih elemenata i proširenja. Nije teško zamisliti ovakav sistem onda kao školjku u kojoj bi se mogla dalje razvijati ovakva vrsta znanja kada bi se sistem usavršio na opisan način.

Ovaj sistem ima i jednu osobinu koja mu omogućava jedan osnovni oblik učenja - na kraju svake sesije se nudi mogućnost da se generisana fraza zapamti kao nova činjenica u bazi sa nekim zadatim raspoloženjem i njegovim stepenom. Stanje se uvek može snimiti `save` komandom.

Primer se može učitati `(load must.clp)` komandom ili iz GUI menija, zatim se resetuje sistem sa `(reset)` i pokrene se sa `(run)`. Nakon generisanja se pokreće pesma, ali da bi to radilo u direktorijumu odakle se pokreće CLIPS primer mora da stoji abc2midi.exe, mora da bude podešen ispravno MIDI izlaz (npr. u Control Panel-u se pokrene aplet Sounds and Audio Devices, zatim na Audio tabu se podesi MIDI music playback Default device npr. kao Microsoft GS Wavetable SW Synth ili šta već nudi zvučna kartica) i MID ekstenzija mora da bude asocirana npr. sa Windows Media Player-om (ili bilo kojim sposobnim da reprodukuje MIDI). Ovakav sistem je lako portovati na neki drugi OS, npr. Unix/Linux - jedino se reprodukcija drugačije realizuje (npr. upotrebom playabc ili bilo kog *i*x MIDI playera).

Primer jedne sesije (prototipa `must.clp` ovakvog jednog sistema):

```
CLIPS> (load must.clp)

CLIPS> (reset)

CLIPS> (run)

Muzicki sistem za generisanje fraza

Kakvo se raspolozenje trazi (srecno tuzno besno napeto blues) ? srecno
Koliko dugacka fraza (u celim) ? 24

Predlozena fraza:

X:1
T:Title
```

```
M:none
C:Composer
K:C
Q:1/4=100
L:1/4
M:4/4
=E1/2 =G1/2 =B1/2 =c =E1/2 =D1/2
Q:1/4=100
L:1/4
M:4/4
=C1/2-| =C1/2 =C3/2 =B1/2 =c =E1/2-| =E1/2 =C
Q:1/4=100
L:1/4
M:4/4
z2 "G/F"=C =G =A1/2-| =A1/2 =E1/2 =D1/2
Q:1/4=100
L:1/4
M:4/4
=C =C1/2 =E1/2 =f''1/2-| =f''1/2 [=c =A ]=E1/2 =D1/2 =C12 =F,4 | z
% (n-shema (id 3 0) (shema -105 100 -104 4 -103 4 -102 4 4 7 7 9 11 11 12 13
4 17 2 19 0 21 -105 100 -104 4 -103 4 -102 4 0 25 11 31 12 33 4 37 0 41 -101
37 -105 100 -104 4 -103 4 -102 4 -106 89 0 45 7 49 9 53 4 57 2 59 0 61 -105
100 -104 4 -103 4 -102 4 0 65 4 67 41 69 12 73 9 73 4 77 2 79 0 81 -7 33)
(rezolucija 16))
Da li da ubacim dobijeno u bazu (Da/Ne) ?d
U koje raspolozenje ova fraza najviv{s}e spada (srecno tuzno besno napeto
blues) ? srecno
Koliko je to tako - izra\v{z}eno brojem izme\d{d}u 0 i 1 ? 1
Slede'{c}a sesija se pokre'{c}e sa (run) ...
CLIPS> (facts)
f-0     (initial-fact)
f-1     (razmak -12 24)
f-2     (skala (note 0 2 4 5 7 9 11) (a1 dur 0 0) (a2 dur7 7 7) (a3 dur 5 5)
(r1 srecno 1) (r2) (r3) (s1 dur 1) (s2) (s3))
f-3     (skala (note 0 2 3 5 7 8 10) (a1 mol 0 0) (a2 mol 7 7) (a3 mol 5 5)
(r1 tuzno 1) (r2) (r3) (s1 mol 1) (s2) (s3))
f-4     (skala (note 0 2 3 5 7 9 11) (a1 mol 0 0) (a2 dur7 7 7) (a3 dur 5 5)
(r1 tuzno 1) (r2) (r3) (s1 mol-prirodni 1) (s2) (s3))
f-5     (skala (note 0 2 3 5 7 8 11) (a1 mol 0 0) (a2 dur7 7 7) (a3 mol 5 5)
(r1 tuzno 1) (r2) (r3) (s1 mol-harmonski 1) (s2) (s3))
```

```
f-6     (skala (note 0 3 5 6 7 10) (a1 dur7 0 0) (a2 dur7 7 7) (a3 dur7 5 5)
(r1 tuzno 0.5) (r2 srecno 0.5) (r3) (s1 blues 1) (s2) (s3))
f-7     (skala (note 0 3 5 6 7 10) (a1 mol7 0 0) (a2 mol7 7 7) (a3 mol7 5 5)
(r1 tuzno 0.5) (r2 srecno 0.5) (r3) (s1 blues 1) (s2) (s3))
f-8     (skala (note 0 3 4 5 7 10) (a1 dur7 0 0) (a2 dur7 7 7) (a3 dur7 5 5)
(r1 tuzno 0.4) (r2 srecno 0.6) (r3) (s1 blues 1) (s2) (s3))
f-9     (skala (note 0 2 4 6 8 10) (a1 aug 0 0) (a2 aug 5 4) (a3 aug 7 8)
(r1 napeto 0.8) (r2 besno 0.5) (r3 srecno 0.5) (s1) (s2) (s3))
f-10    (skala (note 0 1 2 3 4 5 6 7 8 9 10 11) (a1 dim 0 1) (a2 maj9 5 6)
(a3 aug 7 2) (r1 besno 0.9) (r2 napeto 0.6) (r3 blues 0.5) (s1 hromatska 1)
(s2) (s3))
f-11    (skala (note 0 1 2 3 4 5 6 7 8 9 10 11) (a1 aug 0 1) (a2 aug 5 4)
(a3 aug 7 2) (r1 napeto 0.9) (r2 besno 0.6) (r3 blues 0.5) (s1 hromatska 1)
(s2) (s3))
f-12    (n-shema (id 1 0) (rezolucija 16) (shema 0 1 2 3 4 5 5 7 7 9 5 11 4
13 2 15 -1 17)
(komanda "") (a1) (a2) (a3) (r1 srecno 1) (r2) (r3) (s1) (s2) (s3))
f-13    (n-shema (id 2 0) (rezolucija 16) (shema 0 1 4 3 7 5 11 7 12 9 9 9 4
13 2 15 0 17)
 (komanda "") (a1) (a2) (a3) (r1 srecno 1) (r2) (r3) (s1) (s2) (s3))
f-15    (tempo 100)
f-16    (duzxa 1 "" 4 4 1 0 0 4 0 0 1 0 0 5 0 0)
f-17    (duzxa 2 "" 4 4 1 0 0 1 0 0 5 0 0 4 0 0)
f-18    (kracya 1 "" 4 4 1 0 0)
f-19    (kracya 2 "" 3 4 1 0 0)
f-20    (koliko-duzxih 2)
f-21    (koliko-kracyih 2)
f-22    (osobine (lista srecno tuzno besno napeto blues) (faktori))
f-42    (n-shema (id 3 0) (rezolucija 16) (shema -105 100 -104 4 -103 4
-102 4 4 7 7 9 11 11 12 13 4 17 2 19 0 21 -105 100 -104 4 -103 4 -102 4 0
25 11 31 12 33 4 37 0 41 -101 37 -105 100 -104 4 -103 4 -102 4 -106 89 0
45 7 49 9 53 4 57 2 59 0 61 -105 100 -104 4 -103 4 -102 4 0 65 4 67 41 69
12 73 9 73 4 77 2 79 0 81 -7 33) (komanda) (a1) (a2) (a3) (r1 srecno 1)
(r2) (r3) (s1) (s2) (s3))
f-43    (r1 srecno tuzno besno napeto blues)
f-44    (rezolucija 16)
f-45    (koliko-n-shema 3)
For a total of 26 facts.
CLIPS> (dribble-off)
```

Knjige korišćene tokom pisanja ovog rada, kao i sajtovi sa dokumentacijom -

# Literatura

[JD] John Durkin: Expert Systems - Design and Development

[JL] Jean-Louis Lauriere: Problem-Solving and Artificial Intelligence

[GN] Michael R. Genesereth and Nils J. Nilsson: Logical Foundations of Artificial Intelligence

[TB] Donald E. Knuth: The TeXbook

[PG] Predrag Janičić, Goran Nenadić: OSNOVI LaTeX-A

[CLIPS] `http://www.ghg.net/clips/CLIPS.html`

[JESS] `http://herzberg.ca.sandia.gov/jess/`

[PROTEGE] `http://protege.stanford.edu`

[CW] `http://staffweb.cms.gre.ac.uk/~c.walshaw/abc/`

[NB] `http://www.norbeck.nu/abc/abcbnf.htm`

[ABC] `http://ecf-guest.mit.edu/~jc/music/abc/ABC-FAQ.html`

[PLAY] `http://www.geocities.com/Nashville/8773/abcplay.htm`
`http://abc.sourceforge.net/abcMIDI`
`http://abcplus.sourceforge.net/`

[Seymour] `http://ifdo.pugmarks.com/~seymour/runabc/top.html`

[abcPlus] `http://abcplus.sourceforge.net`

[sourceforge] `http://abc.sourceforge.net/abcMIDI`