W3C Editor's Draft

OWL 2 Web Ontology Language: Manchester Syntax

W3C Editor's Draft 02 December 2008

This version:
http://www.w3.org/2007/OWL/draft/ED-owl2-manchester-syntax-20081202/
Latest editor's draft:
http://www.w3.org/2007/OWL/draft/owl2-manchester-syntax/
Previous version:
http://www.w3.org/2007/OWL/draft/ED-owl2-manchester-syntax-20081128/
(color-coded diff)
Authors:
Matthew Horridge, University of Manchester
Peter F. Patel-Schneider, Bell Labs Research, Alcatel-Lucent

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

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May Be Superseded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 <a href="http://www.w3.org/2005/10/Process-20051014/tr.html#q75">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.

W3C Editor's Draft

No Endorsement

Publication as a Working 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. 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.

Contents

- 1 Introduction
- 2 The Grammar
  - 2.1 IRIs, Integers, Literals, and Entities
  - 2.2 Ontologies and Annotations
  - 2.3 Property and Datatype Expressions
  - 2.4 Descriptions
  - 2.5 Frames and Miscellaneous
  - 2.6 Global Concerns
- 3 Quick Reference
- 4 Appendix: Translation to and from OWL 2 Functional-Style Syntax
  - 4.1 Informal Description
  - 4.2 Formal Description for Mapping to OWL 2 Functional-Style Syntax
  - 4.3 Formal Description for Mapping from OWL 2 Functional-Style Syntax
- 5 Appendix: Internet Media Type, File Extension and Macintosh File Type
- 6 References

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

[*OWL 2 Syntax*]. 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

Construct	Syntax	Example
nonterminal symbols	boldface	**ClassExpression**
terminal symbols	single quoted	'PropertyRange'
zero or more	curly braces	{ **ClassExpression** }
zero or one	square brackets	[**ClassExpression**]
alternative	vertical bar	**Assertion** | **Declaration**

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.

```
full-IRI := 'IRI as defined in [RFC3987], enclosed in a pair of < (U+3C) and > (U
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'
```

W3C Editor's Draft

```
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 specific

literal ::= typedLiteral | abbreviatedXSDStringLiteral | abbreviatedRDFTextLiteral | integerLiter
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) 
floatingPointLiteral ::= [ '+' | '-'] ( digits ['.'digits] [exponent] | '.' digits[expone
exponent ::= ('e' | 'E') ['+' | '-'] digits
decimalLiteral ::= ['+' | '-'] digits '.' digits
integerLiteral ::= ['+' | '-'] digits

entity ::= 'Datatype' '(' datatypeIRI ')' | 'Class' '(' classIRI ')'
        | 'ObjectProperty' '(' objectPropertyIRI ')' | 'DataProperty' '('dataPropert
        | '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 } { fr
ontologyIRI ::= IRI
```

```
versionIRI ::= IRI
import ::= 'Import:' IRI
frame ::= classFrame | objectPropertyFrame | dataPropertyFrame | annotationPropertyFrame
```

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
```

W3C Editor's Draft

```
            | 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' objectPropertyExp
                                          { 'o' objectPropertyExpression } }

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

dataPropertyFrame ::= 'DataProperty:' dataPropertyIRI
        { 'Annotations:'      annotationAnnotatedList
        | 'Domain:'           descriptionAnnotatedList
        | 'Range:'            dataRangeAnnotatedList
```

W3C Editor's Draft

```
        | 'Characteristics:' annotations 'Functional'
        | 'SubPropertyOf:'    dataPropertyExpressionAnnotatedList
        | 'EquivalentTo:'     dataPropertyExpressionAnnotatedList
        | 'DisjointWith:'     dataPropertyExpressionAnnotatedList }

annotationPropertyFrame ::= 'AnnotationProperty:' annotationPropertyIRI
        { 'Annotations:'      annotationAnnotatedList }
        | 'Domain:'           IRIAnnotatedList
        | 'Range:'            IRIAnnotatedList
        | 'SubPropertyOf:'    annotationPropertyIRIAnnotatedList

individualFrame ::= 'Individual:' individual
        { 'Annotations:'    annotationAnnotatedList
        | 'Types:'          descriptionAnnotatedList
        | 'Facts:'          factAnnotatedList
        | 'SameAs:'         individualAnnotatedList
        | 'DifferentFrom:'  individualAnnotatedList }

fact ::= [ 'not' ] (objectPropertyFact | dataPropertyFact)
objectPropertyFact ::= objectPropertyIRI individual
dataPropertyFact ::= dataPropertyIRI literal

misc ::= 'EquivalentClasses:' annotations description2List
        | 'DisjointClasses:' annotations description2List
        | 'EquivalentProperties:' annotations objectProperty2List
        | 'DisjointProperties:' annotations objectProperty2List
        | 'EquivalentProperties:' annotations dataProperty2List
        | 'DisjointProperties:' annotations dataProperty2List
        | 'SameIndividual:' annotations individual2List
        | 'DifferentIndividuals:' annotations individual2List
        | 'HasKey:' description annotations ( objectPropertyExpression | dataPropertyExpre
                                               { objectPropertyExpression | c
```

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.

W3C Editor's Draft

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/owl/familie
  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 o hasParent o...

  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
  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"^
  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: ... favoriteNumber, g:favouriteNumber,
DisjointProperties: Annotations: ... favoriteInteger, 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:

Frame	Declaration
Class: *IRI* ...	Declaration(Class(*IRI*))
ObjectProperty: *IRI* ...	Declaration(ObjectProperty(*IRI*))
DataProperty: *IRI* ...	Declaration(DataProperty(*IRI*))
AnnotationProperty: *IRI* ...	Declaration(AnnotationProperty(*IRI*))
Individual: *IRI* ...	Declaration(NamedIndividual(*IRI*))
Individual: *nodeID* ...	

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.

Nonterminal	Form	Transformation (*T*)
Datatype	integer	xsd:integer
Datatype	decimal	xsd:decimal
Datatype	float	xsd:float
Datatype	string	xsd:string
integerLiteral	*integer*	"*integer*"^^xsd:integer
decimalLiteral	*decimal*	"*decimal*"^^xsd:decimal
floatingPointLiteral	*float*	"*float*"^^xsd:float
abbreviatedXSDStringLiteral	*string*	*string*

abbreviatedXSDStringLiteral	*string@tag*	*string@tag*
facet	length	xsd:length
facet	minLength	xsd:minLength
facet	maxLength	xsd:maxLength
facet	pattern	xsd:pattern
facet	langPattern	rdf:langPattern
facet	<=	xsd:minInclusive
facet	<	xsd:minExclusive
facet	>=	xsd:maxInclusive
facet	>	xsd:maxExclusive
datatypeRestriction	*Datatype*[*facet-value list*]	DatatypeRestriction(*T(datatype) T(facet-value list)*)
dataAtomic	{ *literal list* }	OneOf(*T(literal list)*)
dataAtomic	(*dataRange*)	*T(dataRange)*
dataPrimary	*dataAtomic*	*T(dataAtomic)*
dataPrimary	not *dataAtomic*	ComplementOf(*T(dataAtomic)*)
dataConjunction	*dataPrimary* and ...	IntersectionOf(*T(dataPrimary)* ...)
dataConjunction	*dataPrimary*	*T(dataPrimary)*
dataRange	*dataConjunction* or ...	UnionOf(*T(dataConjunction)* ...)
dataRange	*dataConjunction*	*T(dataConjunction)*
inverseObjectProperty	inverse *objectPropertyExpression*	InverseProperty(*T(objectPropertyExpression)*)
atomic	{*individual list*}	OneOf(*T(individual list)*)
atomic	(*description*)	*T(description)*
restriction	*objectPropertyExpression* some *primary*	SomeValuesFrom(*T(objectPropertyExpression) T(primary)*)
restriction	*objectPropertyExpression* only *primary*	AllValuesFrom(*T(objectPropertyExpression) T(primary)*)
restriction	*objectPropertyExpression* value *individual*	HasValue(*T(objectPropertyExpression) individual*)
restriction	*objectPropertyExpression* min *nni*	MinCardinality(*T(objectPropertyExpression) nni*)
restriction	*objectPropertyExpression* min *nni primary*	MinCardinality(*T(objectPropertyExpression) nni T(primary)*)
restriction	*objectPropertyExpression* exactly *nni*	ExactCardinality(*T(objectPropertyExpression) nni*)
restriction	*objectPropertyExpression* exactly *nni primary*	ExactCardinality(*T(objectPropertyExpression) nni T(primary)*)
restriction	*objectPropertyExpression* max *nni*	MaxCardinality(*T(objectPropertyExpression) nni*)
restriction	*objectPropertyExpression* max *nni primary*	MaxCardinality(*T(objectPropertyExpression) nni T(primary)*)
restriction	*objectPropertyExpression* Self	HasSelf(*T(objectPropertyExpression)*)
restriction	*dataPropertyExpression* some *dataRange*	SomeValuesFrom(*T(dataPropertyExpression) T(dataRange)*)
restriction	*dataPropertyExpression* only *dataRange*	AllValuesFrom(*T(dataPropertyExpression) T(dataRange)*)
restriction	*dataPropertyExpression* value *literal*	HasValue(*T(dataPropertyExpression) T(literal)*)

restriction	*dataPropertyExpression* min *nni*	MinCardinality(*T(dataPropertyExpression) nni*)
restriction	*dataPropertyExpression* min *nni dataRange*	MinCardinality(*T(dataPropertyExpression) nni T(dataRange)*)
restriction	*dataPropertyExpression* exactly *nni*	ExactCardinality(*T(dataPropertyExpression) nni*)
restriction	*dataPropertyExpression* exactly *nni dataRange*	ExactCardinality(*T(dataPropertyExpression) nni T(dataRange)*)
restriction	*dataPropertyExpression* max *nni*	MaxCardinality(*T(dataPropertyExpression) nni*)
restriction	*dataPropertyExpression* max *nni dataRange*	MaxCardinality(*T(dataPropertyExpression) nni T(dataRange)*)
primary	*atomic*	*T(atomic)*
primary	not *atomic*	ComplementOf(*T(atomic)*)
conjunction	*classIRI* that *primary* ...	IntersectionOf(*classIRI T(primary)* ...)
conjunction	*primary* and ...	IntersectionOf(*T(primary)* ...)
conjunction	*primary*	*T(primary)*
description	*conjunction* or ...	UnionOf(*T(conjunction)* ...)
description	*conjunction*	*T(conjunction)*
annotation	*annotations annotationPropertyIRI target*	Annotation(T(*annotations*) *annotationPropertyIRI* T(*target*))
annotations		
annotations	Annotations: *annotation* ...	Annotation(T(*annotation*) ...
classFrame	Class: *IRI* Annotations: *annotations annotationPropertyIRI target*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*))
classFrame	Class: *IRI* SubClassOf: *annotations description*	SubClassOf(*T(annotations) IRI T(description)*)
classFrame	Class: *IRI* EquivalentTo: *annotations description*	EquivalentClasses(*T(annotations) IRI T(description)*)
classFrame	Class: *IRI* DisjointWith: *annotations description*	DisjointClasses(*T(annotations) IRI T(description)*)
classFrame	Class: *IRI* DisjointUnionOf: *annotations descriptions*	DisjointUnion(*T(annotations) IRI T(description)*)
objectPropertyFrame	ObjectProperty: *IRI* Annotations: *annotations annotationPropertyIRI target*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*))
objectPropertyFrame	ObjectProperty: *IRI* Domain: *annotations description*	PropertyDomain(*T(annotations) IRI T(description)*)
objectPropertyFrame	ObjectProperty: *IRI* Range: *annotations description*	PropertyRange(*T(annotations) IRI T(description)*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Functional	FunctionalProperty(*T(annotations) IRI*)

objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* InverseFunctional	InverseFunctionalProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Reflexive	ReflexiveProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Irreflexive	IrreflexiveProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Symmetric	SymmetricProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Asymmetric	AsymmetricProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* Characteristics: *annotations* Transitive	TransitiveProperty(*T(annotations) IRI*)
objectPropertyFrame	ObjectProperty: *IRI* SubPropertyOf: *annotations* objectPropertyExpression	SubPropertyOf(*T(annotations) IRI T(objectPropertyExpression)*)
objectPropertyFrame	ObjectProperty: *IRI* EquivalentTo: *annotations* objectPropertyExpression	EquivalentProperties(*T(annotations) IRI T(objectPropertyExpression)*)
objectPropertyFrame	ObjectProperty: *IRI* DisjointWith: *annotations* objectPropertyExpression	DisjointProperties(*T(annotations) IRI T(objectPropertyExpression)*)
objectPropertyFrame	ObjectProperty: *IRI* InverseOf: *annotations* objectPropertyExpression	InverseProperties(*T(annotations) IRI T(objectPropertyExpression)*)
objectPropertyFrame	ObjectProperty: *IRI* SubPropertyChain: *objectPropertyExpression* o ...	SubPropertyOf(PropertyChain(*T(objectPropertyExpression)* ...) *IRI*)
dataPropertyFrame	DataProperty: *IRI* Annotations: *annotations annotationPropertyIRI target*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*))
dataPropertyFrame	DataProperty: *IRI* Domain: *annotations description*	PropertyDomain(*T(annotations) IRI T(description)*)
dataPropertyFrame	DataProperty: *IRI* Range: *annotations dataRange*	PropertyRange(*T(annotations) IRI T(dataRange)*)
dataPropertyFrame	DataProperty: *IRI* Characteristics: *annotations* Functional	FunctionalProperty(*T(annotations) IRI*)
dataPropertyFrame	DataProperty: *IRI* SubPropertyOf: *annotations* dataPropertyExpression	SubPropertyOf(*T(annotations) IRI T(dataPropertyExpression)*)
dataPropertyFrame	DataProperty: *IRI* EquivalentTo:	EquivalentProperties(*T(annotations) IRI T(dataPropertyExpression)*)

	annotations dataPropertyExpression	
dataPropertyFrame	DataProperty: *IRI* DisjointWith: *annotations* dataPropertyExpression	DisjointProperties(*T(annotations) IRI T(dataPropertyExpression)*)
annotationPropertyFrame	AnnotationProperty: *IRI* Annotations: *annotations annotationPropertyIRI target*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*))
annotationPropertyFrame	AnnotationProperty: *IRI* Domain: *annotations IRI*	PropertyDomain(*T(annotations) IRI IRI*)
annotationPropertyFrame	AnnotationProperty: *IRI* Range: *annotations IRI*	PropertyRange(*T(annotations) IRI IRI*)
annotationPropertyFrame	AnnotationProperty: *IRI* SubPropertyOf: *annotations* annotationPropertyIRI	SubPropertyOf(*T(annotations) IRI T(annotationPropertyIRI)*)
individualFrame	Individual: *IRI* Annotations: *annotations annotationPropertyIRI target*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI IRI* T(*target*))
individualFrame	Individual: *nodeID* Annotations: *annotations annotation*	AnnotationAssertion(T(*annotations*) *annotationPropertyIRI nodeID* T(*target*))
individualFrame	Individual: *individual* Types: *annotations description*	ClassAssertion(*T(annotations) T(description) individual*)
individualFrame	Individual: *individual* Facts: *annotations objectPropertyIRI individual2*	PropertyAssertion(*T(annotations) objectPropertyIRI individual individual2*)
individualFrame	Individual: *individual* Facts: *annotations* not *objectPropertyIRI individual2*	NegativePropertyAssertion(*T(annotations) objectPropertyIRI individual individual2*)
individualFrame	Individual: *individual* Facts: *annotations dataPropertyIRI literal*	PropertyAssertion(*T(annotations)* dataPropertyIRI individual T(literal)*)*
individualFrame	Individual: *individual* Facts: *annotations* not *dataPropertyIRI literal*	NegativePropertyAssertion(*T(annotations)* dataPropertyIRI individual T(literal)*)*
individualFrame	Individual: *individual* SameAs: *annotations individual2*	SameIndividual(*T(annotations) individual individual2*)
individualFrame	Individual: *individual* DifferentFrom: *annotations individual2*	DifferentIndividuals(*T(annotations) individual individual2*)
misc	EquivalentClasses: *annotations descriptions*	EquivalentClasses(*T(annotations) T(descriptions)*)
misc	DisjointClasses: *annotations descriptions*	DisjointClasses(*T(annotations) T(descriptions)*)
misc	EquivalentProperties: *annotations objectProperties*	EquivalentProperties(*T(annotations) T(objectProperties)*)

misc	DisjointProperties: *annotations objectProperties*	DisjointProperties(*T(annotations) T(objectProperties)*)
misc	EquivalentProperties: *annotations dataProperties*	EquivalentProperties(*T(annotations) T(dataProperties)*)
misc	DisjointProperties: *annotations dataProperties*	DisjointProperties(*T(annotations) T(dataProperties)*)
misc	SameIndividual: *annotations individuals*	SameIndividual(*T(annotations) individuals*)
misc	DifferentIndividuals: *annotations individuals*	DifferentIndividuals(*T(annotations) individuals*)
misc	HasKey: *annotations description properties*	HasKey(*T(annotations) T(description) T(properties)*)
namespace	Namespace: *prefix Full-IRI*	Namespace(*prefix* = *Full-IRI*)
namespace	Namespace: *Full-IRI*	Namespace(= *Full-IRI*)
import	Import: *IRI*	Import(*IRI*)
ontology	Ontology: *IRI IRI imports annotations frames*	Ontology(*IRI IRI T(imports) T(annotations) T(frames)*)
ontology	Ontology: *IRI imports annotations frames*	Ontology(*IRI T(imports) T(annotations) T(frames)*)
ontology	Ontology: *imports annotations frames*	Ontology(*T(imports) T(annotations) T(frames)*)
ontologyDocument	*namespaces ontology*	*T(namespaces) T(ontology)*

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 transfering 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 "o" 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 Mancheser 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]
: *CURIE Syntax 1.0: A syntax for expressing Compact URIs*. M. Birbeck, S. McCarron, Editors, W3C Working Draft, 26 November 2007, http://www.w3.org/TR/2007/WD-curie-20071126/.

[Manchester OWL DL Syntax]
: *The Manchester OWL Syntax*. Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, Hai H. Wang. OWL Experiences and Directions Workshop, 2006.

[OWL Semantics and Abstract Syntax]
: *OWL Web Ontology Language: Semantics and Abstract Syntax*. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, eds. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/. Latest version available at http://www.w3.org/TR/owl-semantics/.

[OWL 2 Primer]
: *OWL 2 Web Ontology Language: Primer*. Bijan Parsia and Peter F. Patel-Schneider. 2008.

[OWL 2 Syntax]
: *OWL 2 Web Ontology Language:Structural Specification and Functional-Style Syntax* Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. W3C Editor's Draft, 02 December 2008, http://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20081202/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-syntax/.

W3C Editor's Draft

[OWL 2 XML Syntax]
OWL 2 Web Ontology Language:XML Serialization Boris Motik, Peter Patel-Schneider, eds. W3C Editor's Draft, 02 December 2008, http://www.w3.org/2007/OWL/draft/ED-owl2-xml-serialization-20081202/. Latest version available at http://www.w3.org/2007/OWL/draft/owl2-xml-serialization/.

[Protégé 4]
Protégé 4 User Documentation. http://protegewiki.stanford.edu/index.php/Protege4UserDocs, October 2008.

[RDF Syntax]
RDF/XML Syntax Specification (Revised). Dave Beckett, Editor, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/rdf-syntax-grammar/.

[RDF Test Cases]
RDF Test Cases. Jan Grant and Dave Beckett, Editors, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-testcases/.

[RFC3629]
UTF-8, a transformation format of ISO 10646, F. Yergeau, November 2003, http://www.ietf.org/rfc/rfc3629.txt

[RFC3986]
RFC 3986 *Uniform Resource Identifier (URI): Generic Syntax*, T. Berners-Lee, R. Fielding, L. Masinter, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC3987]
RFC 3987 - Internationalized Resource Identifiers (IRIs). M. Duerst, M. Suignard. IETF, January 2005, http://www.ietf.org/rfc/rfc3987.txt.

[BCP 47]
BCP 47 - Tags for Identifying Languages. A. Phillips, M. Davis, eds., IETF, September 2006, http://www.rfc-editor.org/rfc/bcp/bcp47.txt.

[TopBraid Composer]
TopBraid Composer Home Page. http://www.topquadrant.com/topbraid/composer/, October 2008.

[XML Namespaces]
Namespaces in XML 1.0 (Second Edition). Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. 16 August 2006.

[UNICODE]
The Unicode Standard Version 3.0, Addison Wesley, Reading MA, 2000, ISBN: 0-201-61633-5. http://www.unicode.org/unicode/standard/standard.html

[UNISEC]
Unicode Security Considerations, Mark Davis, Michel Suignard, July 2008, http://www.unicode.org/reports/tr36/