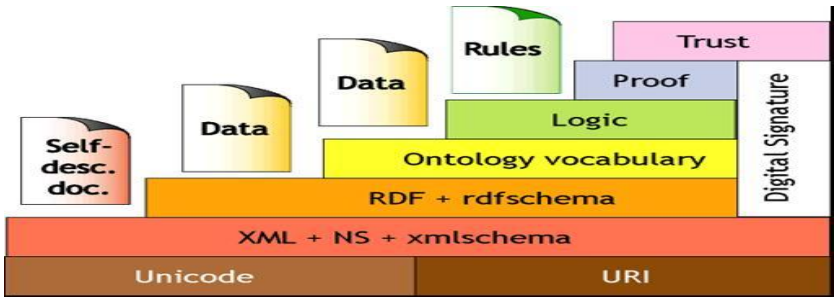
Cursul 4:

OWL: Limbaj de ontologii pentru Web



4.1. Introducere
- 4.1.1. Limite ale puterii expresive a RDF Schema
- 4.1.2. Caracteristici necesare pentru limbajele de ontologii
- 4.1.3. Compatibilitatea OWL cu RDF / RDFS
- 4.1.4. Trei dialecte OWL

4.2. Limbajul OWL
- 4.2.1. Sintaxa
- 4.2.3. Clasele OWL
- 4.2.4. Proprietăţile OWL
- 4.2.5. Definirea subclaselor owl
- 4.2.9. Instanţe
- 4.2.10. Tipuri de date
- 4.2.11. Informaţii privind versiunile
- 4.2.12. Dialectele OWL

4.3. Exemple
- 4.3.1. O ontologie referitoare la imprimante
- 4.3.2. O ontologie referitoare la fauna şi flora africană

4.4. Example de editoare pentru ontologii

4.5. Direcţii de dezvoltare
- 4.5.1. Module şi importuri
- 4.5.2. Valori implicite
- 4.5.3. Ipoteza lumilor închise
- 4.5.4. Ipoteza unicităţii numelor
- 4.5.5. Reguli de înlănţuirea proprietăţilor
- 4.5.6. Coduri ataşate

Rezumat

Bibliografie

4.1. Introducere

4.1.1. Limite ale puterii expresive a RDF Schema

Expresivitatea ***RDF*** şi ***RDF Schema***, este – în mod deliberat – foarte limitată:

- expresivitatea ***RDF*** este – practic – limitată la o serie de predicate binare;
- expresivitatea ***RDF Schema*** este – practic – limitată la o ierarhie de clase şi la o ierahie de proprietăţi (proprietăţile dispunând şi de definiţii pentru domeniile şi codomeniile lor).

=> ***RDF*** şi ***RDF Schema*** permit reprezentarea unor cunoştinţe privind ontologiile: principalele primitive de modelare ale ***RDF/RDFS*** se referă la organizarea vocabularelor în ierarhii de tipuri (*typed hierarchies*);

- relaţii între clase şi subclase;
- restricţii privind domeniile şi codomeniile;
- instanţe ale claselor.

Lipsesc însă următoarele elemente importante:

- orizontul (*scope*) local al proprietăţilor

în ***RDF Schema*** nu putem declara restricţii de domeniu de definiţie care să se aplice numai anumitor clase pentru că `rdfs:range` defineşte domeniul unei proprietăţi, fie aceasta `scrie`, la nivelul tuturor claselor. Exemple:
(1) nu putem declara faptul că dramaturgii scriu piese de teatru, poeţii: poezii, prozatorii: romane şi nuvele etc.;
(2) soluţia gasită (paleativ!!): ierarhia de proprietăţi. Ca să specificăm faptul că numai profesorii pot preda cursuri iar asistenţii nu, deşi sunt şi ei cadre didactice, a trebuit definită supraproprietatea "este AsociatCu";

- disjuncţia claselor

în ***RDF Schema*** nu putem declara decât relaţii între clase şi subclase. Exemplu:
putem declara clasele `animal` şi `plantă` ca subclase ale clasei `vieţuitoare` dar nu putem declara faptul că ele sunt disjuncte;

- combinaţiile booleene de clase

în ***RDF Schema*** nu putem defini o clasă ca reuniune, intersecţie sau complementară a altei clase. Exemplu:
nu putem defini clasa `vieţuitoare` ca reuniune a claselor disjuncte `animal` şi `plantă`;

- restricţiile de cardinalitate

în ***RDF Schema*** nu putem indica numărul de valori distincte pe care le poate lua o proprietate sau pe care trebuie să le ia o proprietate. Exemple, :
nu putem defini faptul că o persoană are exact doi părinţi,
nu putem defini faptul că un departament are cel puţin un angajat etc.;

- caracteristicile speciale ale proprietăţilor

în ***RDF Schema*** nu putem indica faptul că o proprietate – ca "`este paralel cu`" – este tranzitivă, sau că o proprietate – ca "este mama lui" – ia numai valori unice, sau că proprietăţi – ca "`predă`" şi "`este predat de`" – sunt inverse una alteia.

Din păcate,
- expresivitatea – pe de o parte –
- fundamentarea eficientă a raţionamentelor – pe de altă parte –

sunt trăsături complementare pentru orice limbaj pentru ontologii:
cu cât limbajul este mai bogat cu atât este mai ineficient din punctul de vedere al fundamentării raţionamentelor (ajungând până la limita necalculabilităţii).
Trebuie, evident, căutat şi găsit un echilibru între cele două atribute.

4.1.2. Caracteristici necesare pentru limbajele de ontologii

Cu toate acestea, Grupul **de Lucru privind Ontologiile Web din W3C** a identificat în Semantic Web un număr de situaţii de caz în care este nevoie de mult mai multe posibilităţi de reprezentare decât cele oferite de RDF şi RDF Schema. Mai multe grupuri de cercetare din Europa şi din Statele Unite au semnalat, de asemenea, necesitatea unui limbaj mult mai puternic pentru modelare a ontologiilor.

Aceasta a dus la o iniţiativă conjugată de definire a unui limbaj mult mai bogat, denumit **DAML+OIL** (numele provine din combinarea numelui limbajului **DAML+ONT = DARPA** Agent Markup Language, **propus de Statele Unite** şi a numelui limbajului **OIL = Ontology Inference Layer**, propus de **organizaţiile europene**).

La rândul său, **limbajul DAML+OIL** a fost luat de către Grupul de Lucru privind Ontologiile Web din W3C **ca punct de plecare pentru definirea limbajului OWL**, destinat a fi **standardul**, limbajul pentru ontologii general acceptat în Semantic Web.

În continuare, vom prezenta motivaţia şi caracteristicile limbajului OWL, precum şi elementele sale.

Limbajele pentru ontologii = limbaje care permit utilizatorilor să realizeze formalisme explicite pentru conceptele modelului asociat unui domeniu.

Caracteristici principale:
- sintaxă bine-definită;
- semantică formală;
- expresivitate convenabilă;
- bază eficientă pentru raţionamente.

Importanţa unei sintaxe bine-definite este evidentă şi bine-cunoscută în domeniul limbajelor de programare; ea este o condiţie necesară pentru prelucrarea automată a informaţiei. ***XML***, ***RDF*** şi ***RDF Schema*** dispun de o sintaxă bine-definită (dar nu foarte prietenoasă) iar ***DAML+OIL*** şi ***OWL*** se bazează şi ele pe acelaşi tip de sintaxă.[1]

[1] Faptul că sintaxa de tip ***XML*** a ***RDF*** nu este foarte accesibilă nu constituie un dezavantaj esenţial:
- există variante mai prietenoase cu utilizatorul (de exemplu, sintaxa ***OIL***);
- utilizatorii îşi pot realiza propriile ontologii cu instrumente speciale de realizare a ontologiilor în loc să le scrie direct în ***DAML+OIL*** sau în ***OWL***.

O semantică formală = descrie în termeni precişi înţelesul cunoştinţelor (*the meaning of knowledge*). Precizia invocată în această definiţie se referă la faptul că semantica respectivă nu se ocupă de intuiţii subiective şi nici nu permite (nici utilizatorilor nici calculatoarelor) interpretări diferite. Un domeniu în care importanţa unei semantici formale este bine stabilită este – de exemplu – cel al logicii matematice.

O semantică formală permite utilizatorilor – de exemplu – să efectueze raţionamente asupra unui corp de cunoştinţe. În cazul cunoştinţelor legate de ontologii (*ontological knowledge*) putem efectua raţionamente privind:

- apartenenţa la o clasă

(dacă x este o instanţă a clasei C
C este o subclasă a lui D
atunci putem infera faptul că x este o instanţă a clasei D);

- echivalenţa claselor

(dacă o clasă A este echivalentă cu o clasă B
B este echivalentă cu clasa C
atunci şi A este echivalentă cu C);

- consistenţa

(să presupunem că am declarat că:
x este o instanţă a clasei A
A este o subclasă atât a lui $B \cap C$
A este o subclasă atât a lui D
$B \cap D = \varnothing$.
Rezultă că avem o inconsistenţă întrucât clasa A ar trebui să fie vidă iar noi am declarat că ea are cel puţin instanţa x. Aceasta indică existenţa unei erori în ontologia noastră.);

- clasificarea

(dacă am declarat că anumite perechi proprietate–valoare reprezintă o condiţie suficientă pentru apartenenţa la clasa A
şi că un individ x satisface acele condiţii
atunci putem conchide că el este o instanţă a clasei A).

Semantica este o condiţie necesară pentru fundamentarea raţionamentelor (*reasoning support*). Inferenţe (*derivations*) de tipul celor de mai sus pot fi efectuate automat, nu manual. Fundamentarea raţionamentelor permite (între multe altele):

- verificarea consistenţei ontologiei şi bazei de cunoştinţe;
- verificarea existenţei unor relaţii nedorite între clase;
- clasificarea automată a instanţelor unei clase.

Semanticile formale şi fundamentările pentru raţionamente sunt furnizate de obicei prin:

- maparea unui limbaj pentru ontologii peste un formalism logic deja cunoscut;
- utilizarea unor instrumente automatizate de rationare deja existente pentru formalismele respective.

OWL este (în parte) mapat pe o logică descriptivă şi utilizează instrumente de raţionare existente (precum FaCT sau RACER).

Logica descriptivă este un subset al logicii predicatelor pentru care este posibilă asigurarea raţionamentelor în mod eficient.

4.1.3. Compatibilitatea OWL cu RDF / RDFS

La modul ideal, ***OWL*** ar trebui să fie o extensie a ***RDF Schema*** în sensul că ***OWL*** ar trebui să utilizeze semnificaţia claselor şi proprietăţilor din ***RDFS*** şi să adauge primitivele necesare creşterii expresivităţii. Această extensie a ***RDF Schema*** ar fi şi în spiritul arhitecturii ***Semantic Web*** din Figura 1.3.

Din păcate, o simplă extensie a ***RDF Schema*** nu ar asigura echilibrul necesar dintre expresivitate şi fundamentarea raţionamentelor: ***RDF Schema*** dispune de o serie de primitive de modelare (`rdfs:Class`, `rdfs:Property` etc.) extrem de puternice iar îmbogăţirea logicii sale cu astfel de primitive având o mare expresivitate ar conduce la apariţia unor proprietăţi computaţionale necontrolabile.

4.1.4. Trei dialecte OWL

Dată fiind „bogăţia“ setului de cerinţe pe care trebuie să le îndeplinească un limbaj de ontologii, precum şi relativa complementaritate a acestora, ***OWL*** a fost definit de ***Grupul de lucru privind Ontologiile Web*** al ***W3C*** sub forma a 3 sublimbaje (dialecte) distincte, fiecare dintre ele încercând să îndeplinească complet una dintre cerinţe:

OWL Full

Este numele limbajului complet. El:

- utilizează toate primitivele limbajelor ***OWL***;
- permite orice combinaţii arbitrare ale acestor primitive cu RDF şi RDF Schema, inclusiv schimbarea sensului primitivelor (***RDF*** sau ***OWL***) predefinite prin aplicarea primitivelor limbajului unele asupra altora. De exemplu, în ***OWL Full*** putem impune restricţii de cardinalitate asupra clasei tuturor claselor, limitând astfel numărul de clase ce pot fi descrise în orice ontologie.

Avantaje ale ***OWL Full***

Este complet compatibil cu ***RDF***, atât la nivelul sintaxei cât şi la nivelul semanticii:

- orice document corect (*legal*) ***RDF*** este corect şi ***OWL Full*** şi
- orice concluzie validă ***RDF*** / ***RDFS*** este validă şi ***OWL Full***.

Dezavantaje ale ***OWL Full***

Limbajul a devenit atât de puternic încât este nedecidabil şi orice speranţă privind fundamentarea completă a raţionamentelor este nerealistă.

OWL DL

Este numele unui sublimbaj al ***OWL Full***, ***DL*** însemnând ***Description Logics***. El:

- a fost definit în scopul asigurării eficienţei computaţionale,
- restricţionează modul de utilizare a constructorilor din OWL şi RDF; în principal este interzisă aplicarea constructorilor OWL unul asupra altuia, ceea ce face ca limbajul să corespundă unei logici descriptive bine definite.

Avantaje ale ***OWL DL***

Permite o fundamentare eficientă a raţionamentelor: logicile descriptive reprezintă un subset decidabil al ***First Order Logic*** şi permit, prin urmare, automatizarea raţionamentelor.

=> ierarhiile de clase se pot construi automat
inconsistenţele dintr-o ontologie realizată în ***OWL DL*** se pot determina automat.

Dezavantaje ale ***OWL DL***

Se pierde completa compatibilitate cu ***RDF***:

- orice document corect (*legal*) ***OWL DL*** este şi un document corect ***RDF*** ;
- un document ***RDF*** – în general – va trebui însă extins în anumite privinţe şi restrâns în altele pentru a putea deveni un document corect ***OWL DL.***

OWL Lite

Este numele unui sublimbaj al ***OWL DL***. El:

- limitează şi mai mult puterea expresivă a ***OWL***, conţinând numai o submulţime a constructorilor limbajului.

De exemplu: el exclude clasele enumerate, disjuncţia claselor şi cardinalitatea arbitrară.

Avantaje ale ***OWL Lite***

- Este uşor de învăţat (pentru utilizatori);
- Este uşor de implementat (pentru proiectanţi).

Dezavantaje ale ***OWL Lite***

Se pierde puterea expresivă.

Criterii de alegere a unui dialect

- Alegerea ***OWL Lite*** versus ***OWL DL*** depinde de măsura în care utilizatorul are nevoie de constructele mai expresive furnizate de ***OWL DL*** şi ***OWL Full***.
- Alegerea ***OWL DL*** versus ***OWL Full*** depinde de măsura în care utilizatorul are nevoie de facilităţile de metamodelare oferite de ***RDF Schema*** (de exemplu: definirea claselor de clase, ataşarea de proprietăţi claselor etc.). Utilizarea ***OWL Full*** în loc de ***OWL DL*** face ca fundamentarea raţionamentelor să fie mai puţin sigură întrucât o implementare completă a ***OWL Full*** nu este posibilă.

Compatibilitatea celor trei dialecte

- Orice ontologie corectă (*legal*) ***OWL Lite*** este şi o ontologie corectă ***OWL DL***.
- Orice ontologie corectă (*legal*) ***OWL DL*** este şi o ontologie corectă ***OWL Full***.
- Orice concluzie validă ***OWL Lite*** este şi o concluzie validă ***OWL DL***..
- Orice concluzie validă ***OWL DL*** este şi o concluzie validă ***OWL Full***.
- Toate variantele ***OWL*** utilizează pentru sintaxă ***RDF***.
- Instanţele sunt declarate ca în ***RDF***, utilizând descrieri ***RDF*** şi informaţii privind tipurile.
- Constructorii ***OWL*** (precum `owl:Class`, `owl:DatatypeProperty`, `owl:ObjectProperty` sunt specializări ale omologilor lor din ***RDF***.

În Figura 4.1 sunt prezentate relaţiile ierarhice dintre unele primitive de modelare din ***OWL*** şi ***RDF*** / ***RDFS***.

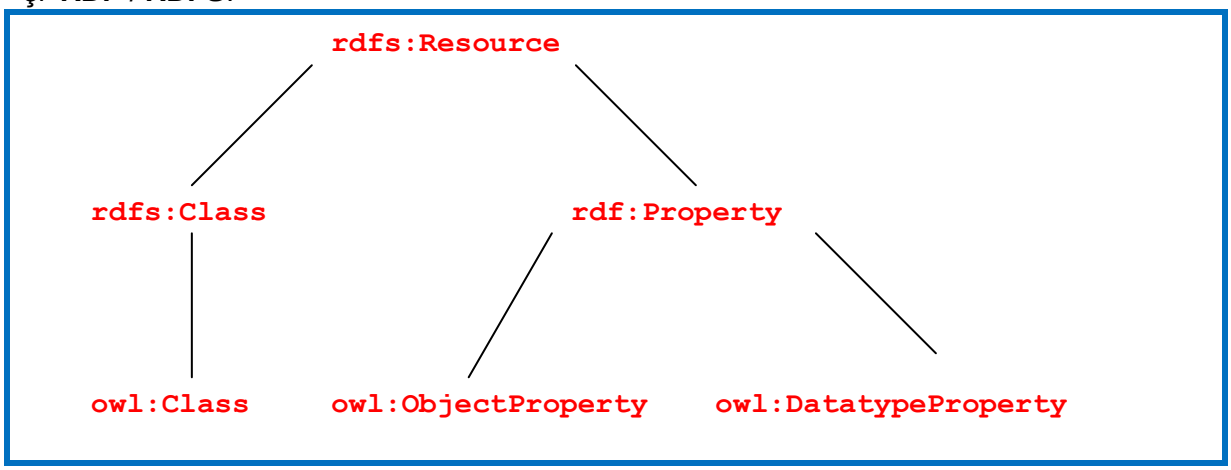


Figura 4.1. Relaţii ierarhice între ***OWL*** şi ***RDF*** / ***RDFS***

Una dintre principalele motivaţii pentru arhitectura stratificată a ***Semantic Web*** este aspiraţia către compatibilitatea de sus în jos şi – corespunzător – către reutilizarea software-ului de pe un nivel pe altul.
Totuşi, completa compatibilitate de sus în jos a ***OWL*** (i.e. orice procesor capabil să recunoască ***OWL*** este capabil să furnizeze intrepretări corecte pentru orice document ***RDF Schema***) este realizată numai pentru ***OWL Full*** şi aceasta cu preţul nerezolvabilităţii algoritmice.

4.2. Limbajul OWL

4.2.1. Sintaxa

OWL utilizează sintaxa ***RDF***/***XML***, care însă nu este foarte "lizibilă". Există şi alte forme sintactice pentru ***OWL***:

- o sintaxă de tip ***XML*** care nu urmăreşte convenţiile ***RDF*** şi este mai uşor de urmărit[2];
- o sintaxă abstractă, utilizată în documentul de specificare a limbajului[3], mult mai compactă mai uşor de citit decât sintaxa XML şi decăt sintaxa RDF/XML.
- o sintaxă grafică, bazată pe convenţiile ***UML***, larg utilizată şi foarte uşor de înţeles.

Document ***OWL*** (ontologie ***OWL***) = un document ***RDF***.

Antetul unui document *OWL*:(similar anetutului unui document *XML*) :

```
<!DOCTYPE  owl  [
    <!ENTITY  xsd  "http://www.w3.org/2001/XMLSchema#" >]  >
```

Elementul-rădăcină al unei ontologii ***OWL*** (Antet) =
= un element `rdf:RDF`, care include şi specificarea unui număr oarecare de domenii de definire (*namespaces*).

```
<rdf:RDF
  xmlns:owl  = http://www.w3.org/2002/07/owl#"
  xmlns:rdfs = http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf  = http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd  = http://www.w3.org/2001XMLSchema#">
```

[2] Definită la adresa <http://www.w3.org/TR/owl-xmlsyntax/>
[3] <http://www.w3.org/TR/owl-semantics>

O ontologie ***OWL*** poate începe cu o serie de declaraţii "cu rol administrativ". Ele sunt grupate într-un element `owl:Ontology`, care conţine

- comentarii;
- controlul versiunii;
- directive de includere a altor ontologii.

Exemplul 1

```
<owl:Ontology  rdf:about="">
  <rdfs:comment>Un exemplu de ontologie OWL</rdfs:comment>
  <owl:priorVersion
    rdf:resource="http://www.dept.ro/univ-ns-vrs1"/>
  <owl:imports
    rdf:resource="http://www.dept.ro/cadreDidactice"/>
  <rdfs:label>OntologieUniv</rdfs:label>
</owl:Ontology>
```

Observaţii 1

- Numai una dintre aceste aserţiuni are importanţă pentru semnificaţia logică a ontologiei: `owl:imports`, deoarece enumeră toate ontologiile al căror conţinut se presupune că face parte din ontologia curentă.
- Proprietatea **`owl:imports`** este **tranzitivă**: dacă ontologia `A` importă ontologia `B` iar ontologia `B` importă ontologia `C` atunci ontologia `A` importă şi ea ontologia `C`. De asemenea, ea este **circulară** (în **Protege**): ontologia A poate importa ontologia B care, la rândul ei, poate importa ontologia A.
- În timp ce domeniile de definire servesc numai pentru evitarea ambiguităţilor, ontologiile importate furnizează definiţii care pot fi efectiv utilizate.
- De obicei, există un element importat pentru fiecare domeniu de definire. Este posibil însă să se importe ontologii suplimentare, de exemplu: ontologii care să furnizeze definiţii fără a introduce nume noi.

4.2.2. Importarea ontologiei *Dublin Core*

Dublin Core ontology se bazeaza pe **Dublin Core Meta Data Terms**.

Dublin Core Meta Data Terms[4] = un set de elemente/termeni care pot fi folositi pt a descrie resurse (de exemplu, în Protégé: clasele, proprietatile şi indivizii dintr-o ontologie)

Dublin Core Meta Data Terms este descries la http://www.dublincore.org/documents/dcmi-terms/

Iata cateva example:

• **title** — Typically, a Title will be a name by which the resource is formally known.

• **creator** — Examples of a Creator include a person, an organisation, or a service. Typically, the name of a Creator should be used to indicate the entity.

• **subject** — Typically, a Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.

• **description** — Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.

• **contributor** — Examples of a Contributor include a person, an organisation, or a service. Typically, the name of a Contributor should be used to indicate the entity.

In order to annotate classes and other ontology entities with the above information and other Dublin Core Meta Data Terms the Dublin Core Meta Data ontology (DC Ontology) must be imported.
Because Dublin Core Meta Data is so frequently used, Protégé-OWL has an automated mechanism for importing it in the very procedure of creating a new project

[4] The *Dublin Core Meta Data Terms* were standardised/developed by *The Dublin Core Meta Data Initiative* (see *http://www.dublincore.org/*)

4.2.3. Clasele OWL

în OWL, clasele

✓ pot fi declarate:

- se specifică numai numele clasei (cu elementul **`owl:Class`**; cazul clasei radacină),
- se specifică numele clasei (cu elementul **owl:Class**) şi supraclasa (cu directiva `rdfs:` **`subClassOf;`**

✓ pot fi definite (căpătând şi un nume sau rămânâd anonime):

- prin enumararea tuturor elementelor (cu elementul **`owl:oneOf`**),
- prin operatii booleene asupra unor clase déjà create (cu elementul **`owl:unionOf`** `etc.`),
- prin impunerea unor restrictii asupra relatiilor dintre clase (cu elementele **`owl:restriction`** şi **`owl:onProperty`**).

4.2.3.1. Declararea clasei cu elementul `owl: Class`

Clasele sunt interpretate ca multimi de indivizi sau ca o reprezentare concretă a conceptelor din domeniul discursului. În *OWL DL*, instrumentul de verificare (*reasoner*) poate "calcula automat" taxonomia domeniului (ierarhia de clase şi supraclase).

Reprezentarea grafică[5]:
prin cercuri sau elipse, într-un mod asemănător diagramelor ***Venn***, în timp ce indivizii sunt reprezentaţi prin romburi.

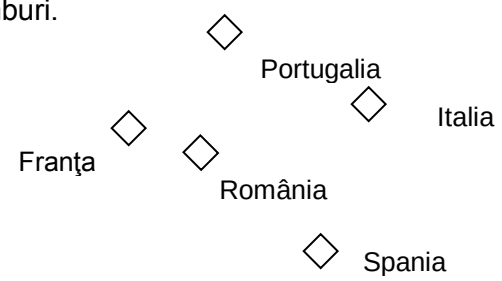


Figura 4.2 Reprezentarea indivizilor (instantelor)

Clasele sunt declarate cu ajutorul unui element **`owl:Class`**; **`owl:Class`** **este o subclasă a** **`rdfs:Class`** **(vezi Figura 4.1)**.

[5] Specific Protégé

Exemplul 2

Putem defini clasele "produs" şi "imprimanta astfel:

```
<owl:Class  rdf:ID="produs">
  <rdfs:comment>Produsele formeaza o clasa.</rdfs:comment>
</owl:Class>

<owl:Class  rdf:ID="imprimanta">
  <rdfs:comment>
    Imprimantele permit afisarea informatiilor şi formeaza o
    subclasa a produselor.  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#produs"/>
</owl:Class>
```

Clase owl predefinite

- **owl:Thing** = este cea mai generală clasă (*everything is a thing*).
- **owl:Nothing** = este clasa vidă.

Rezultă că orice clasă owl este o subclasă a clasei owl:Thing şi o supraclasă a clasei owl:Nothing.

4.2.3.2. Definirea clasei prin enumerare[6]

O clasă poate fi definită prin enumerarea tuturor elementelor sale.

Se utilizează:

- elementul **owl:oneOf** împreună cu atributul rdf:parseType="Collection"
- clasa predefinită **owl:Thing.**

[6] O enumerare este un element owl:oneOf utilizat pentru a defini o clasă prin listarea elementelor sale.

Exemplul 3

```
<owl:Class  rdf:ID="zileleSaptamanii">
  <owl:oneOf  rdf:parseType="Collection">
    <owl:Thing  rdf:about="#luni"/>
    <owl:Thing  rdf:about="#marti"/>
    <owl:Thing  rdf:about="#miercuri"/>
    <owl:Thing  rdf:about="#joi"/>
    <owl:Thing  rdf:about="#vineri"/>
    <owl:Thing  rdf:about="#sambata"/>
    <owl:Thing  rdf:about="#duminica"/>
  </owl:oneOf>
</owl:Class>
```

4.2.3.3. Definirea clasei prin operaţii booleene

O clasă ***OWL*** poate fi declarată şi definită şi cu ajutorul unor operaţii booleene asupra altor clase, déjà definite (ea poate fi privită ca o "expresie de clase").

Se utilizează

- pentru complementare, elementul **`owl:complementOf`**,
- pentru reuniune respectiv intersecţie, elementele **`owl:unionOf`**, respectiv **`owl:intersectionOf`**, impreună cu atributul **`rdf:parseType="Collection"`**.

Exemplul 8

Populaţia universitară constă din studenţi şi cadre didactice:

```
<owl:Class  rdf:ID="populatiaUniversitara">
  <owl:unionOf  rdf:parseType="Collection">
    <owl:Class  rdf:about="#studenti"/>
    <owl:Class  rdf:about="#cadreDidactice"/>
  </owl:unionOf>
</owl:Class>
```

Observaţie

Clasele care se reunesc nu trebuie să fie neaparat disjuncte (cum se întâmplă să fie în acest exemplu). Noua clasă este egală cu reuniunea (nu este o subclasă a acesteia) ⇒ am declarat o echivalenţă de clase.

4.2.3.3. Caracteristici ale claselor

Clase `owl` echivalente

Pot fi declarate cu ajutorul unui element **`owl:equivalentClass`**.

Exemplu

Putem declara echivalenţa claselor `personalDidactic` şi `cadreDidactice` astfel:

```
<owl:Class  rdf:ID="cadreDidactice">
  <owl:equivalentClass  rdf:resource="#personalDidactic"/>
</owl:Class>
```

Clase `owl` disjuncte

Pot fi declarate cu ajutorul elementului **`owl:disjointWith`**.

Exemplu

Putem declara clasa `Conferentiar` ca disjunctă de clasele `Profesor` şi `Asistent` folosind elemente `owl:disjointWith`. Aceste elemente pot fi

- incluse în definiţia anterioară;
- adăugate prin referirea la `ID`-ul lor, cu ajutorul unui atribut `rdf:about`.

Acest mecanism este moştenit din ***RDF***.

```
<owl:Class  rdf:about="#Conferentiar">
  <owl:disjointWith  rdf:resource="#Profesor"/>
  <owl:disjointWith  rdf:resource="#Asistent"/>
</owl:Class>
```

4.2.4. Proprietăţile OWL

În ***OWL*** există următoarele tipuri de proprietăţi:

- proprietăţi–relaţie[7] (*object property*), care corelează clasele
(de exemplu: `estePredatDe`, `monitorizează` etc.);

- proprietăţi–atribut[8] (*dataType property*), care corelează caracteristicile resurselor cu valorile lor
(de exemplu: `telefon`, `gradDidactic`, `dataNasterii`, `inaltime`, `greutate` etc., respectiv literali ***rdf***[9] sau tipuri de date ***XML Schema DataType***[10].)

- proprietăţi-de-adnotare[11] (*annotation property*), care adaugă informaţii (metadate = date despre date) claselor, indivizilor, respectiv proprietăţilor-obiect sau proprietăţilor-tip-de-date. Pot fi:
  - proprietăţile-de-adnotare-obiect,
  - proprietăţile-de-adnotare-tip-de-date.

Reprezentarea grafică[12]:
arce orientate de la clasa / individul domeniu de definiţie la clasa / individul sau la literalul domeniu de valori.

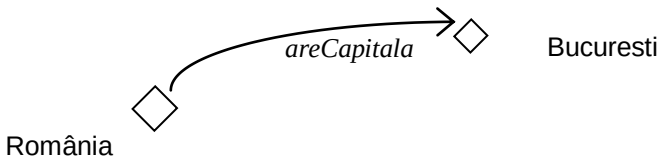


Figura 4.3 O proprietate-relaţie care corelează doi indivizi

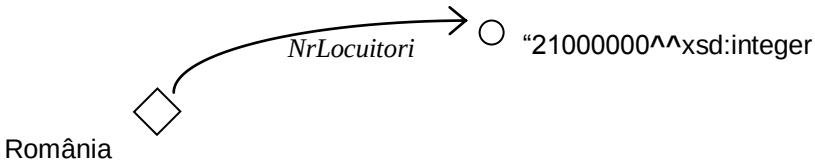


Figura 4.4 O proprietate-atribut care corelează un individ cu un literal de tip `xml:integer`

[7] analogul relaţiilor dintre entităţi, în modelul relaţional al BD
[8] analogul atributelor entităţilor, în modelul relaţional al BD
[9] o intorducere în ***RDF***: http://www.w3c.org/TR/rdf-primer/
[10] pentru informaţii privind tipurile de date ***XML Schema***: http://www.w3c.org/TR/xmlschema-2/
[11] proprietăţile-relaţie şi proprietăţile-atribut pot fi privite ca proprietăţi de adnotare.
[12] specific Protégé

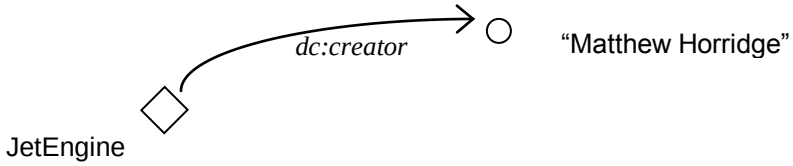


Figura 4.5 O proprietate-adnotare care corelează clasa 'JetEngine' cu literalul (stringul) 'Matthew Horridge'

Proprietăţile (în ***OWL***) se mai numesc şi roluri (în logicile descriptive), relaţii (în ***UML***), atribute (în ***GRAIL***).

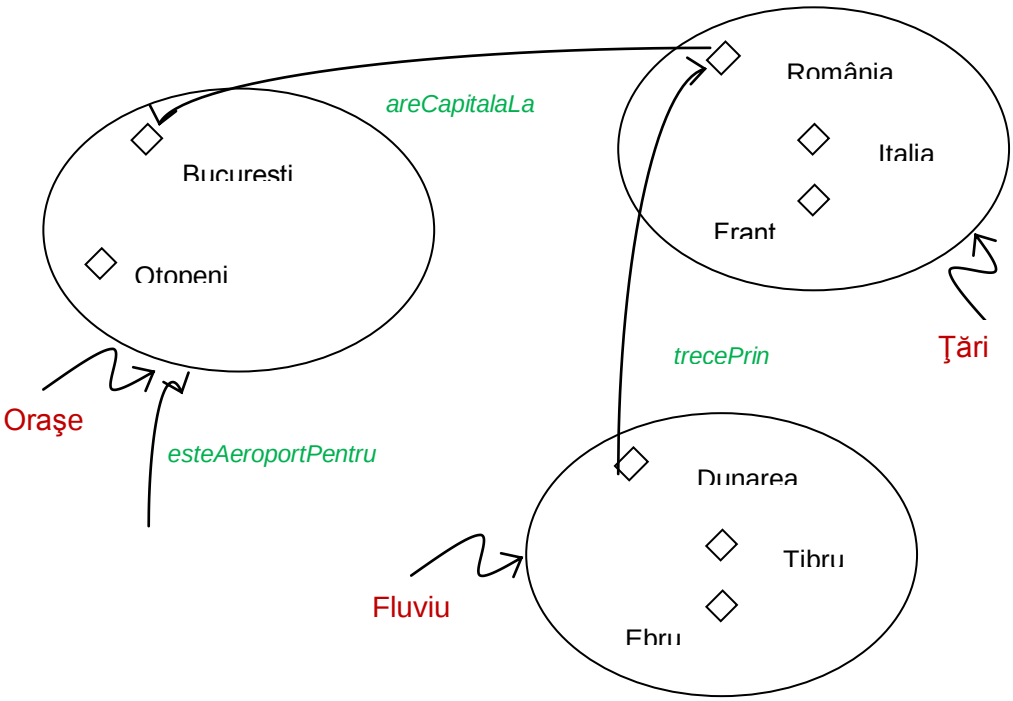


Figura 4.6. Reprezentarea indivizilor, claselor şi proprietăţilor[13]

[13] Specific Protégé

4.2.4.1. Elementul Property

Proprietăţile sunt declarate cu ajutorul unui element **owl:ObjectProperty**, respectiv **owl: DatatypeProperty**, ambele fiind **subclase** ale **rdf:Property** (vezi Figura 4.1).

Exemplul 3

`estePredatDe` este o proprietate–relaţie:

```
<owl:ObjectProperty  rdf:ID="estePredatDe">
  <rdfs:domain  rdf:resource="#curs"/>
  <rdfs:range  rdf:resource="#personalDidactic"/>
  <rdfs:subPropertyOf  rdf:resource="#esteAsociatCu"/>
</owl:ObjectProperty>
```

Exemplul 4

`înalţime` este o proprietate–atribut:

```
<owl:DatatypeProperty  rdf:ID="inaltime">
  <rdfs:range  rdf:resource="http://www.w3.org/2001/XMLSchema
   #nonNegativeInteger"/>
</owl:DatatypeProperty>
```

Observaţie 2

OWL nu dispune de tipuri de date predefinite sau de facilităţi speciale de definire a datelor. În schimb, permite utilizarea anumitor tipuri de date din ***XML Schema***, beneficiind astfel de arhitectura stratificată a ***Semantic Web***.

Observaţie 3

Tipurile de date definite de utilizator sunt de obicei colectate într-o schemă ***XML*** şi utilizate apoi într-o ontologie ***OWL***.

Observaţie 4

Se pot declara mai multe domenii şi codomenii pentru aceeaşi proprietate–obiect. În acest caz, se consideră intersecţia domeniilor (respectiv codomeniilor).

4.2.4.2. Declararea unor caracteristici ale proprietăţilor

Proprietăţi owl echivalente

Pot fi declarate cu ajutorul unui element **`owl:equivalentProperty`**.

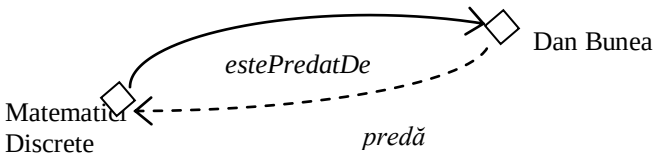
Exemplu

Proprietăţile `predă` şi `conferentiaza`.

```
<owl:ObjectProperty  rdf:ID="conferentiaza">
  <owl:equivalentProperty  rdf:resource="#preda"/>
</owl:ObjectProperty>
```

Proprietăţi owl inverse

Pot fi declarate cu ajutorul unui element **`owl:inverseOf`**.



Exemplu

Proprietăţile `estePredatDe` şi `predă`.

```
<owl:ObjectProperty  rdf:ID="preda">
  <rdfs:range  rdf:resource="#curs"/>
  <rdfs:domain  rdf:resource="#personalDidactic"/>
  <owl:inverseOf  rdf:resource="#estePredatDe"/>
</owl:ObjectProperty>
```

Proprietăţi speciale

Pot fi direct declarate următoarele caracteristici ale unei proprietăţi:

Caracteristică	***Declarată cu ajutorul unui element***	***Exemple***
tranzitivitatea	**owl:TransitiveProperty**	areNotaMaiMareDecat, esteMaiInaltDecat, precedePe etc.
simetria	**owl:SymmetricProperty**	areAceeasiNotaCaSi, esteFrateCu, etc.
unicitatea imaginii (proprietate de tip funcţie, adică o proprietate care, pentru fiecare obiect, ia cel mult o valoare):	**owl:FunctionalProperty**	areCapitala, areÎnaltimea, esteSupervizorul Direct etc.
injectivitatea (proprietate de tip funcţie injectivă, adică o proprietate care, pentru două obiecte distincte, ia valori distincte):	**owl:InverseFunctional Property**	esteCapitala, esteCNPal, etc.

Exemplul 7

```
<owl:ObjectProperty  rdf:ID="areAceeasiNotaCaSi">
  <rdf:type  rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type  rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain  rdf:resource="#student"/>
  <rdfs:range  rdf:resource="#student"/>
</owl:ObjectProperty>
```

Observaţie

În principiu, aceste caracteristici sunt specifice proprietăţilor de tip relaţie (*object property*).

4.2.5. Definirea subclaselor owl

În OWL, subclasele se pot declara şi defini prin:

- directiva `rdfs:subClassOf` (vezi şi declararea claselor),
- impunerea de restricţii asupra proprietăţilor-relaţie dintre clase,
- efectuarea de operaţii booleene asupra claselor (vezi şi definirea claselor),

Rezultatul poate fi o clasă cu nume sau una anonimă (poate fi folosită doar local).

4.2.5.1. Definirea subclaselor owl prin impunerea de restricţii asupra proprietăţilor-relaţie dintre clase

În ***RDF Schema***, cu directiva `rdfs:subClassOf` putem declara că o clasă `A` este o subclasă a clasei `B`, ceea ce înseamnă că orice instanţă a clasei `A` este şi o instanţă a clasei `B`. Aceasta înseamnă de fapt că toate instanţele clasei `A` au anumite proprietăţi suplimentare pe care nu le au chiar toate obiectele din clasa `B` ci doar cele din submulţimea $A \subset B$.

OWL preia această facilitate şi o dezvoltă în sensul că poate crea subclase A_1, A_2, ... ale clasei B nu doar declarându-le numele ci descriind clar caracteristicile lor suplimentare.

Metoda:
se definesc restricţii pe proprietăţi în care este implicată clasa B (ca domeniu sau codomeniu);
B nu apare neaparat cu numele său.

În plus, noile clase apărute pot fi chiar anonime
=> în ***RDFS***: subclasa este declarată prin numele său;
în ***OWL*** subclasa este definită prin restricţiile asupra proprietăţilor clasei de bază deci prin noile sale caracteristici.

Acest lucru se poate realiza în ***OWL*** cu

- ✓ un element `rdfs: subclassOf`
- ✓ un element `owl:Restriction` care conţine:
  - un element `owl:onProperty` (care specifică proprietatea asupra căreia acţionează restricţia);
  - una sau mai multe declaraţii de restricţie; acestea pot fi:
    - ✓ restricţii privind valorile pe care le poate lua proprietatea;
    - ✓ restricţii privind numărul de valori pe care le poate lua proprietatea.

Specificarea mulţimii tuturor valorilor unei proprietăţi

Se realizează cu ajutorul unui element **`owl:allValuesFrom`** şi corespunde **cuantificatorului logic universal**.

Exemplu

Trebuie specificat faptul că un curs de anul întâi poate fi predat **numai** de cadre didactice cu titlul de profesor.

=> se defineste clasa `cursAnulI` ca fiind formata exclusiv din acele cursuri predate de profesorii universitari.

=> se impune asupra proprietăţii `estePredatDe` o restricţie conform căreia această proprietate îşi ia valorile exclusiv din (subclasa) `profesor` (a clasei `cadreDidactice`)

```
<owl:Class  rdf:about="#cursAnulI">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#estePredatDe"/>
      <owl:allValuesFrom  rdf:resource="#Profesor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Specificarea unei mulţimi din care o proprietate îşi ia valorile fără a specifica valoarea

Se realizează cu ajutorul unui element **`owl:someValuesFrom`** şi corespunde **cuantificatorului logic existenţial.**

Exemplu

Trebuie specificat faptul că toate cadrele didactice trebuie să predea cel puţin un curs sau să susţină seminarii/laboratoare pentru cel puţin un curs (fără a spune ce curs anume):

=> se defineste clasa `cadreDidactice` ca fiind formată din acei membri ai personalului care sunt asociaţi (predau, susţin seminarii/laboratoare) cu cel puţin un curs

=> se impune asupra proprietăţii `esteAsociatCu` o restricţie conform căreia această proprietate poate lua o valoare oarecare din `curs`:

```
<owl:Class  rdf:about="#cadreDidactice">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#esteAsociatCu"/>
      <owl:someValuesFrom  rdf:resource="#curs"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Specificarea unei anumite valori pe care trebuie să o ia o proprietate

Se realizează cu ajutorul unui element **`owl:hasValue`**.

Exemplu

Trebuie specificat faptul că toate cursurile de matematică sunt predate numai de Radu Sava:

=> se defineste clasa `cursMatematica` ca fiind formată din acele cursuri care sunt predate de Radu Sava

=> se impune asupra proprietăţii `estePredatDe` o restricţie conform căreia această proprietate poate lua o singura valoare, şi anume Radu Sava (reprezentat prin ID-ul său)

```
<owl:Class  rdf:about="#cursMatematica">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#estePredatDe"/>
      <owl:hasValue  rdf:resource="#rs03"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Specificarea numărului minim/maxim de valori pe care le poate lua o proprietate

Se realizează cu ajutorul unui element **`owl:minCardinality`**, respectiv **`owl:maxCardinality`**.

Exemplul 5

Trebuie specificat faptul că un curs trebuie predat de cel puţin un cadru didactic:

```
<owl:Class  rdf:about="#curs">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty  rdf:resource="#estePredatDe"/>
   <owl:minCardinality  rdf:datatype="&xsd;nonNegativeInteger">
    1
   </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Observaţia 5

- A trebuit să specificăm faptul că literalul "1" trebuie interpretat ca un număr întreg nenegativ şi nu ca un string.
- A trebuit utilizată declaraţia `xsd` a domeniului de definire efectuată în antet pentru a referi documentul ***XML Schema***.

Exemplul 6

Trebuie specificat faptul că – din diverse motive! – departamentul de Informatică trebuie să aibă cel puţin 10 şi cel mult 30 de membri:

=> se defineste clasa `deptInfo` ca fiind acel departament care are intre 10 şi 30 membri

=> se impune asupra proprietăţii `areMembri` o restricţie conform căreia această proprietate poate lua valoarea minimă 10 şi valoarea maximă 30 (se definesc 2 restricţii):

```
<owl:Class  rdf:about="#deptInfo">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty  rdf:resource="#areMembri"/>
   <owl:minCardinality  rdf:datatype="&xsd;nonNegativeInteger">
    10
   </owl:minCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty  rdf:resource="#areMembri"/>
   <owl:maxCardinality  rdf:datatype="&xsd;nonNegativeInteger">
    30
   </owl:maxCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Observaţia 6

Putem specifica numărul exact de valori pe care le poate lua o proprietate prin:

- utilizarea ambelor elemente `minCardinality` şi `maxCardinality` şi specificarea aceleeaşi valori numerice în ambele;
- utilizarea elementului **`Cardinality`.**
- `minCardinality` şi `maxCardinality` nu se pot aplica proprietăţilor transitive (şi nici subproprietăţilor acestora, evident fiind şi ele transitive).

4.2.5.2. Definirea subclaselor owl prin operaţii booleene asupra unor claselor deja declarate/definite

O subclasă ***OWL*** poate fi declarată şi cu ajutorul unor operaţii booleene asupra altor clase, déjà definite (ea poate fi privită ca o "expresie de clase").

Se utilizează

- directiva **`rdfs:subClassOf`**,
- pentru complementare, elementul **`owl:complementOf`**,
- pentru reuniune respectiv intersecţie, elementele **`owl:unionOf`**, respectiv **`owl:intersectionOf`**, impreună cu atributul **`rdf:parseType="Collection"`**.

Exemplu

Cursurile şi cadrele didactice sunt clase disjuncte (i.e. orice curs este o instanţă a complementului clasei cadrelor didactice):

=> definim clasa complementară clasei cadrelor didactice drept clasa anonimă (e necesară şi utilizată doar aici) şi clasa cursuri ca subclasă a acesteia

```
<owl:Class  rdf:about="#cursuri">
 <rdfs:subClassOf>
  <owl:Class>
   <owl:complementOf  rdf:resource="#cadreDidactice"/>
  </owl:Class>
 </rdfs:subClassOf>
</owl:Class>
```

Observăm că această declaraţie putea fi realizată şi cu ajutorul unui element `owl:disjointWith`.

Exemplul 10

Vrem să definim clasa cadrelor didactice de la catedra de Informatică (Info):

=> definim clasa membrilor de personal care fac parte din catedra de informatică (prin restrictionarea proprietăţii `faceParteDin` la valoarea unică `catedraInfo`, drept clasa anonimă (e necesară şi tulizată doar aici) şi o intersectăm cu clasa cadrelor didactice din universitate

```
<owl:Class  rdf:ID="cadreDidacticeInfo">
 <owl:intersectionOf  rdf:parseType="Collection">
  <owl:Class  rdf:about="#cadreDidactice"/>
   <owl:Restriction>
    <owl:onProperty  rdf:resource="#faceParteDin"/>
    <owl:hasValue  rdf:resource="#catedraInfo"/>
   </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Observăm că am utilizat o clasă definită anonim, clasa obiectelor care aparţin catedrei de Informatică, pe care am intersectat-o cu clasa cadrelor didactice din universitate.

Observaţie 8

Combinaţiile booleene de clase pot fi imbricate arbitrar.
De exemplu, personalul administrativ este acel personal care nu este nici didactic nici tehnic.

```
<owl:Class  rdf:ID="personalAdministrativ">
 <owl:intersectionOf  rdf:parseType="Collection">
  <owl:Class  rdf:about="#personal"/>
   <owl:Class>
    <owl:complementOf>
     <owl:Class>
      <owl:unionOf  rdf:parseType="Collection">
       <owl:Class  rdf:about="#personalDidactic"/>
       <owl:Class  rdf:about="#personalTehnic"/>
      </owl:unionOf>
     </owl:Class>
    </owl:complementOf>
   </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

Observaţia 7

=> Elementul `owl:Restriction` defineşte o clasă anonimă, care nu are ID, nu este declarată prin `owl:Class` şi are un orizont local (poate fi folosită numai în locul în care apare restricţia). =>

=> Termenul clasă are 2 semnificaţii:

- clasa declarată printr-un element `owl:Class` şi având un ID;
- clasa locală şi anonimă, care este
  - o colecţie de obiecte ce satisfac anumite condiţii restrictive
  - o combinaţie de alte clase (numită de aceea, uneori, expresie de clase).

4.2.9. Instanţe

Declararea

Instanţele se declară ca în ***RDF***:

```
<rdf:Description  rdf:ID=”949352”>
  <rdf:type  rdf:resource=”#cadruDidactic”/>
</rdf:Description>
```

sau, echivalent:

```
<cadruDidactic  rdf:ID=”949352”>
```

Unicitatea numelor

Spre deosebire de sistemele tipice de gestiune a bazelor de date, ***OWL*** **NU** a adoptat convenţia privind unicitatea numelor: faptul că două instanţe dintr-o clasă au nume sau ID-uri diferite nu implică automat faptul că ele sunt diferite.

Exemplu:
dacă afirmăm că orice curs este predat de cel mult un cadru didactic

```
<owl:ObjectProperty  rdf:ID=”estePredatDe”>
  <rdf:type  rdf:resource=”&owl;FunctionalProperty”/>
</owl:ObjectProperty>
```

şi apoi declarăm că un anumit curs este predat de două cadre didactice

```
<curs  rdf:ID=”CIT1111”>
  <estePredatDe  rdf:resource=”#949318”/>
  <estePredatDe  rdf:resource=”#949352”/>
</curs>
```

nu se va semnala eroare deoarece sistemul va deduce că cele două resurse, `949318` şi `949352`, sunt aparent egale.

Declararea unicităţii

Se realizează cu ajutorul unui element:

- **`owl:differentFrom`**,

dacă este vorba de o singură pereche de instanţe;

- **`owl:allDifferent`** +
- **`owl:distinctMembers rdf:parseType="Collection"`**,

dacă este vorba de mai multe instanţe diferite 2 câte 2.

Exemple

```
<cadruDidactic  rdf:ID="949318">
  <owl:differentFrom  rdf:resource="#949352"/>
</cadruDidactic>

<owl:allDifferent>
  <owl:distinctMembers  rdf:parseType="Collection">
    <cadruDidactic  rdf:about="#949318"/>
    <cadruDidactic  rdf:about="#949352"/>
    <cadruDidactic  rdf:about="#949111"/>
  </owl:distinctMembers>
</owl:allDifferent>
```

Observaţie 9

Elementul `owl:distinctMembers` nu poate fi utilizat decât împreună cu elementul `owl:allDifferent`.

4.2.10. Tipuri de date

Deşi ***XML Schema*** posedă un mecanism de construire a tipurilor de date definite de utilizator (de exemplu, tipul `varstaAdult`: orice întreg mai mare decât 18 şi mai mic decât 150), aceste tipuri derivate nu pot fi folosite decât în ***OWL Full***. La fel, nici unele dintre tipurile predefinite din ***XML Schema*** nu pot fi folosite decât în ***OWL Full***.

Manualul de referinţă al ***OWL*** enumeră toate tipurile de date din ***XML Schema*** care pot fi folositein fiecare dialect (exemplu: *OWL DL*: tipuri foarte uzuale:

- întreg,
- boolean,
- string,
- timp şi dată.

4.2.11. Informaţii privind versiunile

OWL dispune de 4 instrucţiuni care dau informaţii privind versiunile. Nici una dintre instrucţiuni nu are o semnificaţie formală:

- **`owl:priorVersion`**:

este o instrucţiune care face parte din antet şi furnizează informaţii privind versiunile anterioare ale ontologiei curente; aceste informaţii nu au o semantică formală dar pot fi folosite atât de utilizatori cât şi de programele de calculator pentru administrarea ontologiei;

- `owl:versionInfo:`

conţine – în general – un string (de exemplu, secvenţa de cuvintele-cheie RCS/CVS) care dă informaţii despre versiunea curentă;

- **`owl:backwardCompatibleWith:`**

conţine o referinţă la o altă ontologie. Ontologia specificată prin acest element este astfel identificată ca o versiune anterioară a ontologiei curente iar ontologia curentă este astfel declarată ca fiind compatibilă "înapoi" cu aceasta. În particular, se indică faptul că toţi identificatorii din versiunea anterioară au aceeaşi interpretare şi în versiunea curentă. În acest fel, autorii sunt "înştiinţaţi" că îşi pot modifica în deplină siguranţă documentele astfel încât acestea să respecte noua versiune (prin simpla actualizare a declaraţiilor de domenii de definire şi a instrucţiunilor `owl:imports` astfel încât acestea să se refere la ***URL***-ul noii versiuni);

- **`owl:incompatibleWith:`**

conţine tot o referinţă la o altă ontologie. Ontologia specificată prin acest element este astfel identificată tot ca o versiune anterioară a ontologiei curente dar ontologia curentă este astfel declarată ca fiind incompatibilă cu aceasta. În acest fel, autorii sunt "înştiinţaţi" că documentele lor NU se pot "upgrada" la noua versiune a ontologiei fără a verifica mai întâi dacă nu sunt necesare unele modificări.

4.2.12. Dialectele OWL

Vom indica – pentru fiecare element al limbajului ***OWL*** – sublimbajul (***OWL Full***, ***OWL*** sau ***OWL Lite***) în care poate fi folosit.

OWL Full

În ***OWL Full*** pot fi folosiţi toţi constructorii limbajului, în orice combinaţie, atât timp cât rezultatul este corect din punct de vedere al ***RDF***.

OWL DL

Pentru a exploata fundamentul formal şi solvabilitatea computaţională a *Description Logics,* în orice ontologie ***OWL DL*** trebuie respectate următoarele restricţii:

- partiţionarea vocabularului: orice resursă nu poate fi decât
  - ✓ o clasă /
  - ✓ un tip de date /
  - ✓ o proprietate-tip de date /
  - ✓ o proprietate-obiect /
  - ✓ un individ /
  - ✓ o valoare /
  - ✓ parte din vocabularul predefinit

şi numai una dintre acestea. Adică:
o clasă nu poate fi în acelaşi timp şi un individ;
o proprietate nu poate lua în acelaşi timp valori dintr-o clasă şi valori dintr-un tip de date (pentru că ar trebui să fie simultan o proprietate-obiect şi o proprietate-tip de date);

- declararea explicită:

resursele nu trebuie numai partiţionate (conform restricţiei de mai sus)
dar această partiţionare trebuie declarată explicit.
De exemplu, dacă o ontologie conţine următoarele:

```
<owl:Class  rdf:ID="C1">
  <rdfs:subClassOf  rdf:about="#C2"/>
</owl:Class>
```

atunci aceasta înseamnă că `C2` este o clasă (conform specificaţiei de domeniu de definiţie din `rdfs:subClassOf`).
Cu toate acestea, o ontologie ***OWL DL*** trebuie să declare explicit această informaţie:
`<owl:Class  rdf:ID="C2"> ->`

- separarea proprietăţilor:

conform primei restricţii, mulţimea proprietăţilor-relaţie şi mulţimea proprietăţilor-atribut sunt disjuncte;
în consecinţă, declararea acestora se face diferit şi următoarele definiţii nu ar putea fi niciodată folosite pentru o proprietate-atribut:

`owl:inverseOf,`
`owl:FunctionalProperty,`
`owl:InverseFunctionalProperty,`
`owl:SymmetricProperty.`

- restricţiile de cardinalitate nu sunt tranzitive:

asupra proprietăţilor tranzitive (ca şi asupra subproprietăţilor lor, care sunt, în consecinţă, şi ele tranzitive) nu pot fi aplicate restricţii de cardinalitate;

- restricţionarea numărului de clase anonime:

este permisă apariţia claselor anonime numai ca:

- ✓ domeniu de definiţie sau codomeniu
  - fie pentru elementul **owl:disjointWith**
  - fie pentru elementul **owl:equivalentWith**,
- ✓ codomeniu (dar nu domeniu de definiţie) pentru **`rdfs:subClassOf.`**

OWL Lite

O ontologie ***OWL Lite*** trebuie să fie o ontologie ***OWL DL*** şi, de asemenea, trebuie să respecte următoarele restricţii:

- NU sunt permişi următorii constructori:

`owl:oneOf,`
`owl:disjointWith`
`owl:unionOf`
`owl:complementOf`
`owl:hasValue`

- restricţiile de cardinalitate

(cardinalitatea maximă, minimă sau exactă) pot fi aplicate numai asupra valorilor 0 sau 1 şi nu asupra oricăror întregi nenegativi (ca până acum);

- instrucţiunile **`owl:equivalentWith`**

nu se mai pot referi la clase anonime ci numai la identificatori de clase.

4.3. Exemple

4.3.1. O ontologie referitoare la imprimante

Figura 4.5 descrie principalele clase şi subclase, precum şi relaţiile dintre ele.

Această ontologie demonstrează faptul că nodurile-fraţi din arborele asociat nu trebuie să fie neapărat disjuncte. De exemplu, o imprimantă personală poate fi o imprimantă HP sau o imprimantă LaserJet, deşi aceste trei clase sunt toate subclase ale clasei tuturor imprimantelor.

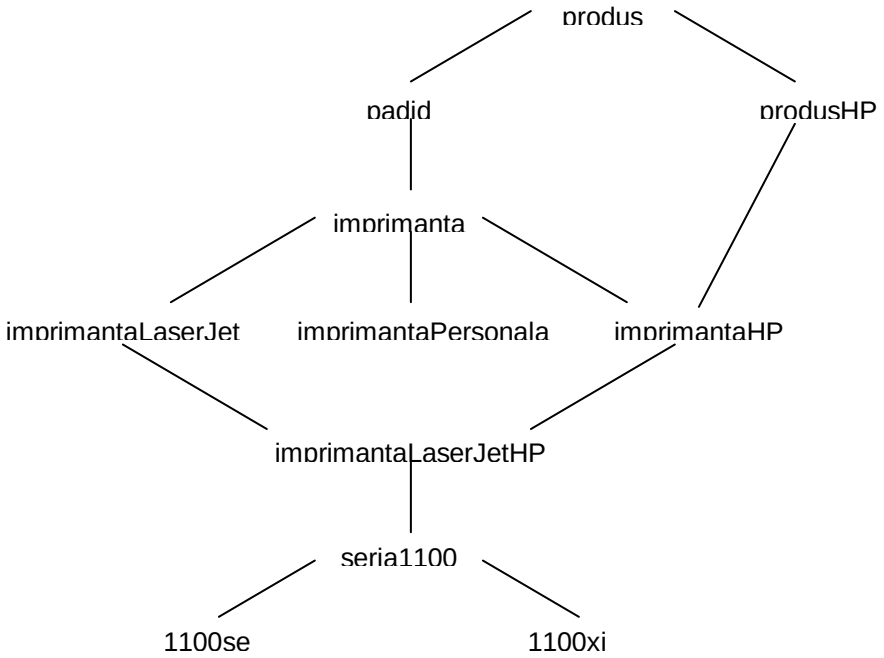


Figura 4.5. Clasele şi subclasele ontologiei referitoare la imprimante

```
<!DOCTYPE  owl  [
    <!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#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  xmlns="http://www.cs.vu.nl/~frankh/spool/printer.owl#">

<owl:Ontology  rdf:about="">
  <owl:versionInfo>
    Frank Hermelin' example version 1.2, 17 oct 2002
  </owl:versionInfo>
</owl:Ontology>

<owl:Class  rdf:ID="produs">
  <rdfs:comment>Produsele formeaza o clasa.</rdfs:comment>
</owl:Class>

<owl:Class  rdf:ID="padid">
  <rdfs:comment>
    Imprimante şi dispozitive digitale de transmitere a imaginilor
    care formeaza o subclasa a produselor.
  </rdfs:comment>
  <rdfs:label>Dispozitiv</rdfs:label>
  </rdfs:subClassOf rdf:resource="#produs"/>
</owl:Class>

<owl:Class  rdf:ID="produsHP">
  <rdfs:comment>
    Produsele HP sunt acele produse care sunt fabricate de compania
    Hewlett Packard.
  </rdfs:comment>
  <owl:intersectionOf  rdf:parseType="Collection">
    <owl:Class  rdf:about="#produs"/>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#fabricat_de"/>
      <owl:hasValue  rdf: dataType="&xsd;string">
        Hewlett Packard
      </owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
```

```
</owl:Class>

<owl:Class  rdf:ID="imprimanta">
  <rdfs:comment>
    Imprimantele sunt dispozitive de listare şi transmitere a
    imaginilor.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#padid"/>
</owl:Class>

<owl:Class  rdf:ID="imprimantaPersonala">
  <rdfs:comment>
    Imprimante de uz personal care formeaza o subclasa a
    imprimantelor.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#imprimanta"/>
</owl:Class>

<owl:Class  rdf:ID="imprimantaHP">
  <rdfs:comment>
    Imprimantele HP sunt imprimante şi produse HP.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#imprimanta"/>
  <rdfs:subClassOf  rdf:resource="#produsHP"/>
</owl:Class>

<owl:Class  rdf:ID="imprimantaLaserJet">
  <rdfs:comment>
    Imprimantele LaserJet sunt acele imprimante care folosesc
    tehnologia Laser Jet.
  </rdfs:comment>
  <owl:intersectionOf  rdf:parseType="Collection">
    <owl:Class  rdf:about="#imprimanta"/>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#tehnologieDePrintare"/>
      <owl:hasValue  rdf: dataType="&xsd;string">
        laser jet
      </owl:hasValue>
    </owl:Restriction>
```

```
  </owl:intersectionOf>
</owl:Class>

<owl:Class  rdf:ID="imprimantaLaserJetHP">
  <rdfs:comment>
    Imprimantele HP LaserJet sunt imprimante LaserJet şi produse
    HP.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#imprimantaLaserJet"/>
  <rdfs:subClassOf  rdf:resource="#imprimantaHP"/>
</owl:Class>

<owl:Class  rdf:ID="seria1100">
  <rdfs:comment>
    Imprimantele din seria 1100 sunt imprimante LaserJet HP cu
    viteza de printare 8ppm şi rezolutia de printare de 600dpi.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#imprimantaLaserJetHP"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#vitezaDePrintare"/>
      <owl:hasValue  rdf: dataType="&xsd;string">
        8ppm
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#rezolutiaDePrintare"/>
      <owl:hasValue  rdf: dataType="&xsd;string">
        600dpi
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="1100se">
  <rdfs:comment>
    Imprimantele 1100se fac parte din seria 1100 şi costa 450$.
```

```
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#seria1100"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#pret"/>
      <owl:hasValue  rdf: dataType="&xsd;integer">
        450
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="1100xi">
  <rdfs:comment>
    Imprimantele 1100xi fac parte din seria 1100 şi costa 350$.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#seria1100"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#pret"/>
      <owl:hasValue  rdf:dataType="&xsd;integer">
        350
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DataTypeProperty  rdf:ID="fabricat_de">
  <rdfs:domain  rdf:resource="#produs"/>
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DataTypeProperty>

<owl:DataTypeProperty  rdf:ID="pret">
  <rdfs:domain  rdf:resource="#produs"/>
  <rdfs:range  rdf:resource="&xsd;nonNegativeInteger"/>
</owl:DataTypeProperty>

<owl:DataTypeProperty  rdf:ID="tehnologieDePrintare">
  <rdfs:domain  rdf:resource="#imprimanta"/>
```

```
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DataTypeProperty>

<owl:DataTypeProperty  rdf:ID="rezolutiaDePrintare">
  <rdfs:domain  rdf:resource="#imprimanta"/>
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DataTypeProperty>

<owl:DataTypeProperty  rdf:ID="vitezaDePrintare">
  <rdfs:domain  rdf:resource="#imprimanta"/>
  <rdfs:range  rdf:resource="&xsd;string"/>
</owl:DataTypeProperty>

</rdf:RDF>
```

4.3.2. O ontologie referitoare la fauna şi flora africană

Figura 4.3 descrie principalele clase şi subclase, precum şi relaţiile dintre ele. Să observăm că informaţiile despre subclase reprezintă numai o parte din informaţia inclusă în ontologie. Întregul graf este mult mai mare. Figra 4.4 conţine reprezentarea grafică a instrucţiunii conform căreia ramurile sunt părţi ale copacilor; codul este însoţit de comentarii.

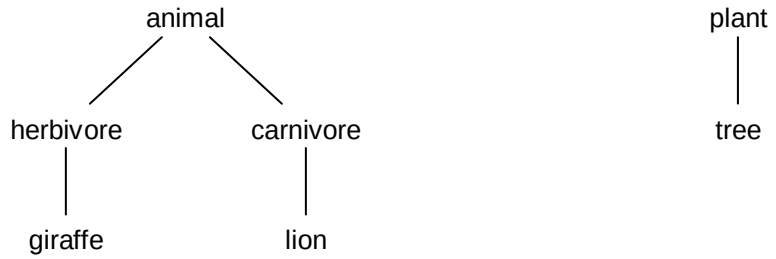


Figura 4.3. Clasele şi subclasele ontologiei referitoare la fauna şi flora africană

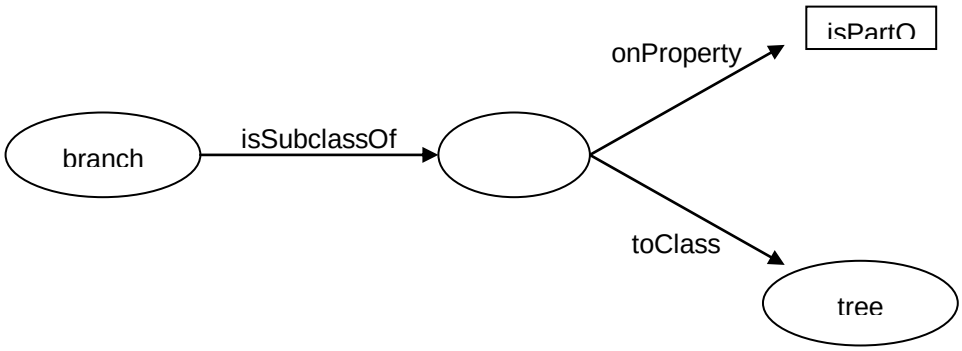


Figura 4.4.Ramurile sunt părţi ale copacilor

```
<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#">
<owl:Ontology  rdf:about="xml:base"/>

<owl:Class  rdf:ID="animal">
  <rdfs:comment>Animals constitute a class.</rdfs:comment>
</owl:Class>

<owl:Class  rdf:ID="plant">
  <rdfs:comment>
    Plants constitute a class disjoint of that of animals.
  </rdfs:comment>
  <owl:disjointWith  rdf:resource="#animal"/>
</owl:Class>

<owl:Class  rdf:ID="tree">
  <rdfs:comment>Trees a type of plant.</rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#plant"/>
</owl:Class>

<owl:Class  rdf:ID="branch">
  <rdfs:comment>Branches are parts of trees.</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#is_part_of"/>
      <owl:allValuesFrom  rdf:resource="#tree"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="leaf">
  <rdfs:comment>Leafs are parts of branches.</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#is_part_of"/>
      <owl:allValuesFrom  rdf:resource="#branch"/>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="herbivore">
  <rdfs:comment>
    Herbivores are those animals that eat only plants or parts of
    plants.
  </rdfs:comment>
  <owl:intersectionOf  rdf:parseType="Collection">
    <owl:Class  rdf:about="#animal"/>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eats"/>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf  rdf: parseType="Collection>
            <owl:Class  rdf:about="#plant"/>
              <owl:Restriction>
                <owl:onProperty  rdf:resource="#is_part_of"/>
                <owl:allValuesFrom  rdf:resource="#plant"/>
              </owl:Restriction>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>

<owl:Class  rdf:ID="carnivore">
  <rdfs:comment>
    Carnivores are those animals that eat only animals.
  </rdfs:comment>
  <owl:intersectionOf  rdf:parseType="Collection">
    <owl:Class  rdf:about="#animal"/>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eats"/>
      <owl:someValuesFrom  rdf:resource="#animal"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

```
<owl:Class  rdf:ID="giraffe">
  <rdfs:comment>
    Giraffes are herbivores and they eat only leaves.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#herbivore"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eats"/>
      <owl:allValuesFrom  rdf:resource="#leaf"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="lion">
  <rdfs:comment>
    Lions are animals that eat only herbivores.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#carnivore"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eats"/>
      <owl:allValuesFrom  rdf:resource="#herbivore"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class  rdf:ID="tasty_plant">
  <rdfs:comment>
    Tasty plants are plants that are eaten both by herbivores and
    carnivores.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="#plant"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eaten_by"/>
      <owl:someValuesFrom>
        <owl:Class  rdf:about="#herbivore"/>
      </owl:someValuesFrom>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty  rdf:resource="#eaten_by"/>
      <owl:someValuesFrom>
        <owl:Class  rdf:about="#carnivore"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:TransitiveProperty  rdf:ID="is_part_of"/>

<owl:ObjectProperty  rdf:ID="eats"/>
  <rdfs:domain  rdf:resource="#animal"/>
</owl:ObjectProperty>

<owl:ObjectProperty  rdf:ID="eaten_by"/>
  <owl:inverseOf  rdf:resource="#eats"/>
</owl:ObjectProperty>

</rdf:RDF>
```

4.4. Example de editoare pentru ontologii

NAME (URL)	DESCRIPTION
Ontology Editors Links	**Descriptions**
Adaptiva	A user-centred ontology building environment, based on using multiple strategies to construct an ontology, minimising user input by using adaptive information extraction
Altova SemanticWorks	Visual RDF and OWL editor that auto-generates RDF/XML or nTriples based on visual ontology design
Compendium	Compendium is a semantic, visual hypertext tool for supporting collaborative domain modelling and real time meeting capture
ConcepTool	A system to model, analyse, verify, validate, share, combine, and reuse domain knowledge bases and ontologies, reasoning about their implication.
GrOWL	Open source graphical ontology browser and editor
IODT	IBMâs toolkit for ontology-driven development
KAON	Open source ontology management infrastructure
KAON2	KAON2 is an an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies. it is capable of manipulating OWL-DL ontologies; queries can be formulated using SPARQL
LinKFactory	Language & Computingâs LinKFactory is an ontology management tool, it provides an effective and user-friendly way to create, maintain and extend extensive multilingual terminology systems and ontologies (English, Spanish, French, etc.). It is designed to build, manage and maintain large, complex, language independent ontologies.
Metatomix	Commercial semantic toolkits and editors
MindRaider	Open source semantic Web outline editor
MnM	MnM is an annotation tool which provides both automated and semi-automated support for annotating web pages with semantic contents. MnM integrates a web browser with an ontology editor and provides open APIs to link to ontology servers and for integrating information extraction tool
Model Futures OWL Editor	Simple OWL tools, featuring UML (XMI), ErWin, thesaurus and imports

OntoEdit/OntoStudio	Engineering environment for ontologies
OWLViz	OWLViz is visual editor for OWL and is available as a Protege plug-in
pOWL	Semantic Web development platform
Protege	Open source visual ontology editor written in Java with many plug-in tools
RDF InferEd	Intellidimensionâs RDF InferEd is an authoring environment with the ability to navigate and edit RDF documents
Redland	The Redland RDF Application Framework is a set of free software libraries that provide support for RDF. It provides parser for RDF/XML, Turtle, N-triples, Atom, RSS; has a SPARQL and GRDDL implementation, and has language interfaces to C#, Python, Obj-C, Perl, PHP, Ruby, Java and Tcl
ReTAX+	ReTAX is an aide to help a taxonomist create a consistent taxonomy and in particular provides suggestions as to where a new entity could be placed in the taxonomy whilst retaining the integrity of the revised taxonomy (c.f., problems in ontology modelling).
Refiner++	REFINER++ is a system which allows domain experts to create and maintain their own Knowledge Bases, and to receive suggestions as to how to remove inconsistencies, if they exist.
SemanticWorks	A visual RDF/OWL Editor from Altova
SWOOP	A lightweight ontology editor
TopBraid Composer	Top Quandrantâs TopBraid Composer is a complete standards-based platform for developing, testing and maintaining Semantic Web applications
WebOnto	WebOnto supports the browsing, creation and editing of ontologies through coarse grained and fine grained visualizations and direct manipulation.
WSMO Studio	A semantic Web service editor compliant with WSMO as a set of Eclipse plug-ins
WSMT Toolkit	The Web Service Modeling Toolkit (WSMT) is a collection of tools for use with the Web Service Modeling Ontology (WSMO), the Web Service Modeling Language (WSML) and the Web Service Execution Environment (WSMX)

Ontology Warehouse

- Protégé (Java-based, downloadable, open source, many sample ontologies, from Stanford University)
- Semantic Turkey (Firefox extension - also based on Java - for managing ontologies and acquiring new knowledge from the Web; developed at University of Rome, Tor Vergata)
- Swoop (Java-based, downloadable, open source, OWL Ontology browser and editor from the University of Maryland)
- OBO-Edit (Java-based, downloadable, open source, developed by the Gene Ontology Consortium for editing biological ontologies)
- Synaptica (Ontology, taxonomy and thesaurus management software available from Factiva from Dow Jones. Web based, supports OWL and SKOS.)
- Knoodl (Free web application/service that is an ontology editor, wiki, and ontology registry. Supports creation of communities where members can collaboratively import, create, discuss, document and publish ontologies. Supports OWL, RDF, RDFS, and SPARQL queries. Available since early Nov 2006 from Revelytix, Inc..)
- TopBraid Composer (Eclipse-based, downloadable, full support for RDFS and OWL, built-in inference engine, SWRL editor and SPARQL queries, visualization, import of XML and UML, from TopQuadrant)
- Java Ontology Editor (JOE) (1998)
- Model Futures OWL Editor (Free) Extensive import and export capabilities (inc. UML, Thesaurus Descriptor, MS Word, CA ERwin Data Modeler, CSV, etc.)
- Ontolingua (Web service offered by Stanford University)
- Chimaera (Other web service by Stanford)
- OntoEdit (Web service offered by the University of Karlsruhe)
- myWeb (Java-based, mySQL connection, bundled with applet that allows online browsing of ontologies (including OBO))
- OilEd (Java-based, downloadable, GPL, many users, from the University of Manchester, no longer maintained)
- ScholOnto (net-centric representations of research)
- WebODE (Web service offered by the Technical University of Madrid)
- KAON (single user and server based solutions possible, open source, from IPE Karlsruhe)
- KMgen (Ontology editor for the KM language. KM: The Knowledge Machine)
- LinKFactory Java-Based Commercial Ontology editor, collaborative multi-user support, highly scalable (several millions of knowledge objects), full support for OWL, minimal processing time, allows a multilingual approach, from Language and Computing
- DOME (DERI Ontology Management Environment), Eclipse-based, open-source
- CMapTools (Java based ontology editor from University of Florida. Supports numerous formats)
- Transinsight (The editor is especially designed for text mining ontologies)
- Be Informed Suite (Commercial tool for building large ontology based applications. Includes visual editors, inference engines, export to standard formats)

4.5. Direcţii de dezvoltare

Evident, ***OWL*** nu este ultima variantă de limbaj pentru ontologii pentru ***Semantic Web***. O serie întreagă de trăsături suplimentare au fost identificate şi enumerate în *OWL Requirements Document*; altele sunt încă în curs de a fi analizate.

4.5.1. Module şi importuri

Importarea ontologiilor realizate de terţi va fi o regulă în ***Semantic Web***. Totuşi, facilităţile de importare ale ***OWL*** sunt destul de rudimentare: se poate importa numai o întreagă ontologie, specificată prin locaţia ei, chiar dacă ar fi suficientă importarea numai a unei mici părţi din aceasta.

Modulele-constructori din limbajele de programare se bazează pe noţiunea de ascundere a informaţiei (*information hiding*): modulul declară că va oferi "mediului înconjurător" o anumită funcţionalitate (specificată prin clauza de export a modulului) dar modulul importator nu trebuie să "ştie" cum se realizează această funcţionalitate. Găsirea unui analog al *information hiding* pentru ontologii şi a unui mod de utilizare a acestuia ca bază pentru un constructor eficient de importare este încă o problemă deschisă.

4.5.2. Valori implicite

Numeroase reprezentări concrete de cunoştinţe permit supradefinirea (*overriding*) valorilor moştenite cu ajutorul unor clase mai bine precizate din ierarhie, tratând valorile moştenite ca valori predefinite (*defaults*). Deşi această tehnică este larg utilizată în practică, nu s-a realizat încă un consens privind corecta formalizare a comportamentului nemonoton al valorilor implicite.

4.5.3. Ipoteza lumilor închise

În prezent, semantica ***OWL*** adoptă modelul logic standard al ipotezei lumilor deschise: nu se poate afirma că o aserţiune este adevărată numai pentru că nu a putut fi demonstrată. Evident, la nivelul imensului şi numai parţial cunoscutului ***World Wide Web***, o astfel de presupunere este corectă. Totuşi, abordarea complementară (ipoteza lumilor închise: o

aserţiune este adevarată atunci când negaţia ei nu poate fi demonstrată) este şi ea utilă în cazul anumitor aplicaţii. Ipoteza lumilor închise este strâns legată de noţiunea de valori implicite şi conduce către aceeaşi comportament nemonoton, ceea ce constituie un motiv puternic pentru neincluderea ei în ***OWL***.

4.5.4. Ipoteza unicităţii numelor

Aplicaţiile tipice de baze de date presupun că indivizii cu nume diferite sunt efectiv distincţi. Acolo unde nu este cazul, ***OWL*** urmează obişnuita paradigmă logică. Dacă doi indivizi (clase sau proprietăţi) au nume diferite, atunci putem încă infera că ei reprezintă unul şi acelaşi individ. Ca şi în cazul ipotezei lumilor deschise, ipoteza neunicităţii numelor este cea mai plauzibilă dintre ipotezele care pot fi făcute în legatură cu ***World Wide Web***; ca şi mai sus însă, există situaţii în care ipoteza unicităţii numelor este utilă. Mai mult: ar putea fi nevoie ca numai pentru anumite segmente dintr-o ontologie să funcţioneze ipoteza de unicitate a numelor în timp ce pentru altele să funcţioneze ipoteza complementară.

4.5.5. Reguli de înlănţuirea proprietăţilor

Aşa cum s-a explicat mai sus, din motive de decidabilitate, ***OWL*** nu permite în mod normal compunerea proprietăţilor, deşi această operaţie ar fi foarte utilă în unele aplicaţii. Mai mult, proprietăţile nu pot fi definite ca reguli generale (Horn sau de alt fel) peste alte proprietăţi. O astfel de integrare a reprezentării cunoştinţelor pe baza regulilor cu reprezentarea cunoştinţelor în stil *DL* este un domeniu de cercetare foarte activ în acest moment.

4.5.6. Coduri ataşate

Un concept general în reprezentarea cunoştinţelor constă în definirea semnificaţiei unui termen nu printr-o definiţie explicită în limbajul respectiv (aşa cum se procedează în ***OWL***) ci prin ataşarea unei secvenţe de cod a cărei execuţie să conducă la calcularea semnificaţiei termenului. Deşi larg folosit, acest concept nu conduce foarte clar la integrarea într-un sistem înzestrat cu o semantică formală, drept care nu a fost inclus în ***OWL***.

Rezumat

- ***OWL*** este standardul propus pentru ontologiile ***Web***. Acesta ne permite să descriem semantica cunoştinţelor într-un mod accesibil pentru calculatoare.
- ***OWL*** se bazează pe ***RDF*** şi ***RDF Schema:*** este utilizată sintaxa ***RDF*** (bazată pe ***XML***); instanţele sunt definite cu ajutorul descrierilor ***RDF***; sunt folosite cele mai multe dintre primitivele de modelare ***RDFS***.
- Maparea ***OWL*** pe logică furnizează o semantică formală şi baza necesară pentru raţionamente. În acest scop au fost utilizate logica predicatelor şi logica descriptivă.
- Deşi ***OWL*** este suficient de bogat pentru a fi folosit în practică, se depun eforturi substanţiale pentru dezvoltarea lui, inclusiv a editoarelor sale grafice.

Bibliografie

Referinţe esenţiale pentru **OWL**:

- M. DEAN, G. SCHREIBER (eds.) F. van HERMELEN, J. HENDLER, I. HORROCKS, D. MCGUINNESS, P. PATEL-SCHNEIDER, L. STEIN: OWL Web Ontology Language Reference, August 2003 <http://www.w3.org/TR/owl-ref/>.
- Matthew HORRIDGE, Holger KNUBLAUCH, Alain RECTOR, Robert STEVENS, Chris WROE: *A practical Guide to Building **OWL** Ontologies Using the **Protégé**-**OWL** Plugin and CO=ODE Tools Edition 1.0*, Copyright©The Manchester University, August 2004.
- D. MCGUINNESS, F. van HERMELEN (eds.): OWL Web Ontology Language Overview, August 2003 <http://www.w3.org/TR/owl-features/>.
- M. SMITH, C. WELTY, D. MCGUINNESS (eds.): OWL Web Ontology Language Guide, August 2003 <http://www.w3.org/TR/owl-guide/>.
- <http://www.w3.org/2001/sw/WebOnt/> pentru informaţii despre OWL.
- <http://www.daml.org> Informaţii despre DAML+OIL. Vezi mai ales paginile /language, /ontologies, /tools.

Următoarele adrese ***Web*** se referă tot la noţiunea generală de ontologie dar într-o abordare diferită. Lexicoanele (*thesauri* = enciclopediile, glosarele, vocabularele, dicţionarele) nu sunt decât nişte ontologii informale.

- <http://www.lub.lu.se/metadata/subject_help.html> O colecţie amplă de pointeri către lexicoane.
- <http://www.topicmaps.org> Hărţile de subiecte (*Topic maps*) constituie un limbaj simplu pentru ontologii utilizat şi el în prezent.
- <http://www.dublincore.org> *Dublin Core* este un exemplu de ontologie larg utilizat în domeniul bibliotecilor online.