COMP718: Ontologies and Knowledge Bases

Lecture 4: OWL 2 and Reasoning

Maria Keet

email: keet@ukzn.ac.za

home: http://www.meteck.org

School of Mathematics, Statistics, and Computer Science
University of KwaZulu-Natal, South Africa

Feb 28/29, 2012

Outline

1. Limitations
2. OWL 2
   - OWL 2 DL
3. OWL 2 profiles
   - OWL 2 EL
   - OWL 2 QL
   - OWL 2 RL
4. Reasoning

Outline

Expressivity limitations

- Qualified cardinality restrictions (e.g., no $\texttt{Bicycle} \sqsubseteq \geq 2\ \texttt{hasComponent.Wheel}$)
- Relational properties (no reflexivity, irreflexivity)
- Data types, missing
  - 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
- 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, missing
  - 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
- 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, missing
  - 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
- 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

1. Limitations
2. OWL 2
   - OWL 2 DL
3. OWL 2 profiles
   - OWL 2 EL
   - OWL 2 QL
   - OWL 2 RL
4. Reasoning

Aims

- Address as much as possible of the identified problems (previous slides and "the next steps for OWL 2" paper)
- Task: compare this with the possible "future extensions" of the "the making of an ontology language" paper

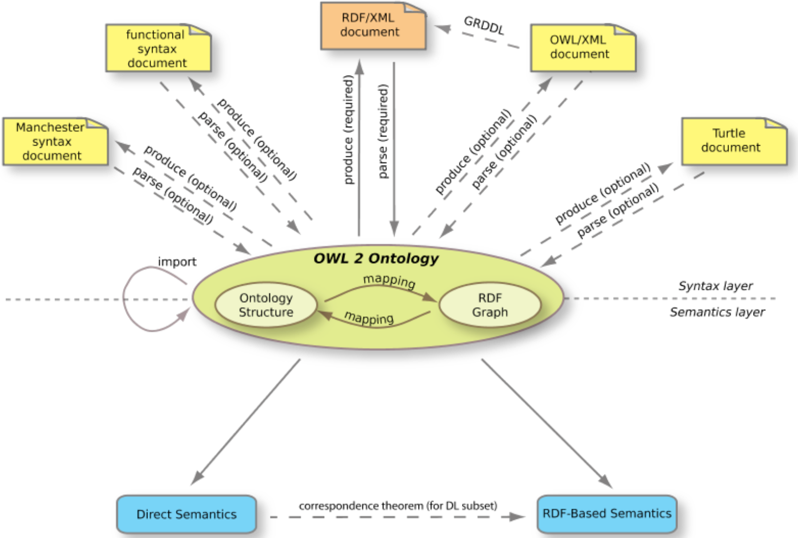
Aims

- Address as much as possible of the identified problems (previous slides and "the next steps for OWL 2" paper)
- Task: compare this with the possible "future extensions" of the "the making of an ontology language" paper

Some general points

- OWL 2 a W3C recommendation since 27-10-'09
- 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



Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL
- 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
- 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
- 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
- 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
- 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 Grace as the mother of Stewie, and Carol a sister of Grace, the ontology entails that Stewie has Carol as aunt
  or, e.g.,: $contains \circ hasPart \sqsubseteq contains$
- 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)

The language: properties of properties

- property chains (`ObjectPropertyChain`), e.g.:
  ```
  SubObjectPropertyOf( ObjectPropertyChain(
    a:hasMother a:hasSister ) a:hasAunt )
  ```
  with having Grace as the mother of Stewie, and Carol a sister of Grace, the ontology entails that Stewie has Carol as aunt
  or, e.g.,: $contains \circ hasPart \sqsubseteq contains$
- 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)

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” [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd: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" [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd: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" [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd: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" [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd: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" [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd: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” [Cuenca Grau et al, 2008, 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"^^xsd:integer xsd:maxExclusive "10"^^xsd:integer)`

OWL 2 DL and DLs

- (In addition to those of OWL-DL/$\mathcal{SHOIN}$)
- qualified cardinality restrictions, $\geq nR.C$ and $\leq nR.C$, semantics:
  - $(\geq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \geq n\}$
  - $(\leq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \leq n\}$
- Properties of roles:
  - Reflexive: $Ref(R)$, with semantics: $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \in (R)^{\mathcal{I}}$
  - Irreflexive: $Irr(R)$, with semantics: $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \notin (R)^{\mathcal{I}}$
  - Asymmetric: Asym(R), with semantics: $\forall x, y : (x,y) \in (R)^{\mathcal{I}}$ implies $(y,x) \notin (R)^{\mathcal{I}}$
- *Limited* role chaining: $R \circ S \sqsubseteq R$, with semantics: $\forall y_1, \ldots, y_4 : (y_1, y_2) \in (R)^{\mathcal{I}}$ and $(y_3, y_4) \in (S)^{\mathcal{I}}$ imply $(y_1, y_4) \in (R)^{\mathcal{I}}$, and regularity restriction (strict linear order $<$ on the properties)

OWL 2 DL and DLs

- (In addition to those of OWL-DL/$\mathcal{SHOIN}$)
- qualified cardinality restrictions, $\geq nR.C$ and $\leq nR.C$, semantics:
  - $(\geq nR.C)^\mathcal{I} = \{x \mid \sharp\{y \mid (x,y) \in R^\mathcal{I} \cap y \in C^\mathcal{I}\} \geq n\}$
  - $(\leq nR.C)^\mathcal{I} = \{x \mid \sharp\{y \mid (x,y) \in R^\mathcal{I} \cap y \in C^\mathcal{I}\} \leq n\}$
- Properties of roles:
  - Reflexive: $Ref(R)$, with semantics: $\forall x : x \in \Delta^\mathcal{I}$ implies $(x,x) \in (R)^\mathcal{I}$
  - Irreflexive: $Irr(R)$, with semantics: $\forall x : x \in \Delta^\mathcal{I}$ implies $(x,x) \notin (R)^\mathcal{I}$
  - Asymmetric: Asym(R), with semantics: $\forall x, y : (x,y) \in (R)^\mathcal{I}$ implies $(y,x) \notin (R)^\mathcal{I}$
- *Limited* role chaining: $R \circ S \sqsubseteq R$, with semantics: $\forall y_1, \ldots, y_4 : (y_1, y_2) \in (R)^\mathcal{I}$ and $(y_3, y_4) \in (S)^\mathcal{I}$ imply $(y_1, y_4) \in (R)^\mathcal{I}$, and regularity restriction (strict linear order $<$ on the properties)

OWL 2 DL and DLs

- (In addition to those of OWL-DL/$\mathcal{SHOIN}$)
- qualified cardinality restrictions, $\geq nR.C$ and $\leq nR.C$, semantics:
  - $(\geq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \geq n\}$
  - $(\leq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \leq n\}$
- Properties of roles:
  - Reflexive: $Ref(R)$, with semantics: $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \in (R)^{\mathcal{I}}$
  - Irreflexive: $Irr(R)$, with semantics: $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \notin (R)^{\mathcal{I}}$
  - Asymmetric: Asym(R), with semantics: $\forall x, y : (x,y) \in (R)^{\mathcal{I}}$ implies $(y,x) \notin (R)^{\mathcal{I}}$
- *Limited* role chaining: $R \circ S \sqsubseteq R$, with semantics: $\forall y_1, \ldots, y_4 : (y_1, y_2) \in (R)^{\mathcal{I}}$ and $(y_3, y_4) \in (S)^{\mathcal{I}}$ imply $(y_1, y_4) \in (R)^{\mathcal{I}}$, and regularity restriction (strict linear order $<$ on the properties)

Definition ((Regular) Role Inclusion Axioms (HorrocksEtAl06))

Let $\prec$ be a regular order on roles. A **role inclusion axiom** (RIA for short) is an expression of the form $w \sqsubseteq R$, where w is a finite string of roles not including the universal role U, and $R \neq U$ is a role name. A **role hierarchy** $\mathcal{R}_h$ is a finite set of RIAs. An interpretation $\mathcal{I}$ **satisfies** a role inclusion axiom $w \sqsubseteq R$, written $\mathcal{I} \models w \sqsubseteq R$, if $w^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. An interpretation is a **model** of a role hierarchy $\mathcal{R}_h$ if it satisfies all RIAs in $\mathcal{R}_h$, written $\mathcal{I} \models \mathcal{R}_h$. A RIA $w \sqsubseteq R$ is $\prec$-**regular** if R is a role name, and

- $w = RR$, or
- $w = R^-$, or
- $w = S_1 ... S_n$ and $S_i \prec R$, for all $1 \geq i \geq n$, or
- $w = RS_1 ... S_n$ and $S_i \prec R$, for all $1 \geq i \geq n$, or
- $w = S_1 ... S_n R$ and $S_i \prec R$, for all $1 \geq i \geq n$.

Finally, a role hierarchy $\mathcal{R}_h$ is **regular** if there exists a regular order $\prec$ such that each RIA in $\mathcal{R}_h$ is $\prec$-regular.

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

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 $\mathcal{EL}^{++}$ (PTime complete)
- Reasoner: e.g. CEL `http://code.google.com/p/cel/`

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 and data property inclusion
- property equivalence
- transitive object properties
- reflexive object properties
- domain and range restrictions
- assertions
- functional data properties
- keys
- In short: $\sqcap \; \exists \; \top \; \bot \; \sqsubseteq \; \sqcap \; \exists \; \top \; \bot$

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

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

- 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 DataPropertyAssertion)

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

OWL 2 RL Overview

- Development motivated by: what fraction of OWL 2 DL can be expressed by rules (with equality)?
- 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*
- Reasoning in PTime

Supported in OWL 2 RL

- 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.
- No $\forall$ and $\neg$ on lhs, and $\exists$ and $\sqcup$ on rhs of $\sqsubseteq$

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

Outline

Reasoning services for DL-based OWL ontologies

- OWL ontology is a first-order logical theory $\Rightarrow$ verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- Main ('standard') reasoning tasks for the OWL ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
- Non-standard reasoning services, such as explanation, repair, least common subsumer, ...
- Note: Not all OWL languages are equally suitable for all these reasoning tasks

Reasoning services for DL-based OWL ontologies

- OWL ontology is a first-order logical theory ⇒ verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- Main ('standard') reasoning tasks for the OWL ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
- Non-standard reasoning services, such as explanation, repair, least common subsumer, ...
- Note: Not all OWL languages are equally suitable for all these reasoning tasks

Reasoning services for DL-based OWL ontologies

- OWL ontology is a first-order logical theory $\Rightarrow$ verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- Main ('standard') reasoning tasks for the OWL ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
- Non-standard reasoning services, such as explanation, repair, least common subsumer, ...
- Note: Not all OWL languages are equally suitable for all these reasoning tasks

Reasoning services for DL-based OWL ontologies

- OWL ontology is a first-order logical theory $\Rightarrow$ verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- Main ('standard') reasoning tasks for the OWL ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
- Non-standard reasoning services, such as explanation, repair, least common subsumer, ...
- Note: Not all OWL languages are equally suitable for all these reasoning tasks

Reasoning services for DL-based OWL ontologies

- Consistency of the ontology
  - Is the ontology $K = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for K?
- Concept (and role) consistency
  - is there a model of T in which C (resp. R) has a nonempty extension?
- Concept (and role) subsumption
  - i.e., is the extension of C (resp. R) contained in the extension of D in every model of T?
- Instance checking
  - is a a member of concept C in K, i.e., is the fact $C(a)$ satisfied by every interpretation of K?
- Instance retrieval
  - find all members of C in K, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of K
- Query answering
  - compute all tuples of individuals t s.t. query $q(t)$ is entailed by K, i.e., $q(t)$ is satisfied by every interpretation of K

Note: Reasoning with OWA (vs. CWA)

- **O**pen **W**orld **A**ssumption
  - Absence of information is interpreted as unknown information
  - Assumes incomplete information
  - Good for describing knowledge in a way that is extensible
- **C**losed **W**orld **A**ssumption
  - Absence of information is interpreted as negative information
  - Assumes we have complete information
  - Good for constraining information and validating data in an application

Note: Reasoning with OWA (vs. CWA)

- **O**pen **W**orld **A**ssumption
  - Absence of information is interpreted as unknown information
  - Assumes incomplete information
  - Good for describing knowledge in a way that is extensible
- **C**losed **W**orld **A**ssumption
  - Absence of information is interpreted as negative information
  - Assumes we have complete information
  - Good for constraining information and validating data in an application

Note: Reasoning with OWA (vs. CWA)

- **O**pen **W**orld **A**ssumption
  - Absence of information is interpreted as unknown information
  - Assumes incomplete information
  - Good for describing knowledge in a way that is extensible
- **C**losed **W**orld **A**ssumption
  - Absence of information is interpreted as negative information
  - Assumes we have complete information
  - Good for constraining information and validating data in an application

Note: Reasoning with OWA (vs. CWA)

- **O**pen **W**orld **A**ssumption
  - Absence of information is interpreted as unknown information
  - Assumes incomplete information
  - Good for describing knowledge in a way that is extensible
- **C**losed **W**orld **A**ssumption
  - Absence of information is interpreted as negative information
  - Assumes we have complete information
  - Good for constraining information and validating data in an application

Example

Which alumni do not have a PhD?

Alumnus	Degree Obtained
Delani	PhD in history
Sally	PhD in politics
Peter	MSc in Informatics
Dalila	PhD in politics

Example

Which alumni do not have a PhD?

Alumnus	Degree Obtained
Delani	PhD in history
Sally	PhD in politics
Peter	MSc in Informatics
Dalila	PhD in politics

- Query under CWA says "Peter"
- Query under OWA cannot say "Peter", because we do not know if Peter also obtained a PhD. To retrieve "Peter" we have add an axiom somehow stating that Peter does not have a PhD (e.g., by being an instance of *PhD student*, declaring the degrees to be disjoint & covering, ...).

Automated reasoning examples

- Subsumption reasoning, like in the exercise ($\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$)
- Example with Schrödinger's cat
- Example with the `sampleClassification.owl`
- Exercise with instance classification and KB consistency (and OWA)
- Exercise with finding the errors in a 'dirty' ontology

Summary

1. Limitations
2. OWL 2
   - OWL 2 DL
3. OWL 2 profiles
   - OWL 2 EL
   - OWL 2 QL
   - OWL 2 RL
4. Reasoning