Semantic Web Technologies

Lecture 2: OWL 2

Maria Keet

email: keet -AT- inf.unibz.it

home: http://www.meteck.org

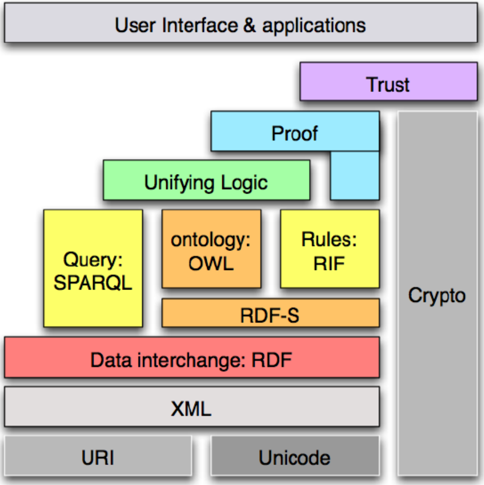
blog: http://keet.wordpress.com

KRDB Research Center
Free University of Bozen-Bolzano, Italy

17 November 2009

Outline

Semantic Web Languages



Outline

Expressivity limitations

- Qualified cardinality restrictions (e.g., no $\texttt{Bicycle} \sqsubseteq \geq 2$ $\texttt{hasComponent.Wheel}$)
- Relational properties (no reflexivity, irreflexivity)
- Data types
  - restrictions to a subset of datatype values (ranges)
  - relationships between values of data properties on one object
  - relationships between values of data properties on different objects
  - aggregation functions
- ‘keys’
- Other things like annotations, imports, versioning, species validation (see p315 of the paper)

Expressivity limitations

- Qualified cardinality restrictions (e.g., no $\texttt{Bicycle} \sqsubseteq \geq 2\ \texttt{hasComponent.Wheel}$)
- Relational properties (no reflexivity, irreflexivity)
- Data types
  - restrictions to a subset of datatype values (ranges)
  - relationships between values of data properties on one object
  - relationships between values of data properties on different objects
  - aggregation functions
- 'keys'
- Other things like annotations, imports, versioning, species validation (see p315 of the paper)

Expressivity limitations

- Qualified cardinality restrictions (e.g., no $\texttt{Bicycle} \sqsubseteq \geq 2$ $\texttt{hasComponent.Wheel}$)
- Relational properties (no reflexivity, irreflexivity)
- Data types
    - restrictions to a subset of datatype values (ranges)
    - relationships between values of data properties on one object
    - relationships between values of data properties on different objects
    - aggregation functions
- 'keys'
- Other things like annotations, imports, versioning, species validation (see p315 of the paper)

Expressivity limitations

- Qualified cardinality restrictions (e.g., no $\texttt{Bicycle} \sqsubseteq \geq 2$ $\texttt{hasComponent.Wheel}$)
- Relational properties (no reflexivity, irreflexivity)
- Data types
  - restrictions to a subset of datatype values (ranges)
  - relationships between values of data properties on one object
  - relationships between values of data properties on different objects
  - aggregation functions
- 'keys'
- Other things like annotations, imports, versioning, species validation (see p315 of the paper)

Syntax problems

- Having both frame-based legacy (Abstract syntax) and axioms (DL) was deemed confusing
- Type of ontology entity. e.g.,

```
Class(A partial
   restriction(hasB someValuesFrom(C))
```

  - `hasB` is data property and `C` a datatype?
  - `hasB` an object property and `C` a class?

OWL-DL has a strict separation of the vocabulary, but the specification does not precisely specify how to enforce this separation at the syntactic level

More syntax problems

- RDF's triple notation, difficult to read and process
- OWL 1 provides mapping from the Abstract Syntax into OWL RDF, but not the converse:
  - an RDF graph G is an OWL-DL ontology if there exists an ontology $\mathcal{O}$ in Abstract Syntax s.t. the result of the normative transformation of $\mathcal{O}$ into triples is precisely G, which makes checking whether G is an OWL-DL ontology very hard in practice:
  - examine all 'relevant' ontologies $\mathcal{O}$ in abstract syntax, check whether the normative transformation of $\mathcal{O}$ into RDF yields precisely G.

More syntax problems

- RDF's triple notation, difficult to read and process
- OWL 1 provides mapping from the Abstract Syntax into OWL RDF, but not the converse:
  - an RDF graph G is an OWL-DL ontology if there exists an ontology $\mathcal{O}$ in Abstract Syntax s.t. the result of the normative transformation of $\mathcal{O}$ into triples is precisely G, which makes checking whether G is an OWL-DL ontology very hard in practice:
  - examine all 'relevant' ontologies $\mathcal{O}$ in abstract syntax, check whether the normative transformation of $\mathcal{O}$ into RDF yields precisely G.

More syntax problems

- RDF's triple notation, difficult to read and process
- OWL 1 provides mapping from the Abstract Syntax into OWL RDF, but not the converse:
  - an RDF graph G is an OWL-DL ontology if there exists an ontology $\mathcal{O}$ in Abstract Syntax s.t. the result of the normative transformation of $\mathcal{O}$ into triples is precisely G, which makes checking whether G is an OWL-DL ontology very hard in practice:
  - examine all 'relevant' ontologies $\mathcal{O}$ in abstract syntax, check whether the normative transformation of $\mathcal{O}$ into RDF yields precisely G.

Problems with the semantics

- RDF's blank nodes, but unnamed individuals not directly available in $\mathcal{SHOIN}(D)$
- Frames and axioms

Outline

Limitations of OWL

OWL 2
- Overview
- OWL 2 DL

OWL 2 profiles
- OWL 2 EL
- OWL 2 QL
- OWL 2 RL

Limitations of OWL 2

Aims

- Address as much as possible of the identified problems (previous slides and JWS 2008 paper)
- Task: compare this with the possible "future extensions" of the JWS 2003 paper

Aims

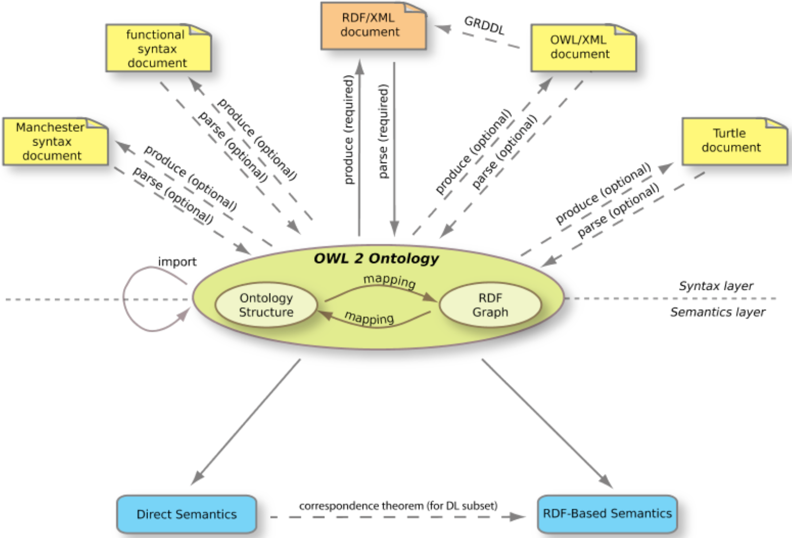
- Address as much as possible of the identified problems (previous slides and JWS 2008 paper)
- Task: compare this with the possible “future extensions” of the JWS 2003 paper

Outline

Some general points

- OWL 2 a 3 weeks young W3C recommendation
- Any OWL 2 ontology can also be viewed as an RDF graph (The relationship between these two views is specified by the Mapping to RDF Graphs document)
- Direct, i.e. model-theoretic, semantics ($\Rightarrow$ "OWL 2 DL") and an RDF-based semantics ($\Rightarrow$ "OWL 2 full")
- Primary exchange syntax for OWL 2 is RDF/XML, others are optional
- Three profiles, which are sub-languages of OWL 2 (syntactic restrictions)

The Structure of OWL 2



Outline

Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL (next slide)
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL (next slide)
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL (next slide)
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL (next slide)
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL (next slide)
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

The language: properties of properties

- property chains (`ObjectPropertyChain`), e.g.:

  ```
  SubObjectPropertyOf( ObjectPropertyChain(
    a:hasMother a:hasSister ) a:hasAunt )
  ```

  with having Lois as the mother of Stewie, and Carol a sister of Lois, the ontology entails that Stewie has Carol as aunt (Note: the example in the JWS08 paper is wrong, which we shall discuss in the part-whole lecture)
- ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, ObjectHasSelf, FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, and DisjointObjectProperties **only on simple object properties** (i.e., has no direct or indirect subproperties that are either transitive or are defined by means of property chains—so we still can't represent parthood fully)

The language: properties of properties

- property chains (`ObjectPropertyChain`), e.g.:

  ```
  SubObjectPropertyOf( ObjectPropertyChain(
     a:hasMother a:hasSister ) a:hasAunt )
  ```

  with having Lois as the mother of Stewie, and Carol a sister of Lois, the ontology entails that Stewie has Carol as aunt
  (Note: the example in the JWS08 paper is wrong, which we shall discuss in the part-whole lecture)
- ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, ObjectHasSelf, FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, and DisjointObjectProperties **only on simple object properties** (i.e., has no direct or indirect subproperties that are either transitive or are defined by means of property chains—so we still can't represent parthood fully)

The language: other extensions

- qualified cardinality restrictions
- The `HasKey` 'key' that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - "relevant mainly for query answering" [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., `DatatypeRestriction( xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer )`

The language: other extensions

- qualified cardinality restrictions
- The `Haskey` ‘key’ that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - “relevant mainly for query answering” [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., `DatatypeRestriction( xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer )`

The language: other extensions

- qualified cardinality restrictions
- The Haskey 'key' that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - "relevant mainly for query answering" [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., DatatypeRestriction(xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer)

The language: other extensions

- qualified cardinality restrictions
- The Haskey 'key' that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - "relevant mainly for query answering" [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., `DatatypeRestriction( xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer )`

The language: other extensions

- qualified cardinality restrictions
- The Haskey ‘key’ that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - “relevant mainly for query answering” [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., DatatypeRestriction(xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer)

The language: other extensions

- qualified cardinality restrictions
- The `Haskey` 'key' that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - "relevant mainly for query answering" [JWS08 p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., `DatatypeRestriction( xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer )`

Partial table of features

Language ⇒ **Feature** ⇓	OWL 1 **Lite**	OWL 1 **DL**	OWL 2 **DL**	OWL 2 Profiles EL	OWL 2 Profiles QL	OWL 2 Profiles RL
Role hierarchy	+	+	+	.	+	.
N-ary roles (where $n \geq 2$)	–	–	–	.	?	.
Role chaining	–	–	+	.	–	.
Role acyclicity	–	–	–	.	–	.
Symmetry	+	+	+	.	+	.
Role values	–	–	–	.	–	.
Qualified number restrictions	–	–	+	.	–	.
One-of, enumerated classes	?	+	+	.	–	.
Functional dependency	+	+	+	.	?	.
Covering constraint over concepts	?	+	+	.	–	.
Complement of concepts	?	+	+	.	+	.
Complement of roles	–	–	+	.	+	.
Concept identification	–	–	–	.	–	.
Range typing	–	+	+	.	+	.
Reflexivity	–	–	+	.	–	.
Antisymmetry	–	–	–	.	–	.
Transitivity	+	+	+	.	–	.
Asymmetry	?	?	+	–	+	+
Irreflexivity	–	–	+	.	–	.
.	.	.	.	.	.	.

Exercise: verify the question marks in the table (tentatively all "–") and fill in the dots (any "±" should be qualified at to what the restriction is)

Outline

Rationale

- Computational considerations
  - Consult "OWL profiles" page *Table 10. Complexity of the Profiles*
- Robustness of implementations w.r.t. *scalable* applications
- Already enjoy substantial user base

Outline

OWL 2 EL Overview

- Intended for large ‘simple’ ontologies
- Focussed on type-level knowledge (TBox)
- Better computational behaviour than OWL 2 DL (polynomial vs. exponential/open)
- Based on the DL language $\mathcal{EL}^{++}$

Supported class restrictions

- existential quantification to a class expression or a data range
- existential quantification to an individual or a literal
- self-restriction
- enumerations involving a single individual or a single literal
- intersection of classes and data ranges

Supported axioms, restricted to allowed set of class expressions

- class inclusion, equivalence, disjointness
- object property inclusion (w. or w.o. property chains), and data property inclusion
- property equivalence
- transitive object properties
- reflexive object properties
- domain and range restrictions
- assertions
- functional data properties
- keys

NOT supported in OWL 2 EL

- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- class negation
- enumerations involving more than one individual
- disjoint properties
- irreflexive, symmetric, and asymmetric object properties
- inverse object properties, functional and inverse-functional object properties

Outline

OWL 2 QL Overview

- Query answering over a large amount of instances with same kind of performance as relational databases (Ontology-Based Data Access)
- Expressive features cover several used features of UML Class diagrams and ER models ('COnceptual MOdel-based Data Access')
- Based on *DL-Lite*$_{\mathcal{R}}$ (*more is possible with UNA and in some implementations*)

Supported Axioms in OWL 2QL, restrictions

- Subclass expressions restrictions:
  - a class
  - existential quantification (ObjectSomeValuesFrom) where the class is limited to owl:Thing
  - existential quantification to a data range (DataSomeValuesFrom)
- Super expressions restrictions:
  - a class
  - intersection (ObjectIntersectionOf)
  - negation (ObjectComplementOf)
  - existential quantification to a class (ObjectSomeValuesFrom)
  - existential quantification to a data range (DataSomeValuesFrom)

Supported Axioms in OWL 2QL

- There are some restrictions on class expressions, object and data properties occurring in functionality assertions cannot be specialized
- subclass axioms
- class expression equivalence (involving subClassExpression), disjointness
- inverse object properties
- property inclusion (not involving property chains and SubDataPropertyOf)
- property equivalence
- property domain and range
- disjoint properties
- symmetric, reflexive, irreflexive, asymmetric properties
- assertions other than individual equality assertions and negative property assertions (DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and

NOT supported in OWL 2 QL

- existential quantification to a class expression or a data range in the subclass position
- self-restriction
- existential quantification to an individual or a literal
- enumeration of individuals and literals
- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- property inclusions involving property chains
- functional and inverse-functional properties
- transitive properties
- keys
- individual equality assertions and negative property assertions

Outline

OWL 2 RL Overview

- Scalable reasoning in the context of RDF(S) application
- Rule-based technologies (forward chaining rule system, over *instances*)
- Inspired by Description Logic Programs and pD*

Supported in OWL 2 RL

- There are more restrictions on class expressions (see table 2, e.g. no SomeValuesFrom on the right-hand side of a subclass axiom)
- All axioms in OWL 2 RL are constrained in a way that is compliant with the restrictions in Table 2. Thus, OWL 2 RL supports all axioms of OWL 2 apart from disjoint unions of classes and reflexive object property axioms.

Outline

Another section on speculation about future extensions

- The ‘leftover’ from OWL 1’s “Future extensions” (UNA, CWA, defaults), parthood relation (primarily: antisymmetry, restrictions on current usage of properties)
- New “future of OWL”, a.o.:
  - Syntactic sugar: ‘macros’, ‘n-aries’
  - Query languages: EQL-lite and nRQL w.r.t. SPARQL
  - Integration with rules: RIF, DL-safe rules, SBVR
  - Orthogonal dimensions: temporal, fuzzy, rough, probabilistic

Another section on speculation about future extensions

- The 'leftover' from OWL 1's "Future extensions" (UNA, CWA, defaults), parthood relation (primarily: antisymmetry, restrictions on current usage of properties)
- New "future of OWL", a.o.:
  - Syntactic sugar: 'macros', 'n-aries'
  - Query languages: EQL-lite and nRQL w.r.t. SPARQL
  - Integration with rules: RIF, DL-safe rules, SBVR
  - Orthogonal dimensions: temporal, fuzzy, rough, probabilistic

Another section on speculation about future extensions

- The 'leftover' from OWL 1's "Future extensions" (UNA, CWA, defaults), parthood relation (primarily: antisymmetry, restrictions on current usage of properties)
- New "future of OWL", a.o.:
  - Syntactic sugar: 'macros', 'n-aries'
  - Query languages: EQL-lite and nRQL w.r.t. SPARQL
  - Integration with rules: RIF, DL-safe rules, SBVR
  - Orthogonal dimensions: temporal, fuzzy, rough, probabilistic

Summary