Ontology Development

Lalana Kagal
Decentralized Information Group

What is an ontology ?

- Formal specifications of the terms in a domain and the relations among them
  - Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220.
- Consists of concepts in a domain of discourse, properties of each concept, and restrictions on properties (such as range of values)

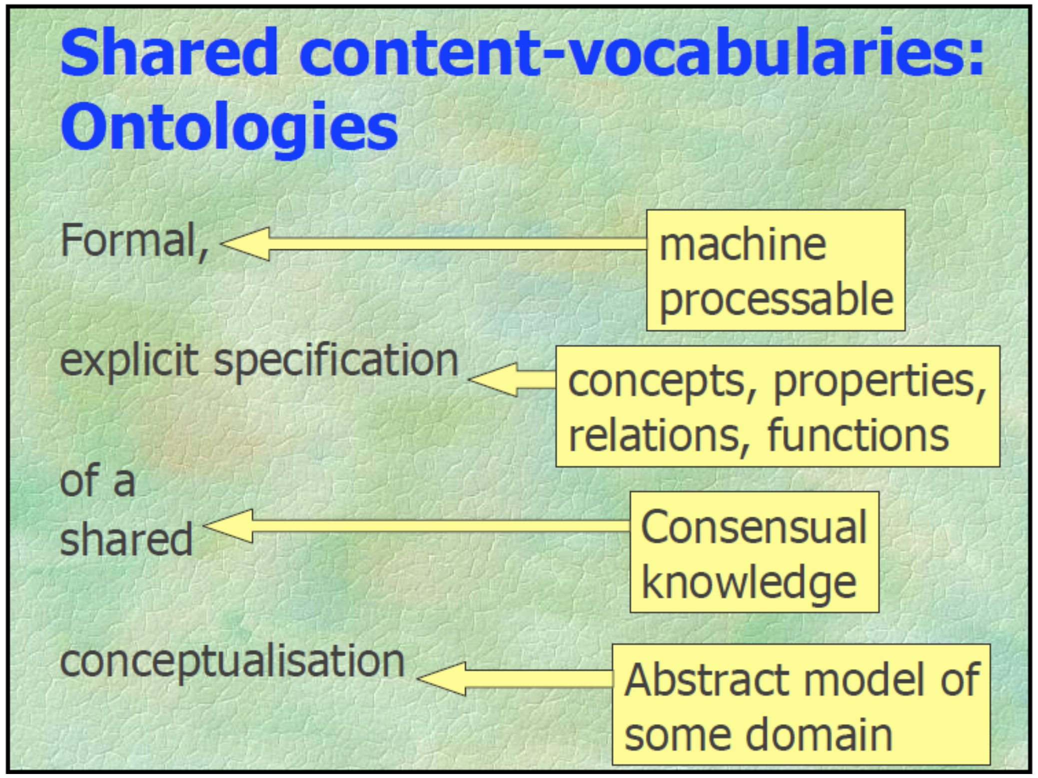


Image courtesy Frank Van Harmelen
http://knoesis.wright.edu/faculty/pascal/esslli06/slides/1b-semweb-languages.pdf

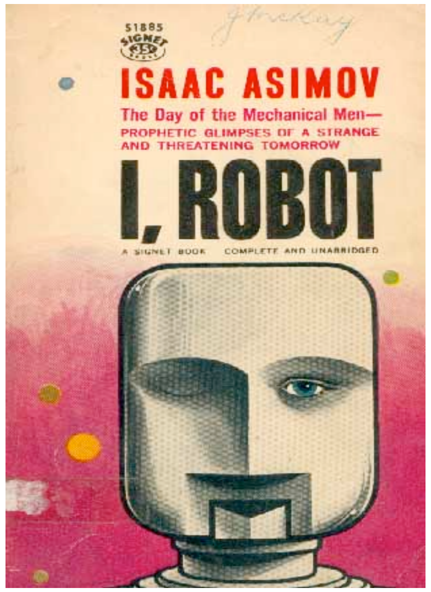
Why do we need ontologies ?

- Why do we need ontologies ?
  - Shared understanding of domain of interest
  - To enable reuse of domain knowledge
  - To make domain assumptions explicit
  - To analyze domain knowledge

Ontology, vocabulary and taxonomy ?

- What are taxonomies and vocabularies ? Are they the same ? Are they related to ontologies ?

Do I need to understand AI ?



- NO
- Some parts of SW languages are based on description logic
  - decidable fragment of FOL
  - includes efficient inference procedures for most common decision problems such as
    - instance checking (is a particular instance a member of a given concept)
    - relation checking (does a relation hold between two instances)
    - subsumption (is a concept a subset of another concept)
    - concept consistency (is there no contradiction among the definitions or chain of definitions)

Ontology Languages

- Two W3C recommendations for defining ontologies
  - RDF Vocabulary Description Language (RDFS)
    - RDFS defines the use of RDF to describe RDF vocabularies
  - Web Ontology Language (OWL)
    - OWL 1 provides more expressivity than RDFS and is used to express vocabularies / ontologies
    - OWL 1 has three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full (with reducing computational guarantees)
    - OWL 2 provides more expressivity over OWL 1

Ontology Languages

- RDFS
  - Set theory – rdfs:Class
  - Relation – rdf:Property, rdfs:domain, rdfs:range
  - Hierarchy – rdfs:subClassOf, rdfs:subPropertyOf
  - Built-in Datatype – xsd:string, xsd:dataTime
- OWL 1
  - Description Logic
    - Class, Thing, Nothing
    - DatatypeProperty, ObjectProperty, AnnotationProperty,…
  - Class axioms
    - oneOf, disjointWith, unionOf, complementOf, intersectionOf …
    - Restriction, onProperty, cardinality, hasValue…
  - Property axioms
    - inverseOf , TransitiveProperty , SymmetricProperty
    - FunctionalProperty, InverseFunctionalProperty
  - Equality– equivalentClass , sameAs , differentFrom…
  - Ontology annotation – imports, versionInfo

Notations

- property names begin with lowercase letter
  - parent is a property
  - use parent instead of hasparent
    - Ann (is) parent (of) Alice or Alice('s) parent (is) Ann
    - These slides assume Alice('s) parent (is) Ann
- class names begin with uppercase letter
  - Parent is a Class

RDFS

- @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
- rdfs:Resource
  - All things described by RDF are called resources and are instances of the class rdfs:Resource
  - This is the class of everything
  - All other classes are subclasses of this class
  - rdfs:Resource is an instance of rdfs:Class

  Alice rdf:type rdfs:Resource

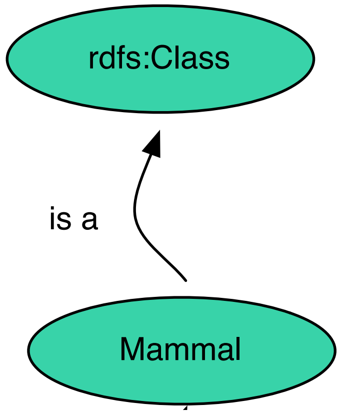
  with @keywords shorthand: Alice a rdfs:Resource

RDFS

- rdfs:Class

  Mammal rdf:type rdfs:Class

  with @keywords shorthand: Mammal a rdfs:Class



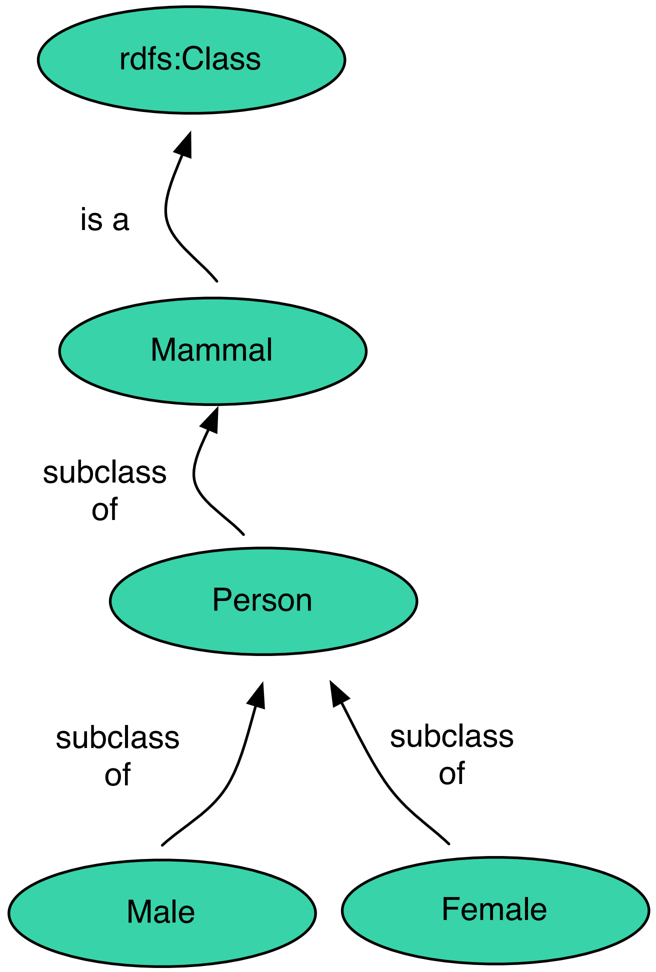
RDFS

- rdfs:subClassOf

  Person rdfs:subClassOf Mammal.
  Male rdfs:subClassOf Person.
  Female rdfs:subClassOf Person.

- multiple rdfs:subClassOf = intersection

  Parent rdfs:subClassOf Person.
  Mother rdfs:subClassOf Parent, Female.



RDFS

- rdf:type
  JoeLambda rdf:type Person

  with @keywords shorthand: JoeLambda a Person.

- RDFS reasoning

  Person rdfs:subClassOf Mammal.
  JoeLambda a Person.

  => JoeLambda a Mammal.

RDFS

- rdf:Property

  parent rdf:type rdf:Property

  with @keywords shorthand: parent a rdf:Property

- rdfs:subPropertyOf

  mother rdfs:subPropertyOf parent

  father rdfs:subPropertyOf parent

- RDFS reasoning
  mother rdfs:subPropertyOf parent.
  JoeLambda mother Alice.
  => JoeLambda parent Alice.

RDFS

- rdfs:range & rdfs:domain

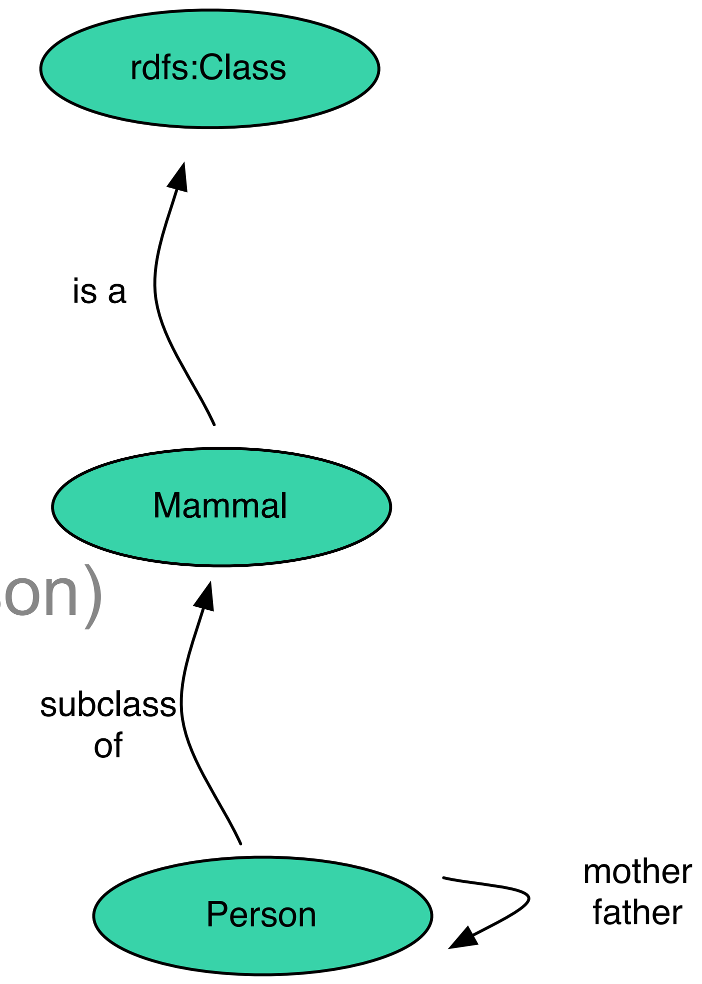
  To define range and domain of parent property

  father rdfs:range Male.

  father rdfs:domain Person.

- having range and domain set causes a property to act like a function from the set of instances of the domain (i.e Person) to the set of instances of the range (i.e. Male)

- RDFS reasoning
  JoeLambda father Bob.
  => Bob a Male. JoeLambda a Person.



Family ontology

```
@keywords a.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://dig.csail.mit.edu/2010/LinkedData/testdata/family#> .

Mammal a rdfs:Class.
Person rdfs:subClassOf Mammal.
Male rdfs:subClassOf  Person.
Female rdfs:subClassOf  Person.

parent a rdf:Property;
    rdfs:range Person;
    rdfs:domain Person.
mother rdfs:subPropertyOf parent;
    rdfs:range Female;
    rdfs:domain Person.
father rdfs:subPropertyOf parent;
    rdfs:range Male;
    rdfs:domain Person.
```

Exercise 1

- Define sibling property and brother, sister property based on sibling

Exercise 1

- Define sibling property and brother, sister property based on sibling

```
sibling a rdf:Property.
brother rdfs:subPropertyOf sibling;
        rdfs:range Male;
        rdfs:domain Person.
sister rdfs:subPropertyOf sibling;
        rdfs:range Female;
        rdfs:domain Person.
```

Exercise 2

- Given the following, can you infer that every sister is Female ? And how ?

```
sibling rdf:type rdf:Property.
sister rdfs:subPropertyOf sibling;
    rdfs:range Person;
    rdfs:domain Female.
```

Exercise 2

- Given the following, can you infer that every sister is Female ? And how ?

sibling rdf:type rdf:Property.

sister rdfs:subPropertyOf sibling;

    rdfs:range Person;

    rdfs:domain Female.

No

Exercise 2

- Given the following, can you infer that every sister is Female ? And how ?

sibling rdf:type rdf:Property.

sister rdfs:subPropertyOf sibling;
    rdfs:range Person;
    rdfs:domain Female.

No

sibling rdf:type rdf:Property.

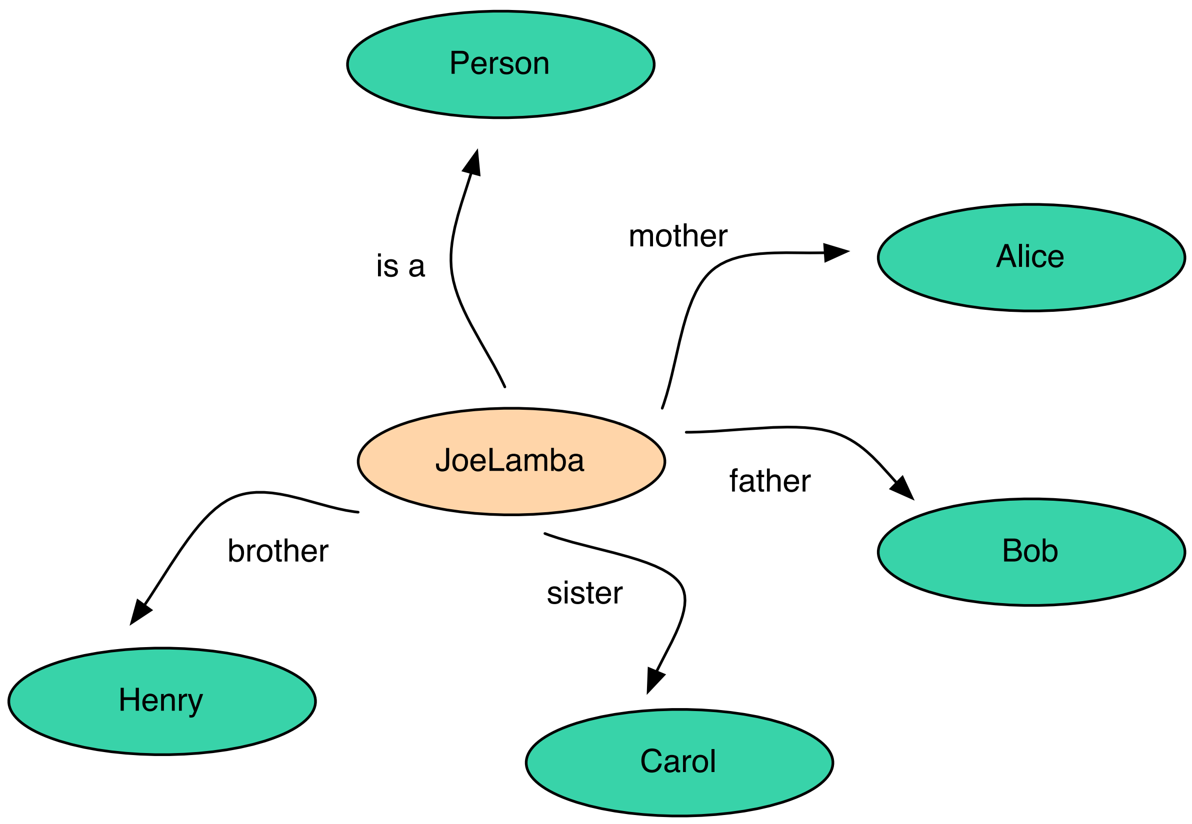
sister rdfs:subPropertyOf sibling;
    rdfs:range **Female**;
    rdfs:domain **Person**.

Exercise 3

- Define a instance of Person with mother, father, sister, and brother relationships

Exercise 3

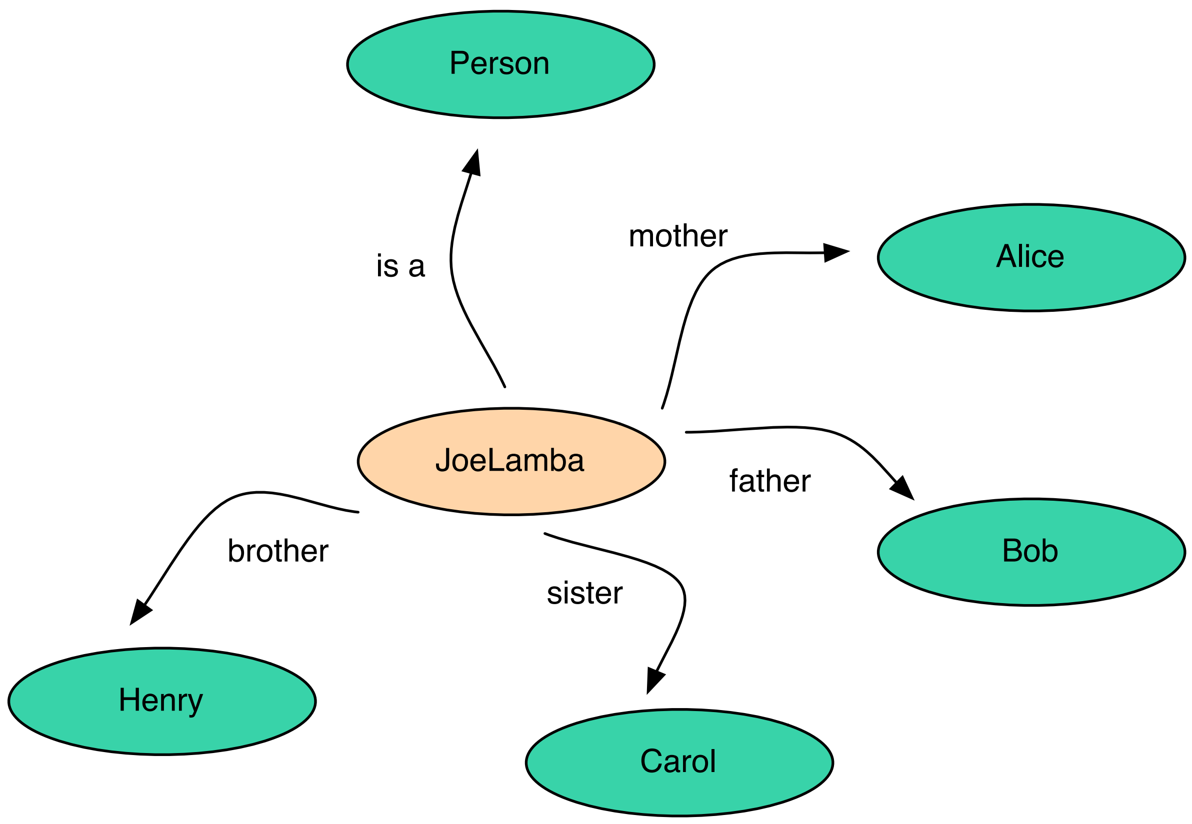
- Define a instance of Person with mother, father, sister, and brother relationships



Exercise 3

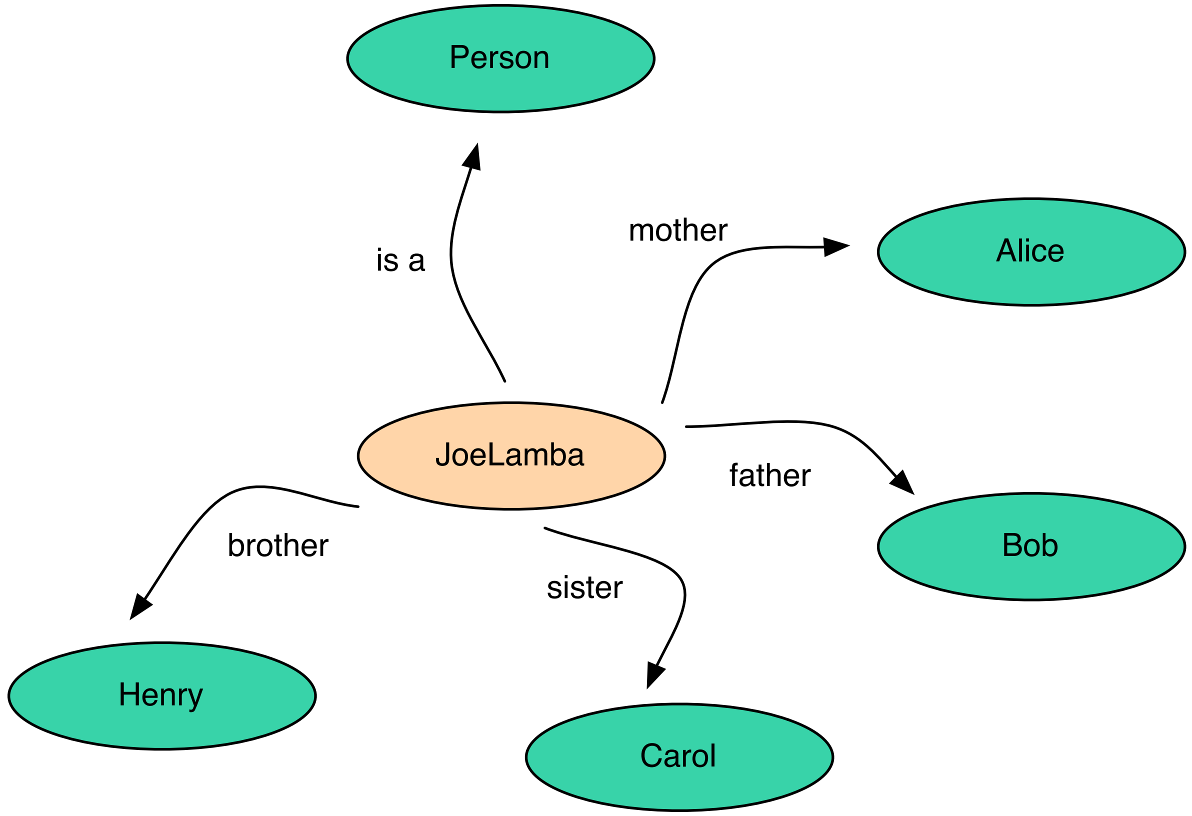
- Define a instance of Person with mother, father, sister, and brother relationships

JoeLambda a Person;
mother Alice;
father Bob;
sister Carol;
brother Henry.



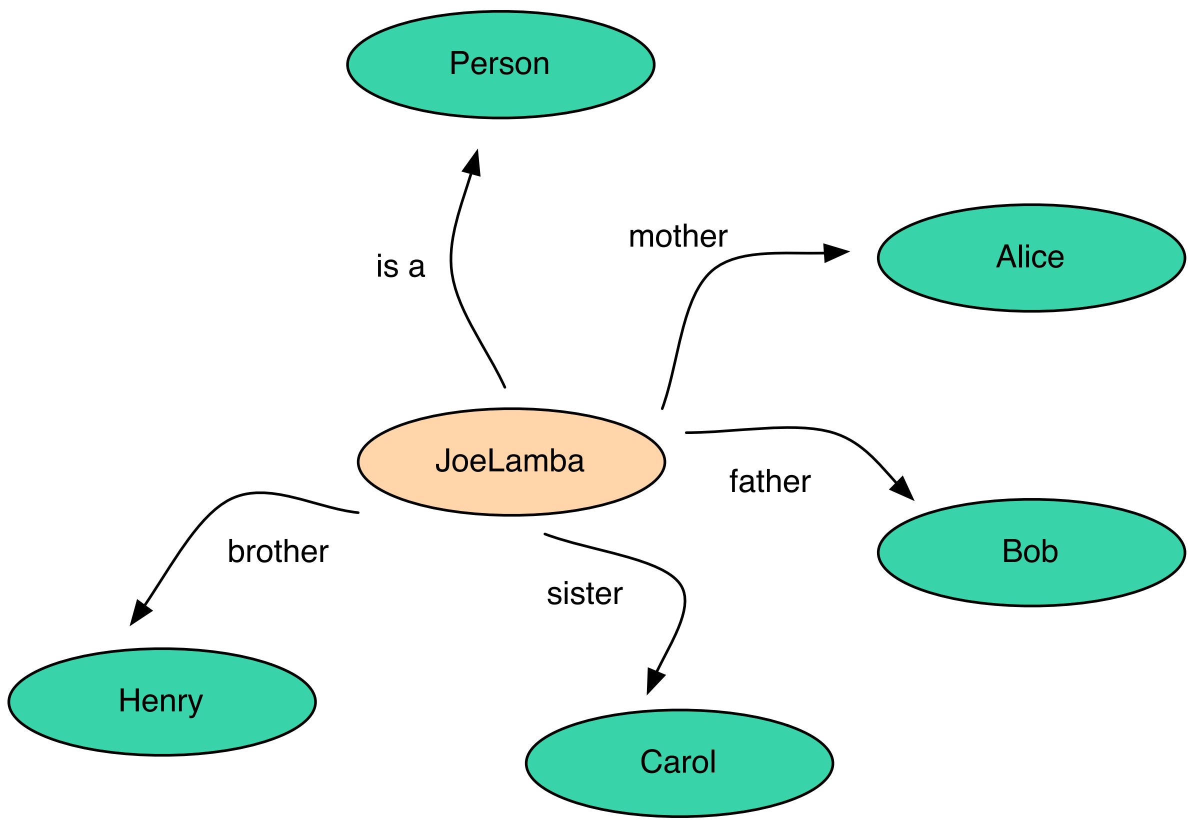
Exercise 4

- If you know about Mammal and Person classes, and sibling, brother, sister, parent, mother and father relationships, what are the RDFS inferences you can make from the following ?



Exercise 4

- If you know about Mammal and Person classes, and sibling, brother, sister, parent, mother and father relationships, what are the RDFS inferences you can make from the following ?



JoeLambda a Mammal.

Alice a Female, Person, Mammal.

Carol a Female, Person, Mammal.

Bob a Male, Person, Mammal.

Henry a Male, Person, Mammal.

OWL 1.0

- RDFS is a vocabulary for describing properties and classes
- OWL adds more vocabulary
  - relations between classes
  - property types (classes of properties)
  - characteristics of properties (properties of properties)
  - cardinality constraints (upper/lower limit on number of)
  - equality between classes and instances
  - enumerated classes

OWL

- @prefix owl: <http://www.w3.org/2002/07/owl#>.

- owl:Class rdfs:subClassOf rdfs:Class.

  Mammal a owl:Class.

  Person rdfs:subClassOf Mammal.

OWL

- Relations between classes
  - equivalentClass, intersectionOf, unionOf, complementOf

```
Male a owl:Class.
Female a owl:Class.

Person  a owl:Class;
        owl:unionOf  ( Male Female ) .

Female owl:complementOf Male.
```

Exercise 5

- If Parent is a subclass of Person, define Father and Mother classes. Hint: use intersectionOf

```
Male a owl:Class.
Female a owl:Class.

Person  a owl:Class;
        owl:unionOf  ( Male
Female ) .

Female owl:complementOf Male.

Parent rdfs:subClassOf Person.
```

Exercise 5

- If Parent is a subclass of Person, define Father and Mother classes. Hint: use intersectionOf

Male a owl:Class.
Female a owl:Class.

Person a owl:Class;
        owl:unionOf (Male
Female) .

Female owl:complementOf Male.

Parent rdfs:subClassOf Person.

Father a owl:Class;
        owl:intersectionOf (Male Parent) .

Mother a owl:Class;
        owl:intersectionOf (Female Parent) .

Exercise 6

- Redefine mother and father properties using Mother and Father classes

```
Father  a owl:Class;
        owl:intersectionOf  ( Male Parent ) .
Mother  a owl:Class;
        owl:intersectionOf  ( Female Parent ) .

parent rdf:type rdf:Property;
       rdfs:range Parent;
       rdfs:domain Person.

father rdfs:subPropertyOf parent;
        rdfs:range Male;
        rdfs:domain Person.

mother rdfs:subPropertyOf parent;
        rdfs:range Female;
        rdfs:domain Person.
```

Exercise 6

- Redefine mother and father properties using Mother and Father classes

```
Father  a owl:Class;
        owl:intersectionOf  ( Male Parent ) .
Mother  a owl:Class;
        owl:intersectionOf  ( Female Parent ) .

parent rdf:type rdf:Property;
       rdfs:range Parent;
       rdfs:domain Person.

father rdfs:subPropertyOf parent;
        rdfs:range Male;
        rdfs:domain Person.

mother rdfs:subPropertyOf parent;
        rdfs:range Female;
        rdfs:domain Person.
```

```
father rdfs:subPropertyOf parent;
        rdfs:range Father;
        rdfs:domain Person.

mother rdfs:subPropertyOf parent;
        rdfs:range Mother;
        rdfs:domain Person.
```

OWL

- Property types
  - ObjectProperty: relations between instances of two classes

    mother a owl:ObjectProperty; rdfs:domain Person; rdfs:range Parent.

  - DatatypeProperty: relations between instances of classes and RDF literals and XML Schema datatypes

    ```
    @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
    dateofbirth a owl:DatatypeProperty;
                rdfs:range xsd:date;
                rdfs:domain Person.
    ```

Exercise 7

- What are some DatatypeProperty we can add to family ontology ?

Exercise 7

- What are some DatatypeProperty we can add to family ontology ?

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

age a owl:DatatypeProperty;
        rdfs:range xsd:integer;
        rdfs:domain Person.

JoeLambda age "8"^^xsd:integer.
```

OWL

OWL

- Characteristics of properties

OWL

- Characteristics of properties
  - TransitiveProperty: P(x,y) and P(y,z) implies P(x,z)

    relatedTo a owl:TransitiveProperty.

    Joe relatedTo Henry. Henry relatedTo Ann. => Joe relatedTo Ann.

OWL

- Characteristics of properties
  - TransitiveProperty: P(x,y) and P(y,z) implies P(x,z)

    relatedTo a owl:TransitiveProperty.

    Joe relatedTo Henry. Henry relatedTo Ann. => Joe relatedTo Ann.

  - SymmetricProperty: P(x,y) iff P(y,x)

    sibling a owl:SymmetricProperty.

    JoeLambda sibling Carol => Carol sibling JoeLambda.

OWL

OWL

- Characteristics of properties

OWL

- Characteristics of properties
  - FunctionalProperty: P(x,y) and P(x,z) implies y = z
    birthmother a owl:FunctionalProperty.
    JoeLambda birthmother Alice. JoeLambda birthmother Alicia. => Alice = Alicia

OWL

- Characteristics of properties
  - FunctionalProperty: P(x,y) and P(x,z) implies y = z
    birthmother a owl:FunctionalProperty.
    JoeLambda birthmother Alice. JoeLambda birthmother Alicia. => Alice = Alicia
  - inverseOf:P1(x,y) iff P2(y,x)
    wife owl:inverseOf husband.
    JoeLambda wife Amy => Amy husband JoeLambda

OWL

- Characteristics of properties
  - FunctionalProperty: P(x,y) and P(x,z) implies y = z
    birthmother a owl:FunctionalProperty.
    JoeLambda birthmother Alice. JoeLambda birthmother Alicia. => Alice = Alicia
  - inverseOf:P1(x,y) iff P2(y,x)
    wife owl:inverseOf husband.
    JoeLambda wife Amy => Amy husband JoeLambda
  - InverseFunctionalProperty: P(y,x) and P(z,x) implies y = z
    email a owl:InverseFunctionalProperty.
    JoeLambda email abc@ex.com. JosephLamba email abc@ex.com => JoeLambda = JosephLamba

Exercise 8

- Define a functional property and an inverseOf property

- TransitiveProperty: P(x,y) and P(y,z) implies P(x,z)
  relatedTo a owl:TransitiveProperty.

- SymmetricProperty: P(x,y) iff P(y,x)
  sibling a owl:SymmetricProperty.
  JoeLambda sibling Carol => Carol sibling JoeLambda.

- FunctionalProperty: P(x,y) and P(x,z) implies y = z
  birthmother a owl:FunctionalProperty.
  JoeLambda birthmother Alice. JoeLambda birthmother Alicia. => Alice = Alicia

- inverseOf:P1(x,y) iff P2(y,x)
  wife owl:inverseOf husband.
  JoeLambda wife Amy => Amy husband JoeLambda

- InverseFunctionalProperty: P(y,x) and P(z,x) implies y = z
  email a owl:InverseFunctionalProperty.
  JoeLambda email abc@ex.com.
  JosephLamba email abc@ex.com =>
  JoeLambda = JosephLambs

Exercise 8

- Define a functional property and an inverseOf property

  - TransitiveProperty: P(x,y) and P(y,z) implies P(x,z)
    relatedTo a owl:TransitiveProperty.
  - SymmetricProperty: P(x,y) iff P(y,x)
    sibling a owl:SymmetricProperty.
    JoeLambda sibling Carol => Carol sibling JoeLambda.
  - FunctionalProperty: P(x,y) and P(x,z) implies y = z
    birthmother a owl:FunctionalProperty.
    JoeLambda birthmother Alice. JoeLambda birthmother Alicia. => Alice = Alicia
  - inverseOf:P1(x,y) iff P2(y,x)
    wife owl:inverseOf husband.
    JoeLambda wife Amy => Amy husband JoeLambda
  - InverseFunctionalProperty: P(y,x) and P(z,x) implies y = z
    email a owl:InverseFunctionalProperty.
    JoeLambda email abc@ex.com. JosephLamba email abc@ex.com => JoeLambda = JosephLambs

spouse a owl:FunctionalProperty.

Alice spouse Bob. Alice spouse Bobby. => Bob = Bobby

child owl:inverseOf parent.

Alice child JoeLambda => JoeLambda parent Alice

Exercise 9

- What can you infer if

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

Exercise 9

- What can you infer if

  spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

  => Bobby spouse Ann. Henry spouse Annie.

  niece owl:inverseOf aunt. Alice niece Ann.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

niece owl:inverseOf aunt. Alice niece Ann.

=> Ann aunt Alice.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

niece owl:inverseOf aunt. Alice niece Ann.

=> Ann aunt Alice.

friend a owl:TransitiveProperty. Joe friend Tim. Harry friend Joe.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

niece owl:inverseOf aunt. Alice niece Ann.

=> Ann aunt Alice.

friend a owl:TransitiveProperty. Joe friend Tim. Harry friend Joe.

=> Harry friend Tim.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

niece owl:inverseOf aunt. Alice niece Ann.

=> Ann aunt Alice.

friend a owl:TransitiveProperty. Joe friend Tim. Harry friend Joe.

=> Harry friend Tim.

friend a owl:TransitiveProperty. Ann friend Mary. Harry friend Mary.

Exercise 9

- What can you infer if

spouse a owl:SymmetricProperty. Ann spouse Bobby. Annie spouse Henry.

=> Bobby spouse Ann. Henry spouse Annie.

niece owl:inverseOf aunt. Alice niece Ann.

=> Ann aunt Alice.

friend a owl:TransitiveProperty. Joe friend Tim. Harry friend Joe.

=> Harry friend Tim.

friend a owl:TransitiveProperty. Ann friend Mary. Harry friend Mary.

=> No additional inferences

OWL

- Property restrictions
  - Allows you to define an anonymous class of all individuals that satisfy the restriction
  - allValuesFrom
    - requires that for every instance of the class that has instances of the specified property, the values of the property are all members of the class indicated by the owl:allValuesFrom clause
    - does not require the class to have the property, but if it does, all properties must be in the class specified.

```
PersonsWithOnlyDaugthers rdfs:subclassOf Person,
  [   a owl:Restriction;
          owl:onProperty child;
           owl:allValuesFrom Female ].
```

```
PersonsWithOnlySons rdfs:subclassOf Person,
  [   a owl:Restriction;
          owl:onProperty child;
           owl:allValuesFrom Male ].
```

OWL

- Property restrictions
- someValuesFrom
  - similar to allValuesFrom but mean that at least one of the keyword of a SemWebPaper must be a SemWebKeyword
  - it requires that there be at least one property that is in the class specified, but there may be properties that aren't.
  - allValuesFrom versus someValuesFrom - universal versus existential quantification

```
PersonsWithAtLeastOneDaugther rdfs:subclassOf Person,
[   a owl:Restriction;
      owl:onProperty child;
      owl:someValuesFrom Female ].

SemWebPaper rdfs:subclassOf Paper,
[   a owl:Restriction;
      owl:onProperty keyword;
      owl:someValuesFrom SemWebKeyword ].
```

OWL

- Property restrictions

- hasValue allows us to specify classes based on the existence of particular property values

```
JoesSiblings rdfs:subclassOf Person,
  [   a owl:Restriction;
          owl:onProperty brother;
           owl:hasValue JoeLambda ].
```

- cardinality constraints
  owl:cardinality specifies exactly the number of elements in a relation
  owl:maxCardinality can be used to specify an upper bound
  owl:minCardinality can be used to specify a lower bound

```
PersonsWithTwoParents rdfs:subclassOf Person,
  [   a owl:Restriction;
           owl:cardinality "2"^^xsd:nonNegativeInteger;
           owl:onProperty parent ] .
```

Exercise 9

- Use property restrictions (allValuesFrom, someValuesFrom, hasValue or cardinality constraints) to define a class of Joe's brothers who have at least one child

Exercise 9

- Use property restrictions (allValuesFrom, someValuesFrom, hasValue or cardinality constraints) to define a class of Joe's brothers who have at least one child

```
JoesSiblings rdfs:subclassOf Person,
  [   a owl:Restriction;
        owl:onProperty brother;
        owl:hasValue JoeLambda ].

JoesBrothersWithAtLeastOneChild  rdfs:subClassOf  JoesSiblings, Male,
   [   a owl:Restriction;
         owl:minCardinality "1"^^xsd:nonNegativeInteger;
         owl:onProperty child ] .
```

OWL

- Equivalence between Classes and Properties: equivalentClass, equivalentProperty
- Identity between Individuals
  JoeLambda owl:sameAs JosephLamba.
- Different Individuals: differentFrom, AllDifferent
- Enumerated Classes: List instances that belong to the class

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Person rdf:about="#Joe"/>
    <Person rdf:about="#Alice"/>
    <Person rdf:about="#Henry"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White"/>
    <owl:Thing rdf:about="#Rose"/>
    <owl:Thing rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>
```

Summary

- Ontology
  - Formal specifications of the terms in a domain and the relations among them
  - Advantages: common understanding of domain, helps in reuse and making assumptions explicit
- W3C ontology languages
  - RDFS
    - concepts for defining classes, properties and their hierarchies
  - OWL
    - extends expressivity of RDFS
    - class relations, property types, characteristics of properties, property restrictions and equivalence relations, etc.

References

- Ontology Development 101, http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
- RDFS, http://www.w3.org/TR/rdf-schema/
- Description Logic, http://en.wikipedia.org/wiki/Description_logic
- OWL 1, http://www.w3.org/TR/2004/REC-owl-guide-20040210/
- OWL 2, http://www.w3.org/TR/owl2-overview/