Semantic Web

Lecture 12: SW Programming

Dr. Knarig Arabshian
Knarig.arabshian@hofstra.edu

Hello Semantic Web World Example

- Say hello to the Semantic Web
- Say hello to some friends of the Semantic Web
- Expand friend list to include friends from different source
- Say hello to all Semantic Web's friends
- Say hello to a restricted list of only friends with email addresses
- Say hello to a restricted list of only friends with gmail.com addresses

What do we need?

- Compiling and execution tools: Java SDK
- Code-editing tools: Eclipse (eclipse.org)
- Ontology editing tool: Protege (protege.stanford.edu)
- Semantic Web Programming Framework: Jena API (downloads.sourceforgenet/jena)
- Ontology Reasoner: Pellet (clarkparsia.com/pellet)

```
<swp:me>
        rdf:type foaf:Person ;
        foaf:depiction <http://semwebprogramming.org> ;
        foaf:family_name “Web” ;
        foaf:givenname “Semantic” ;
        foaf:homepage <http://semwebprogramming.org> ;
        foaf:knows <swp:Reasoner>, <swp:Statement>, <swp:Ontology>
        foaf:name “Semantic Web”;
        foaf:nic     “Webby” ;
        foaf:phone <tel:123-456-7890>
        foaf:schoolHomepage <http://www.web.edu> ;
        foaf:title “Dr” ;
        foaf:wokInfoHomepage <http://semwebprogramming.com>;


<swp:Reasoner>
        rdf:type foaf:Person ;
        rdfs:seeAlso <http://reasoner.com> ;
        foaf:mbox <mailto:reason@hotmail.com>
        foaf:name “Ican Reason” .

<swp:Statement>
        rdf:type foaf:Person ;
        rdfs:seeAlso <http://statement.com> ;
        foaf:mbox <mailto:statement@gmail.com>
        foaf:name “Makea Statement” .

<swp:Ontology>
        rdf:type foaf:Person ;
        rdfs:seeAlso <http://ontology.com> ;
        foaf:mbox <mailto:ontology@gmail.com>
        foaf:name “I. M. Ontology” .
```

```
public class HelloSemanticWeb {

        static String defaultNameSpace = "http://org.semwebprogramming/
chapter2/people#";

        Model _friends = null;
        Model schema = null;
        InfModel inferredFriends = null;

        public static void main(String[] args) throws IOException {


                HelloSemanticWeb hello = new HelloSemanticWeb();

                //Load my FOAF friends
                System.out.println("Load my FOAF Friends");
                hello.populateFOAFFriends();

        //....
}
```

Two variables that contain the default namespace in setting up a URI and a model to hold the SW friends

_friends model holds the web of data

PopulateFOAFFriends() will populate the _friends model

```
private void populateFOAFFriends(){
        _friends = ModelFactory.createOntologyModel();
        InputStream inFoafInstance = FileManager.get().open("Ontologies/
FOAFFriends.rdf");
        _friends.read(inFoafInstance,defaultNameSpace);

        }
```

First step: create model by calling the ModelFactory.createOntologyModel() method. Load the model with the file

```
public static void main(String[] args) throws IOException {
// Say Hello to myself
        System.out.println("\nSay Hello to Myself");
        hello.mySelf(hello._friends);
//...
}

private void mySelf(Model model){
        //Hello to Me - focused search
        runQuery(" select DISTINCT ?name where{ swp:me foaf:name ?name  }",
model);

}
```

Query searches for the instance swp:me which retrieves the name “Semantic Web”

```
private void runQuery(String queryRequest, Model model){

        StringBuffer queryStr = new StringBuffer();
        // Establish Prefixes
        //Set default Name space first
        queryStr.append("PREFIX people" + ": <" + defaultNameSpace + "> ");
        queryStr.append("PREFIX rdfs" + ": <" + "http://www.w3.org/
2000/01/rdf-schema#" + "> ");
        queryStr.append("PREFIX rdf" + ": <" + "http://www.w3.org/
1999/02/22-rdf-syntax-ns#" + "> ");
        queryStr.append("PREFIX foaf" + ": <" + "http://xmlns.com/foaf/
0.1/" + "> ");
        queryStr.append(queryRequest);
        Query query = QueryFactory.create(queryStr.toString());
        QueryExecution qexec = QueryExecutionFactory.create(query, model);

        //..
}
```

Establish the query prefixes
QueryExecutionFactory() creates the query

```
        try {
                ResultSet response = qexec.execSelect();

                while( response.hasNext()){
                        QuerySolution soln = response.nextSolution();
                        RDFNode name = soln.get("?name");
                        if( name != null ){
                                System.out.println( "Hello to " +
                                name.toString() );
                        }
                        else
                                System.out.println("No Friends
                                found!");
                        }
                } finally { qexec.close();}
        }
```

ExecSelect() method executes the query

Iterate through the result set to list the hello candidates

Output:
Hello to Semantic Web

```
private void myFriends(Model model){
                //Hello to just my friends - navigation
                RunQuery(

" select DISTINCT ?name where

{   swp:me foaf:knows ?friend .
        ?friend foaf:name ?name } ",


model);

}
```

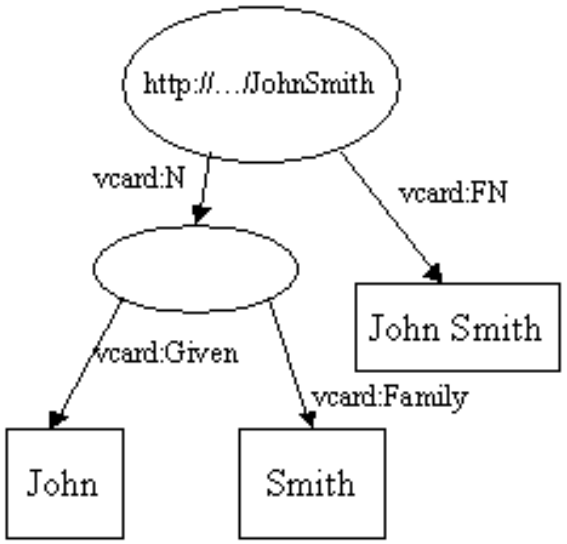
Reads: query for all the people I know and return their names.

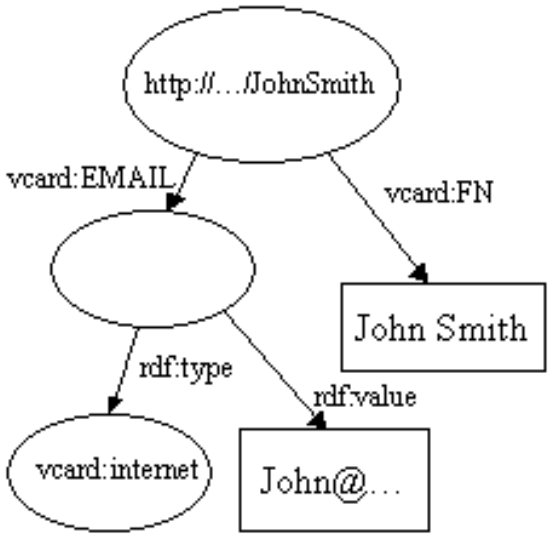
swp:me refers to the instance representing myself
Return all the friend values assigned to the foaf:knows property
For every friend, get the value of its foaf:name property

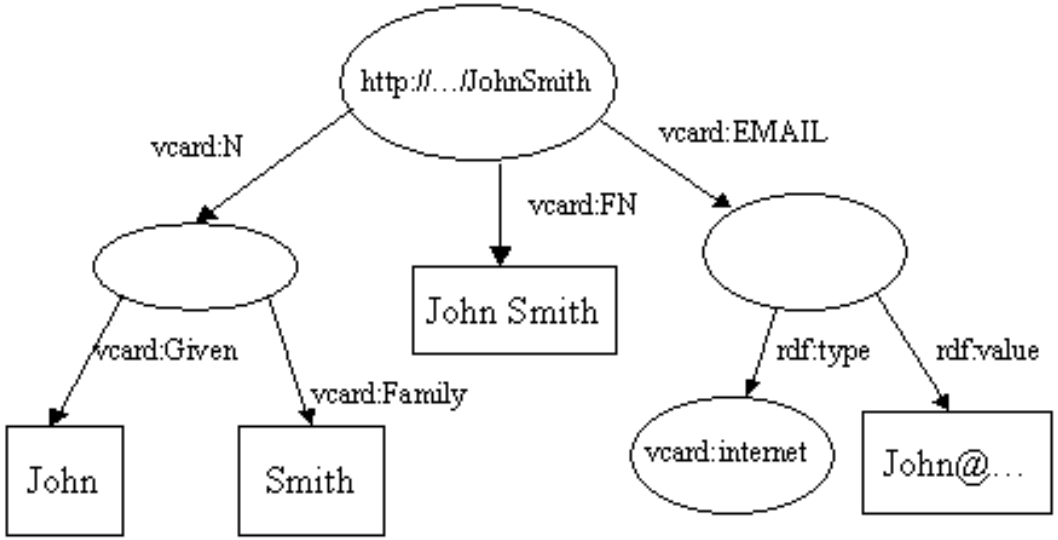
Output:
Hello to I. M. Ontology
Hello to Ican Reason
Hello to Makea Statement

Operations on Models

- Jena provides three operations for manipulating Models
  - Union
  - Intersection
  - Difference.
- The union of two Models is the union of the sets of statements which represent each Model.
  - Key operation supported by the design of RDF
  - Enables data from different data sources to be merged.


http://…/JohnSmith
vcard:N
vcard:FN
John Smith
vcard:Given
vcard:Family
John
Smith


http://…/JohnSmith
vcard:EMAIL
vcard:FN
John Smith
rdf:type
rdf:value
vcard:internet
John@…


http://…/JohnSmith
vcard:N
vcard:FN
vcard:EMAIL
John Smith
vcard:Given
vcard:Family
rdf:type
rdf:value
John
Smith
vcard:internet
John@…

```
// read the RDF/XML files
model1.read(new InputStreamReader(in1), "");
model2.read(new InputStreamReader(in2), "");

// merge the Models
Model model = model1. ```(model2);

// print the Model as RDF/XML
model.write(system.out, "RDF/XML-ABBREV");
```

Ontology Merging

- Add friends from a different source
- Source is based on an entirely different view of the world
- Use Protege to create a People ontology and associated instances of people
- This will be our second vocabulary
- Create Individual class with hasName datatype property and hasFriend object property

People Ontology

```
:Individual_5
    rdf:type :Individual ;
    :hasFriend :Individual_6, :Individual_7 ;
    :hasName “Sem Web” .
:Individual_6
    rdf:type :Individual ;
    :hasFriend :Individual_5 ;
    :hasName “Web O. Data” .
:Individual_7
    rdf:type :Individual ;
    :hasFriend :Individual_5 ;
    :hasName “Mr. Owl” .
```

```
private void populateNewFriends() throws IOException {
      InputStream inFoafInstance = FileManager.get().open("Ontologies/
additionalFriends.owl");
      _friends.read(inFoafInstance,defaultNameSpace);
      inFoafInstance.close();

}
```

If hello.myFriends() is run again after the second ontology is read, what happens?

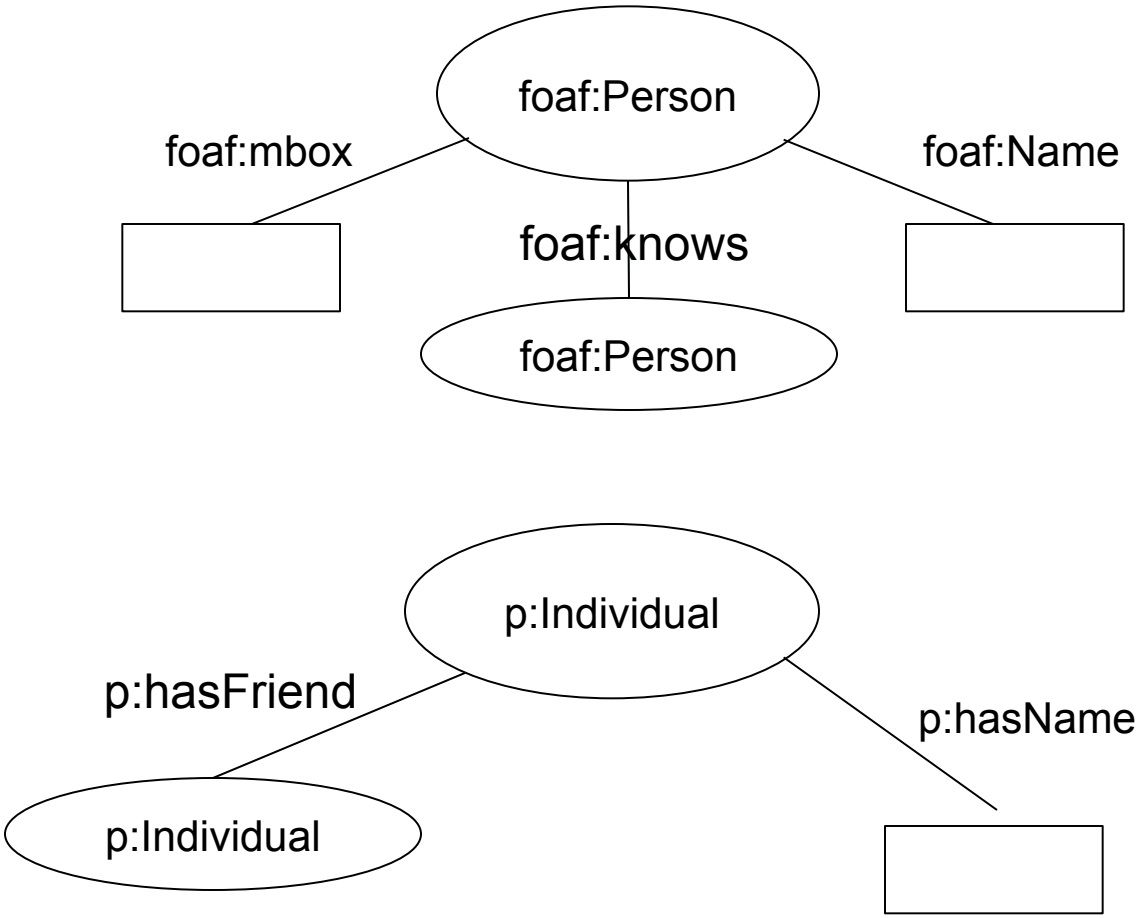
Only old friends are displayed because the ontology definition of the new model doesn't match the SPARQL query of the original hello.myFriends query

First one used the FOAF ontology. Second one uses our own People ontology

Ontology Merging

- Differences between the two ontologies
  - FOAF ontology refers to a person using the foaf:Person class
  - New ontology refers to a person using the people:Individual class
  - Both are semantically equivalent but syntactically different
- Because there is a difference in syntax, helloFriends query can't find the results for instances of the Individual class
- In order to align these two different names, we need to align the semantics
- We need semantic equivalence while allowing syntactic differences to remain

Ontology Merging



```
public static void main(String[] args) throws IOException {
// Add the ontologies
                System.out.println("\nAdd the Ontologies");
                hello.populateFOAFSchema();
                hello.populateNewFriendsSchema();

//...
}
```

Create new model to hold the ontologies and populate them

```
private void populateFOAFSchema() throws IOException{

        InputStream inFoaf = FileManager.get().open("Ontologies/foaf.rdf");
        InputStream inFoaf2 = FileManager.get().open("Ontologies/foaf.rdf");
        schema = ModelFactory.createOntologyModel();
        //schema.read("http://xmlns.com/foaf/spec/index.rdf");
        //_friends.read("http://xmlns.com/foaf/spec/index.rdf");

        // Use local copy for demos without network connection
        schema.read(inFoaf, defaultNameSpace);
        _friends.read(inFoaf2, defaultNameSpace);
        inFoaf.close();
        inFoaf2.close();
}
```

Ontology Alignment

```
private void populateNewFriendsSchema() throws IOException {
    InputStream inFoafInstance = FileManager.get().open("Ontologies/
    additionalFriendsSchema.owl”);
    _friends.read(inFoafInstance,defaultNameSpace);
    inFoafInstance.close();
}
```

- Separate new friend statements into two files
  - One containing new instances (friends)
  - One containing the ontology or schema (additionalFriendsSchema.owl)
- Models contain both ontologies and both sets of instance data
- But no information on how ontologies relate to one another
- Requires additional statements to align the two ontologies

Ontology Alignment

- Additions bridge ontologies in key areas
- Allows query to include all friends instead of just those from one or other ontology
- Alignment not necessary; query can be changed to incorporate the knowledge of both ontologies
- Not a good approach—every time ontology changes, would need to change the query.

Ontology Alignment

- Allows a SW program to incorporate new ontologies and data incrementally as the application and users learn about them
- Add ontologies together and adjust them over time
- No need for a priori knowledge of how the alignment occurs
- Add alignment statements during the running of your SW application
- Benefit: integration of SW data can be automated

Ontology Alignment

- First add two files of instance data based on different vocabularies
  - Does not break existing query but also does not expand its capability at returning friends
- Next, add statements to align the two ontologies
  - Program can dynamically add alignment when needed
  - Original query works as before without any problems

Ontology Alignment

- people:Individual equivalent to foaf:Person
- people:hasName equivalent to foaf:name
- people:hasFriend subproperty of foaf:knows
- Two choices: make pair fully equivalent or make one a specialization of the other
- foaf:Person and people:Individual refer to the same concept and hence are equivalent
- Same with foaf:name and people:hasName
- foaf:knows relationship between two people indicates some knowledge of a person but not necessarily a friendship with that person
- people:hasFriend is a specialization of foaf:knows

```
private void addAlignment(){

                // State that :individual is equivalentClass of
                //  foaf:Person
Resource resource = schema.createResource(defaultNameSpace +
"Individual");
Property prop = schema.createProperty("http://www.w3.org/2002/07/
owl#equivalentClass");
Resource obj = schema.createResource("http://xmlns.com/foaf/0.1/
Person");
schema.add(resource,prop,obj);

//...

}
```

Align class foaf:Person with the class people:Individual by declaring them equivalent

Creates resources for the subject and object along with the property

```
//State that :hasName is an equivalentProperty of foaf:name
resource = schema.createResource(defaultNameSpace + "hasName");
//prop = schema.createProperty("http://www.w3.org/2000/01/rdf-
schema#subPropertyOf");
prop = schema.createProperty("http://www.w3.org/2002/07/
owl#equivalentProperty");
obj = schema.createResource("http://xmlns.com/foaf/0.1/name");
schema.add(resource,prop,obj);
```

Align property foaf:name with the property people:hasName by declaring these equivalent

```
//State that :hasFriend is a subproperty of foaf:knows
resource = schema.createResource(defaultNameSpace + "hasFriend");
prop = schema.createProperty("http://www.w3.org/2000/01/rdf-
schema#subPropertyOf");
obj = schema.createResource("http://xmlns.com/foaf/0.1/knows");
schema.add(resource,prop,obj);


//State that sem web is the same person as Semantic Web
resource = schema.createResource("http://org.semwebprogramming/chapter2/
people#me");
prop = schema.createProperty("http://www.w3.org/2002/07/owl#sameAs");
obj = schema.createResource("http://org.semwebprogramming/chapter2/
people#Individual_5");
schema.add(resource,prop,obj);
```

Aligns foaf:knows with people:hasFriend by declaring people:hasFriend to be a subproperty of foaf:knows

Lastly, declare the two instances “Semantic Web” and “Sem Web” to be the same

Ontology Alignment

- Regardless of how many friends SW may have, if they are expressed according to either of these two ontologies, a query requesting the SW's friends returns them all
- Data need not be in the same file or same location
- Alignment statements can be kept separate in a different file or location
- Can maintain different alignment files for different alignment missions
- Any data read into the model that conforms to these two ontologies allows the program to identify all the friends
- If another ontology needs to be merged, only need to align the relations between the ontologies

Ontology Alignment

- Alignment statements provide logical foundation to add statements that equate the instance data
- But as things stand, the instance data remains the same
  - Ican Reason is still an instance of foaf:Person and not people:Individual
- Query will still not say hello to any of the new friends
- What do we need?
  - Reasoner to establish all the inferred relationships and return individuals that are inferred to be part of a class
  - Jena has a default OWL reasoner
  - Reasoner classifies ontology and based on the inferences produces a set of inferred statements

```
    private void bindReasoner(){
        Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
        reasoner = reasoner.bindSchema(schema);
        inferredFriends = ModelFactory.createInfModel(reasoner,
_friends);

    }
```

Method acquires Jena OWL reasoner and binds it to the schema

Schema allows some precalculations to improve performance of reasoning

Creates model bound to the reasoner based on the _friends model

Extends _friends model to include all inferences from the ontology

Can get the list of inferred friends by calling createInfModel

```
private void runPellet( ){

Reasoner reasoner = PelletReasonerFactory.theInstance().create();
reasoner = reasoner.bindSchema(schema);
inferredFriends = ModelFactory.createInfModel(reasoner, _friends);

ValidityReport report = inferredFriends.validate();

//printIterator(report.getReports(), "Validation Results");

        }
```

Same thing can be done using an external description logic reasoner like Pellet

Program can easily use different reasoners for different purposes

```
main {
        //Run reasoner to  align the instances
                System.out.println("\nRun a Reasoner");
                hello.bindReasoner();
                //System.out.println("Running Pellet");
                //hello.runPellet();

                //Say hello to all my friends
                System.out.println("\fFinally- Hello to all my friends!");
                hello.myFriends(hello.inferredFriends);
}
```

```
private void myFriends(Model model){
                  //Hello to just my friends - navigation
                  RunQuery(

" select DISTINCT ?name where

{   swp:me foaf:knows ?friend .
        ?friend foaf:name ?name } ",


model);

}
```

Now say hello to everyone in the SW model by passing the inferredFriends model to the myFriends query

Output:
Run a Reasoner
Finally - Hello to all my friends!
Hello to I. M. Ontology
Hello to Ican Reason
Hello to Makea Statement
Hello to Mr. Owl
Hello to Web O. Data