A LaTeX-Style Syntax for OWL 2

Matteo Matassoni
University of Trento – DISI
#150629
<m.matassoni.1@studenti.unitn.it>

Contents

1 Preliminary Definitions

The grammar presented in this document uses the following two "special" terminal symbols, which affect the process of transforming an input sequence of characters into a sequence of regular (i.e., not "special") terminal symbols:

- whitespace is a nonempty sequence of space (U+20), horizontal tab (U+9), line feed (U+A), or carriage return (U+D) characters, and
- a comment is a sequence of characters that starts with the % (U+25) character and does not contain the line feed (U+A) or carriage return (U+D) characters.

2 General Definitions

⟨*nonNegativeInteger*⟩ ::= a nonempty finite sequence of digits between 0 and 9

⟨*quotedString*⟩ ::= a finite sequence of characters in which ‘’ (U+22) and \ (U+5C) occur only in pairs of the form \‘’ (U+5C, U+22) and \\ (U+5C, U+5C), enclosed in a pair of ‘’ (U+22) characters

⟨*languageTag*⟩ ::= @ (U+40) followed a nonempty sequence of characters matching the langtag production from [2]

⟨*nodeID*⟩ ::= a finite sequence of characters matching the BLANK_NODE_LABEL production of [3]

⟨*fullIRI*⟩ ::= an IRI as defined in [1], enclosed in a pair of ⟨ *(U+3C) and* ⟩ (U+3E) characters

⟨*prefixName*⟩ ::= a finite sequence of characters matching the as PNAME_NS production of [3] and not matching any of the keyword terminals of the syntax

⟨*abbreviatedIRI*⟩ ::= a finite sequence of characters matching the PNAME_LN production of [3] and not matching any of the keyword terminals of the syntax

⟨*simpleIRI*⟩ ::= a finite sequence of characters matching the PN_LOCAL production of [3] and not matching any of the keyword terminals of the syntax

⟨*IRI*⟩ := ⟨*fullIRI*⟩ | ⟨*abbreviatedIRI*⟩ | ⟨*simpleIRI*⟩

3 Including additional input files

Inclusion of additional input files (both local and distributed) allows to organize the ontology in a modular fashion. To specify the input file the following format should be applied:

⟨*includeExternalFile*⟩ ::= '`\input`' '`{`' ⟨*INPUT_FILE*⟩ '`}`'

⟨*INPUT_FILE*⟩ ::= ⟨*URL*⟩ | ⟨*ABSOLUTE_FILEPATH*⟩

Please note that this command may occur anywhere inside the ontology document.

4 Ontologies

⟨*ontologyDocument*⟩ ::= ⟨*defaultPrefixDeclaration*⟩? ⟨*prefixDeclaration*⟩* ⟨*Ontology*⟩

⟨*defaultPrefixDeclaration*⟩ ::= '`\ns`' ⟨*fullIRI*⟩

⟨*prefixDeclaration*⟩ ::= '`\ns`' ⟨*prefixName*⟩ ⟨*fullIRI*⟩

⟨*Ontology*⟩ ::= '`\begin`' '`{`' '`ontology`' '`}`' ⟨*ontologyIRIs*⟩ (⟨*directlyImportsDocument*⟩
| ⟨*ontologyAnnotation*⟩)* ⟨*axioms*⟩ '`\end`' '`{`' '`ontology`' '`}`'

⟨*ontologyIRIs*⟩ ::= ['`[`' ⟨*ontologyIRI*⟩ ['`,`' ⟨*versionIRI*⟩] '`]`']

⟨*ontologyIRI*⟩ ::= ⟨*IRI*⟩

⟨*versionIRI*⟩ ::= ⟨*IRI*⟩

⟨*directlyImportsDocument*⟩ ::= '`\import`' ⟨*IRI*⟩

⟨*ontologyAnnotation*⟩ ::= ⟨*Annotation*⟩

⟨*axioms*⟩ ::= ⟨*Axiom*⟩*

Please note that prefixes: *owl:*, *rdf:*, *rdfs:*, *xml:* and *xsd:* are built-in and should not be explicitly declared.

5 Annotations

5.1 Annotations of Ontologies, Axioms, and other Annotations

⟨*Annotation*⟩ ::= '`\a`' '`{`' ⟨*AnnotationProperty*⟩ '`,`' ⟨*AnnotationValue*⟩ '`}`' ⟨*annotationAnnotations*⟩

⟨*annotationAnnotations*⟩ ::= ['`[`' ⟨*Annotation*⟩ ('`,`' ⟨*Annotation*⟩)* '`]`']

⟨*AnnotationValue*⟩ ::= ⟨*AnonymousIndividual*⟩ | ⟨*IRI*⟩ | ⟨*Literal*⟩

5.2 Annotation Axioms

⟨*AnnotationAxiom*⟩ ::= ⟨*AnnotationAssertion*⟩
  | ⟨*SubAnnotationPropertyOf*⟩
  | ⟨*AnnotationPropertyDomain*⟩
  | ⟨*AnnotationPropertyRange*⟩

5.2.1 Annotation Assertion

⟨*AnnotationAssertion*⟩ ::= ⟨*AnnotationSubject*⟩ '`\a`' '`{`' ⟨*AnnotationProperty*⟩
    '`,`' ⟨*AnnotationValue*⟩ '`}`' ⟨*axiomAnnotations*⟩

⟨*AnnotationSubject*⟩ ::= ⟨*IRI*⟩ | ⟨*AnonymousIndividual*⟩

5.2.2 Annotation Subproperties

⟨*SubAnnotationPropertyOf*⟩ ::= ⟨*subAnnotationProperty*⟩ '`\aisa`' ⟨*superAnnotationProperty*⟩
    ⟨*axiomAnnotations*⟩

⟨*subAnnotationProperty*⟩ ::= ⟨*AnnotationProperty*⟩

⟨*superAnnotationProperty*⟩ ::= ⟨*AnnotationProperty*⟩

5.2.3 Annotation Property Domain

⟨*AnnotationPropertyDomain*⟩ ::= ⟨*AnnotationProperty*⟩ '`\adomain`' ⟨*IRI*⟩ ⟨*axiomAnnotations*⟩

5.2.4 Annotation Property Range

⟨*AnnotationPropertyRange*⟩ ::= ⟨*AnnotationProperty*⟩ '`\arange`' ⟨*IRI*⟩ ⟨*axiomAnnotations*⟩

6 Entities, Literals, and Anonymous Individuals

6.1 Classes

⟨*Class*⟩ ::= ⟨*IRI*⟩

6.2 Datatype

⟨*Datatype*⟩ ::= ⟨*IRI*⟩

6.3 ObjectProperty

⟨*ObjectProperty*⟩ ::= ⟨*IRI*⟩

6.4 DataProperty

⟨*DataProperty*⟩ ::= ⟨*IRI*⟩

6.5 AnnotationProperty

⟨*AnnotationProperty*⟩ ::= ⟨*IRI*⟩

6.6 Individual

⟨*Individual*⟩ ::= ⟨*NamedIndividual*⟩ | ⟨*AnonymousIndividual*⟩

6.6.1 Named Individuals

⟨*NamedIndividual*⟩ ::= ⟨*IRI*⟩

6.6.2 Anonymous Individuals

⟨*NamedIndividual*⟩ ::= ⟨*nodeID*⟩

6.7 Literals

⟨*Literal*⟩ ::= ⟨*typedLiteral*⟩ | ⟨*stringLiteralNoLanguage*⟩ | ⟨*stringLiteralWithLanguage*⟩

⟨*typedLiteral*⟩ ::= ⟨*lexicalForm*⟩ '`[`' ⟨*Datatype*⟩ '`]`'

⟨*lexicalForm*⟩ ::= ⟨*quotedString*⟩

⟨*stringLiteralNoLanguage*⟩ ::= ⟨*quotedString*⟩

⟨*stringLiteralWithLanguage*⟩ ::= ⟨*quotedString*⟩ '`[`' ⟨*languageTag*⟩ '`]`'

6.8 Entity Declarations and Typing

⟨*Declaration*⟩ ::= ⟨*Entity*⟩ ⟨*axiomAnnotations*⟩

⟨*Entity*⟩ ::= ⟨*Class*⟩ '`\c`'
| ⟨*Datatype*⟩ '`\dt`'
| ⟨*ObjectProperty*⟩ '`\o`'
| ⟨*DataProperty*⟩ '`\d`'
| ⟨*AnnotationProperty*⟩ '`\a`'
| ⟨*NamedIndividual*⟩ '`\i`'

7 Property Expressions

7.1 Object Property Expressions

⟨*ObjectPropertyExpression*⟩ ::= ⟨*ObjectProperty*⟩
| '`(`' ⟨*InverseObjectProperty*⟩ '`)`'

7.1.1 Inverse Object Properties

⟨*InverseObjectProperty*⟩ ::= '`\oinvof`' ⟨*ObjectProperty*⟩

7.2 Data Property Expressions

⟨*DataPropertyExpression*⟩ ::= ⟨*DataProperty*⟩

8 Data Ranges

⟨*DataRange*⟩ ::= ⟨*Datatype*⟩
| ⟨*AtomicDataRange*⟩
| '`(`' ⟨*NonAtomicDataRange*⟩ '`)`'

8.1 Atomic Data Ranges

⟨*AtomicDataRange*⟩ ::= ⟨*SequenceDataIntersectionOf*⟩
| ⟨*SequenceDataUnionOf*⟩
| ⟨*DataOneOf*⟩

8.1.1 Sequence Intersection of Data Ranges

⟨*SequenceDataIntersectionOf*⟩ ::= '`\drandof`' '`{`' ⟨*DataRange*⟩ ('`,`' ⟨*DataRange*⟩)+ '`}`'

8.1.2 Sequence Union of Data Ranges

⟨*SequenceDataUnionOf*⟩ ::= '`\drorof`' '`{`' ⟨*DataRange*⟩ ('`,`' ⟨*DataRange*⟩)+ '`}`'

8.1.3 Enumeration of Literals

⟨*DataOneOf*⟩ ::= '`\droneof`' '`{`' ⟨*Literal*⟩ ('`,`' ⟨*Literal*⟩)* '`}`'

8.2 Non-atomic Data Ranges

⟨*NonAtomicDataRange*⟩ ::= ⟨*DataIntersectionOf*⟩
| ⟨*DataUnionOf*⟩
| ⟨*DataComplementOf*⟩
| ⟨*DatatypeRestriction*⟩

8.2.1 Intersection of Data Ranges

⟨*DataIntersectionOf*⟩ ::= ⟨*DataRange*⟩ '`\drand`' ⟨*DataRange*⟩

8.2.2 Union of Data Ranges

⟨*DataUnionOf*⟩ ::= ⟨*DataRange*⟩ '`\dror`' ⟨*DataRange*⟩

8.2.3 Complement of Data Ranges

⟨*DataComplementOf*⟩ ::= '`\drnot`' ⟨*DataRange*⟩

8.2.4 Datatype Restrictions

⟨*DatatypeRestriction*⟩ ::= ⟨*Datatype*⟩ '`\drres`' ⟨*DatatypeRestrictionExpression*⟩

⟨*DatatypeRestrictionExpression*⟩ ::= '`{`' ⟨*constrainingFacet*⟩ ⟨*restrictionValue*⟩
('`,`' ⟨*constrainingFacet*⟩ ⟨*restrictionValue*⟩)* '`}`'

⟨*constrainingFacet*⟩ ::= ⟨*IRI*⟩

⟨*restrictionValue*⟩ ::= ⟨*Literal*⟩

9 Class Expressions

⟨*ClassExpression*⟩ ::= ⟨*Class*⟩
| ⟨*AtomicClassExpression*⟩
| '`(`' ⟨*NonAtomicClassExpression*⟩ '`)`'

9.1 Atomic Class Expression

9.1.1 Propositional Connectives and Enumeration of Individuals

⟨*AtomicClassExpression*⟩ :: = ⟨*SequenceObjectIntersectionOf*⟩ | ⟨*SequenceObjectUnionOf*⟩
| ⟨*ObjectOneOf*⟩
| ⟨*ObjectSomeValueFrom*⟩ | ⟨*ObjectAllValueFrom*⟩ | ⟨*ObjectHasValue*⟩
| ⟨*ObjectMinCardinality*⟩ | ⟨*ObjectMaxCardinality*⟩ | ⟨*ObjectExactCardinality*⟩
| ⟨*DataSomeValueFrom*⟩ | ⟨*DataAllValueFrom*⟩ | ⟨*DataHasValue*⟩ |
| ⟨*DataMinCardinality*⟩ | ⟨*DataMaxCardinality*⟩ | ⟨*DataExactCardinality*⟩

Sequence Intersection of Class Expressions

⟨*SequenceObjectIntersectionOf*⟩ ::= '`\candof`' '`{`' ⟨*ClassExpression*⟩ ('`,`' ⟨*ClassExpression*⟩
)+ '`}`'

Sequence Union of Class Expressions

⟨*SequenceObjectUnionOf*⟩ ::= '`\corof`' '`{`' ⟨*ClassExpression*⟩ ('`,`' ⟨*ClassExpression*⟩
)+ '`}`'

Enumeration of Individuals

⟨*ObjectOneOf*⟩ ::= '`\ooneof`' '`{`' ⟨*Individual*⟩ ('`,`' ⟨*Individual*⟩)* '`}`'

9.1.2 Object Property Restrictions

Existential Quantification

⟨*ObjectSomeValuesFrom*⟩ ::= '`\oexists`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' '`{`' ⟨*ClassExpression*⟩ '`}`'

Universal Quantification

⟨*ObjectAllValuesFrom*⟩ ::= '`\oforall`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' '`{`' ⟨*ClassExpression*⟩ '`}`'

Individual Value Restriction

⟨*ObjectHasValue*⟩ ::= '`\ohasvalue`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' '`{`' ⟨*Individual*⟩ '`}`'

9.1.3 Object Property Cardinality Restrictions

Minimum Cardinality

⟨*ObjectMinCardinality*⟩ ::= '`\o[>=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' ['`{`' ⟨*ClassExpression*⟩ '`}`']

Maximum Cardinality

⟨*ObjectMaxCardinality*⟩ ::= '`\o[<=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' ['`{`' ⟨*ClassExpression*⟩ '`}`']

Exact Cardinality

⟨*ObjectExactCardinality*⟩ ::= '`\o[=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*ObjectPropertyExpression*⟩ '`}`' ['`{`' ⟨*ClassExpression*⟩ '`}`']

9.1.4 Data Property Restrictions

Existential Quantification

⟨*DataSomeValuesFrom*⟩ ::= '`\dexists`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' '`{`' ⟨*DataRange*⟩ '`}`'

Universal Quantification

⟨*DataAllValuesFrom*⟩ ::= '`\dforall`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' '`{`' ⟨*DataRange*⟩ '`}`'

Literal Value Restriction

⟨*DataHasValue*⟩ ::= '`\dhasvalue`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' '`{`' ⟨*Literal*⟩ '`}`'

9.1.5 Data Property Cardinality Restrictions

Minimum Cardinality

⟨*DataMinCardinality*⟩ ::= '`\d[>=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' ['`{`' ⟨*DataRange*⟩ '`}`']

Maximum Cardinality

⟨*DataMaxCardinality*⟩ ::= '`\d[<=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' ['`{`' ⟨*DataRange*⟩ '`}`']

Exact Cardinality

⟨*DataExactCardinality*⟩ ::= '`\d[=`' ⟨*nonNegativeInteger*⟩ '`]`' '`{`' ⟨*DataPropertyExpression*⟩ '`}`' ['`{`' ⟨*DataRange*⟩ '`}`']

9.2 Non-atomic Class Expression

⟨*NonAtomicClassExpression*⟩ ::= ⟨*ObjectIntersectionOf*⟩ | ⟨*ObjectUnionOf*⟩ | ⟨*ObjectComplementOf*⟩ | ⟨*ObjectHasSelf*⟩

9.2.1 Propositional Connectives

Intersection of Class Expressions

⟨*ObjectIntersectionOf*⟩ ::= ⟨*ClassExpression*⟩ '`\cand`' ⟨*ClassExpression*⟩

Union of Class Expressions

⟨*ObjectUnionOf*⟩ ::= ⟨*ClassExpression*⟩ '`\cor`' ⟨*ClassExpression*⟩

Complement of Class Expressions

⟨*ObjectComplementOf*⟩ ::= '`\cnot`' ⟨*ClassExpression*⟩

9.2.2 Object Property Restrictions

Self-Restriction

⟨*ObjectHasSelf*⟩ ::= '`\ohasself`' ⟨*ObjectPropertyExpression*⟩

10 Axiom

⟨*Axiom*⟩ ::= ⟨*Declaration*⟩ | ⟨*ClassAxiom*⟩ | ⟨*ObjectPropertyAxiom*⟩
| ⟨*DataPropertyAxiom*⟩ | ⟨*DatatypeDefinition*⟩ | ⟨*HasKey*⟩
| ⟨*Assertion*⟩ | ⟨*AnnotationAxiom*⟩

⟨*axiomAnnotations*⟩ ::= ['`[`' ⟨*Annotation*⟩ ('`,`' ⟨*Annotation*⟩)* '`]`']

10.1 Class Expression Axioms

⟨*ClassAxiom*⟩ ::= ⟨*NonSequenceClassAxiom*⟩ | ⟨*SequenceClassAxiom*⟩

10.1.1 Non-sequence Class Expression Axioms

⟨*NonSequenceClassAxiom*⟩ ::= ⟨*SubClassOf*⟩ | ⟨*EquivalentClasses*⟩ | ⟨*DisjointClasses*⟩
| ⟨*DisjointUnion*⟩

Subclass Axioms

⟨*SubClassOf*⟩ ::= ⟨*subClassExpression*⟩ '`\cisa`' ⟨*superClassExpression*⟩ ⟨*axiomAnnotations*⟩

⟨*subClassExpression*⟩ ::= ⟨*ClassExpression*⟩

⟨*superClassExpression*⟩ ::= ⟨*ClassExpression*⟩

Equivalent Classes

⟨*EquivalentClasses*⟩ ::= ⟨*ClassExpression*⟩ '`\ceq`' ⟨*ClassExpression*⟩ ⟨*axiomAnnotations*⟩

Disjoint Classes

⟨*DisjointClasses*⟩ : = ⟨*ClassExpression*⟩ '`\cdisjoint`' ⟨*ClassExpression*⟩ ⟨*axiomAnnotations*⟩

Disjoint Union of Class Expression

⟨*DisjointUnion*⟩ ::= ⟨*Class*⟩ '`\cdisjunion`' ⟨*disjointClassExpressions*⟩ ⟨*axiomAnnotations*⟩

⟨*disjointClassExpressions*⟩ ::= '`{`' ⟨*ClassExpression*⟩ ('`,`' ⟨*ClassExpression*⟩)*
'`}`'

10.1.2 Sequence Class Expression Axioms

⟨*SequenceClassAxiom*⟩ ::= ⟨*SequenceEquivalentClasses*⟩ | ⟨*SequenceDisjointClasses*⟩

Sequence Equivalent Classes

⟨*SequenceEquivalentClasses*⟩ ::= '`\calleq`' '`{`' ⟨*ClassExpression*⟩ ('`,`' ⟨*ClassExpression*⟩
)+ '`}`' ⟨*axiomAnnotations*⟩

Sequence Disjoint Classes

⟨*SequenceDisjointClasses*⟩ : = '`\calldisjoint`' '`{`' ⟨*ClassExpression*⟩ ('`,`' ⟨*ClassExpression*⟩
)+ '`}`' ⟨*axiomAnnotations*⟩

10.2 Object Property Axioms

⟨*ObjectPropertyAxiom*⟩ ::= ⟨*NonSequenceObjectPropertyAxiom*⟩ | ⟨*SequenceObjectPropertyAxiom*⟩

10.2.1 Non-sequence Object Property Axioms

⟨*NonSequenceObjectPropertyAxiom*⟩ ::= ⟨*SubObjectPropertyOf*⟩ | ⟨*EquivalentObjectProperties*⟩
| ⟨*DisjointObjectProperties*⟩ | ⟨*InverseObjectProperties*⟩ | ⟨*ObjectPropertyDomain*⟩
| ⟨*ObjectPropertyRange*⟩ | ⟨*FunctionalObjectProperty*⟩ | ⟨*InverseFunctionalObjectProperty*⟩
| ⟨*ReflexiveObjectProperty*⟩ | ⟨*IrreflexiveObjectProperty*⟩ | ⟨*SymmetricObjectProperty*⟩
| ⟨*AsymmetricObjectProperty*⟩ | ⟨*TransitiveObjectProperty*⟩

Object Subproperties

⟨*SubObjectPropertyOf*⟩ ::= ⟨*subObjectPropertyExpression*⟩ '`\oisa`' ⟨*superObjectPropertyExpression*⟩
⟨*axiomAnnotations*⟩

⟨*subObjectPropertyExpression*⟩ ::= ⟨*ObjectPropertyExpression*⟩ | ⟨*propertyExpressionChain*⟩

⟨*propertyExpressionChain*⟩ ::= '`\ochain`' '`{`' ⟨*ObjectPropertyExpression*⟩ ('`,`'
⟨*ObjectPropertyExpression*⟩)+ '`}`'

⟨*superObjectPropertyExpression*⟩ ::= ⟨*ObjectPropertyExpression*⟩

Equivalent Object Properties

⟨*EquivalentObjectProperties*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oeq`' ⟨*ObjectPropertyExpression*⟩
⟨*axiomAnnotations*⟩

Disjoint Object Properties

⟨*DisjointObjectProperties*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\odisjoint`' ⟨*ObjectPropertyExpression*⟩
⟨*axiomAnnotations*⟩

Inverse Object Properties

⟨*InverseObjectProperties*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oinv`' ⟨*ObjectPropertyExpression*⟩
⟨*axiomAnnotations*⟩

Object Property Domain

⟨*ObjectPropertyDomain*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\odomain`' ⟨*ClassExpression*⟩
⟨*axiomAnnotations*⟩

Object Property Range

⟨*ObjectPropertyRange*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\orange`' ⟨*ClassExpression*⟩
⟨*axiomAnnotations*⟩

Functional Object Properties

⟨*FunctionalObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\ofunc`' ⟨*axiomAnnotations*⟩

Inverse-Functional Object Properties

⟨*InverseFunctionalObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oinvfunc`' ⟨*axiomAnnotations*⟩

Reflexive Object Properties

⟨*ReflexiveObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oreflex`' ⟨*axiomAnnotations*⟩

Irreflexive Object Properties

⟨*IrreflexiveObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oirreflex`' ⟨*axiomAnnotations*⟩

Symmetric Object Properties

⟨*SymmetricObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\osym`' ⟨*axiomAnnotations*⟩

Asymmetric Object Properties

⟨*AsymmetricObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\oasym`' ⟨*axiomAnnotations*⟩

Transitive Object Properties

⟨*TransitiveObjectProperty*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`\otrans`' ⟨*axiomAnnotations*⟩

10.2.2 Sequence Object Property Axioms

⟨*SequenceObjectPropertyAxiom*⟩ :: = ⟨*SequenceEquivalentObjectProperties*⟩
| ⟨*SequenceDisjointObjectProperties*⟩

Sequence Equivalent Object Properties

⟨*SequenceEquivalentObjectProperties*⟩ ::= '`\oalleq`' '`{`' ⟨*ObjectPropertyExpression*⟩ ('`,`' ⟨*ObjectPropertyExpression*⟩)+ '`}`' ⟨*axiomAnnotations*⟩

Sequence Disjoint Object Properties

⟨*SequenceDisjointObjectProperties*⟩ ::= '`\oalldisjoint`' '`{`' ⟨*ObjectPropertyExpression*⟩ ('`,`' ⟨*ObjectPropertyExpression*⟩)+ '`}`' ⟨*axiomAnnotations*⟩

10.3 Data Property Axioms

⟨*DataPropertyAxiom*⟩ ::= ⟨*NonSequenceDataPropertyAxiom*⟩ | ⟨*SequenceDataPropertyAxiom*⟩

10.3.1 Non-sequence Object Property Axioms

⟨*NonSequenceDataPropertyAxiom*⟩ ::= ⟨*SubDataPropertyOf*⟩ | ⟨*EquivalentDataProperties*⟩ | ⟨*DisjointDataProperties*⟩
| ⟨*DataPropertyDomain*⟩ | ⟨*DataPropertyRange*⟩ | ⟨*FunctionalDataProperty*⟩

Data Subproperties

⟨*SubDataPropertyOf*⟩ ::= ⟨*subDataPropertyExpression*⟩ '`\disa`' ⟨*superDataPropertyExpression*⟩ ⟨*axiomAnnotations*⟩

⟨*subDataPropertyExpression*⟩ := ⟨*DataPropertyExpression*⟩

⟨*superDataPropertyExpression*⟩ := ⟨*DataPropertyExpression*⟩

Equivalent Data Properties

⟨*EquivalentDataProperties*⟩ ::= ⟨*DataPropertyExpression*⟩ '`\deq`' ⟨*DataPropertyExpression*⟩ ⟨*axiomAnnotations*⟩

Disjoint Data Properties

⟨*DisjointDataProperties*⟩ ::= ⟨*DataPropertyExpression*⟩ '`\ddisjoint`' ⟨*DataPropertyExpression*⟩ ⟨*axiomAnnotations*⟩

Data Property Domain

⟨*DataPropertyDomain*⟩ ::= ⟨*DataPropertyExpression*⟩ '`\ddomain`' ⟨*ClassExpression*⟩ ⟨*axiomAnnotations*⟩

Data Property Range

⟨*DataPropertyRange*⟩ ::= ⟨*DataPropertyExpression*⟩ '`\drange`' ⟨*DataRange*⟩ ⟨*axiomAnnotations*⟩

Functional Data Properties

⟨*FunctionalDataProperty*⟩ ::= ⟨*DataPropertyExpression*⟩ '`\dfunc`' ⟨*axiomAnnotations*⟩

10.3.2 Sequence Object Property Axioms

⟨*SequenceDataPropertyAxiom*⟩ ::= ⟨*SequenceEquivalentDataProperties*⟩
| ⟨*SequenceDisjointDataProperties*⟩

Equivalent Data Properties

⟨*SequenceEquivalentDataProperties*⟩ ::= '`\dalleq`' '`{`' ⟨*DataPropertyExpression*⟩ ('`,`' ⟨*DataPropertyExpression*⟩)+ '`}`' ⟨*axiomAnnotations*⟩

Disjoint Data Properties

⟨*SequenceDisjointDataProperties*⟩ ::= '`\dalldisjoint`' '`{`' ⟨*DataPropertyExpression*⟩ ('`,`' ⟨*DataPropertyExpression*⟩)+ '`}`' ⟨*axiomAnnotations*⟩

10.4 Datatype Definitions

⟨*DatatypeDefinition*⟩ := ⟨*Datatype*⟩ '`\dtdef`' ⟨*DataRange*⟩ ⟨*axiomAnnotations*⟩

10.5 Keys

⟨*HasKey*⟩ := ⟨*ClassExpression*⟩ '`\key`' ⟨*HasKeyExpression*⟩ ⟨*axiomAnnotations*⟩

⟨*HasKeyExpression*⟩ ::= '`{`' [⟨*ObjectPropertyExpression*⟩ ('`,`' ⟨*ObjectPropertyExpression*⟩)*] '`}`' '`{`' [⟨*DataPropertyExpression*⟩ ('`,`' ⟨*DataPropertyExpression*⟩)*] '`}`'

10.6 Assertion

⟨*Assertion*⟩ ::= ⟨*NonSequenceAssertion*⟩ | ⟨*SequenceAssertion*⟩

⟨*sourceIndividual*⟩ ::= ⟨*Individual*⟩

⟨*targetIndividual*⟩ ::= ⟨*Individual*⟩

⟨*targetValue*⟩ ::= ⟨*Literal*⟩

10.6.1 Non-sequence Assertion

⟨*NonSequenceAssertion*⟩ ::= ⟨*SameIndividual*⟩ | ⟨*DifferentIndividuals*⟩ | ⟨*ClassAssertion*⟩
| ⟨*ObjectPropertyAssertion*⟩ | ⟨*NegativeObjectPropertyAssertion*⟩ |
| ⟨*DataPropertyAssertion*⟩ | ⟨*NegativeDataPropertyAssertion*⟩

Individual Equality

⟨*SameIndividual*⟩ ::= ⟨*Individual*⟩ '`\ieq`' ⟨*Individual*⟩ ⟨*axiomAnnotations*⟩

Individual Inequality

⟨*DifferentIndividuals*⟩ ::= ⟨*Individual*⟩ '`\idiff`' ⟨*Individual*⟩ ⟨*axiomAnnotations*⟩

Class Assertions

⟨*ClassAssertion*⟩ ::= ⟨*ClassExpression*⟩ '`(`' ⟨*Individual*⟩ '`)`' ⟨*axiomAnnotations*⟩

Positive Object Property Assertions

⟨*ObjectPropertyAssertion*⟩ ::= ⟨*ObjectPropertyExpression*⟩ '`(`' ⟨*sourceIndividual*⟩ '`,`' ⟨*targetIndividual*⟩ '`)`' ⟨*axiomAnnotations*⟩

Negative Object Property Assertions

⟨*NegativeObjectPropertyAssertion*⟩ ::= '!' ⟨*ObjectPropertyExpression*⟩ '`(`' ⟨*sourceIndividual*⟩ '`,`' ⟨*targetIndividual*⟩ '`)`' ⟨*axiomAnnotations*⟩

Positive Data Property Assertions

⟨*DataPropertyAssertion*⟩ := ⟨*DataPropertyExpression*⟩ '`(`' ⟨*sourceIndividual*⟩ '`,`' ⟨*targetValue*⟩ '`)`' ⟨*axiomAnnotations*⟩

Negative Data Property Assertions

⟨*NegativeDataPropertyAssertion*⟩ := '!' ⟨*DataPropertyExpression*⟩ '(' ⟨*sourceIndividual*⟩ ',' ⟨*targetValue*⟩ ')' ⟨*axiomAnnotations*⟩

10.6.2 Sequence Assertion

⟨*SequenceAssertion*⟩ ::= ⟨*SequenceSameIndividual*⟩ | ⟨*SequenceDifferentIndividuals*⟩

Sequence Individual Equality

⟨*SequenceSameIndividual*⟩ ::= '`\ialleq`' '{' ⟨*Individual*⟩ (',' ⟨*Individual*⟩)+ '}' ⟨*axiomAnnotations*⟩

Individual Inequality

⟨*SequenceDifferentIndividuals*⟩ ::= '`\ialldiff`' '{' ⟨*Individual*⟩ (',' ⟨*Individual*⟩)+ '}' ⟨*axiomAnnotations*⟩

11 Sample Ontology

In this section a sample ontology is outlined.

```
% define base namespace for this ontology
\ns <http://basenamespace.owl#>

% define additional custom namespaces
% to refer to concept/property/object defined in a given
    namespace use a prefix notation ns:name, e.g., owl:Thing,
     owl:Nothing
\ns ns1: <http://www.namespace1.com/ns1#>
\ns ns2: <http://www.namespace2.com/ns2#>

\begin{ontology}
% import ontologies
\import <http://www.firstontology.org/first.owl>
\import <http://www.firstontology.org/second.owl>

% now you can specify the axioms of the ontology
Person \cisa owl:Thing
Person \a{rdfs:label, "Person"}
Person \a{rdfs:comment, "This is a class for representing
    people"}
Person \cisa \candof{\d[>=1]{hasName}, \d[=1]{hasSurname},
    \dexists{hasSex}{Sex}, \d[=1]{hasAge}}

% if some already existing ontologies are required to be
    reused without cutting-and-pasting them into the current
    ontology, the input command can be used
```

```
\input{/input.txt}

% hasName, hasSurname, and hasAge are datatype properties
hasName \ddomain Person
hasName \drange xsd:string
hasName \a{rdfs:comment, "Person's name"}
hasSurname \ddomain Person
hasSurname \drange xsd:string
hasSurname \a{rdfs:comment, "Person's surname"}
hasAge \ddomain Person
hasAge \drange xsd:integer

% let us define another datatype property by enumerating
    possible values
hasHairColor \ddomain Person
hasHairColor \drange \droneof{"Blonde", "Brown", "Red"}

% if we have another property hasLastName, we can define it
    as equal to the property hasSurname
hasLastName \deq hasSurname

% sex can be defined as a concept containing two objects,
    male and female
Sex \ceq \ooneof{M, F}

M \a{rdfs:label, "Male"}
F \a{rdfs:label, "Female"}

Father \cisa Person
Child \cisa (Person \cand \oexists{hasFather}{Father})

% if we want to reuse some concept defined in the other
    namespace
Father \cisa ns1:Father
ns1:isFatherOf \oinv hasFather

% property has father is functional
hasFather \ofunc

% let's populate the ontology with some individual data
Person(john)

% to express the fact that john and johny both refer to the
    same individual one can use the object equality construct
john \ieq johny

hasName(john, "John")
hasSurname(john, "Wild")
hasAge(john, "35"[xsd:integer])
hasSex(john, M)
```

```
Child(katty)
% to express the fact that john is different from katty one
    can use the object difference construct
katty \idiff john
hasName(katty, "Katty")
hasName(katty, "Wild")
hasSex(katty, F)
hasAge(katty, "3"[xsd:integer])

% John is Katty's father
hasFather(katty, john)

\end{ontology}
```

References

[1] M. Duerst and M. Suignard. RFC 3987: Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard), see `http://www.ietf.org/rfc/rfc3987.txt`, January 2005.

[2] A. Phillips and M. Davis. BCP 47 – Tags for Identifying Languages. BCP 47 Standard, see `http://www.rfc-editor.org/rfc/bcp/bcp47.txt`, September 2006.

[3] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. Latest version available as `http://www.w3.org/TR/rdf-sparql-query/`, January 2008.