* Create a new project at protege with uri: <http://faculty-semweb.googlecode.com/svn/trunk/ontorepo/xxx.owl>
* Enumerate important terms

Person

Man

Woman

Male

Female

Children

Sister

Brother

Sibling

Uncle

Aunt

Nephew

Niece

Son

Daughter

Father

Mother

* Organize the terms (again and again)

Gender

Male

Female

Person

Child

Son

Daughter

Sibling

Sister

Brother

Uncle

Aunt

Nephew

Niece

Father

Mother

* Organize the terms (again)

Gender

Male

Female

Person

Parent

Father

Mother

Relative

Uncle

Nephew

Aunt

Niece

Child

Son

Daughter

Sibling

Sister

Brother

* paraphrase and formalise the definitions

Man = Person AND Male

Woman = Person AND Female

Person = Man OR Woman

Daughter = Child AND Woman

Son = Child AND Man

Aunt = Woman AND (hasNephew OR hasNiece)

Nephew = Man AND (hasUncle OR hasAunt)

Niece = Woman AND (hasUncle OR hasAunt)

Child = Person AND hasParent

Sibling = Person AND hasSibling

Brother = Man AND Sibling

Sister = Woman AND Sibling

Relative = Sibling OR Parent OR Child OR Aunt OR Nephew OR Niece OR Uncle

...

* define classes and class hierarchies in Protege
* enter necessary properties (hasParent, hasChild...)
* run (default from protege menu) pellet reasoner (step1.a) and classify
* run dig reasoner (pellet) (step1.b)
  + run “pellet.bat dig” command
  + register localhost:8081 path at Protege Preferences page
  + Run Classifier from Protege
    - See under Relative
  + Demo DIG on Brother class
    - Remove Man from Necessary&Sufficient conditions
    - Add “hasGender has Male” condition
    - Run Classifier again
      * See Brother under Man
* Add Instances and make classification again (step2)

M1(Tolga)

F1(Pınar)

M2(Cem)

F4(Gülcan)

F5(Sevda)

F6(Lale)

M3(Fatih)

M4(Barış)

F7(Elif)

F2(Özlem)

F3(Zuhal)

* SPARQL Örnekleri (asserted model üzerinde işler...) (step2)

------

SELECT ?subject ?object

WHERE { ?subject rdfs:subClassOf ?object }

-----

PREFIX fam: <PREFIX fml: <http://faculty-semweb.googlecode.com/svn/trunk/ontorepo//family.owl#>

SELECT ?subject ?gender

WHERE { ?subject :hasGender ?gender }

-----

SELECT ?subject ?child

WHERE { ?subject :hasChild ?child }

-----

SELECT ?subject

WHERE { ?subject :name “Fatih”}

-----

SELECT ?subject ?parent

WHERE { ?subject :hasParent ?parent .

?parent :name "Fatih"

}

-----

SELECT \*

WHERE { ?subject rdf:type :Man .

}

-----

SELECT ?sname ?cname

WHERE { ?subject :name ?sname .

?subject :hasConsort ?consort .

?consort :name ?cname .

FILTER (?cname = "Lale") .

}

-----

-----

-----

* SWRL girişi (step3)

Tümünü girmeye gerek yok...

Man’in hasChild bilgileri girildi... #8 işletilirse, Woman’larla da hasChild/hasParent ilişkileri kurulur...

1. hasParent(?x, ?y) ∧ hasSister(?y, ?z) → **hasAunt**(?x, ?z)
2. hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
3. hasChild(?x, ?y) ∧ Woman(?x) → hasDaughter(?x, ?y)
4. hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
5. hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
6. hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → **hasNephew**(?x, ?z)
7. hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → **hasNiece**(?x, ?z)
8. hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
9. hasChild (?x, ?y) ∧ hasChild(?z, ?y) ∧ differentFrom(?x, ?z) → hasConsort(?x, ?z)
10. hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
11. hasChild(?x, ?y) ∧ Man(?x) → hasSon(?x, ?y)
12. hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → **hasUncle**(?x, ?z)