Logic and Reasoning in the Semantic Web (part II – OWL)

Fulvio Corno, Laura Farinetti
Politecnico di Torino
Dipartimento di Automatica e Informatica
e-Lite Research Group – http://elite.polito.it
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- OWL reasoning tools

Part II
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- OWL reasoning tools
OWL supports…

- Superset of RDF/RDFS
  - Fact stating facilities from RDF
  - Class property structuring from RDF Schema

- New logical operators
  - Boolean operators
  - Property hierarchies
  - Properties can be defined transitive, functional, inverse...
  - Individuals can be defined instances
  - Equivalence and disjointness statements on classes
  - Equivalence statements on properties
  - Equality and inequality can be asserted between individuals
OWL keywords

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

- **OWL DL**
  - Negation
  - Disjunction
  - Full Cardinality
  - Enumerated types

- **RDF Schema**

- **OWL Full**
  - Allow meta-classes etc
RDFS vs OWL

In RDFS you can:
- declare classes like Artist, Museum or Paintings
- state that Painter is a subclass of Artist
- state that rembrandt is an instances of class Painter
- state that hasPainted is a property, with domain Painter and range Painting.
- state that rembrand is an instance of Dutchman with deathdate value 1669.
RDFS vs OWL

- In OWL you can also:
  - state that Country and Person are disjoint classes
  - state that the nl and england are distinct individuals of the class Country
  - declare hasPainted as inverse property of paintedBy
  - state that the class stateless is defined as those members of the class Person that have no values for the property nationality
  - state that the class Canadian is defined as those members of the class Person that have canada as a value of the property nationality
  - state that age is a functional property.
And now?

- OWL 1 comes with 2 formal reasoning systems
  - OWL-Lite with 1st-order logic
  - OWL-DL with Description Logic
- OWL 1 adds profiles OWL-EL, OWL-QL, OWL-RL
  - Different kinds of entailments are applicable to each profile
  - Different computational complexity
- We can reason by combining
  - Available ontologies
  - Available facts
The ingredients

- Ontology (e.g., in OWL-DL)
- Facts (e.g., objects and instances)
- A reasoning algorithm (for DL)
- A reasoning **engine**
- A way to express queries
- A way to present results
Description Logics: Syntax

- **Concepts** corresponds to classes
- **Roles** correspond to class properties
- **Constructors** mix of set notation and FO quantification

  Booleans: $C \cap D$, $C \sqcup D$, $\neg C$
  Qualified quantification: $\forall \text{R} \cdot C$, $\exists \text{R} \cdot C$

- Variable free notation for concepts (classes)

  - $\text{artist}(x) = \text{person}(x) \land \exists y \text{ created}(x, y) \land \text{Artwork}(y)$
  - is written as $\text{Artist} \sqsubseteq \text{Person} \cap \exists \text{created. Artwork}$
OWL-DL Semantics

- The semantics of OWL-DL constructs is derived by the corresponding Description Logic operators.
- The formal definition is extremely concise
  - [http://www.w3.org/TR/owl-semantics/semantics-all.html#3](http://www.w3.org/TR/owl-semantics/semantics-all.html#3)
  - ...but not so straightforward to understand!
Description Logic: Semantics

- Interpretations are pairs \((\Delta, \cdot^\mathcal{I})\), with a universe \(\Delta\) and a mapping \(\mathcal{I}\) from
  - concept names to subsets of \(\Delta\)
  - role names to binary relations

- Booleans: 
  \[
  \begin{align*}
  C \cap D, & \quad (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I} \\
  C \cup D & \quad (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \\
  \neg C & \quad (\neg C)^\mathcal{I} = \Delta \setminus C^\mathcal{I}
  \end{align*}
  \]

  Qualified quantification:
  \[
  \begin{align*}
  \forall R.C & \quad \forall R.C^\mathcal{I} = \{x \in \Delta \mid \forall y \in \Delta : R^\mathcal{I}(x, y) \rightarrow y \in C^\mathcal{I}\} \\
  \exists R.C & \quad \exists R.C^\mathcal{I} = \{x \in \Delta \mid \exists y \in \Delta : R^\mathcal{I}(x, y) \& y \in C^\mathcal{I}\}
  \end{align*}
  \]
## Modular Definition of Description Logics

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept name</td>
<td>$C$</td>
<td>$C^T$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C_1 \sqcap C_2$</td>
<td>$C_1^T \sqcap C_2^T$</td>
</tr>
<tr>
<td>univ. quant.</td>
<td>$\forall R.C$</td>
<td>${d_1 \mid \forall d_2 \in \Delta^T . (R^T d_1 d_2 \rightarrow d_2 \in C^T)}$</td>
</tr>
<tr>
<td>top</td>
<td>$\top$</td>
<td>$\Delta^T$</td>
</tr>
<tr>
<td>negation ($C$)</td>
<td>$\neg C$</td>
<td>$\Delta^T \setminus C^T$</td>
</tr>
<tr>
<td>disjunction ($\cup$)</td>
<td>$C_1 \sqcup C_2$</td>
<td>$C_1^T \cup C_2^T$</td>
</tr>
<tr>
<td>exist. quant. ($\exists$)</td>
<td>$\exists R.C$</td>
<td>${d_1 \mid \exists d_2 \in \Delta^T . (R^T d_1 d_2 \land d_2 \in C^T)}$</td>
</tr>
<tr>
<td>number restr. ($\geq$)</td>
<td>$\geq nR$</td>
<td>${d_1 \mid</td>
</tr>
<tr>
<td>number restr. ($\leq$)</td>
<td>$\leq nR$</td>
<td>${d_1 \mid</td>
</tr>
<tr>
<td>one-of ($\mathcal{O}$)</td>
<td>${a_1, \ldots, a_n}$</td>
<td>${d \mid d = a_i^T \text{ for some } a_i}$</td>
</tr>
<tr>
<td>filler ($\mathcal{B}$)</td>
<td>$\exists R.{a}$</td>
<td>${d \mid d = R^T d a^T}$</td>
</tr>
<tr>
<td>role name</td>
<td>$R$</td>
<td>$R^T$</td>
</tr>
<tr>
<td>role conj. ($\mathcal{R}$)</td>
<td>$R_1 \sqcap R_2$</td>
<td>$R_1^T \sqcap R_2^T$</td>
</tr>
<tr>
<td>inverse roles ($\mathcal{I}$)</td>
<td>$R^{-1}$</td>
<td>${(d_1, d_2) \mid R^T(d_2, d_1)}$</td>
</tr>
</tbody>
</table>
OWL Formal semantics

- [http://www.w3.org/TR/owl-semantics/semantics-all.html#3](http://www.w3.org/TR/owl-semantics/semantics-all.html#3)

Ingredients:
- Vocabularies Vxx
- Mapping functions (which provide ‘meaning’) EC, ER
- Interpretation of syntax constructs
- Interpretation of axioms and facts
## Interpretation of constructs

<table>
<thead>
<tr>
<th>Abstract Syntax</th>
<th>Interpretation (value of EC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>complementOf(c)</td>
<td>$O - EC(c)$</td>
</tr>
<tr>
<td>unionOf(c₁ ... cₙ)</td>
<td>$EC(c₁) \cup ... \cup EC(cₙ)$</td>
</tr>
<tr>
<td>intersectionOf(c₁ ... cₙ)</td>
<td>$EC(c₁) \cap ... \cap EC(cₙ)$</td>
</tr>
<tr>
<td>oneOf(i₁ ... iₙ), for i_j individual IDs</td>
<td>${S(i₁), ..., S(iₙ)}$</td>
</tr>
<tr>
<td>oneOf(v₁ ... vₙ), for v_j literals</td>
<td>${S(v₁), ..., S(vₙ)}$</td>
</tr>
<tr>
<td>restriction(p x₁ ... xₙ), for n &gt; 1</td>
<td>$EC(\text{restriction}(p \ x₁)) \cap ... \cap EC(\text{restriction}(p \ xₙ))$</td>
</tr>
<tr>
<td>restriction(p allValuesFrom(r))</td>
<td>${x \in O \mid \exists &lt;x,y&gt; \in ER(p) \implies y \in EC(r)}$</td>
</tr>
<tr>
<td>restriction(p someValuesFrom(e))</td>
<td>${x \in O \mid \exists &lt;x,y&gt; \in ER(p) \land y \in EC(e)}$</td>
</tr>
<tr>
<td>restriction(p value(i)), for i an individual ID</td>
<td>${x \in O \mid &lt;x,S(i)&gt; \in ER(p)}$</td>
</tr>
<tr>
<td>restriction(p value(v)), for v a literal</td>
<td>${x \in O \mid &lt;x,S(v)&gt; \in ER(p)}$</td>
</tr>
<tr>
<td>restriction(p minCardinality(n))</td>
<td>${x \in O \mid \text{card}([y \in O\cupLV : &lt;x,y&gt; \in ER(p)]) \geq n}$</td>
</tr>
<tr>
<td>restriction(p maxCardinality(n))</td>
<td>${x \in O \mid \text{card}([y \in O\cupLV : &lt;x,y&gt; \in ER(p)]) \leq n}$</td>
</tr>
<tr>
<td>restriction(p cardinality(n))</td>
<td>${x \in O \mid \text{card}([y \in O\cupLV : &lt;x,y&gt; \in ER(p)]) = n}$</td>
</tr>
<tr>
<td>Individual(\text{annotation}(p₁ o₁) ... \text{annotation}(pₖ oₖ) \ \text{type}(c₁) ... \text{type}(cₘ) pᵥ₁ ... pᵥₙ)</td>
<td>$EC(\text{annotation}(p₁ o₁)) \cap ... \cap EC(\text{annotation}(pₖ oₖ)) \cap \cap \cap EC(pᵥ₁) \cap ... \cap EC(pᵥₙ)$</td>
</tr>
<tr>
<td>Individual(i \ annotation(p₁ o₁) ... \text{annotation}(pₖ oₖ) \ \text{type}(c₁) ... \text{type}(cₘ) pᵥ₁ ... pᵥₙ)</td>
<td>${S(i)} \cap EC(\text{annotation}(p₁ o₁)) \cap ... \cap EC(\text{annotation}(pₖ oₖ)) \cap \cap \cap EC(pᵥ₁) \cap ... \cap EC(pᵥₙ)$</td>
</tr>
<tr>
<td>value(p \ Individual(…))</td>
<td>${x \in O \mid \exists y \in \text{EC}(\text{Individual(…)) : &lt;x,y&gt; \in ER(p)}}$</td>
</tr>
<tr>
<td>value(p id) for id an individual ID</td>
<td>${x \in O \mid &lt;x,S(id)&gt; \in ER(p)}$</td>
</tr>
<tr>
<td>value(p v) for v a literal</td>
<td>${x \in O \mid &lt;x,S(v)&gt; \in ER(p)}$</td>
</tr>
<tr>
<td>annotation(p o) for o a URI reference</td>
<td>${x \in R \mid &lt;x,S(o)&gt; \in ER(p)}$</td>
</tr>
<tr>
<td>annotation(p \ Individual(…))</td>
<td>${x \in R \mid \exists y \in \text{EC}(\text{Individual(…)) : &lt;x,y&gt; \in ER(p)}}$</td>
</tr>
</tbody>
</table>
## Interpretation of axioms and facts

<table>
<thead>
<tr>
<th>Directive</th>
<th>Conditions on interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class(c complete</strong>&lt;br&gt;annotation(p₁ o₁) … annotation(pₖ oₖ)&lt;br&gt;descr₁ … descrₙ)</td>
<td>S(c) ∈ EC(annotation(p₁ o₁)) … S(c) ∈ EC(annotation(pₖ oₖ))&lt;br&gt;EC(c) = EC(descr₁) ∩ … ∩ EC(descrₙ)</td>
</tr>
<tr>
<td><strong>Class(c partial</strong>&lt;br&gt;annotation(p₁ o₁) … annotation(pₖ oₖ)&lt;br&gt;descr₁ … descrₙ)</td>
<td>S(c) ∈ EC(annotation(p₁ o₁)) … S(c) ∈ EC(annotation(pₖ oₖ))&lt;br&gt;EC(c) ⊆ EC(descr₁) ∩ … ∩ EC(descrₙ)</td>
</tr>
<tr>
<td><strong>EnumeratedClass(c</strong>&lt;br&gt;annotation(p₁ o₁) … annotation(pₖ oₖ)&lt;br&gt;i₁ … iₙ)</td>
<td>S(c) ∈ EC(annotation(p₁ o₁)) … S(c) ∈ EC(annotation(pₖ oₖ))&lt;br&gt;EC(c) = { S(i₁), …, S(iₙ) }</td>
</tr>
<tr>
<td><strong>Datatype(c</strong>&lt;br&gt;annotation(p₁ o₁) … annotation(pₖ oₖ) )</td>
<td>S(c) ∈ EC(annotation(p₁ o₁)) … S(c) ∈ EC(annotation(pₖ oₖ))&lt;br&gt;EC(c) ⊆ LV</td>
</tr>
<tr>
<td><strong>DisjointClasses(d₁ … dₙ)</strong></td>
<td>EC(dᵢ) ∩ EC(dⱼ) = { } for 1 ≤ i &lt; j ≤ n</td>
</tr>
<tr>
<td><strong>EquivalentClasses(d₁ … dₙ)</strong></td>
<td>EC(dᵢ) = EC(dⱼ) for 1 ≤ i &lt; j ≤ n</td>
</tr>
<tr>
<td><strong>SubClassOf(d₁ d₂)</strong></td>
<td>EC(d₁) ⊆ EC(d₂)</td>
</tr>
<tr>
<td><strong>DatatypeProperty(p</strong>&lt;br&gt;annotation(p₁ o₁) … annotation(pₖ oₖ)&lt;br&gt;super(s₁) … super(sₙ)&lt;br&gt;domain(d₁) … domain(dₙ) range(r₁) … range(rₙ) )</td>
<td>S(p) ∈ EC(annotation(p₁ o₁)) … S(p) ∈ EC(annotation(pₖ oₖ))&lt;br&gt;ER(p) ⊆ O × LV ∩ ER(s₁) ∩ … ∩ ER(sₙ) ∩ EC(d₁) × LV ∩ … ∩ EC(dₙ) × LV ∩ O × EC(r₁) ∩ … ∩ O × EC(rₙ)&lt;br&gt;[ER(p) is functional]</td>
</tr>
</tbody>
</table>

...
## Interpretation of axioms and facts II

<table>
<thead>
<tr>
<th>Directive</th>
<th>Conditions on interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectProperty(p annotation(p₁ o₁) ... annotation(pₖ oₖ) super(s₁) ... super(sₙ) domain(d₁) ... domain(dₙ) range(r₁) ... range(rₙ))</td>
<td>S(p) ∈ EC(annotation(p₁ o₁)) ... S(p) ∈ EC(annotation(pₖ oₖ)) ER(p) ⊆ O×O ∩ ER(s₁) ∩...∩ ER(sₙ) ∩ EC(d₁)×O ∩...∩ EC(dₙ)×O ∩ O×EC(r₁) ∩...∩ O×EC(rₙ) [ER(p) is the inverse of ER(i)] [ER(p) is symmetric] [ER(p) is functional] [ER(p) is inverse functional] [ER(p) is transitive]</td>
</tr>
<tr>
<td>AnnotationProperty(p annotation(p₁ o₁) ... annotation(pₖ oₖ))</td>
<td>S(p) ∈ EC(annotation(p₁ o₁)) ... S(p) ∈ EC(annotation(pₖ oₖ))</td>
</tr>
<tr>
<td>OntologyProperty(p annotation(p₁ o₁) ... annotation(pₖ oₖ))</td>
<td>S(p) ∈ EC(annotation(p₁ o₁)) ... S(p) ∈ EC(annotation(pₖ oₖ))</td>
</tr>
<tr>
<td>EquivalentProperties(p₁ ... pₙ)</td>
<td>ER(pᵢ) = ER(pⱼ) for 1 ≤ i &lt; j ≤ n</td>
</tr>
<tr>
<td>SubPropertyOf(p₁ p₂)</td>
<td>ER(p₁) ⊆ ER(p₂)</td>
</tr>
<tr>
<td>SameIndividual(i₁ ... iₙ)</td>
<td>S(iⱼ) = S(iₖ) for 1 ≤ j &lt; k ≤ n</td>
</tr>
<tr>
<td>DifferentIndividuals(i₁ ... iₙ)</td>
<td>S(iⱼ) ≠ S(iₖ) for 1 ≤ j &lt; k ≤ n</td>
</tr>
<tr>
<td>Individual([i] annotation(p₁ o₁) ... annotation(pₖ oₖ) type(c₁) ... type(cₘ) pv₁ ... pvₙ)</td>
<td>EC(Individual([i] annotation(p₁ o₁) ... annotation(pₖ oₖ) type(c₁) ... type(cₘ) pv₁ ... pvₙ)) is nonempty</td>
</tr>
</tbody>
</table>
Definitions (I)

- Let D be a datatype map. An Abstract OWL interpretation, I, with respect to D with vocabulary consisting of $V_L$, $V_C$, $V_D$, $V_I$, $V_{DP}$, $V_{IP}$, $V_{AP}$, $V_O$, satisfies an OWL ontology, O, iff:
  - each URI reference in O used as a class ID (datatype ID, individual ID, data-valued property ID, individual-valued property ID, annotation property ID, annotation ID, ontology ID) belongs to $V_C$ ($V_D$, $V_I$, $V_{DP}$, $V_{IP}$, $V_{AP}$, $V_O$, respectively);
  - each literal in O belongs to $V_L$;
  - I satisfies each directive in O, except for Ontology Annotations;
  - there is some $o \in R$ with $<o, S(owl:Ontology) > \in ER(rdf:type)$ such that for each Ontology Annotation of the form $Annotation(p v)$, $<o, S(v) > \in ER(p)$ and that if O has name n, then $S(n) = o$;
  - I satisfies each ontology mentioned in an owl:imports annotation directive of O.
Definitions (II)

- A collection of abstract OWL ontologies and axioms and facts is consistent with respect to datatype map D iff there is some interpretation I with respect to D such that I satisfies each ontology and axiom and fact in the collection.

- A collection $O$ of abstract OWL ontologies and axioms and facts entails an abstract OWL ontology or axiom or fact $O'$ with respect to a datatype map $D$ if each interpretation with respect to map $D$ that satisfies each ontology and axiom and fact in $O$ also satisfies $O'$.
Reasoning

- With the definition of the semantics, we may now define some reasoning methods
  - Reasoning on the structure of the ontology
  - Reasoning on relationships among classes
  - Reasoning on instances
Concept Reasoning

Based on these semantics there are two basic reasoning services:

- **Concept satisfiability**, $\models C \neq \bot$.
  - Check whether for some interpretation $\mathcal{I}$ we have $C^\mathcal{I} \neq \emptyset$.
  - $\models \forall \text{creates}. \text{Sculpture} \sqcap \exists \text{creates}. (\text{Artwork} \sqcap \neg \text{Sculpture}) = \bot$.

- **Concept subsumption**, $\models C_1 \subseteq C_2$.
  - Check whether for all interpretations $\mathcal{I}$ we have $C_{1}^\mathcal{I} \subseteq C_{2}^\mathcal{I}$.
  - $\forall \text{creates}. \text{Painting} \sqcap \exists \text{creates}. \top \sqsubseteq \exists \text{creates}. \text{Painting}$. 
Terminological Reasoning

\[ \mathcal{I} = \{ \text{Painting} \sqsubseteq \text{Artwork} \sqcap \neg \text{Sculpture}, \]
\[ \text{Painter} \sqsubseteq \exists \text{creates. Paintings}, \]
\[ \text{Sculpturer} \sqsubseteq \exists \text{creates. Artwork} \sqcap \forall \text{creates. Sculpture} \} \]

- **Concept satisfiability**, \( \Sigma \models C \neq \bot \).
  - Check whether there is an interpretation \( \mathcal{I} \) such that \( \mathcal{I} \models \Sigma \) and \( C^\mathcal{I}_1 \subseteq C^\mathcal{I}_2 \).
  - **Concept unsatisfiability**: \( \Sigma \models \text{Painter} \sqcap \text{Sculpturer} = \bot \).

- **Subsumption**, \( \Sigma \models C_1 \sqsubseteq C_2 \).
  - Check whether for all interpretations \( \mathcal{I} \) such that \( \mathcal{I} \models \Sigma \) we have \( C^\mathcal{I}_1 \subseteq C^\mathcal{I}_2 \).
  - **Subsumption**: \( \Sigma \models \text{Painter} \sqsubseteq \neg \text{Sculpurer} \)
Assertional reasoning

\[ \mathcal{A} = \{ \text{rembrandt:Artist}, \text{nightwatch:Painting}, \text{(rembrandt,nightwatch):created} \} \]
and \( \Sigma = \langle \mathcal{T}, \mathcal{A} \rangle \)

- **Consistency**, \( \Sigma \not\models \bot \).
  - Check whether there exists \( \mathcal{I} \) such that \( \mathcal{I} \models \Sigma \).
  - \( \Sigma \models \mathcal{A} \neq \bot \) but \( \Sigma \models \mathcal{A} \cup \{ \text{rembrandt:Sculpturor} \} = \bot \)

- **Instance Checking**, \( \Sigma \models a : C \).
  - Check whether \( a^\mathcal{I} \in C^\mathcal{I} \) for all interpretations \( \mathcal{I} \models \Sigma \).
  - \( \text{rembrandt} \in \Sigma \text{ Painter} \).

- **Defined reasoning tasks**:
  - **Retrieval**: \( \text{retrieve(Artists)} = \{ \text{rembrandt} \} \).
  - **Realization**: find most specific concepts in \( \mathcal{T} \) for instances in \( \mathcal{A} \)
    \( \text{realize(rembrandt)} = \text{Painter} \)
Is this applicable?

- Logic = Perfect reasoning under perfect conditions
- therefore…
  - unlimited time
  - homogeneous knowledge
  - correct and consistent knowledge
Is Logic good?

**Pro**
- Strong theoretical basis
- Well known properties
- Well known implementation techniques

**Con**
- Strict (no "good enough" answers)
- Abrupt (no intermediate answers)
- Inefficient (no time/quality trade-off)
Practical limitations

- Terminologies will be sloppy
  - Made by non-experts
  - Made by machines
  - Inference rules break!
- No standard vocabulary
  - Communication problems
  - Need approximate equivalence
- Computational explosion
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- OWL reasoning tools
What is an OWL-DL reasoner

- The official normative definition:
  - An *OWL consistency checker* takes a document as input, and returns one word being Consistent, Inconsistent, or Unknown. [J. J. Carroll, J. D. Roo, OWL Web Ontology Language Test Cases, W3C Recommendation http://www.w3.org/TR/owl-test/ (2004).]
  - Rather restrictive... and not very useful for ontology development, debug and querying
Practical Description Logics

- Most theoretical works on Description Logics are concerned with the “upper” part of the ontology (classes, relationships)
- Object Instances are equally important, if not more, in the Semantic Web
- Reasoning over instances is “easier”, but their number may be far larger than the number of classes
## DL Jargon

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Stands for</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABox</td>
<td>Assertional Box</td>
<td>Component that contains assertions about individuals, i.e. OWL facts such as type, property-value, equality or inequality assertions.</td>
</tr>
<tr>
<td>TBox</td>
<td>Terminological Box</td>
<td>Component that contains axioms about classes, i.e. OWL axioms such as subclass, equivalent class or disjointness axioms.</td>
</tr>
<tr>
<td>KB</td>
<td>Knowledge Base</td>
<td>A combination of an ABox and a TBox, i.e. a complete OWL ontology.</td>
</tr>
</tbody>
</table>
Classical Types of Logic Inference

- **Consistency checking**, which ensures that an ontology does not contain any contradictory facts.
  - The OWL Abstract Syntax & Semantics document [S&AS] provides a formal definition of ontology consistency that Pellet uses.
  - In DL terminology, this is the operation to check the consistency of an ABox with respect to a Tbox.
  - Equivalent to OWL Consistency Checking
Classical Types of Logic Inference

- **Concept satisfiability**, which checks if it is possible for a class to have any instances. If class is unsatisfiable, then defining an instance of the class will cause the whole ontology to be inconsistent.
Classical Types of Logic Inference

- **Classification**, which computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all or only the direct subclasses of a class.
Classical Types of Logic Inference

- **Realization**, which finds the most specific classes that an individual belongs to; or in other words, computes the direct types for each of the individuals. Realization can only be performed after classification since direct types are defined with
Reasoning approaches

- **Forward Chaining Inference**
  - Compute all the facts that are entailed by the currently asserted facts

- **Backward Chaining Inference**
  - Starting from an unknown fact that we want to know (whether it’s true or not), try to construct a chain of entailments rooting back in the known facts
Forward Chaining
Backward chaining
Comparison

**Forward Chaining**
- After reasoning, all queries are straightforward
- Much memory may be needed for inferred model
- May be computationally intensive at startup
- Difficult to update when facts are removed/modified

**Backward Chaining**
- Does not compute whole model
- Usually faster
- Each query needs to re-compute part of the model (caching is essential)
- No start-up overhead
- Lower memory requirements
- Efficiency depending on exploration strategies/heuristics
Reasoning and Rules

• Reasoning amounts to applying a set of rules (e.g., Entailment rules) to a knowledge base
• Some sets of rules are hard-coded to OWL-dialect semantics
  • Implemented through ad-hoc algorithms
  • Programmed in general-purpose rule systems
    • SWRL, JenaRules, Drools, …
• General-purpose rule systems might also be used by programmers to manipulate the knowledge base (even outside «official» entailments)
Some OWL reasoners

- Fact++
  - C++, OpenSource, OWL-DL, [http://owl.man.ac.uk/factplusplus/](http://owl.man.ac.uk/factplusplus/)
- Hermit
- Kaon2
- Pellet
- RacerPro
- Vampire
Pellet: an OWL-DL Reasoner

- Pellet is a complete and feature-rich reasoner for OWL-DL
- Available open-source at http://clarkparsia.com/pellet
- Available standalone, or integrated with the major ontology development environments (including Protégé)
Using Reasoners in Protégé

(hands-on demo)
References

- Course material for “Practical Reasoning for the Semantic Web” course at the 17th European Summer School in Logic, Language and Information (ESSLLI)
  - [http://www.few.vu.nl/~schlobac/](http://www.few.vu.nl/~schlobac/)

References

  - [http://www.w3.org/TR/owl-semantics/](http://www.w3.org/TR/owl-semantics/)
References

- http://media.cwi.nl/survey/
- http://dose.sourceforge.net/
- http://www.mkbergman.com
License

- This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License.

- To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.