W3C Editor's Draft

OWL 2 Web Ontology Language
Data Range Extension: Linear Equations

W3C Editor's Draft 11 June 2009

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-dr-linear-20090611/
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-20090610/ (color-coded diff)
Authors:
Bijan Parsia, University of Manchester
Uli Sattler, University of Manchester

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.

W3C Editor's Draft

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May Be Superseded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

This document has not changed since the previous version. It is being republished only simplify management of the OWL 2 document set.

Please Comment By 30 July 2009

The OWL Working Group seeks public feedback on this Editor's Draft. Please send your comments to public-owl-comments@w3.org (public archive). If possible, please offer specific changes to the text that would address your concern. You may also wish to check the Wiki Version of this document and see if the relevant text has already been updated.

No Endorsement

Publication as a Editor's Draft 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.

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.

Editor's Note: We should include illustrative examples

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
        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 w
        DataComparison(Arguments(totalDoseInMg weightinKg) leq(totalDoseI
```

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 ')'
LinearExpression:= 'plus' '(' Term { Term } ')'
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
```

Editor's Note: Arguments may be removed in favor of positional variables.

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

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

It is also possible for user defined (in)equations to be named:

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

Editor's Note: Need to add restrictions a la those for datatypes. We have to be careful not to allow interactions between the datatypes and Comparisons that end up being equivalent to datatypes.

Editor's Note: It's unclear if we need for there to be an entity corresponding to DataComparisons. This last production certain suggests that it's the case.

3.2 RDF Mapping

Editor's Note: Mapping table awaiting finalizing the MathML

(In)equations in RDF are expressed using MathML as below. The equations are serialized as *rdf:XMLLiteral*s.

```
<!DOCTYPE rdf:RDF [
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
]>
<rdf:RDF xmlns="http://example.org/#" xmlns:rdfs="http://www.w3.org/2000/
    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/Ma
                                xmlns:xsi="http://www.w3.org/2001/XMLSche
                                xsi:schemaLocation="http://www.w3.org/199
                                                http://www.w3.org/Math/XMLSch
                                <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:hasVal
            </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

Editor's Note: Just examples for now. Two issues need to be solved: 1) we need to decide whether we want lambda notation (i.e., arguments) or positional variables (and if the latter...how to express them) and 2) I need to figure out how to derive our constraints from the MML schema and integrate it with the OWL/ XML schema

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.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"/>
            <DataComparsion>
                <lambda xmlns="http://www.w3.org/1998/Math/MathML"
                    xsi:schemaLocation="http://www.w3.org/1998/Math/MathM
                    http://www.w3.org/Math/XMLSchema/mathml2/mathml2.xsd"
                    <bvar>
                        <ci>x</ci>
                    </bvar>
                    <bvar>
                        <ci>y</ci>
                    </bvar>
                    <apply>
                        <leq/>
                        <ci>x</ci>
                        <ci>y</ci>
                    </apply>
                </lambda>
            </DataComparsion>
        </DataAllValuesFrom>
    </EquivalentClasses>
</Ontology>
```

3.4 Manchester Syntax

Editor's Note: Out of synch with FS.

```
ComparisonRelation :=
    '>' |
    '<' |
    '>=' |
```

```
      '<='  |
      '='  |
      '!='
datarangeRestriction ::= ComparisonRelation '[' facet restrictionValue
{ ',' facet restrictionValue } ']'
dataRange ::= datatype | dataComplementOf | dataOneOf |
datatypeRestriction | datarangeRestriction
```

4 Semantics

Editor's Note: Needs to catch up with the new datatype regime

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 *5*, compute the value of both terms, and then checks whether *lt* holds between them.

In what follows, *y1* and *y2* refer to variables, *t1* and *t2* to terms, and *l1* and *l2* to linear expressions.

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

- http://staff.fim.uni-passau.de/forschung/mip-berichte/MIP-0005.ps
- Quantifier Elimination of Real Closed Fields in the Context of Applied Description Logic -- discusses the Racer implementation of non-linear (in)equations
- Algorithms in Real Agebraic Geometry -- pretty comprehensive, downloadable textbook
- RISC-CLP(Real) an early Prolog system with non-linear inequations. Good examples.

- Comparison of Several Decision Algorithms for the Existential Theory of the Reals
- Description Logics with Concrete Domains - A Survey
- Description Logic Systems with Concrete Domains: Applications for the Semantic Web
- Datatypes in OWL

Comparisons:

- Simple Comparisons (note, for continuous domains they are polynomial)
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.8859