Predstavljanje znanja pomoću ontologija

Ranogajec, Andrija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Andrija Ranogajec

Zagreb, 2011.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Bojan Jerbić, dipl. ing.

Student:

Andrija Ranogajec

Zagreb, 2011.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem se prof. dr.sc. Bojanu Jerbiću za mentorstvo, asistentu Tomislavu Stipančiću za pomoći i savjete kod pisanja diplomskog rad i svima ostalima koji su mi na bilo koji način pomogli kod izrade ovog rada.

Najveća hvala mojoj braći, Zoranu i Mariju za potporu i strpljenje tijekom studija.

Andrija Ranogajec

Sadržaj:

Popis slika:

Popis kratica:

XML Extensible Markup Language
RDF Resource Description Framework
RDFS Resource Description Framework Shema
OWL Ontology Web Language
DAML Darpa Agent Markup Language
OIL Ontology Inference Layer
DL Description Logics
DC Dublin Core
GO Gene Ontology
NNF negacijska normalna forma
OOP objektno orijentirano programiranje
KIF Knowledge Interchange Format
URI Uniform Resource Identifier
URL Uniform Resource Locator

Sažetak:

Ontologije su formalni rječnici termina koji se koriste za razumijevanje i obradu podataka u računalu. Jezici ontologija su jezici koji se koriste za formalno zapisivanje ontologija. Prije pojave semantičkog web-a i snažnog rasta među povezivanjem računala bilo je dovoljno da ontologija bude razumljiva jednom računalu, odnosno onome računalnu na kojem se izvodi sustav koji koristi ontologiju. Iz navedenog razloga, prvi jezici ontologija nisu interoperabilni i standardizirani, već su prilagođeni radu s jednim sustavom. Kroz primjere je opisana deskriptivna logika uz naglašenu ulogu ontologija u tom kontekstu. Navedena je primjena, oznake i simboli te semantika i interpretacija deskriptivne logike. Opisane su dodirne točke ontologija sa deskriptivnom logikom i rezoniranjem. Kao najznačajniji algoritam u ovom području ukratko je opisan tableau algoritam za deskriptivnu logiku. U idućem dijelu ovog rada ukratko su opisana osnovna svojstva starijih jezika ontologija: jezika zasnovanih na predikatnoj logici prvog reda, jezika okvira i jezika opisnih logika. Detaljnije su opisana svojstva i sintaksa trenutno najzastupljenije platformski neovisne hijerarhije jezika za opisivanje ontologija, sastavljene od: XML, XML Schema, RDF, RDF Schema i OWL jezika. U radu se upoznaje s pojmom ontologije, konceptima vezanim uz izgradnju ontologije te formalizmom deskriptivne logike. Prestavlja se alat za rad s ontologijama - Protégé te prikazuju prvi koraci u toj radnoj okolini. Rad s klasama, relacijama, individuama, ontološkim grafom i rasuđivanjem ontologija. Na poslijetku obrađen je zadatak gdje je bilo potrebno izraditi bazu znanja o domeni koja je karakteristična za industrijsku primjenu kod poslova montaže. Navedena primjena podrazumijeva uporabu robota za obavljanje montažnih operacija te senzora za prikupljanje informacija iz okoline.

1. UVOD

U računalnim i informatičkim znanostima ontologije su obrazac podatak koji predstavlja koncepte unutar neke domene i odnose između tih koncepata, te se koristi za razumijevanje objekata unutar te domene. Ontologije su popularne u raznim istraživačkim područjima. Obrada prirodnog jezika, inženjerstvo znanja, informacijski sustav, te gospodarenje znanjem samo su neka od područja gdje se primjenjuju i istražuju ontologije. Ontologije definiraju objekte nekog područja (domene), njihova svojstva, te ograničavaju i postavljaju moguće odnose objekata i svojstava u tom području (domeni). Najjednostavnije rečeno ontologije su grane metafizike koje se bave identificiranjem stvari koje doista postoje u najopćenitijim terminima.

Ontologije zajedno sa skupom individualnih instanci klasa tvore bazu znanja. Iako je znanje izraženo ontologijama često prirodno, razumsko i poznato svakom čovjeku, računala nisu upoznata s njime i ne mogu donositi zaključke zasnovane na takvom zaključku. No u koliko je izgrađena ontologija koja precizno i detaljno opisuje neku domenu, onda je u postupke djelovanja u toj domeni moguće uključiti i računala. Da bi se ontologija izradila potrebno je pridržavati se osnovnih principa kao što su jasnoća, proširivost, minimalna ontološka vezanost (samo nužni pojmovi), te standardizacija imena. Razvoj ontologije mora biti iterativan proces i ne postoji samo jedan način izgradnje domene za kojega se može tvrditi da je točan.

Ontologije su korištene u umjetnoj inteligenciji, software inženjeringu, informacijskoj arhitekturi i semantički web kao oblik reprezentacije znanja o svijetu odnosno nekom njegovom dijelu. Semantički web je jedno od glavnog područja u kojem računala intenzivno koriste ontologije. On je nadogradnja već postojeće infrastrukture mehanizmima semantičkog povezivanja podataka i usluga. Ontologije u semantičkom web-u služe za definiranje rječnika metapodataka pomoću kojih se opisuju podaci i usluge dostupne web-om, te ontologije definiraju pravila zaključivanja koja su zasnovana na tim metapodacima.

Ako su ontologije zapisane u formatu koji je razumljiv računalu jedino onda su i korisne. Radi toga su razvijeni jezici koji određuju kako se ontologija treba zapisivati.

Povećanjem veza i različitih međudjelovanja računala, pojavila se potreba za razvojem ontoloških jezika čiji zapisi su razumljivi svim računalima, odnosno nisu ovisni o računalnoj platformi. Iz tih razloga moderni ontologijski jezici zasnovani su na jeziku koji nije platformski ovisan, a to je XML.

Na mnogo načina ontologije mogu doprinijeti većoj funkcionalnosti web-a, mogu pružiti jednostavniji način i poboljšati preciznost u pretraživanjima web-a, što znači da pretraživači mogu tražiti samo one stranice koje se odnose na određeni koncept.

U idućim poglavljima opisana su najistaknutija svojstva ontologije, jezici koji se koriste za izražavanje ontologije, deskriptivna logika i Protege program, te zaključak o ovom radu.

2. ONTOLOGIJE

Ontologija je konceptualni model neke domene našeg svijeta. Cilj ontologija je postizanje zajedničkog shvaćanja strukture informacija među računalima i ljudima. Ontologije sadrže ponovno iskoristive dijelove znanja o specifičnim domenama. No ti dijelovi informacija evaluiraju tijekom vremena, nisu statički, zato ontologije zahtijevaju modifikaciju, što znači promjene na domenama, prilagodba različitim poslovima ili promjene u konceptu, te je potrebna stalna podrška za podržavanjem tih promjena. To je izuzetno bitno u nekontroliranom području kao što je web, gdje se promjene događaju bez obavijesti. Rastom semantičkog web-a te nekontrolirane promjene, imat će još većeg utjecaja jer će računala koristiti podatke. Problem podrške za podržavanjem promjena je jako velik, jer postoje velike zavisnosti između podataka, aplikacija i ontologija. Pri kraju će promjene imati daleko veće posljedice. U praksi je vrlo teško uskladiti promjene na ontologiji sa modifikacijom aplikacije i podataka koji koristite.

2.1. Povijest ontologije

Ontologije kao pojam proučava se već dugi niz godina i potječe iz grčkih riječi ontos („biti“, „biće“, „bivstvujuće“) i logos („riječ/govor“). Nastao je u 17. stoljeću i potiče iz filozofske discipline, stare kao i sam Aristotel. Upravo je on prvi iznio definiciju da je ontologija metafizika, koja se bavi različitim tipova bivanja, odnosno metafizička studija prirode stvari i postojanja. U novije vrijeme ontologije se proučavaju unutar područja umjetne inteligencije, a posebno u sklopu istraživanja semantičkog web-a. Prema Tomu Gruberu, koji iznosi moderniju definiciju, primjenjiviju u području računarstva, značenje ontologije u kontekstu računalnih znanosti jest opis zajedničkih, formalnih specifikacija koncepata i odnosa, odnosno ontologija je općeniti rečeno *set definicija formalnog rječnika.* Ovdje koncepti označavaju apstraktne modele pojava u svijetu koji ističu bitna svojstva tih pojava. Tijekom druge polovice 20. stoljeća filozofi su naširoko raspravljali o mogućim metodama izgradnje ontologija, bez da su gradili, dok su računalni znanstvenici izgrađivali velike i masivne ontologije sa puno manje rasprava oko toga kako će biti

izgrađene. Početkom 21. stoljeća projekt kognitivne znanosti zbližava ta dva kruga stručnjaka.

2.2. Elementi ontologije

Većina ontologija općenito opisuju:

- Individue: osnovne objekte "početne razine"
- Klase: zbirke ili tipove objekata
- Atribute: pripadajuća svojstva, pojave, karakteristike ili parametre koje objekt može imati ili distribuirati
- Odnose: način na koji se objekti odnose jedni prema drugima

Individue su osnovana, početna razina sastavnica ontologije. Individue u ontologiji mogu uključiti konkretne objekte poput ljudi, životinja, molekula, planeta, kao i apstraktne individue poput brojeva i riječi. Ontologija ne mora uključiti individuu, no jedna od općih svrha ontologije jest omogućavanje sredstava za klasifikaciju individua pa i ako te individue nisu dio same ontologije.

Klase su apstraktne grupe, skupine ili zbirke objekata. Mogu sadržavati individue, druge klase ili kombinaciju jednih i drugih. U prirodi klase ontologije mogu biti ekstenzijalne ili intenzionalne. Klasa ontologije je ekstenzionalna ako i samo ako je okarakterizirana samo po svom članstvu. Točnije klasa C je ekstenzionalna samo ako za klasu C ako „C" ima točno iste članove kao C. Tada su C i „C" identični. Ako klasa ne zadovoljava takve uvijete onda je intenzionalna.

Atributi se mogu pridruživati objektima u ontologijama. Svaki atribut ima barem ime i vrijednost, i koristi se za pohranu informacija koje su specifične za objekte na koje prianjaju. Vrijednost atributa naravno može biti kompleksni oblik podataka. Potrebno je definirati atribute koncepata, ontologija definira domenu.

Odnosi – korištenje atributa važno je da bi se opisali odnosi (relacije) između objekata u ontologijama. Relacija je atribut čija vrijednost jest drugi atribut u

ontologijama. Na primjer, imamo stari i novi model automobila, stariji model bi mogao biti opisan kao sljedeći atribut – nasljednik (ime nekog prijašnjeg automobila). Snaga ontologija proizlazi iz toga što možemo točno opisati te odnose. Skup odnosa (relacija) opisuje semantiku domene.

2.3. Tipovi ontologija

Ovisno o ulozi koji zauzima u radu i razine općenitosti kojom su definirane, ontologije se dijele na nekoliko tipova: domenske ontologije, meta ontologije, općenite ontologije reprezentacijske ontologije te metodološke ontologije.

Domenske ontologije karakterizira prikupljanje znanja iz određene uske domene. Oblikuje specifičnu domenu ili dio svijeta. Primjerice, ontologija mikroprocesora sadrži koncepte koji su bitni za gradnju računala. Ti koncepti nisu primjenjivi u proučavanju sisavaca, te je zbog toga navedena kao domenske ontologije. Domenske ontologije mogu se ponovno upotrebljavati samo u domeni koju obuhvaćaju (medicina, inženjerstvo i sl.)

Meta ontologije pružaju rječnik za opisivanje on-line sadržaja. Dublin Core (DC) je primjer meta ontologije koja služi za nadogradnju metoda pretraživanja Web stranica pružajući rječnik za opisivanje svojstva stranice. DC definira metapodatke koji se vežu za stranicu , a služe za što bolje pretraživanje stranica. Tipični metapodaci definirani DC-om su „Description“, koji sadrži kratak opis stranice i „Creator“ koji sadrži ime tvorca stranice. Tvorci web stranice nadopunjuju stranice metapodacima, metapodatke uzimaju web pretraživači, tumače ih na osnovi prepoznavanja ontologije, te ih koriste kod naprednog pretraživanja Web-a.

Općenite ontologije služe za pohranjivanje znanja o svijetu, te definiranje općenitih koncepata i oznaka za objekte poput prostora, stanja, vremena i događaja. Opće ontologije su već dobro razrađene i postoji nekoliko općih ontologija koje se često koriste u istraživanjima, ali takve ontologije su obično neupotrebljive za neke konkretne zadatke koje zahtjeva određena domena ili neka aplikacija.

Reprezentacijske ontologije ne vežu se ni uz jednu specifičnu domenu. Ove ontologije služe za specifikaciju jezika pomoću kojih se predstavlja znanje. Primjer ovakve ontologije jest Frame Ontology. Ova ontologija definira koncepte okvir, utičnica i ograničenje utičnice, koji služe za definiranje znanja u formalizmu sličnom objektno orijentiranim jezicima.

Metodološke ontologije predstavljaju rječnike i znanje o standardnim metodama rješavanja određenog problema. Problemi i metode nisu iz određene domene djelatnosti, već predstavljaju općenite probleme i metode rješavanja problema.

U praksi najčešće ontologije imaju svojstva različitih tipova, pa je teško za neku specifičnu ontologiju točno odrediti kojemu tipu pripada. Standardni je postupak da se nove ontologije grade na osnovu starih. Tako se primjerice specifičnije ontologije grade na osnovu općenitijih. Podjela ontologija na tipove je samo orijentacija koja pomaže bržem i boljem razumijevanju i prihvaćanju pojedine ontologije.

2.4. Osnovne mogućnosti ontologije

Specifikacija ontologije sa općenitog stajališta sastoji se od pet osnovnih konstrukta, a to su: razredi, instanci, relacija, funkcija i aksioma.

Razredi predstavljaju osnove koncepta koji se nalaze u nekoj domeni. Na primjer, u ontologiji sveučilišta postoji razredi koje čine: osoba, profesor, student, fakultet, semestar, ispiti itd.

Instance predstavljaju same objekte, odnosno pojedine iz nekog određenog razreda. Na primjer studentica s imenom Martina jedna je instanca razreda student.

Relacije označavaju određeni put, odnosno veze između razreda i instanci u domeni. Primjeri relacija su: je – podvrsta, je, polaže ispit. U ontologiji sveučilišta možemo utvrditi sljedeće relacije: je – podvrsta (student, osoba), čime se određuje da je svaki objekt razreda student istovremeno i objekt razreda osoba. Slično, relacija polaže – ispit (student, profesor, ispit), određuje da su objekti razreda student povezani s objektima razreda ispiti i profesor relacijom polaže – ispit. Relacija može sadržavati i instancu. Primjer, relacija polaže – ispit (Martina, Stjepan, Informatika) označava da student Martina polaže ispit iz Informatike kod profesora Stjepana.

Funkcije su posebna vrsta relacije za koje vrijedi da prvih n-1 argumenata relacije jedinstveno određuje n-ti element relacije. Tako funkcija položio – ispit (Martina, Stjepan, Informatika) vraća jedinstvenu vrijednost istina ako je Martina položila ispit Informatike kod profesora Stjepana. Navedena funkcija ekvivalentna je postojanju točno jedne relacije položio – ispit (Martina, Stjepan, Informatika, istina). Kada bi postojala još neka relacija položio – ispit (Martina, Stjepan, Informatika, laž) vrijednost zadnjeg argumenta relacije ne bi bila jedinstvena. Funkcije se mogu izražavati pomoću relacija, ali jezici ontologija često podržavaju eksplicitnu definiciju funkcija.

Aksiomi su tvrdnje koje su unutar određene ontologije uvijek istinite. Pomoću aksioma izražavaju se temeljne postavke neke domene. Primjeri aksioma su rečenice: „Ako je osoba student tada osoba nije profesor“.

Izražajnost ontologija je ekvivalentna, pa uz ovih pet navedenih konstrukta, treba spomenuti da se ontologije mogu izražavati i pomoću konstrukta predikatne logike prvog reda: konstantnih simbola, funkcijskih simbola, predikatnih simbola i varijabli. Razredi i instance, u predikatnoj logici prvog reda, mogu se izraziti kao konstantni simboli na primjer, razredi i instance Martina, Stjepan, Student, Profesor, Ispit i Informatika postaju konstantni simboli. Veze između instanci i razreda mogu se izraziti predikatnim simbolom je-tipa. Primjer (Martina, Student) označava da je Martina objekt iz razreda Student. Relacije i funkcije su izravno podržane pomoću predikatnih, odnosno funkcijskih simbola. Aksiomi se tvore koristeći varijable s univerzalnim i/ili egzistencijalnim kvantifikatorima, konstantne, funkcijske i predikatne simbole, te logistička pravila. Aksiomi a) Osoba koja je Student nije istovremeno Profesor.

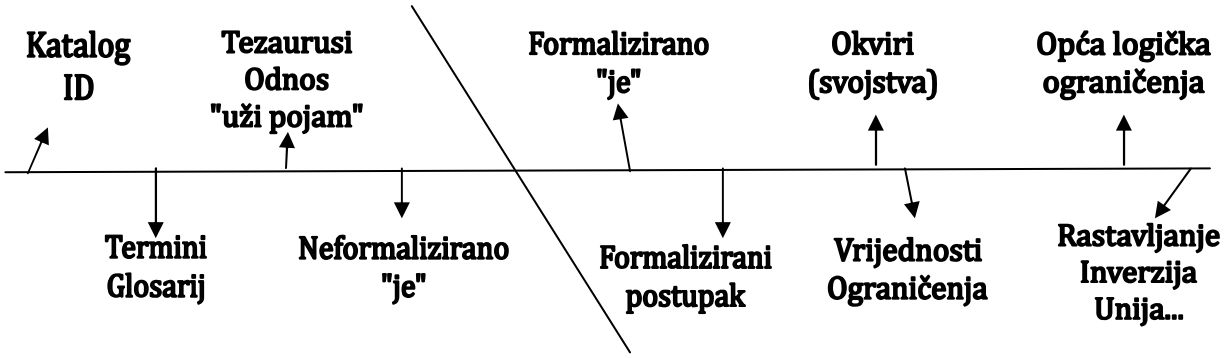
$$(Ax)\ (je\text{-}tipa(x, Student) \rightarrow \neg je\text{-}tipa\ (x,Profesor))$$

Ontologije je moguće i izraditi i na druge načine, upravo zbog općenitosti predikatne logike prvog reda. Primjerice, može se odlučiti da se pripadanje pojedine instance nekom razredu ne određuje pomoću je-tipa, već da se razredi predstavljaju kao predikatni simbol. U tom primjeru bi se pripadnost Martina razredu Student izrazila na ovakav način - Student (Martina). Zbog ovakvih nejednoznačnosti kod izražavanja, razvijaju se drugi jezici koji su namijenjeni prvenstveno za izražavanje ontologija, i kao takvi pojednostavljuju izražavanje osnovnih koncepata.

2.5. Logička izražajnost ontoloških jezika

Aplikacije koje koriste ontologije pod snažnim su utjecajem logičke izražajnosti ontoloških jezika.

Što je ontologija?



Slika 1. McGuinnessina taksonomija ontologija

Prva i najjednostavnija kategorija, *katalog ID*, sastoji se od konačnog popisa naziva koji se koristi kod nadziranja rječnika. Sljedeća kategorija su *glosariji*, koji se sastoje od popisa naziva sa značenjem svakog naziva iskazanog u prirodnom jeziku. *Tezaurusi* dodaju glosarijima određene osnovne semantičke odnose, kao što je sinonimija (tezaurus za glazbenu domenu može označiti da „CD“ i „compact disc“ imaju isto značenje, odnosno znače isti stvar). Često tezauruse također uključuju *neformalni „je“*, odnosno da bi ga razabrao ljudsko korisnik, ali ne i da ga prosuđuje na bilo koji strojno obradivi način.

Desno od dijagonalne crte ontologijama se počinju dodavati svojstva i odnosi koji se mogu prosuđivati formalno. Time dolazimo do složenosti okvirnih sustava. Ako je odnos podklasnosti prikazan kao prijelazan, ontologija omogućuje nasljeđivanje. Sljedeća logička finesa koja se obično dodaje jesu „*vrijednosti ograničenja*“ pomoću kojih se tvrdnje ograničavaju na domenu ili raspon ontoloških odnosa. Na primjer, svaki imenovani umjetnik na CD-u je osoba. Nakon toga je uobičajeno dodati standardni među-skup odnosa klasične teorije skupova. To uključuje *uniju*. Na primjer, „Kategorija glazba 80-tih sastoji se od izdanja iz 1982, 1983….itd.“, *presjek* na primjer „Kategorija

glazba koju je preporučilo studentsko odjeljenje 10 sastoji se od onih pjesama koje su zabilježene na iPod-ovima svakog standardnog odjeljenja 10“ i razdvajanje na pr., „Jazz CD-i nisu heavy metal CD-i“.

3. DESKRIPTIVNA LOGIKA

Deskriptivna logika razvila se kao ekstenzija semantičkih mreža, okvira i logike. Možemo reći da je to inteligentna aplikacija koja zaključivanjem stvara novo znanje iz već postojećeg. Namijenjena je terminološkom semantičkom opisu neke domene, rasuđivanju i zaključivanju. Danas čini okosnicu ideje *semantičkog weba* i koristi se u izračunu računalnih ontologija. Deskriptivna logika je skupina jezika, a ne samo jedan jezik. Većinu opisne logike čine odlučljivi formalizmi koji su podskup logike prvog reda. Deskriptivna logika pruža samo jedan način oblikovanja ontologije, dok je ontologija općenitiji pojam.

Deskriptivna logika ili opisna logika (engl. *Description logics*) pojavljuje se 1985. g. kao nastavak Minskyeve ideje – KL – ONE jezik. Otada se razvija sve do današnjih dana, sa intezivnim razvojem od sredine 90-tih.

Deskriptivna logika se fokusira na predstavljanje onoga što nazivamo terminološko znanje ili konceptualno znanje, ona opisuje dio našeg svijeta ne podrazumijevajući da zna sve o njemu. Ona operira nad konačnom domenom, ima samo unarne i binarne predikate i nema eksplicitne varijable.

Cilj DL je formalizirati osnovnu terminologiju modelirane domene, pohraniti je u ontologiju i omogućiti zaključivanje na tom znanju.

3.1. Temeljni pojmovi

Prema paradigmi u kojem je znanje o domeni primjene izdvojeno u zasebni podsustav, osnovu "inteligentnog" ponašanja čine entiteti i relacije tog podsustava. Znanje u tom podsustavu mora biti tako organizirano da je na temelju njega moguće rasuđivanje (eng. reasoning) - izvođenje informacija koje nisu eksplicitno prisutne. Tek uz mogućnost rasuđivanja skup podataka postaje znanje, a programska podrška postaje "inteligentna". Iako danas postoji velik broj različitih pristupa predstavljanju znanja, moguće je nabrojati 5 glavnih paradigmi:

(i) Matematička logika,

(ii) Okviri,

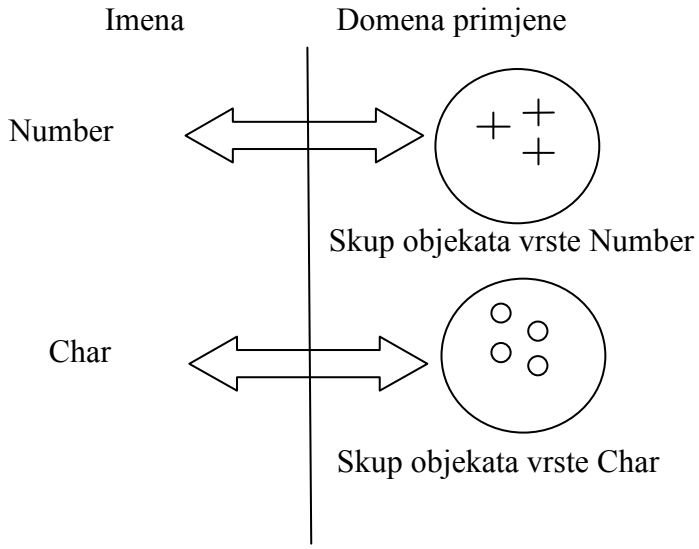
(iii) Pravila,

(iv) Konceptualne, Semantičke mreže i

(v) Umjetne neuronske mreže.

Svaki od ovih pogleda dolazi iz različitih izvora. Tako npr. okviri i semantičke mreže imaju temelje u psihologiji, umjetne neuronske mreže u neurologiji, a matematička logika u logici. Različite paradigme na različite načine pristupaju znanju, njegovom sadržaju i postupcima rasuđivanja. Deskriptivne logike ili opisne logike (eng. description logics) su formalni jezici za predstavljanje znanja utemeljeni na matematičkoj logici. Naziv "opisne logike" dolazi iz činjenica da se pojmovi iz neke domene nazivaju "opis koncepta" i semantika izraza koji se pojavljuju je utemeljena na logici. Opisne logike su poslužile kao teoretska osnova za razne sustave i implementacije predstavljanja znanja. Najpoznatija i najraširenija implementacija je Web Ontology Language (OWL). Sustavi utemeljeni na deskriptivnoj logici modeliraju različite domenu primjene. Tako npr. domena primjene može biti medicinska dijagnostika, taksonomija životinja ili programsko inženjerstvo. U deskriptivnim logikama, kao i u ostalim inačicama matematičke logike, za prikaz objekata iz domene primjene koristimo skup imena. Imena u DL predstavljaju pojedine objekte, skupove objekata i relacije između objekata domene primjene. Preslikavanje između imena i objekata naziva se interpretacija jer nam govori kako treba interpretirati imena koje koristimo. Interpretacija uspostavlja korespondenciju između imena i stvarnih objekata u nekoj domeni. Domena primjene naziva se još i interpretacijska domena radi toga što je povezana s interpretacijom koja se koristi.

Primjer koji slijedi na slici 2. prikazuje jednu domenu primjene i pridružena imena. Mali križić na desnoj strani slike predstavljaju pojedine objekte – individue iz domene primjene. Više pojedinih objekata sa sličnim svojstvima grupirani su u skupove. Tako na slici 2. imamo skup križića koji predstavljaju brojeve, a tom skupu je pridruženo ime Number. Ime Number se, dakle, koristi kao oznaka, kao opis koncepta za skup pojedinaca koji svi imaju zajedničkosvojstvo da su "broj". Na sl. 2. također imamo ime Char koje koristimo kao ime koncepta za skup pojedinaca koji su vrste "slovo".



Slika 2. Interpretacija imena

Već u ovom jednostavnom primjeru može se uočiti da je vrlo prirodno da se stvarni objekti iz domene primjene koji imaju slična svojstva grupiraju u skupove. Ime koje koristimo u logičkom prikazu, a koje pridružujemo skupu pojedinaca iz domene primjene nazivamo koncept.

3.2. Primjena deskriptivne logike

Koraci u primjeni su:

1. Formalizirati osnovnu terminologiju domene koja se želi modelirati, što uključuje dobavljanje znanja
2. Pohraniti znanje u obliku ontologije/terminologije

Deskriptivna logika primjenjuje se u :

- medicinskoj informatici – kao terminologije
  - SNOMED – sistematizirana nomenklatura medicine – 450000 pojmova
  - *open*GALEN project – medicinska terminologija
- U bioinformatici – kao terminologije
  - GeneOntology (GO) – 17000 pojmova

- Semantički web - OWL
    - Cilj: Dati semantički opis sadržaja web stranica
    - Ostvarenje: Pokazati na koncepte definirane u nekoj drugoj ontologiji –> još uvijek samo vizija

Glavni problemi primjene deskriptivne logike je sporost zaključivanja, kao i tromost i nezainteresiranost tržišta.

3.3. Oznake deskriptivne logike

Oznake deskriptivne logike su:

AL – atributni jezik, osnovni jezik DL-a
C - negacija složenih koncepata (negacija dozvoljena na lijevoj strani izraza)
I - inverzna svojstva
N - ograničenja kardinaliteta
H - hijerarhija svojstava
(D) - podatkovna svojstva
S - kratica za ALC s tranzitivnim svojstvima
O - pobrojane klase u svojstvima
R – ograničeni složeni aksiomi za uključivanje uloga, refleksivnost i nerefleksivnost, razdvojenost uloga

3.4. Simboli deskriptivne logike

Simboli deskriptivne logike su vrlo slični simbolima koji se koriste u ostalim logikama. Ovdje navodimo koji se sve simboli koriste i koje im je značenje: Skup svih individua ($\top$), prazan skup ($\bot$), negacija i komplement ($\neg$), konjunkcija i presjek ($\sqcap$), disjunkcija i unija ($\sqcup$), univerzalni kvantifikator ($\forall$), egzistencijalni kvantifikator ($\exists$), inkluzija koncepata ($\sqsubseteq$), ekvivalencija koncepata ($\equiv$), definiranje koncepta ($\doteq$) te pripadnost konceptu, odnosno ulozi ($:$).

3.5. Semantika i interpretacija deskriptivne logike

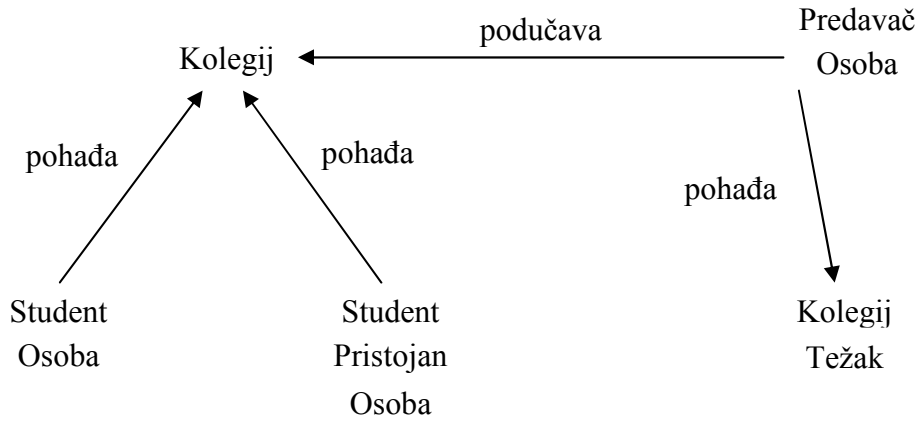
Semantika deskriptivne logike je bazirana na *interpretaciji.* Interpretacija u deskriptivnoj logici je potpuna slika svijeta, odnosno opis svijeta. Interpretacija DL ALC je logika prvog reda koja ima samo unarne (koncepte) i binarne (uloge) predikate. Formalno, semantika je osnovana na uređenom paru : $(\Delta^I , \cdot^I)$,.

- Δ^I : domena interpretacije *I*
- $\cdot^I$: interpretacijska funkcija koja preslikava svako ime koncepta *A* u podskup A^I od Δ^I i svako ime uloge *R* u binarnu relaciju R^I nad Δ^I

Semantika složenih koncepata – primjer :

$(\neg C)^I = \Delta^I \setminus C^I$
$(C \sqcap D)^I = C^I \sqcap D^I$
$(C \sqcup D)^I = C^I \sqcup D^I$
$(\exists R.C)^I = \{ d \mid \text{postoji } e \in \Delta^I \text{ s } (d, e) \in R^I \text{ i } e \in C^I \}$
$(\forall R.C)^I = \{ d \mid \text{za sve } e \in \Delta^I, (d, e) \in R^I \text{ i } e \in C^I \}$

Semantika DC ALC - primjer interpretacije



Slika 3. Interpretacija semantike DC ACL

U ovoj interpretaciji vrijede izrazi:

$Osoba \sqcap \exists pohađa . Kolegij$

$Osoba \sqcap \forall pohađa . (\neg Kolegij \sqcup Težak)$

Interpretacija svim definiranim konceptima i relacijama dodjeljuje skupove.

Primjer:

Koncept (A)	**Skup u kojeg se koncep preslikava u interpretaciji I (A^I)**
Student	{Petar, Katarina}
Osoba	{Petar, Katarina, Ivan}
Predavač	{Ivan}
Kolegij	{Geografija, Robotika}
Težak	{Robotika}

Relacija (A)	**Skup u kojeg se relacija preslikava u interpretaciji I (R^I)**
pohađa	{(Petar, Geografija), (Katarina, Geografija), (Ivan, Robotika)}
podučava	{(Ivan, Geografija)}

Slika 4. Primjer interpretacije koncepta

3.6. Tableau

Tableau je najpoznatiji i najčešće korišten algoritam za rasuđivanje u opisnoj logici. Njegova svojstva su: **odlučljivost** - izvođenje algoritma završava za bilo koji upit, **ispravnost** - ako je nešto algoritmom dokazano to je sigurno i istina, **potpunost** - ako je nešto istina to se može algoritmom i dokazati. Njegov zadatak je naći ispravne i potpune postupke odlučivanja za zadovoljivost (i podrazumijevanje) u DL koji su prilagođeni uspješnoj implementaciji.

Ideja tableau algoritma je pokušati izgraditi model od C_0, razbijajući ga sintaksno i tako zaključujući nova ograničenja na takvom modelu. Potrebno je transformirati svaki koncept C_0 u ekvivalentni C_1 u negacijskoj normalnoj formi (NNF).

Pritom vrijede transformi:

$\neg(C \sqcap D) = \neg C \sqcup \neg D$
$\neg(C \sqcup D) = \neg C \sqcap \neg D$
$\neg\neg C = C$
$\neg\forall R.C = \exists R.\neg C$
$\neg\exists R.C = \forall R.\neg C$

Tableau algoritam – primjer:

- *Reasoner* gradi stablo dovršenja (engl. completion tree).
- Čvorovi stabla dovršenja su koncepti, a grane su uloge.
- Stablo se širi korištenjem pravila dovršenja. U principu, dodaje se novi čvor u stablo, ukoliko imamo kvantifikatore uloga.
- Nadalje, pravilo unije je nedeterminističko, pa je potrebno ispitati svaku granu. Ako dođe u jednoj od grana do sudara (engl. clash), ispituje se dalje druga grana.
- Sudar je definiran kao čvor u kojem vrijedi A i $\neg A$, za neki koncept A (ili ako vrijedi)

Kaže se da je stablo dovršenja potpuno (engl. complete) ako nije dalje moguće primijeniti niti jedno pravilo dovršenja.

3.7. Ponašanje Tableau algoritma

Ako algoritam počinje s C_0, koji je NNF-u, tableau algoritam ponavlja primjenu pravila dovršenja u kojem god redoslijedu želi. C_0 je zadovoljiv ako i samo ako se pravila dovršenja mogu primijeniti na takav način da rezultiraju u potpunom stablu bez sudara.

3.8. Obitelj deskriptivnih logika

Tijekom zadnjih 15-tak godina uočeno je da je polazišnu logiku ALC moguće proširiti dodatnim svojstvima a da se kompleksnost rasuđivanja bitno ne poremeti. Zahtjev za proširenjem polazišne logike dolazi i sa strane sustava utemeljenih na DL. Svako proširenje DL otvara nove mogućnosti, daje veću snagu u definiranju ontologija i proširuje moguće primjene DL. Prvo proširenje ALC logike je dodavanje tranzitivnih uloga. Intuitivno, tranzitivne uloge su one koje se prenose: Ako su parovi <a, b> i <c, d> u relaciji R, onda će i par <a, c> biti u relaciji R. Formalno: Podskup tranzitivnih uloga označimo sa R+. Pojedinačnu ulogu označimo sa R, a pojedince označimo kao a, b, c. Tranzitivne uloge se definiraju kao:

ako < a, b >2 R^I i < b, c >2 R^I i R 2 R+onda < a, c >2 R^I

Ne zaboravimo da su uloge, u stvari, binarne relacije, pa smo tranzitivnim ulogama definirali tranzitivne binarne relacije. Pogledajmo jedan primjer tranzitivnih uloga u domeni objektno orijentiranog programiranja (OOP). Jedna od glavnih značajki OOP-a je pojam nasljeđivanja. Neka klasa A može naslijediti klasu B, a klasu B može naslijediti neka treća klasa C. Očito je da pojam nasljeđivanja uspostavlja relaciju (ulogu) između dvije klase. Tu ulogu bismo mogli nazvati "inherits" i napisati < A, B >: inherits sa značenjem da A nasljeđuje B. Ako još imamo < B, C >: inherits, onda će vrijediti i < A, C >: inherits jer je uloga inherits tranzitivna.

4. JEZICI ONTOLOGIJA

Ontologije su formalni rječnici koji se koriste za razumijevanje i obradu podataka u računalu i ti jezici se koriste za formalno zapisivanje ontologija. Prije pojave semantičkog weba i rasta međupovezivanja računala bilo je dovoljno da ontologije budu razumljive onom računalu na kojem se izvodi sustav koji koristi ontologiju. Zbog spomenutog razloga prvi jezici nisu standardizirani, već su prilagođeni radu s jednim sustavom. Pojavom semantičkog web-a javlja se i potreba za razvojem ontološkog jezika čiji je zapis neovisan o razumijevanju i platformi svih računala. Takvi, takozvani moderni jezici zasnivaju se na platformski neovisnom jeziku XML. Uočeno je da se za opisivanje nekih domena ne trebaju koristiti složeni jezici, nego su dovoljni jezici koji podržavaju samo osnovna svojstva ontologije, zato se grade hijerarhije jezika ontologija. Najjednostavniji jezici čine osnovu hijerarhije na kojoj se grade sve složeniji jezici, jer što je jezik složeniji njime se mogu opisivati i složenije ontologije.

Jezici su zasnovani na predikatnoj logici prvog reda, čije proširenje je i način gradnje ontoloških jezika. Primjer ontoloških jezika prvog reda bliskih predikatnoj logici su CycL i KIF.

CycL je formalni jezik s sintaksom naslijeđenom od predikatne logike prvog reda. On proširuje predikatnu logiku prvog reda konceptima drugog reda: tipiziranjem, refikacijom i mikroteorijama. Tipiziranje zahtjeva da svaka varijabla ima produženi tip. Zhtjeva se i da argumenti funkcija i predikata (relacija) imaju definirani tip. Primjerice konstante Martina i Stjepan su tipa Čovjek. Predikat je _ viši (x, y) zahtjeva da varijabla x i y poprime vrijednost tipa Čovjek.

KIF (eng. Knowledge Interchange Format) razvijen je s ciljem omogućavanja razmjene znanja među udaljenim računalnim sustavima. KIF se može koristiti za izražavanje i razmjenu ontologija, iako mu to nije namjena. Sastoji se od konstanti, definicija i rečenica. Specifično za KIF je korištenje četiri različita tipa konstanti: objektni, funkcijski, relacijski i logički. Objektne konstante predstavljaju pojedine objekte iz svijeta koji se opisuje, funkcijske i relacijske predstavljaju funkcije i relacije,a logičke konstante predstavljaju stanje svijeta i mogu biti istina ili laž.

Najzastupljenije platformski neovisne hijerarhije jezika za opisivanje ontologije su sastavljen od XML, XML SHEMA, RDF, RDF Shema i OWL jezika.

4.1. XML i XML Shema

XML (eng. Extensible Markup Language)

XML je jezik koji je razvijen za označavanje i opisivanje podataka. XML nije razvijen za semantički web, ali su ga njegovi konstruktori prihvatili. U začetku je zamišljen kao jednostavan način za slanje podataka preko Web-a. XML dopušta autorima na mreži da definiraju svoje vlastite oznake i s tim vlastiti format dokumenta, podvrgnut sintaksi specificiranoj u XML-u. Najjednostavnije rečeno XML je skup podataka iz neke domene zapisanih na standardan i strukturiran način. Struktura XML je strogo hijerarhijska i sastoji se od elemenata koji su hijerarhijski ugniježđeni. Svaki element ugnježđuje sve elemente koji su mu hijerarhijski podređeni i/ili tekst koji predstavlja podatke vezan uz taj element. Podaci opisani XML-om mogu se razmjenjivati između računala različitih arhitektura. Svako računalo mora imati XML procesor/prevoditelja koji podatke iz XML dokumenta prevede u oblik razumljiv lokalnom računalu.

```
<poruka>
  <za>Mario Maric</za>
  <od>Perice Peric</od>
  <naslov>Podsjetnik</naslov>
  <sadrzaj>Ovo je tekst  poruke</sadrzaj>
</poruka>
```

Slika 5. Primjer XML dokumenta

Slika 5 prikazuje jednostavni primjer XML dokumenta koji sadrži podatke o jednoj poruci elektroničke pošte. Unutar teksta postoje elementi „poruke“, „za, „od“, „naslov“ i „sadržaj“. Svaki element počinje s oznakom (eng. tag) oblika <ime_elemenata>, a završava s oznakom oblika </ime_elemenata>. Sve što se nalazi između oznake početka i kraja elemenata smatra se djelom elemenata.

XML dokumenata i njihovo značenje nisu unaprijed određeni već ih pisac XML dokumenata stvara prema potrebama domene u kojoj se XML koristi. Svaki XML dokument je građen kao stablo i ima istaknuti korijenski element koji sadrži elemente, znakove podataka i atribute.

XML jezikom moguće je definirati dokumente kojima se razmjenjuju podaci neke domene među različitim računalima. Međutim sam XML nije dovoljan za definiranje ontologije domene. Definirati ontologiju, gledano iz perspektive XML-.a znači propisati strukturu XML dokumenata koji sadrže podatke te domene, odnosno propisati hijerarhiju elemenata koju elementi te domene moraju poštovati, definirati imena elemenata i njihove moguće vrijednosti, definirati imena atributa i njihove moguće vrijednosti, te propisati proceduru pomoću koje se XML dokument može označiti kao pripadnik određene domene.

Osnovni problem izražavanja ontologije pomoću XML-a je nemogućnost jedinstvenog imenovanja elemenata i atributa. Mehanizam koji omogućava definiranje jedinstvenih imena elemenata i atributa u XML dokumentima nazivaju se XNL Namespace. Taj mehanizam uvodi imenike elemenata i atribute. Unutar jednog elementa ne dolazi do podudaranja imena atributa i elementa. Unutar jednog XML dokumenta mogu se koristiti elementi i atributi iz različitih imenika, što se označava posebnom sintaksom.

```
<?xml version="1.0"?>
<knjige>
  <knjiga xmlns="urn:BooksAreUs.org:BookInfo"
          xmlns:val="http://Currency.org">
    <naslov>Digitalna tipografija</naslov>
    <cijena val:valuta="HRK">249.95</cijena>
  </knjiga>
</knjige>
```

Slika 6. Prikazuje primjer korištenja imenika unutar XML dokumenta.

Slika 6 prikazuje primjer korištenja imenika unutar XML dokumenata. Unutar elemenata „knjiga“ koriste se dva imenika, Imenici koji se koriste unutar nekog elementa definiraju se ključnom riječju xmlns kao atributi tog elementa. Prvi imenik koji se koristi unutar elementa 'knjiga' jedinstveno je odrađen URI-em "urn:BooksAreUs.org:BookInfo". Taj imenik je podrazumijevani, što znači da sva imena elemenata i atributa unutar elementa 'knjiga' pripadaju tom imeniku, osim ako

eksplicitno nije naznačeno suprotno. Tako podrazumijevanom imeniku pripadaju elementi 'knjiga', 'naslov' i 'cijena'.

Drugi korišteni imenik je jedinstveno određen URI-em "http://Currency.org". Ovaj imenik nije podrazumijevani, za svaki element koji pripada ovom imeniku, a nalazi se unutar elementa 'knjiga' mora eksplicitno biti naznačeno pripadanje imeniku. Za takvo označavanje uvode se prikrate, tako je uvedena prikrata 'val' koja unutar ovog XML dokumenta označava imenik određen URI-em "http://Currency.org". Unutar elemenata 'cijena' koristi se atribut 'valuta'. Imenici u XML dokumente uvode mogućnosti jedinstvenog imenovanja pojedinih elemenata i atributa. No pomoću XML imenika nije moguće definirati elemente 'knjiga' koji pripadaju imeniku 'Književna djela' sadrži druge elemente imena 'naslov' i 'cijene', da su ti elementi tipa znakovni niz i broj, te da znakovni niz koji predstavlja naslov knjige ne smije biti duži od 1000 znakova.

XML Shema (eng. Extensible Markup Language Shema)

U pokušaju da se nadoknadi nedostatak semantičkog pomagala XML-a učinjen je izgradnjom niza „shema“ koje mu se mogu dodavati. Najšire korištena je „XML shema“, koju je uglavnom razvio Microsoft. Svaka primjena XML sheme na XML dokumentu proizvodi daljnju datoteku koja ispisuje svoj rječnik, odnosno elemente i atribute, svoj model sadržaja i svoje tipove podataka. XML shema je jezik za definiranje strukture XML dokumenata. Pomoću XML sheme jezika moguće je definirati sheme XML dokumenata. Ovakva svojstva omogućavaju gradnju jednostavnih ontologija. XML Shema ima dva važna svojstva.

Prvo važno svojstvo je XML Shema jezika je da ima mogućnost definiranja strukture XML dokumenata, što obuhvaća: definiranje elemenata i atributa koji se pojavljuju u dokumentu, definiranje redoslijeda elementa, definiranje hijerarhije elemenata te definiranje broja pojavljivanja pojedinog elementa.

Drugo važno svojstvo je mogućnost definiranja tipova podataka koje pojedini element ili atribut sadrži. U sklopu ovog svojstva moguće je: definirati da li je pojedini element ili atribut prazan ili sadrži tekst, odrediti tip podatka koji tekst sadrži

(decimalni, datum, cjelobrojni itd.), definirati podrazumijevanu vrijednost, konstantu vrijednosti elemenata i atributa te odrediti skup vrijednosti koje neki element ili atribut može imati.

XML Shema ima mogućnost nasljeđivanja tipova podataka, što omogućuje gradnju novih tipova podataka nadogradnjom starih tipova podataka.

Slika 7 (u nastavku) prikazuje jednostavnu XML shemu napisanu u XML Shema jeziku (a) i XML dokument koji sadrži podatke definirane tom shemom (b). Dokument na (b) dijelu slike napisan je u skladu sa shemom na (a) dijelu.

```
<?xml version="1.0"?>
<xs:schema   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://primjer.hr"
                xmlns="http://primjer.hr">
    <xs:element name="poruka">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="za" type="xs:string"/>
          <xs:element name="od" type="xs:string"/>
          <xs:element name="naslov" type="xs:string"/>
           <xs:element name="sadrzaj" type="xs:string"/>
         </xs:sequence>
       </xs:complexType>
   </xs:element>
</xs:schema>
```

(a)

```
<?xml version="1.0"?>
<poruka        xmlns="http://primjer.hr"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://primjer.hr poruka.xsd">
           <za>Ivan Ivic</za>
           <od>Perica Peric</od>
          <naslov>Podsjetnik</naslov>
          <sadrzaj>Ovo je tekst poruke</sadrzaj>
</poruka>
```

(b)

Slika 7. Jednostavne XML Shema (a) i XML document u skladu s shemom (b)

Prvi element u dokumentu jest 'poruka' kao što je i određeno shemom. Također je definirano da je podrazumijevani imenik ovog dokumenta onaj definiran shemom.

Ako je potrebno koristiti formalnu proceduru potvrde usklađenosti dokumenta sa shemom, moguće je korištenjem elementa 'xsi:schemaLocation' (definiran u zasebnom imeniku) postaviti lokaciju dokumenta koji sadrži shemu. U tom slučaju, XML procesor/prevoditelj koji obrađuje XML dokument može formalno potvrditi valjanost dokumenta uspoređujući ga sa shemom. Pošto su svi podelementi elementa 'poruka' u skladu s onima definiranim u shemi, te sadrže podatke čiji su tipovi u skladu s onima definiranim u shemi, XML prevoditelj potvrđuje usklađenost dokumenta sa shemom. XML Schema jezik je složen, te ovdje nisu prikazane sve njegove mogućnosti.

Pomoću XML Sheme moguće je izraziti samo hijerarhijske veze među objektima u svijetu. Ipak zbog svoje jednostavnosti XML Shema jezik je često korišten u današnjem svjetu elektronike i njenog poslovanja.

4.2. RDF i RDF Shema

RDF (eng. Resource Description Framework)

RDF je jezik za predstavljanje informacija I osnovni jezik za izražavanje na Semantičkom Webu, koji omogućava izricanje tvrdnji o definiram resursima. Budući da je RDF podatkovni model a ne jezik, potrebno ga je iskazati u jeziku. Namijenjen je za predstavljanje metapodataka na Web resursima. RDF je dobio sintaksu u XML-u. RDF domena je neovisna, te ovisno o korisniku definira se osnovna terminologija u shemu zvanom RDF Shema. U RDF-u se može definirati vokabular, odrediti koje svojstvo se primjenjuje na određenu vrstu objekta i koju vrijednost se može izabrati te opisuju odnose između objekata. Pošto već imamo sredstvo referenciranja na bilo koji postojeći resurs (preko URI-a), RDF nam pruža mogućnost da se izraze relacije i struktura među tim resursima. RDF je namjenjen situacijama kada je informacije potrebno obraditi aplikacijom – računalnim agentom, a ne izravno predstaviti ljudima. RDF pruža format za izražavanje informaija koji je neovisan o platformi , te se kao takav može izmjenjivati između različitih aplikacija.

Osnovni pojmovi RDF-a su izvori, svojstva i izjave.

Izvori - o izvorima se može razmišljati kao o objektima, “predmet” o kojemu želimo razgovarati. Izvori mogu biti autori, knjige, mjesta, hoteli, pretraživački upiti itd.

Svaki izvor ima URI (Uniform Resource Identifier). URI može biti URL (Uniform Resource Locator, ili Web adresa) ili neka druga vrsta jedinstvenog identifikatora koji ne mora nužno voditi prema izvoru. URI schema nije definira samo Web lokacije, ali i najrazličitije objekte kao što su telefonski brojevi, ISBN brojevi i geografske lokacije.

Svojstva - Svojstva su određena vrsta izvora koji opisuju odnose između izvora.

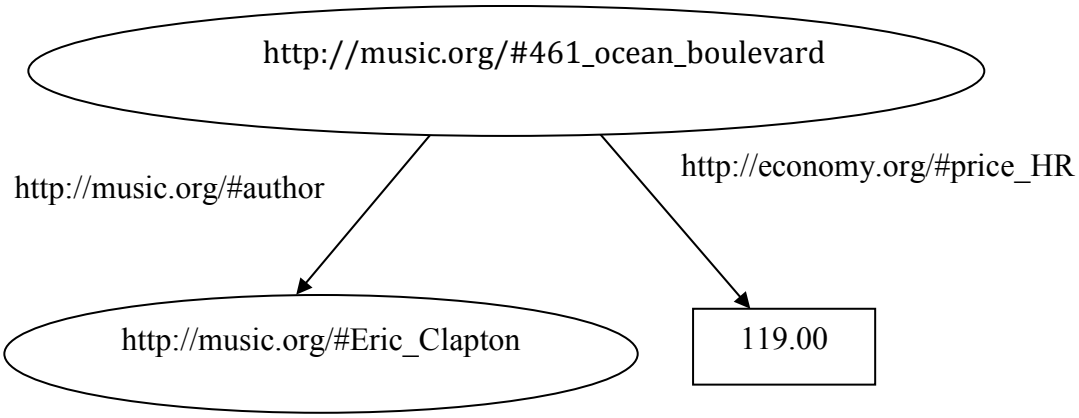
Izjave - Izjave potvrđuju svojstva izvora. Izjava je objekt – atribut – vrijednost trojka, sastojeći se od izvora, svojstva i vrijednosti. Vrijednost može biti bilo izvor ili literali.

Literali su doslovne vrijednosti, primjer u izjavi:

http://www.vecernji-list.hr autor «Ivica Ivić».

Ivica Ivić i autor su literali, jer nisu definirani putem URLrefa već im je vrijednost izravno iznesena u izjavi.

Na slici 8 (u nastavku) prikazan je primjer jedne strukture opisane pomoću RDF-a. Elipsom se označavaju resursi, a pravokutnikom podatkovne vrijednosti (string, integer, float...). Jedna RDF tvrdnja se sastoji od tri komponente: subjekt, predikat i objekt. Slika 9 prikazuje jedan primjer RDF tvrdnje. Ova tvrdnja govori da je Eric Clapton autor albuma "461 Ocean Boulevard". Sintaksa upotrebljavana u ovom primjeru naziva se Notation3, koja je ujedno i najjednostavniji oblik zapisa RDF datoteka.



Slika 8. Primjer RDF grafa

```
<http://music.org/#461_Ocean_Boulevard>
<http://music.org/#author>
<http://music.org/#Eric_Clapton>
```

Slika 9. Primjer RDF tvrdnje u trojkama

RDF model definira tri načina zapisivanja RDF izjava: grafovima, trojkama i XML-om. U modelu grafova, subjekti i objekti predstavljeni su čvorovima u grafu, dok su predikati predstavljeni usmjerenim lukovima koji povezuju čvorove. Informacije istovjetne onima predstavljenima grafom mogu se izraziti putem trojki, koje su ekvivalentne onima iz grafa. Za razliku od grafa gdje jedan čvor uvijek predstavlja jedan resurs te ga ja moguće iskoristiti za predstavljanje više izjava, u modelu predstavljanja izjavi putem trojki svaki put je potrebno eksplicitno navesti sva tri dijela izjave.

RDF Shema (eng. Resource Description Framework Shema)

RDF nam pruža mogućnost slaganja rečenica koje opisuju resurse i relacije među njima. RDF schema (RDFS) podiže stvari na još višu razinu omogućavanjem definiranja klasa (classes) i svojstava (properties), pristupom koji je jako sličan objektnom programiranju. Na taj način resursi mogu postati instance pojedine klase, a klase mogu postati podklase neke druge klase. Osim navedenog, RDFS pruža i mogućnost definiranja hijerarhije svojstava (npr. "hodati" je pod-svojstvo od "ići"), definiranja domene svojstava (koji resursi smiju imati dano svojstvo), te definiranja ranga svojstava (kakva svojstva resurs smije imati). RDF nema pretpostavke o primjeni na određenu domenu, niti definira semantiku domene. To ovisi o korisniku koji će to učiniti u RDF Schemi (RDFS).

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/ ... "
xmlns:rdfs="http://www.w3.org/ ... "
xml:base=http://zoo.org/animals
#>
<rdfs:Class rdf:ID="animal" />

<rdfs:Class rdf:ID="horse">
 <rdfs:subClassOf
     rdf:resource="#animal"/>

</rdfs:Class>

</rdf:RDF>
```

Slika 10. prikazuje primjer korištenja RDF scheme

RDF Schema (RDFS) jezik omogućuje definiranje tipova i svojstva koji se koriste u RDF izjavama, pridruživanje svojstva tipovima, te gradnju veza između različitih tipova i svojstava. Sam RDFS jezik je izgrađen kao skup predefiniranih tipova i predikata RDF-a. Odnosno, RDFS jezik je proširenje RDF jezika dodatnim unaprijed definiranim tipovima i svojstvima. Pomoću predefiniranih tipova i predikata grade se izjave koje opisuju nove tipove, njihova svojstva, te veze između različitih tipova i svojstva. Novodefinirani tipovi i svojstva naknadno se mogu koristiti u gradnji RDF izjava.

Tipovi resursa

Osnova gradnje novog tipa resursa je definiranje imena tipa resursa, te definiranje njegovih veza s ostalim tipovima resursa. Za definiranje novog tipa koriste se sljedeći predefinirani tipovi i predikati: 'rdfs:class', 'rdf:type', 'rdfs:subClassOf' i 'rdfs:resource'. Novi tip resursa određuje se tako da se stvori novi resurs (korištenjem 'rdf:ID' atributa), te mu se pridijeli tip 'rdfs:class' korištenjem predefiniranog predikata 'rdf:type'.

Na slici 11 (u nastavku) definiraju se četiri nova tipa resursa: 'Osoba', 'Student', 'Profesor' i 'Brucoš'. Sva četiri tipa pripadaju imeniku i rječniku određenom s URI-em '"http://primjer.org/sheme/osobe". Prilikom definiranja tipa 'Osoba' korišten je osnovni

oblik RDF izjave, dok definicije druga tri tipa koriste skraćeni oblik pisanja RDF izjava. Tipovi 'Student' i 'Profesor' definirani su kao podtipovi tipa 'Osoba' korištenjem predikata 'rdfs:subClassOf'. Podtip određenog tipa resursa je specijalizacija određenog tipa resursa. Na primjer, tip resursa 'Student' je posebna vrsta, odnosno specijalizacija, tipa resursa 'Osoba'. Značenje specijalizacije je sljedeće. Svaki resurs tipa 'Student' (specijaliziranog tipa) ujedno je i resurs tipa 'Osoba' (osnovnog tipa).

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://primjer.org/sheme/osobe">

  <rdf:Description rdf:ID="Osoba">
      <rdf:type rdf:resource=”http://www.w3.org/2000/01/rdf-schema#class"/>
  <rdf:Description>

   <rdfs:class rdf:ID="Student">
       <rdfs:subClassOf rdf:resource="#Osoba"/>
   </rdfs:Class>

    <rdfs:class rdf:ID="Profesor">
        <rdfs:subClassOf rdf:resource="#Osoba"/>
    </rdfs:class>

    <rdfs:class rdf:ID="Brucoš">
         <rdfs:subClassOf rdf:resource="#Student"/>
    </rdfs:Class>
</rdf:RDF
```

Slika 11. Definiranje tipova resursa RDFS-om

Višestrukim korištenjem specijalizacija mogu se graditi složene hijerarhije tipova. Na primjer, tip resursa 'Brucoš' je specijalizacija tipa 'Student' koji je specijalizacija tipa 'Osoba'. Prilikom takve uzastopne specijalizacije tipova vrijedi svojstvo tranzitivnosti specijalizacije. Odnosno svaki resurs tipa 'Brucoš' je istovremeno i tipa 'Student', ali zbog tranzitivnosti i tipa 'Osoba'. Prilikom gradnje hijerarhije tipova moguće je odrediti jedan tip kao specijalizaciju više vrsta tipova. Na primjer, moguće je tip resursa 'Profesor' definirati kao specijalizaciju tipova 'Osoba' i 'Nastavnik'. Tip

'rdfs:resource' je osnovni tip. Odnosno, svi drugi tipovi koji se pojavljuju su implicitno specijalizacije ovog tipa. Posljedica je da su svi resursi tipa 'rdfs:resource'.
Nakon što je definiran, tip resursa može se koristiti u RDF izjavama. Slika 12 prikazuje korištenje novo definiranog tipa resursa 'Student' u RDF izjavama.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:pr="http://primjer.org/”>
   <rdf:Descritpion rdf:about="http://primjer.org/predmeti/6">
      <pr:studenti rdf:nodeID="studenti_lokalno_ime"/>
   </rdf:Description>
   <rdf:Bag rdf:nodeID="studenti_lokalno_ime">
      <rdf:li rdf:resource=”http://primjer.org/Ivica”>
        <rdf:type rdf:resource=”http://primjer.org/sheme/osobe#Student”>
      </rdf:li>
      <rdf:li rdf:resource=”http://primjer.org/Marica”>
         <rdf:type rdf:resource=”http://primjer.org/sheme/osobe#Student”>
      </rdf:li>
      <rdf:li rdf:resource=”http://primjer.org/Perica”>
         <rdf:type rdf:resource=”http://primjer.org/sheme/osobe#Student”>
       </rdf:li>
    </rdf:Bag>
</rdf:RDF
```

Slika 12. Korištenje novodefiniranih tipova u drugom dokumentu

Dosad prikazane mogućnosti definiranja tipova resursa, omogućuju definiranje imena novih tipova, gradnju hijerarhije tipova, te pridruživanje tipa određenom resursu. Međutim, ne omogućuje se definiranje svojstava određenog tipa. Na primjer, nije moguće za tip resursa 'Osoba' odrediti da ima svojstva 'Ime' i 'Prezime'. U RDFS-u svojstva se (engl. properties) definiraju odvojeno od tipova resursa. Definirati svojstvo znači: imenovati predikat koji određuje svojstvo, definirati na koje tipove resursa subjekata i objekata se predikat odnosi, te definirati značenje samog predikata.
RDFS pruža osnovne mogućnosti za definiranje ontologija. Moguće je definirati tipove resursa, svojstva pojedinog tipa resurse, te veze među različitim tipovima resursa. Međutim u nekim slučajevima potrebno je ugraditi dodatno znanje u ontologije. Nove mogućnosti izražavaju se višim ontološkim jezicima, poput DAML, OIL, DAML+OIL i OWL-a.

4.3. OWL (Web Ontology Language)

OWL je jezik za stvaranje ontologija, razvijen kao sljedbenik iz RDF I RDFS-a, a baziran na formalnoj semantici XML-a. Formalnu osnovu za definiciju OWL-a (Web Ontology Language) daje deskriptivna logika. Resursi koji su definirani koristeći RDF i RDFS se mogu detaljnije specificirati koriteći OWL jezične konstrukte koji su preuzeti iz deskripivne logike. OWL je standardni Web jezik za opisivanje ontologija nastao kao evolucija jezika OIL, DAML i OIL+DAML. Sintaksno gledano, OWL je dodatni rječnik koji se koristi u RDF izjavama. OWL jezik, slično kao i RDF Schema jezik, omogućava definiranje tipova resursa, njihovih svojstava i veza. U odnosu na RDF Schema jezik, OWL dodatno omogućuje: definiranje karakteristika svojstava, ograničavanje vrijednosti svojstva putem tipova resursa, ograničavanje broja pojavljivanja svojstva, izražavanje jednakosti svojstava, definiranje novih tipova resursa operacijama nad skupovima postojećih resursa, nabrajanjem resursa, istovjetnošću te razlikom tipova resursa. OWL definira i nekoliko predikata za izražavanje jednakosti i/ili različitosti pojedinih resursa, te definira način dohvata OWL-om izražene ontologije. Rezultat dodatnih mogućnosti OWL-a jest mogućnost gradnje detaljnijih ontologija. Detaljnije ontologije bolje karakteriziraju svojstava resursa, tipove resursa i veze među resursima, što pospješuje postupke zaključivanja računalnih agenata koji koriste ontologiju. Informacije opisane OWL-om postaju znanje, a ne više skup podataka. Kada se OWL-u pridoda i mogućnost izvođenja činjenica koje nisu eksplicitno navedene, informacije stvarno postaju znanje, a OWL postaje jezik za predstavljanje znanja. OWL uvodi skup jezičnih oznaka, u kojem su najznačajniji konstrukti za uniju i presjek klasa (ili svojstava), konstrukti za definiranje kardinaliteta (dozvoljen broj instanci u pojedinoj klasi) te konstrukti koji označavaju logičke kvatifikatore (allValuesFrom i someValuesFrom). Predviđene su tri mogućnosti korištenja OWL ontologija.

OWL Full **-** upotrebljava osnovni OWL jezik dozvoljavajući njegovu kombinaciju na proizvoljan način s RDF-om i RDF Schemom. Ovo uključuje i mogućnost promjene prethodno definiranog temeljnog značenja (RDF-a ili OWL-a) njihovom međusobnom primjenom. Prednost OWL Full je njegova sintaktička i semantička kompatibilnost s RDF-om: svaki legalni RDF dokument je također legalni

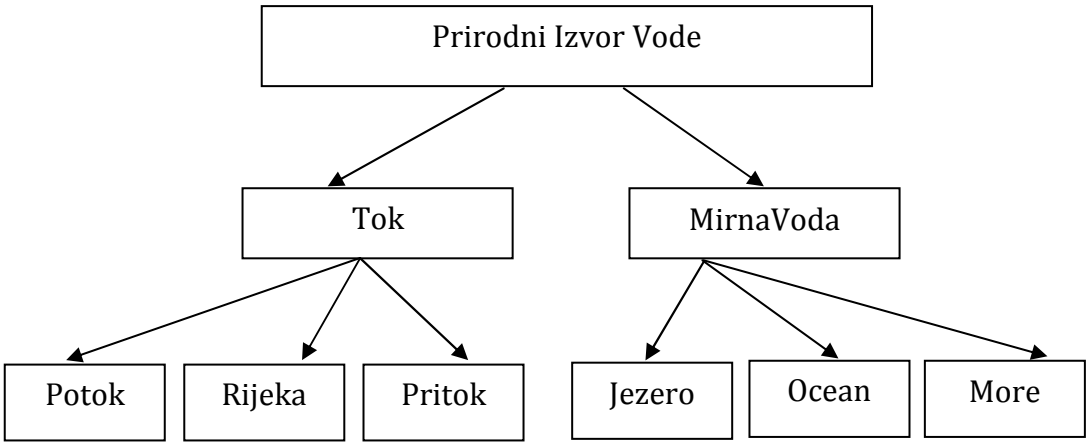
OWL Full dokument i svaki valjani RDF/RDF Schema zaključak je također valjani OWL Full zaključak. Maksimalna sloboda u korištenju OWL primitiva, ali nema garancije za izračunljivost.

OWL DL **-** OWL DL (deskriptivna logika) je podjezik od OWL Full koji posjeduje ograničenja u svezi upotrebe konstrukata iz OWL-a i RDF-a. Prednost je osiguranje efikasne razumijevajuće podrške. Nedostatak je gubljenje potpune kompatibilnosti s RDF-om. RDF dokument će se općenito trebati proširiti na određeni način i ograničiti na neki drugi način prije nego što bude legalni OWL DL dokument. Svaki legalni OWL DL dokument je legalni RDF dokument. Ograničavanje na OWL sintaksu koja je zračunljiva (semantički ekvivalentna jednom obliku deskriptivne logike).

OWL Lite – još jače ograničavanje sintakse zbog pojednostavljenja sintakse. Koristi se u aplikacijama gdje je dovoljna gotovo minimalna ekspresivnost ontologija. Daljna ograničenja OWL DL na podskup jezičnih konstrukata. Primjerice OWL Lite isključuje nabrajajuće klase, disjunktne izjave i proizvoljne cardinality. Prednost je jezik koji je lakši za dosegnuti korisnicima i lakši za implementirati. Nedostatak je ograničena izražajnost.

Razvojni programeri ontologija prihvaćanjem OWL trebaju razmotriti koji im podjezik najviše pristaje. Ukoliko su im potrebni izražajniji konstrukti osigurat će ih upotrebom OWL DL za razliku od upotrebe OWL Lite. Izbor između OWL DL i OWL Full ovisi o dodatku kojim korisnici zahtijevaju omogućavanje metamodela RDF Scheme (definiranje klasa, dodavanje svojstava na klase). Kada se usporedi upotreba OWL Full s OWL DL, razumijevajuća podrška je manje predvidljiva jer je potpuna OWL Full implementacija je nemoguća.

RDF Schema jezik omogućuje definiranje svojstava koje neki tip resursa ima. Međutim, tim svojstvima nije moguće pridružiti karakteristike. OWL jezik omogućuje definiranje karakteristika svojstava resursa. Koje sve karakteristike se mogu izraziti, te na koji način se karakteristike zapisuju u XML- u prikazano je na primjeru hijerarhije tipova resursa na slici 13.



Slika 13. Hijerarhija tipova resursa koji opisuju prirodne izvore vod

Rad s resursima

OWL definira i nekoliko predikata koji se koriste u dokumentima koji definiraju primjerke resursa. Dvije su skupine tih predikata: jedni omogućuju uvođenje OWL ontologije (dokumenata u kojima su definirani tipovi resursa i svojstava) u dokument koji opisuje primjerke resursa, a druga omogućuje izražavanje jednakosti i nejednakosti između primjeraka resursa. Slika 14 prikazuje primjer uvoza OWL ontologije, te definiranja jednakosti i različitosti resursa.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:vode="http://primjer.org/vode/prirodniizvor#"
         xml:base="http://SvijetskaMora.hr">

 <owl:Ontology rdf:about="http://primjer.org/vode/prirodniizvor">
    <owl:priorVersion rdf:resource="
http://primjer.org/vode/prirodniizvor/sijecanj2002#"/>
   <owl:versionInfo>prirodne vode v 2.0</owl:versionInfo>
   <owl:imports
rdf:resource="http://primjer.org/dokumenti/TipoviPrirodneVode.owl"/>
   </owl:Ontology>

  <vode:More rdf:ID="JadranskoMore">
      <owl:sameIndividualAs rdf:resource="http://WorldSeas.com#Adriatic" />
      <owl:differentFrom rdf:resource="http://SvijetskaMora#SredozemnoMore"/>
 </vode:More>
```

Slika 14. Korištenje OWL-a prilikom definiranja primjeraka resursa

Za razliku od RDF-a gdje se prilikom korištenja tipova resursa, samo navodi XML imenik u kojem se definicije tipova nalaze, OWL definira i lokaciju na kojoj se nalazi document u kojemu se nalaze definicije. Na slici 14, linijom "xmlns:vode="http://primjer.org/vode/prirodniizvor# " određuje se XML imenik u kojem se nalaze definicije rječnika prirodnih voda. Posebnim tipom resursa 'owl:ontology' definira se okacija ontologije (dokument s definicijom tipova i svojstva) koja odgovara tom imeniku, trenutno važeća inačica ontologije i jedna prethodna inačica. OWL omogućuje istovremeno izražavanje različitosti velikog broja resura pomoću predikata 'owl:AllDifferent' i 'owl:distinctMember'. U ovom poglavlju iznesen je samo kratki pregled mogućnosti OWL jezika.

5. PROTÉGÉ

5.1. Uvod u Protégé

Protégé je namjenjen za rad s ontologijama. Razvija se na Stanford University u suradnji s Univerity of Manchester. To je besplatan, opensoruce softverski alat za konstrukciju ontologija te rad s njima. Protégé je pisan u programskom jeziku Java, a za stvaranje bogatog grafičkog sučelja korištena je *swing* biblioteka. Budući da je pisan u Javi, prenosiv je i može se izvršavati na svim računalima koja imaju Java Virtual Machine. Podržava ga jaka zajednica razvojnih programera te akademskih, vladinih i korporacijskih korisnika, a koji ga koriste za rješavanje problema povezanim sa znanjem u područjima kao što su biomedicina, obavještajni poslovi te korporacijsko modeliranje. Pisan je kao framework što znači da je Protégé radna okolina na koju se mogu povezati razni softverski alati.

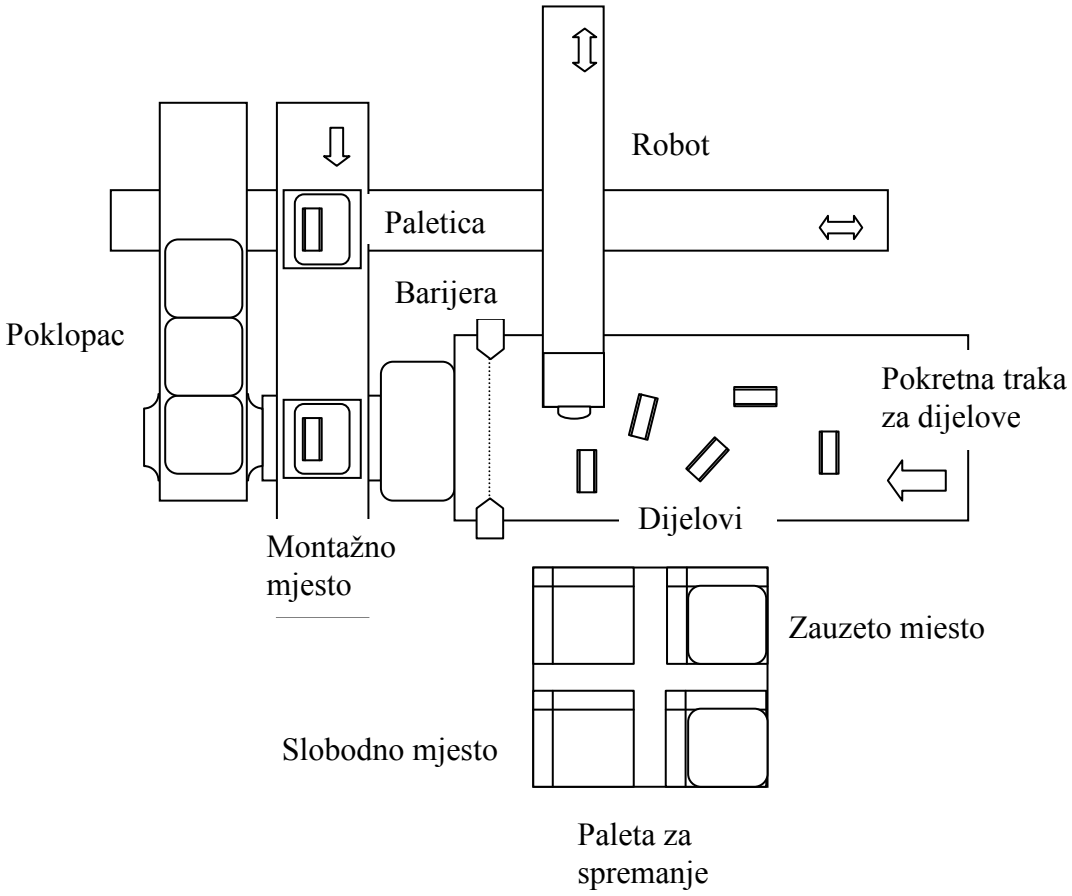
Protégé podržava dva glavna načina modelovanja ontologija, preko Protégé-Frames i Protégé-OWL, odnosno editovanje baza znanja preko predefinsanih formi-okvira i editovanje owl ontologija, što predstavlja najširu primjenu. Protégé editor može raditi sa različitim formama ontologija kao što su na primer: RDF(S), OWL, i XML Schema itd. Protégé se zasniva na Java platformi, što smo već spomenuli, može se nadograditi, podržava pluginove napravljene od strane korisnika, što predstavlja osnovu za brze modifikacije i razvoj rješenja prema potrebama.

Potrebno je sagraditi ontologiju za neki dio svijeta (domenu) i postavlja se pitanje odakle krenuti? Ono što je prirodno i najjednostavnije su koncepti. Čovjek će prvo uočiti koje se klase, tj. koncepti javljaju u domeni koju proučava.

U svrhu ovog rada, služilo se softverskim alatom Protégé 4.1 s alatom za rasuđivanje HermiT 1.3.5.

5.2. Projektni zadatak

Pomoću Protégé-OWL sustava za uređivanje ontologija i OWL jezika potrebno je razviti ontologiju koja je karakteristična za industrijsku primjenu kod poslova montaže. Navedena primjena podrazumijeva uporabu robota za obavljanje montažnih operacija te senzora za prikupljanje informacija iz okoline. Robot uzima dijelove sa pokretne trake i umeće ih u kučište, zatvara kučište i odlaže ih na paletu. Koristi se senzor optičke barijere koji daje informaciju o dostupnosti dijela za montažu te dva senzora koja nam daju informacije da li su kučište i njegov poklopac na odgovarajućem mjestu, također se koristi vizijski sustav koji je smješten na samom robotu.



Slika 15. Shematski prikaz radne okoline

Detaljan opis:

Pokretnom trakom dolaze dijelovi koji se umeću u kučište, a ono dolazi s drugom pokretnom trakom. Dijelovi dolaze u različitoj orijentaciji. Dio po dio dolazi do

optičke barijere koja se dolaskom dijela u njezino polje prekida i pokretna traka se zaustavlja. Kučište u koje se umeće dio nalazi se u maloj paletici i kreće se drugom pokretnom trakom. Kada je paletica sa kučištem stigla na mjesto montaže tj. kada se senzor aktivirao i kada je optička barijera prekinuta robot kreće sa operacijama. Prvo slijedi ″test″ paletice sa kučištem: da li je paletica koja je stigla na mjesto montaže spremna za umetanje dijela sa pokretne trake. Dalje slijedi ″inspection″ dijela na pokretnoj traci, da li je dio u odgovarajućoj orijentaciji za prihvat. Zatim slijedi operacija ″pick up″ kojom robot uzima dio sa pokretne trake i umeće ga u kučište. Nakon umetanja robot uzima poklopac, kojeg je senzor detektirao da je na mjestu i zatvara kučište. Iduća operacija je ″test″ palete tj. da li je na paleti slobodno mjesto za završnu operaciju storage kučišta. Ukoliko je odgovor pozitivan robot uzima kučište sa paletica i sprema ga na veliku paletu. Robot završava sa radom dok veliku paletu popuni sa kučištima.

5.3. Klase

Kako bi se napravila bazu znanja, odnosno skup operacija montaže, služilo se člankom „Probabilistic Approach to Robot Group Control“, koji je nastao na Fakultetu stojarstva i brodogradnje. Pomoću tog članka, sastavljen je skup montažnih operacija, zajedno sa industrijskom okolinom, te senzorima iz nje. Ovi podaci su razvrstani u pripadne klase.

Odabirom kartice *Classes* ekran će biti podijeljen na tri dijela. Možemo uočiti sljedeće dijelove ekrana: *Class hierarchy, Annotations te Description*.
Promatrajmo *Class hierarchy* dio ekrana. Tamo se pojavljuje već jedna stavka. To je osnovni koncept imena *Thing* koji predstavlja vašu domenu. Odnos između koncepata se gradi hijerarhijski, a *Thing* je koncpet koji je najviše hijerarhije. Svaki novi koncept koji dodamo mora biti u hijerarhiji ispod koncepta *Thing*. Hijerarhijski odnos između klasa reprezentira samu hijerarhiju između pojmova.

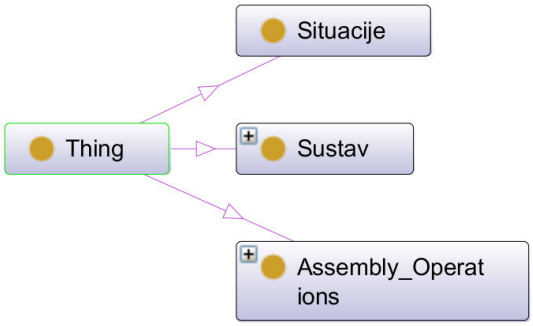
Klikom na pojedini koncept u ovom dijelu ekrana pruža se mogućnost korištenja nekih od sljedeća tri gumba: *Add subbclas, Add sibiling class, Delete selected classes*. Gumbi su reprezentirani sličicom koja bi trebala asocirati na ono što rade, a puno ime

postaje vidljivo tek kada se kursor miša neko vrijeme zadrži nad gumbom. Oni, redom, dodaju klasu u hijerarhiju jedan nivo niže od pojma koji je obilježen prije stiskanja gumba; dodaju klasu na isti nivo u hijerarhiji kao što je i obilježeni pojam; brišu klasu iz hijerarhije.

Korištene su sljedeće klase:

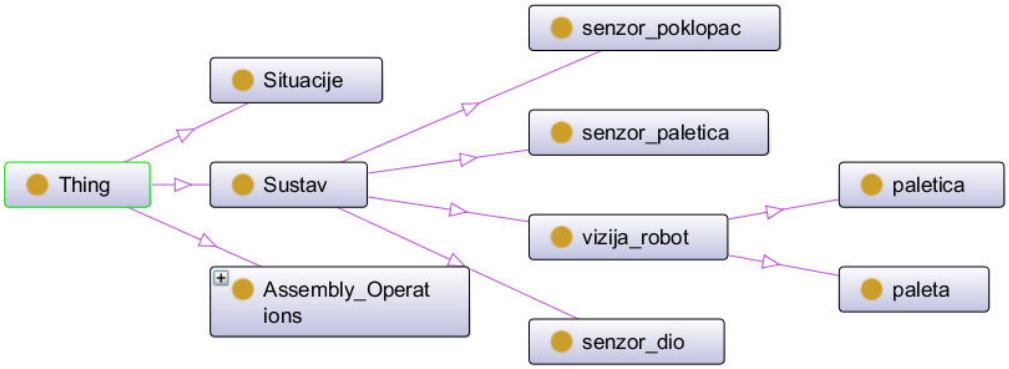
- *Assembly_operations*, koja služi kao baza montažnih operacija; montažne operacije su podijeljene u razne kategorije radi lakšeg snalaženja;
- *Situacije*, koja sadrži moguće kombinacije stanja senzora sustava (individua);
- *Sustav*, koja sadrži listu svih senzora koji se nalaze u sustavu;

U mojoj ontologiji imamo tri glavna koncepta koja se nalaze odmah ispod koncepta *Thing*, što možemo vidjeti na slijedećoj slici.



Slika 16. Najviši hierarhijski nivo ontologije

Počnimo sa klasom *Sustav*, ona nam predstavlja senzore koje koristimo u sustavu. Da bi robot mogao raditi samostalno, bez obzira na nivo samostalnosti, on mora biti "svjestan" sebe i svoje okoline, tj. robot uz pomoć senzora dobiva podatke o sebi i o okolini. U mojem radu slučaj okoline je pojednostavljen te koristimo tri senzora i kameru koja je smještena na robotu, što je prikazano na slici 16.



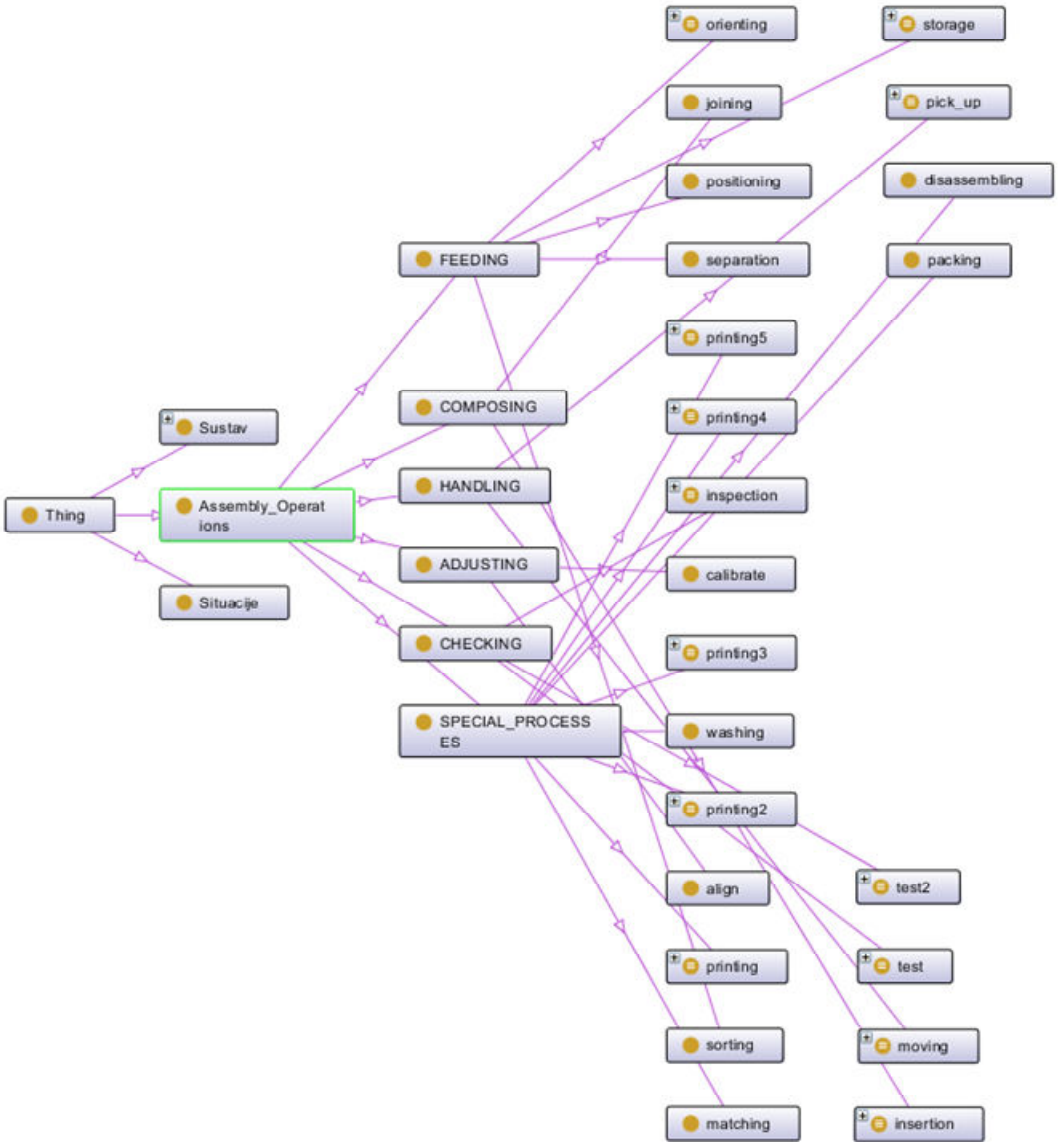
Slika 17. Prikaz senzora sustava

Klasa *Sustav* sastoji se od podklasa:

- *Senzor_dio* – optička barijara koja detektira dolazak dijelova na poretnoj traci
- *Senzor_paletica* – induktivni senzor koji detektira da je paletica sa kučištem stigla na montažno mjesto
- *Senzor_poklopac* – kapavcitivni senzor koji detektira da je poklopac za zatvaranje kučišta na odgovarajućem mjestu
- *Vizija_robot* – vizijski sustav koji nam služi za provjeu mjesta montaže, orijentaciju dijela i provjeru za spremanje dijelova na paletu

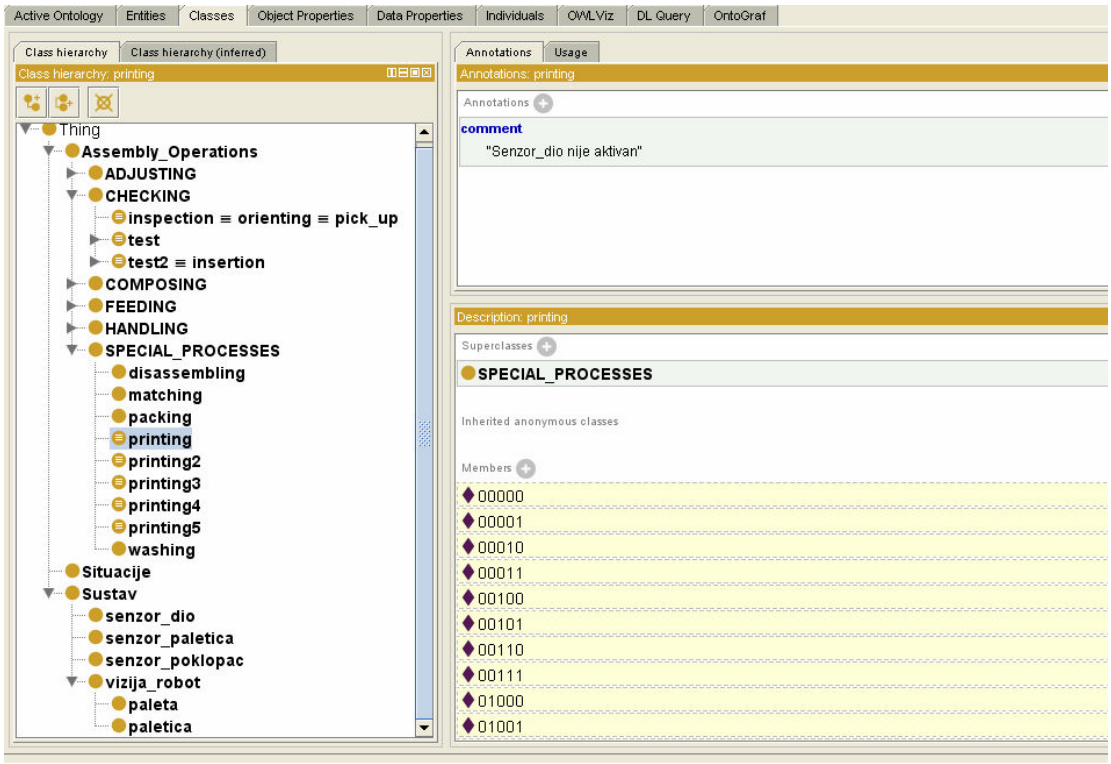
Klasa *Assembly_operations* sastoji se od niza montažnih operacija:

- *ADJUSTING*
  - *align*, *calibrate*
- *CHECKING*
  - *inspection*, *test*
- *COMPOSING*
  - *insertion*, *joining*
- *FEEDING*
  - *orienting*, *positioning*, *separation*, *sorting*, *storage*
- *HANDLING*
  - *pick_up*, *moving*
- *SPECIAL_PROCESSES*
  - *dissasembling*, *matching*, *packing*, *printing*, *washing*



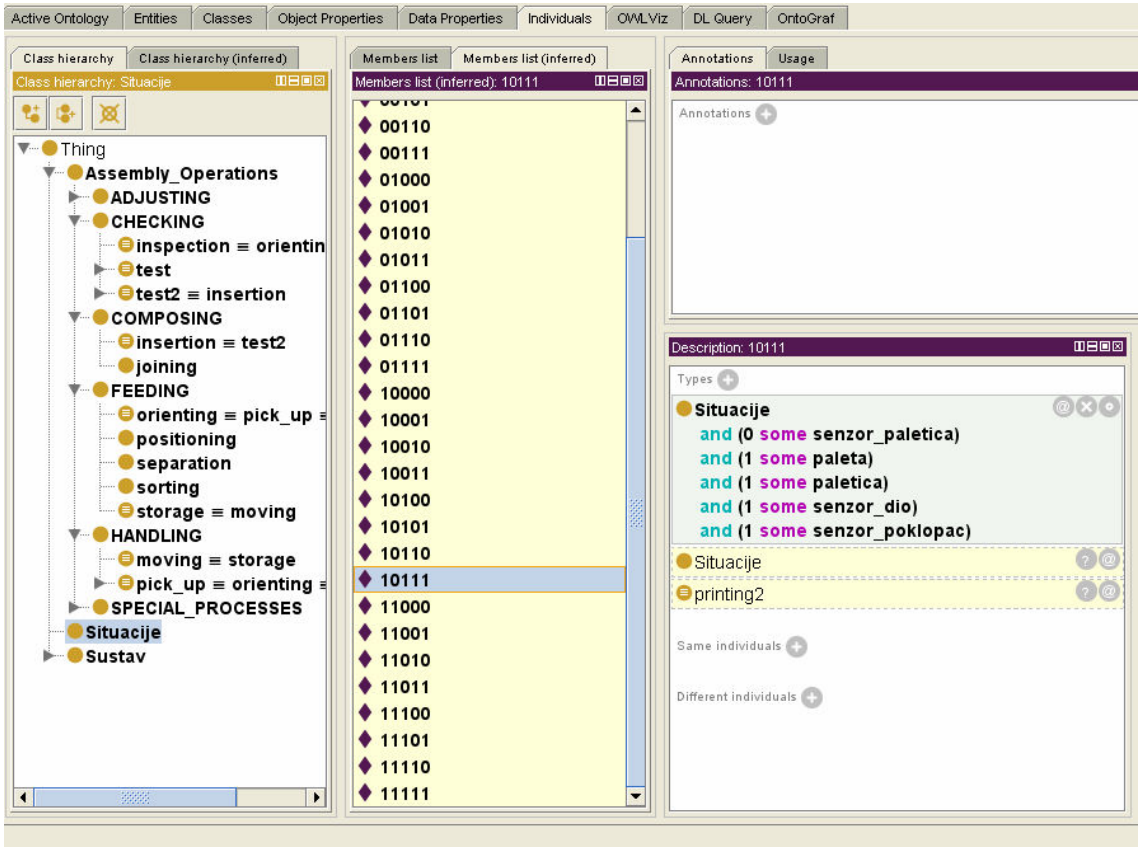
Slika 18. Prikaz montažnih operacija unutar ontologije

Promotrimo sada *Description* dio ekrana. Odaberemo li neki koncept iz hijerarhije, u ovom dijelu ekrana će se pojaviti informacije o njemu kao što su *Equivalent classes, Superclasses, Inherited anonymous classes, Members, Keys, Disjoint classes*. Ukoliko se uz pojedinu rubriku javlja gumb sa znakom plus, u nju je moguće dodavati nove podatke.



Slika 19. Class hierarchy

Klasa *Situacije* sadržava individue, koje predstavljaju stanja senzora sustava, pomoću kojih se vidi svrha ove ontologije; ukoliko znamo stanje senzora sustava, ontologija će rasuditi koje moguće montažne operacije će se izvršiti.



Slika 20. Class Situacije sa individuama

5.4. Relacije

Dalje, kako bi se povezala klasa *Assembly_operations*, tj. njezine podklase sa klasom *Sustav*, odnosno kako bi se pridružili senzori sustava pripadnim operacijama, koriste se svojstva/relacije (*Object properties*).

Promotrimo sada dio u Protégé-u za rad sa svojstvima. Otvorimo *Object Properties* karticu. Dočekat će nas ekran podijeljen na četiri dijela: *Object property hierarchy, Annotations, Characteristics, Description*. U dijelu ekrana *Object propery hierarchy* možemo davati svojstva objektima, tj. zadavati veze između između pojedinih objekata. Ovdje, kao i kod klasa, imamo hijerarhijsku organizaciju što nam omogućuje izjavljivanje da sve individue koje zadovoljavaju neko svojstvo moraju istovremeno zadovoljavati i njegovo nadsvojstvo. Drugi najvažniji dio jest *Description* dio ekrana. Nakon što se iz hijerarhije odabere svojstvo, u ovom dijelu ekrana se pojavljuju neke od informacija o svojstvu. Nas će zanimati rubrika *Domains* koja će

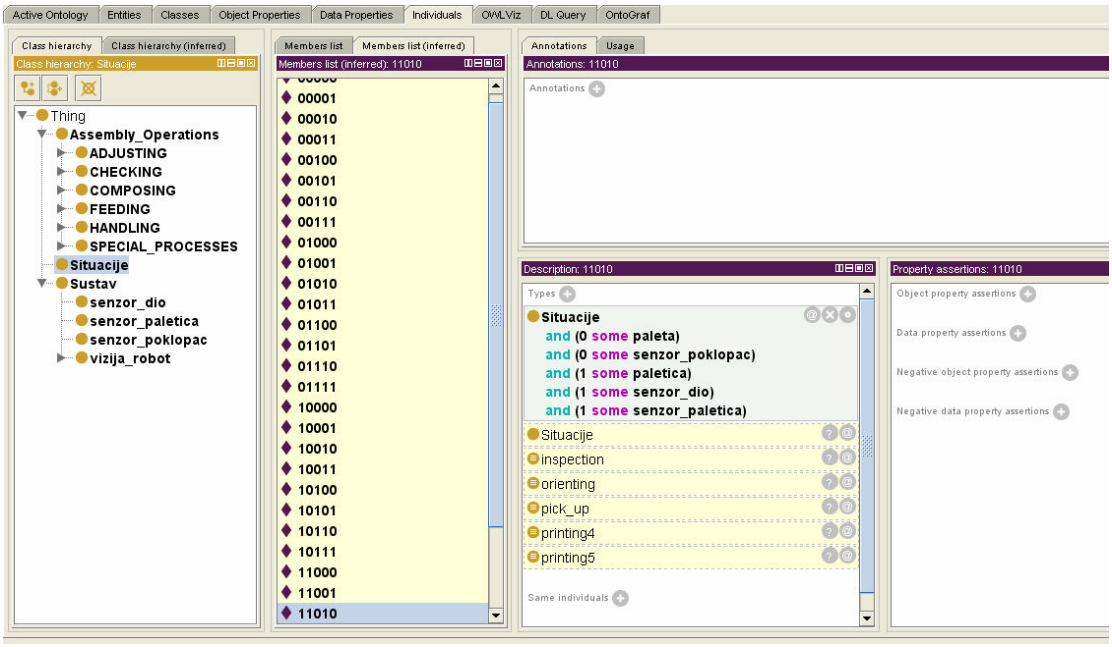
sadržavati domenu relacije (svojstva) te rubrika *Ranges* koja će sadržavati sliku relacije (svojstva). Spomenimo još i rubriku *Characteristics* u kojoj možemo detaljnije opisati neku svojstvo tako što ćemo opisati kakvom je relacijom to svojstvo opisano.

Relacije/svojstva koja se koriste:

- 0/1, koja povezuje operacije montaže i stanje senzora, odnosno kada se u klasi *Assembly_operations* odabere neka operacija, da bi joj se pridružili određeni senzori, koristi se ova relacija (sintaksa: „*ime_relacije* some *naziv senzora* „);

5.5. Individue

Rad s individuama započinje odabirom kartice *Individuals* nakon čega će ekran biti podjeljen na 5 dijelova. Jedan dio nam je *Class hierarchy* u kojem ćemo birati klasu s čijim individuama ćemo raditi. Preostala četiri dijela služe za opisvanje individue.
Nakon što se u *Class hierarchy* dijelu odabire klasa, u dijelu ekrana s imenom *Members list* toj klasi možemo dodavati individue pritiskom na tipku *Add individual*.
Nakon dodavanje individue u klasu, dogodit će se dvije stvari: u *Members list* dijelu će individua biti odabrana, a u *Description* dijelu ćemo vidjeti da je rubrika *Types* neprazna. U toj rubrici piše klasa kojoj odabrana individua pripada. Pritiskom na gumb sa znakom plus koji se nalazi pokraj imena rubrike možemo dodati još neke klase kojima će ova individua pripadati.



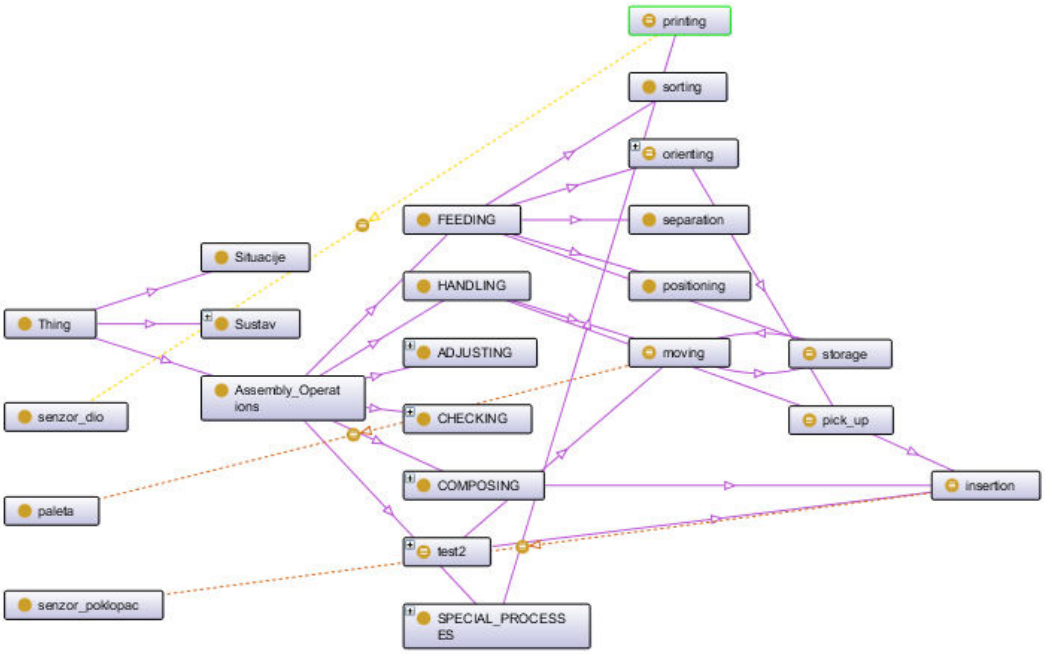
Slika 21. Prikaz individua

Individue u mojem slučaju su kombinacije stanja senzora koje se mogu dogoditi u sustavu, pošto sustav ima 3 senzora i dvije vizijske provjere dolazi se do ukupno do 32 moguće kombinacije. Naime svaki senzor ima dva moguća stanja, u binarnom zapisu ili 1 ili 0, tj. ili je aktivan ili nije.

5.6. Ontološki graf

Nakon što se izgradi neka ontologija, ali i tijekom same izgradnje, možemo pogledati što se nalazi iza *OntoGraph* kartice. Pritiskom na samu karticu, ekran će biti podijeljen na ova dva dijela: *Class hierarchy* te *OntoGraph* dio. Odabirom nekog koncepta iz *Class hierarchy* dijela on se pojavljuje na bijeloj podlozi *OntoGraph* dijela.

To je prikaz do sada opisanih dijelova ontologije sa klasama i individuama te relacijama između njih.



Slika 22. Prikaz pomoću OntoGraph-a

Korisniku je dana mogućnost biranja kako će graf biti strukturiran, koje klase i individue će biti prikazane te koje relacije će biti prikazane. Osim toga, dana je mogućnost pohranjivanja grafičkog prikaza ontologije u slikovnu datoteku kako bi se prikaz ontologije mogao koristiti i izvan Protégé-a.

5.7. Rasuđivanje

Rasuđivanje je važan dio u izgradnji ontologija, gdje ima dvostruku ulogu. Jedna uloga je osiguranje da je do sada izgrađena ontologija logički konzistentna (nema kontradiktornih opisa). Druga uloga je olakšavanje stvaranja ontologija budući da je rasuđivanje u stanju automatski, iz do sada sagrađene hijerarhije klasa, izgraditi novu hijerarhiju gdje su izvedeni svi mogući zaključci.

U Protégé verziji 4, hijerarhija koju smo sami izgradili se naziva *asserted hierarchy*, dok se hijerarhija koja sa dobije pokretanjem rasuđivanja naziva *inferred hierarchy*. Da bismo dobili izvedenu hijerarhiju, potrebno je pokrenuti rasuđivanje iz izbornika *Reasoner*. Najpoznatiji programi za rasuđivanje, bazirani na OWL semantici, su *Pellet, FaCT++ i HermiT*. *Pellet* dolazi s Protégé verzijom 3.4.4, dok preostala dva

dolaze s novim verzijama. U posljednjoj verziji dolazi *HermiT*, open source rasuđivanje koje se razvija na University of Oxford, a koje se može pohvaliti da je mnogo efikasniji od svih rasuđivanja do sada.

5.8. Rezultati ontologije

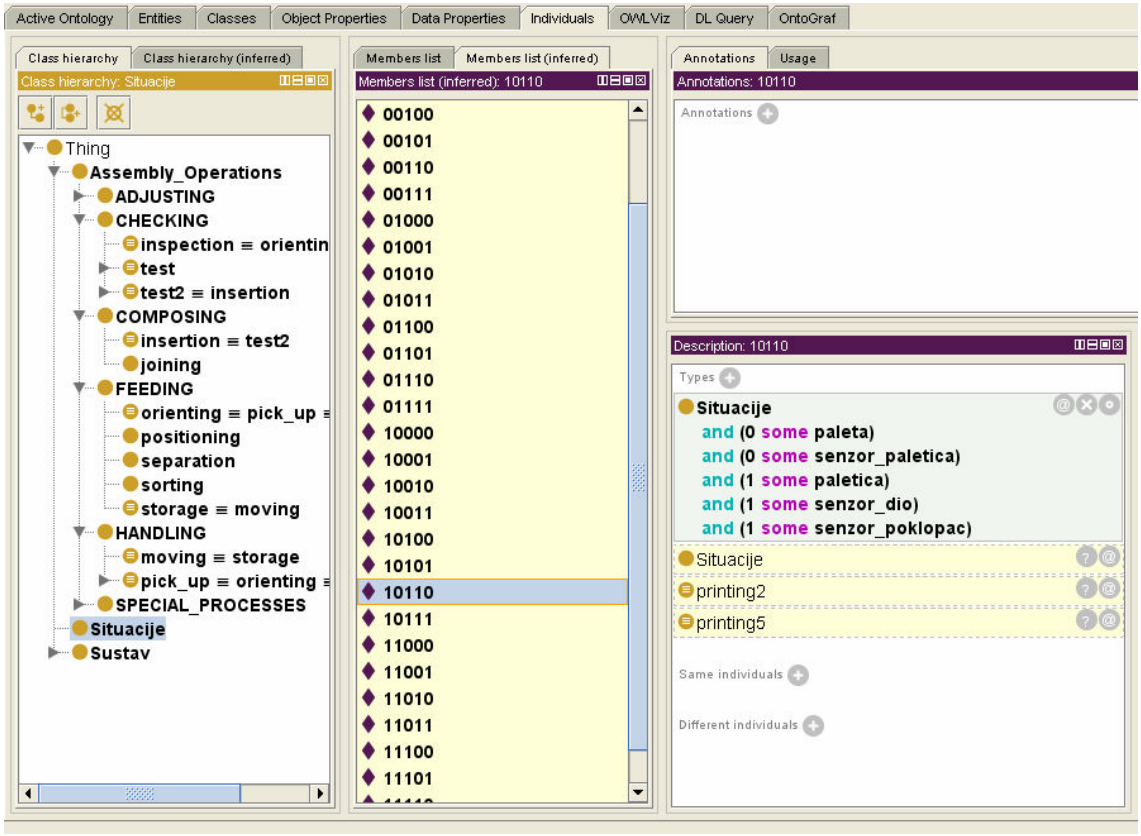
Na iduća tri primjera prikazat ćemo rezultate ontologije, odnosno vidjet ćemo koje riješenje je sustava u ovisnosti o stanjima senzora sustava.

Primjer 1.

Stanje na ulazu je sljedeće:

- *Senzor_dio* – aktivan
- *Senzor_paletica* – nije aktivan
- *Senzor_poklopac* – aktivan
- *Paletica* – vizija ok
- *Paleta* – vizija nije ok

Ovakvo stanje sustava zapisano je sa *Individual* ***10110***. Na slici 23 vidjet ćemo koji je rezultat ovakvog stanja na ulazu, tj. na senzorima.



Slika 23. Primjer 1 – rasuđivanje ontologije

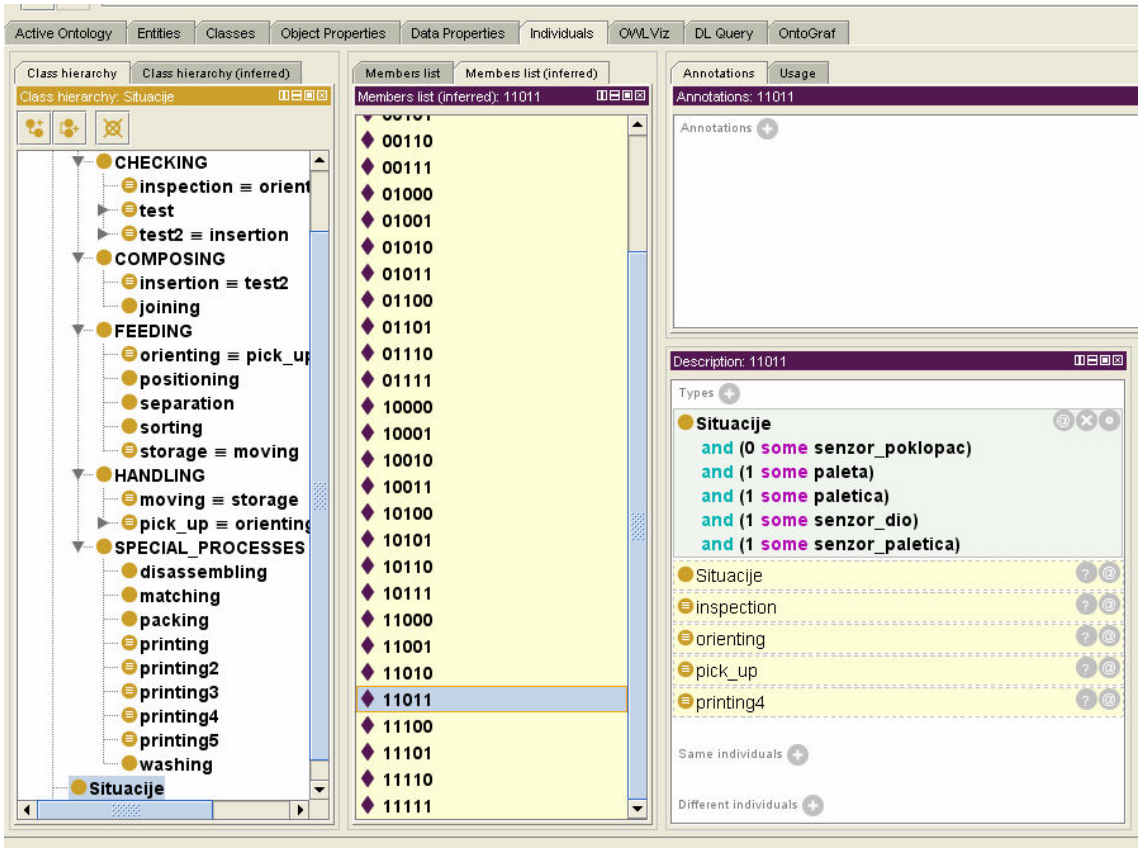
Sa slike jasno je vidljivo da kao rezultat stanja senzora kao što je zadano u primjeru ontologija daje riješenje montažnih operacija *printing2,* i riješenje *printing5*. Ako u klasi *printing2* pogledamo svojstvo *Annotations* vidjet ćemo da nam piše *„Senzor_paletica nije aktivan“*, na isti način dobimo objašnjenje za klasu *printing5*, a ono glasi *„Vizija_robot paleta nije zadovoljena“ i „Paleta za spremanje je popunjena“*.

Primjer 2.

Stanje na ulazu je sljedeće:

- *Senzor_dio* – aktivan
- *Senzor_paletica* – aktivan
- *Senzor_poklopac* – nije aktivan
- *Paletica* – vizija ok
- *Paleta* – vizija ok

Ovakvo stanje sustava zapisano je sa *Individual* ***11011***. Na slici 24 vidjet ćemo koji je rezultat ovakvog stanja na ulazu, tj. na senzorima.



Slika 24. Primjer 2 – rasuđivanje ontologije

Sa slike jasno je vidljivo da kao rezultat stanja senzora kao što je zadano u primjeru ontologija daje riješenje montažnih operacija *inspection,* riješenje *orienting,* riješenje *pick_up* i riješenje *printing4*.

Iz ovakvih rijšenja lako dolazimo da zaključka da je robot obavio montažne operacije *inspection, orienting* i *pick_up*, nakon čega je trebao zatvoriti kučište sa poklopcem, al nam rezultat *printing4* daje objašnjenje *„Senzor_poklopac nije aktivan“*.

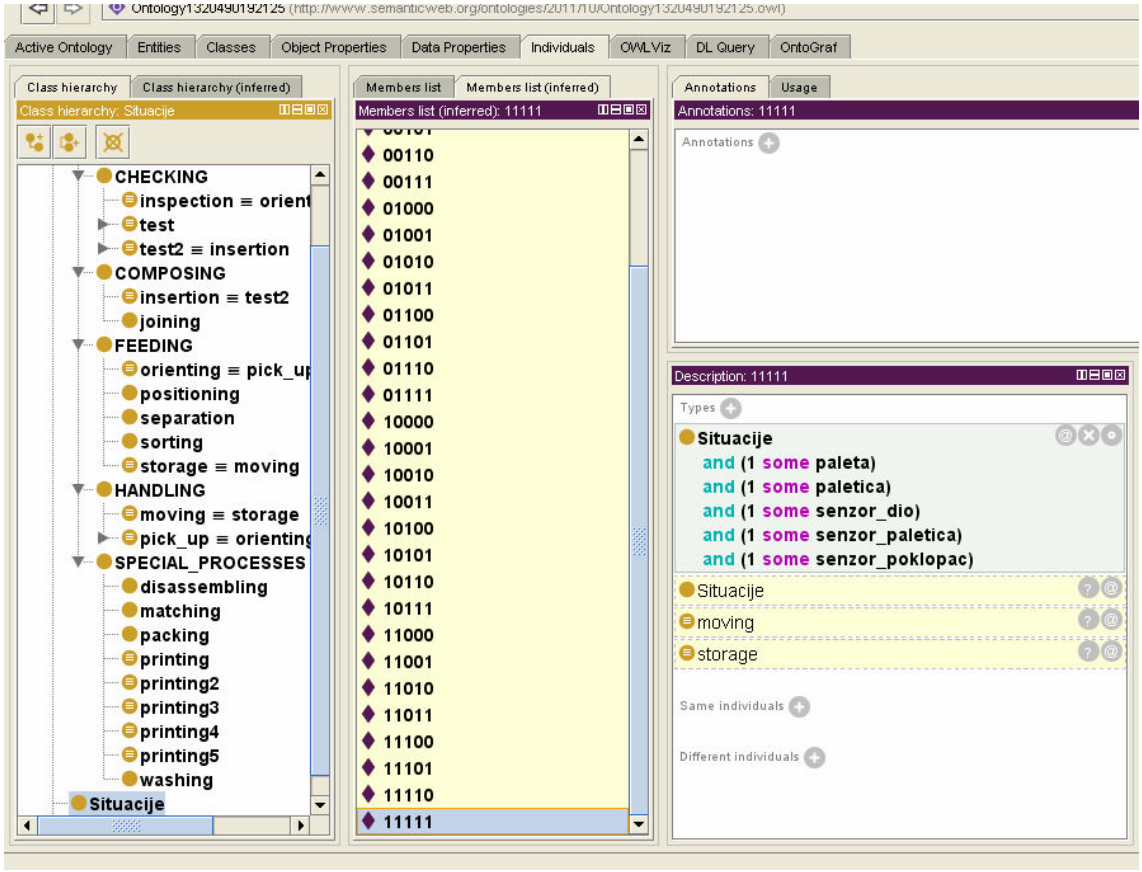
Primjer 3.

Stanje na ulazu je sljedeće:

- *Senzor_dio* – aktivan

- *Senzor_paletica* – aktivan
- *Senzor_poklopac* – aktivan
- *Paletica* – vizija ok
- *Paleta* – vizija ok

Ovakvo stanje sustava zapisano je sa *Individual* ***11111***. Na slici 25 vidjet ćemo koji je rezultat ovakvog stanja na ulazu, tj. na senzorima.



Slika 25. Primjer 3 – rasuđivanje ontologije

Promatrajući riješenje ontologije kad su svi senzori aktivni, tj. kada dobivamo dobre informacije izvrše se odgovarajuće montažne operacije. Robot završava sa montažnim operacijama *moving* i *storage* kučišta sa umetnutim dijelom i poklopcem na paletu.

6. ZAKLJUČAK

Ontologije su osnovna tehnologija koja omogućuje nadogradnju običnog weba. Jednostavno izloženo, ontologije su riječnici termina koji se koriste u određenom području ljudskog djelovanja te znanje koje povezuje te termine. U semantičkom web-u, ontologije se koriste prilikom označavanja web-om izloženih podataka i usluga standardnim terminima. Inteligentni agenti semantičkog web-a čitaju oznake navedene uz podatke i usluge, te zbog korištenja standardnih termina ontologija, razumiju njihovo značenje. Na osnovu znanja ugrađenog u ontologiju agenti donose različite zaključke, te pomožu korisniku prilikom pretraživanja podataka i korištenja usluga semantičkog web-a. Dva su glavna uvijeta za gradnju semantičkog weba:

- gradnja standardnog jezika za izražavanje ontologija, te
- gradnja samih ontologija.

Jezik za izražavanje ontologija mora biti standardiziran jer ga moraju razumijeti svi agenti koji djeluju na semantičkom Webu. Zbog zahtjeva za standardiziranošću, XML jezik se nametnuo kao osnovni jezik za gradnju ontologija. XML omogućuje gradnju jednostavnih ontologija i razmijenu podataka zasnovanih na tim ontologijama. Iako jednostavan, XML jezik čini osnovu gospodarenja podacima i elektroničkog poslovanja današnjice. Na osnovu XML-a, pojavljuju se i složeniji jezici za izražavanje ontologija RDF i OWL. Ovi jezici omogućuju gradnju složenijih ontologija koje sadrže veću količinu znanja, a time omogućuju i gradnju boljih agenata. Za razliku od gradnje standardnih jezika ontologija, u području gradnje samih ontologija nije se daleko odmaklo. Jedina široko prihvaćena ontologija je Dublin Core ontologija koja omogućuje označavanje web stranica podacima o autoru, vremenu nastanaka, jeziku izlaganja te ključnim riječima. Nepostojanje široko prihvaćenih i javno dostupnih ontologija glavna je prepreka pojavljivanju semantičkog web-a.

Izradom same ontologije, koja je vrlo jednostavna smatram da se ova ontologija može dodatno proširiti i iskoristiti na konkretnim primjerima, za sada u laboratoriju, a kasnije i u samoj industriji. Daljni rad trebao bi se temeljiti na predstavljanju ontologija na webu te povezivanju sa sustavom.

7. LITERATURA

1. Ontologije i semantički web, Ivan Benc, seminar, Fakultet elektrotehnike i računalstva, http://www.zemris.fer.hr/predmeti/krep/Benc.pdf

2. Ontologije na semantičkom web-u, *Catherine Legg***,** University of Waikato, Hamilton-Tauranga, New Zealand

3. http://en.wikipedia.org/wiki/Ontology
4. Predstavljanje znanja zasnovano na integraciji ontologija i Bayesovih mreža, doktorska disertacija, Marin Prcela, Sveučilište u Zagrebu, Fakultet elektrotehnike i računalstva, 2010.
5. Foundations of the Semantic Web: Ontology Engineering, Alan Rector & colleagues, Special acknowledgement to Jeremy Rogers & Chris Wroe, University of Southampton

6. Deskriptivna logika, Alan Jović, Fakultet elektrotehnike i računarstva, Zagreb, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave,

7. Uvod u OWL, Damir Kirasić, Sveušilište u Zagrebu, Fakultet elektrotehnike i računarstva, Centar informacijske potpore
8. Semantički web, Krešimir Pavić, Sveučilište u Zagrebu, Fakultet elektrotehinke i računalstva, http://www.zemris.fer.hr/predmeti/krep/Pavic.pdf

9. Predstavljanje ontologija na Web-u, Marin Prcela, Laboratorij za informacijske sustave, Zavod za Elektroniku, Institut Ruđer Bošković, http://www.fer.unizg.hr/_download/repository/Marin_Prcela_SW.pdf

10. Metode učenja ontologija – trenutno stanje i problemi, Damir Jurić, Sveučilište u Zagrebu, Fakultet elektrotehnike i računalstva
11. CYC Ontologija, Dubravko Antunović, Prirodoslovno-matematički fakultet: Matematički odsjek, Sveučilište u Zagrebu
12. Protégé, seminar, Saša Stanko, Prirodoslovno-matematički fakultet: Matematički odjel, Zagreb, 2010.
13. A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3, Matthew Horridge, The University Of Manchester, 2011

14. Knowledge Representation in Protégé – OWL, http://www.co-ode.org/resources/tutorials/iswc2005

15. Making OWL Easier: Practical Ontology Development in using Protégé-OWL-CO-ODE Tools, Alan Rector, Hai Wang, Jeremy Rogers, Information Management Group / Bio Health Informatics Forum, Department of Computer Science, University of Manchester
16. Probabilistic Approach to Robot Group Control, Tomislav Stipančić, Bojan Jerbić, Petar Čurković, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje
17. http://co-ode.org
18. http://owl.cs.manchester.ac.uk
19. http://www.semanticweb.org
20. http://protege.stanford.edu
21. http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary