COMP718: Ontologies and Knowledge Bases
Lecture 3: The Web Ontology Language OWL

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

21/28 February 2012
Outline

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Recap previous week

- First Order Predicate Logic, model theoretic-semantics
- Description Logics
- Tableau reasoning (exercises with the graph and with vegans and vegetarians)
  - Soundness (if $\Gamma \vdash \phi$ then $\Gamma \models \phi$) and completeness (if $\Gamma \models \phi$ then $\Gamma \vdash \phi$) [recollect “$\vdash$” derivable with a set of inference rules, and “$\models$” as implies, i.e., every truth assignment that satisfies $\Gamma$ also satisfies $\phi$]
  - If the algorithm is incomplete, then there exist entailments that cannot be computed (hence, missing some results)
  - If the algorithm is unsound then false conclusions can be derived from true premises, which is even more undesirable
Recap previous week

- First Order Predicate Logic, model theoretic-semantics
- Description Logics
- Tableau reasoning (exercises with the graph and with vegans and vegetarians)
  - Soundness (if $\Gamma \vdash \phi$ then $\Gamma \models \phi$) and completeness (if $\Gamma \models \phi$ then $\Gamma \vdash \phi$) [recollect “$\vdash$” derivable with a set of inference rules, and “$\models$” as implies, i.e., every truth assignment that satisfies $\Gamma$ also satisfies $\phi$]
  - If the algorithm is incomplete, then there exist entailments that cannot be computed (hence, missing some results)
  - If the algorithm is unsound then false conclusions can be derived from true premises, which is even more undesirable
Recap previous week

- First Order Predicate Logic, model theoretic-semantics
- Description Logics
- Tableau reasoning (exercises with the graph and with vegans and vegetarians)
  - Soundness (if \( \Gamma \vdash \phi \) then \( \Gamma \models \phi \)) and completeness (if \( \Gamma \models \phi \) then \( \Gamma \vdash \phi \)) [recollect “\( \vdash \)” derivable with a set of inference rules, and “\( \models \)” as implies, i.e., every truth assignment that satisfies \( \Gamma \) also satisfies \( \phi \)]
  - If the algorithm is incomplete, then there exist entailments that cannot be computed (hence, missing some results)
  - If the algorithm is unsound then false conclusions can be derived from true premises, which is even more undesirable
Recap previous week

- First Order Predicate Logic, model theoretic-semantics
- Description Logics
- Tableau reasoning (exercises with the graph and with vegans and vegetarians)
  - Soundness (if $\Gamma \vdash \phi$ then $\Gamma \models \phi$) and completeness (if $\Gamma \models \phi$ then $\Gamma \vdash \phi$) [recollect “$\vdash$” derivable with a set of inference rules, and “$\models$” as implies, i.e., every truth assignment that satisfies $\Gamma$ also satisfies $\phi$]
  - If the algorithm is incomplete, then there exist entailments that cannot be computed (hence, missing some results)
  - If the algorithm is unsound then false conclusions can be derived from true premises, which his even more undesirable
Outline

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Toward one ontology language for the Web

- Plethora of ontology languages; e.g., KIF, KL-ONE, LOOM, F-logic, DAML, OIL, DAML+OIL, ....
- Lack of a lingua franca; hence, ontology interoperation problems even on the syntactic level
- Advances in expressive DL languages and, more importantly, in automated reasoners for expressive DL languages (mainly: FaCT++, then Racer)
- Limitations of RDF(S) as Semantic Web ‘ontology language’
### 1 Introduction
- W3C’s layer cake
- Limitations of RDFS

### 2 OWL
- Design of OWL
- OWL family of languages
- OWL and Description Logics
- OWL Syntaxes
- Layering OWL on top of RDF(S)
The layer cake

User Interface & applications

Trust

Proof

Unifying Logic

Query: SPARQL

ontology: OWL

Rules: RIF

RDF-S

Data interchange: RDF

XML

URI

Unicode

Crypto
Stack of Languages

- XML
  - Surface syntax, no semantics
- XML Schema
  - Describes structure of XML documents
- RDF
  - Datamodelfor “relations” between “things”
- RDF Schema
  - RDF Vocabulary Definition Language
Outline

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
RDFS as an Ontology Language

- Classes
- Properties
- Class hierarchies
- Property hierarchies
- Domain and range restrictions
Expressive limitations of RDF(S)

- Only binary relations
- Characteristics of Properties (e.g. inverse, transitive, symmetric)
- Local range restrictions (e.g. for Class Person, the property hasName has range xsd:string)
- Complex concept descriptions (e.g. Person is defined by Man and Woman)
- Cardinality restrictions (e.g. a Person may have at most 1 name)
- Disjointness axioms (e.g. nobody can be both a Man and a Woman)
Layering issues

- Syntax
  - Only binary relations in RDF
  - Verbose Syntax
  - No limitations on graph in RDF
    - Every graph is valid

- Semantics
  - Malformed graphs
  - Use of vocabulary in language
    - e.g. `<rdfs:Class, rdfs:subClassOf, ex:a>`
  - Meta-classes
    - e.g. `<ex:a, rdf:type, ex:a>`
1 Introduction
- W3C’s layer cake
- Limitations of RDFS

2 OWL
- Design of OWL
- OWL family of languages
- OWL and Description Logics
- OWL Syntaxes
- Layering OWL on top of RDF(S)
The place of OWL in the layer cake
Stack of Languages

- **XML**
  - Surface syntax, no semantics
- **XML Schema**
  - Describes structure of XML documents
- **RDF**
  - Datamodel for “relations” between “things”
- **RDF Schema**
  - RDF Vocabulary Definition Language
- **OWL**
  - A more expressive Vocabulary Definition Language
Outline

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Design Goals for OWL

- **Shareable**
- **Changing** over time
- **Interoperability**
- **Inconsistency** detection
- Balancing **expressivity and complexity**
- **Ease of use**
- Compatible with **existing standards**
- **Internationalization**
Requirements for OWL

- Ontologies are **object on the Web**
- with **their own meta-data**, versioning, etc...
- Ontologies are **extendable**
- They contain **classes, properties, data-types, range/domain, individuals**
- **Equality** (for classes, for individuals)
- **Classes as instances**
- **Cardinality** constraints
- **XML** syntax
Objectives for OWL

Objectives:
- layered language
- complex datatypes
- digital signatures
- decidability (in part)
- local unique names (in part)

Disregarded:
- default values
- closed world option
- property chaining
- arithmetic
- string operations
- partial imports
- view definitions
- procedural attachments
Objectives for OWL

Objectives:

- layered language
- complex datatypes
- digital signatures
- decidability (in part)
- local unique names (in part)

Disregarded:

- default values
- closed world option
- property chaining
- arithmetic
- string operations
- partial imports
- view definitions
- procedural attachments
Objectives for OWL

Objectives:

- layered language
- complex datatypes
- digital signatures
- decidability (in part)
- local unique names (in part)

Disregarded:

- default values
- closed world option
- property chaining
- arithmetic
- string operations
- partial imports
- view definitions
- procedural attachments
Leveraging experiences with OWL’s predecessors SHOE, OIL, DAML-ONT, and DAML+OIL (frames, OO, DL)

OWL extends RDF Schema to a full-fledged knowledge representation language for the Web

- Logical expressions (and, or, not)
- (in)equality
- local properties
- required/optional properties
- required values
- enumerated classes
- symmetry, inverse
Outline

1. Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2. OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Species of OWL

- OWL Lite
  - Classification hierarchy
  - Simple constraints

- OWL DL
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalization in a DL

- OWL Full
  - Very high expressiveness
  - Losing tractability
  - All syntactic freedom of RDF (self-modifying)
Species of OWL

- **OWL Lite**
  - Classification hierarchy
  - Simple constraints

- **OWL DL**
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalization in a DL

- **OWL Full**
  - Very high expressiveness
  - Losing tractability
  - All syntactic freedom of RDF (self-modifying)
Species of OWL

- **OWL Lite**
  - Classification hierarchy
  - Simple constraints

- **OWL DL**
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalization in a DL

- **OWL Full**
  - Very high expressiveness
  - Losing tractability
  - All syntactic freedom of RDF (self-modifying)
Species of OWL

- **OWL Lite**
  - Classification hierarchy
  - Simple constraints

- **OWL DL**
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalization in a DL

- **OWL Full**
  - Very high expressiveness
  - Losing tractability
  - All syntactic freedom of RDF (self-modifying)
Features of OWL languages

- **OWL Lite**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - (unqualified) cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - someValuesFrom
  - allValuesFrom

- **OWL DL**
  - Negation
  - Disjunction
  - (unqualified) Full cardinality
  - Enumerated classes
  - hasValue

- **OWL Full**
  - Meta-classes
  - Modify language
Features of OWL languages

- **OWL Lite**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - (unqualified) cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - someValuesFrom
  - allValuesFrom

- **OWL DL**
  - Negation
  - Disjunction
  - (unqualified) Full cardinality
  - Enumerated classes
  - hasValue

- **OWL Full**
  - Meta-classes
  - Modify language
Features of OWL languages

- **OWL Lite**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - (unqualified) cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - someValuesFrom
  - allValuesFrom

- **OWL DL**
  - Negation
  - Disjunction
  - (unqualified) Full cardinality
  - Enumerated classes
  - hasValue

- **OWL Full**
  - Meta-classes
  - Modify language
Features of OWL languages

- **OWL Lite**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - (unqualified) cardinality
  - 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - someValuesFrom
  - allValuesFrom

- **OWL DL**
  - Negation
  - Disjunction
  - (unqualified) Full cardinality
  - Enumerated classes
  - hasValue

- **OWL Full**
  - Meta-classes
  - Modify language
OWL Full

- **No restriction on use of vocabulary** (as long as legal RDF)
  - Classes as instances (and much more)
- **RDF style model theory**
  - Reasoning using FOL engine
  - Semantics should correspond to OWL DL for restricted KBs
OWL DL

- Use of vocabulary restricted
  - Cannot be used to do “nasty things” (e.g., modify OWL)
  - No classes as instances (this will be discussed in a later lecture)
  - Defined by abstract syntax

- Standard DL-based model theory
  - Direct correspondence with a DL
  - Automated reasoning with DL reasoners (e.g., Racer, Pellet, FaCT++)
OWL Lite

- No explicit negation or union
- Restricted cardinality (0/1)
- No nominals (oneOf)
- DL-based semantics
  - Automated reasoning with DL reasoners (e.g., Racer, Pellet, FaCT++)
More on OWL species

- OWL Full is *not* a Description Logic
- OWL Lite has strong syntactic restrictions, but only limited semantics restrictions cf. OWL DL
  - Negation can be encoded using disjointness
  - With negation an conjunction, you can encode disjunction

For instance:

Class(C complete unionOf(B C))

is equivalent to:

DisjointClasses(notB B)
DisjointClasses(notC C)
Class(notBandnotC complete notB notC)
DisjointClasses(notBandnotC BorC)
Class(C complete notBandnotC)
OWL family of languages

More on OWL species

- OWL Full is *not* a Description Logic
- OWL Lite has strong syntactic restrictions, but only limited semantics restrictions cf. OWL DL
  - Negation can be encoded using disjointness
  - With negation an conjunction, you can encode disjunction

- For instance:

  Class(C complete unionOf(B C))

  is equivalent to:

  DisjointClasses(notB B)
  DisjointClasses(notC C)
  Class(notBandnotC complete notB notC)
  DisjointClasses(notBandnotC BorC)
  Class(C complete notBandnotC)
Outline

1. Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2. OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
OWL Lite corresponds to the DL $SHIF(D)$. It has:

- Named classes ($A$)
- Named properties ($P$)
- Individuals ($C(o)$)
- Property values ($P(o, a)$)
- Intersection ($C \cap D$)
- Union ($C \sqcup D$)
- Negation ($\neg C$)
- Existential value restrictions ($\exists P.C$)
- Universal value restrictions ($\forall P.C$)
- Unqualified (0/1) number restrictions ($\geq nP$, $\leq nP$, $= nP$), $0 \leq n \leq 1$
OWL DL corresponds to the DL $SHOIN(D)$. In addition to all of OWL Lite, it has also:

- Arbitrary number restrictions ($\geq nP$, $\leq nP$, $= nP$), $0 \leq n$
- Property value ($\exists P.\{o\}$)
- Enumeration ($\{o_1, \ldots, o_n\}$)
**OWL constructs (summarised from the standard)**

<table>
<thead>
<tr>
<th>OWL Construct</th>
<th>DL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${o_1, \ldots, o_n}$</td>
<td>${giselle, juan}$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P. C$</td>
<td>$\forall hasChild. Doctor$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P. C$</td>
<td>$\exists hasChild. Lawyer$</td>
</tr>
<tr>
<td>value</td>
<td>$\exists P. {o}$</td>
<td>$\exists citizenOf. {RSA}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2hasChild$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1hasChild$</td>
</tr>
<tr>
<td>cardinality</td>
<td>$= nP$</td>
<td>$= 2hasParent$</td>
</tr>
</tbody>
</table>

+ XML Schema datatypes: int, string, real, etc...
# OWL Axioms

<table>
<thead>
<tr>
<th>OWL Axiom</th>
<th>DL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td><em>Human</em> $\sqsubseteq$ <em>Animal</em> $\sqcap$ <em>Biped</em></td>
</tr>
<tr>
<td>EquivalentClasses</td>
<td>$C_1 \equiv ... \equiv C_n$</td>
<td><em>Man</em> $\equiv$ <em>Human</em> $\sqcap$ <em>Male</em></td>
</tr>
<tr>
<td>SubPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td><em>hasDaughter</em> $\sqsubseteq$ <em>hasChild</em></td>
</tr>
<tr>
<td>EquivalentProperties</td>
<td>$P_1 \equiv ... \equiv P_n$</td>
<td><em>cost</em> $\equiv$ <em>price</em></td>
</tr>
<tr>
<td>SameIndividual</td>
<td>$o_1 = ... = o_n$</td>
<td><em>President Zuma</em> = <em>J Zuma</em></td>
</tr>
<tr>
<td>DisjointClasses</td>
<td>$C_i \sqsubseteq \neg C_j$</td>
<td><em>Male</em> $\sqsubseteq$ $\neg$ <em>Female</em></td>
</tr>
<tr>
<td>DifferentIndividuals</td>
<td>$o_i \neq o_j$</td>
<td><em>sally</em> $\neq$ <em>shereen</em></td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^-$</td>
<td><em>hasChild</em> $\equiv$ $\neg$ <em>hasParent</em></td>
</tr>
<tr>
<td>Transitive</td>
<td>$P^+ \sqsubseteq P$</td>
<td>$\neg$ <em>ancestor</em> $\sqsubseteq$ <em>ancestor</em></td>
</tr>
<tr>
<td>Symmetric</td>
<td>$P \equiv P^-$</td>
<td><em>connectedTo</em> $\equiv$ $\neg$ <em>connectedTo</em></td>
</tr>
</tbody>
</table>
Outline

1. Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2. OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Syntaxes of OWL

- **RDF**
  - Official exchange syntax
  - Hard for humans
  - RDF parsers are hard to write!

- **XML**
  - Not the RDF syntax
  - Still hard for humans, but more XML than RDF tools available

- **Abstract syntax**
  - Not defined for OWL Full
  - To some, considered human readable

- **User-usable ones**
  - e.g., Manchester syntax, informal and limited matching with UML, pseudo-NL verbalizations (mainly in English only)
Example from [OwlGuide]:

```xml
<!ENTITY vin "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#" >
<!ENTITY food "http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#" >
         xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#">
  ... 
  <owl:Class rdf:ID="Wine"> <rdfs:subClassOf rdf:resource="#EdibleThing"/>
    <rdfs:label xml:lang="en">wine</rdfs:label>
    <rdfs:label xml:lang="fr">vin</rdfs:label> ...
  </owl:Class>

  <owl:Class rdf:ID="Pasta"> <rdfs:subClassOf rdf:resource="#EdibleThing"/>
    ... </owl:Class> </rdf:RDF>
```
OWL Abstract syntax

Class( professor partial )
Class( associateProfessor partial academicStaffMember)

DisjointClasses( associateProfessor assistantProfessor )
DisjointClasses( professor associateProfessor )

Class( faculty complete academicStaffMember)

In DL syntax:

associateProfessor ⊑ academicStaffMember
associateProfessor ⊑ ¬ assistantProfessor
professor ⊑ ¬ associateProfessor
faculty ≡ academicStaffMember
DatatypeProperty(age range(xsd:nonNegativeInteger))
ObjectProperty(lecturesIn)

ObjectProperty(isTaughtBy domain(course) range(academicStaffMember))
SubPropertyOf(isTaughtBy involves)

ObjectProperty(teaches inverseOf(isTaughtBy)
domain(academicStaffMember) range(course))

EquivalentProperties(lecturesIn teaches)

ObjectProperty(hasSameGradeAs Transitive Symmetric domain(student)
rangle(student))
In DL syntax:

\[ 
\top \sqsubseteq \forall \text{age} : \text{nonNegativeInteger} \\
\top \sqsubseteq \forall \text{isTaughtBy}.\text{course} \\
\top \sqsubseteq \forall \text{isTaughtBy}.\text{academicStaffMember} \\
\text{isTaughtBy} \sqsubseteq \text{involves} \\
\text{teaches} \equiv \text{isTaughtBy}^- \\
\top \sqsubseteq \forall \text{teaches}^- . \text{academicStaffMember} \\
\top \sqsubseteq \forall \text{teaches}.\text{course} \\
\text{lecturesIn} \equiv \text{teaches} \\
\text{hasSameGradeAs}^+ \sqsubseteq \text{hasSameGradeAs} \\
\text{hasSameGradeAs} \equiv \text{hasSameGradeAs}^- \\
\top \sqsubseteq \forall \text{hasSameGradeAs}^- . \text{student} \\
\top \sqsubseteq \forall \text{hasSameGradeAs}.\text{student} 
\]
Individual (949318 type(lecturer))

Individual (949352 type(academicStaffMember) value(age "39"^^&xsd;integer))

ObjectProperty(isTaughtBy Functional)

Individual (CIT1111 type(course) value(isTaughtBy 949352) value(isTaughtBy 949318))

DifferentIndividuals (949318 949352) DifferentIndividuals (949352 949111 949318)
More examples

In DL syntax:

949318 : lecturer
949352 : academicStaffMember
⟨949352, "39" ^ &xsd; integer⟩ : age
⊤ ⊑≤ 1 isTaughtBy
CIT 1111 : course
⟨CIT 1111, 949352⟩ : isTaughtBy
⟨CIT 1111, 949318⟩ : isTaughtBy
949318 ≠ 949352
949352 ≠ 949111
949111 ≠ 949318
949352 ≠ 949318
More examples

Class(firstYearCourse partial restriction (isTaughtBy allValuesFrom (Professor)))

Class(mathCourse partial restriction (isTaughtBy hasValue (949352)))

Class(academicStaffMember partial restriction (teaches someValuesFrom (undergraduateCourse)))

Class(course partial restriction (isTaughtBy minCardinality (1)))

Class(department partial restriction (hasMember minCardinality(10)) restriction (hasMember maxCardinality(30)))
In DL syntax:

\[
\text{firstYearCourse} \sqsubseteq \forall \text{isTaughtBy}. \text{Professor}
\]
\[
\text{mathCourse} \sqsubseteq \exists \text{isTaughtBy}. \{949352\}
\]
\[
\text{academicStaffMember} \sqsubseteq \exists \text{teaches.undergraduateCourse}
\]
\[
\text{course} \sqsupseteq 1 \text{isTaughtBy}
\]
\[
\text{department} \sqsupseteq 10 \text{hasMember} \sqcap \leq 30 \text{hasMember}
\]
More examples

\[
\text{Class}(\text{course partial complementOf(staffMember)})
\]

\[
\text{Class}(\text{peopleAtUni complete unionOf(staffMember student)})
\]

\[
\text{Class}(\text{facultyInCS complete intersectionOf ( faculty restriction (belongsTo hasValue (CSDepartment))))}
\]

\[
\text{Class}(\text{adminStaff complete intersectionOf ( staffMember complementOf(unionOf(faculty techSupportStaff))})
\]

*In DL syntax:*

\[
\text{course} \sqsubseteq \neg \text{staffMember}
\]

\[
\text{peopleAtUni} \equiv \text{staffMember} \sqcup \text{student}
\]

\[
\text{facultyInCS} \equiv \text{faculty} \sqcap \exists \text{belongsTo.}\{\text{CSDepartment}\}
\]

\[
\text{adminStaff} \equiv \text{staffMember} \sqcap \neg (\text{faculty} \sqcup \text{techSupportStaff})
\]
Outline

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Layering on top of RDF(S)

- RDF(S) bottom layer in Semantic Web stack
- Higher languages layer on top of RDFS

**Syntactic Layering**

- *Every valid RDF statement is a valid statement in a higher language*
- This *includes* triples containing keywords of these languages(!)

**Semantic Layering**

For RDFS graph $G$ and higher-level language $L$:

If $G \models_{RDFS} G'$ then $G \models_{L} G'$, and *ideally*

if $G \models_{L} G'$ then $G \models_{RDFS} G'$
Syntactically layering OWL on RDF(S)

**OWL Lite, OWL DL**
- OWL Lite, OWL DL subsets of RDF
- Allowed triples defined through mapping from abstract syntax
- *Partial* layering:
  - *every* OWL Lite/DL ontology is an RDF graph
  - *some* RDF graphs are OWL Lite/DL ontologies

**OWL Full**
- OWL Full encompasses RDF
- *Complete* layering:
  - *every* OWL Full is an RDF graph
  - *all* RDF graphs are OWL Full ontologies
### Semantically layering OWL on RDF(S)

#### OWL Lite, OWL DL
- OWL Lite/DL semantics *not* related to RDFS semantics
- Redefine semantics of RDFS keywords, e.g., `rdfs:subClassOf`
- Work ongoing to describe correspondence between subset of RDFS and OWL Lite/DL

#### OWL Full
- OWL Full semantics is *extension* of RDFS semantics
- OWL Full is undecidable
- OWL Full semantics hard to understand
**OWL Lite/DL vs. RDF**

- RDF Graph defined through translation from Abstract Syntax
- Example:

```
Class(Human partial Animal
    restriction(hasLegs cardinality(2))
    restriction(hasName allValuesFrom(xsd:string)))
```

```
Human rdf:type owl:Class
Human rdfs:subClassOf Animal
Human rdfs:subClassOf :X1
:X1 rdf:type owl:Restriction
:X1 owl:onProperty hasLegs
:X1 owl:cardinality "2" xsd:nonNegativeInteger
Human rdfs:subClassOf :X2
:X2 rdf:type owl:Restriction
:X2 owl:onProperty hasName
:X2 owl:allValuesFrom xsd:string
```
Layering OWL on top of RDF(S)

**OWL Lite/DL vs. RDF**

- RDF Graph defined through translation from Abstract Syntax
- Example:

  ```
  Class(Human partial Animal
    restriction(hasLegs cardinality(2))
    restriction(hasName allValuesFrom(xsd:string)))
  ```

  ```
  Human       rdf:type       owl:Class
  Human       rdfs:subClassOf Animal
  Human       rdfs:subClassOf _:X1
  _:X1        rdf:type       owl:Restriction
  _:X1        owl:onProperty hasLegs
  _:X1        owl:cardinality ”2”8sd:nonNegativeInteger
  Human       rdfs:subClassOf _:X2
  _:X2        rdf:type       owl:Restriction
  _:X2        owl:onProperty hasName
  _:X2        owl:allValuesFrom xsd:string
  ```
Not every RDF graph is OWL Lite/DL ontology

Example:

\[
A \quad \text{rdf:type} \quad A
\]

How to check whether an RDF graph \(G\) is OWL DL?

1. Construct an OWL ontology \(O\) in Abstract Syntax
2. Translate to RDF graph \(G'\)
3. If \(G = G'\), then \(G\) is OWL DL
   - Otherwise, go to step (1)
Not every RDF graph is OWL Lite/DL ontology

Example:

A  rdf:type  A

How to check whether an RDF graph $G$ is OWL DL?

Construct an OWL ontology $O$ in Abstract Syntax
Translate to RDF graph $G'$
If $G = G'$, then $G$ is OWL DL
  Otherwise, go to step (1)
Summary

1 Introduction
   - W3C’s layer cake
   - Limitations of RDFS

2 OWL
   - Design of OWL
   - OWL family of languages
   - OWL and Description Logics
   - OWL Syntaxes
   - Layering OWL on top of RDF(S)
Introduction

The future of OWL... is now

- Section 8 of Horrocks et. al.’s paper outlines possible “Future extensions”
- OWL 2 has become a W3C recommendation on 27 Oct 2009
- We look at the new recommendation in the following lectures