Syntax highlighting for OWL syntaxes

Eugeniu Costetchi

06/05/2020

Contents

1 Introduction **1**

2 Languages **2**
2.1 SPARQL 2
2.2 Functional-Style syntax 2
2.3 Manchester syntax 3
2.4 Turtle 4

3 License and contributions **5**

4 Long Examples **5**
4.1 SPARQL 5
4.2 Functional-Style syntax 6
4.3 Manchester syntax 11
4.4 Turtle 15

Abstract

Listings-OWL is a `listings` extension to provide syntax highlighting for SPARQL 1.1 and the most in use OWL syntaxes. This library implements Turtle, Manchester, and Functional-Style OWL syntaxes.

1 Introduction

This library implements syntax highlighting for the most in use serialisations for OWL language.

To use the package download it first into the document folder and refer to it in the header as follows: `\usepackage{listings-owl}`

Then create your listing and indicate the language for highlighting. The possible variants offered by this library are: `SPARQL1.1`, `Turtle`, `SPARQL1.1`, `Functional-style` and `Manchester-syntax`.

2 Languages

This section demonstrates usage of the four languages each in a separate subsection. Each subsection comprises of a small example demonstrating the syntax highlight followed by the LaTeXcode to generate it.

2.1 SPARQL

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE {
   ?x dc:title ?title
   FILTER regex(?title, "^SPARQL")
}
```

```
\begin{lstlisting}[language=SPARQL1.1]
   PREFIX dc:  <http://purl.org/dc/elements/1.1/>
   SELECT ?title
   WHERE
   {
      ?x dc:title ?title
      FILTER regex(?title, "^SPARQL")
   }
\end{lstlisting}
```

2.2 Functional-Style syntax

```
Prefix(:=<http://example.com/owl/families/>)
Ontology(<http://example.com/owl/families>
Import( <http://example.org/otherOntologies/families.owl> )

Declaration( Class( :Teenager ) )

SubClassOf( :Teenager
   DataSomeValuesFrom( :hasAge
      DatatypeRestriction( xsd:integer
         xsd:minExclusive "12"^^xsd:integer
         xsd:maxInclusive "19"^^xsd:integer
      )
   )
)
```

```
\begin{lstlisting}[language=Functional-style]
Prefix(:=<http://example.com/owl/families/>)
Ontology(<http://example.com/owl/families>
Import( <http://example.org/otherOntologies/families.owl> )
```

```
Declaration( Class( :Teenager ) )

SubClassOf( :Teenager
   DataSomeValuesFrom( :hasAge
      DatatypeRestriction( xsd:integer
         xsd:minExclusive "12"^^xsd:integer
         xsd:maxInclusive "19"^^xsd:integer
      )
   )
)
\end{lstlisting}
```

2.3 Manchester syntax

```
Ontology: <http://example.com/owl/families>

AnnotationProperty: rdfs:comment

Individual: John
    Types:
        Father,
        hasChild exactly 5 OWL:Thing,
        hasChild max 4 Parent,
        hasChild exactly 3 Parent,
        hasChild min 2 Parent
    Facts:
        hasWife  Mary,
        hasAge  51
    SameAs:
        Jack,
        otherOnt:JohnBrown
    DifferentFrom:
        Bill
```

```
\begin{lstlisting}[language=Manchester-syntax]
Ontology: <http://example.com/owl/families>

Individual: John
    Types:
        Father,
        hasChild exactly 5 OWL:Thing,
        hasChild max 4 Parent,
        hasChild exactly 3 Parent,
        hasChild min 2 Parent
    Facts:
        hasWife  Mary,
        hasAge  51
```

```
    SameAs:
        Jack,
        otherOnt:JohnBrown
    DifferentFrom:
        Bill
\end{lstlisting}
```

2.4 Turtle

```
@prefix : <http://example.com/owl/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<http://example.com/owl/families> rdf:type OWL:Ontology .

:minorAge rdf:type rdfs:Datatype .

:toddlerAge rdf:type rdfs:Datatype ;
    OWL:equivalentClass [
        rdf:type rdfs:Datatype ;
        OWL:oneOf [
            rdf:type rdf:List ;
            rdf:first 1 ;
            rdf:rest [ rdf:type rdf:List ;
                     rdf:first 2 ;
                     rdf:rest rdf:nil
            ]
        ]
    ] .
```

```
\begin{lstlisting}[language=Turtle]
@prefix : <http://example.com/owl/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<http://example.com/owl/families> rdf:type OWL:Ontology .

:minorAge rdf:type rdfs:Datatype .

:toddlerAge rdf:type rdfs:Datatype ;
    OWL:equivalentClass [
        rdf:type rdfs:Datatype ;
        OWL:oneOf [
            rdf:type rdf:List ;
            rdf:first 1 ;
            rdf:rest [ rdf:type rdf:List ;
                     rdf:first 2 ;
                     rdf:rest rdf:nil
            ]
```

```
        ]
    ] .
\end{lstlisting}
```

3 License and contributions

Listings-OWL may be distributed and/or modified under the conditions of the LaTeX Project Public License version 1.3 or later. The Current Maintainer of this work is Eugeniu Costetchi (costezki.eugen@gmail.com), who welcomes contributions to the package on GitHub.

The goal of this library is to provide a toolkit to draw quickly and efficiently beautiful OWL diagrams. Building such a library can be challenging for one pair of hands alone. If you have used this library yourself or have seen it used, we would greatly appreciate your opinion and feedback . Please open a Github issue or privately email the Current Maintainer with as much information as you can.

We also welcome technical feedback and bug reports in the form of GitHub Issues and pull requests.

4 Long Examples

This set of long examples are possibly useful and should be removed from this document. They are kept currently for consultation and bug-fixing due to their extensive coverage of syntactic features. Their code is not provided but can be consulted directly in the LaTeXcode this document.

4.1 SPARQL

```
PREFIX cdm: <http://publications.europa.eu/ontology/cdm#>
PREFIX cmr: <http://publications.europa.eu/ontology/cdm/cmr#>
PREFIX lg: <http://publications.europa.eu/resource/authority/language/>
PREFIX lbl: <http://publications.europa.eu/resource/authority/label-type/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ev: <http://eurovoc.europa.eu/>
PREFIX 2euvoc4: <http://publications.europa.eu/ontology/euvoc#>
PREFIX sk23os: <http://www.w3.org/2004/02/skos/core#>
prefix ds-type: <http://publications.europa.eu/resource/authority/dataset-type/>

using named graph <http://special.graph/1>

select ?workCellarId ?work_title ?conceptLabel ?language ?dateDocument
    ?dateCreation ?manifestationType ?mimeType (?itemCellarId as  ?download)
WHERE {
VALUES (?conceptLabel ?concept)  {
("endemic disease"  <http://eurovoc.europa.eu/1758>)
```

```
('epidemiology' <http://eurovoc.europa.eu/838>)
("epidemic" <http://eurovoc.europa.eu/837>)
('disease surveillance' <http://eurovoc.europa.eu/c_abfaf2ea>)
("health control"   <http://eurovoc.europa.eu/192>)
("public hygiene"   <http://eurovoc.europa.eu/3371>)
("respiratory disease"  <http://eurovoc.europa.eu/1756>)
("hospital infection"   <http://eurovoc.europa.eu/c_9b88f778>)
("patient safety"   <http://eurovoc.europa.eu/c_60d3928d>)
("patient's rights" <http://eurovoc.europa.eu/3370>)
("illness"  <http://eurovoc.europa.eu/1754>)
}
#?workCellarId owl:sameAs ?workPublicId .
?workCellarId a cdm:work ;
cdm:work_is_about_concept_eurovoc ?concept ;
cdm:work_title ?work_title .
OPTIONAL {?workCellarId cdm:work_date_document ?dateDocument . }
OPTIONAL {?workCellarId cdm:work_date_creation ?dateCreation . }

#?expressionCellarId owl:sameAs ?expressionPublicId .
?expressionCellarId cdm:expression_belongs_to_work ?workCellarId .
OPTIONAL { ?expressionCellarId cdm:expression_uses_language ?language . }



#?manifestationCellarId owl:sameAs ?manifestationPublicId .
?manifestationCellarId cdm:manifestation_manifests_expression ?expressionCellarId .
OPTIONAL { ?manifestationCellarId cdm:manifestation_type ?manifestationType . }


# item is resolved to the phisical file directly
#?itemCellarId owl:sameAs ?itemPublicId .
?itemCellarId cdm:item_belongs_to_manifestation ?manifestationCellarId .
OPTIONAL { ?itemCellarId cmr:manifestationMimeType ?mimeType . }
avg(some average)
bind(if(true, 'ssd',5454)) as ?newVariable
}
limit 1000
```

4.2 Functional-Style syntax

```
Prefix(:=<http://example.com/owl/families/>)
Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Ontology(<http://example.com/owl/families>
   Import( <http://example.org/otherOntologies/families.owl> )

   Declaration( NamedIndividual( :John ) )
   Declaration( NamedIndividual( :Mary ) )
   Declaration( NamedIndividual( :Jim ) )
   Declaration( NamedIndividual( :James ) )
   Declaration( NamedIndividual( :Jack ) )
   Declaration( NamedIndividual( :Bill ) )
```

```
Declaration( NamedIndividual( :Susan ) )
Declaration( Class( :Person ) )
AnnotationAssertion( rdfs:comment :Person "Represents the set of all people." )
Declaration( Class( :Woman ) )
Declaration( Class( :Parent ) )
Declaration( Class( :Father ) )
Declaration( Class( :Mother ) )
Declaration( Class( :SocialRole ) )
Declaration( Class( :Man ) )
Declaration( Class( :Teenager ) )
Declaration( Class( :ChildlessPerson ) )
Declaration( Class( :Human ) )
Declaration( Class( :Female ) )
Declaration( Class( :HappyPerson ) )
Declaration( Class( :JohnsChildren ) )
Declaration( Class( :NarcisticPerson ) )
Declaration( Class( :MyBirthdayGuests ) )
Declaration( Class( :Dead ) )
Declaration( Class( :Orphan ) )
Declaration( Class( :Adult ) )
Declaration( Class( :YoungChild ) )
Declaration( ObjectProperty( :hasWife ) )
Declaration( ObjectProperty( :hasChild ) )
Declaration( ObjectProperty( :hasDaughter ) )
Declaration( ObjectProperty( :loves ) )
Declaration( ObjectProperty( :hasSpouse ) )
Declaration( ObjectProperty( :hasGrandparent ) )
Declaration( ObjectProperty( :hasParent ) )
Declaration( ObjectProperty( :hasBrother ) )
Declaration( ObjectProperty( :hasUncle ) )
Declaration( ObjectProperty( :hasSon ) )
Declaration( ObjectProperty( :hasAncestor ) )
Declaration( ObjectProperty( :hasHusband ) )
Declaration( DataProperty( :hasAge ) )
Declaration( DataProperty( :hasSSN ) )
Declaration( Datatype( :personAge ) )
Declaration( Datatype( :minorAge ) )
Declaration( Datatype( :majorAge ) )
Declaration( Datatype( :toddlerAge ) )

SubObjectPropertyOf( :hasWife :hasSpouse )
SubObjectPropertyOf(
   ObjectPropertyChain( :hasParent :hasParent )
   :hasGrandparent
)
SubObjectPropertyOf(
   ObjectPropertyChain( :hasFather :hasBrother )
   :hasUncle
)
SubObjectPropertyOf(
   :hasFather
   :hasParent
)
```

```
EquivalentObjectProperties( :hasChild otherOnt:child )
InverseObjectProperties( :hasParent :hasChild )
EquivalentDataProperties( :hasAge otherOnt:age )
DisjointObjectProperties( :hasSon :hasDaughter )
ObjectPropertyDomain( :hasWife :Man )
ObjectPropertyRange( :hasWife :Woman )
DataPropertyDomain( :hasAge :Person )
DataPropertyRange( :hasAge xsd:nonNegativeInteger )

SymmetricObjectProperty( :hasSpouse )
AsymmetricObjectProperty( :hasChild )
DisjointObjectProperties( :hasParent :hasSpouse )
ReflexiveObjectProperty( :hasRelative )
IrreflexiveObjectProperty( :parentOf )
FunctionalObjectProperty( :hasHusband )
InverseFunctionalObjectProperty( :hasHusband )
TransitiveObjectProperty( :hasAncestor )
FunctionalDataProperty( :hasAge )

SubClassOf( :Woman :Person )
SubClassOf( :Mother :Woman )
SubClassOf(
   :Grandfather
   ObjectIntersectionOf( :Man :Parent )
)
SubClassOf( :Teenager
   DataSomeValuesFrom( :hasAge
      DatatypeRestriction( xsd:integer
         xsd:minExclusive "12"^^xsd:integer
         xsd:maxInclusive "19"^^xsd:integer
      )
   )
)
SubClassOf(
   Annotation( rdfs:comment "States that every man is a person." )
   :Man
   :Person
)
SubClassOf(
   :Father
   ObjectIntersectionOf( :Man :Parent )
)
SubClassOf(
   :ChildlessPerson
   ObjectIntersectionOf(
      :Person
      ObjectComplementOf(
         ObjectSomeValuesFrom(
            ObjectInverseOf( :hasParent )
            owl:Thing
         )
      )
   )
)
```

```
SubClassOf(
   ObjectIntersectionOf(
      ObjectOneOf( :Mary :Bill :Meg )
      :Female
   )
   ObjectIntersectionOf(
      :Parent
      ObjectMaxCardinality( 1 :hasChild )
      ObjectAllValuesFrom( :hasChild :Female )
   )
)

EquivalentClasses( :Person :Human )
EquivalentClasses(
   :Mother
   ObjectIntersectionOf( :Woman :Parent )
)
EquivalentClasses(
   :Parent
   ObjectUnionOf( :Mother :Father )
)
EquivalentClasses(
   :ChildlessPerson
      ObjectIntersectionOf(
      :Person
      ObjectComplementOf( :Parent )
   )
)
EquivalentClasses(
   :Parent
   ObjectSomeValuesFrom( :hasChild :Person )
)
EquivalentClasses(
   :HappyPerson
   ObjectIntersectionOf(
      ObjectAllValuesFrom( :hasChild :HappyPerson )
      ObjectSomeValuesFrom( :hasChild :HappyPerson )
   )
)
EquivalentClasses(
   :JohnsChildren
   ObjectHasValue( :hasParent :John )
)
EquivalentClasses(
   :NarcisticPerson
   ObjectHasSelf( :loves )
)
EquivalentClasses(
   :MyBirthdayGuests
   ObjectOneOf( :Bill :John :Mary)
)
EquivalentClasses(
   :Orphan
   ObjectAllValuesFrom(
```

```
      ObjectInverseOf( :hasChild )
      :Dead
   )
)
EquivalentClasses( :Adult otherOnt:Grownup )
   EquivalentClasses(
      :Parent
      ObjectSomeValuesFrom(
         :hasChild
         :Person
   )
)

DisjointClasses( :Woman :Man )
   DisjointClasses(
   :Mother
   :Father
   :YoungChild
)
HasKey( :Person () ( :hasSSN ) )

DatatypeDefinition(
   :personAge
   DatatypeRestriction( xsd:integer
      xsd:minInclusive "0"^^xsd:integer
      xsd:maxInclusive "150"^^xsd:integer
   )
)
DatatypeDefinition(
   :minorAge
   DatatypeRestriction( xsd:integer
      xsd:minInclusive "0"^^xsd:integer
      xsd:maxInclusive "18"^^xsd:integer
   )
)
DatatypeDefinition(
   :majorAge
   DataIntersectionOf(
      :personAge
      DataComplementOf( :minorAge )
   )
)
DatatypeDefinition(
   :toddlerAge
   DataOneOf( "1"^^xsd:integer "2"^^xsd:integer )
)

ClassAssertion( :Person :Mary )
ClassAssertion( :Woman :Mary )
ClassAssertion(
   ObjectIntersectionOf(
      :Person
      ObjectComplementOf( :Parent )
   )
```

```
      :Jack
   )
   ClassAssertion(
      ObjectMaxCardinality( 4 :hasChild :Parent )
      :John
   )
   ClassAssertion(
      ObjectMinCardinality( 2 :hasChild :Parent )
      :John
   )
   ClassAssertion(
      ObjectExactCardinality( 3 :hasChild :Parent )
      :John
   )
   ClassAssertion(
      ObjectExactCardinality( 5 :hasChild )
      :John
   )
   ClassAssertion( :Father :John )
   ClassAssertion( :SocialRole :Father )

   ObjectPropertyAssertion( :hasWife :John :Mary )
   NegativeObjectPropertyAssertion( :hasWife :Bill :Mary )
   NegativeObjectPropertyAssertion(
      :hasDaughter
      :Bill
      :Susan
   )
   DataPropertyAssertion( :hasAge :John "51"^^xsd:integer )
   NegativeDataPropertyAssertion( :hasAge :Jack "53"^^xsd:integer )

   SameIndividual( :James :Jim )
   SameIndividual( :John otherOnt:JohnBrown )
   SameIndividual( :Mary otherOnt:MaryBrown )
   DifferentIndividuals( :John :Bill )
)
```

4.3 Manchester syntax

```
Ontology: <http://example.com/owl/families>
AnnotationProperty: rdfs:comment
Datatype: personAge
    EquivalentTo:
        xsd:integer[>= 0 , <= 150]
Datatype: rdf:PlainLiteral
Datatype: toddlerAge
    EquivalentTo:
        {1 , 2}
Datatype: xsd:nonNegativeInteger
Datatype: minorAge
Datatype: majorAge
    EquivalentTo:
        (personAge and  not minorAge)
```

```
Datatype: xsd:integer
ObjectProperty: hasBrother
ObjectProperty: hasRelative
    Characteristics:
        Reflexive
ObjectProperty: otherOnt:child
    EquivalentTo:
        hasChild
ObjectProperty: parentOf
    Characteristics:
        Irreflexive
ObjectProperty: hasSon
    DisjointWith:
        hasDaughter
ObjectProperty: hasWife
    SubPropertyOf:
        hasSpouse
    Domain:
        Man
    Range:
        Woman
ObjectProperty: hasHusband
    Characteristics:
        InverseFunctional,
        Functional
ObjectProperty: hasParent
    DisjointWith:
        hasSpouse
    InverseOf:
        hasChild
ObjectProperty: hasUncle
    SubPropertyChain:
        hasFather o hasBrother
ObjectProperty: hasGrandparent
    SubPropertyChain:
        hasParent o hasParent
ObjectProperty: hasFather
    SubPropertyOf:
        hasParent
ObjectProperty: hasDaughter
    DisjointWith:
        hasSon
ObjectProperty: hasSpouse
    DisjointWith:
        hasParent
    Characteristics:
        Symmetric
ObjectProperty: hasAncestor
    Characteristics:
        Transitive
ObjectProperty: hasChild
    EquivalentTo:
        otherOnt:child
    Characteristics:
```

```
        Asymmetric
    InverseOf:
        hasParent
ObjectProperty: loves
DataProperty: otherOnt:age
    EquivalentTo:
        hasAge
DataProperty: hasAge
    Characteristics:
        Functional
    Domain:
        Person
    Range:
        xsd:nonNegativeInteger
    EquivalentTo:
        otherOnt:age
DataProperty: hasSSN
Class: OWL:Thing
Class: Father
    SubClassOf:
        Man
         and Parent
Class: Mother
    EquivalentTo:
        Parent
         and Woman
    SubClassOf:
        Woman
Class: Human
    EquivalentTo:
        Person
Class: YoungChild
Class: Dead
Class: Grandfather
    SubClassOf:
        Man
         and Parent
Class: Person
    Annotations:
        rdfs:comment "Represents the set of all people."
    EquivalentTo:
        Human
    HasKey:
        hasSSN
Class: MyBirthdayGuests
    EquivalentTo:
        {Bill , John , Mary}
Class: Adult
    EquivalentTo:
        otherOnt:Grownup
Class: SocialRole
Class: HappyPerson
    EquivalentTo:
        (hasChild some HappyPerson)
```

```
        and (hasChild only HappyPerson)
Class: Parent
    EquivalentTo:
        (Father or Mother),
        hasChild some Person
Class: Teenager
    SubClassOf:
        hasAge some xsd:integer[> 12 , <= 19]
Class: Orphan
    EquivalentTo:
         inverse (hasChild) only Dead
Class: NarcisticPerson
    EquivalentTo:
        loves some  Self
Class: ChildlessPerson
    EquivalentTo:
        Person
         and (not (Parent))
    SubClassOf:
        Person
         and (not ( inverse (hasParent) some OWL:Thing))
Class: Woman
    SubClassOf:
        Person
    DisjointWith:
        Man
Class: JohnsChildren
    EquivalentTo:
        hasParent value John
Class: otherOnt:Grownup
    EquivalentTo:
        Adult
Class: Female
Class: Man
    Annotations: rdfs:comment "States that every man is a person."
    SubClassOf:
        Person
    DisjointWith:
        Woman
Individual: Mary
    Types:
        Person,
        Woman
    SameAs:
        otherOnt:MaryBrown
Individual: John
    Types:
        Father,
        hasChild exactly 5 OWL:Thing,
        hasChild max 4 Parent,
        hasChild exactly 3 Parent,
        hasChild min 2 Parent
    Facts:
     hasWife  Mary,
```

```
     hasAge  51
    SameAs:
        Jack,
        otherOnt:JohnBrown
    DifferentFrom:
        Bill
Individual: Jim
Individual: otherOnt:JohnBrown
    SameAs:
        John
Individual: otherOnt:MaryBrown
    SameAs:
        Mary
Individual: James
Individual: Susan
Individual: Bill
    Facts:
      not  hasWife  Mary,
      not  hasDaughter  Susan
    DifferentFrom:
        John
Individual: Father
    Types:
        SocialRole
Individual: Jack
    Types:
        Person
         and (not (Parent))
    Facts:
      not  hasAge  53
    SameAs:
        John
Individual: Meg
DisjointClasses:
    Father,Mother,YoungChild
```

4.4 Turtle

```
@prefix : <http://example.com/owl/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.com/owl/families> rdf:type OWL:Ontology .

:majorAge rdf:type rdfs:Datatype ;
    OWL:equivalentClass [
        rdf:type rdfs:Datatype ;
        OWL:intersectionOf (
            :personAge
            [ rdf:type rdfs:Datatype ;
              OWL:datatypeComplementOf :minorAge]
```

```
        )
    ] .

:minorAge rdf:type rdfs:Datatype .

:personAge rdf:type rdfs:Datatype ;
    OWL:equivalentClass [
        rdf:type rdfs:Datatype ;
        OWL:onDatatype xs:integer ;
        OWL:withRestrictions (
        [ xs:maxInclusive 150 ]
        [ xs:minInclusive 0] )
    ] .

:toddlerAge rdf:type rdfs:Datatype ;
    OWL:equivalentClass [
        rdf:type rdfs:Datatype ;
        OWL:oneOf [
            rdf:type rdf:List ;
            rdf:first 1 ;
            rdf:rest [ rdf:type rdf:List ;
                     rdf:first 2 ;
                     rdf:rest rdf:nil
            ]
        ]
    ] .

:hasAncestor rdf:type OWL:ObjectProperty ,
      OWL:TransitiveProperty .

:hasBrother rdf:type OWL:ObjectProperty .

:hasChild rdf:type OWL:AsymmetricProperty ,
   OWL:ObjectProperty ;
  OWL:equivalentProperty otherOnt:child .

:hasDaughter rdf:type OWL:ObjectProperty ;
     OWL:propertyDisjointWith :hasSon .

:hasFather rdf:type OWL:ObjectProperty ;
   rdfs:subPropertyOf :hasParent .

:hasGrandparent rdf:type OWL:ObjectProperty ;
    OWL:propertyChainAxiom ( :hasParent
                             :hasParent
                           ) .

:hasHusband rdf:type OWL:FunctionalProperty ,
     OWL:InverseFunctionalProperty ,
     OWL:ObjectProperty .

:hasParent rdf:type OWL:ObjectProperty ;
   OWL:inverseOf :hasChild ;
   OWL:propertyDisjointWith :hasSpouse .
```

```
:hasRelative rdf:type OWL:ObjectProperty ,
  OWL:ReflexiveProperty .

:hasSon rdf:type OWL:ObjectProperty .

:hasSpouse rdf:type OWL:ObjectProperty ,
    OWL:SymmetricProperty .

:hasUncle rdf:type OWL:ObjectProperty ;
  OWL:propertyChainAxiom ( :hasFather
                           :hasBrother
                         ) .

:hasWife rdf:type OWL:ObjectProperty ;
    rdfs:domain :Man ;
    rdfs:range :Woman ;
    rdfs:subPropertyOf :hasSpouse .

:loves rdf:type OWL:ObjectProperty .

:parentOf rdf:type OWL:IrreflexiveProperty ,
                   OWL:ObjectProperty .

otherOnt:child rdf:type OWL:ObjectProperty .

:hasAge rdf:type OWL:DatatypeProperty ,
                 OWL:FunctionalProperty ;
    rdfs:domain :Person ;
    OWL:equivalentProperty otherOnt:age ;
    rdfs:range xs:nonNegativeInteger .

:hasSSN rdf:type OWL:DatatypeProperty .

otherOnt:age rdf:type OWL:DatatypeProperty .

:Adult rdf:type OWL:Class ;
       OWL:equivalentClass otherOnt:Grownup .

:ChildlessPerson rdf:type OWL:Class ;
 OWL:equivalentClass [ rdf:type OWL:Class ;
    OWL:intersectionOf (
        :Person
        [ rdf:type OWL:Class ;
          OWL:complementOf :Parent
        ]
    )
] ;
 rdfs:subClassOf [ rdf:type OWL:Class ;
   OWL:intersectionOf (
        :Person
        [   rdf:type OWL:Class ;
            OWL:complementOf [
                rdf:type OWL:Restriction ;
```

```
                OWL:onProperty [ OWL:inverseOf :hasParent
                            ] ;
                OWL:someValuesFrom OWL:Thing
            ]
        ]
    )
 ] .

:Dead rdf:type OWL:Class .

:Father rdf:type OWL:Class ;
        rdfs:subClassOf [
            rdf:type OWL:Class ;
            OWL:intersectionOf ( :Man
                                 :Parent
            )
        ] .

:Female rdf:type OWL:Class .

:Grandfather rdf:type OWL:Class ;
    rdfs:subClassOf [
        rdf:type OWL:Class ;
        OWL:intersectionOf ( :Man :Parent )
        ] .

:HappyPerson rdf:type OWL:Class ;
    OWL:equivalentClass [
        rdf:type OWL:Class ;
        OWL:intersectionOf (
            [ rdf:type OWL:Restriction ;
              OWL:onProperty :hasChild ;
              OWL:someValuesFrom :HappyPerson
            ]
            [ rdf:type OWL:Restriction ;
              OWL:onProperty :hasChild ;
              OWL:allValuesFrom :HappyPerson
            ]
        )
    ] .

:Human rdf:type OWL:Class ;
       OWL:equivalentClass :Person .

:JohnsChildren rdf:type OWL:Class ;
    OWL:equivalentClass [
        rdf:type OWL:Restriction ;
        OWL:onProperty :hasParent ;
        OWL:hasValue :John
    ] .

:Man rdf:type OWL:Class ;
     rdfs:subClassOf :Person ;
     OWL:disjointWith :Woman .
```

```
[ rdf:type OWL:Axiom ;
  rdfs:comment "States that every man is a person." ;
  OWL:annotatedSource :Man ;
  OWL:annotatedTarget :Person ;
  OWL:annotatedProperty rdfs:subClassOf
] .

:Mother rdf:type OWL:Class ;
    OWL:equivalentClass [
        rdf:type OWL:Class ;
        OWL:intersectionOf ( :Parent :Woman )
    ] ;
    rdfs:subClassOf :Woman .

:MyBirthdayGuests rdf:type OWL:Class ;
    OWL:equivalentClass [
        rdf:type OWL:Class ;
        OWL:oneOf ( :John :Mary :Bill )
      ] .


:Person rdf:type OWL:Class ;
        rdfs:comment "Represents the set of all people." ;
        OWL:hasKey ( :hasSSN
                   ) .


:Teenager rdf:type OWL:Class ;
    rdfs:subClassOf [
        rdf:type OWL:Restriction ;
            OWL:onProperty :hasAge ;
            OWL:someValuesFrom [
                rdf:type rdfs:Datatype ;
                OWL:onDatatype xs:integer ;
                OWL:withRestrictions (
                    [ xs:maxInclusive 19 ]
                    [ xs:minExclusive 12 ]
                )
           ]
    ] .

:Woman rdf:type OWL:Class ;
       rdfs:subClassOf :Person .


:Bill rdf:type OWL:NamedIndividual .

[ rdf:type OWL:NegativePropertyAssertion ;
  OWL:sourceIndividual :Bill ;
  OWL:targetIndividual :Mary ;
  OWL:assertionProperty :hasWife
] .
```

```
[ rdf:type OWL:NegativePropertyAssertion ;
  OWL:sourceIndividual :Bill ;
  OWL:targetIndividual :Susan ;
  OWL:assertionProperty :hasDaughter
] .

:Father rdf:type :SocialRole ,
                 OWL:NamedIndividual .

:Jack rdf:type OWL:NamedIndividual ,
    [ rdf:type OWL:Class ;
      OWL:intersectionOf (
        :Person
        [ rdf:type OWL:Class ;
          OWL:complementOf :Parent
        ]
      )
    ] ;
    OWL:sameAs :John .

[ rdf:type OWL:NegativePropertyAssertion ;
  OWL:targetValue 53 ;
  OWL:sourceIndividual :Jack ;
  OWL:assertionProperty :hasAge
] .

:James rdf:type OWL:NamedIndividual .

:Jim rdf:type OWL:NamedIndividual .

:John rdf:type :Father ,
   OWL:NamedIndividual ,
   [ rdf:type OWL:Restriction ;
     OWL:onProperty :hasChild ;
     OWL:onClass :Parent ;
     OWL:qualifiedCardinality "3"^^xs:nonNegativeInteger
   ] ,
   [ rdf:type OWL:Restriction ;
     OWL:onProperty :hasChild ;
     OWL:cardinality "5"^^xs:nonNegativeInteger
   ] ,
   [ rdf:type OWL:Restriction ;
     OWL:onProperty :hasChild ;
     OWL:onClass :Parent ;
     OWL:minQualifiedCardinality "2"^^xs:nonNegativeInteger
   ] ,
   [ rdf:type OWL:Restriction ;
     OWL:onProperty :hasChild ;
     OWL:onClass :Parent ;
     OWL:maxQualifiedCardinality "4"^^xs:nonNegativeInteger
   ] ;
  :hasAge 51 ;
  :hasWife :Mary ;
  OWL:sameAs otherOnt:JohnBrown .
```

```
:Mary rdf:type :Person , :Woman , OWL:NamedIndividual ;
    OWL:sameAs otherOnt:MaryBrown .

:Meg rdf:type OWL:NamedIndividual .

:Susan rdf:type OWL:NamedIndividual .

otherOnt:JohnBrown rdf:type OWL:NamedIndividual .

otherOnt:MaryBrown rdf:type OWL:NamedIndividual .

[ rdf:type OWL:AllDifferent ;
  OWL:distinctMembers ( :Bill
                        :John
                      )
] .

[ rdf:type OWL:Class ;
    rdfs:subClassOf [
        rdf:type OWL:Class ;
        OWL:intersectionOf (
            :Parent
            [ rdf:type OWL:Restriction ;
              OWL:onProperty :hasChild ;
              OWL:allValuesFrom :Female
            ]
            [ rdf:type OWL:Restriction ;
              OWL:onProperty :hasChild ;
              OWL:maxCardinality "1"^^xs:nonNegativeInteger
            ]
       )
  ] ;
  OWL:intersectionOf (
    :Female
    [ rdf:type OWL:Class ;
      OWL:oneOf ( :Mary :Bill :Meg )
    ]
  )
] .

[ rdf:type OWL:AllDisjointClasses ;
  OWL:members ( :Father
                :Mother
                :YoungChild
              )
] .
```