OWL
Web Ontology Language

Erdoğan Doğdu

Notes from "Semantic Web for the Working Ontologist" Book

Property characteristics

- Inverse
- Symmetric
- Transitive

Inverse

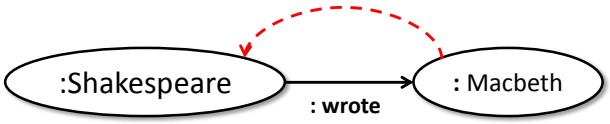
- owl:inverseOf
  - Specifies a property is the inverse of another

:wrote **owl:inverseOf** :writtenBy

:Shakespeare :wrote :Macbeth

→ (inference)

: Macbeth :writtenBy :Shakespeare



Inverse inferencing rule

- By SPARQL CONSTRUCT

```
CONSTRUCT {?y ?q ?x}
WHERE { ?p owl:inverseOf ?q .
        ?x ?p ?y . }
```

Example

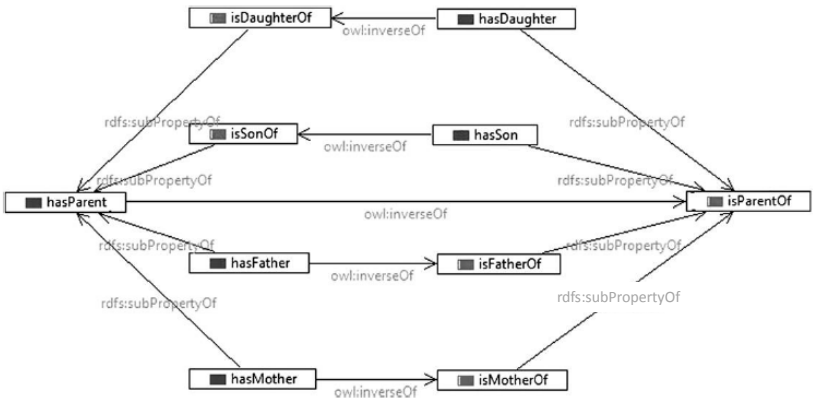


FIGURE 8.1

Display of family relationships, and how they are connected. The figure shows only the `subPropertyOf` relationships, not the `inverseOf` relationships.

Example

:signedTo owl:inverseOf :signedOut.
:signedOut rdfs:subPropertyOf :hasPossession.
:borrows rdfs:subPropertyOf :hasPossession.
:Amit :borrows :MobyDick.
:Marie :borrows :Orlando.
:LeavesOfGrass :signedTo :Jim.
:WutheringHeights :signedTo :Yoshi.
→ (inferred)

:Jim :signedOut :LeavesOfGrass.
:Yoshi :signedOut :WutheringHeights.
(Using :subPropertyOf)
:Amit :hasPossession :MobyDick.
:Marie :hasPossession :Orlando.
:Jim :hasPossession :LeavesOfGrass.
:Yoshi :hasPossession :WutheringHeights.

Example

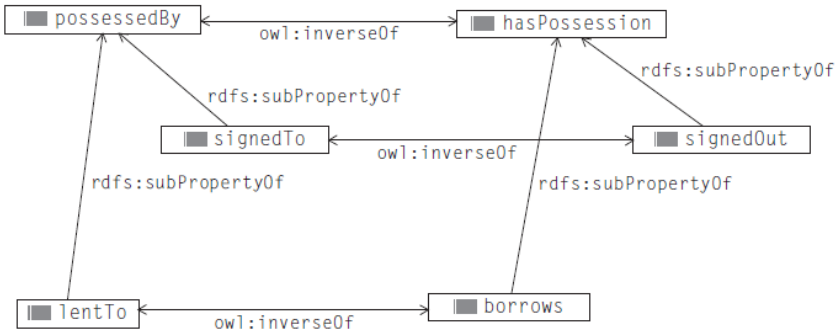


FIGURE 8.2

Systematic combination of `inverseOf` and `subPropertyOf`.

Extending the modelling language

:Food myowl:superClassOf :BakedGood;
myowl:superClassOf :Confectionary;
myowl:superClassOf :PackagedFood;
myowl:superClassOf :PreparedFood;
myowl:superClassOf :ProcessedFood.

- *What is myowl:superClassOf?*

myowl:superClassOf **owl:inverseOf** rdfs:subClassOf.

- *Even can say this…*

rdfs:subClassOf owl:inverseOf **rdfs:superClassOf**.
(AAA rule applies, *but not recommended!*)

More on inverse

bio:AnneHathaway bio:married lit:Shakespeare.

- SELECT ?who
  WHERE {?lit:Shakespeare bio:married ?who}
- Answer: ?
- Solution:
  bio:married owl:inverseOf bio:married.
- Self-inverse? Very common, so:
  – owl:SymmetricProperty

Symmetric Property

:p rdf:type owl:SymmetricProperty.

In SPARQL:

CONSTRUCT {?p owl:inverseOf ?p. }

WHERE {?p a owl:SymmetricProperty . }

Ex:

bio:married rdf:type owl:SymmetricProperty.

More on inverse

:possessedBy owl:inverseOf :hasPossession.
:signedTo owl:inverseOf :signedOut.
:lentTo owl:inverseOf :borrows.
→
:hasPossession owl:inverseOf :possessedBy.
:signedOut owl:inverseOf :signedTo.
:borrows owl:inverseOf :lentTo.

- *How?*
  - owl:inverseOf owl:inverseOf owl:inverseOf.

  or
  - **owl:inverseOf** rdf:type **owl:SymmetricProperty**.

Transitivity

- owl:TransitiveProperty
- :p rdf:type owl:TransitiveProperty.
- In SPARQL

```
CONSTRUCT { ?x ?p ?z .}
WHERE  {    ?x ?p ?y .
            ?y ?p ?z .
            ?p a owl:TransitiveProperty . }
```

- Ex: *geographical containment (if Osaka is in Japan, and Japan is in Asia, then Osaka is in Asia)*

Example

:hasParent rdfs:subPropertyOf :hasAncestor.
:hasAncestor rdf:type owl:TransitiveProperty.
:Alexia :hasParent :WillemAlexander.
:WillemAlexander :hasParent :Beatrix.
:Beatrix :hasParent :Wilhelmina.
→

:Beatrix :hasAncestor :Wilhelmina.
:Alexia :hasAncestor :WillemAlexander.
:WillemAlexander :hasAncestor :Beatrix.
:Alexia :hasAncestor :Beatrix.
:WillemAlexander :hasAncestor :Wilhelmina.
:Alexia :hasAncestor :Wilhelmina.

Note: hasAncestor is transitive, but hasParent is not!

Equivalence

- Equivalent classes (in RDFS)

  :Analyst rdf:subClassOf :Researcher.

  :Researcher rdf:subClassOf :Analyst.

  – *every Analyst is a Researcher and every Researcher is an Analyst*
- Equivalent classes (in OWL)

  :Analyst owl:equivalentClass :Researcher.

Equivalent classes

- *In SPARQL*

```
CONSTRUCT {?r rdf:type ?b .}
WHERE { ?a owl:equivalentClass ?b .
        ?r rdf:type ?a . }
```

  - and

```
CONSTRUCT {?r rdf:type ?a .}
WHERE { ?a owl:equivalentClass ?b .
        ?r rdf:type ?b . }
```

- *Or following is sufficient:*

  owl:equivalentClass rdf:type owl:SymmetricProperty.

  owl:equivalentClass rdfs:subPropertyOf rdfs:subClassOf.

Equivalent properties

- Equivalent properties (in RDFS)

  :borrows rdfs:subPropertyOf :checkedOut.

  :checkedOut rdfs:subPropertyOf :borrows.

- Equivalent properties (in OWL)

  :borrows owl:equivalentProperty :checkedOut.

- In SPARQL

```
CONSTRUCT {?a :checkedOut ?b . }
WHERE { ?a :borrows ?b . }
```

Equivalent properties

- Following are sufficient:

owl:equivalentProperty rdfs:subPropertyOf rdfs:subPropertyOf.
owl:equivalentProperty rdf:type owl:SymmetricProperty.

Same individuals

- Individual
  - Member of a class
- Ex:
  - CONSTRUCT {lit:Shakespeare ?p ?o . }
    WHERE { spr:WilliamShakespeare ?p ?o . }

  Similarly,
  - CONSTRUCT {?s ?p lit:Shakespeare. }
    WHERE { ?s ?p spr:WilliamShakespeare. }

Same individuals

- In SPARQL (3 rules)
  - CONSTRUCT {?s ?p ?x. }
    WHERE { ?s ?p ?y.
    ?x owl:sameAs ?y . }
  - CONSTRUCT {?x ?p ?o. }
    WHERE { ?y ?p ?o .
    ?x owl:sameAs ?y . }
  - CONSTRUCT {?s ?x ?o. }
    WHERE { ?s ?y ?o .
    ?x owl:sameAs ?y . }
- Or

  owl:sameAs rdf:type owl:SymmetricProperty.

Functional properties

- owl:FunctionalProperty
- A property that **can only take one value** for any particular individual

CONSTRUCT {?a ***owl:sameAs*** ?b . }
WHERE { ?p rdf:type **owl:FunctionalProperty** .
?x ?p ?a .
?x ?p ?b . }

Example

math:hasSquare rdf:type owl:FunctionalProperty.

:x math:hasSquare :A.

:x math:hasSquare :B.

→

:A owl:sameAs :B.

Example

lit:Shakespeare fam:hasFather bio:JohannesShakspere.

lit:Shakespeare fam:hasFather bio:JohnShakespeare.

fam:hasFather rdf:type owl:FunctionalProperty.

→

bio:JohannesShakspere owl:sameAs bio:JohnShakespeare.

Inverse functional properties

- owl:InverseFunctionalProperty
- the inverse of an owl:Functional Property
  - math:hasSquare: *functional property*,
  - math:hasSquareRoot: *inverse func. prop.*
- In SPARQL

```
CONSTRUCT { ?a owl:sameAs ?b . }
WHERE {   ?p rdf:type owl:InverseFunctionalProperty .
          ?a ?p ?x .
          ?b ?p ?x . }
```

Combining functional and inverse functional properties

- A single property can be both an owl:FunctionalProperty and an owl:InverseFunctionalProperty
  - one-to-one property
- Ex:

```
:hasIdentityNo rdfs:domain :Student.
:hasIdentityNo rdfs:range xsd:Integer.
:hasIdentityNo rdf:type owl:FunctionalProperty.
:hasIdentityNo rdf:type owl:InverseFunctionalProperty.
```

Functional vs. Inverse Func.

- Functional Only— **hasMother**
  - Some has one mother, but some people can have the same mother
- Inverse Functional Only— **hasDiary**
  - A person may have many diaries, each is authored by one person only.
- Both Functional and Inverse Functional—**taxID**
  - exactly one taxID for each person and exactly one person per taxID

Property types

- **owl:DatatypeProperty**
  - Property can have data value as object.
  - :std1 :hasIDno “123456789”
- **owl:ObjectProperty**
  - Property can have resource as object.
  - :std1 :takesClass :bil546