W3C Editor's Draft

OWL 2 Web Ontology Language
Data Range Extension: Linear Equations

W3C Working Group Note 8 August 2012

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-dr-linear-20120808/
Latest editor's draft:
http://www.w3.org/2007/OWL/draft/owl2-dr-linear/
Previous version:
http://www.w3.org/2007/OWL/draft/ED-owl2-dr-linear-20091027/
Authors:
Bijan Parsia, University of Manchester
Uli Sattler, University of Manchester

A color-coded version of this document showing changes made since the previous version is also available.

This document is also available in these non-normative formats: PDF version.

Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 Document Overview describes the overall state of OWL 2, and should be read before other OWL 2 documents.

This document specifies a syntax and semantics for incorporating linear equations with rational coefficients solved in the reals in OWL 2.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Summary of Changes

There have been no substantive changes since the previous version. For details on the minor changes see the change log and color-coded diff.

Please Send Comments

Please send any comments to public-owl-comments@w3.org (public archive). Although work on this document by the OWL Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion among developers is welcome at public-owl-dev@w3.org (public archive).

No Endorsement

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

W3C Editor's Draft

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. The group does not expect this document to become a W3C Recommendation. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

1 Overview

OWL 2 has a sophisticated set of built-in numeric dataranges and rather expressive constructors for building new dataranges out of the basic dataranges. A major restriction on the sort of data ranges that can be built with existing constructors (i.e., datatype facets) is that only **unary** dataranges can be defined --- i.e., only datatypes may be defined. One can say that the value of a data property has to be an integer greater than 5, but one cannot say that the value of one data property is greater than that of another data property. Furthermore, one might wish to relate the values of two properties by more complex equations than mere comparisons.

This document defines an extension to OWL for defining dataranges in terms of linear (in)equalites with rational coefficients solved over the algebraic reals. These dataranges can be used in OWL axioms to, for example, define classes in terms of a constraint on the relationships between values of distinct data properties.

This extension is restricted in two respects for the sake of reasonable implementability:

- The datarange definition language is limited to linear (in)equations with rational coefficients. Transcendental functions (such as *sin*) are not permitted. Similarly, non-linear polynomials are not permitted. While both are essential to many applications, they are not likely to be widely implemented due to practical and theoretical problems that still must be dealt with.
- Equation-based dataranges can only constrain values of data properties of a single individual. That is, one cannot compare the boiling point of the one individual, *water*, with the boiling point of another individual, say, *copper*. Restrictions using dataranges this way are known as **path free**. While the theory for arbitrary data restrictions is well known, there is still a dearth of optimizations that would make their inclusion practical.

These restrictions may be lifted, to various degrees, in future versions of this specification.

2 Examples

Consider the relation between the boiling point and the melting point of a substance. For example, for water (at 1 atmosphere) the boiling point is 100C and the melting point 0C. This can be represented in plain OWL quite easily:

```
ClassAssertion(DataHasValue(melting_point "0"^^xsd:decimal) water)
ClassAssertion(DataHasValue(boiling_point "100"^^xsd:decimal) water)
```

From these assertions it follows that the boiling point of water is greater than its melting point. This is, in fact, a general principle for substances: the boiling point of a normal physical substance is greater or equal to its melting point. This physical law can be expressed with a datarange with two free variables x and y, representing the melting and boiling point, respectively.:

```
EquivalentClasses(NormalSubstance DataAllValuesFrom(melting_point boiling_point
        DataComparison(Arguments(x y) leq( x y ))))
```

With this definition (and given that *melting_point* and *boiling_point* are functional), one can infer:

```
ClassAssertion(NormalSubstance water)
```

When administering drugs, there are many factors that go into determining the maximum safe dose. Often, the maximum the maximum single dose of a drug is computed in terms of milligram of drug per kilogram of body weight.

```
EquivalentClasses(SafelyDosedPatient DataAllValuesFrom(tookDrugInAmount weight
        DataComparison(Arguments(totalDoseInMg weightinKg) leq(totalDoseInMg times(2, weightInKg)))))
```

This axiom states that the safe dose is 2 milligrams per kilogram, and thus that a safe dose (in milligrams) for a person of a given weight must be less than 2 times the weight (in kilograms) of the patient.

As safe doses vary with age and other factors, one could define a number of such classes with varying constraints on the safety of the dose.

3 Syntax

3.1 Functional Syntax

As with built-in OWL 2 data ranges, linear (in)equations may be used to form universal, existential, and quantified restrictions on (sets of) data properties.

```
ComparisonRelation := 
    'gt' |
    'lt' |
    'geq' |
    'leq' |
    'eq' |
    'neq'
Variable := NCName
Rational := Integer / NonZeroInteger
Term := 'times' '(' [ Rational ] Variable ')' | Variable
LinearExpression:= 'plus' '(' Term { Term } ')' | Term | Variable
Arguments := 'Arguments' '(' NCName { NCName } ')'
Comparison := 'DataComparison' '(' Arguments ComparisonRelation'(' Variable Variable ')' ')'
ScaledComparison := 'DataComparison' (' Arguments ComparisonRelation '(' Term Term ')' ')'
LinearComparison := 'DataComparison' '(' Arguments ComparisonRelation '(' LinearExpression
LinearExpression ')' ')'
DataComparison := Comparison | ScaledComparison | LinearComparison
```

The definition of a DataRange is extended with the various comparisons:

```
DataRange := 
    Datatype |
    DataComplementOf |
    DataOneOf |
    DatatypeRestriction |
    DataComparison
```

It is not currently possible for user defined (in)equations to be named, though it is easy to spec a natural syntax:

```
DataComparisonDefinition := 'DataComparisonDefinition' '(' axiomAnnotations IRI DataRange ')'
```

In order to retain decidability with naming, there needs to be acyclicity condition akin to those for datatypes. Furthermore, since there are *DataComparisons* which are equivalent to datatypes, the datatype and data comparison conditions must appropriately interact.

3.2 RDF Mapping

(In)equations in RDF are expressed using MathML as below. The equations are serialized as *rdf:XMLLiteral*s. The content of those literals must conform to the "owl-linear-comparisons-mathml.xsd".

```
<!DOCTYPE rdf:RDF [
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>
<rdf:RDF xmlns="http://example.org/#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about="http://example.org/"/>
    <owl:DatatypeProperty rdf:about="#boiling_point"/>
    <owl:DatatypeProperty rdf:about="#melting_point"/>

    <owl:Class rdf:about="#NormalSubstance">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperties rdf:parseType="Collection">
                    <owl:DatatypeProperty rdf:about="#boiling_point"/>
                    <owl:DatatypeProperty rdf:about="#melting_point"/>
                </owl:onProperties>
                <owl:allValuesFrom>
                    <owl:DataComparison>
                        <rdf:value rdf:parseType="Literal">
                            <lambda xmlns="http://www.w3.org/1998/Math/MathML"
                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                xsi:schemaLocation="http://www.w3.org/1998/Math/MathML
                                            owl-linear-comparisons-mathml.xsd">
                                <bvar>
                                    <ci>x</ci>
                                </bvar>
                                <bvar>
                                    <ci>y</ci>
                                </bvar>
                                <apply>
                                    <leq/>
                                    <ci>x</ci>
                                    <ci>y</ci>
                                </apply>
                            </lambda>
                        </rdf:value>
                    </owl:DataComparison>
                </owl:allValuesFrom>
            </owl:Restriction>
        </owl:equivalentClass>
    </owl:Class>

    <rdf:Description rdf:about="#water">
        <rdf:type>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#boiling_point"/>
                <owl:hasValue rdf:datatype="&xsd;integer">100</owl:hasValue>
            </owl:Restriction>
        </rdf:type>
        <rdf:type>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#melting_point"/>
                <owl:hasValue rdf:datatype="&xsd;integer">0</owl:hasValue>
            </owl:Restriction>
        </rdf:type>
    </rdf:Description>
 </rdf:RDF>
```

3.3 XML Syntax

For the XML syntax, the terminals of the functional syntax are mapped into corresponding MathML elements. Consider the water example:

```
<Ontology xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2002/07/owl# owlxml-datarange.xsd"
    xmlns="http://www.w3.org/2002/07/owl#"
    ontologyIRI="http://example.org/">
    <ClassAssertion>
        <DataHasValue>
            <DataProperty IRI="melting_point"/>
            <Literal datatypeIRI="xsd:decimal">0</Literal>
        </DataHasValue>
        <NamedIndividual IRI="water"/>
    </ClassAssertion>
    <ClassAssertion>
```

```
        <DataHasValue>
            <DataProperty IRI="boiling_point"/>
            <Literal datatypeIRI="xsd:decimal">100</Literal>
        </DataHasValue>
        <NamedIndividual IRI="water"/>
    </ClassAssertion>

    <EquivalentClasses>
        <Class IRI="NormalSubstance"/>
        <DataAllValuesFrom>
            <DataProperty IRI="melting_point"/>
            <DataProperty IRI="boiling_point"/>
            <DataComparison>
                <lambda xmlns="http://www.w3.org/1998/Math/MathML">
                    <bvar>
                        <ci>x</ci>
                    </bvar>
                    <bvar>
                        <ci>y</ci>
                    </bvar>
                    <apply>
                        <leq/>
                        <ci>x</ci>
                        <ci>y</ci>
                    </apply>
                </lambda>
            </DataComparison>
        </DataAllValuesFrom>
    </EquivalentClasses>
</Ontology>
```

In order to validate, one must use an extended version of the XML Schema. See Appendix A for the schema.

4 Semantics

The semantics of all constructs where data ranges can occur (DataSomeValuesFrom, DataAllValuesFrom, DataMinCardinality, DataExactCardinality, DataMaxCardinality, DataComplementOf) is defined in Section 2 of the Semantics. This section defines the meaning of DataComparisons.

As explained in the Semantics document, this is accomplished by extending the datatype interpretation function $\cdot^{DT}$ to *DataComparison*. First some notation: for an expression *exp*, a variable *y* and a value *v*, *exp[y -> v]* is the expression obtained by replacing all occurrences of *y* in *exp* with *v*.

Next, on the value space of *owl:real*, the equality = and ordering < are defined as usual, and the operators + and * are the usual addition and multiplication operators on the real numbers.

The value of terms is then defined as follows:

- $(times(v1\ v2))^{DT} = v1 * v2$
- $(plus\ (t1\ ...\ tk))^{DT} = (t1)^{DT} + ... + (tk)^{DT}$

Intuitively, in order to find out whether a pair *(5,60)* of numbers is in, say, *DataComparison(Arguments(y1 y2) lt (times("4"^^owl:real y1) times("1"^^owl:real y1)))*^{DT}, one replaces all occurrences of *y1* in both *times(...)* terms with *5*, all occurrences of *y2* in both terms with *60*, computes the value of both *times(...)* terms (the first giving 20, the second giving 60), and then checks whether *lt* holds between them. Since this is indeed the case, the pair *(5,60)* is in *DataComparison(...)*.

In what follows, *y1* and *y2* refer to variables, *t1* and *t2* to terms, and *L1* and *L2* to linear expressions.

- $(DataComparison(Arguments(y1\ y2)\ comprel(y1\ y2)))^{DT}$ =
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v1 < v2* } if *comprel* is *lt*
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v2 < v1* } if *comprel* is *gt*
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v1 < v2* or *v1 = v2* } if *comprel* is *leq*
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v2 < v1* or *v1 = v2* } if *comprel* is *geq*
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v2 = v1* } if *comprel* is *eq*
  - { *(v1, v2) in* $((owl:real)^{DT})^2$ | *v2 < v1* or *v1 < v2* } if *comprel* is *neq*

- $(DataComparison(Arguments(y1\ y2)\ comprel(t1\ t2)))^{DT}$ =

- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t1[y1->v1][y2->v2])^{DT} < (t2[y1->v1][y2->v2])^{DT} }* if *comprel* is *lt*
- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t2[y1->v1][y2->v2])^{DT} < (t1[y1->v1][y2->v2])^{DT} }* if *comprel* is *gt*
- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t1[y1->v1][y2->v2]))^{DT} < (t2[y1->v1][y2->v2])^{DT}* or *(t1[y1->v1][y2->v2])^{DT} = (t2[y1->v1][y2->v2])^{DT} }* if *comprel* is *leq*
- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t2[y1->v1][y2->v2]))^{DT} < (t1[y1->v1][y2->v2])^{DT}* or *(t1[y1->v1][y2->v2])^{DT} = (t2[y1->v1][y2->v2])^{DT} }* if *comprel* is *geq*
- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t2[y1->v1][y2->v2]))^{DT} = (t1[y1->v1][y2->v2])^{DT} }* if *comprel* is *eq*
- *{ (v1, v2) in ((owl:real)^{DT})^2 | (t2[y1->v1][y2->v2]))^{DT} < (t1[y1->v1][y2->v2])^{DT}* or *(t1[y1->v1][y2->v2])^{DT} < (t2[y1->v1][y2->v2])^{DT} }* if *comprel* is *neq*

- *(DataComparison(Arguments(y1 ... yk) comprel(L1 L2)))^{DT} =*
  - *{ (v1,..., vk) in ((owl:real)^{DT})^k | (L1[y1->v1]...[yk->vk])^{DT} < (L2[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *lt*
  - *{ (v1, ...,vk) in ((owl:real)^{DT})^k | (L2[y1->v1]...[yk->vk])^{DT} < (L1[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *gt*
  - *{ (v1,..., vk) in ((owl:real)^{DT})^k | (L1[y1->v1]...[yk->vk])^{DT} < (L2[y1->v1]...[yk->vk])^{DT}* or *(L1[y1->v1]...[yk->vk])^{DT} = (L2[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *leq*
  - *{ (v1,...,vk) in ((owl:real)^{DT})^k | (L2[y1->v1]...[yk->vk])^{DT} < (L1[y1->v1]...[yk->vk])^{DT}* or *(L1[y1->v1]...[yk->vk])^{DT} = (L2[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *geq*
  - *{ (v1,..., vk) in ((owl:real)^{DT})^k | (L2[y1->v1]...[yk->vk])^{DT} = (L1[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *eq*
  - *{ (v1,..., vk) in ((owl:real)^{DT})^k | (L2[y1->v1]...[yk->vk])^{DT} < (L1[y1->v1]...[yk->vk])^{DT}* or *(L1[y1->v1]...[yk->vk])^{DT} < (L2[y1->v1]...[yk->vk])^{DT} }* if *comprel* is *neq*

5 Implementation Considerations

There is a rich literature on implementing linear solvers. The key papers for the integration between OWL and a linear solver are:

- Description Logics with Concrete Domains - A Survey
- Description Logic Systems with Concrete Domains: Applications for the Semantic Web

6 Appendix: XML Schemas

This schema is named "owlxml-with-linear-comparisons.xsd".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3.org/2002/07/owl#" xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:m="http://www.w3.org/1998/Math/MathML">
    <xsd:import namespace="http://www.w3.org/1998/Math/MathML"
        schemaLocation="owl-comparisons-mathml.xsd"/>
    <xsd:redefine schemaLocation="http://www.w3.org/2009/09/owl2-xml.xsd">
        <xsd:group name="DataRange">
            <xsd:choice>
                <xsd:group ref="owl:DataRange"/>
                <xsd:element ref="owl:DataComparison"/>
            </xsd:choice>
        </xsd:group>
    </xsd:redefine>

    <xsd:complexType name="DataComparison">
        <xsd:complexContent>
            <xsd:extension base="owl:DataRange">
                <xsd:sequence>
                    <xsd:element ref="m:lambda" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="DataComparison" type="owl:DataComparison"/>
</xsd:schema>
```

This schema is named "owl-linear-comparisons-mathml.xsd".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3.org/1998/Math/MathML" xmlns="http://www.w3.org/1998/Math/MathML"
    xmlns:m="http://www.w3.org/1998/Math/MathML" elementFormDefault="qualified">
    <xsd:element name="gt"/>
```

```
    <xsd:element name="lt"/>
    <xsd:element name="geq"/>
    <xsd:element name="leq"/>
    <xsd:element name="eq"/>
    <xsd:element name="neq"/>

    <xsd:element name="ci" type="xsd:NCName"/>
    <xsd:element name="sep"/>
    <xsd:element name="cn">
        <xsd:complexType mixed="true">
            <xsd:sequence>
                <xsd:element ref="sep" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="type">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:pattern value="real|rational"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="bvar">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="m:ci" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="times">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="m:cn"/>
                <xsd:element ref="m:ci" maxOccurs="1" minOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="plus">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="m:times" minOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="apply">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:choice minOccurs="1" maxOccurs="1">
                    <xsd:element ref="m:gt"/>
                    <xsd:element ref="m:lt"/>
                    <xsd:element ref="m:geq"/>
                    <xsd:element ref="m:leq"/>
                    <xsd:element ref="m:eq"/>
                    <xsd:element ref="m:neq"/>
                </xsd:choice>
                <xsd:choice minOccurs="2" maxOccurs="2">
                    <xsd:element ref="m:ci"/>
                    <xsd:element ref="m:times"/>
                    <xsd:element ref="m:plus"/>
                </xsd:choice>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="lambda">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="m:bvar" minOccurs="1" maxOccurs="unbounded"/>
                <xsd:element ref="m:apply"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

7 Appendix: Change Log (Informative)

7.1 Changes Since Draft of 11 June 2009

- Slight change to the functional syntax grammar (which doesn't change the recognized language).
- Added XML Schemas for the expression language and for an extenion to the XML Serialization.
- Removed naming of (in)equations.
- Removed Manchester syntax.