
# Inteligenta Artificiala

**Universitatea Politehnica Bucuresti**
**Anul universitar 2010-2011**

Adina Magda Florea

http://turing.cs.pub.ro/ia_10 si curs.cs.pub.ro

# Curs 9
# Modelul cunostintelor structurate

- Retele semantice
- Unitati
- Web semantic

# 1. Modelul *RETELELOR SEMANTICE*

- primul model structurat de reprezentare a cunostintelor
- introdus pentru a descrie semantica cuvintelor si a propozitiilor limbajului natural
- folosit ca metoda de reprezentare a cunostintelor in sistemele bazate pe cunostinte

# 1.1 Baza de cunostinte

- Radu i-a trimis Ioanei o scrisoare.
- Radu este student.
- Ioana este eleva.
- Adresa lui Radu este Luterana, 15.

- Ocupatie (radu, student)
- Ocupatie (ioana, eleva)
- Trimite (radu, ioana, scrisoare)
- Adresa (radu, luterana - 15)

# Gruparea in entitati

- Radu
  - Ocupatie (radu, student)
  - Trimite (radu, ioana, scrisoare)
  - Adresa (radu, luterana - 15)
- Ioana
  - Ocupatie (ioana, eleva)
  - Trimite (radu, ioana, scrisoare)

**Radu**

Ocupatie: student
Adresa: luterana-15

**Ioana**

Ocupatie: elev

$$(\exists x)(\text{ISA}(x, \text{eveniment-trimitere}) \wedge \text{Expeditor}(x, \text{radu}) \wedge \text{Destinatar}(x, \text{ioana}) \wedge \text{Obiect}(x, \text{scrisoare}))$$

Predicatul ISA indica apartenenta unui obiect la o multime.

$$\text{ISA}(t_1, \text{eveniment-trimitere}) \wedge \text{Expeditor}(t_1, \text{radu}) \wedge \text{Destinatar}(t_1, \text{ioana}) \wedge \text{Obiect}(t_1, \text{scrisoare})$$

**Radu**

| | |
|---|---|
| ISA: | Persoana |
| Ocupatie: | student |
| Adresa: | luterana-15 |

**Ioana**

| | |
|---|---|
| ISA: | Persoana |
| Ocupatie: | elev |

**T1**

| | |
|---|---|
| ISA: | Eveniment-trimitere |
| Expeditor: | Radu |
| Destinatar: | Ioana |
| Obiect: | scrisoare |

Predicatul AKO descrie incluziunea unei multimi intr-o alta multime

$$(\forall x)\,(\text{Eveniment-trimitere}\,(x) \rightarrow \text{AKO}(x, \text{Eveniment})$$

$$(\forall x)\,(\text{Persoana}\,(x) \rightarrow \text{AKO}(x, \text{Fiinta})$$

**Eveniment-trimitere**

AKO: Eveniment
Expeditor: Persoana
Destinatar: Persoana
Obiect: ClasaObiect

**Persoana**

AKO: Fiinta
Ocupatie: (student, elev, inginer)
Adresa: string

**Relatie individual-generic**, sau instanta-clasa, notata **ISA** (prescurtare de la IS A).

**Relatia generic-generic**, sau subclasa-clasa, notata **AKO** (prescurtare de la A Kind Of).

Obicete particulare / obiecte generice

Sloturi

Inferente specifice

**Mostenirea proprietatilor (atributelor) :**

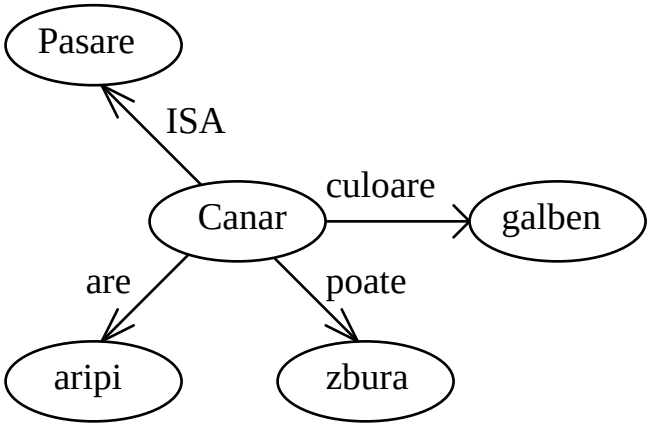
1) Mostenirii proprietatilor de la clasa la instanta:

Daca un obiect $O_1$ este o particularizare (legat prin relatia ISA) a unui obiect generic O si obiectul O are un atribut (proprietate) A, atunci si instanta $O_1$ are atributul A.

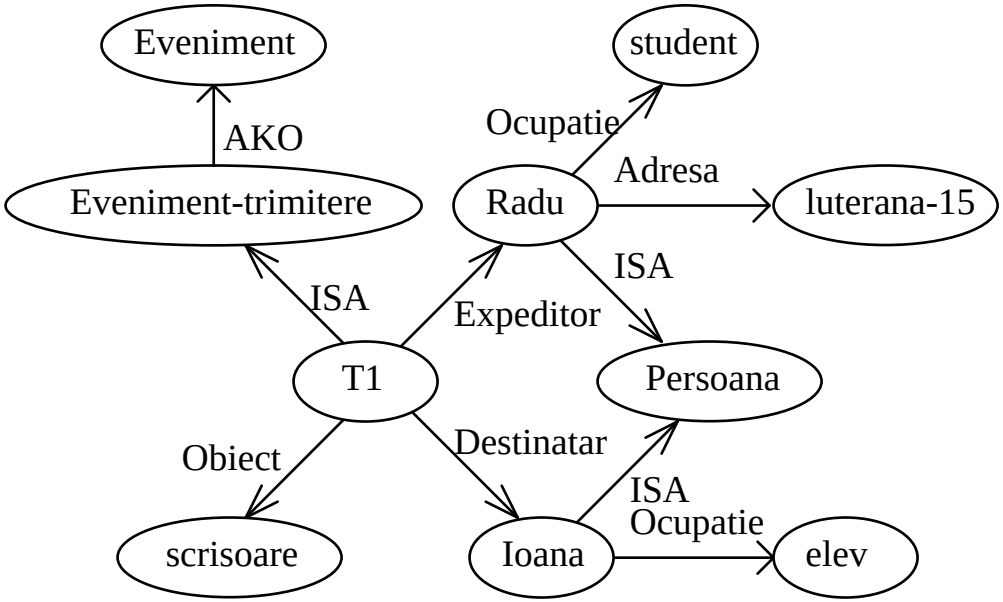
2) Aplicarea mostenirii proprietatilor intre o clasa si o superclasa, de-a lungul unei relatii sau a unui lant de relatii AKO

Daca o clasa $C_1$ este o subclasa a unei clase C (legata prin una sau mai multe relatii AKO) si clasa C are proprietatea A, atunci clasa $C_1$ are de asemenea proprietatea (atributul) A.

# 1.2 Retele semantice

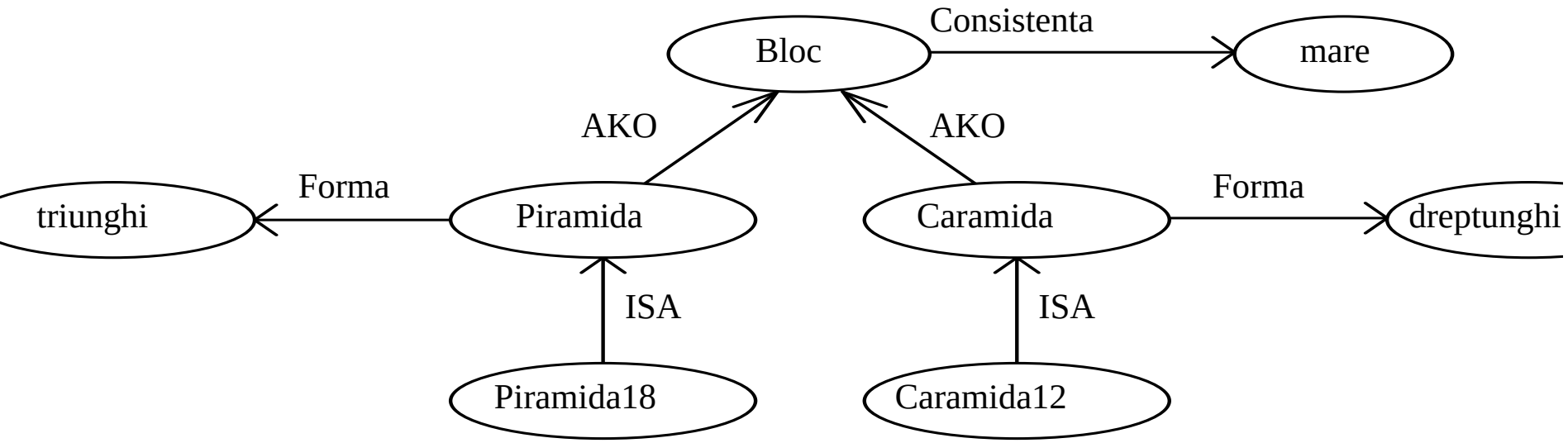



(a)




(b)

# Inferente specifice retelelor semantice




**M ostenirea valorilor in retele semantice**

**Algoritm**: **Mostenirea valorilor atributelor** intr-o ierarhie de clase

Algoritmul determina valoarea unui atribut A al unei instante O

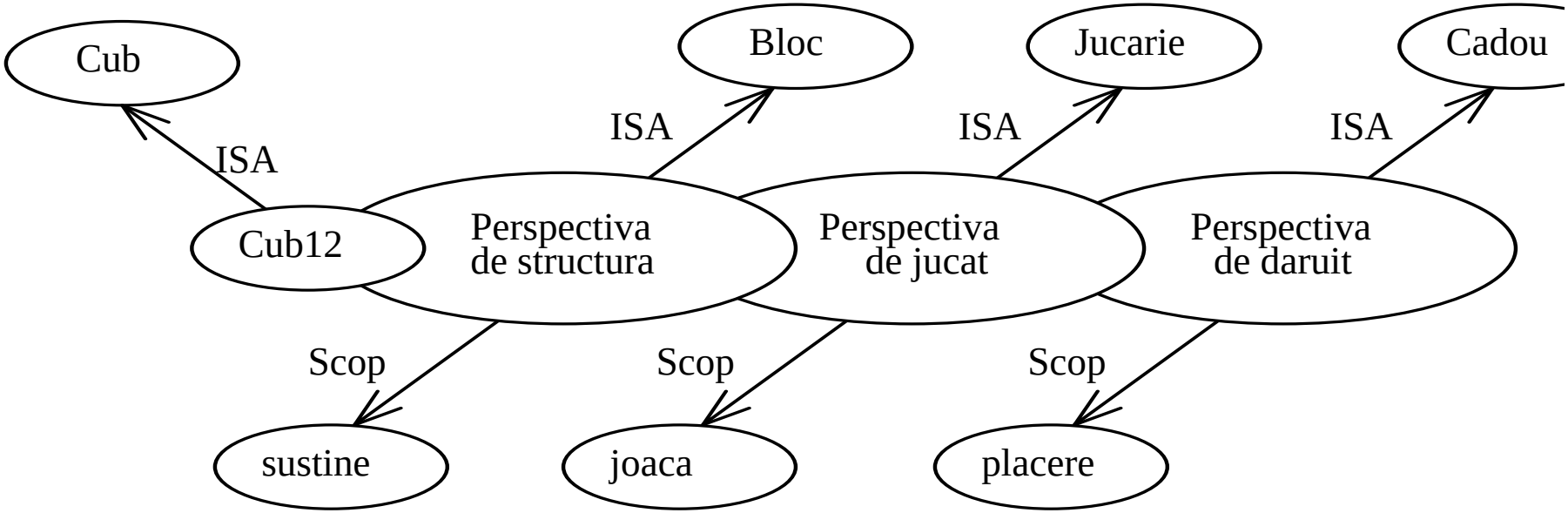
DetVal (O, A, V)

1. Formeaza o lista L cu nodul O si toate nodurile legate de O prin relatia ISA
2. **cat timp L != [ ] executa**
   - 2.1. Elimina primul nod, N, din lista L
   - 2.2. **daca** atributul A al nodului N are valoarea V **atunci**
     - 2.2.1. Depune V in nodul punctat de atributul A al obiectului O
     - 2.2.3. **intoarce** SUCCES
   - 2.3. Adauga toate nodurile legate prin relatia AKO de nodul N, la sfirsitul listei L
3. **intoarce** INSUCCES

**sfarsit**.

# Perspective

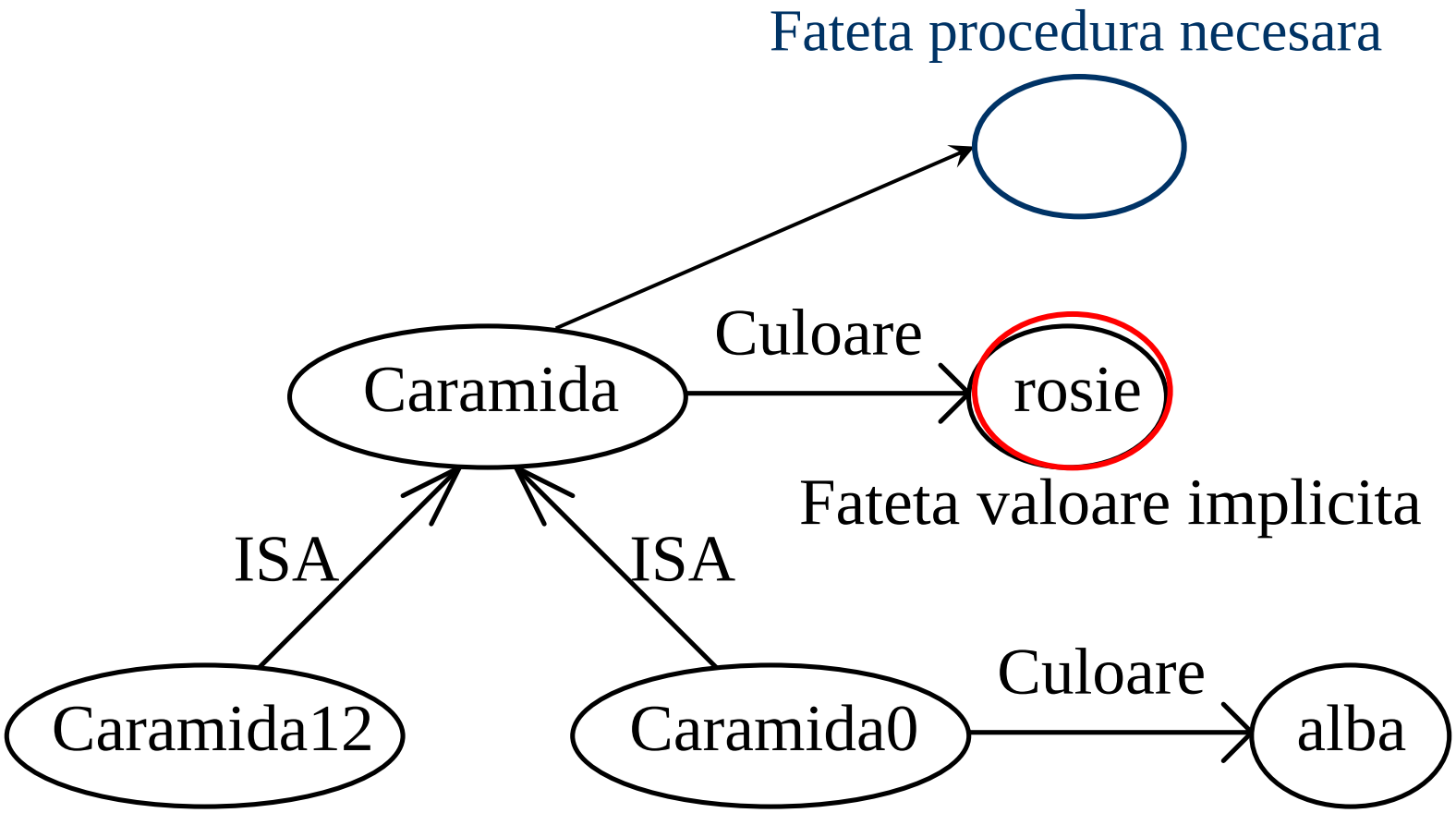
- Perspectiva - un obiect poate avea sensuri diferite in contexte diferite



# Fatete

- Caracteristici asociate atributelor din retea
- Modalitati de considerare a valorilor unor atribute

- **Fateta valoare** - valoarea obisnuita a unui atribut
- **Fateta valoare implicita** - caracterizeaza tipic valoarea unui atribut
- **Fateta procedura necesara** - contine o procedura sau o functie care poate calcula valoarea atributului pe baza valorii altor atribute




**Mostenirea valorilor implicite in retele semantice si a valorilor procedura necesara**

**Algoritm**: **Mostenirea valorilor implicite** ale atributelor intr-o ierarhie de clase

Algoritmul determina valoarea unui atribut A al unei instante O

DetValImp(O, A, V)

1. Formeaza o lista L cu nodul O si toate nodurile legate de O prin relatia ISA
2. **cat timp L != [ ] executa**
   - 2.1. Elimina primul nod, N, din lista L
   - 2.2. **daca** atributul A al nodului N are valoarea implicita V

     **atunci**
     - 2.2.1. Depune V in nodul punctat de atributul A al obiectului O
     - 2.2.3. **intoarce** SUCCES
   - 2.3. Adauga toate nodurile legate prin relatia AKO de nodul N, la sfirsitul listei L
3. **intoarce** INSUCCES

**sfarsit**.

# 1.3. Strategii de control

- Strategia de control indica ordinea de aplicare a inferentelor si modul de inspectare a retelei

- Doua stategii de control de baza
  - Stategia N
  - Strategia Z

# Strategia N

**Algoritm**: Strategia N de determinare a valorii unui atribut
Algoritmul determina valoarea unui atribut A al unei instante O utilizind strategia N.

DetValN (O, A, V)

1. **daca** DetVal (O,A,V) = SUCCES
   **atunci intoarce** SUCCES
2. **daca** DetValImp (O,A,V) = SUCCES
   **atunci intoarce** SUCCES
3. **daca** DetProcNec (O,A,V) = SUCCES
   **atunci intoarce** SUCCES
4. **intoarce** INSUCCES

**sfarsit**.

# Strategia Z

**Algoritm**: Strategia Z de determinare a valorii unui atribut.
Algoritmul determina valoarea unui atribut A al unei instante O utilizind strategia Z.

DetValZ (O, A, V)

1. Formeaza o lista L cu nodul O si toate nodurile legate de O prin relatia ISA
2. **cat timp L != [ ] executa**
   - 2.1. Elimina primul nod, N, din lista L
   - 2.2. **daca** fateta valoare a atributului A a nodului N este V
     **atunci**
     - 2.2.1. Depune V in nodul punctat de atributul A al obiectului O
     - 2.2.2. **intoarce** SUCCES

2.3. **daca** fateta valoare implicita a atributului A a nodului N este V
**atunci**
2.3.1. Depune V in nodul punctat de atributul A al obiectului O
2.3.2. **intoarce** SUCCES
2.4. **daca** fateta procedura necesara a atributului A a nodului N este proc ($A_1$,..., $A_n$,V)
**atunci**
2.4.1. Determina valorile atributelor $A_1$,..., $A_n$ ale instantei O
2.4.2. **daca** s-au gasit valori pentru $A_1$,..., $A_n$
**atunci**
i. **executa** proc ($A_1$,..., $A_n$,V)
ii. Depune V in nodul punctat de atributul A al obiectului O
iii. **intoarce** SUCCES
3. **intoarce** INSUCCES
**sfarsit**.

# 2. Modelul Unitatilor

- **Unitate** - colectie de atribute (sloturi), cu valori asociate si posibile restrictii asupra valorilor, ce descriu un obiect al universului problemei

- Unitatile pot desemna
  - **obiecte generice**
  - **instante**

# 2.1 Reprezentarea relatiilor

- Retele semantice
  - **AKO**
  - **ISA**
- Unitati
  - **SuperClasses**
  - **SubClasses**
  - **MemberOf**

Un obiect particular poate fi instanta a mai multe unitati generice, iar o unitate generica poate fi subclasa a mai multe clase

taxonomia de unitati poate fi graf.

# Sloturi

- Fiecare slot are un nume si una sau mai multe valori

- **Tipuri de sloturi**
  - sloturi membru - **MemberSlot** - descriu atributele fiecarul membru al clasei
  - sloturi proprii - **OwnSlot** - descriu atributele ce caracterizeaza clasa ca un intreg

**Unit** Camion

**SuperClasses**: Vehicul
**SubClasses**: CamionMare, CamionMediu, CamionMic
**MemberOf**: ObiecteFizice

**Unit** CamionMare

**SuperClasses**: Camion
**SubClasses**: CamionMareRosu, CamionMareRemorca

**Unit** CamionMareRosu

**SuperClasses**: CamionMare
**MemberSlot**: Sofer
**Value**: necunoscut
**MemberSlot**: Inaltime
**Value**: necunoscut
**MemberSlot**: Culoare
**Value**: rosie
**MemberSlot**: Pret
**Value**: necunoscut
**OwnSlot**: CelMaiMare
**Value**: CMR10
**OwnSlot**: CelMaiScump
**Value**: CMR210

**Unit** CMR1

**MemberOf**: CamionMareRosu, ProprietateFirmaX
**OwnSlot**: Sofer
**Value**: Paul
**OwnSlot**: Inaltime
**Value**: 1.75
**OwnSlot**: Culoare
**Value**: rosie
**OwnSlot**: Pret
**Value**: 30 000
**OwnSlot**: Proprietar
**Value**: X

**Unit** CMR2

**MemberOf**: CamionMareRosu
**OwnSlot**: Sofer
**Value**: Tudor
**OwnSlot**: Inaltime
**Value**: 1.80
**OwnSlot**: Culoare
**Value**: rosie
**OwnSlot**: Pret
**Value**: 50 000

# 2.2 Reguli de mostenire

- In urma mostenirii atributelor de la clasa la instanta, sloturile membru ale clasei devin sloturi proprii ale instantei, iar sloturile proprii ale clasei nu se mostenesc la instante.
- Orice slot membru al unei clase este mostenit de subclasele descendente din acea clasa, in urma mostenirii atributelor de la clasa la subclasa

# Fatete

- Fatete - modalitati de reprezentare a proprietatilor atributelor
- Tipuri de fatete
  - **fateta valoare**
  - **fateta domeniu de valori**
  - **fatete ce descriu restrictii**
  - **fateta valoare implicita**
  - **fateta mostenire**
  - **fateta valoare activa**
  - **fateta comentariu**

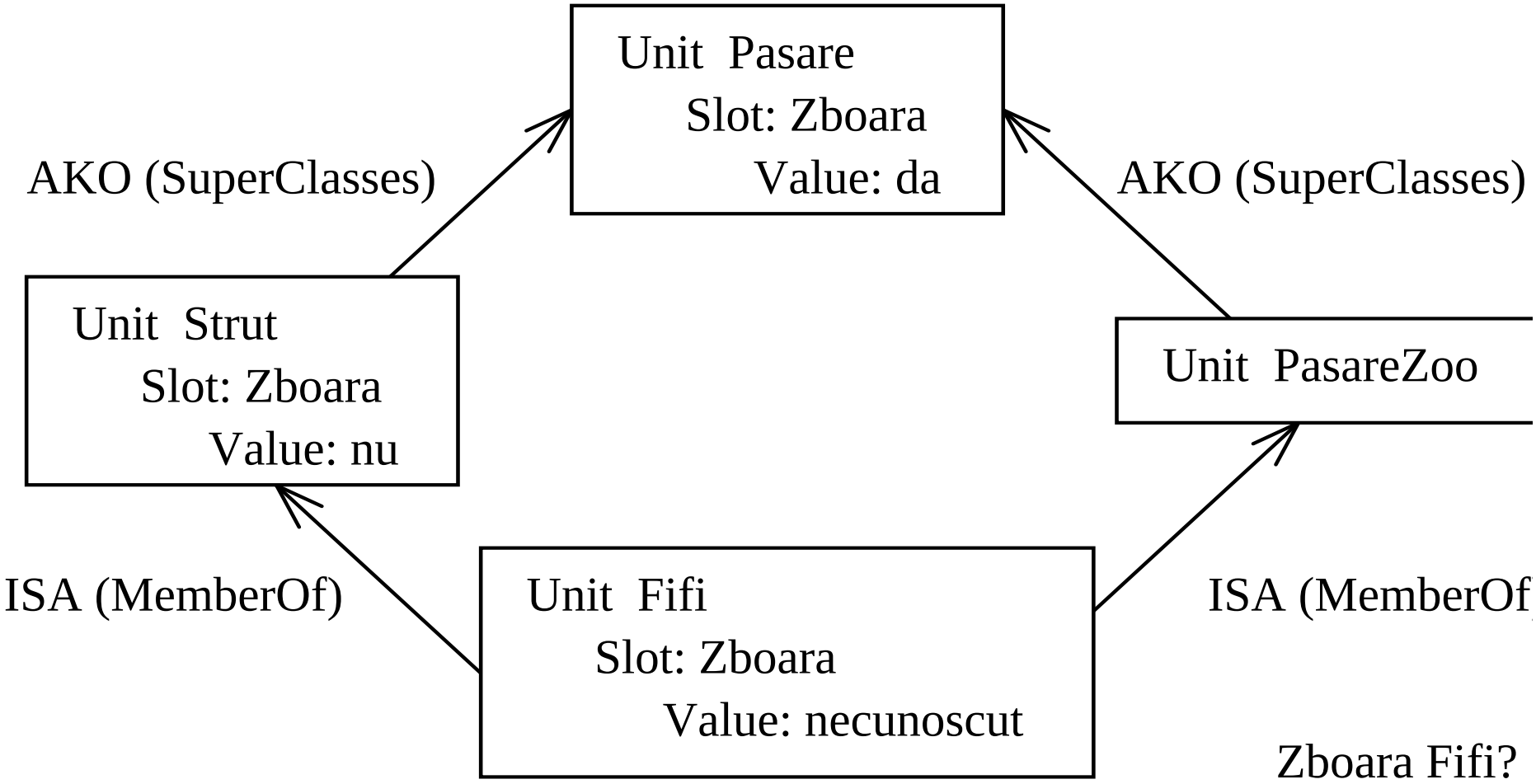
**Unit** CamionMareRosu

**SuperClasses**: CamionMare
**MemberSlot**: Sofer
**Value**: necunoscut /*fateta valoare */
**ValueClass**: Persoana /*fateta domeniu de valori; indica unitatea Persoana */
**Cardinality**: 2 /*fateta numar de valori; un camion poate avea doi soferi posibili */
**Default**: Paul /*fateta valoare implicita*/
**Restrict**: (oneof Paul, Tudor, Gelu, Mihai, Barbu) /*fateta de descriere a restrictiei */
**MemberSlot**: Inaltime
**Value**: necunoscut
**ValueClass**: real
**Cardinality**: 1
**Restrict**: X.Inaltime > 1.50
**MemberSlot**: Culoare
**Value**: rosie
**ValueClass**: string
**Cardinality**: 1
**Comment**: "Culoarea tuturor membrilor unitatii" /*fateta comentariu */

# 2.3 Inferente specifice unitatilor

- Forma de inferenta specifica - **mostenirea atributelor**

- Forma taxonomiei de unitati este un **graf orientat aciclic**, in care exista o relatie ordine partiala impusa de relatiile **ISA** sau **MemberOf** si **AKO** sau **Subclass/Superclass (relatii ierarhice)**

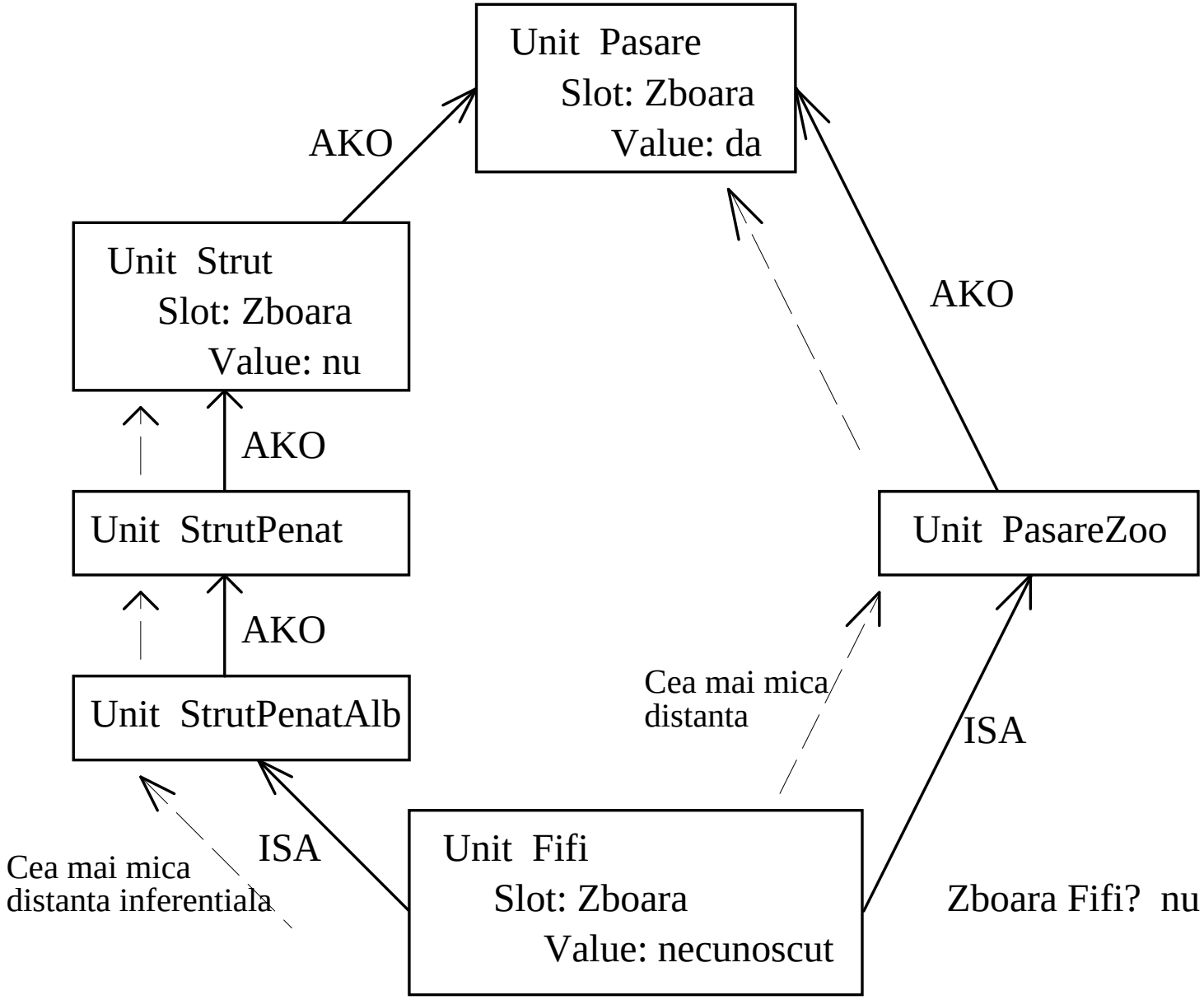



**Taxonomie de unitati de tip graf orientat aciclic**

## Probleme

# Mosteniri multiple de atribute

- **Distanta dintre unitati** - se foloseste lungimea caii intre unitatea curenta U pentru care se doreste aflarea valorii slotului S si unitatea U' unde s-a gasit aceasta valoare, considerind corecta valoarea slotului din unitatea cea mai apropiata de unitatea U




# Probleme

**Distanta si distanta inferentiala intre unitati**

# Mosteniri multiple de atribute

**Distanta inferentiala**

- **Clasa1** este mai aproape de **Clasa2** decat de **Clasa3** daca si numai daca **Clasa1** are o cale inferentiala care trece prin **Clasa2** spre **Clasa3**.
- **Clasa1** este mai aproape de **Clasa2** decat de **Clasa3** daca **Clasa2** este intre **Clasa1** si **Clasa3** de-a lungul unui lant de relatii ierarhice.

**Algoritm**: **Mostenirea atributelor bazata pe distanta inferentiala**

Algoritmul determina valoarea V a slotului S al unitatii U

1. Formeaza o lista L cu unitatea U si toate unitatile legate de U prin relatia MemberOf
2. Formeaza o lista de candidati CAND = [ ]
3. **cat timp** L != [ ] **executa**
   - 3.1. Elimina prima unitate, X, din lista L
   - 3.2. **daca** slotul S al lui X are valoare
     **atunci** CAND = CAND $\cup${X}
   - 3.3. **altfel** adauga in lista L toate unitatile legate de X prin relatia SuperClass
4. **pentru** fiecare unitate C $\in$ CAND **executa**
   - 4.1. Verifica daca exista un alt element C' $\in$ CAND cu o distanta inferentiala fata de U mai mica decat cea a lui C
   - 4.2. **daca** C' exista
     **atunci** elimina C din CAND

5. **daca** card (CAND) = 0
   **atunci intoarce** INSUCCES  /* nu s-a gasit valoare pentru S */
6. **daca** card (CAND) = 1
   **atunci**
   6.1. Fie C1 unicul element al listei CAND
   6.2. Depune valoarea slotului S al lui C1 ca valoare a slotului S al lui U
   6.3. **intoarce** SUCCES
7. **daca** card (CAND) > 1  /* contradictie, S este monovaloare */
   **atunci intoarce** CONTRADICTIE

**sfarsit**.

# 2.4 Reprezentari combinate

Baza de cunostinte formata din:

- cunostinte declarative: unitati

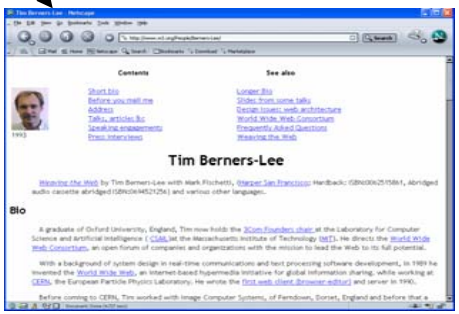
- cunostinte procedurale: reguli

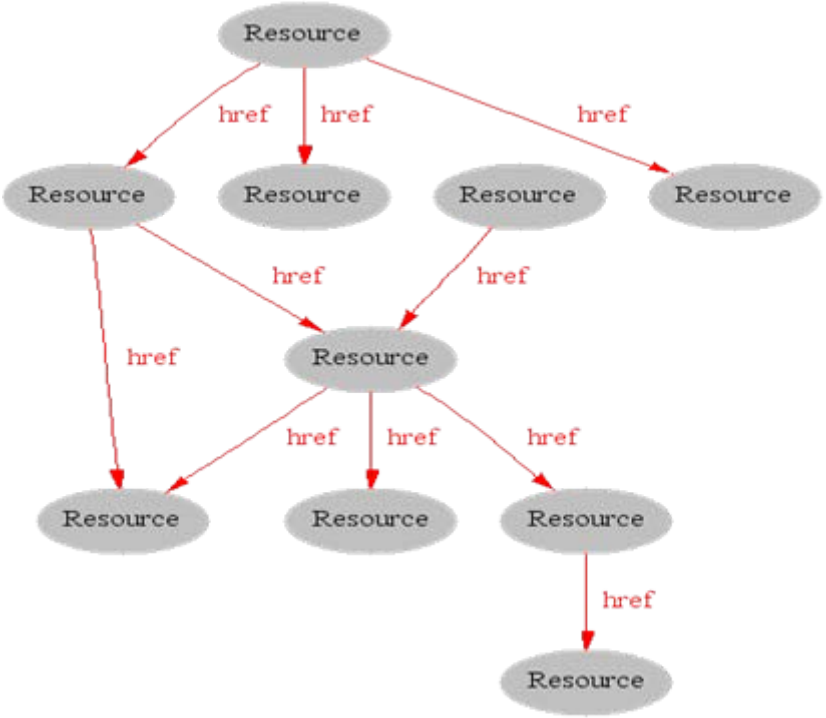
**daca** Camion.Inaltime > 2
**si** Camion.Culoare = rosu
**atunci**
Camion.Pret = 1000

Inferente in sistem: specifice unitati si specifice reguli

# 3. Web semantic

- Astazi avem web sintactic
- Markup se refera la:
  - afisarea informatiei (dim. font, culoare, etc.)
  - Hhyper-linkuri pt a lega continut

# Web semantic

- Semantic Web
  - Necesita o reprezentare a continutului
  - Semantica – cum reprezentam?
- Adnotari
- Conventii asupra semnificatiei adnotarilor
- Utilizarea ontologiilor pt a specifica adnotarile
  - Vocabular de termeni
  - Noi termeni care se formeaza din cei exsitenti + relatii intre termeni
  - Semantica specificata formal

# 3.1 Ontologie

- In stiinta calculatoarelor o **ontologie** este o **reprezentare formala a unei multimi de concepte** dintr-un anumit domeniu impreuna cu **relatiile** dintre aceste concepte
- O ontologie contine:
  - o **descriere ierarhica** a celor mai importante concepte dintr-un domeniu
  - descrie **principalele proprietati** ale fiecarui concept pe baza unui mecanism de tip atribut-valoare
  - **indivizii** din domeniul de interes sunt asignati unuia sau mai multor concepte in scopul de a le da un tip corespunzator.

# Descrierea unei ontologii

- Limbaje bazate pe logica cu predicate – CycL, F-logic, OCML, Ontolingua;
- Limbaje bazate pe web - DAML+OIL, OWL, RDF, RDF Schema, SHOE;
- Limbaje bazate pe logici de descriere: OWL

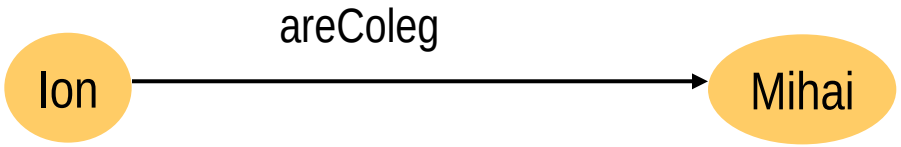
# 3.2 RDF

- XML
  - Specifica un arbore al documentului
  - Nu identifica continutul documentului
- RDF incearca sa exprime continutul
- RDF permite descrierea resurselor
- O resursa este un obiect
  - pt care se poate da o descriere
  - este identificat printr-un URI dar sau printr-o descriere abstracta (nu neaparat se mapeaza la o adresa de retea)
  - Ex: docs, imags, videoclips, services, unele in afara web (oameni, obiecte)
- Un literal este
  - O valoare (string, integer, ..)
  - Nu i se poate da o descriere
  - Exista literali cu tip: sirul + referinta URI la o XML Schema care descrie tipul

# RDF

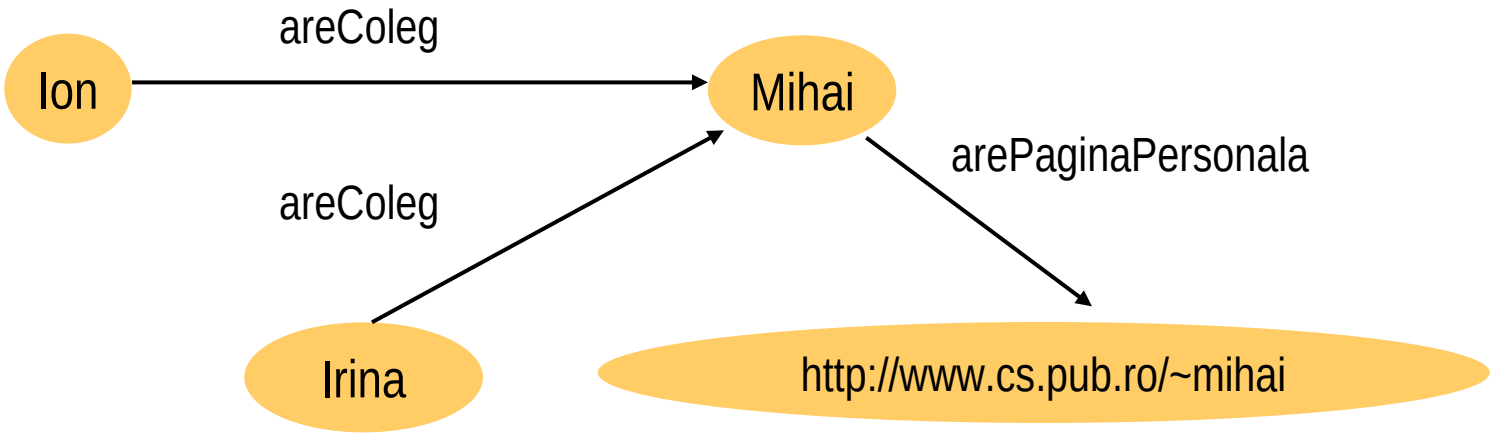
- RDF se bazeaza pe o gramatica simpla
- Un document RDF este o multime de instructiuni sau triplete
- Fiecare instructiune este formata din
  - *Subiect:* o resursa
  - *Obiect:* o resursa sau un literal
  - *Predicat:* o resursa (proprietate rdf:Property)
  - Subiectul este legat de obiect prin predicat (predicate binare)

# RDF




Poate fi reprezentat utilizand serializare XML:

<Ion,areColeg,Mihai>



# RDF Schema

- RDF ofera un formalism pentru adnotarea metadatelor si modul de scriere in XML dar nu ofera semnificatie unor relatii standard cum ar fi subClassOf sau type
  - de ex <Persoana,subClassOf,Animal> nu are os emnifiactei speciala

- RDF Schema defineste o “schema de vocabular” care permite definirea ontologiilor
  - ofera semnificatie unor relatii (predicate) standard din RDF (de ex subClasOf)
  - aceasta semnificatie indica cum trebuie interpretata relatia

# RDF Schema

- Exemple din RDF Schema :
  - Class
  - Property
  - type
  - subClassOf
  - range
  - domain
- Acestea reprezinat constructorii din RDF Schema utilizati pentru crearea vocabularului:

<Persoana,**type**,**Class**>

<areColeg,**type**,**Property**>

<Profesor,**subClassOf**,Persoana>

<Irina,**type**,Profesor>

<areColeg,**range**,Persoana>

<areColeg,**domain**,Persoana>

# 3.3 OWL

- Web Ontological Language
- OWL impune restrictii suplimentare
  - bazat pe RDF dar limiteaza "libertatea" RDF
  - odera os emantica formala
- Bazat de Logici de descriere
- Clase
- Indivizi
- Proprietati

# OWL

- Clase

```
<owl:Class rdf:ID="Student"/>
<owl:Class rdf:ID="Department"/>
<owl:Class rdf:ID="Course"/>
```

- Indivizi

```
<Department rdf:ID="CS"/>
```

- Proprietati

```
<owl:ObjectProperty rdf:ID="takes">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#Course"/>
</owl:ObjectProperty>
```

# Clase OWL

- Cum se construieste o clasa?

**(a) Prin specificarea unui nume de clasa**

```
<owl:Class rdf:ID="Student"/>
```

**(b) Prin specificarea nume clasa + descendenta**

```
<owl:Class rdf:ID="StudentFemeie">
  <rdfs:subClassOf rdf:resource="#Student"/>
</owl:Class>
```

**(c) combinatie de operatori logici**: **owl:IntersectionOf, owl:unionOf, owl:complementOf**

**sau enumerare owl:oneOf** (enumerare toti indivizii)

**(d) Restrictii asupra proprietatilor**

# Clase OWL

- Construirea claselor pe baza restrictiilor aplicate proprietatilor
- Obiectele care satisfac restrictia asupra proprietatii formeaza o clasa anonima
- **owl:Restriction**
- O restrictie poate fi de 2 tipuri
  - **owl:ObjectRestriction –** se aplica pe o proprietate obiect
  - **owl:dataRestriction –** se aplica pe o proprietate tip de date
- Proprietatea asupra careia se aplica restrictia este specificata prin **owl:onProperty**

# Clase OWL

```
<owl:ObjectProperty rdf:ID="urmeaza">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#Curs"/>
</owl:ObjectProperty>


<owl:Restriction>
        <owl:onProperty rdf:resource="#urmeaza"/>
        <owl:minCardinality
         rdf:datatype="&xsd;nonNegativeInteger"> 1
        </owl:minCardinality>
</owl:Restriction>
```

- O clasa anonima in care membrii urmeaza cel putin un curs. Stiind ca domeniul este Student se poate infera ca este o subclasa a Student.

# Clasa Student

```
<owl:Class rdf:about="Student">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#urmeaza"/>
         <owl:minCardinality

         rdf:datatype="&xsd;nonNegativeInteger">
              1
         </owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

# Clasa StudentBun

```
<owl:Class rdf:ID="StudentBun">
  <owl:IntersectionOf  rdf:parseType="Collection">
    <rdfs:Class rdf:about="#Student"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#urmeaza"/>
      <owl:minCardinality

       rdf:datatype="&xsd;nonNegativeInteger">
          3
      </owl:minCardinality>
      <owl:maxCardinality

       rdf:datatype="&xsd;nonNegativeInteger">
          5
      </owl:maxCardinality>
    </owl:Restriction>
  </owl:IntersectionOf>
</owl:Class>
```

# Exemplu de ontologie in Protege