ISIT315
Week 4

Web Ontology Langauge (OWL)

https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

Limitations of expressive power of RDF schema

- RDF/RDFS
  - organise vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classess
- missing
  - local scope of properties
    - e.g. `rdfs:range` defines the range of a property say `eats` for all classes, but RDF schema cannot declare range restrictions that apply to some classes only, e.g. we cannot say cows eat only plants while other animals may eat meat

Limitations of expressive power of RDF schema

- disjointness of classess
  - e.g. `male` and `female` are disjoint
  - but in RDF schema, we can only state subclass relationship, e.g. `female` is a subclass of `person`
- boolean combinations of classes
  - sometimes we wish to build new classes by combining other classes using union ($\cup$), intersection ($\cap$), complement (\).
    - e.g. we wish to define the class person to be disjoint union of classes `male` and `female`. RDF schema does not allow.

Limitations of expressive power of RDF schema

- cardinality restrictions
  - to place restrictions on how many distinct values a property may or may not take
    - e.g. a person has exactly two parents, a course is taught by at least one lecturer
    - not possible to express in RDF schema
- special characteristics of properties
  - RDF schema cannot allow properties such as *inverse* (`eats` and `is eaten by`) to express

OWL 2

- OWL = Web Ontology Language
  - is a language for expressing *ontologies*
  - *An* <u>*ontology*</u> *provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base.*
  - Ontologies are the backbone of the Semantic Web.
  - They provide the knowledge that is required for semantic applications of all kinds.

Notes

- OWL 2 is not a programming language:
  - It is *declarative*, i.e. it describes a state of affairs in a logical way
    - is a knowledge representation language designed to formulate, exchange and reason with knowledge about a domain of interest
  - Then reasoners can be used to infer further information about that state of affairs.
  - How these inferences are realized algorithmically is not part of the OWL document but depends on the specific implementations.

Requirements for ontology language

- Allow users to write explicit, formal conceptualisations of domain models
- Well-defined syntax
- Efficient reasoning support
- Formal semantics
- Sufficient expressive power
- Convenience of expression

Formal semantics

- Describes the meaning of knowledge precisely
  - Precisely: does not open to different interpretations by different people/machine
- Allow people to reason about the knowledge
  - Class membership
    - If x is an instance of a class C, and C is a subclass of D, then we infer x is an instance of D
  - Equivalence of class
    - If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C

Reasoning about knowledge

- Consistency
  - Suppose we have declared *x* to be an instance of class A and A is a subclass of B $\cap$ C, A is a subclass of D and B and D are disjoint, then we have inconsistency because A should be empty but has an instance *x*. This is an indication of error
- Classification
  - If we have declared that certain property-value pairs are a sufficient condition for memberships in a class A, then if an individual x satisfies such conditions, we can conclude that *x* must be an instance of A

Three sublanguages of OWL

- OWL Full
- OWL DL (Descriptive Logic)
- OWL Lite

OWL Lite

- Supports those users primarily needing a classification hierarchy and simple constraints.
- Thesauri and other taxonomies.

OWL DL

- Supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- So named due to its correspondence with *description logics*

OWL Full

- Maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.
- It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

The following set of relations hold; but not their inverses

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

List of OWL Lite language constructs

- see http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.1

OWL Lite

- Class
- rdfs: subClassOf
- rdf: Property
- rdfs: subPropertyOf
- rdfs: domain
- rdfs: range
- Individual

OWL Lite Equality and Inequality

- equivalentClass
- equivalentProperty
- sameAs
- differentAs
- AllDifferent
- See https://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.1

equivalentClass

- Two classes may be stated to be equivalent.
- Equivalent classes have the same instances.
- Equality can be used to create synonymous classes.
- Example
  - Car can be stated to be *equivalentClass* to Automobile.
  - From this a reasoner can deduce that any individual that is an instance of Car is also an instance of Automobile and vice versa.

equivalentProperty

- Two properties may be stated to be equivalent.
- Equivalent properties relate one individual to the same set of other individuals.
- Equality may be used to create synonymous properties.
- Example
  - hasLeader may be stated to be the *equivalentProperty* to hasHead.
  - From this a reasoner can deduce that if X is related to Y by the property hasLeader, X is also related to Y by the property hasHead and vice versa.
  - A reasoner can also deduce that hasLeader is a subproperty of hasHead and hasHead is a subProperty of hasLeader.

sameAs

- Two individuals may be stated to be the same.
- Example:
  - The individual Deborah may be stated to be the same individual as DeborahMcGuinness.

differentFrom

- An individual may be stated to be different from other individuals.
- Example
  - the individual Frank may be stated to be different from the individuals Deborah and Jim.
  - Thus, if the individuals Frank and Deborah are both values for a property that is stated to be functional (thus the property has at most one value), then there is a contradiction.

AllDifferent

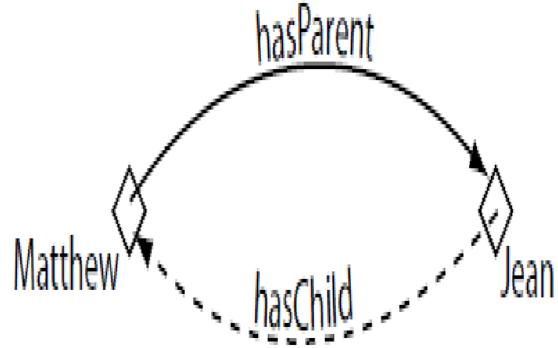
- A number of individuals may be stated to be mutually distinct.
- Example,
  - Frank, Deborah, and Jim could be stated to be mutually distinct using the AllDifferent construct.
  - The AllDifferent construct is particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects.
  -

Property characteristics

- *ObjectProperty*
- *DatatypeProperty*
- *inverseOf*
- *TransitiveProperty*
- *SymmetricProperty*
- *FunctionalProperty*
- *InverseFunctionalProperty*

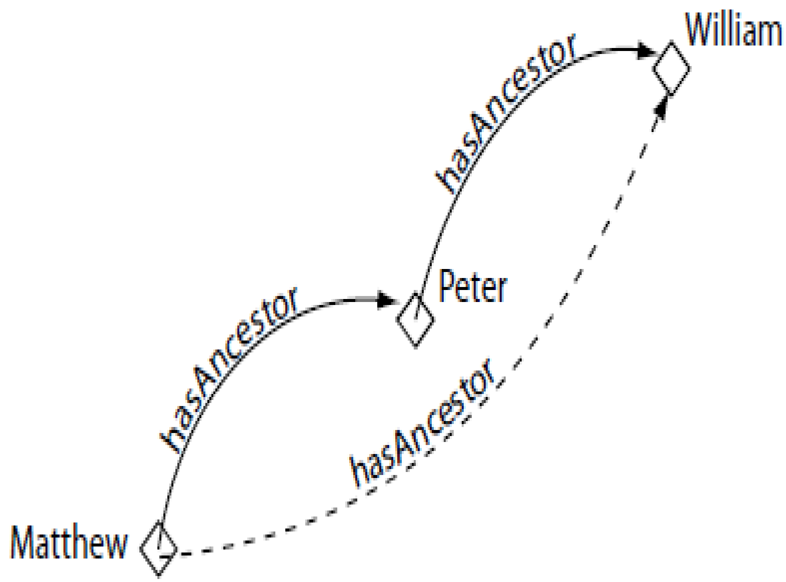
inverseOf

- If some property links individual *a* to individual *b*, then its <u>inverse</u> property will link individual *b* to individual *a*
- Example,
  if hasChild is the inverse of hasParent
    Matthew hasParent Jean
    then a reasoner can deduce that
    Jean hasChild Matthew.



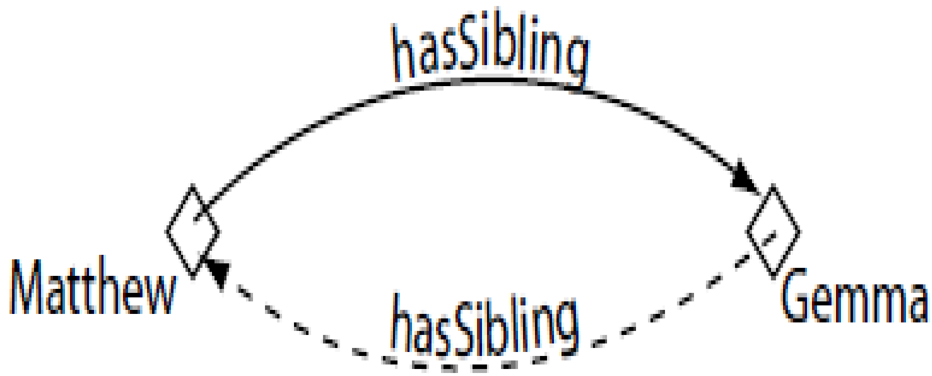
TransitiveProperty

If a property P is transitive, and the property relates to individual *a* to individual *b* and also individual *b* to individual *c*, then we infer that *a* is related to *c* via property P



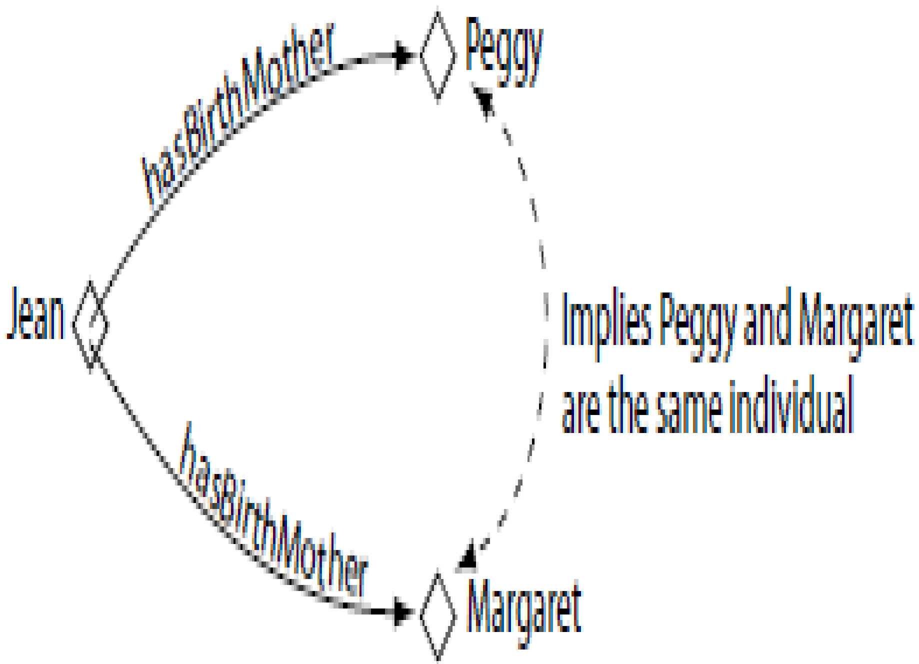
SymmetricProperty

If a property P is symmetric and the property relates individual a to individual b then individual b is also related to individual a via property P



FunctionalProperty

- for a given individual, there is <u>at most one individual</u> that is related to the individual via the property
- also known as <u>single valued (unique)</u> property



InverseFunctionalProperty

If a properties is <u>inverse functional</u> it means the inverse property is functional

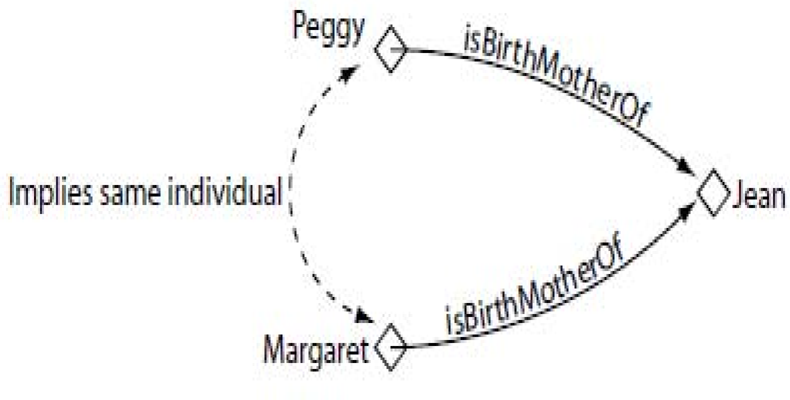


Figure 4.19: An Example Of An Inverse Functional Property: isBirthMotherOf

OWL Lite Property Restrictions –how many values can be used.

- allValuesFrom
  - this property on this particular class has a local range restriction associated with it.
- someValuesFrom
  - A particular class may have a restriction on a property that at least one value for that property is of a certain type.

OWL Lite Restricted Cardinality – concerning cardinalities of value 0 or 1

- minCardinality
  - *minCardinality* = 1 then any instance of that class will be related to at least one individual by that property.
  - *minCardinality* = 0, then the property is optional with respect to a class.
- maxCardinality
  - *maxinCardinality* = 1 then any instance of that class will be related to at most one individual by that property.
  - *maxCardinality* = 0, then the property is no value with respect to that property.
- Cardinality
  - Provided as convenience when it is useful to state a property on a class has both *minCardinality* 0 and *maxCardinality* 0 or both *minCardinality* 1 and *maxCardinality* 1

OWL Lite Class Intersection

- intersectionOf
  - intersections of named classes and restrictions.
- Example
  - the class EmployedPerson can be described as the *intersectionOf* Person and EmployedThings
  - From this a reasoner may deduce that any particular EmployedPerson has at least one employer.

List of OWL DL and Full language constructs

- see http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.2