Protege

Mladen Vidović
mvidovic@singidunum.ac.rs

Univerzitet Singidunum
Centar Novi Sad

26. maj 2022.

Sadržaj

1 Protege

Download i instalacija

- Protege je open source alat za rad sa ontologijama koji koristi OWL Web Ontology Language za predstavljanje deskriptivne logike. OWL je baziran na RDF-u.
- Može se preuzeti sa: https://protege.stanford.edu/products.php#desktop-protege
- Zahteva java runtime environment (JRE).

Ontologija

- Ontologija je opis koncepata i veza između koncepata za neki domen. Koristimo ih kao baze znanja, u kojima se nalaze svi termini, koncepti, uloge i veze između njih.
- Služe za konstrukciju sistema baziranih na znanju, u kombinaciji sa modulom za zaključivanje i interfejsom (aplikacijom).
- Ovi sistemi omogućuju primenu ekspertskog znanja u aplikacijama poput recommender sistema.
- Baza znanja se sastoji iz dva dela, TBox, koji čine termini, odnosno koncepti, uloge i veze između njih, i ABox koji čine konkretne vrednosti individuala.
- Razlika u odnosu na baze podataka - baze podataka nam pokazuju kako se podaci skladište, baze znanja opisuju domensko znanje.

Ontologija

- Za kreiranje nove ontologije u Protege-u biramo file -> new. Ovo otvara novi prozor sa novom ontologijom.
- Da bismo imenovali ontologiju, podešavamo njen IRI (International Resource Identifier, kao URI, samo sa većim brojem dozvoljenih znakova).
- Možemo takođe navesti i IRI za verziju ontologije.
- Služe za jedinstvenu identifikaciju ontologije, ukoliko ontologija ima samo IRI ali ne version IRI, ne bi smela da postoji još jedna ontologija sa istim IRI-om i bez version IRI-a. Slično tome, ne bi smele da postoje dve ontologije sa istim IRI i version IRI vrednostima.

Klase

- Klase predstavljaju termine, odnosno uloge u domenu - skup individualnih elemenata.
- Svaka korisnička klasa implicitno nasleđuje osnovnu klasu owl:thing, odnosno sve što kreiramo je nešto (nekakva stvar).
- Da li je neki individualni element pripadnik klase zavisi od njegovog logičkog opisa.
- Klasa ne mora da ima ime, može biti izraz, ali je poželjno formirati je tako da može da sadrži individualne elemente.

Dodavanje klase

- Da bismo dodali podklasu, odaberemo nadklasu i zatim opciju add subclass (prva sa leve u podmeniju za klase), ili desni klik na klasu -> add subclass.
- Dodati klasu Electronics shop kao podklasu od owl:thing.
- Dodati klasu Person kao podklasu od owl:thing.
- Kao podklase Person dodati klase Buyer i Salesman.
- Dodati Electronic device kao podklasu od thing (ili sibling od person).
- Možemo odmah dodati i celu hijerarhiju. Desni klik na nadklasu -> add subclasses.
- Na ovaj način dodati Handheld device (Phone) i Computer (Laptop, Desktop), pri čemu su Handheld device i Computer podklase od Electronic device; Phone je podklasa od Handheld device; a Laptop i Desktop su podklase od computer.
- Jedna klasa može da ima više klasa kao pretke. Kreirati klasu Tablet device, i kao nadklase dodati i handheld device i computer.

Disjunkcija klasa

- OWL ne predstavlja zatvoren svet.
- Gde god u hijerarhiji da se nalazi neka klasa, pretpostavlja se da se klase mogu preklapati. U našem slučaju, ako ne navedemo eksplicitno suprotno, prodavac može istovremeno da bude telefon.
- Zato je potrebno definisati disjunkciju klasa (disjoint with svojstvo). Kada su dve klase disjunktne, nikada ne mogu da budu podklasa jedna drugoj, i nijedan individual ne može da bude istovremeno član obe klase.
- Disjunkcija klasa se nasleđuje, tako da ako definišemo da je osoba disjunktna sa elektronskim uređajem, to će se propagirati na podklase, tako da telefon ne može da bude kupac.
- Da li je handheld device disjunktno sa computer?

Rezoner (Reasoner)

- Rezoneri služe za zaključivanje informacija koje nisu eksplicitno navedene u ontologiji.
- Postoji više rezonera, sa različitim implementacijama.
- Na primeru koristiti ugrađeni rezoner, HermiT.
- Rezoneri takođe proveravaju da li je ontologija zadovoljiva i konzistentna. Prebaciti pogled hijerarhije klasa na Inferred, i proveriti da li postoje neki problemi sa ontologijom.
- Nekonzistentne klase će biti označene crvenom bojom, premeštene klase plavom bojom.

Rezoner

- Klasa Tablet je ekvivalentna sa owl:nothing, i označena je crvenom bojom zato što je nekonzistentna.
- Podklasa je dve klase koje su međusobno disjunktne.
- Možemo proveriti i objašnjenje klikom na upitnik u interfejsu.

Veze

- Veze se uspostavljaju između:
  - individuala i individiuala - object property
  - individuala i vrednosti - data property
- Postoji i annotation property za definisanje metapodataka.
- Property mogu da formiraju hijerarhiju, odnosno moguće je da property nasleđuje drugi property (istog tipa).

Veze

- Da bismo kreirali vezu između individuala, prelazimo na Object properties tab.
- Već je definisan topObjectProperty koji povezuje sve owl:thing individuale.
- Naši object property-i će biti njegovi potomci. Kreirati property hasEmployee koji će povezati prodavnicu i prodavca. Takođe kreirati svojstva hasSupervisor i hasSalesman, kao potomke tog svojstva.
- Prema konvenciji, svojstva se pišu kao lowerCamelCase i obično počinju sa is ili has.
- Definisati i inverzna svojstva i označiti ih u polju Inverse of.

Karakteristike svojstava

- Svojstvo može da bude:
  - funkcionalno,
  - inverzno funkcionalno,
  - tranzitivno,
  - simetrično,
  - asimetrično,
  - refleksivno,
  - irefleksivno.

Funkcionalno svojstvo

- Ako za svojstvo navedemo da je funkcionalno (Functional), onda individual može da se poveže samo sa jednim invidivudalom preko ovog svojstva. A -> B i A -> C => B = C.
- Na primer, za hasSupervisor navedemo da je funkcionalno svojstvo, i povežemo prodavnicu sa Markom tim svojstvom. Ako sada povežemo istu prodavnicu preko svojstva hasSupervisor sa Petrom, onda možemo zaključiti da su Marko i Petar ista osoba (individual).
- Ukoliko bismo naveli eksplicitno da su Marko i Petar različite osobe, naša ontologija bi postala nekonzistentna.
- Ako stavimo da je hasSupervisor funkcionalno, da li moramo staviti i da je isSupervisorAt funkcionalno?

Inverzno funkcionalno svojstvo

- Ako za svojstvo navedemo da je inverzno funkcionalno, to znači da je njegovo inverzno svojstvo funkcionalno, odnosno, za individual koji ima to svojstvo, postoji samo jedan individual koji je tim svojstvom povezan sa datim. A -> B i C -> B => A = C.
- Ako za isSupervisorAt navedemo da je inverzno funkcionalno, i individualima Marko i Petar dodelimo svojstvo isSupervisorAt prema istoj prodavnici, možemo zaključiti da su Marko i Petar ista osoba.

Tranzitivno svojstvo

- Ako je svojstvo tranzitivno i povezuje individuale A sa B i B sa C, onda možemo zaključiti da povezuje i A sa C. A -> B i B -> C => A -> C.
- Dodajemo property isSubordinateOf koji će povezati dva zaposlena. Ako je Pera podređen u odnosu na Marka, a Marko je podređen u odnosu na Milana, onda je Pera podređen u odnosu na Milana.

Simetrično svojstvo

- Ako je svojstvo simetrično i povezuje individual A sa B, onda možemo zaključiti da isto svojstvo povezuje i B sa A. A -> B => B -> A.
- Dodajemo svojstvo isColleagueOf za povezivanje dva zaposlena simetričnom vezom.

Asimetrično svojstvo

- Ako je svojstvo asimetrično i povezuje individual A sa B, onda to isto svojstvo ne može da poveže B sa A. A -> B => NOT(B -> A).

Refleksivno svojstvo

- Ako je svojstvo refleksivno, onda individual koji ima to svojstvo mora da ima vezu ka samom sebi preko te veze.
- Na primer ako uvedemo svojstvo Poznaje za osobu, onda bi to svojstvo moglo biti refleksivno, jer osoba poznaje sigurno sebe. To ne znači da ne može da poznaje i druge osobe.

Irefleksivno svojstvo

- Ako je svojstvo irefleksivno i povezuje individual A sa B, onda A i B ne mogu da budu isti individual.
- Ako želimo da navedemo da zaposleni ne može da bude kolega samom sebi, možemo navesti da je isColleagueOf irefleksivno.

Domain i Range

- Svojstva mogu da imaju domen i opseg. Svojstvo povezuje individuale iz domena sa individualima iz opsega.
- Šta bi bio domen, a šta opseg za svojstvo hasEmployee?
- Domen i opseg ne predstavljaju ograničenje samo po sebi, odnosno individuali i drugih klasa bi mogli da imaju svojstvo hasEmployee, ali bi u tom slučaju rezoner zaključio da su ti individuali prodavnice, odnosno pripadaju klasi Electronics_shop.

Zadatak

- Definisati domen i opseg za sva svojstva u našoj ontologiji.
- Rezonerom proveriti da li je ontologija konzistentna.

Restrikcije

- Moguće je definisati restrikcije za neka svojstva. Restrikcije mogu biti egzistencijalne, univerzalne i restrikcije kardinaliteta.
- Egzistencijalne restrikcije navode da je individual jedne klase preko određenog svojstva u vezi sa barem jednim individualom druge klase. Definiše se sa ključnom reči **some**.
- Prodavac mora da bude zaposlen u barem jednoj prodavnici.
- Univerzalne restrikcije navode da je individual jedne klase isključivo u vezi sa individualima druge navedene klase preko datog svojstva. Definiše se koristeći ključnu reč **only**.
- Prodavnice elektronike prodaje isključivo elektroniku (dodati svojstvo isSelling).

Restrikcije

- Restrikcije kardinaliteta se mogu definisati sa min, max i exactly.

Dodavanje restrikcija

- Restrikcije su zapravo anonimne klase, jer definišu skup individuala.
- Da bismo klasi dodelili restrikciju, navodimo je kao nadklasu.
- Restrikcije kreiramo u object restriction creator tab-u tako što biramo property, biramo filler (desnu stranu izraza) i tip restrikcije, ili u Class expression editoru, tako što ručno kucamo izraz.

Zadatak

- Dodati restrikcije:
  - Prodavnica elektronike mora da prodaje isključivo elektroniku i da zapošljava neke prodavce.
  - Prodavac mora da radi u nekoj prodavnici.
  - Dodati klasu Supervisor. To je klasa koja nasleđuje Salesman i ima tačno 0 nadređenih.

Primitivne i definisane klase

- Do sada kreirane klase i njihove restrikcije su samo **potrebni** uslovi za definisanje klase. Potrebni uslovi se definišu kao nadklase.
- Da bi neki individual pripadao klasi Electronics shop, potrebno je da prodaje samo elektroniku i da zapošljava neke prodavce.
- Klase koje imaju samo potrebne uslove su primitivne klase.
- Ovo ne znači da ako neki individual ima ta svojstva, da je nužno pripadnik klase Electronics shop.
- Da bismo ovo ostvarili moramo dodati potrebne i dovoljne uslove. Tako dobijamo definisane klase.
- Potrebni i dovoljni uslovi se definišu kao ekvivalentne klase.
- Odabrati Electronics shop klasu, pa Edit -> Convert to defined class.
- Sada, ako se pojavi individual koji ispunjava uslove navedene u Equivalent to za Electronics shop, može se zaključiti da pripada njegovoj klasi.
- Definisati i Supervisor kao Salesman koji ima tačno 0 podređenih, ali tako da bude definisana klasa.

Proširenje

- Dodati Part klasu, sa podklasama CPU, GPU, HDD, RAM, Screen, Keyboard.
- Dodati klasu ValuePartition, sa podklasom SizeValuePartition. SizeValuePartition ima podklase Small, Medium i Large. Svaka od podklasa SizeValuePartitiona je disjunktna sa svojim sibling klasama (nešto nije istovremeno i veliko i malo).
- ValuePartition je šablon za rad sa "enumeracijama". Funkcioniše tako što kreiramo object property, recimo hasSize čiji je opseg SizeValuePartition. Definisati je kao funkcionalno svojstvo.
- Navodimo da je SizeValuePartition ekvivalentan izrazu Small or Medium or Large.
- U čemu je razlika u odnosu na ranije? Šta smo dobili time što klasu definišemo kao uniju podklasa?
- Dodati i OperatingSystemValuePartition sa vrednostima IOS, Android, MacOS, Linux i Windows.

Zadatak

- Definisati Telefon kao mali uređaj koji ima CPU, ekran i tastaturu.
- Definisati Laptop kao uređaj srednje veličine koji ima CPU, GPU, HDD, RAM, ekran i tastaturu.
- Definisati Desktop kao veliki uređaj koji ima CPU, GPU, HDD i RAM.
- Definisati Tablet kao mali uređaj koji ima CPU i ekran.
- Kreirati klasu IPhone kao mali uređaj sa ekranom, CPU-om i tastaturom koji ima IOS operativni sistem, tako da je rezoner automatski klasifikuje kao telefon.

Saveti

- Ne postoji fiksno pravilo za imenovanje elemenata ontologije, ali treba ostati barem konzistentan unutar ontologije.
- Ne treba previše detaljno opisivati relacije. Domen, opseg i pre svega tipove relacija treba definisati samo ako je potrebno za zaključivanje.
- Kod klasa takođe definisati podklase do nivoa koji je potreban za ontologiju (zavisno od svrhe).
- Oprezno koristiti univerzalne restrikcije - princip otvorenog sveta.