Abstract

OWL 2 extends the W3C OWL Web Ontology Language with a small but useful set of features that have been requested by users, for which effective reasoning algorithms are now available, and that OWL tool developers are willing to support. The new features include extra syntactic sugar, additional property and qualified cardinality constructors, extended datatype support, simple metamodeling, and extended annotations. The Manchester syntax is a user-friendly compact syntax for OWL 2; it is frame-based, as opposed to the axiom-based other syntaxes for OWL 2. The Manchester Syntax is used in the OWL 2 Primer, and this document provides the language used there. It is expected that tools will extend the Manchester Syntax for their own purposes, and tool builders may collaboratively extend the common language. It is already used in Protégé 4 and TopBraid composer.
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May Be Superseded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/.

Set of Documents

This document is being published as one of a set of 11 documents:

1. Structural Specification and Functional-Style Syntax
2. Direct Semantics
3. RDF-Based Semantics
4. Conformance and Test Cases
5. Mapping to RDF Graphs
6. XML Serialization
7. Profiles
8. Quick Reference Guide
9. New Features and Rationale
10. Manchester Syntax (this document)
11. rdf:text: A Datatype for Internationalized Text

First Public Working Draft

This description of the Manchester Syntax is derived from the syntax used in various OWL tools and in the OWL 2 Primer as of November 2008. It is expected that tools will extend the Manchester Syntax for their own purposes, and tool builders may collaboratively extend the common language.

The Working Group expects this document, when done, to be a Working Group Note, not a W3C Recommendation. As expressed in the document conformance clause, OWL systems are not required to read or write this syntax.

Please Comment By 2009-01-23

The OWL Working Group seeks public feedback on this First Public Working 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 for internal-review comments and changes being drafted which may address your concerns.
1 Introduction

The Manchester OWL syntax is a user-friendly syntax for OWL 2 descriptions, but it can also be used to write entire OWL 2 ontologies. The original version of the Manchester OWL syntax [Manchester OWL DL Syntax] was created for OWL 1 DL [OWL Semantics and Abstract Syntax]; it is here updated for OWL 2 ontologies.
The Manchester syntax is used in Protégé 4 [Protégé 4] and TopBraid composer [TopBraid Composer], particularly for entering and displaying descriptions associated with classes. Some tools (e.g., Protégé 4) extend the syntax to allow even more compact presentation in some situations (e.g., for explanation) or to replace IRIs by label values, but this document does not include any of these special-purpose extensions.

Editor's Note: Issue 146 relates to the potential use of label annotations instead of IRIs and may affect the Manchester Syntax. Input on this aspect of the syntax is encouraged.

The Manchester OWL syntax gathers together information about names in a frame-like manner, as opposed to RDF/XML [RDF Syntax], the functional-style syntax for OWL 2 [OWL 2 Syntax], and the XML syntax for OWL 2 [OWL 2 XML Syntax]. It is thus closer to the abstract syntax for OWL 1 DL [OWL Semantics and Abstract Syntax], than the above syntaxes for OWL 2. Nevertheless, parsing the Manchester OWL syntax into the OWL 2 structural specification is quite easy, as it is easy to identify the axioms inside each frame.

An example ontology in the Manchester OWL syntax can be found in the OWL Primer [OWL 2 Primer].

2 The Grammar

The Manchester syntax of OWL 2 is defined using a standard BNF notation, which is summarized in the table below.

Table 1. The BNF Notation Used in this Document

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonterminal symbols</td>
<td>boldface</td>
<td>ClassExpression</td>
</tr>
<tr>
<td>terminal symbols</td>
<td>single quoted</td>
<td>'PropertyRange'</td>
</tr>
<tr>
<td>zero or more</td>
<td>curly braces</td>
<td>{ ClassExpression }</td>
</tr>
<tr>
<td>zero or one</td>
<td>square brackets</td>
<td>[ ClassExpression ]</td>
</tr>
<tr>
<td>alternative</td>
<td>vertical bar</td>
<td>Assertion</td>
</tr>
</tbody>
</table>

Because comma-separated lists occur in very many places in the syntax, to save space the grammar has three meta-productions, one for non-empty lists, one for lists of minimum length two, and one for non-empty lists with annotations in them.

```
<NT>List  ::=  <NT> { , <NT> }
<NT>2List  ::=  <NT> , <NT>List
<NT>AnnotatedList  ::=  [ annotations ] <NT> { , [ annotations ] <NT> }
```
Documents in the Manchester OWL syntax consist of sequences of Unicode characters [UNICODE] and are encoded in UTF-8 [RFC3829].

The grammar for the Manchester syntax does not explicitly show white space. White space is allowed between any two terminals or non-terminals except inside nonNegativeInteger, prefix, reference, full-IRI, lexicalValue, integerLiteral, decimalLiteral, floatingPointLiteral, and languageTag. White space is required between two terminals or non-terminals if its removal could cause ambiguity. Generally this means requiring white space except before and after punctuation (e.g., commas, parentheses, braces, and brackets).

White space is a sequence of blanks (U+20), tabs (U+9), line feeds (U+A), carriage returns (U+D), and comments. Comments are maximal sequences of Unicode characters starting with a '#' and not containing a line feed or a carriage return. Note that comments are only recognized where white space is allowed, and thus not inside the above non-terminals.

The syntax uses the keywords 'and', 'or', and 'not', which are used in descriptions, that can be confused with their use as IRIs. When there is an ambiguity the keyword use is to be used.

2.1 IRIs, Integers, Literals, and Entities

Names are IRIs (the successors of URIs) and can either be given in full or can be abbreviated using CURIEs [CURIE].

This syntax uses short forms for common data values, e.g., strings and numbers, and short forms for some common datatypes, e.g., integer. These correspond to the obvious long forms.

```plaintext
full-IRI := 'IRI as defined in [RFC3987], enclosed in a pair of < (U+3C) and > (U+3E) characters'
NCName := 'as defined in [XML Namespaces]
irelative-ref := 'as defined in [RFC3987]
namespace := full-IRI
prefix := NCName
reference := irelative-ref
curie := [ [ prefix ] ':' ] reference
IRI := full-IRI | curie
nonNegativeInteger ::= zero | positiveInteger
positiveInteger ::= nonZero { digit }
digits ::= digit { digit }
digit ::= zero | nonZero
nonZero := '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
zero ::= '0'
```
classIRI ::= IRI
Datatype ::= datatypeIRI | 'integer' | 'decimal' | 'float' | 'string'
datatypeIRI ::= IRI
objectPropertyIRI ::= IRI
dataPropertyIRI ::= IRI
annotationPropertyIRI ::= IRI
individual ::= individualIRI | nodeID
individualIRI ::= IRI
nodeID ::= 'a node ID of the form _:name as specified in the N-Triples specification '
literal ::= typedLiteral | abbreviatedXSDStringLiteral | abbreviatedRDFTextLiteral | integerLiteral
typedLiteral ::= lexicalValue '^'^ Datatype
abbreviatedXSDStringLiteral ::= quotedString
abbreviatedRDFTextLiteral ::= quotedString '@' languageTag
languageTag ::= 'a nonempty (not quoted) string defined as specified in BCP 47 '
lexicalValue ::= quotedString
quotedString ::= 'a finite sequence of characters in which " (U+22) and \ (U+5C) occur only in pairs of the form " (U+22, U+5C) and \ (U+22, U+22), enclosed in a pair of " (U+22) characters'
floatingPointLiteral ::= [ '+' | '-' ] ( digits [ '.' digits ] [ exponent ] ) | '. digits [exponent]
exponent ::= ('e' | 'E') [ '+' | '-' ] digits
decimalLiteral ::= [ '+' | '-' ] digits
integerLiteral ::= [ '+' | '-' ] digits
entity ::= 'Datatype' '(' datatypeIRI ')' | 'Class' '(' classIRI ')' | 'ObjectProperty' '(' objectPropertyIRI ')' | 'DataProperty' '(' dataPropertyIRI ')' | 'AnnotationProperty' '(' annotationPropertyIRI ')' | 'NamedIndividual' '

Editor's Note: There are currently only short forms for four major datatypes. Other short forms could be added. Note that all datatypes can be accessed via their IRIs. Input on this aspect of the syntax is encouraged.

Editor's Note: The syntax for floating point literals is fairly restrictive and could be extended (for example to make the trailing f optional if there is an exponent. Note that all literals can be written using the long form. Input on this aspect of the syntax is encouraged.

2.2 Ontologies and Annotations

annotations ::= 'Annotations:' annotationAnnotatedList
annotation ::= annotationPropertyIRI annotationTarget
annotationTarget ::= nodeID | IRI | literal
ontologyDocument ::= { namespace } ontology
namespace ::= 'Namespace:' [ prefix ] full-IRI
ontology ::= 'Ontology:' [ ontologyIRI [ versionIRI ] ] [ import ] [ annotations ] [ frame ]
goalontologyIRI ::= IRI
The 'rdf', 'rdfs', 'owl', and 'xsd' prefixes are pre-defined as follows and cannot be changed.

Namespace: rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Namespace: rdfs <http://www.w3.org/2000/01/rdf-schema#>
Namespace: xsd <http://www.w3.org/2001/XMLSchema#>
Namespace: owl <http://www.w3.org/2002/07/owl#>

2.3 Property and Datatype Expressions

objectPropertyExpression ::= objectPropertyIRI | inverseObjectProperty
inverseObjectProperty ::= 'inverse' objectPropertyIRI
dataPropertyExpression ::= dataPropertyIRI
dataRange ::= dataConjunction 'or' dataConjunction { 'or' dataConjunction } |
| dataConjunction
dataConjunction ::= dataPrimary 'and' dataPrimary { 'and' dataPrimary } |
| dataPrimary
dataPrimary ::= [ 'not' ] dataAtomic
dataAtomic ::= Datatype |
| '{' literal { ',', literal } '}'
| datatypeRestriction | '(' dataRange ')'
datatypeRestriction ::= Datatype '[ facet restrictionValue { ',', facet restrictionValue } ]'
| facet ::= 'length' | 'minLength' | 'maxLength' | 'pattern' | 'langPattern' | '<'
restrictionValue ::= literal

In a datatypeRestriction, the facets and restrictionValues must be valid for the datatype, as in the OWL 2 Syntax [OWL 2 Syntax], after making the obvious change for the comparison facets.

2.4 Descriptions

description ::= conjunction 'or' conjunction { 'or' conjunction } |
| conjunction
conjunction ::= classIRI 'that' [ 'not' ] restriction { 'and' [ 'not' ] restriction } |
| primary 'and' primary { 'and' primary } |
| primary
primary ::= [ 'not' ] ( restriction | atomic )
restriction ::= objectPropertyExpression 'some' primary |
| objectPropertyExpression 'only' primary |
| objectPropertyExpression 'value' individual
objectPropertyExpression ::=
  'Self'
| objectPropertyExpression 'min' nonNegativeInteger [ primary ]
| objectPropertyExpression 'max' nonNegativeInteger [ primary ]
| objectPropertyExpression 'exactly' nonNegativeInteger [ primary ]
| dataPropertyExpression 'some' dataPrimary
| dataPropertyExpression 'only' dataPrimary
| dataPropertyExpression 'value' literal
| dataPropertyExpression 'min' nonNegativeInteger [ dataPrimary ]
| dataPropertyExpression 'max' nonNegativeInteger [ dataPrimary ]
| dataPropertyExpression 'exactly' nonNegativeInteger [ dataPrimary ]
atomic ::= classIRI
| '{' individual { ',' individual } '}'
| '{' description '}'

Editor’s Note: Some syntax in previous versions of the Manchester Syntax is no longer allowed (e.g., onlysome, xor, and ValuePartition). These were somewhat problematic. Input on including this syntax is welcome.

2.5 Frames and Miscellaneous

classFrame ::= 'Class:' classIRI
  { 'Annotations:' annotationAnnotatedList
  | 'SubClassOf:' descriptionAnnotatedList
  | 'EquivalentTo:' descriptionAnnotatedList
  | 'DisjointWith:' descriptionAnnotatedList
  | 'DisjointUnionOf:' annotations description2List }

objectPropertyFrame ::= 'ObjectProperty:' objectPropertyIRI
  { 'Annotations:' annotationAnnotatedList
  | 'Domain:' descriptionAnnotatedList
  | 'Range:' descriptionAnnotatedList
  | 'Characteristics:' objectPropertyCharacteristicAnnotatedList
  | 'SubPropertyOf:' objectPropertyExpressionAnnotatedList
  | 'EquivalentTo:' objectPropertyExpressionAnnotatedList
  | 'DisjointWith:' objectPropertyExpressionAnnotatedList
  | 'InverseOf:' objectPropertyExpressionAnnotatedList
  | 'SubPropertyChain:' annotations objectPropertyExpression 'o' objectPropertyExpression }

objectPropertyCharacteristic ::= 'Functional' | 'InverseFunctional'
  | 'Reflexive' | 'Irreflexive' | 'Symmetric' | 'Asymmetric' | 'Transitive'

dataPropertyFrame ::= 'DataProperty:' dataPropertyIRI
  { 'Annotations:' annotationAnnotatedList
  | 'Domain:' descriptionAnnotatedList
  | 'Range:' dataRangeAnnotatedList
  | 'Characteristics:' dataPropertyCharacteristicAnnotatedList
  | 'SubPropertyOf:' dataPropertyExpressionAnnotatedList
  | 'EquivalentTo:' dataPropertyExpressionAnnotatedList
  | 'SubPropertyChain:' annotations dataPropertyExpression 'o' dataPropertyExpression }

atomic ::= classIRI
| '{' individual { ',' individual } '}'
| '{' description '}'
2.6 Global Concerns

The Manchester syntax has the same restrictions on multiple use of IRIs as in OWL 2 [OWL 2 Syntax]. That is, in an ontology and the ontologies that it imports, no IRI can be used as more than one of an object property, a data property, or an annotation property; nor can a IRI be used as both a class and a datatype.

The Manchester syntax also has the same restriction on declaration of IRIs as does OWL 2. If a IRI is used as a property in an ontology then there must be a property frame for it in the ontology, or in an ontology that is imported by it, unless it is one of the built-in OWL 2 properties. If a IRI is used as a class in an ontology then there must be a class frame for it in the ontology, or in an ontology that is imported by it, unless it is one of the built-in OWL 2 classes. The only datatypes allowed are the built-in OWL 2 datatypes.
The Manchester syntax has the same global restrictions on the use of properties, anonymous individuals, and owl:topDataProperty as OWL 2 [OWL 2 Syntax] does. The details of these restrictions are complex, but one basic restriction is that no object property that is used in a number restriction (‘min’, ‘max’, or ‘exactly’) or ‘self’ restriction can be transitive or have a transitive property as a descendant sub-property, or be the inverse of or equivalent to such properties.

3 Quick Reference

This is a made-up partial ontology that provides a quick reference guide to the Manchester Syntax. Not all of the ontology makes logical sense so that all aspects of the syntax can be shown in a small example.

All colon-terminated keyword constructs except Ontology: (e.g., Import:, Class:, Domain:, SubClassOf:) are optional and can be repeated. Most keyword constructs take a comma-separated list of sub-constructs, which is sometimes indicated by ",...". Annotations are allowed for elements in these lists of sub-constructs except where annotations are explicitly noted (e.g., in DisjointUnionOf:, in DisjointClasses:).

Namespace: <http://ex.com/owl/families#>
Namespace: g <http://ex.com/owl2/families#>

Ontology: <http://example.com/owl/families> <http://example.com/owl2/families.owl>
Import: <http://ex.com/owl2/families.owl>
Annotations: creator John,
Annotations: rdfs:comment "Creation Year"
creationYear 2008,
mainClass Person

ObjectProperty: hasWife
Annotations: ...
Characteristics: Functional, InverseFunctional, Reflexive, Irreflexive,
Domain: Annotations: rdfs:comment "General domain",
creator John
Person,
Annotations: rdfs:comment "More specific domain"
Man
Range: Person, Woman
SubPropertyOf: hasSpouse, loves
EquivalentTo: isMarriedTo, ...
DisjointWith: hates, ...
InverseOf: hasSpouse, inverse hasSpouse
SubPropertyChain: Annotations: ... hasChild ○ hasParent ○...

DataProperty: hasAge
Annotations: ...
Characteristics: Functional
Domain: Person ,...  
Range: integer ,...  
SubPropertyOf: hasVerifiedAge ,...  
EquivalentTo: hasAgeInYears ,...  
DisjointWith: hasSSN ,... 

AnnotationProperty: creator  
Annotations: ...  
Domain: Person ,...  
Range: integer ,...  
SubPropertyOf: initialCreator ,... 

Class: Person  
Annotations: ...  
SubClassOf: owl:Thing that hasFirstName exactly 1 and hasFirstName only string[minLength 1] ,...  
SubClassOf: hasAge exactly 1 and hasAge only not integer[< 0] ,...  
SubClassOf: hasGender exactly 1 and hasGender only {female , male} ,...  
SubClassOf: hasSSN max 1, hasSSN min 1  
SubClassOf: not hates Self ,...  
EquivalentTo: g:People ,...  
DisjointWith: g:Rock , g:Mineral ,... 
DisjointUnionOf: Annotations: ... Child, Adult 

Individual: John  
Annotations: ...  
Types: Person , hasFirstName value "John" or hasFirstName value "Jack"^^xsd:string  
Facts: hasWife Mary, not hasChild Susan, hasAge 33, hasChild :child1  
SameAs: Jack ,...  
DifferentFrom: Susan ,... 

Individual: :child1  
Annotations: ... 
Types: Person ,...  
Facts: hasChild Susan ,... 

DisjointClasses: Annotations: ... g:Rock, g:Scissor, g:Paper 
EquivalentProperties: Annotations: ... hates, loathes, despises 
DisjointProperties: Annotations: ... hates, loves, indifferent 
EquivalentProperties: Annotations: ... favouriteNumber, g:favouriteNumber, g:favouriteInteger, favouriteReal 
DisjointProperties: Annotations: ... favouriteInteger, favouriteReal 
SameIndividual: Annotations: ... John, Jack, Joe, Jim 
DifferentIndividuals: Annotations: ... John, Susan, Mary, Jill 
HasKey: Annotations: ... hasSSN Person
4 Appendix: Translation to and from OWL 2 Functional-Style Syntax

Most of the translation between the Manchester OWL syntax and OWL 2 is obvious. The translation given here is with the OWL 2 Functional-Style Syntax [OWL 2 Syntax].

4.1 Informal Description

In many cases there is a one-to-one correspondence between the Manchester OWL syntax and the OWL 2 Functional-Style Syntax. For example, dataComplementOf in the Manchester OWL syntax corresponds directly to dataComplementOf in the OWL 2 Functional-Style Syntax. All that is required is to translate the keywords and adjust to a parenthesized syntax.

IRIs and their parts are the same in the Manchester OWL syntax and the OWL 2 Functional-Style Syntax, no change is needed for them, except that the "special" datatypes are translated into the corresponding XML Schema datatypes. Literals are mostly the same, but the abbreviated syntaxes for numbers and strings have to be translated in the obvious way. The syntax for data ranges in the Manchester OWL syntax corresponds exactly with the syntax in the OWL 2 Functional-Style Syntax.

The syntax for annotations in the Manchester OWL syntax closely corresponds to the syntax in the OWL 2 Functional-Style Syntax. The only special processing that needs to be done is to determine which frame to attach entity annotations to in the reverse mapping. Translating to the Functional-Style syntax and back again can thus loose some non-logical information in the Manchester syntax.

Descriptions also correspond closely between the Manchester OWL syntax and the OWL 2 Functional-Style Syntax.

The translation of frame axioms is performed by splitting them into pieces that correspond to single axioms. This is done by taking each of the pieces of the frame (Annotations:, Domain:, Range:, etc) and making new frames for each of them. The new frame is of the same kind (Class:, ObjectProperty:, etc.) and for the same IRI. Then each resultant frame that contains an AnnotatedList with more than one element is broken into a frame for each element of the list in a similar manner.

The resultant axioms and any miscellaneous axioms then correspond closely to the OWL 2 Functional-Style Syntax axioms and can be directly translated. The only special cases are that annotations directly in frames become annotations in entity annotation axioms and that (negative) property assertions have to be disambiguated depending on whether the property is an object property or a data property.
Translations of OWL 2 Functional-Style Syntax axioms back to frames can be done piecemeal or the axioms on a single entity can be all combined together, which is done here.

The remaining top-level constructs of an ontology (namespaces, imports, ontology annotations, and the ontology name) can be directly translated.

**4.2 Formal Description for Mapping to OWL 2 Functional-Style Syntax**

Formally the transformation takes an ontology in the Manchester OWL syntax and produces an ontology in the Functional-Style syntax. The transformation needs access to the imported ontologies.

First, for each frame in the ontology, produce the appropriate declaration as follows:

<table>
<thead>
<tr>
<th>Frame</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class: IRI ...</td>
<td>Declaration( Class(IRI) )</td>
</tr>
<tr>
<td>ObjectProperty: IRI ...</td>
<td>Declaration( ObjectProperty(IRI) )</td>
</tr>
<tr>
<td>DataProperty: IRI ...</td>
<td>Declaration( DataProperty(IRI) )</td>
</tr>
<tr>
<td>AnnotationProperty: IRI</td>
<td>Declaration( AnnotationProperty(IRI) )</td>
</tr>
<tr>
<td>Individual: IRI ...</td>
<td>Declaration( NamedIndividual(IRI) )</td>
</tr>
<tr>
<td>Individual: nodeID ...</td>
<td></td>
</tr>
</tbody>
</table>

Second, split up frames into single axioms in three stages. The first stage splits apart top-level pieces of frames that have multiple top-level pieces, transforming F: IRI p1 p2 ... into F: IRI p1 F: IRI p2 ... for F: one of the frame keywords (Class:, ...), until no more transformations are possible. The second stage splits apart the pieces of each of the top-level pieces, transforming F: IRI P: s1 s2 ... into F: IRI P: s1 F: IRI P: s2 ... for P: one of the keywords immediately inside a frame (Annotations:, SubClassOf:, ...), until no more transformations are possible. The third stage just removes any frame containing only a IRI.

Next, perform the actual syntax transformation. Any piece of syntax with no transformation listed here is just copied through.

<table>
<thead>
<tr>
<th>Nonterminal</th>
<th>Form</th>
<th>Transformation (T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datatype</td>
<td>integer</td>
<td>xsd:integer</td>
</tr>
<tr>
<td>Datatype</td>
<td>decimal</td>
<td>xsd:decimal</td>
</tr>
<tr>
<td>Datatype</td>
<td>float</td>
<td>xsd:float</td>
</tr>
<tr>
<td>Datatype</td>
<td>string</td>
<td>xsd:string</td>
</tr>
<tr>
<td>integerLiteral</td>
<td>integer</td>
<td>integer^^xsd:integer</td>
</tr>
<tr>
<td>decimalLiteral</td>
<td>decimal</td>
<td>decimal^^xsd:decimal</td>
</tr>
<tr>
<td>floatingPointLiteral</td>
<td>float</td>
<td>float^^xsd:float</td>
</tr>
<tr>
<td>abbreviatedXSDStringLiteral</td>
<td>string</td>
<td>string</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>abbreviatedXSDStringLiteral</td>
<td><code>string@tag</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>length xsd:length</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>minLength xsd:minLength</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>maxLength xsd:maxLength</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>pattern xsd:pattern</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>&lt; xsd:minExclusive</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>&lt;= xsd:minInclusive</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>&gt;= xsd:maxInclusive</code></td>
<td></td>
</tr>
<tr>
<td>facet</td>
<td><code>&gt;</code> xsd:maxExclusive</td>
<td></td>
</tr>
<tr>
<td>datatypeRestriction</td>
<td>Datatype{facet-value list} DatatypeRestriction(T(datatype) T(facet-value list))</td>
<td></td>
</tr>
<tr>
<td>dataAtomic</td>
<td><code>{ literal list } OneOf(T(literal list))</code></td>
<td></td>
</tr>
<tr>
<td>dataPrimary</td>
<td>(dataRange) T(dataRange)</td>
<td></td>
</tr>
<tr>
<td>dataPrimary</td>
<td>dataAtomic T(dataAtomic)</td>
<td></td>
</tr>
<tr>
<td>dataPrimary</td>
<td>hot dataAtomic ComplementOf(T(dataAtomic))</td>
<td></td>
</tr>
<tr>
<td>dataConjunction</td>
<td>dataPrimary and ... IntersectionOf(T(dataPrimary) ...)</td>
<td></td>
</tr>
<tr>
<td>dataConjunction</td>
<td>dataPrimary T(dataPrimary)</td>
<td></td>
</tr>
<tr>
<td>dataRange</td>
<td>dataConjunction or ... UnionOf(T(dataConjunction) ...)</td>
<td></td>
</tr>
<tr>
<td>dataRange</td>
<td>dataConjunction T(dataConjunction)</td>
<td></td>
</tr>
<tr>
<td>inverseObjectProperty</td>
<td>inverse objectPropertyExpression InverseProperty(T(objectPropertyExpression))</td>
<td></td>
</tr>
<tr>
<td>atomic</td>
<td>{individual list} OneOf(T(individual list))</td>
<td></td>
</tr>
<tr>
<td>atomic</td>
<td>(description) T(description)</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression some primary SomeValuesFrom(T(objectPropertyExpression) T(primary))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression only primary AllValuesFrom(T(objectPropertyExpression) T(primary))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression value individual HasValue(T(objectPropertyExpression) individual)</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression min nni MinCardinality(T(objectPropertyExpression) nni)</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression min nni primary MinCardinality(T(objectPropertyExpression) nni T(primary))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression exactly nni ExactCardinality(T(objectPropertyExpression) nni)</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression exactly nni primary ExactCardinality(T(objectPropertyExpression) nni T(primary))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression max nni MaxCardinality(T(objectPropertyExpression) nni)</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression max nni primary MaxCardinality(T(objectPropertyExpression) nni T(primary))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>objectPropertyExpression Self HasSelf(T(objectPropertyExpression))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression some dataRange SomeValuesFrom(T(dataPropertyExpression) T(dataRange))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression only dataRange AllValuesFrom(T(dataPropertyExpression) T(dataRange))</td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression value literal HasValue(T(dataPropertyExpression) T(literal))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dataPropertyExpression</td>
<td>MinCardinality(T(dataPropertyExpression) nni)</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression</td>
<td>MinCardinality(T(dataPropertyExpression) nni T(dataRange))</td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression</td>
<td>ExactCardinality(T(dataPropertyExpression) nni T(dataRange))</td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression</td>
<td>MaxCardinality(T(dataPropertyExpression) nni T(dataRange))</td>
</tr>
<tr>
<td>restriction</td>
<td>dataPropertyExpression</td>
<td>MaxCardinality(T(dataPropertyExpression) nni T(dataRange))</td>
</tr>
<tr>
<td>primary</td>
<td>atomic</td>
<td>T(atomic)</td>
</tr>
<tr>
<td>primary</td>
<td>not atomic</td>
<td>ComplementOf(T(atomic))</td>
</tr>
<tr>
<td>conjunction</td>
<td>classIRI that primary</td>
<td>IntersectionOf(classIRI T(primary) ...)</td>
</tr>
<tr>
<td>conjunction</td>
<td>primary and</td>
<td>IntersectionOf(T(primary) ...)</td>
</tr>
<tr>
<td>conjunction</td>
<td>primary</td>
<td>T(primary)</td>
</tr>
<tr>
<td>description</td>
<td>conjunction or</td>
<td>UnionOf(T(conjunction) ...)</td>
</tr>
<tr>
<td>annotation</td>
<td>annotations</td>
<td>Annotation(T(annotations) annotationPropertyIRI T(target))</td>
</tr>
<tr>
<td>annotations</td>
<td>Annotations: annotation</td>
<td>Annotation(T(annotations) ...</td>
</tr>
<tr>
<td>classFrame</td>
<td>Class: IRI Annotations:</td>
<td>AnnotationAssertion(T(annotations) annotationPropertyIRI IRI T(target))</td>
</tr>
<tr>
<td>classFrame</td>
<td>Class: IRI SubClassOf:</td>
<td>SubClassOf(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>classFrame</td>
<td>Class: IRI EquivalentTo:</td>
<td>EquivalentClasses(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>classFrame</td>
<td>Class: IRI DisjointWith:</td>
<td>DisjointClasses(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>classFrame</td>
<td>Class: IRI DisjointUnionOf:</td>
<td>DisjointUnion(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Annotations: annotationPropertyIRI target</td>
<td>AnnotationAssertion(T(annotations) annotationPropertyIRI IRI T(target))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Domain: annotations description</td>
<td>PropertyDomain(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Range: annotations description</td>
<td>PropertyRange(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Functional</td>
<td>FunctionalProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations InverseFunctional</td>
<td>InverseFunctionalProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Reflexive</td>
<td>ReflexiveProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Irreflexive</td>
<td>IrreflexiveProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Symmetric</td>
<td>SymmetricProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Asymmetric</td>
<td>AsymmetricProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI Characteristics: annotations Transitive</td>
<td>TransitiveProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI SubPropertyOf: annotations objectPropertyExpression</td>
<td>SubPropertyOf(T(annotations) IRI T(objectPropertyExpression))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI EquivalentTo: annotations objectPropertyExpression</td>
<td>EquivalentProperties(T(annotations) IRI T(objectPropertyExpression))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI DisjointWith: annotations objectPropertyExpression</td>
<td>DisjointProperties(T(annotations) IRI T(objectPropertyExpression))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI InverseOf: annotations objectPropertyExpression</td>
<td>InverseProperties(T(annotations) IRI T(objectPropertyExpression))</td>
</tr>
<tr>
<td>objectPropertyFrame</td>
<td>ObjectProperty: IRI SubPropertyChain: objectPropertyExpression</td>
<td>SubPropertyOf(PropertyChain(T(objectPropertyExpression) ...) IRI)</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI Annotations: annotations annotationPropertyIRI target</td>
<td>AnnotationAssertion(T(annotations) annotationPropertyIRI IRI T(target))</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI Domain: annotations description</td>
<td>PropertyDomain(T(annotations) IRI T(description))</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI Range: annotations dataRange</td>
<td>PropertyRange(T(annotations) IRI T(dataRange))</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI Characteristics: annotations Functional</td>
<td>FunctionalProperty(T(annotations) IRI)</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI SubPropertyOf: annotations objectPropertyExpression</td>
<td>SubPropertyOf(T(annotations) IRI T(objectPropertyExpression))</td>
</tr>
<tr>
<td>dataPropertyFrame</td>
<td>DataProperty: IRI EquivalentTo:</td>
<td>EquivalentProperties(T(annotations) IRI T(dataPropertyExpression))</td>
</tr>
<tr>
<td>Frame Type</td>
<td>Property</td>
<td>Expression</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td><code>dataPropertyFrame</code></td>
<td>DataProperty: IRI</td>
<td>DisjointWith: annotations dataPropertyExpression</td>
</tr>
<tr>
<td><code>annotationPropertyFrame</code></td>
<td>AnnotationProperty: IRI</td>
<td>Annotations: annotations annotationPropertyIRI target</td>
</tr>
<tr>
<td><code>annotationPropertyFrame</code></td>
<td>AnnotationProperty: IRI</td>
<td>Domain: annotations IRI</td>
</tr>
<tr>
<td><code>annotationPropertyFrame</code></td>
<td>AnnotationProperty: IRI</td>
<td>Range: annotations IRI</td>
</tr>
<tr>
<td><code>annotationPropertyFrame</code></td>
<td>AnnotationProperty: IRI</td>
<td>SubPropertyOf: annotations annotationPropertyIRI</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: IRI</td>
<td>Annotations: annotations annotationPropertyIRI target</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: nodeID</td>
<td>Annotations: annotations annotation</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>Types: annotations description</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>Facts: annotations objectPropertyIRI individual2</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>Facts: annotations not objectPropertyIRI individual2</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>Facts: annotations dataPropertyIRI literal</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>Facts: annotations not dataPropertyIRI literal</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>SameAs: annotations individual2</td>
</tr>
<tr>
<td><code>individualFrame</code></td>
<td>Individual: individual</td>
<td>DifferentFrom: annotations individual2</td>
</tr>
<tr>
<td><code>misc</code></td>
<td>EquivalentClasses: annotations descriptions</td>
<td></td>
</tr>
<tr>
<td><code>misc</code></td>
<td>DisjointClasses: annotations descriptions</td>
<td></td>
</tr>
<tr>
<td><code>misc</code></td>
<td>EquivalentProperties: annotations objectProperties</td>
<td></td>
</tr>
</tbody>
</table>
Finally, put the declarations produced in the first step into the ontology.

### 4.3 Formal Description for Mapping from OWL 2 Functional-Style Syntax

The mapping from the Functional-Style Syntax back to the Manchester Syntax essentially just runs the above translation in reverse.

First, create a trivial frame containing only a IRI for each declaration in the ontology. Second, turn the Functional-Style Syntax into the Manchester Syntax by running the syntax transformation above in reverse. The non-determinism in the mapping of entity annotations is resolved by uniformly making them annotations in individual frames. Third, collapse frames for the same entity into one frame by running that part of the forward transformation in reverse. This step does not affect the meaning of an ontology and is thus optional.

### 5 Appendix: Internet Media Type, File Extension and Macintosh File Type

**Contact**
- Sandro Hawke

**See also**
How to Register a Media Type for a W3C Specification Internet Media Type registration, consistency of use TAG Finding 3 June 2002 (Revised 4 September 2002)

The Internet Media Type / MIME Type for the OWL Manchester Syntax is "text/owl-manchester".

It is recommended that OWL Manchester Syntax files have the extension ".omn" (all lowercase) on all platforms.

It is recommended that OWL Manchester Syntax files stored on Macintosh HFS file systems be given a file type of "TEXT".

The information that follows will be submitted to the IESG for review, approval, and registration with IANA.

**Type name**
- text

**_subtype name**
- owl-manchester

**Required parameters**
- None

**Optional parameters**
- charset This parameter may be required when transferring non-ascii data across some protocols. If present, the value of charset is always UTF-8.

**Encoding considerations**
The syntax of the OWL Manchester Syntax is expressed over code points in Unicode [UNICODE]. The encoding is always UTF-8 [RFC3629].

**Security considerations**
The OWL Manchester Syntax uses IRIs as term identifiers. Applications interpreting data expressed in the OWL Manchester Syntax should address the security issues of Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8, as well as Uniform Resource Identifiers (URI): Generic Syntax [RFC3986] Section 7. Multiple IRIs may have the same appearance. Characters in different scripts may look similar (a Cyrillic "о" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (LATIN SMALL LETTER E followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER E WITH ACUTE). Any person or application that is writing or interpreting data in the OWL Manchester Syntax must take care to use the IRI that matches the intended semantics, and avoid IRIs that may look similar. Further information about matching of similar characters can be found in Unicode Security Considerations [UNISEC] and Internationalized Resource Identifiers (IRIs) [RFC3987] Section 8.

**Interoperability considerations**
There are no known interoperability issues.

**Published specification**
This specification.

**Applications which use this media type**
This media type is used by Protege 4 and TopBraid Composer.
Additional information
None.

Magic number(s)
OWL Manchester Syntax documents may have the strings 'Namespace:' or 'Ontology:' (case dependent) near the beginning of the document.

File extension(s)
".omn"

Base URI
There are no constructs in the OWL Manchester Syntax to change the Base URI.

Macintosh file type code(s)
"TEXT"

Person & email address to contact for further information
Sandro Hawke <sandro@w3.org>

Intended usage
COMMON

Restrictions on usage
None

Author/Change controller
The OWL Manchester Syntax is the product of the W3C OWL Working Group in cooperation with OWL ontology tool builders; the specification may be extended by groups of OWL tool builders; W3C reserves change control over this specification.

6 References

[CURIE]

[Manchester OWL DL Syntax]

[OWL Semantics and Abstract Syntax]

[OWL 2 Primer]

[OWL 2 Syntax]
[OWL 2 XML Syntax]

[Protégé 4]

[RDF Syntax]

[RDF Test Cases]

[RFC3629]

[RFC3986]

[RFC3987]

[BCP 47]

[TopBraid Composer]

[XML Namespaces]

[UNICODE]

[UNISEC]