Univerzitet u Beogradu
Matematički fakultet

„Mašina pravila” ekspertski sistem dokazivanja zasnovan na pravilima

master rad

Student:
Miloš Radosavljević 1024/2011

Mentor:
prof. dr Miroslav Marić

septembar, 2016.

Mentor:	prof. dr Miroslav Marić Matematički fakultet, Univerzitet u Beogradu
Članovi komisije:	prof. dr Filip Marić Matematički fakultet, Univerzitet u Beogradu
	doc. dr Mladen Nikolić Matematički fakultet, Univerzitet u Beogradu
Datum odbrane:	_____

Sažetak: U ovom master radu biće objašnjeno kako funkcionišu ekspertski sistemi zasnovani na pravilima, kako se može izgraditi jedan ekspertski sistem i koji su učesnici u procesu izgradnje ekspertskog sistema. Biće prikazani alati koji su zasnovani na mašini pravila i koji se koriste u mikrobiološkoj laboratoriji za validaciju podataka, otkrivanje i praćenje bakterija. Centralna tema ovog rada biće ekspertski sistem zasnovan na pravilima, način na koji se zapisuju pravila i kako se dalje primenjuju za dobijanje željenog rezultata. Kada izabrati takav sistem, a kada izbeći. Biće definisana osnovna svojstva ekspertskih sistema, način predstavljanja znanja i metode zaključivanja u ekspertskim sistemima.

Klučne reči: mašina pravila, produkciona mašina pravila, ekspertski sistemi, predstavljanje znanja, mehanizmi zaključivanja, ulančavanje unapred, ulančavanje unazad, mašine pravila u laboratorijskim informacionim sistemima, mašina pravila u mikrobiologiji.

Abstract: In this master degree work will be explain rule based expert system, how to build expert system and define participants. Tools which are designed on rule engine system and used in microbiological laboratory for data validation and bacteria infections monitoring, will be presented. The central theme of this work will be rule-based expert system, type of rule representation and how executing of this rule lead to a desired solution. Whether is necessary to use this system or not. Basic properties of expert systems, knowledge representation and type of inferencing and reasoning in expert systems will be shown.

Keywoards: rules engine, production rules engine, expert system, knowledge representation, inference, forward chaining, backward chaining rule engine in laboratory information systems, rule engine in microbiology.

Sadržaj:

1 Uvod

Savremene aplikacije, sistemi, softverski servisi i moduli razvijaju se velikom brzinom. Problemi prilikom održavanja ovakvih sistema sve su veći. Vreme „odsustva" neke aplikacije ili sistema i nepristupačnost korisnicima za rad pri instalaciji, unapređenju i održavanju sistema potrebno je učiniti veoma malim (gotovo da ne postoji). Dodavanje poslovne logike koja ne utiče na funkcionalne delove koda, već proširuje sistem i njegovu bazu, ne bi trebalo da izazove nedostupnost sistema (aplikacije) u tim trenucima. Pored navedenih činjenica, treba dodati i otpornost sistema na greške kao jednu od osnovnih osobina odličnih, profitabilnih i pouzdanih sistema. Najveći faktor za pojavu greške jeste sam čovek usled čijeg umora, nepažnje, nedovoljne stručnosti i sličnih uzoraka, može doći do ozbiljnih propusta. Ovakve greške mogu uticati na rad sistema, ali i na opasnost po podatke i druge korisnike koji direktno ili posredno zavise od stabilnosti sistema (aplikacije). Usled toga poželjno je unaprediti i poboljšati implementaciju i pristup prilikom organizacije sistema i aplikacija, uvesti automatizaciju delova koji se ponavljaju, a uticaj ljudskog faktor svesti na minimum.

Potrebno je projektovati sistem koji bi na osnovu definisanih pravila obrađivao podatke i izvršavao naredne korake. Svaki sledeći korak bi zavisio od primenjenog pravila na odgovarajuće podatke, prateći algoritam izvršavanja, dok je čoveka potrebno usmeravati na kontrolu pojedinih delova sistema i ažuriranju baze podataka, odnosno baze potrebnih pravila. Ovako opisan proces najjednostavnije se može implementirati korišćenjem ekspertskih sistema zasnovanih na pravilima. Zbog svoje pouzdanosti, skalabilnosti, brzine itd. ovi sistemi sve više nalaze primenu u savremenim poslovnim aplikacijama, bankarskim sistemima, sistemima za bezbednost i zdravstvenim sistemima. Ovakav jedan sistem biće predstavljen u daljem tekstu. Detaljno će biti opisana arhitektura ovakvih sistema, tipovi sistema, način funkcionisanja, kao i njihova primena u laboratorijskom informacionom sistemu.

2 Kratak osvrt na veštačku inteligenciju

Veštačka inteligencija kao jedna od oblasti računarstva, čiji je cilj da razvije mašinu koja „razmišlja nalik ljudskom biću", sa širokom primenom i otvorenim putem za dalji razvoj, ostvaruje sve veće doprinose u svetu nauke i tehnologije.

Prvi značajni radovi vezani za matematičku logiku, sistem formalne logike i teoriju grafova pojavljuju se u XV i XVI veku. Značajnu ulogu u razvoju mašinske (računarske) inteligencije ima Alan Tjuring[1]. Na temu veštačke inteligencije objavljuje rad „Računarske mašine i inteligencija" i definiše test za „merenje inteligencije". Postavlja prve korake u razvoju „inteligentnih" sistema. Tjuringova mašina i danas predstavlja jedan od standarda u definisanju domena kompleksnih problema.

Izraz „veštačka inteligencija", uvodi se sredinom 50-ih godina. Smatra se da je izraz prvi upotrebio i definisao Džon Makarti[2] u leto 1956. godine na sastanku tadašnjih pet vodećih naučnika iz oblasti računarskih nauka na *Dartmouth Collegu* (Hanover, USA). Na ovom zasedanju postavljeni su temelji za razvoj veštačke inteligencije.

[1] Alan Tjuring (en. *Alan Turing*) [23. jun 1912 - 7. jun 1954] - engleski matematičar, logičar i kriptograf. Smatra se ocem modernog računarstva. Razvio je Tjuringovu mašinu, formulisao Čerč - Tjuringovu tezu, definisao više tehnika za razbijanje šifara (učestvovao u dekriptovanju Enigme).

[2] Džon Mekarti (en. John McCarthy) [4. septembra 1927. - 23. oktobar 2011] - američki naučnik koji je 1971. dobio Tjuringovu nagradu zbog doprinosa na polju veštačke inteligencije. Projektovao je programski jezik Lisp.

Iako nijedna definicija inteligencije i inteligentnih sistema nije zvanično prihvaćena, dosta relevantnom se može smatrati sledeća definicija: inteligencija je sposobnost sistema da se prilagodi promenama u svetu i što je ta sposobnost veća, odnosno što je prefinjenija snaga prilagođavanja, sistem je inteligentniji [2]. Još jedna „klasična" definicija veštačke inteligencije koja se smatra prihvatljivom, glasi: ako su u dve odvojene prostorije smeštene jedna ljudska osoba i neka naprava i ako na identične probleme one pružaju odgovore na osnovu kojih se ne može pogoditi (značajno efikasnije od nasumičnog pogađanja) u kojoj sobi je čovek, a u kojoj naprava, onda možemo smatrati da ta naprava ima atribute veštačke inteligencije [1]. Pod „inteligentnim" sistemima mogu se smatrati sistemi koji imaju sposobnost pamćenja (skladištenja) znanja i mogućnost njegove obrade, odnosno reprezentacije. Sistem se ne može smatrati inteligentnim bez sposobnosti da te podatke obrađuje i reprezentuje. Pored navedenih osobina, jako bitna karakteristika inteligentnih sistema je i brzina obrade znanja, kao i brzina učenja, tj. sticanja novog znanja. Mašine i sistemi koji poseduju ovakav oblik znanja, tj. inteligencije spadaju u domen veštačke inteligencije.

U prvim godinama razvoja veštačke inteligencije nastaju i prvi „inteligentni" sistemi koji se bave teorijom igara, uspešno igraju šah, nastaju prvi roboti i automatski dokazivači teorema. Nešto kasnije pojavljuju se prvi sistemi koji se uspešno koriste u vojnoj i kosmičkoj industriji za testiranje pilota, razvijaju se sistemi za prepoznavanje govora, kao i sistemi za kontrolu saobraćaja. Poslednjih godina funkcionisanje uređaja i aplikacija koje se svakodnevno koriste bilo bi gotovo nemoguće bez prisustva veštačke inteligencije. Dalji razvoj veštačke inteligencije omogućio je definisanje različitih inteligentnih sistema i podoblasti veštačke inteligencije, kao što su: neuronske mreže, genetski algoritmi, stabla odlučivanja, ekspertski sistemi. Ekspertski sistemi kao jedna značajna oblast veštačke inteligencije biće detaljnije predstavljeni u daljem tekstu.

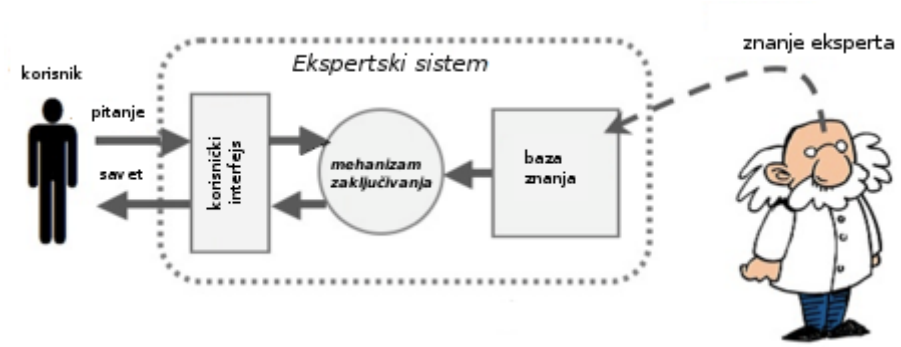
3 Ekspertski sistemi

Ekspertski sistem je inteligentni računarski program koji koristi znanje i postupke zaključivanja u procesu rešavanja problema za koje je potreban visok stepen stručnosti i iskustva iz domena kome se ekspertski sistem obraća. Nastali su u okviru veštačke inteligencije. Cilj je da programi dejstvuju inteligentno, a ne mehanički. Prva istraživanja vezana za ekspertske sisteme javljaju se 1960-ih godina, a prvi ekspertski sistemi nastaju početkom 1970-ih godina i to uglavnom u okviru najvećih i najpoznatijih univerzitetskih ustanova. Neki od ekspertskih sistema iz tog perioda su: Dendral, Age, Mycin, Prospector, Caduceus, WM, Baobab, Clot, Oncocin itd.

Proteklih decenija razvijeno je hiljade ekspertskih sistema za rešavanje problema iz različitih domena primene. Prema podacima iz 1988, ekspertski sistemi primenjuju se u oko 150 oblasti, da bi se nakon samo četiri godine 1992. broj ekspertskih sistema znatno povećao. U Americi je bilo najmanje 3000 ekspetskih sistema, u Japanu preko 400, u Evropi, takođe veliki broj [8]. Danas se praktično i ne zna tačan broj ekspertskih sistema. Područja primene ekspertskih sistema su velika, počevši od medicine, Veb-a, transporta (saobraćaj), finansija i bankarstva, do vojske, geologije i rudarstva, trgovine, kao i u mnogim drugim oblastima.

— prof. dr V. Milačić, 1990.

„Pod ekspertskim sistemom se podrazumeva uspostavljanje unutar računara dela veštine nekog eksperta koja se bazira na znanju i u takvom je obliku da sistem (računar) može da ponudi *inteligentan savet* ili da preduzme *inteligentnu odluku* o funkciji koja je u postupku. Poželjna dopunska karakteristika, koju mnogi smatraju osnovnom, jeste sposobnost sistema da na zahtev *verifikuje svoju liniju rezonovanja*, tako da direktno obaveštava onoga (korisnika) koji postavlja pitanje. Usvojeni način da se ostvare ove karakteristike je *programiranje na bazi pravila*."



Slika 1: Ilustracija ekspertskog sistema

Ekspertske sisteme (slično kao i inteligentne sisteme) karakterišu sledeća svojstva: mora da sadrže znanje (iz konkretne oblasti - domena), da vrše zaključivanje, prosuđivanje i odlučivanje na osnovu nepouzdanih i nepotpunih informacija i da omoguće tumačenje svog ponašanja. Upravo ova poslednja osobina odvaja ekspertske sisteme od svih ostalih „neinteligentnih" sistema. Idealni ekspertski sistem treba da poseduje sledeće osobine:

- velika specifična znanja iz oblasti (domena);
- primena tehnike pretraživanja;
- podrška heurističkoj analizi;
- sposobnost sticanja novih znanja iz postojećih;
- sposobnost pojašnjenja sopstvenih zaključaka.

Uopšteno posmatrajući svi ekspertski sistemi mogu se podeliti u dve grupe: one koji *analiziraju* neki problem i one koji vrše sintezu sa ciljem *rešavanja* nekog problema. Pored navedene globalne podele, svi ekspertski sistemi takođe se mogu kvalifikovati i u neki od sledećih tipova:

- ekspertski sistemi zasnovini na pravilima (en. *Rule-based systems*);
- ekspertski sistemi zasnovani na okvirima (en. *Frame-based systems*);
- hibridni ekspertski sistemi (en. *Hybrid systems*);
- ekspertski sistemi zasnovani na modelima (en. *Model systems*);

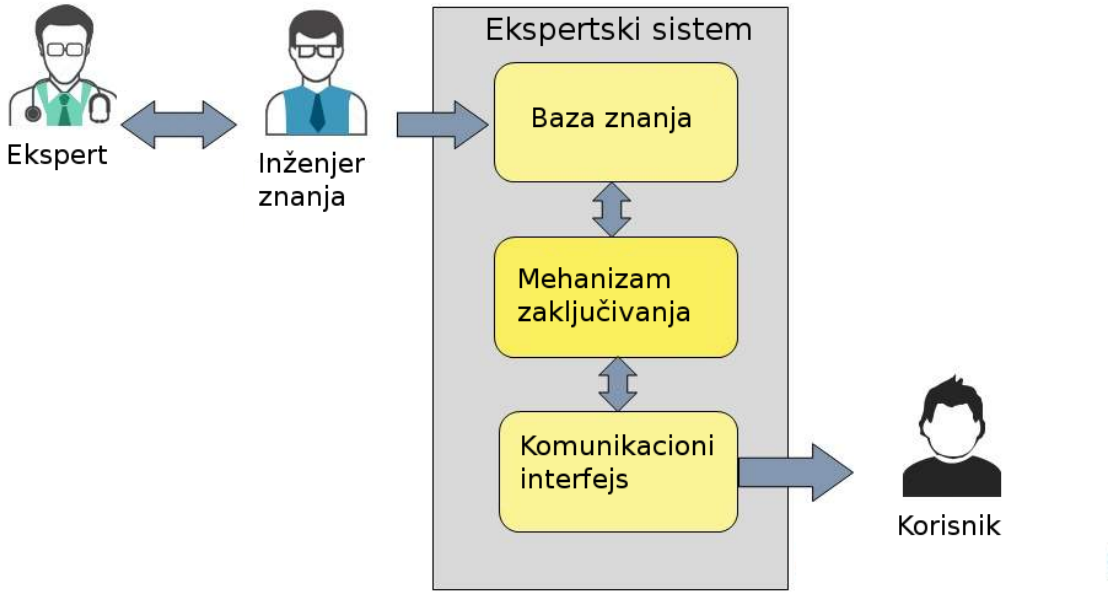
- ekspertski sistemi za sopstvenu upotrebu (en. *Ready-made systems (Off-the-shelf)*);
- ekspertski sistemi za rad u realnom vremenu (en. *Real-time systems*).

Osim nabrojanih tipova ekspertskih sistema [6], postoji još nekoliko grupacija, tj. podela ekspertskih sistema na osnovu domena problema, krajnjih korisnika, eksperata, na osnovu vrsta informacija koje sistem pruža itd., ali ovde neće biti navedeni.

Ekspertski sistemi se sastoje iz više delova: dela za rešavanje problema (baza znanja, mehanizam zaključivanja i globalna baza podataka), o čemu će u daljem tekstu biti više reči, i okruženja - korisnički interfejs (sredstva za otkrivanje grešaka u razvoju sistema, grafički prikaz rezultata, korisnička podrška, upustva).

3.1 Izgradnja ekspertskih sistema

Proces implementacije i izgradnje ekspertskog sistema naziva se *iznženjerstvo znanja*. Obuhvata skup metoda, pravila i postupaka kao što su prikupljanje, predstavljanje i memorisanje znanja, ali i upotrebu ljudskih resursa za rešavanje složenih problemskih situacija. Značajan deo u impelmentaciji ekspertskih sistema je upravo proces interakcije između graditelja sistema (inženjer znanja) i jedne ili više osoba koje su eksperti u određenoj oblasti za koju se sistem izrađuje. Inženjer znanja od eksperta vrši ekstrakciju njegovih procedura, strategija i postupaka za rešavanje problema i ugrađuje to znanje u ekspertski sistem. Rezultat procesa je skup programa i pravila koji zadovoljavaju (rešavaju) neki problem na način kako to radi čovek - ekspert. Slika 2 prikazuje učesnike u procesu izgradnje ekspertskog sistema i obuhvata: eksperta, inženjera znanja, korisnika i sam ekspertski sistem.



Slika 2: Učesnuci u izgradnji ekspertskog sistema

Ekspert je osoba (ne nužno jedna) koja je stekla reputaciju u svojoj oblasti zbog stručnih sposobnosti načina rešavanja problema. Koristeći svoje znanje, sposobnost, metode i veštine stečene kroz bogato iskustvo, on omogućava da se skrati proces pronalaženja rešenja. Pored znanja, eksperta karakteriše i intuicija koja ga vodi ka rešavanju nekog problema. Osnovne osobine koje ekspert treba da poseduje jesu: da prepozna i formalizuje problem, da reši problem brzo i uspešno, objasni način rešavanja, uči na osnovu iskustva, da ima mogućnost restrukturiranja znanja, kao i određivanje relevantnosti rešenja. Ekspertski sistem treba da obuhvati i objedini te sposobnosti, veštine i iskustvo jednog ili više eksperata.

Inženjer znanja je osoba koja poznaje oblast računarskih nauka, veštačke inteligencije, projektovanja softvera i koja zna kako se izgrađuje ekspertski sistem. On kroz pitanja i razgovor sa ekspertom prikuplja znanje, organizuje podatke i odlučuje kako će znanje i podaci biti reprezentovani u sistemu.

Korisnik je osoba koja koristi ekspertski sistem.

Pored prikazanih učesnika u izgradnji ekspertskih sistema (Slika 2), mogu se navesti još dva elementa koji učestvuju u procesu izgradnje ekspertskih sistema, ali nisu od velikog značaja (nisu prikazani na slici). To su:

- *alat za izgradnju ekspertskih sistema* je program ili skup programa koji se koristi u izgradnji ekspertskih sistema;
- *osoblje* uključuje sve one koji unose i održavaju podatke.

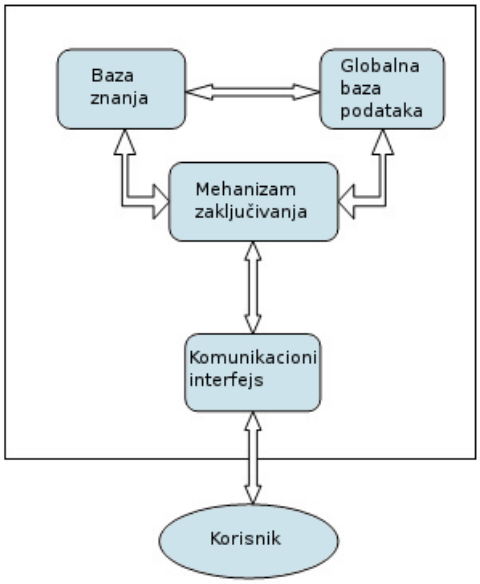
3.2 Komponente ekspertskih sistema

Ekspertski sistem treba da razlikuje tri glavna zadatka inženjerstva znanja [3]:

- prezentacija i memorisanje velike količine znanja iz problemske oblasti;
- aktiviranje prikupljenog znanja iz oblasti nekog domena za rešavanje problema;
- odgovor na korisnikovo pitanje.

Kao što se već moglo videti iz procesa izgradnje ekspertskih sistema (Slika 2), pored učesnika u izgradnji, sam ekspertski sistem sačinjavaju nekoliko komponenti. Osnovne komponente ekspertskih sistema su (Slika 3):

- baza znanja (en. *knowledge base*) i globalna baza podataka (činjenice);
- mehanizam zaključivanja (en. *inference engine*);
- komunikacioni interfejs.



Slika 3: Komponente ekspertskog sistema

- *Baza znanja* - baza činjenica i heuristika za koje je namenjen ekspertski sistem. Uključuje činjenice, relacije između činjenica i moguće metode za rešavanje problema u oblasti date aplikacije;
- *Mehanizam zaključivanja* - deo sistema koji je sposoban da na osnovu informacija iz baze znanja izvede zaključke. On radi tako što činjenice iz baze znanja kombinuje sa informacijama i podacima dobijenim od strane korisnika i sa podacima iz globalne baze podataka. Pri radu se koriste kontrolne strategije koje odlučuju u kom trenutku treba primeniti neko pravilo iz baze znanja na nove činjenice. Na ovaj način se simulira ljudsko razmišljanje. U daljem tekstu će biti više reči o principu rada ovog dela sistema;
- *Komunikacioni interfejs* - deo koji omogućava interakciju između korisnika i ekspertskog sistema;
- *Globalna baza podataka* - radna memorija za beleženje trenutnih statusa sistema, ulaznih podataka za određeni problem. Čuva činjenice i zaključke dobijene tokom rada sistema. Razlikuje se od baze znanja po tome što sadrži informacije koje se odnose na tekući problem odlučivanja.

3.3 Svojstva ekspertskih sistema

Osnovno svojstvo i srž svakog ekspertskog sistema je „znanje" akumulirano u procesu izgradnje i u kasnijem radu sistema. Može se podeliti na činjenično i heurističko znanje.

Činjenice predstavljaju glavni deo podataka o prirodi sistema, njegovim aktivnostima i ciljevima koje sistem ostvaruje kroz te aktivnosti. Sistem na osnovu činjenica koje su dostupne u nekom trenutku i podataka koji se nalaze u bazi znanja vrši zaključivanje i na taj način pokušava da pronađe najbolje moguće rešenje. Svi ovi podaci mogu biti raspoloživi, dokumentovani i verifikovani u domenu ekspertskog sistema, predstavljeni na određeni način i pomoću određene tehnike.

Heurističko znanje je iskustvo i osećaj za izbor rešenja; znanje dobre prakse, dobrog rasuđivanja i razumevanja na terenu. To je znanje koje podupire „umetnost dobrog nagađanja" [10]. Čine ga pravila rasuđivanja i veština u izboru i donošenju odluka kojima se utiče na promenu stanja sistema. Cilj je minimizovati (smanjiti) skup pravila ili postupaka koji daju zadovoljavajuće rešenje za specifičan problem. Obično se ostvaruje uz pomoć dodatne komunikacije sa korisnicima sistema. Nivo performansi ekspertskog sistema pre svega jeste funkcija veličine i kvaliteta baze znanja u kojoj su objedinjene činjenice i heuristika, a ne određenog formalizma zaključivanja i postupka koji se koristi u pretraživanju činjenica.

Prema suštinskom značaju i upotrebi u ekspertskim sistemima postoje sledeći tipovi znanja:

- proceduralno znanje - znanje o tome kako se neki problem rešava;
- deklarativno znanje - može se posmatrati kao skup kratkih iskaza koji su tačni ili netačni i odnose se samo na određeni problem (činjenice, ono šta se zna);
- meta znanje - znanje o znanju (znanje kako upotrebiti i upravljati znanjem);
- strukturno znanje - opisuje domenske koncepte, objekte i njihove atribute i veze (mentalni model eksperta);
- eksplicitno znanje - znanje dato u pisanoj ili drugoj prenosnoj formi i može se naći u knjigama, časopisima i sl. Ovo znanje je obično prihvaćeno kao univerzalno tačno;
- implicitno znanje - znanje koje čovek ekspert gradi na osnovu iskustva i kombinovano sa eksplicitnim tipom znanja, čini čoveka ekspertom. Znanje je dostupno i može se prenositi;
- heurističko znanje - iskustvena pravila ili situacije koje eksperti koriste i znaju.

Važno svojstvo ekspertskih sistema je *ekspertiza visokog nivoa*. Opisuje najbolja rešenja i razmišljanja vrhunskih eksperata u datoj oblasti. Ekspertiza visokog nivoa sakupljena je i ugrađena u sistem tako da u procesu rešavanja problema omogućava dolaženje do najpreciznijih i najefikasnijih rešenja. Pored ekspertize visokog nivoa važno svojstvo ekspertskih sistema je mogućnost predviđanja. Proističe iz osobine da se ekspertski sistemi mogu koristiti kao model za rešavanje problema u određenoj oblasti. Daje odgovore za zadate probleme i prikazuje kako se odgovori mogu menjati u zavisnosti od novih situacija. Celokupno znanje ugrađeno u ekspertski sistem prikupljeno je kroz interakciju sa ključnim osobljem u nekoj službi, odeljenju ili oblasti, ali i kroz korišćenje sistema u samom procesu rada. Na taj način, ova kolekcija znanja postaje trajni zapis usklađenih najboljih metoda i postupaka koje ljudi koriste i dalje usavršavaju kako bi ubrzali proces pri rešavanju određenog problema. Međutim, gomilanjem nepotrebnog znanja dolazi do lošeg i sporog funkcionisanja samog sistema. Ovo je veoma važno u poslovnim sistemima, ali o tome će biti reči nešto kasnije.

4 Predstavljanje znanja

Deo sistema koji odvaja i čini ekspertske sisteme superiornim u odnosu na obične konvencionalne programe, sisteme i algoritme, jeste mogućnost „intuitivnog" zaključivanja i mogućnost sticanja novog znanja (učenje), odnosno ponovne reprodukcije stečenog znanja. Izbor same reprezentacije znanja jedan je od ključnih problema i zavisi od mnogo faktora. U direktnoj vezi je kako sa prirodom određenog znanja, tako i sa izborom mehanizama za zaključivanje, kao i domenom problema za koji se implementira ekspertski sistem. Pošto mehanizmi za zaključivanje moraju biti prilagođeni reprezentaciji znanja i njegovoj prirodi, u jednom slučaju bivaju zasnovani na klasičnoj logici, a u drugom na modalnoj logici, teoriji verovatnoće, pravilima itd. Osnovni kriterijumi za predstavljanje znanja su:

- transparentnost - mera u kojoj se može identifikovati smešteno znanje;
- eksplicitnost - mera direktnog predstavljanja znanja;
- prirodnost - mera prirodne predstave znanja (znanja u njegovoj prirodnoj formi);
- efikasnost - mera u kojoj se zadata struktura može iskoristiti za predstavu celokupnog znanja ekspertskog sistema;
- modularnost - mera u kojoj se delovi znanja mogu skladištiti nezavisno jedno od drugog.

4.1 Predstavljanje znanja u ekspertskim sistemima

Osnovna namena reprezentacije znanja (*en. Knowledge representation*) jeste izbor određenog „tipa" znanja, odnosno predstavljanje znanja (eksperta sistema) u simboličkom obliku. Obično se za jedan ekspertski sistem uzima dobro usredsređena oblast znanja koja je vezana za jednu specifičnu temu, a ne više njih (npr. tema „zarazne bolesti krvi"). Tako da se predstavljanje (reprezentovanje) znanja može definisati i kao metod *kodiranja znanja iz određenog domena u bazi znanja nekog ekspertskog sistema.* Predstavljanje znanja obuhvata:

- strukturu (model) kojom se opisuju elementi znanja;
- interpretativni proces koji je neophodan u korišćenju znanja.

Ne postoji idealan tip predstavljanja znanja koji može da pokrije i okarakteriše sve delove nekog ekspertskog sistema. Ljudsko znanje se sastoji iz pojedinih činjenica kao osnovnih elemenata. U mnogim formalnim sistemima pod činjenicom, kao oblikom deklarativnog znanja, obično se podrazumeva iskaz - rečenica koja ima dve moguće vrednosti, tačna ili nije tačna. Međutim, u praksi često nemamo precizno definisane granice. Vrlo lako može se pojaviti delimična tačnost nekog iskaza ili delimična netačnost. Takođe, veoma često javlja se potreba da se znanje prikaže u nekom grafičkom obliku (intuitivnije ljudskom shvatanju).

Osnovne tehnike (modeli) za predstavljanje znanja u ekspertskim sistemima su: *pravila*, *mreže zaključivanja*, *semantičke mreže*, *okviri*, *O-A-V trojke*, *objekti*, *iskazna logika i logika prvog reda.*

4.1.1 Pravila

Pravila (en. *Rules*) su jedna od najčešće korišćenih metoda za predstavljanje znanja u okviru ekspertskih sistema. Vrlo su čitljiva i daju jasnu sliku o prirodi problema. Najveći broj ekspertskih sistema zasnovan je na pravilima ili bar jednim delom koristi pravila.

Pravilo je struktura za predstavljanje znanja koja uspostavlja vezu između neke poznate informacije i informacije za koju se na osnovu tačnosti prve može zaključiti da i ona takođe važi. Teorijska zasnovanost ove metode počiva na iskaznoj logici i na njenom proširenju predikatskim računom. Pravila predstavljaju pojednostavljenu računarsku implementaciju predikatskog računa, a jedna od osnovnih tehnika zaključivanja je modus ponens. Proces zaključivanja zasnovan je na matematičkoj logici sa levom i desnom stranom (uslov - zaključak), pri čemu desna strana može osim nove činjenice značiti i izvršenje neke procedure. Takođe, može sadržati i meta pravila, odnosno pravila koja se koriste da bi se opisalo saznanje o upotrebi drugog pravila.

```
IF auto neće da upali AND farovi ne rade
THEN akumulator je prazan
```

Pravila se sastoje iz dva dela: *„ako"* (en. *IF*) i *„onda"* (en. *THEN*). U *IF* delu nalaze se jedan ili više iskaza na osnovu kojih se izvode zaključci. Ovi iskazi se nazivaju *premise*. Ukoliko pravilo sadrži više premisa onda su one povezane nekim logičkim veznicima (i, ili). *THEN* deo pravila takođe sadrži iskaze, ali se oni zovu *zaključci*. Princip funkcionisanja je identičan modus ponensu, ako su premise iz *IF* dela pravila tačne onda su i zaključci iz *THEN* dela tačni.

Sa povećanjem broja pravila raste i složenost sistema. Potrebno je da pravila budu jasno definisana, ažurna i da se izbegava njihovo dupliranje. Zaključci jednog pravila najčešće predstavljaju premise drugih pravila. Ovaj princip nadovezivanja pravila (zaključak jednog pravila je premisa drugog) naziva se ulančavanje pravila (en. *rule chaining*).

```
IF auto neće da upali AND farovi ne rade
THEN akumulator je prazan

IF akumulator je prazan
THEN napuni akumulator
```

Pravila se koriste za opisivanje proceduralnog znanja jer ukazuju na način rešavanja nekog problema i izvođenja zaključaka o tom problemu. Ekspertski sistemi koji za predstavljanje znanja koriste samo pravila nazivaju se *ekspertski sistemi zasnovani na pravilima* (en. Rule-Based expert systems).

Mogući problem koji se pojavljuje kod sistema zasnovanih na pravilima je održavanje *konzistentnosti baze znanja* (en. consistency maintenance) i najčešće može nastati iz sledećih razloga:

P_1 nezadovoljivo pravilo (en. *unsatisfiable rule*) - pravilo čija premisa nikada ne može biti zadovoljena;

P_2 neupotrebljivo pravilo (en. *unusable rule*) - pravilo čijim se izvršavanjem ništa ne postiže;

P_3 sadržano pravilo (en. *subsumed rule*) - postoji neko opštije pravilo;

P_4 redudantno pravilo (en. *redundant rule*) - pravilo koje ne doprinosi zaključivanju nijedne ciljne činjenice u bazi znanja. Problemi $P_1 - P_3$ su specijalni slučajevi P_4;

P_5 nekonzistentan par pravila (en. *inconsistent rule pair*) - dva pravila čije premise mogu biti zadovoljene u istoj situaciji, ali će izvesti kontradiktorne zaključke;

P_6 nekonzistentan skup pravila (en. *inconsistent rule set*) - primenom pravila iz nekonzistentnog skupa može dovesti do kontradiktornih zaključaka;

P_7 cikličan skup pravila (en. *circular rule set*) - nijedno pravilo se ne može izvršiti dok se ne izvrši neko drugo pravilo iz zadatog skupa;

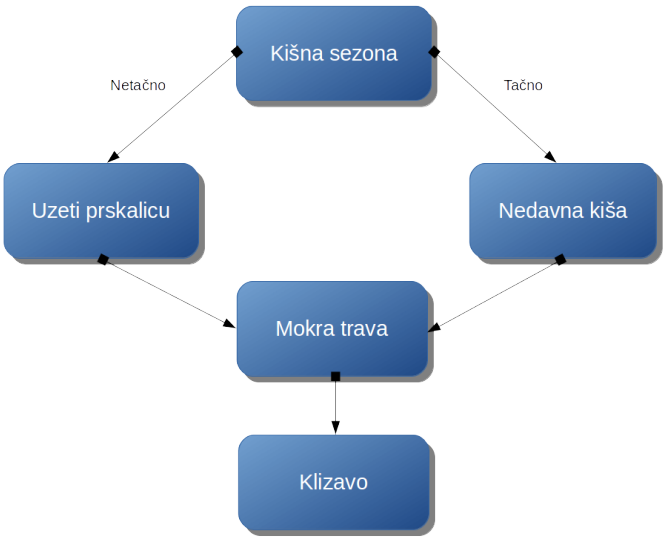
P_8 nekorišćene ulazne vrednosti (en. *unused input*) - vrednosti koje se nikada neće koristiti;

P_9 nekompletan skup pravila (en. *incomplete rule set*) - postoje određena ciljna stanja ili zaključci do kojih nikada nije moguće doći.

Kako bi se ovakvi problemi izbegli potrebno je bazu pravila održavati ažurnom. Neupotrebljiva i zastarela pravila treba blagovremeno ukloniti, a dodati neka nova pravila koja će ubrzati rad sistema.

4.1.2 Mreže zaključivanja

Sistemi zasnovani na pravilima u ogromnim ekspertskim sistemima mogu biti jako komplikovani. Proširenjem sistema, raste i broj pravila. Ljudi uglavnom vole da imaju jasnu sliku o problemu koji rešavaju. Kada se čvorovi grana koriste za predstavljanje premisa i zaključkaka pravila, a grane koriste za predstavljanje logičkih veze između njih nastaju *mreže zaključivanja* (en. *Inference networks*). Zbog stvaranja precizne slike o samoj prirodi problema široku primenu imaju u biologiji, medicini i genetici.



Slika 4: Primer mreže zaključivanja

4.1.3 Semantičke mreže

Pravila i mreže zaključivanja služe za predstavljanje male količine znanja. Sa druge strane, da bi se domensko znanje na kvalitetan način formalizovalo i unelo u ekspertske sisteme potrebno je na pogodan način opisati njihove osnovne pojmove i koncepte. Ovakav pristup ostvaruje se korišćenjem *semantičkih mreža* (en. *Semantic networks*).

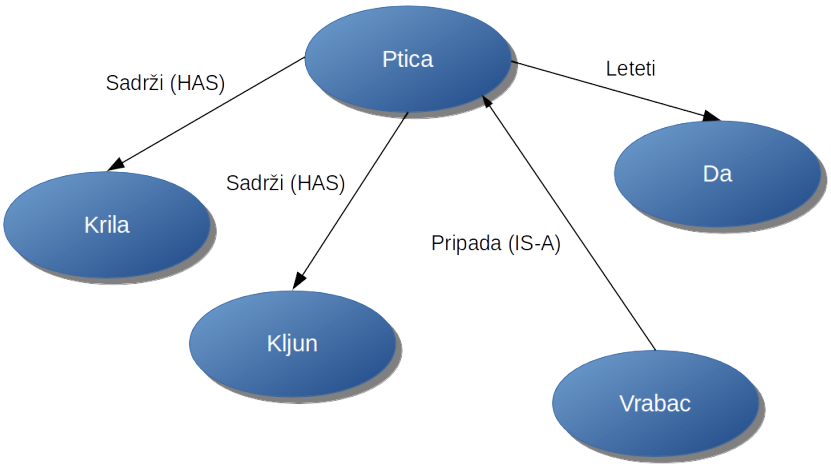
Metode za predstavljanje znanja uz pomoć grafova, pri čemu tačke na grafu predstavljaju objekte (koncepte), a lukovi veze između njih nazivaju se semantičke mreže. Graf kao struktura korisna je za prikaz znanja na način blizak ljudima (grafički prikaz), ali i struktura pogodna za predstavljanje znanja u ekspertskim sistemima. Ima dosta dobre osobine prilikom pretrage čvorova, na jednostavan način mogu se ukloniti, ali i dodati (konkatenirati) novi čvorovi. Sve ove pomenute osobine nasleđuju i semantičke mreže.

Ono što je zajedničko za sve semantičke mreže je deklarativni grafički prikaz koji se može koristiti za predstavljanje znanja i podršku automatskim sistemima za rasuđivanje o znanju. Dok su neke verzije kranje neformalne, sa druge strane ima veliki broj formalno definisani logički sistemi. Iako postoje mnoge varijacije semantičkih mreža, sve su sposobne da predstave individualne objekte, kategorije objekata i njihove međusobne relacije. Tipična grafička notacija prikazuje imena objekata ili kategorija u elipsama i povezuje ih označenim linijama. Priroda semantičkih mreža je intuitivna i vizuelna. Najčešće korišćene vrste semantičkih mreža su [11]:

- *implikacione mreže* (en. implicational networks) - koriste implikacije kao primarni odnos za povezivanje čvorova. Služe za predstavljanje znanja, zaključaka i obrazaca, tako da se slobodno može reći da njima pripadaju i mreže zaključivanja (4.1.2);
- *definicijske mreže* (en. definitional networks) - mreže koje opisuju strukturno znanje, naglašavaju odnos između konceptnog tipa i novodefinisanog podtipa;
- *mreže zasnovane na tvrdnji* (en. assertional networks) - za razliku od definicijskih mreža, za informaciju kod ovih mreža pretpostavlja se da je kontigentno tačna, osim ako nije eksplicitno drugačije naznačena. Neke od ovih tipova mreža su predložene za model konceptualnih struktura semantike prirodnih jezika;
- *izvršne mreže* (en. executable networks) - uključuju neke mehanizme ili procedure koji mogu da obavljaju zaključke, definišu poruke ili pronalaze neke šablone ili asocijacije;
- *mreže učenja* (en. learning networks) - grade i proširuju svoje znanje na osnovu primera. Novo znanje može promeniti stari izgled mreže, dodavanjem ili brisanjem čvora ili promenom težine luka;
- *hibridne mreže* (en. hybrid networks) - kombinacija dve ili više prethodno nabrojanih tehnika, bilo u jednoj ili više odvojenih mreža koje su tesno povezane.

Koncept (objekat) može biti bilo šta (mačka, auto, kuća itd.). Koncepti su međusobno spojeni usmerenim vezama. Ove veze se nazivaju lukovi i dele se na tri osnovne vrste:

- *IS-A* - slično nasleđivanju kod objekata. Kada kažemo da važi „O2 IS-A O1", znači da koncept O2 ima sve karakteristike koncepta O1 i može imati još neke dodatne osobine;
- *HAS* - slično vezi agregacije. Kada važi „B HAS A", sledi da koncept B sadrži u sebi koncept A;
- *Ostale veze* - odnose se na druge asocijacije.



Slika 5: Primer semantičke mreže

Slika 5 prikazuje primer jedne semantičke mreže. Relacije su predstavljene označenim vezama: veza *Pripada* (en. *IS-A*) između Vrabac i Ptica, odgovara logičkom iskazu Vrabac $\in$ Ptica, što govori da Vrabac nasleđuje sve osobine objekta Ptica. Veza *Sadrži* (en. *HAS*) između Ptica i Krila i/ili Ptica i Kljun označava da objekat Ptica u sebi sadrži Krila, odnosno Kljun. Veza (ostale veze): *Leteti* između Ptica i Da označava druge asocijacije, odnosno osobine objekta Ptica.

Semantičke mreže čine princip izvođenja nasleđivanjem dosta zgodnijim i čitljivijim. Kategorije i objekti služe za organizovanje i pojednostavljanje baze znanja korišćenjem nasleđivanja. Ako kažemo Voće je potklasa Hrane, a Jabuka je potklasa Voća sledi zaključak, da je svaka Jabuka jestiva. Iz navedenog sledi da Jabuke nasleđuju osobinu jestivosti u ovom slučaju njihovog pripadanja kategoriji Hrana - koja ima osobinu jestivosti.

Notacija koju semantičke mreže omogućavaju za određene vrste rečenica je često pogodnija, ali ako se uklone problemi ljudskog interfejsa, suštinski koncepti - objekti, relacije, kvantifikacije i slično su isti. Osim u ekspertskim sistemima, semantičke mreže imaju veliku primenu u istraživanju podataka, kao i u razvoju semantičkog Veb-a.

4.1.4 Okviri

Okviri (*en. Frames*) su preteča savremenih klasa iz koncepta objektno orijentisanog programiranja. Predložio ih je i definisao čuveni Marvin Minski[3] 1974. godine sa idejom da omogući predstavljanje strukturalnog, proceduralnog i deklarativnog znanja. Predstavljaju strukturu podataka koja služi za reprezentaciju stereotipskog znanja o nekom konceptu ili objektu.

[3]Marvin Minski (en. Marvin Minsky) [Njujork, 9. avgust 1927 - 24. januar 2016] je američki matematičar, smatran jednim od najznačajnijih naučnika na polju veštačke inteligencije. Zajedno sa Džonom Mekartijem osnovavali su laboratoriju veštačke inteligencije na MIT-u.

Slika 6: Primer okvira

Svaki okvir sadrži naziv koji se odnosi na konkretnu instancu i razlikuje se od svih ostalih instanci iste klase. Klasa predstavlja skup zajedničkih karakteristika niza sličnih objekata (potpuno isto kao u objektno orijentisanom pristupu). Karakteristike (en. *properties*) su atributi okvira i imaju konkretne vrednosti. Još jedan naziv koji se koristi da opiše karakteristike je *slot* (prorez). Svaki slot ima svoju vrednost. Slotovi mogu biti:

- statički;
- dinamički.

Dok statički slotovi imaju vrednost koja se ne menja u toku vremena, vrednost dinamičkih slotova se može menjati.

Okviri podržavaju veze nasleđivanja, agregacije i asocijacije. Zapravo, agregacije i asocijacije se implementiraju kao slotovi, gde je vrednost slota pokazivač prema drugom okviru. Okviri imaju elemente koji definišu neku vrstu ponašanja - *aspekti* (en. facet). Svaki slot može imati dva aspekta: *IF-NEEDED* i *IF-CHANGED*.

> *IF-NEEDED* - aktivira se ako je potrebno preuzeti vrednost za neki slot. Može da sadrži komande ili pravila koja definišu kako se dolazi do vrednosti.
>
> *IF-CHANGED* - aktivira se pri promeni vrednosti slota. Ovaj aspekt najčešće sadrži komande za proveru da li je nova vrednost izvan granica.

Osim aspekata, okviri mogu sadržati i metode. Metode predstavljaju elemente koji opisuju ponašanje karakteristično za okvir i po potrebi se mogu pozvati i izvršiti. Takođe, metode

omogućavaju komunikaciju između okvira. Za predstavljanje znanja u ekspertskim sistemima okviri se mogu koristiti sami - *ekspertski sistemi zasnovani na okvirima* (en. Frame-Based expert system) ili u kombinaciji sa pravilima - *hibridni ekspertski sistemi* (en. Hybrid expert system). Danas se okviri sve više napuštaju i zamenjuju se objektima odnosno, klasama.

4.1.5 Trojke Objekat-Atribut-Vrednost

Trojke Objekat-Atribut-Vrednost (en. *O-A-V triplets*) su ekspertski sistemi zasnovani na pravilima koji koriste obične promenljive za formiranje premisa i zaključaka. Služe za predstavljanje deklarativnog znanja. Problem kod ekspertskih sistema zasnovanih samo na pravilima nastaje kada postoji veliki broj promenljivih, jer nemaju nikakvu strukturu i nisu grupisane. Da bi se problem rešio, uvedene su O-A-V trojke. Promenljive se grupišu kao atributi nekih objekata. Atribut može imati jednu ili više vrednosti, dok iskaz može imati vrednosti sa verovatnoćom (zadatim koeficijentom). O-A-V trojke se mogu koristiti jedino u kombinaciji sa pravilima. Proširuju nedostatak pravila i na jednostavan način rastavljaju se na manje celine koje se kasnije upotrebljavaju u sistemima za zaključivanje. Može se reći da predstavljaju dosta pojednostavljenu verziju okvira jer ne podržavaju asocijacije, veze nasleđivanja, agregaciju, kao ni predstavljanje proceduralnog znanja. U modernim ekspertskim sistemima O-A-V trojke se sve više zamenjuju objektima tj. klasama. Međutim, zbog svoje jednostavnosti i sadržajnosti još uvek veliki broj mehanizama za zaključivanje i dalje koriste O-A-V princip kao osnovnu metodu za predstavljanje znanja, odnosno definisanje pravila.

4.1.6 Objekti (Klase)

Osnovna ideja objekata (klasa) je da se napravi odgovarajuća struktura koja bi u sebi opisivala neko stanje (karakteristike), ponašanje, ali i odnose (relacije). Međutim, sama ideja klase nije nastala odjednom. Preteča klasa bili su okviri koji su imali veoma slične karakteristike, ali nisu mogli u potpunosti da opišu ponašanje. Kada su klase i objektno orijentisani pristup postali popularni, javila se ideja da se mogu iskoristiti i kao metode za predstavljanje znanja u okviru ekspertskih sistema budući da mogu da opišu strukturno, proceduralno i deklarativno znanje. Posmatrajući objektno orijentisani koncept programiranja, može se reći da je klasa opšti predstavnik nekog skupa atributa, koji imaju istu strukturu i ponašanje. Ona predstavlja pojednostavljenu sliku objekta, obuhvata njegove karakteristike, ponašanje i relacije sa drugim objektima. Objekat je konkretan primerak neke klase (instanca klase).

Naziv klase je ono što razlikuje jednu klasu od druge i mora biti jedinstven. Atributi klase predstavljaju njene karakteristike, a njihove konkretne vrednosti predstavljaju trenutno stanje objekta klase. Metode klase opisuju ponašanje koje klasa može da ima. Kao i kod okvira, klase, takođe, mogu imati veze nasleđivanja, agregacije i asocijacije.

U savremenim ekspertskim sistemima klase se uvek koriste u kombinaciji sa pravilima. Atributi i metode klasa služe za formiranje premisa i zaključaka pravila. Prednost korišćenja klasa je laka integracija ekspertskog sistema u postojeće programe napisane u nekom objektno orijentisanom programskom jeziku (Java, Python, C++..). Omogućavaju lakše održavanje koda, testiranje i njegovu ponovnu upotrebljivost.

4.1.7 Iskazna i predikatska logika

Iskazna i predikatska logika (logika prvog reda) [1] najčešće služe kao osnova (srž) nekih „intuitivnijih" metoda za predstavljanje znanja. Kao tehnike nisu najpogodnije za zapis u računarskom svetu, a pre svega i oblik zapisivanja nije razumljiv ljudskom način razmišljanja, uglavnom ne nalaze direktnu primenu za predstavljanje znanja u ekspertskim sistemima.

U iskaznoj logici promenljive reprezentuju iskaze. Iskazi mogu biti kombinovani u složenije iskaze logičkim veznicima. Mnogi problemi se mogu opisati i reprezentovati korišćenjem iskazne logike. Tri osnovna aspekta iskazne logike su: sintaksa (ili jezik), semantika (ili značenje iskaza) i deduktivni sistemi. I semantika i deduktivni sistemi grade se nad isto definisanom sintaksom, tj. nad istim skupom formula. Centralni problemi u iskaznoj logici su ispitivanje da li je data iskazna formula valjana (tautologija) i da li je data iskazna formula zadovoljiva. Postoji više metoda i pristupa za ispitivanje valjanosti i zadovoljivosti. Neki od njih su semantičke, a neki deduktivne (tj. sintaksno-deduktivne) prirode. Ključna veza između ova dva koncepta je tvrđenje da je iskazna formula valjana (što je semantička kategorija) ako i samo ako je ona teorema (što je deduktivna kategorija). Zahvaljujući ovoj vezi, sintaksa iskazne logike (jezik iskazne logike), njena semantika (konvencije o značenju formula) i njena deduktivna svojstva čine kompaktnu celinu.

Logika prvog reda, predikatska logika, dosta je izražajnija od iskazne logike. Za razliku od iskazne logike uvode se kvantifikatori (univerzalni i egzistencijalni). Zahvaljujući kvantifikatorima, u logici prvog reda mogu se formulisati tvrđenja koja nije moguće formulisati na jeziku iskazne logike. Dozvoljeno je samo kvantifikovanje promenljivih. U okviru logike prvog reda mogu se opisati mnoge matematičke teorije. Kao i iskazna logika, logika prvog reda ima tri aspekta: svoju sintaksu (ili jezik), svoju semantiku (ili značenje iskaza) i svoje deduktivne sisteme. I semantika i deduktivni sistemi grade se nad isto definisanom sintaksom, tj. nad istim skupom formula. Kao i u iskaznoj logici, centralni problemi u predikatskoj logici su ispitivanje da li je data formula valjana i da li je data formula zadovoljiva. Za razliku od iskazne logike, ovi problemi nisu odlučivi, te ne postoje efektivni algoritmi za njihovo rešavanje. Problem ispitivanja valjanosti za predikatsku logiku je poluodlučiv, pa postoje metode koje za svaku valjanu formulu mogu da dokažu da je ona valjana, ali u suprotnom smeru to nije moguće. Postoji više metoda i pristupa za ispitivanje i dokazivanje valjanosti i zadovoljivosti. Neki od njih su semantičke, a neki deduktivne (tj. sintaksno-deduktine) prirode. Kao i u iskaznoj logici, ključna veza između ova dva koncepta je tvrđenje da je formula valjana.

4.2 Predstavljanje neizvesnog znanja

Kada je potrebno predstaviti neizvesno znanje u okviru ekspertskih sistema, metode koje se koriste drugačije su od prethodno navedenih. U pitanju su sve one situacije u kojima ne postoji dovoljno činjenica ili znanja da bi se moglo tvrditi da je neko rešenje ili zaključak sigurno tačan. Problem definisanja neizvesnosti u okviru ekspertskih sistema generalno se deli na dve situacije:

- situaciju kada su poznate verovatnoće važenja zaključaka na osnovu premisa;
- situaciju kada nisu poznate verovatnoće važenja zaključaka na osnovu premisa.

Prvobitno razvijeni ekspertski sistemi najčešće su koristili model neizvesnog znanja za predstavljanje podataka. Uglavnom je to matematički model zasnovan na teoriji verovatnoće. U daljem tekstu neće biti mnogo susreta sa neizvesnim znanjem i praktičnom primenom ovakvih ekspertskih sistema, već će biti pomenute samo neke osnovne metode i njihov kratak opis. Često u ekspertskim sistemima postoji veliki broj tehnika zaključivanja u kojima je princip rada upravo baziran na skupu čiji su elementi predstavljeni pomoću neizvesnog znanja. Takođe, neizvesno znanje se može koristiti i za predstavljanje određenih tipova heuristika.

U prvom slučaju (od gorepomenutih), *kada su poznate verovatnoće*, za izračunavanje se koristi Bajesova formula uslovnih verovatnoća, dok u drugom slučaju *kada nisu poznate verovatnoće*, koristi se ili teorija izvesnosti (en. *certainty theory*) ili rasplinuta teorija (en. *fuzzy theory*). Jedan

od osnovnih pojmova u teoriji verovatnoće je verovatnosna mera P. To je funkcija koja uzima vrednosti u intervalu [0, 1] i predstavlja verovatnoću da se određeni događaj realizuje. Najčešće se izračunava kao uslovna verovatnoća *a priori*, verovatnoća da se u budućnosti realizuje događaj A, ukoliko se već desio događaj B. Takođe, za ekspertske sisteme važan je i obrnuti problem, verovatnoća da se desio događaj B pre događaja A, ako se događaj A ostvario. Ova verovatnoća se naziva *a posteriori* i može se predstaviti Bajesovom formulom za uslovne verovatnoće:

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

Nedostatak teorije verovatnoće je nemogućnost da na odgovarajući način modeluje čovekovo razmišljanje ukoliko ne postoji dovoljno informacija. Ovaj problem otklonjen je *teorijom izvesnosti*, tako što se približnije modeluje način čovekovog razmišljanja. Dodaje se *faktor izvesnosti* koji može imati vrednost između -1 i 1 i označava subjektivan nivo izvesnosti (koji definiše ekspert). Vrednost -1 označava da je neki iskaz netačan, 1 da je iskaz tačan, a 0 nemogućnost da se utvrdi tačnost. Sve između je stepen slaganja ili neslaganja.

5 Rezonovanje i zaključivanje

U prethodnom delu teksta bilo je reči o ekspertskim sistemima, njihovoj osnovnoj nameni, arhitekturi i svojstvima. Opisane su osnovne tehnike za predstavljanje znanja u ekspertskim sistemima. Međutim, ono što karakteriše ekspertske sisteme i što ih čini „inteligentnim" sistemima jeste mogućnost rezonovanja i zaključivanja (*en. Reasoning and Inference*).

5.1 Rezonovanje u ekspertskim sistemima

Za razliku od predstavljanja znanja i njegove primene za opisivanje određenih informacija iz nekog domena na način pogodan sistemu (mašini), rezonovanje ima za cilj da koristeći znanje učini sistem (mašinu) „inteligentnom". Formalnije se može definisati kao proces rada sa znanjem, činjenicama i strategijama rešavanja problema (algoritmi, tehnike, heuristike itd.) radi dolaženja do zaključka o istim. Neke od najzastupljenijih metoda rasuđivanja, odnosno načina za donošenje odluka (zaključaka) u domenu veštačke inteligencije, ali i uopšteno, su:

- deduktivno rezonovanje - klasičan oblik razmišljanja i zaključivanja (modus ponens) u kome se na osnovu postojećeg skupa činjenica (premisa) izvode novi zaključci ili međuzaključci koristeći pravila izvođenja. Predstavlja zaključivanje koje ide od opšteg ka pojedinačnom. Tačnost ovog zaključivanja zavisi isključivo od tačnosti premisa, ispravnosti pravila izvođenja i njegove konkretne primene na premise. Neka su poznate sledeće činjenice: *Svi ljudi su smrtni*, *Sokrat je čovek*. Odatle se primenom dedukcije može izvesti: *Sokrat je smrtan*. Dedukcija je ključna u automatskom rezonovanju, naročito u automatskom i interaktivnom dokazivanju teorema. Tako je, na primer, koristeći neke od deduktivnih sistema (Hilbertov deduktivni sistem, prirodnu dedukciju, račun sekvenci) moguće napraviti dokazivač teorema zapisan na jeziku iskazne logike;

- induktivno rezonovanje - ljudi često zaključuju uopštavanjem pojedinih slučajeva (generalizacijom) u slučajeve određenog tipa. Suština induktivnog razmišljanja je prelazak od pojedinačnog ka opštem. Indukcija je po svojoj prirodi probabilističko rezonovanje. Donose se opšti zaključci na osnovu specifičnih, pojedinačnih opažanja. Uobičajeno je da se korišćenjem indukcije donose zaključci o svim pripadnicima određene klase na osnovu nekih pojedinačnih primeraka te klase. Međutim, očigledno je da ukoliko neko svojstvo

važi za određeni broj pripadnika neke klase, ne znači da će da važi za sve pripadnike te klase. Za razliku od deduktivnog zaključivanja, kod indukcije iz tačnosti premisa i nakon primene pravila izvođenja, ne sledi uvek izvođenje ispravnog zaključka. Izvedeni zaključak je samo verovatan, a pouzdanost raste sa brojem instanci koje ga podržavaju. Neka su date činjenice: *Na reci je uočen jedan beli labud*, *Na reci je uočen drugi beli labud*, *Na reci je uočeno još nekoliko desetina belih labudova.* Odatle je indukcijom moguće doneti zaključak: *Svi labudovi na reci su beli.* Međutim, proverom se može ustanoviti da na reci postoje labudovi koji nisu beli, pa zaključak nije garantovano tačan;

- abduktivno rezonovanje - drugačije se naziva i izvođenjem najboljeg zaključka. *Ako je tačno B i $A \Rightarrow B$, onda se može izvesti da važi A.* Ovakav način rezonovanja nalazi svoje opravdanje u odnosu uzroka i posledica, ako je A uzrok B i znamo da važi B, onda se može pretpostaviti da važi A. Slično kao i kod primene induktivnog rezonovanja, ne može se tvrditi da će zaključak abduktivnog rezonovanja biti tačan ukoliko su tačne premise i ukoliko su pravila izvođenja korektno primenjena. Zaključak se donosi sa određenom pouzdanošću koja obično zavisi od toga da li postoje alternative zaključku. Potrebno je proceniti da li su sve relevantne opcije razmotrene. Ukoliko postoje činjenice: *Ako je padala kiša, onda je trava mokra*, *Trava je mokra.* Abdukcijom je moguće izvesti zaključak: *Padala je kiša.* Očigledno je da ovakav zaključak ne mora biti tačan. Trava može biti mokra i zato što je neko zalivao. Ukoliko postoji takva informacija, da je neko zalivao travu, potrebno je nju razmotriti i promeniti zaključak. Promena zaključaka u prisustvu novih činjenica na lep način ilustruje nemonotonost abduktivnog rezonovanja.

Najčešća situacija je da ekspertski sistemi vrše deduktivno ili abduktivno rasuđivanje, dok oblast induktivnog rasuđivanja spada u domen mašinskog učenja i ređe se pojavljuje u okviru ekspertskih sistema. Sve ove vrste rezonovanja zasnivaju se na matematičkom pravilu zaključivanja modus ponens.

Pored toga, rasuđivanje se može klasifikovati i prema kriterijumu postojanosti činjenica o problemu. Ukoliko se činjenice o problemu ne menjaju u toku rasuđivanja u pitanju je *monotono rasuđivanje* (en. *monotonic reasoning*). Međutim, ako dođe do promene činjenica u toku rasuđivanja onda se radi o *nemonotonom rasuđivanju* (en. *non-monotonic reasoning*). U slučaju nemonotonog rasuđivanja algoritam za zaključivanje mora da sadrži i sistem za održavanje istinitosti (en. *truth maintenance system*). Ovaj sistem automatski ažurira listu činjenica i briše sve zaključke koji su nastali na osnovu činjenica koje više ne važe.

5.2 Principi zaključivanja u ekspertskim sistemima

Može se reći: *zaključivanje je proces kojim se na osnovu baze znanja dobijaju nove informacije na osnovu postojećih, a koje nisu eksplicitno sadržane u bazi.* Granica između rezonovanja i zaključivanja je veoma uska, tako da se u literaturi često može naći i ovakva definicija: kada čovek razmišlja o nečemu i donosi neke zaključke to se zove rasuđivanje (koristi intuiciju, preskačući „nebitne" korake - mentalni proces logike), dok sa druge strane, ekspertski sistemi (mašine) mogu samo da simuliraju čovekovo razmišljanje i takav proces se naziva zaključivanje. Zaključivanje u najužem smislu predstavlja niz koraka. Na primer: $P \Rightarrow Q \wedge \neg Q$. Može se zaključiti da važi $\neg P$, koristeći *modus tolens* kao pravilo zaključivanja. Takođe, zaključci ne moraju biti striktno deduktivne prirode. Primer jednog zaključivanja može se predstaviti i na sledeći način: *„nebo je vedro"*, *„ukoliko pada kiša, nebo nije vedro"*. Na osnovu datih činjenica može se zaključiti da *„napolju ne pada kiša"*.

U kontekstu veštačke inteligencije, pa tako i u ekspertskim sistemima, zaključivanje i rezonovanje su deo softvera (aplikacije, sistema) koji mogu da dođu do određenog rešenje polazeći od zadatog skupa činjenica. Može se naglasiti da je zaključivanje „implementacija" deduktivnog i abduktivnog rezonovanja.

Mehanizam zaključivanja uzima iz baze znanja podatke (zavisno od sistema i tehnika implementacije) u vidu okvira, objekata, pravila, kao i instance činjenica iz radne memorije. Zatim, primenjujući određeni algoritam zaključivanja na definisane podatke, dobija nove instance činjenica koje smešta ponovo u radnu memoriju ili dolazi do rešenja (zaključka).

Pred mehanizmom zaključivanja pored izbora principa, odnosno načina zaključivanja, postavljaju se i sledeća pitanja:

- 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?

Osim navedenih pitanja, postoji i niz drugih problema koje mehanizam zaključivanja treba uspešno da otkloni. Neke od osnovnih metoda zaključivanja u ekspertskim sistemima su:

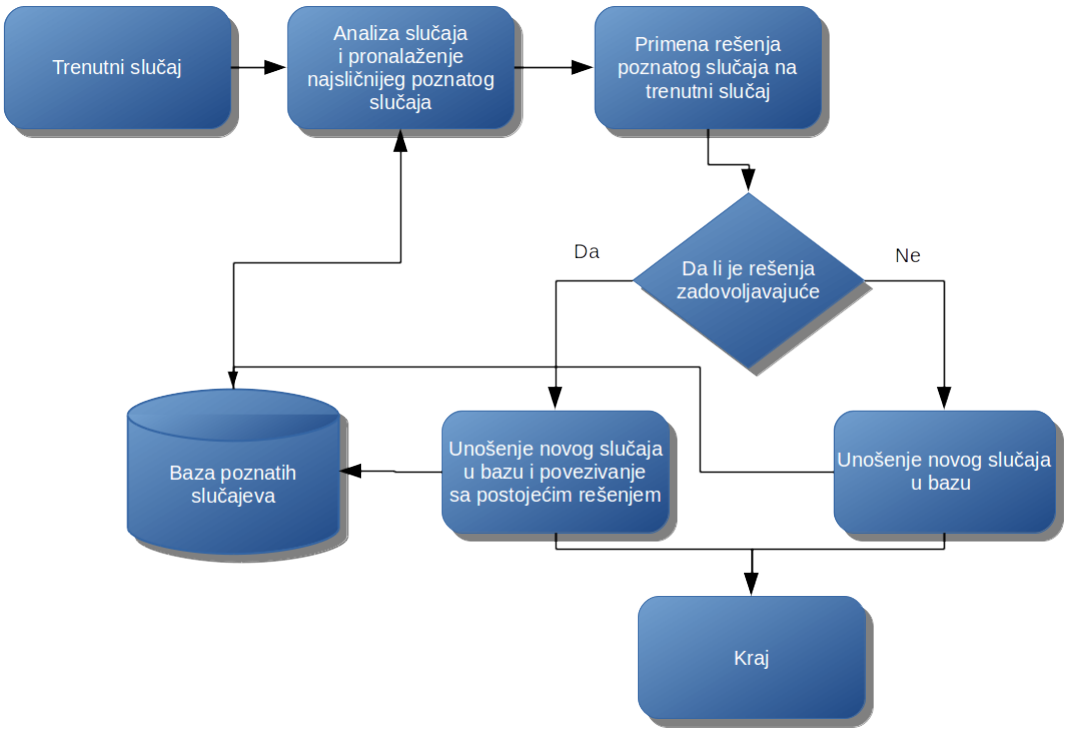
- *na osnovu već rešenih slučajeva*;
- *ulančavanje unapred (nizanje unapred)*;
- *ulančavanje unazad (nizanje unazad)*;
- *na osnovu faktora izvesnosti*;
- *na osnovu pomućenih pravila.*

Od svih navedenih tehnika[4] najviše treba obratiti pažnju na ulančavanje unapred i unazad. Mašina pravila čiji će princip funkcionisanja biti opisan u daljem tekstu upravo kombinacijom ove dve metode vrši zaključivanje, odnosno izbor narednog pravila u procesu zaključivanja. Kombinacija dve ili više tehnika u procesu zaključivanja naziva se *hibridna* metoda zaključivanja.

5.2.1 Zaključivanje na osnovu već rešenih slučajeva

Zaključivanje na osnovu već rešenih slučajeva (en. *Case-based inference*) zasniva se na jednom od principa ljudskog razmišljanja - pronalaženje rešenja na osnovu prethodnih iskustava iz sličnih situacija. U kontekstu zaključivanja na osnovu već rešenih slučajeva, situacije nazivamo *slučajevima*. Slučajevi se definišu kao određeni događaji ili situacije koji imaju jedinstvene karakteristike. Već rešeni slučajevi se unose u ekspertski sistem na neki formalni način i obično se na osnovu nekog algoritma za poređenje najsličnijeg po određenom kriterijumu (temperatura, pritisak...) trenutni slučaj poredi sa već postojećim u bazi i vrši se njegovo rangiranje. Uvodi se i povratna sprega, koja proverava da li je dobijeno rešenje zadovoljavajuće. Ako jeste, novi slučaj se unosi u bazu znanja i povezuje sa postojećim rešenjem, a ukoliko nije, slučaj se unosi u bazu, ali se za njega formira neko novo rešenje. Predstavljanje slučajeva i njihovih rešenja je veoma trivijalno i obično su to neke baze podataka ili objekti u radnoj memoriji.

[4]Metode koje se najčešće koriste u sistemima gde su pravila osnovna tehnika za predstavljanje znanja

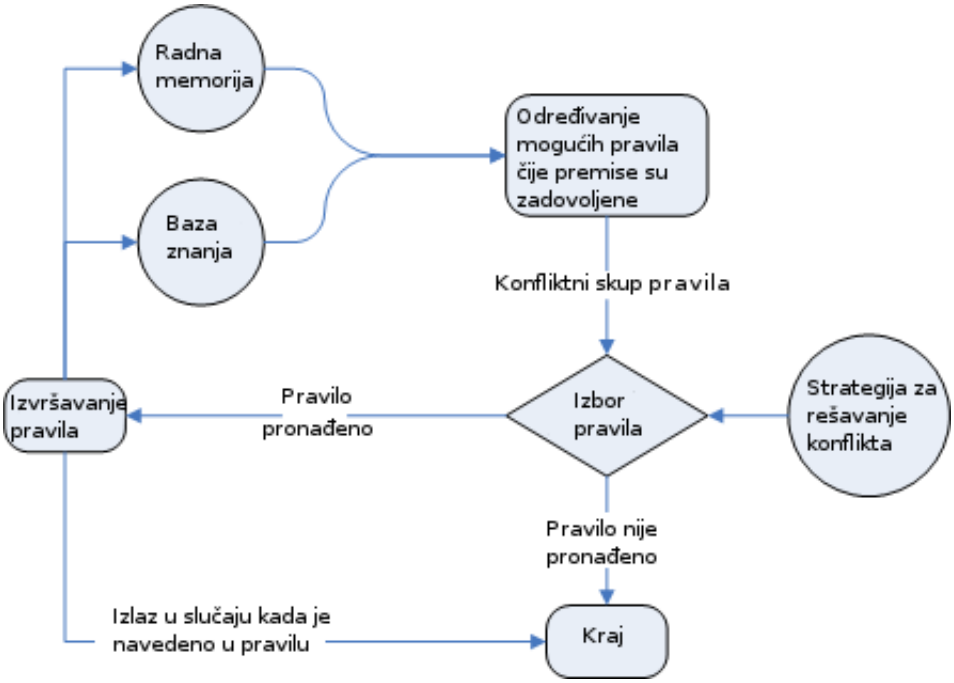


Slika 7: Algoritam zaključivanja na osnovu rešenih slučajeva

Metoda zaključivanja zasniva se na kvantitativnom i kvalitativnom poređenju najsličnijeg slučaja iz baze. Ukoliko se nađe slučaj sa potpuno istim karakteristikama, vrši se rangiranje u zavisnosti od važnosti pojedinačnih kriterijuma prema sličnosti sa traženim.

5.2.2 Zaključivanje, algoritam - ulančavanje unapred

Ulančavanje unapred (*en. Forward chaining)* predstavlja proces zaključivanja gde se polazeći od baze znanja stiže do cilja (podaci utiču na rezultat, karakteristično za sisteme bazirane na pravilima). Polazi se od skupa poznatih činjenica o problemu i izvode se nove činjenice korišćenjem pravila čije premise odgovaraju postojećim činjenicama. Ovaj proces se ponavlja sve dok se ne stigne do neke ciljane činjenice ili dok postoje pravila čije premise odgovaraju početnim ili izvedenim činjenicama. Koristi se modus ponens ili rezolucija kao način zaključivanja, dok se ne dobije zaključak (ne dođe do cilja) ili ne iscrpe sva primenljiva pravila. Ulančavanje unapred je jedna od najčešće korišćenih tehnika za zaključivanje. Koristi se isključivo u kombinaciji sa pravilima i moguće je izvesti dosta zaključaka na osnovu malo ulaznih podataka. Naziva se još i *strategija vođena podacima* (en. *data-driven strategy*). Na slici 8 prikazan je algoritam za ulančavanje unapred.



Slika 8: Ulančavanje unapred - algoritam

Prvi korak je formiranje konflitnog skupa pravila (en. *conflict rule set*). Tu spadaju sva pravila čije premise odgovaraju činjenicama, pa se zaključak pravila može izvesti. Dakle, u ovom koraku prolazi se kroz skup svih pravila, uočavaju se ona koja se mogu izvršiti i smeštaju se u konfliktni skup pravila. Konflikt nastaje zbog mogućnosti istovremenog izvršavanja većeg broja pravila, pa je pitanje koje pravilo trebalo prvo izvršiti. Radi optimizacije, u ovom koraku često se koristi rete algoritam (lat. *rete* - mreža). Drugi korak je rešavanje konflikta (en. *conflict resolution*) i izbor samo jednog pravila za izvršavanje. Postiji nekoliko mogućih strategija za rešavanje konflikta:

- izbor prvog pravila;
- izbor pravila sa najvišim prioritetom;
- izbor najspecifičnijeg pravila;
- izbor pravila koje se odnosi na najskorije dodate činjenice;
- izbor pravila koja nisu izvršena;
- izvršavanje svih pravila iz konfliktnog skupa ali u odvojenim linijama zaključivanja.

Treći korak je izvršavanje tj. okidanje pravila (en. *rule firing*). Kada se pravilo okine njegovi zaključci se dodaju postojećim činjenicama u radnoj memoriji i koriste se u narednom ciklusu zaključivanja. Kada se izvrši i treći korak, završava se jedan ciklus zaključivanja. Zaključivanje se najčešće vrši u više ciklusa, a prekida se ako se stigne do neke ciljne činjenice ili više nema pravila koja se mogu izvršiti.

Primer: U sledećem, vrlo jednostavnom primeru, biće prikazano kako se od početnog skupa pravila i činjenica dolazi do određenog zaključka koristeći tehniku ulančavanja unapred, kao mehanizam zaključivanja.

- baza znanja sadrži sledeća pravila:

```
pravilo_1:
  IF auto neće da "upali" AND napon na akumulatoru < 12V
  THEN akumulator je prazan
pravilo_2:
  IF akumulator je prazan
  THEN napuni akumulator
pravilo_3:
  IF auto nece da "upali" AND napon na akumulatoru = 12V
  THEN anlaser je neispravan
pravilo_4:
  IF anlaser neispravan
  THEN zameni anlaser
```

- radna memorija sadrži sledeće početne činjenice

```
auto neće da "upali"
napon na akumulatoru 11V
```

1. Početak:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [ ]

   izvršeno pravilo [ ]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

2. Ciklus 1, Korak 1:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [**pravilo_1**]

   izvršeno pravilo [ ]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

3. Ciklus 1, Korak 2:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [pravilo_1]

   izvršeno pravilo [**pravilo_1**]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

4. Ciklus 1, Korak 3:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, **akumulator je prazan(pravilo_1)**]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [pravilo_1]

   izvršeno pravilo [pravilo_1]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

5. Ciklus 2, Korak 1:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, akumulator je prazan(pravilo_1)]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [**pravilo_2**]

   izvršeno pravilo []

   Baza znanja: [~~pravilo_1~~, pravilo_2, pravilo_3, pravilo_4]

6. Ciklus 2, Korak 2:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, akumulator je prazan(pravilo_1)]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [pravilo_2]

   izvršeno pravilo [**pravilo_2**]

   Baza znanja: [~~pravilo_1~~, pravilo_2, pravilo_3, pravilo_4]

7. Ciklus 2, Korak 3:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, akumulator je prazan(pravilo_1), **napuni akumulator(pravilo_2)**]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [pravilo_2]

   izvršeno pravilo [pravilo_2]

   Baza znanja: [~~pravilo_1~~, pravilo_2, pravilo_3, pravilo_4]

8. Ciklus 3, Korak 1:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, akumulator je prazan(pravilo_1), napuni akumulator(pravilo_2)]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [**-**]

   izvršeno pravilo [ ]

   Baza znanja: [~~pravilo_1~~, ~~pravilo_2~~, pravilo_3, pravilo_4]

9. Kraj:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, akumulator je prazan(pravilo_1), napuni akumulator(pravilo_2)]

   Mehanizam za zaključivanje:

   konfliktni skup pravila [-]

   izvršeno pravilo [ ]

   Baza znanja: [~~pravilo_1~~, ~~pravilo_2~~, pravilo_3, pravilo_4]

ZAKLJUČAK - napuni akumulator

Prednosti sistema koji koriste ulančavanje unapred kao tehniku zaključivanja je to što su dobri za probleme kod kojih podaci imaju ključnu ulogu (planiranje, monitoring, interpretacija). Loša osobina je to što nemaju predstavu o važnosti pitanja, postavljaju nepotrebna pitanja (sva zadata).

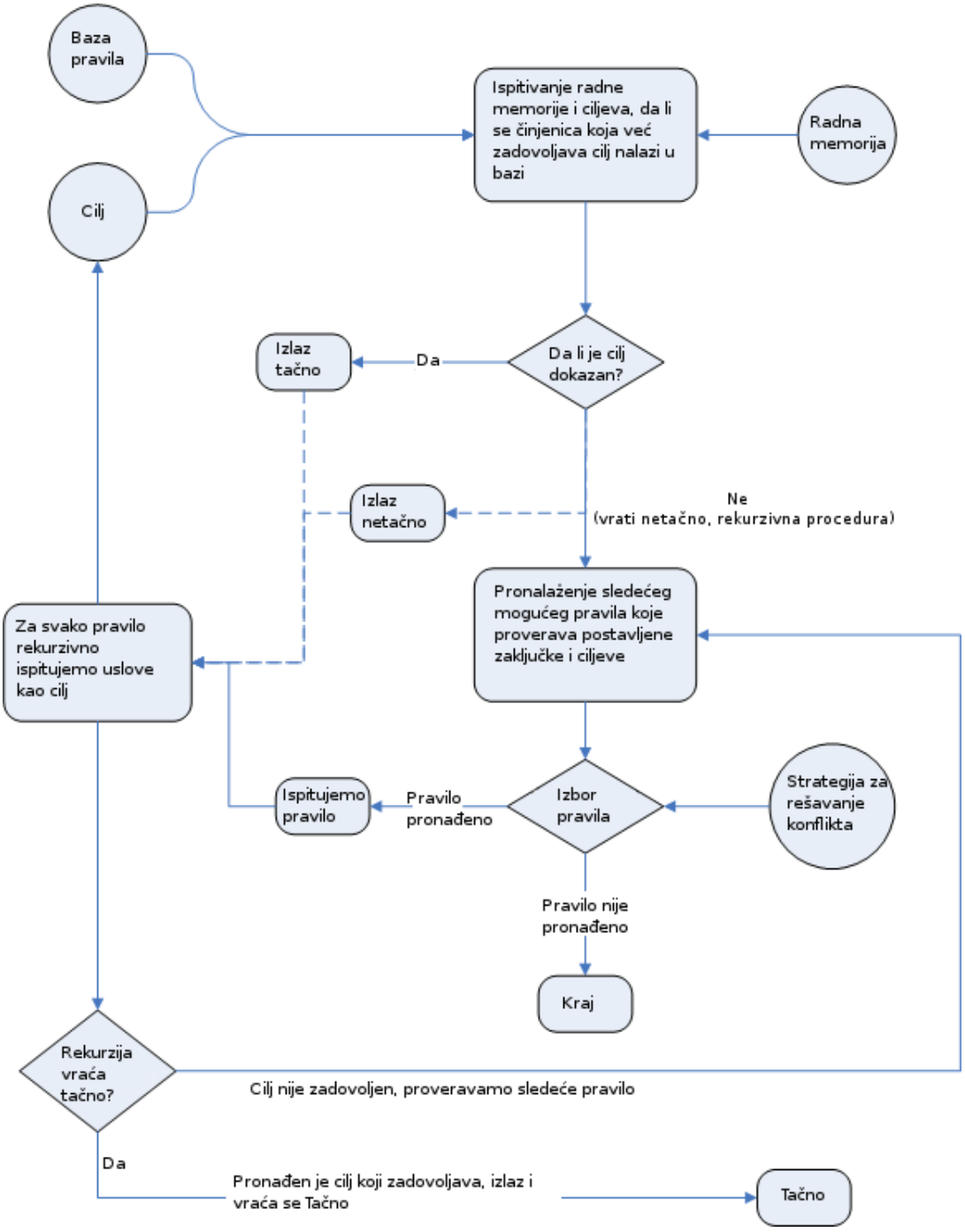
5.2.3 Zaključivanje, algoritam - ulančavanje unazad

Slično ulančavanju unapred i ulančavanje unazad (*en. Backward chaining*) je metoda za zaključivanje koja se može koristiti isključivo u kombinaciji sa pravilima. Međutim, ulančavanje unazad je ciljno vođena strategija (en. *goal-driven strategy*). Dakle, ne ide se od podataka ka zaključcima, već obrnuto: traže se podaci da bi se dokazali ili opovrgli zaključci. Najjednostavnije se može definisati kao strategija zaključivanja u kojoj se pokušava dokazati neka teorema prikupljanjem podataka potrebnih za njeno dokazivanje.

Počinje se od zadatog cilja, odnosno činjenicom koju je potrebno dokazati. Prvi korak je provera da li se ta činjenica već ne nalazi u memoriji. Ako se ne nalazi, sistem traži pravila (jedno ili više njih) koja u sebi sadrže tu činjenicu. Kada ih nađe, sistem proverava da li se premise ovih pravila nalaze u memoriji u vidu činjenica, ako se nalaze, početni cilj je ostvaren i činjenica je dokazana. Ako se ne nalaze, traži se novi skup pravila koji u svom delu ima premise, pravila iz prethodnog kruga. Ceo proces se ponavlja dok se ne naiđe na primitive[5]. Tada korisnik daje podatke o tačnosti primitiva i time dokazuje ili opovrgava početnu činjenicu. Korisno je imati pravila kojima se bira cilj (nizanjem unapred) i to se može postići posebnim pravilima koja utiču na biranje cilja.

U okviru ovog algoritma ciljevi se dokazuju hijerarhijski. To znači da se prvo dokazuju ciljevi nižeg nivoa, pa tek onda ciljevi višeg nivoa. Hijerahijska struktura u kojoj se čuvaju ciljevi zove se agenda ciljeva. Ulančavanje unazad koristi se u trenucima kada postoji samo par rešenja problema i treba dokazati koje je rešenje tačno. To su situacije u kojima bi pretraživanje celog problemskog prostora ulančavanjem unapred dovelo do eksplozije činjenica i međuzaključaka. Posledice ovakvih situacija su prikupljanje nepotrebnih podataka.

[5]Primitive su činjenice koje su premise pravila i kao takve se dalje ne rastavljaju, ne nalaze se u *THEN* delu nijednog pravila.



Slika 9: Ulančavanje unazad - algoritam

Primer: U sledećem primeru biće prikazano na koji način funkcioniše mehanizam zaključivanja prilikom izbora ulančavanja unazad. Za bazu znanja i početne činjenice uzeti su podaci iz prethodnog primera (5.2.2).

- baza znanja sadrži sledeća pravila:

```
pravilo_1:
  IF auto neće da "upali" AND napon na akumulatoru < 12V
  THEN akumulator je prazan
pravilo_2:
  IF akumulator je prazan
  THEN napuni akumulator
pravilo_3:
  IF auto neće da "upali" AND napon na akumulatoru = 12V
  THEN anlaser je neispravan
pravilo_4:
  IF anlaser neispravan
  THEN zameni anlaser
```

- radna memorija sadrži sledeće početne činjenice

```
auto neće da "upali"
napon na akumulatoru 11V
```

1. Početak:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva [napuni akumulator]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

2. Ciklus 1, Korak 1:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

3. Ciklus 1, Korak 2:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**, akumulator je prazan **(pravilo_2)**]

   Baza znanja: [pravilo_1, pravilo_2, pravilo_3, pravilo_4]

4. Ciklus 2, Korak 1:

   Radna memorija: [auto neće da "upali", napon na akumulatoru = 11V, **akumulator je prazan(pravilo_2)**]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**, akumulator je prazan (pravilo_2) - **nedokazano**]

   Baza znanja: [pravilo_1, ~~pravilo_2~~, pravilo_3, pravilo_4]

5. Ciklus 2, Korak 2:

   Radna memorija: [auto neće da upali, napon na akumulatoru = 11V, akumulator je prazan(pravilo_2)]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**, akumulator je prazan (pravilo_2) - **nedokazano**, **auto nece da "upali", napon na akumulatoru < 12V(pravilo_1)**]

   Baza znanja: [pravilo_1, ~~pravilo_2~~, pravilo_3, pravilo_4]

6. Ciklus 3, Korak 1:

   Radna memorija: [**auto neće da upali, napon na akumulatoru = 11V**]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**, akumulator je prazan (pravilo_2) - **nedokazano**, auto nece da "upali", napon na akumulatoru < 12V(pravilo_1) - **dokazano**]

   Baza znanja: [~~pravilo_1~~, ~~pravilo_2~~, pravilo_3, pravilo_4]

7. Ciklus 3, Korak 1A:

   Radna memorija: [auto neće da upali, napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **nedokazano**, akumulator je prazan (pravilo_2) - **dokazano**, auto nece da "upali", napon na akumulatoru < 12V(pravilo_1) - **dokazano**]

   Baza znanja: [~~pravilo_1~~, ~~pravilo_2~~, pravilo_3, pravilo_4]

8. Ciklus 3, Korak 1B:

   Radna memorija: [auto neće da upali, napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **dokazano**, akumulator je prazan (pravilo_2) - **dokazano**, auto nece da "upali", napon na akumulatoru < 12V(pravilo_1) - **dokazano**]

   Baza znanja: [~~pravilo_1~~ , ~~pravilo_2~~, pravilo_3, pravilo_4]

9. Kraj:

   Radna memorija: [auto neće da upali, napon na akumulatoru = 11V]

   Mehanizam za zaključivanje:

   Agenda ciljeva: [napuni akumulator - **dokazano**, akumulator je prazan (pravilo_2) - **dokazano**, auto nece da "upali", napon na akumulatoru < 12V(pravilo_1) - **dokazano**]

   Baza znanja: [~~pravilo_1~~, ~~pravilo_2~~, pravilo_3, pravilo_4]

ZAKLJUČAK - napuni akumulator - TAČNO

Kao osnovna prednost ovih sistema može se izdvojiti da su dobri za potvrdu hipoteza. Pored toga, usredsređeni su na problem (dijagnoza, savet, otklanjanje grešaka (en. *debug*)). Glavni nedostak sistema zasnovanih na algoritmu ulančavanje unazad jeste taj što će pratiti liniju zaključivanja iako ne treba (postoje poboljšanja, meta-pravila i faktori uverenja).

5.2.4 Zaključivanje na osnovu faktora izvesnosti

Prvobitno razvijeni ekspertski sistemi zasnovani na pravilima (MYCIN i EMUCIN) kao tehniku zaljučivanja koriste zaključivanje na osnovu *faktora izvesnosti* (en. *Certainty factory*). U to vreme, 1970-ih godina, ova tehnika je veoma popularna zbog matematičke i računarske jednostavnosti.

Osnovna ideja zaključivanja na osnovu faktora izvesnosti svodi se na sledeće korake [5]:

- utvrditi koje činjenice važe i koji je njihov faktor izvesnosti;
- upariti činjenice i premise, izvesti zaključak i izračunati faktor izvesnosti svakog novog zaključka na osnovu faktora izvesnosti pravila i premisa;
- ukoliko ima još pravila koja nisu izvršena, a mogu upasti u skup pravila koja treba izvršiti - ponoviti proces.

Ukoliko postoji premisa pravila sa faktorom izvesnosti CF(premisa) = 0.7, a faktor izvesnosti pravila CF(pravilo) = 0.6, faktor izvesnosti zaključka CF(z) izračunava se na osnovu sledeće formule:

$$\text{CF(z)} = \text{CF(premisa)} \cdot \text{CF(pravilo)} = 0.7 \cdot 0.6 = 0.42$$

Ukoliko pravilo sadrži više premisa, u zavisnosti od toga kojom logičkom operacijom su one povezane (najčešće *AND* i *OR*), faktor izvesnosti se izračunava:

- *AND*:

$$\text{CF(z)} = \min[\text{CF(premisa\_1)}, \text{CF(premisa\_2)}, ..., \text{CF(premisa\_N)}] \cdot \text{CF(pravilo)}$$

- *OR*:

$$\text{CF(z)} = \max[\text{CF(premisa\_1)}, \text{CF(premisa\_2)}, ..., \text{CF(premisa\_N)}] \cdot \text{CF(pravilo)}$$

Često se dešava da u okviru jednog ekspertskog sistema, dva pravila sa različitim premisama vode ka istom zaključku. Kada važi samo premisa za *pravilo_1* ili premisa za *pravilo_2*, to nije problem, zato što će se prilikom izračunavanja faktora izvesnosti zaključka aktivirati samo jedno pravilo i ono će ući u formulu. Problem je kada važe premise za *pravilo_1* i *pravilo_2*, jer je potrebno iskombinovati dve vrednosti kako bi dobili faktor izvesnosti zaključka CF(z). Ukupan faktor izvesnosti za ova dva pravila CF1(pravilo_1) i CF2(pravilo_2) može se izračunati na osnovu formule [9]:

$$CF(z) = \begin{cases} CF1(z) + CF2(z) \cdot (1 + CF1(zak.)), & \text{CF1, CF2} \geq 0 \\ CF1(z) + CF2(z) + CF1(z)CF2(z), & \text{CF1, CF2} < 0 \\ \frac{CF1(z)+CF2(z)}{1-min(|(CF1(z)|,|CF2(z)|)} & \text{inače.} \end{cases} \quad (1)$$

Model je nastao kako bi se predstavile subjektivne verovatnoće u ekspertskim sistemima. Pomoću faktora izvesnosti na jednostavan način se mogu predstavi uslovne verovatnoće i uslovne nezavisnosti pravila.

5.2.5 Zaključivanje na osnovu rasplinutih pravila

U rasplinutoj logici (en. *Fuzzy logic*) nije precizno definisana pripadnost jednog elementa određenom skupu, već se pripadnost meri na primer u procentima. Ove mere pripadnosti, skalirane, mogu da uzimaju vrednosti od 0 do 1. Rasplinuta logika kao koncept puno je prirodnija nego što se to na prvi pogled može zaključiti. Naime, postoje situacije u kojima nije moguće znanje o sistemu reprezentovati na apsolutno precizan način. Najčešće se koristi u kombinaciji sa pravilima da bi se predstavilo neizvesno ili neprecizno znanje u okviru ekspertskih sistema. Tehnika zaključivanja na osnovu rasplinutih pravila (en. *fuzzy rules*) svodi se na nekoliko koraka. Svako pravilo predstavlja odnos između dva ili više rasplinuta skupa tj. definiše preslikavanje između dva skupa. Rasplinuti skupovi su osnovni elementi kojima se opisuje nepreciznost. Diskretan skup sadrži elemente sa istim svojstvima (skup jabuka, skup celih brojeva itd.), dok rasplinuti skupovi sadrže elemente sa sličnim svojstvima (skup visokih ljudi, skup niskih ljudi, skup brzih automobila itd.). U diskretnim skupovima element pripada (1) ili ne pripada (0) određenom skupu. Sa druge strane elementi u rasplinutim skupovima mogu delimično da pripadaju skupu, odnosno pripada skupu (1 - 100%), ne pirpada skupu (0 - 0%) ili u određenoj meri pripada skupu (0.7 - 70%). Algoritam rasplinutog zaključivanja je sledeći [4]:

Preslikavanje se formalizuje formiranjem rasplinute asocijativne matrice (FAM). Matrica predstavlja tabelu istinitosti rasplinutih skupova. Izračunavanje FAM-a se vrši na osnovu vektorskih specifikacija oba skupa po principu MAX-MIN, odnosno proizvod rasplinutih skupova. FAM se izračunava samo jednom - na početku procesa zaključivanja. Drugi korak je relativno trivijalan i podrazumeva prikupljanje činjenica. Činjenice mogu biti unete u formi pomućenih iskaza, ali i ne moraju. Treći korak je nešto složeniji. Unete činjenice transformišu se u vektorsku formu i množe sa FAM. Time se dobijaju zaključci u vektorskoj formi. Poslednji korak je defazifikacija (en. *defuzzyfication*) u kojoj se rezultirajući vektor svodi na jednu vrednost. Ta vrednost predstavlja zaključak koji se unosi u skup poznatih činjenica. Četvrti korak podrazumeva preispitivanje uslova za prekid. Ukoliko uslovi nisu ispunjeni ceo proces se ponavlja ispočetka, ali se zaključci izvedeni u poslednjem ciklusu unose u listu činjenica.

Pomućena logika najčešće se koristi za modeliranje složenih sistema u kojima je primenom drugih metoda veoma teško utvrditi međusobnu zavisnost promenljivih u modelu [7].

5.3 Ostale tehnike zaključivanja

Pored navedenih tehnika zaključivanja često se u ekspertskim sistemima koriste tabele i stabla odlučivanja. Dobre karakteristike stabla odlučivanja i mogućnost primene algoritama *pretraživanje po dubini* (DFS) i *pretraživanje po širini* (BFS), imaju važnu ulogu u savremenim ekspertskim sistemima. Takođe, treba napomenuti da razne heuristike utiču na performanse zaključivanja i mogu znatno ubrzati rad ekspertskog sistema. Najveći problem je izbor sledećeg pravila u skupu pravila, posebno kada taj skup poprilično naraste. Upravo u ovim situacijama stabla odlučivanja i heuristike nalaze svoju primenu.
Izbor metode zaključivanja najviše zavisi od načina predstavljanja znanja, ali takođe, i od domena za koji se sistem implementira. Kombinacijom više tehnika zaključivanja u okviru jednog ekspertskog sistema poboljšava se karakteristika tog sistema. Nedostaci jedne tehnike uklanjaju se implementacijom neke druge tehnike i obratno. Upravo hibridni sistemi imaju najveće praktične primene.

6 Mašina pravila - produkcioni sistem pravila

Mašina pravila (*en. Rule engine*) je ekspertski sistem koji se sastoji od skupa parova tipa „uslov - akcija". Ovako definisana pravila nazivaju se produkciona pravila ili još pravila odlučivanja, pa se tako često u literaturi i praksi ovakvi sistemi nazivaju produkcioni ekspertski sistem ili *produkcioni sistem pravila* (en. *Production Rule System*).

```
when
  <condition>
then
  <actions>
```

Osnovu sistema čine baza činjenica koje se smeštaju u radnu memoriju, baza znanja i procedura za izvršavanje produkcionih pravila i mehanizma zaključivanja (Slika 10).

Slika 10: Arhitekturalni prikaz mašine pravila

Na slici 10 može se videti da je arhitektura sistema zasnovanog na mašini pravila veoma slična arhitekturi ostalih ekspertskih sistema.

Baza znanja: produkciona pravila u deklarativnom obliku (*uslov - akcija*) baziraju se na principima prediktivne logike, oblika „*AKO... TADA...*" (*IF... THEN...*). Svako produkciono pravilo sadrži uslov koji mora biti zadovoljen pre nego što se izvrši akcija.

Radna memorija (činjenice): sadrži skup informacija koje se testiraju kako bi se zaključila istinitost nekog uslova. Predstavlja činjenice iz globalne baze podataka koje se dovlače u radnu memoriju.

Mehanizam zaključivanja: jednostavna programska petlja koja testira istinitost uslova pravila i izvršava akcije kad je uslov tačan. Mehanizam zaključivanja se implementira korišćenjem nekog od algoritama zaključivanja (poglavlje 5.2).

Osim nabrojanih komponenti, u literaturi često važnu arhitekturalnu ulogu ima i *interfejs*. Omogućava komunikaciju mašine pravila sa korisnicima sistema. Zbog njegove značajnosti, u daljem tekstu *interfejs* se podrazumeva kao sastavni deo mašine pravila, tako da neće biti posebno naveden. Opšti oblik produkcionih pravila je:

```
IF situacija s THEN akcija a
IF uslov u THEN posledica p
IF p THEN s TO DEGREE d
```

Primer: Za određeni celi broj X, dati savet kako se on predstavlja pomoću rimskih cifara.

- baza znanja sadrži sledeća pravila:

```
pravilo_1:
      IF X == null OR X == undefined
      THEN unesite vrednost za X
pravilo_2:
      IF X != 0 AND X > 1 AND X <= 3
      THEN stampaj 'I' AND X -= 1
pravilo_3:
      IF X != 0 AND X == 4
      THEN stampaj 'IV' AND X -= 4
pravilo_4:
      IF X != 0 AND X >= 5 AND X <= 8
      THEN stampaj 'V' AND X -= 5
pravilo_5:
      IF X != 0 AND X == 9
      THEN stampaj 'IX' AND X -= 9
pravilo_6:
      IF X != 0 AND X >= 10 AND X <= 39
      THEN stampaj 'X' AND X -= 10
pravilo_7:
      IF X != 0 AND X == 40
```

```
        THEN stampaj 'XL' AND X -= 10
pravilo_8:
        IF X != 0 AND X >= 50 AND X <= 59
        THEN stampaj 'L' AND X -= 50
pravilo_9:
        IF X != 0 AND X >= 100 AND X <= 399
        THEN stampaj 'C' AND X -= 100
pravilo_10:
        IF X != 0 AND X >= 400 AND X < 500
        THEN stampaj 'CD' AND X -= 400
pravilo_11:
        IF X != 0 AND X == 500
        THEN stampaj 'D' AND X -= 500
pravilo_12:
        IF X != 0 AND X >= 900 AND X < 1000
        THEN stampaj 'CM' AND X -= 900
pravilo_13:
        IF X != 0 AND X >= 1000
        THEN stampaj 'M' AND X -= 1000
pravilo_14:
        IF X == 0 AND
        THEN stampaj 'KRAJ'
```

- radna memorija sadrži činjenicu:

```
X ima vrednost 25
```

- mehanizam za zaključivanje: [**pravilo_6, pravilo_6, pravilo_4, pravilo_14**]

Mehanizam zaključivanja pretražuje pravila na osnovu zadatih činjenica. Kako postiji samo jedna činjenica „X ima vrednost 25", mehanizam zaključuje: X je između 10 i 39, tako da će izabrati *pravilo_6*. Primenom ovog pravila štampa se X i umanjuje vrednost X za 10. Sada X ima vrednost 15. Kako je 15 isto između 10 i 39, opet se primenjuje *pravilo_6*. Štampa se X i umanjuje za 10. Vrednost promenljive X je sada 5. Jasno se sada vidi da će mehanizam primeniti *pravilo_4*. Štampa se V i umanjuje X za 5. X sada ima vrednost 0 i to je oznaka za kraj. „Okida" se *pravilo_14*, štampa se KRAJ i nema više pravila koja se mogu primeniti (skup pravila je prazan).

Ovaj jednostavan primer služi samo za demonstraciju mašine pravila i načina kako se pravila mogu predstaviti i opisati u sistemu. U praksi postoje dosta složenije situacije i pravila se opisuju deklarativno, na ljudima mnogo jasniji i čitljiviji način. Ovim se omogućava lakša komunikacija korisnika sa sistemom i međusobnom interakcijom, mašina pravila dolazi do preciznijeg rešenja (zaključka).

Mehanizam zaključivanja je osnovna komponenta u sistemu mašine pravila, što se može uočiti na osnovu navedenog primera. Kvalitet zaključivanja zavisi od dobro odabranih pitanja, dobro odabranih odgovora i što je najvažnije njihovog pravilnog ocenjivanja. Mehanizam zaključivanja pretražuje pravila koja mogu biti u bilo kom redosledu, što odražava neproceduralnost sistema. Vrši se testiranje uslova svakog pravila, sve dok se ne naiđe na uslov koji je tačan. Tada se izvršava

(„okida") određena akcija koja je vezana za taj uslov i ažurira baza činjenica. Zatim, mehanizam zaključivanja počinje ponovo da pretražuje skup pravila. Postupak se ponavlja beskonačno, ili dok se mehanizam zaključivanja na neki način ne zaustavi. Sistem prestaje sa radom kada nijedan uslov nije tačan, odnosno kada ne postoji više pravila koja se mogu primeniti. Mehanizam zaključivanja u mašini pravila zasnovan (implementiran) je na već pomenutim tehnikama, ulančavanju unapred i ulančavanju unazad, odnosno kombinacijom ova dva metoda, *hibridni* pristup. Kako je već opisan algoritam rada ovih metoda (poglavlje 5.2.2 i 5.2.3), biće predstavljen samo sažeti princip (nekoliko koraka) zaključivanja u mašini pravila:

- pronalaze se sva primenljiva pravila (en. *Match*);
- bira se pravilo koje će se izvršiti (en. *Conflict resolution*);
  - odbaciti pravila koja su već ranije primenjena, a nemaju ulogu u daljem radu
  - izabrati novo pravilo
- dodati novu činjenicu u bazi znanja (en. *Act*).

Strategija za rešavanje konflikata:

- izabrati pravilo koje daje nove zaključke (en. *No duplication*);
- izabrati prvo pravilo čiji je uslov ispunjen (en. *Do one*);
- primenjivati sva pravila (en. *Do all*);
- primenjivati najspecifičnije pravilo (en. *Do the most specific*);
- izabrati pravilo čiji uslov koristi najnovije činjenice (en. *Do the most recent*).

Da bi se minimizirao potrebni broj testiranja uslova potrebno je urediti skup pravila u odgovarajućem redosledu, ili primeniti rete algoritam. Na ovaj način izbegava se nepotrebno testiranje. Pogodnim uređenjem skupa pravila moguće je ostvariti paralelizaciju procesa testiranja, čime se može značajno ubrzati rad sistema. Ovo naročito postaje korisno kada skup pravila i činjenica naraste. Ukoliko se izbor pravila vrši linearnim pretraživanjem liste pravila, i to stalno iz početka, dolazi do otežavajućih okolnosti. Vreme koje je potrebno da se testira svaki uslov varira. Sa povećanjem složenosti i u slučaju da je uslov iskazan kao kombinacija poduslova, postoji verovatnoća da se sistem nikada ne zaustavi. Kako bi se ovakvi slučajevi izbegli, za pretraživanje pravila uvodi se stablo odlučivanja, dodaju se težinski faktori, vrši se izbor pravila na osnovu prethodnih slučajeva ili se primenjuje neka od heuristika.
Uvođenjem dodatnih principa u okviru mašine pravila, gubi se na jednostavnosti i modularnosti originalnog produkcionog sistema. Dodavanje novih pravila ili brisanje već definisanih postaje komplikovanije. Zato je potrebno dobro izbalansirati sistem i organizovati kôd tako da je moguće lako proširiti i ukloniti dodatne elemente sistema u zavisnosti od potrebe.

6.1 Zašto se koristi mašina pravila

Najčešće postavljena pitanja vezana za temu korišćenja mašine pravila su:

- Kada treba koristiti mašinu pravila?
- Šta su prednosti ovakvih sistema kada se kodira preko *if ... then* pristupa?
- Zašto izabrati mašinu pravila, a ne pristup preko nekog skript okruženja (en. *shell*)?

6.1.1 Prednosti sistema sa mašinom pravila

Postoji veliki broj beneficija kod korišćenja sistema zasnovanih na pravilima. Mašine pravila omogućavaju pre svega *deklarativni* pristup programiranja *„Šta da radimo"*, a ne *„Kako to uraditi"*. Ovakav pristup predstavlja ključnu prednost, budući da se pomoću pravila može znatno olakšati način izražavanja rešenja za veoma složene probleme. Pravila su mnogo čitljivija od klasičnog koda i vrlo jasno se može posmatrati kako od početnog uslova doći do rešenja, kombinacijom opisanih pravila. Dalje, veoma bitna osobina ekspertskih sistema zasnovanih na pravilima jeste *razdvajanje logike od samih podataka*. Podaci su u domenu objekata, dok je sama logika u pravilima. Ovo u osnovi krši principe objektno orijentisane paradigme, ali samim razdvajanjem omogućava se lakše održavanje sistema i njegovo proširenje u budućnosti. Logika je nezavisna celina i ne zavisi od samih podataka. Predstavlja jako bitnu karakteristiku u sistemima gde je sama logika vezana za više različitih domena. *Centralizacija znanja* omogućava da se korišćenjem pravila kreira „magacin" znanja (baza znanja) koja je izvršna. To znači da ukoliko se dokaže da je neka činjenica tačna ona se može upotrebiti kasnije kao takva bez njenog daljeg ispitivanja. Kada su pravila precizno definisana veoma je lako, odnosno ne zahteva dodatni napor za pisanje jasne *dokumentacije*, a ostavlja se i mogućnost pisanja samih pravila na „prirodnom" (govornom) jeziku. Ono što je značajno napomenuti kao veoma bitnu osobinu ovih sistema je *laka integracija* u već postojeće sisteme i aplikacije, kao i *visoka brzina i skalabilnost*, tj. mogućnost *dodavanja novih pravila* bez zaustavljanja (prekida rada) sistema.
Gore navedene osobine u današnjim modernim sistemima i aplikacijama jako su tražene. Sistemi zasnovani na mašini pravila nalaze svoju primenu pre svega zbog brzine, čitljivosti koda i male „nepristupačnosti" (en. *downtime*) sistema. U daljem tekstu biće prikazano kada treba integrisati mašinu pravila u okviru sistema (aplikacije), a kada je ona suvišna i ne treba razmatrati njenu implementaciju.

6.1.2 Kada izgraditi sistem koji je zasnovan na mašini pravila

Najkraći odgovor na pitanje kada zasnovati sistem koristeći mašinu pravila i integrisati je u okviru nekog sistema je: „kada ne postoji zadovoljavajući tradicionalni (imperativni ili objektno orijentisani) programerski pristup za rešavanje nekog problema". Problem ne mora da bude veoma kompleksan. Može biti takve prirode da objektno orijentisani pristup ili imperativni pristup podacima ne dovodi na jednostavan način do rešenja, bez dodatnog gomilanja koda i proširenja modela samih podataka.
Dobar izbor da se sistema bazira na korišćenju mašine pravila je: ukoliko su sistemi i aplikacije koje se razvijaju i koriste veoma dinamične, tj. ukoliko dolazi do česte promene njihovih logičkih delova. Sama logika može biti i veoma jednostavna, ali se pravila često menjaju. Upravo zbog razdvajanja logike od samih podataka kod ovakvih sistema mogu se dodavati nova pravila bez ikakvih posledica na sam sistem. Na ovaj način ostvaruje se veća *agilnost* sistema i otpornost na pojavu greške, čime se omogućava bezbednija nadogradnja i izmena softvera.

Stručnjaci (eksperti) iz nekog domena oblasti mogu biti vrlo dostupni i otvoreni za saradnju, ali nisu tehnički previše pismeni. Oni često poseduju bogato znanje o poslovnim pravilima, procesima i veoma su stručni u nekoj oblasti. Međutim, eksperti nisu tehnička lica, tako da im je strano ponašanje sistema i sama kodifikacija podataka (formalna logika, programiranje). Mašina pravila omogućava da izraze logiku u svojim uslovima, kodifikaciju poslovnog znanja u pravila koja su čitljivija, razumljivija i bliža govornom jeziku eksperta.
Obično u modernim objektno orijentisanim aplikacijama i drugim sistemima ukoliko postoji želja da se uključi mašina pravila, najbolje je da ona sadrži ključne delove poslovne logike kako bi se pojednostavio tradicionlni način objektno orijentisane paradigme. Na prvi pogled može se učiniti da je ovo inverzno razmišljanje od objektno orijentisanog, koje zahteva enkapsulaciju logike u svojim objektima. Zapravo, ovo ne predstavlja izbacivanje objektno orijentisane prakse. U realnim (praktičnim) uslovima implementacije savremenih aplikacija, poslovna logika je samo jedan deo celokupnog sistema.
Često se može primetiti kako se javlja potreba za mnogo uslovnih izjava (*if, else, switch*) i druge „prljave" logike. Dobra praksa je ovakve delove izdvojiti kao zasebne celine (en. *patern*) i premestiti iz funkcionalnih delova koda. Upravo pravila omogućavaju odvajanje i predstavljanje ovakve logike. Takođe, ukoliko postoji logika u sistemu ili aplikaciji koja se često popravlja i koja se stalno, iznova i iznova modifikuje, treba razmisliti o uvođenju mašine pravila. Ukoliko se u sistemu javljaju složeni problemi za koje ne postoje algoritmi i obrasci izvršavanja, treba razmotriti o uvođenje pravila.
Pored mogućnosti integracije mašine pravila u okviru neke aplikacije ili sistema, postoji mogućnost da mašina pravila funkcioniše kao zasebna celina, odnosno servis. Iako je dosta bolja implementacija mašine pravila u okviru nekog sistema, servis zasnovan na mašini pravila može veoma uspešno da funkcioniše. Postoje mnogobrojni praktični primeri. Ovi servisi su veoma pouzdani i skalabilni.

Prilikom održavanja sistema zasnovanih na pravilima, potrebno je odlučiti ko je osoba zaduž-ena za ažuriranje baze pravila. Ovo je jedan od najvažnijih, ako ne i najvažniji proces u radu mašine pravila. Opcija je mnogo, ali različite organizacije imaju različite zahteve. Često za održavanje samih pravila nisu zaduženi proizvođači sistema i programeri projekta, već korisnici sistema ili eksperti neke oblasti. Ovakav pristup je dobar zato što ljudi koji koriste sistem organizuju bazu pravila onako kako njima odgovara, a samim tim više su upoznati sa principom rada sistema. Loša strana je nestručnost kadrova i marljivost ljudi, pa ukoliko se baza blagovremeno ne ažurira i održava, raste broj nepotrebnih pravila. Zato je dobro imati kombinovani pristup, odnosno da graditelji sistema s vremena na vreme učestvuju u održavanju baze pravila.

6.2 Mane ekspertskih sistema zasnovanih na mašini pravila

Često implementacijom mašine pravila u složenije sisteme i aplikacije ljudi zaboravljaju da je mašina pravila samo deo nekog kompleksnog sistema ili rešenja. Nisu namenjene za upravljanje procesom rada i nisu dizajnirane i namenjene kao alat i podsistem za generisanje nekih novih mašina pravila. Mašine pravila su dinamični delovi sistema. Dinamični u smislu da se pravila mogu čuvati i ažurirati nezavisno od podataka. Zato se često koriste za rešavanje problema u „podizanju"(en. *deploy*) aplikacija na nekom serveru. Ukoliko je ovo razlog izbora mašine pravila, treba dobro razmisliti. Mašina pravila najbolje radi kada je potrebno napisati deklarativan program.
Ključnom manom sistema izgrađenih na mašini pravila može se smatrati to što je zasnovana na strogoj listi (skripti) pravila. Loša organizacija, nestručno i neefikasno vezivanje pravila može dovesti do sporog izvršavanja aplikacije (raste složenost u toku vremena), pa i do scenarija

promene ponašanja aplikacije, odnosno neupotrebljivosti sistema. Takođe, do sporog izvršavanja pravila može se doći i ukoliko postoji veliki broj zastarelih podataka (pravila). Zato je potrebno povremeno vršiti kontrolu pravila i blagovremeno odbaciti suvišna i zastarela pravila. Treba još napomenuti da kao i kod pisanja proceduralnih programa, gde je poznato da postoji „čvrsta veza"(en. *tight coupling*[6]) i „labava veza"(en. *loose coupling*) između podataka, tako isto važi i za pravila. Generalno se smatra da je „labava"(en. *loose*) ili „slaba"(en. *week*) spojnica poželjnija u smislu dizajna, jer omogućava dodatnu fleksibilnost samog koda. „Čvrsta spojnica" u sistemima zasnovanim na mašini pravila označava da „okidanje" jednog pravila povlači neko drugo pravilo itd. Drugim rečima, postoji jasan lanac izvršavanja pravila (verovatno očigledan). Dakle, ukoliko postoji „čvrsta veza" između pravila sistem je slabo fleksibilan. Ovo ne znači da je „čvrsta veza" između pravila loša, ali treba imati na umu pri implementaciji sistema. Sa druge strane, „slaba veza" između pravila omogućava jednostavniju promenu pravila, odnosno dodavanje ili uklanjanje nekog od pravila ne bi trebalo suštinski da utiče na neka druga pravila, a samim tim i na funkcionisanje sistema.

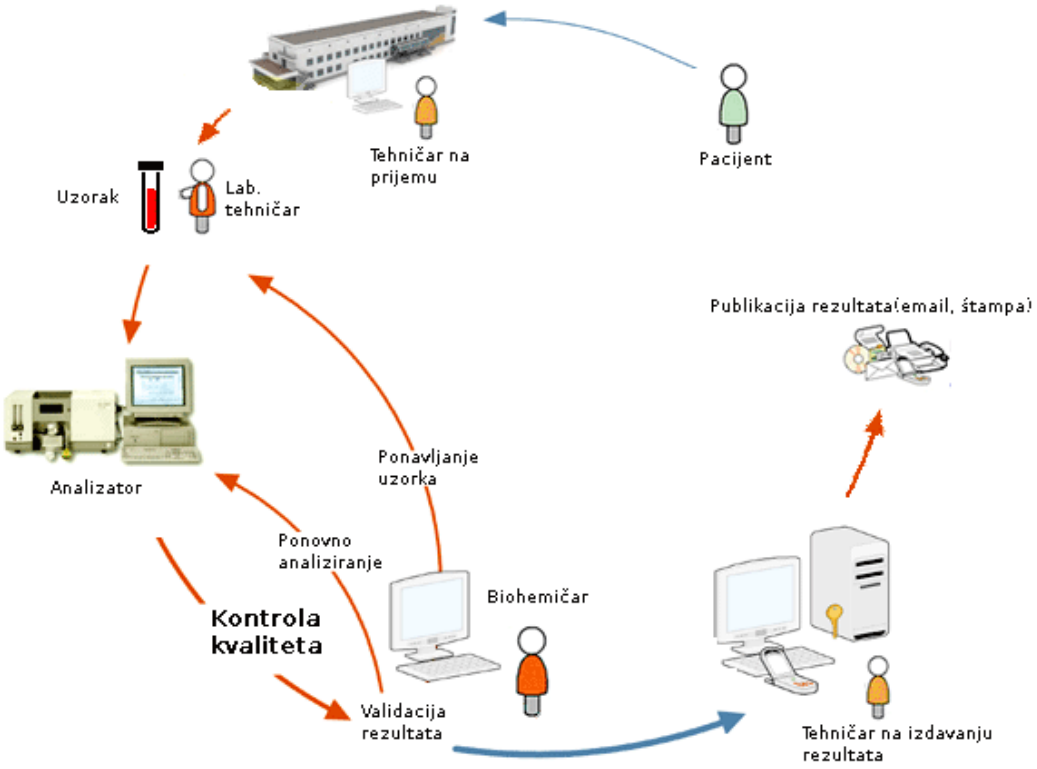
[6]coupling - veza, spojnica

7 Mašina pravila u LIS-u

Nakon teorijskog dela koji je vezan za ekspertske sisteme, tj. za mašinu pravila (produkcioni ekspertski sistem) kao centralne teme ovog rada i nakon predstavljanja osnovnih delova sistema zasnovanih na pravilima, opisana je praktična primena jednog ovakvog sistema, načina kako se mašina pravila ugrađuje u okviru zdravstvenog informacionog sistema, odnosno implementacija u informacioni sistem za mikrobiološke laboratorije.

7.1 Laboratorijski informacioni sistem

Laboratorijski informacioni sistem (LIS) je softversko rešenje za unapređenje procesa rada u biohemijskoj laboratoriji. Omogućava dvosmernu komunikaciju sa uređajima za analizu uzoraka u laboratoriji. Na osnovu uzoraka pacijenata koji se postavljaju u analizator[7], dobijaju se rezultati pacijenata u sistemu. LIS se razvija sa ciljem da olakša, ubrza i poveća pouzdanost obrade uzoraka i izdavanja rezultata, eliminiše greške identifikacije pacijenta, pomaže u praćenju statusa uzorka u realnom vremenu, omogućava kontrolu analizatora i praćenje istorije rezultata pacijenata, pruža mogućnost za uređivanje testova i referentnih vrednosti, vrši kontrolu prijema pacijenata i olakšava uređivanje podataka i analiza vezanih za rad u laboratoriji.



Slika 11: Laboratorijski informacioni sistem

[7]Aparat koji vrši analizu uzoraka (krv, urin, bris).

Upravo zbog ogromne količine podataka, ali pre svega i zbog velike važnosti podataka (rezultata uzoraka), greške u ovakvim sistemima mogu biti veoma opasne, pa je potrebno organizovati sistem tako da prostor za pojavu grešaka ne postoji. Potrebno je vršiti stalnu kontrolu sistema. Vremenom zbog mnogih faktora (amortizacije, prašine, kvaliteta seruma, vlage, toplote...) analizatori, kao i ostali merni uređaji, dovode do loših, nepreciznih, netačnih, nevalidnih rezultata uzoraka, ali i do drastičnih grešaka koje mogu uticati na rad u samoj laboratoriji i biti opasne po zdravlje i život ljudi. Kod obezbeđivanja kvaliteta krajnjeg rezultata važno je da se zna kakve su potrebe i zahtevi korisnika tog rezultata i da se ti zahtevi implementiraju u proces rada. Ovo je važno imati na umu jer korisnik (pacijent, lekar, laborant) posmatra rezultat sa medicinske tačke gledišta, posmatra ga u odnosu na referentne vrednosti ili u odnosu na raniju istoriju rezultata, a ne razmišlja o rezultatu sa aspekta nepreciznosti i netačnosti. Pored same kontrole kvaliteta i skupa pravila koja su vezana za njih, oblast u okviru laboratorije u kojoj mašina pravila nalazi svoje široke primene jeste mikrobiologija.
Kako je mikrobiologija jedna od najzahtevnijih, a pre svega veoma osetljiva oblast u procesu rada u samoj laboratoriji, zasnovana je na „poštovanju i ispunjavanju" liste propisanih pravila koja mogu povremeno da se menjaju i dopunjuju, mašina pravila je idealni sistem za praktičnu primenu i implementiranje u informacioni sistem laboratorije. Najbolji način da se opiše i razume kako mašina pravila funkcioniše u LIS-u jeste samo opisivanje procesa rada u mikrobiologiji. Prvo je potrebno dati širu sliku o samoj mikrobiologiji, navesti neke osnovne termine iz ove oblasti, kao i načine za predstavljanje i organizaciju podataka, a zatim definisati pravila koja se koriste da opišu sistem i donesu željeno rešenje.

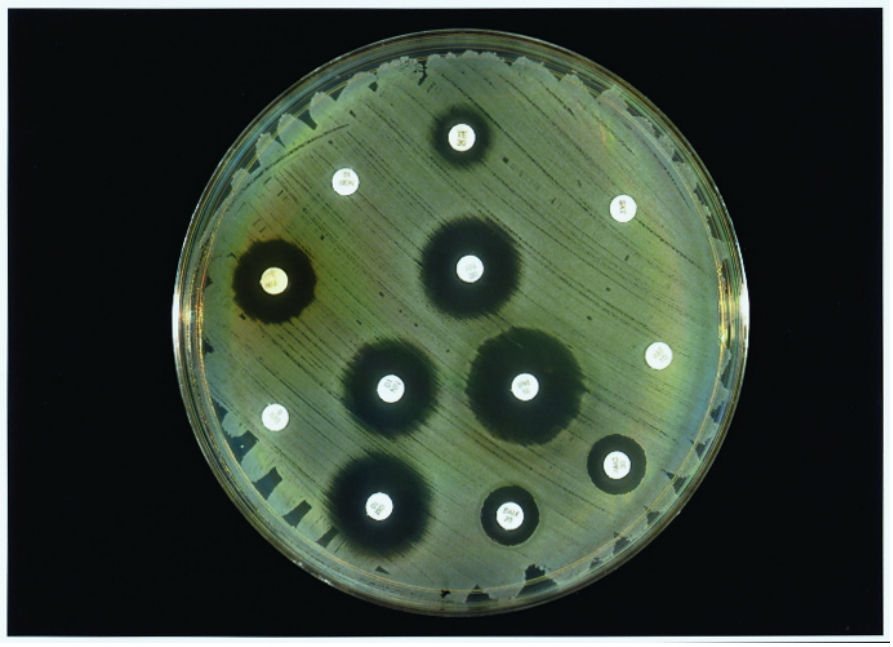
7.2 Ukratko o mikrobiologiji i osnovni pojmovi

Mikrobiologija je grana medicine koja se bavi proučavanjem, prevencijom, dijagnostikom i lečenjem zaraznih bolesti. Osim pomenutog, ovo polje nauke bavi se i proučavanjem različite kliničke primene mikroorganizama za poboljšanje zdravlja ljudi. Postoje sledeće vrste mikroorganizama koji izazivaju zarazne bolesti:

- bakterije - jednoćelijski mikroorganizmi koji žive samostalno ili u raznim grupacijama ili kolonijama;
- gljivice - jednoćelijski i višećelijski organizmi, po broju vrsta spadaju među najrasprostranjenije organizme na Zemlji;
- paraziti - organizmi koji se hrane na račun drugog organizma (nazvanog domaćin) u toku dužeg vremenskog perioda;
- virusi - acelularni-nećelijski, ultramikroskopski mikrorganizmi nesposobni da se razmnožavaju van ćelije domaćina;
- prioni - posebni oblici proteina koji mogu izazvati neke bolesti kod ljudi i životinja. Nastaju mutacijom gena koji kodira jedan protein ljudskog tela „prion protein".

Nabrojani mikroorganizmi koji mogu da izazovu bolest nekog drugog organizma nazivaju se *patogenim*. Prema patogenim osobinama ili riziku po zdravlje drugih organizama svi mikroorganizmi se mogu grupisati na [12]: mikroorganizme sa niskim ličnim i društvenim rizikom, mikroorganizme sa srednjim ili ograničenim ličnim i društvenim rizikom, mikroorganizme sa visokim ličnim i društvenim rizikom i mikroorganizme sa visokim rizikom koji uzrokuju često neizlečive bolesti.

Mikrobiolog kao osoba koja je odgovorna za rad u laboratoriji proučava karakteristike patogena, njihove načine prenošenja, mehanizme infekcija i razvoja. Osnovni zadatak mikrobiologa, pored same detekcije mikroorganizama i njegove dalje izolacije, jeste i priprema izveštaja koji kasnije lekar može upotrebiti u lečenju pacijenta. Kada je penicilin prvi put primenjen u medicini skoro sve vrste stafilokoka bile su osetljive na ovaj antibiotik. Danas postoji više od 70% vrsta mikroorganizama na koje penicilin ne utiče. Zbog povećanja broja bakterija na koje antibiotik nema dejstvo potrebno je ispitati osetljivost patogena na određene tipove antibiotika. Metoda za određivanje osetljivosti bakterija prema antibiotiku naziva se antibiogram.
Antibiogram je rezultat ispitivanja osetljivosti nekog izolovanog patogena na različite antibiotike. Za pravljenje antibiograma neophodno je najpre iz kliničkog uzorka (krv, urin, bris grla, uha, oka) izolovati i identifikovati patogen, a zatim u čistoj kulturi dobijenoj presejavanjem prikladnom tehnikom, izvesti ispitivanje osetljivosti. Nakon izolacije organizma na pločici (disku), postavljaju se kružići filter papira (antibiogram tablete) koji sadrže određene vrste antibiotika. U procesu inkubacije (18 sati na temperaturi od 37 C^o) antibiotici „difunduju"(razgrađuju se) sa filter papira na pločicu sa bakterijama u koncentričnim krugovima. Koncentracija antibiotika opada sa udaljavanjem od filter papira. Nakon inkubacije, formira se gust bakterijski tepih na kome se oko filter papira uočavaju zone odsustva rasta. Zona inhibicije (smanjenje pojave bakterija) biće manja ukoliko su bakterije manje osetljive na dati antibiotik i obratno. Na slici 12 prikazana je jedna takva pločica sa izolovanim mikroorganizmom (streptokoka) i antibioticima na odgovarajućim mestima.



Slika 12: Reagovanje patogena na određene antibiotike

Kao što se može videti na slici 12, patogen različito reaguje na određeni tip antibiotika. Na osnovu uzorka kreira se antibiogram, tako što se meri prečnik zone inhibicije rasta bakterije oko filter papira i upoređuje se sa standardnim vrednostima prema kojima se propisuje da li je neki mikroorganizam osetljiv ili otporan. Antibiogram sada predstavlja vektor parova antibiotik - rezultat, gde se kao rezultat mogu naći sledeće vrednosti:

- reizistentan, otporan (R) - ukoliko antibiotik ne utiče na zadati patogen;
- intermedijeran, prelazno (I) - ukoliko antibiotik delimično utiče na patogen;
- senzitivan, osetljiv (S) - patogen je osetljiv na zadati antibiotik.

Tako dobijeni antibiogram je rezultat ispitivanja uticaja antibiotika na zadati patogen i u odgovarajućem formatu prosleđuje se dalje lekaru ili nekom drugom odgovornom licu koje postavlja dijagnozu i definiše način lečenja.

7.3 Apstrakcija rešenja i tehnički detalji sistema

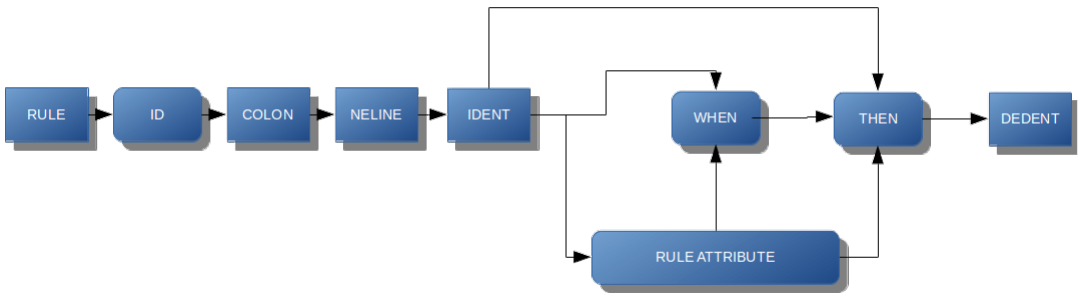
Krajnji rezultat proučenog uzorka (patogena) jeste kreiranje antibiograma sa vrednostima R, I i S. Međutim, predstoji čitav jedan proces pre samog generisanja antibiograma, a to je detektovanje i analiza nalaza. Upravo u ovoj fazi može se videti uloga mašine pravila, koja primenjući definisana pravila i činjenice, pomaže korisniku da u nizu koraka detektuje određeni mikroorganizam i predlaže najbolje moguće rešenje. Naravno, postoji mogućnost da sistem ne dođe do konačnog odgovora, što će biti opisano u daljem tekstu. Takođe, biće prikazano i šta treba raditi u takvim situacijama.

Pre same priče kako sistem funkcioniše i kako se realizuju i definišu pravila, treba napomenuti i tehničke detalje vezane za izgradnju aplikacije (sistema). Celokupni laboratorijski informacioni sistem izgrađen je u programskom jeziku *Python 2.7* [13], za bazu podataka izabrana je *Postgre 9.2* [14], a korisnički interfejs i proces komunikacije krajnjih korisnika sa sistemom, implementiran je korišćenjem Veb tehnologija (*HTML5, CSS-Bootstrap i AngularJS*) [15]. Ovakav izbor programske paradigme (objektno orijentisane) ostvaruje mogućnost lake integracije sa različitim sistemima zasnovanim na pravilima, omogućava predstavljanje pravila pomoću objekata (klasa), kao i korišćenje O-A-V zapisa, koji su opisani u nekim od prethodnih poglavlja.
Mašina pravila, deo sistema koji služi za upravljanje procesom zaključivanja, odnosno za pronalaženje i izvršavanje (okidanje) željenog pravila (iz baze pravila), implementirana je koriš-ćenjem *Python* biblioteke pod nazivom *Intellect* [16]. *Intellect* je biblioteka „otvorenog-koda"(en. *open-source*). Podržava sintaksu zasnovanu na pravilima (en. *Drools syntax*), odnosno pruža mogućnost pravljenja produkcionih sistema zasnovanih na pravilima. Zbog jasnoće (čitljivosti) pravila (kako autor biblioteke *Intellect* naglašava) umesto *IF* dela pravila koristi se *WHEN*, ali osim sintaksne razlike ova promena ne predstavlja nikakva dodatna odstupanja od značenja i izvršavanja pravila. Pravila se zapisuju pomoću objekata (klasa). Svako pravilo sadrži atribute. Pravilo se aktivira ako svi njegovi uslovi odgovaraju činjenicama iz liste promenljivih koje su navedene kao atributi. Aktivirano pravilo se smešta u agendu. Agenda je lista pravila čiji su uslovi zadovoljeni (aktivirana pravila), a koja još uvek nisu izvršena. *Intellect* pored predstavljanja pravila oblika *when ... then*, omogućava definisanje i navođenje samih agendi *agenda-group*. *Agenda-group* sadrži atribut (*ID pravila*), gde se navodi naziv agende i ovom komandom se označavaju pravila koja će se sigurno izvršiti ukoliko činjenica sadrži „naziv agende", odnosno ID pravila.

Primer: Pravilo koje ispituje da li atribut ocena, objekta Test ima vrednost 1, 2, 3, 4 ili 5, i ispisuje kao rezultat na standardnom izlazu „Interval je uspešno pronađen!”. Sledi prikaz pravila i sintaksni dijagram u sistemu Intellect.

```
from intellect.testing import Test

rule "da li ocena ima vrednost u intervalu od 1 do 5":
  when:
    $test := Test(ocena in [1,2,3,4,5])
  then:
    print "Interval je uspešno pronađen!"
```



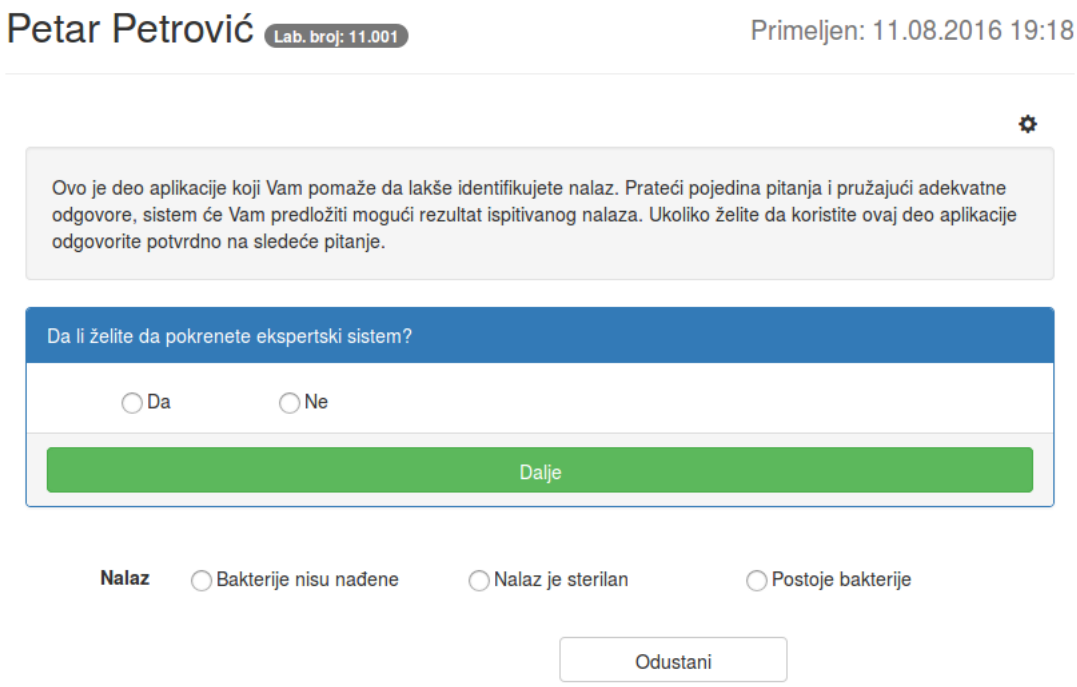
Slika 13: Sintaksni dijagram pravila u sistemu *Intellect*

Osim gore navedenih ključnih reči, *Intellect* sadrži i još neke rezervisane naredbe koje se nazivaju još i *akcije* (en. *action*). Ovaj skup naredbi je neophodan za funkcionisanje samog procesa zaključivanja. Najčešće korišćene naredbe (akcije) su:

- resetovanje pravila (*reset*);
- ponovno učitavanje novih ili istih pravila (*learn*);
- modifikovanje pravila (*modify*);
- naredba za „zaboravljanje” određene grupe pravila (*forget*);
- naredba za zaustavljanje procesa zaključivanja (*halt*).

Nabrojane *akcije* u sistemu *Intellect* neće biti posebno objašenjene i definisane, već će njihovo značenje i primena biti prikazani prilikom opisa funkcionisanja sistema i predstavljanja samih pravila, kroz praktične primere.

Sa druge strane, iz ugla krajnjeg korisnika, nakon procesa kreiranja uputa, prikupljanja potrebnih podataka i uzimanja određenog uzorka za testiranje (bris grla, bris rane, bris oka itd.), prelazi se na korak detekcije uzorka. Na korisničkom interfejsu pojavljuje se opcija za pokretanje mašine pravila[8]. Ukoliko krajnji korisnik izabere opciju za pokretanje mašine pravila, sistem se inicijalizuje. Ispisuje se pozdravna poruka, sa potrebnim informacijama i korisnik se odlučuje da li želi da koristi sistem u daljem radu. Ukoliko je odgovor krajnjeg korisnika potvrdan, predstoji niz pitanja. Odgovarajući na pitanja, korisnik prenosi svoja zapažanja vezana za nalaz (patogen) i pomaže mašini pravila da kombinacijom činjenica i pravila dođe do željenog rešenja (zaključka). Na slici 14 prikazan je izgled korisničkog interfejsa u inicijalnoj fazi mašine pravila (*Intellect*).



Slika 14: Prikaz interfejsa sistema zasnovanog na pravilima u mikrobiologiji - inicijalna faza

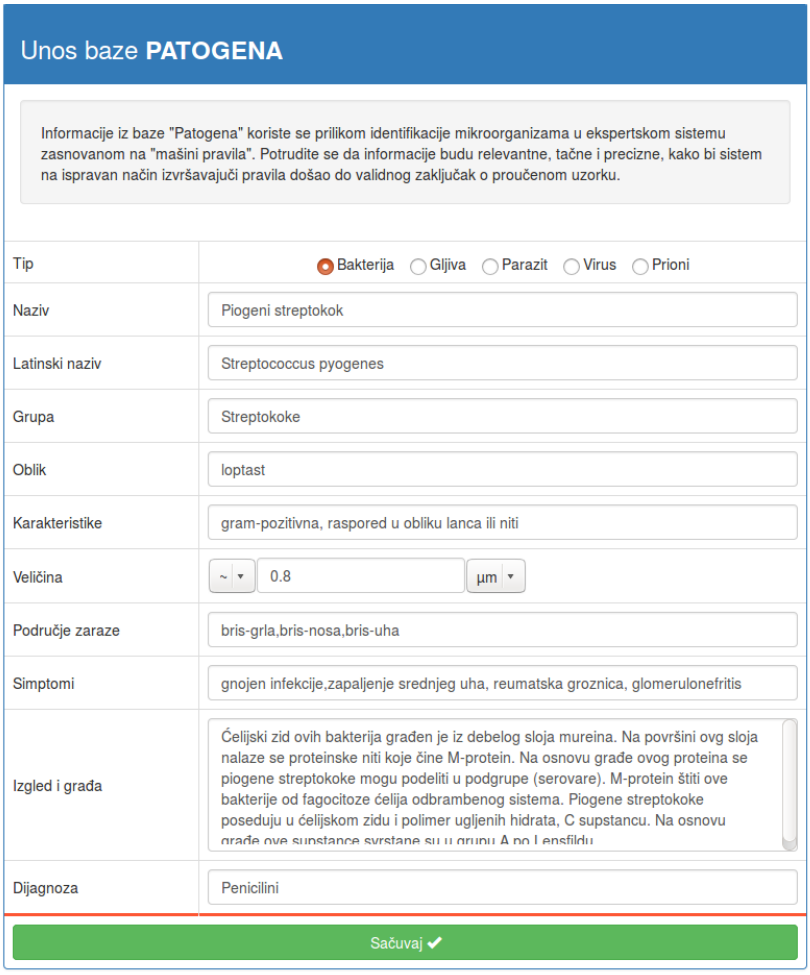
[8]Zbog prisustva mikroobiologa, ponekad krajnji korisnici ne žele da pokrenu mašinu pravila, pa je ostavljena mogućnost izbora.

7.4 Kako *Intellect* - mašina pravila funkcioniše u LIS-u

Sve što je potrebno da sistem „radi i funkcioniše", jeste primena pravih podataka u pravom trenutku. Zvuči dosta jednostavno. Najvažnije je organizovati podatke (činjenice i pravila) na pogodan način, a zatim samo primeniti tehniku (algoritam) zaključivanja, dodati heuristiku i u određenim delovima uključiti pomoć krajnjeg korisnika.

7.4.1 Opis i organizacija podataka u sistemu

Podaci potrebni za funkcionisanje sistema mogu se podeliti u dve osnovne grupe: baza činjenica i baza pravila (poglavlje 6). Baza činjenica ili globalna baza podataka, sadrži sve potrebne podatke neophodne za proces zaključivanja, a to su lista bakterija sa njihovim karakteristikama (veličina, oblik, simptomi itd.), podaci o pacijentu, uzorcima, antibioticima i ostali podaci nad kojima se primenjuju pravila. Ove podatke unose krajnji korisnici sistema preko odgovarajućeg korisničkog interfejsa (Slika 15).



Slika 15: Korisnički interfejs za unošenje baze činjenica (podataka)

Prilikom pokretanja mašine pravila ovi podaci se dovlače u radnu memoriju i smeštaju u odgovarajuće strukture podataka. Većom količinom podataka u bazi činjenica proširuju se i opcije mogućih rešenja. Opisom novih činjenica sistem „uči". Činjenice se opisuju jezikom krajnjih korisnika. Što je ova baza tačnija i preciznija, sistem je kvalitetniji i pouzdaniji. Rezonovanjem činjenica i primenom odgovarajućih pravila dolazi se do zaključka, odnosno krajnjeg rešenja.

Baza pravila (*rules.policy*) sadrži spisak pravila predstavljenih pomoću objekata (O-A-V). Izgled, opis i struktura pravila biće prikazana u narednom poglavlju (7.4.2). Bazu pravila uglavnom definišu graditelji ekspertskog sistema u saradnji sa ekspertima neke oblasti (mikrobiolozima i biohemičarima), a nešto ređe i sami eksperti. Predstavlja deklarativni opis znanja nekog eksperta. Od kvaliteta podataka, odnosno pravila zavisi brzina rada sistema, kao i izbor sledećeg koraka prilikom prikupljanja podataka iz baze činjenica. Uvećanjem ove baze i nepoštovanjem osnovnih principa za predstavljanje znanja u mašini pravila, dolazi do sporijeg rada sistema. Potrebno je imati manji broj kvalitetnijih i sadržajnijih (preciznijih) pravila. Okidanjem pravila sistem izvršava određenu (opisanu) akciju i prelazi na naredni korak. Ukoliko nema više raspoloživih podataka sistem završava rad i predstavlja najbolje moguće rešenje.

7.4.2 Definisanje, struktura i primena pravila

Kako je već bilo dosta reči o samim pravilima, njihovom izgledu, značenju i osobinama u okviru ekspertskih sistema i uopšteno, u ovom poglavlju biće prikazana određena grupa pravila (praktična primena) koja se koriste u procesu zaključivanja u okviru LIS-a. Definisana pravila predstavljaju znanje eksperta (mikrobiologa) i opisuju metode, niz koraka koje mikrobiolog primenjuje prilikom detekcije nekog patogena. Prikazom samih pravila biće opisana njihova struktura, atributi koje pravila sadrže, kao i način kada se primenjuju i koji je rezultat primenjenih pravila[9]. Sva pravila sadrže ključnu reč *rule*, dok agende počinju sa *agenda-group*. Nakon ključne reči sledi naziv pravila, a zatim i sama pravila u O-A-V obliku. Objekti sadrže niz atributa, koji proširuju samu sintaksu pravila. Pravilo se izvršava ukoliko su sve vrednosti u *when* delu zadovoljene. Sledi kratak prikaz osnovne grupe pravila.

Pravilo koje opisuje građu mikroorganizma, odnosno tri osnovna tipa u izgradnji mikroorganizma koja se mogu uočiti. Takođe, postoji mogućnost nejasnog (sterilnog) nalaza, pa je ostavljen prostor i takvog odgovora. Ovo pravilo se izvršava („okida") u nekom od prvih koraka prilikom same inicijalizacije sistema.

```
rule gradja_mikroorganizma:
  when:
    $question := Engine(type_=="rule" and answer == "Da")
  then:
    $question.set_up(
      "radio", "Na disku (ploči) uočavate organizme?",
      ["jednoćelijske", "bezćelijske", "višećelijske", "nejasno"]
    )
```

[9]Izdvojena je samo određena grupa pravila kao reprezentativni primer, dok su ostala pravila slična navedenim i funkcionišu na identičan način.

Pravilo koje zaustavlja ekspertski sistem u slučaju greške ili nekog neželjenog stanja. Pravilo u svom *then* delu sadrži i akciju *halt* koja označava završetak zaključivanja u sistemu *Intellect*.

```
rule prekid_zakljucivanja:
    when:
        $question := Engine(type_=="rule" and
        answer == "Ne" or error == True)
    then:
        $question.set_up("exit", "Odgovor",
        "Ekspertski sistem je zaustavljen.
        Pokušajte detekciju ručnim putem. Hvala!")
        halt
```

Sledeće pravilo opisuje oblik mikroorganizma, konkretno oblik bakterije. Sve bakterije imaju neki od navedenih oblika. Takođe, može se zaključiti da je reč o bakterijama, jer do pravila „oblik_bakterije" dolazi se nakon koraka koji se odnosi na gram-pozitivne, odnosno gram-negativne bakterije.

```
rule oblik_bakterije:
  when:
    $question := Engine(
      type_=="rule" and
      answer == "gram-pozitivne" or answer == "gram-negativne")
  then:
    $question.set_up(
      "radio", "Označite oblik uočenog mikroorganizma?",
      ["loptast(krug)","štapićast","izvijen","končast"]
    )
```

Način kako se definišu agende. Agenda, kao što je već naglašeno, predstavlja pravilo koje se izvršava bez ispitivanja njegove validnosti. Agenda postavlja promenljivu *gram_positive* na False, odnosno označava gram-negativne bakterije i izvršava *loader* koji primenjuje ovo pravilo na sve gram-negativne činjenice.

```
rule gram_negativne:
    agenda-group "crvena"
    then:
        gram_positive = False
        loader(gram_positive)
```

Pravilo upoređuje veličinu mikroorganizma. Ukoliko je veličina veća od zadate, vrši se filtriranje činjenica i poziva metoda *resolver*.

```
rule velicina_mikroorganizma:
  when:
    $question := Engine(
      type_=="rule" and greaterThan(size))
  then:
    filter(size)
    resolver(size)
```

Veoma značajno pravilo koje služi za otklanjanje konflikata. Konflikti se mogu javiti zbog mogućnosti ponavljanja pravila, loše definisanosti, kao i zbog povećanja složenosti sistema. Mehanizam zaključivanja vraća se korak unazad, akcijom *modify* vrši se modifikacija pravila i ponovo se učitavaju činjenice, ali sada na modifikovanoj grupi pravila.

```
rule razresi_konflikte:
  when:
    $question := Engine(
      type_=="rule" and
      answer == "" and conflict == confilct.exist)
  then:
    rollback(rule)
    modify Engine(conflict)
    loader()
```

Prikazana grupa pravila predstavlja osnovne i najviše korišćene tipove pravila u inpementaciji rešenja za laboratorijski informacioni sistem. Ostala pravila izgrađena su na sličan način, proširujući listu atributa, ili su izvedena nekom od navedenih kombinacija, primenjujući logičke veznike (*and, or*). Suštinski sva pravila izgrađena su na osnovu nekog od gore prikazanih modela.

7.4.3 Mehanizam zaključivanja mašine pravila u LIS-u

U procesu zaključivanja prilikom izbora pravila, *Intellect* koristi *hibridnu* tehniku, odnosno kombinaciju dva napomenuta algoritma *ulančavanje unapred* i *ulančavanje unazad*. Nakon inicijalizovanja mašine pravila, baza činjenica dovlači se u radnu memoriju. Zbog brzine rada aplikacije vrši se filtriranje određene grupe činjenica, a zatim se pronalaze sva primenljiva pravila. Korisniku se postavlja određena grupa pitanja (definisanih pravilima) i na osnovu odgovora sužava se izbor odluke u narednim koracima. Ova pitanja su neophodna zbog nemogućnosti ekspertskog sistema da analizira izgled patogena. Odgovori na pitanja predstavljaju korisnikovo zapažanje posmatranog patogena kroz mikroskop ili analizator (veličina, oblik, reakcija na UV itd.). Mašina pravila odgovore spaja sa definisanim činjenicama i primenjujući pravila određuje naredne korake. Primena narednog pravila zavisi od činjenica i odgovora korisnika. Ukoliko primenjena pravila nemaju dalju ulogu u radu, ona se odbacuju. U svakom sledećem koraku sistem „uči". Na taj način sistem sužava izbor pitanja i pokušava da svede na minimalni broj koji je potreban za realizaciju rešenja. Kao što je već napomenuto, sistem takođe primenjuje grupu agendi u procesu zaključivanja. Ovde su agende definisane kao pravila koja važe za sve mikroorganizme. Kada sistem detektuje agendu, ona se izvršava bez dalje provere o njenoj validnosti. Osim već pomenutih pravila, veoma bitnu ulogu u povezivanju laboratorijskog informacionog sistema i mašine pravila imaju objekti (klase). Komunikacija između mašine pravila i aplikacije koja prikuplja odgovore korisnika ostvarena je implementacijom dve osnovne klase:

- *class Supervisor(object)*;
- *class Engine(object).*

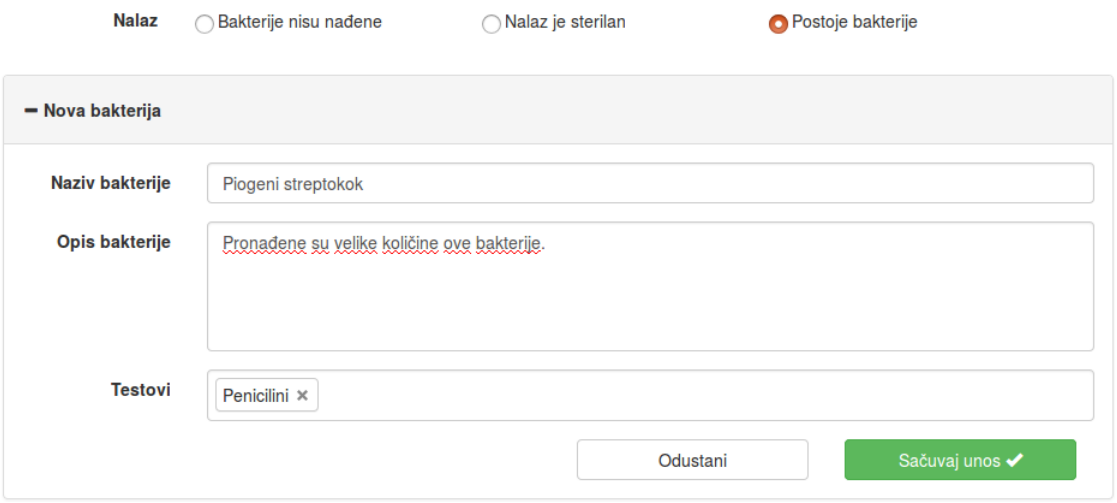
Klasa *Supervisor* sadrži metode koje vrše interakciju, odnosno služe za postavljanje određene grupe pitanja i prikupljanje odgovora krajnjih korisnika na postavljena pitanja. Metode *resolve_question* i *resolve_answer* na osnovu „zahteva" (en. *request*) prepakuju odgovore korisnika i kreiraju instance klase *Engine* potrebne u procesu zaključivanja. Objekti klase *Engine* prosleđuju se mašini pravila koja u odgovarajući atribut - *answer* smešta rezultat zaključivanja dobijen u izvršenom koraku. Ukoliko neko od definisanih pravila ponovo zahteva ulogu krajnjeg korisnika,

pitanje se definiše na osnovu činjenica i pravila. Metoda *resolver* potom spakuje željeno pitanje i preko metoda klase *Supervisor* prosledi do krajnjeg korisnika. Klasa *Engine* sadrži i atribut *answers* gde se korisniku definišu neka od ponuđenih rešenja (odgovora) koja su unapred zadata, ili sistem, na osnobvu prethodnih slučajeva ili heuristika, zaključi da je to najefikasniji odgovor u datom koraku.
Veoma bitna karakteristika u principu zaključivanja kod sistema *Intellect* su akcije *modify* i *forget*. One omogućavaju realizaciju konflikata koji se mogu javiti u procesu zaključivanja. Ukoliko dođe do konflikta *Intellect* prvo primenjuje neko od pravila *modify*, pokušavajući da izmeni pravilo i prilagodi, tako da se izaberu nove činjenice iz baze činjenica. Zatim se primenjuje prvo pravilo čiji je uslov ispunjen na novoučitane činjenice. Ukoliko *modify* ne omogući rešenje javljenog konflikta, primenjuje se neko od pravila koje sadrži *forget* akciju.

Mehanizam za zaključivanje se zaustavlja ukoliko naiđe na akciju *halt* ili ukoliko je skup primenljivih pravila prazan, odnosno ne postoji pravilo koje se može primeniti. Dobijeni zaključak predstavlja najbolje moguće rešenje koje sistem na osnovu primenjenih pravila i odgovora može da izvede kao zaključak.

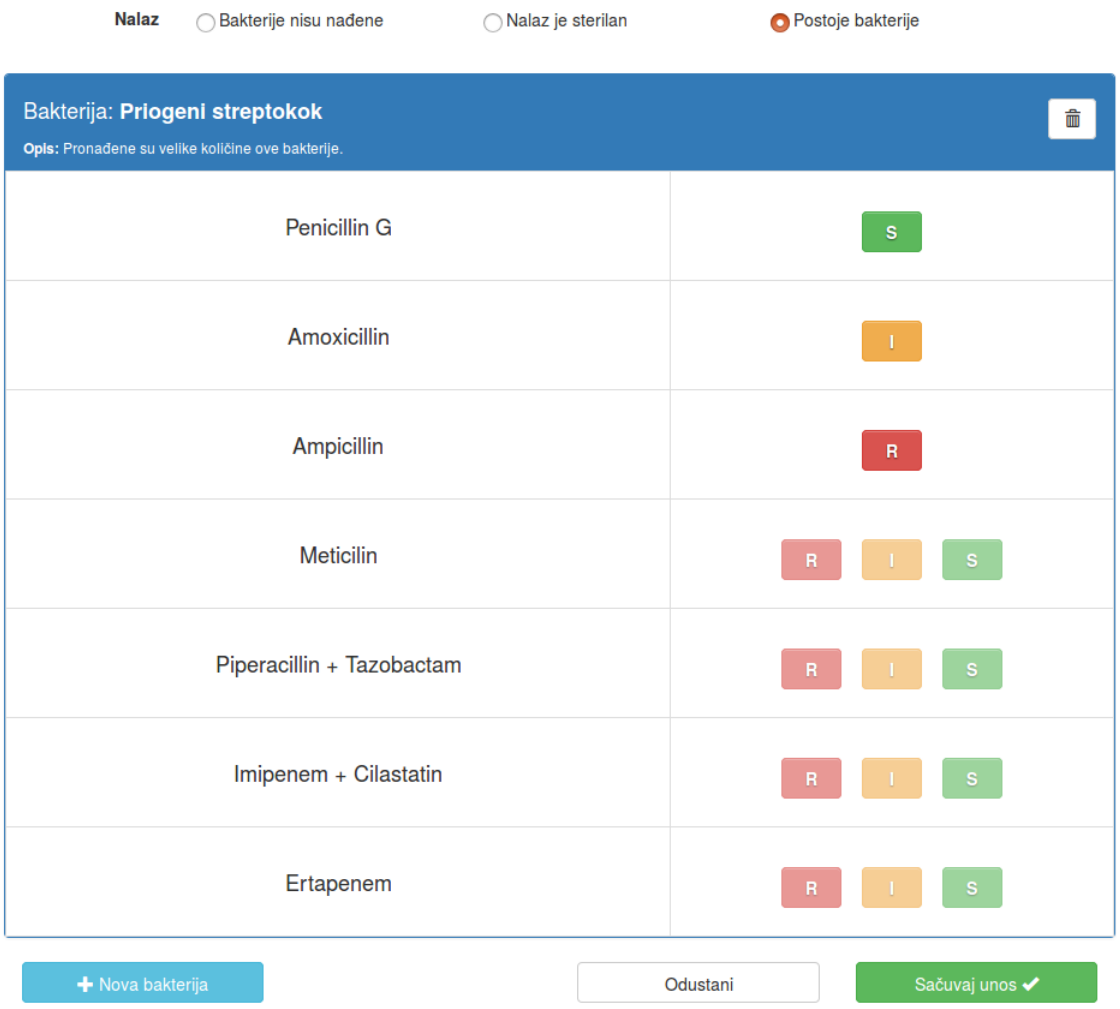
7.5 Antibiogram i predstavljanje rešenja

Nakon opisanih akcija, detekcije mikroorganizma primenom mašine pravila predstoji primena rezultata koji je ekspertski sistem predložio. Ekspertski sistem kao rezultat predlaže određeni tip mikroorganizma i takođe definiše dijagnozu za pronađeni mikroorganizam. Dijagnoza predstavlja niz antibiotika, koji se na osnovu nekog iskustva pripisuju za tu grupu mikroorganizama. Da bi se formirao antibiogram, potrebno je uneti pronađeni mikroorganizam i dodati predložene antibiotike (Slika 16).



Slika 16: Korisnički interfejs: unos mikroorganizma i antibiotika - formiranje antibiograma

Kao rezultat, potrebno je formirati antibiogram, sa vrednostima (R, I, S). Određeni mikroorganizmi različito reaguju na izabgranu grupu antibiotika. Kako je ekspertski sistem predložio grupu antibiotika, potrebno je te antibiotike postaviti na disk i posmatrati reakciju mikroorganizama. Posmatra se zona inhibicije (poglavlje 7.2). Na osnovu dobijenog rezultata popunjava se antibiogram (Slika 17). Antibiogram predstavlja završnu fazu detekcije nekog patogena od strane laboratorijskog informacionog sistema i rezultat antibiograma se prosleđuje dalje lekaru za prepisivanje određene dijagnoze, odnosno grupe antibiotika koji su najbolje reagovali na ispitivani mikroorganizam.



Slika 17: Korisnički interfejs: Antibiogram

8 Zaključak

U ovom master radu data je šira slika o ekspertskim sistemima kao jedne od oblasti veštačke inteligencije koja se sve više razvija i primenjuje. Predstavljen je princip izgradnje ekspertskih sistema, pre svega ekspertskih sistema zasnovanih na pravilima, njihove osnovne komponente, učesnici i karakteristike. Opisana je mašina pravila, sistem koji na osnovu definisanih pravila i činjenica određenom tehnikom (algoritmom) zaključivanja pronalazi najpogodnije rešenje. Treba još napomenuti i *ulančavanje unapred* i *ulančavanje unazad* kao dva najbitnija algoritma u ovim sistemima, gde je kombinacijom ovih algoritama implementiran mehanizam zaključivanja koji se koristi u mašini pravila.

Ekspertski sistem zasnovan na pravilima primenjen je u mikrobiologiji u okviru laboratorijskog informacionog sistema. Opisan je proces detekcije mikroorganizma na osnovu pravila i činjenica i predstavljanje najboljeg rešenja koje će biti uključeno u izgradnji antibiograma. Prikazan je korisnički interfejs za popunjavanje baze činjenica, kao i najčešće primenjivana pravila u sistemu *Intellect*. Kao što se može videti, zbog prirode i kompleksnosti problema za koji je definisan ekspertski sistem (mikrobiologija) i dalje postoji veliki uticaj krajnjeg korisnika (čoveka), a samim tim i veća mogućnost za pojavu greške. Takođe, u određenim koracima bitno je zapažanje krajnjeg korisnika, tj. u zavisnosti od njegovog odgovora sistem donosi zaključak. Potrebno je još i napomenuti da je baza činjenica relativno mala, pa je usled toga predstavljeno krajnje rešenje testne prirode.
Jedan od ciljeva koje u budućnosti sistem treba da omogući jeste da na osnovu slike (fotografije) uzorka (diska), sistem sam prepozna reakciju nekog mikroorganizma na određenu grupu antibiotika i na osnovu rezultata generiše antibiogram. Pored eliminisanja ljudskog faktora, ono što predstavlja bitnu osobinu jeste popunjavanje (proširenje) baze činjenica, ali i preciziranje samih pravila. Potrebno je unaprediti funkcionisanje heuristika prilikom izbora pravila. Uključivanjem više heuristika pri izboru pravila i činjenica znatno će unaprediti brzinu funkcionisanja sistema i smanjiti potrebna pitanja za krajnjeg korisnika. Sve ove navedene mane ostavljaju prostor za dalje usavršavanje i unapređivanje sistema i aplikacije, ali takođe prikazuju i neke od loših osobina ekspertskih sistema. Sa povećanjem kompleksnosti ekspertskog sistema otežava se i njegova integracija, odnosno kombinacija sa drugim sistemima. Raste i složenost problema. Potrebno je dobro definisati sve ključne delove svakog podsistema. Može se reći da veoma važnu ulogu (ako ne i najbitniju) u projektovanju ekspertskog sistema ima interakcija graditelja sistema i eksperta iz oblasti za koji se sistem implementira.

Težnja veštačke inteligencije, a samim tim i ekspertiskih sistema kao podoblasti veštačke inteligencije je da proizvedu računar (mašinu, sistem) koji „razmišlja nalik ljudskom biću". Ovakav pristup ostavlja mogućnost da se potpuno zanemari prisustvo ljudi, tj. da se njihov uticaj svede na minimum, a mašini sa druge strane omogući ogroman prostor i „znanje" za upravljanje bitnim aspektima ljudskih života. Pitanje je vremena kada će upravo ekspertski sistemi odlučivati o terapiji ljudi, finansijskim izveštajima, o budućim koracima u strategiji neke firme, pa i kretanju razvoja dalje civilizacije, a ljude pretvoriti u „zavisnike" koji će pažljivo pratiti i prihvatati zaključke „inteligentnih" sistema.

Literatura:

[1] Predrag Janičić, Mladen Nikolić, „*Veštačka inteligencija*", Beograd (Srbija), 2010.

[2] Christopher Evans, „*The Mighty Micro: The Impact of the Computer Revolution*", New York (USA), 1979

[3] Alempije Veljović, Mirosav Radojičić, Jasmina Vesić, „*Menadžment informacioni sistemi*", Čačak (Srbija), 2008

[4] Lotfi A. Zadeh, „*Fuzzy algorithms, Information and Control*", Brooklyn, NY (USA), 1965

[5] Durkin John, „*Expert Systems - Design and Developmen*", New York, 1994

[6] Efraim Turban, Jay E. Aronson, Ting-Peng Liang, „*Decision Support System And Intelligent System, 7th Edition*", New Delphi (USA), 207

[7] Ajith Abraham, „*Rule-based Expert Systems*", Oklahoma State University, Stillwater (USA)

[8] Efraim Turban, Jay Liebowitz, „*Managing Expert Systems*", Harrisburg, Pennsylvania (USA), 1992

[9] David Heckerman, „*The Certainty-Factor Model*", Los Angeles, California (USA)

[10] Robert S. Engelmore, Edward Feigenbaum, „*The Building Blocks of Expert Systems"*, `http://www.wtec.org/loyola/kb/c1_s1.htm`

[11] Semantičke mreže: `http://www.jfsowa.com/pubs/semnet.htm`

[12] Osnovna podela mikroorganizama (patogena): `https://sr.wikipedia.org/wiki/Patogen`

[13] Zvanična dokumentacija programskog jezika *Python 2.7*: `https://docs.python.org/2/`

[14] Zvanična dokumentacija baze podataka *Postgre*: `https://www.postgresql.org/`

[15] Osnovi Veb programiranja, *w3schools*: `http://www.w3schools.com/`

[16] *Python* biblioteka *Intellect*: `https://pypi.python.org/pypi/Intellect`

[17] Ilustracija ekspertskog sistema: `http://www.igcseict.info/theory/7_2/expert/`

[18] Prikaz učesnika u izgradnji ekspertskog sistema: `http://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_expert_systems.htm`

[19] Šematski prikaz algoritama - ulančavanje unapred i ulančavanje unazad: `https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch01.html`