Outline

• Ontologies.
• Descriptions Logics
• Ontology Web Language (OWL)
• Tools
Ontologies

An ontology is a representation scheme that describes a formal conceptualization of a domain of interest.

The specification of an ontology comprises several levels:
• Meta-level: specifies a set of modeling categories.
• Intensional level: specifies a set of conceptual elements (instances of categories) and of rules to describe the conceptual structures of the domain.
• Extensional level: specifies a set of instances of the conceptual elements described at the intensional level.
Intensional level of an ontology language

(The intensional level of) an Ontology is typically rendered as a diagram (e.g., Semantic Network, Entity-Relationship schema, UML Class Diagram).

E.g., an ontology rendered as UML Class Diagram
Ontological Engineering

The real ontologies that exist have been created along four main routes:

- By a team of trained ontologist/logicians, who architect the ontology and write axioms. For instance, the Common Information Model (CIM), a standard developed by the electric power industry, contains semantic models, rendered as UML class diagram) which can be mapped into ontologies.

- By importing categories, attributes and values from an existing database or databases. For instance, DBPEDIA was created from Wilkipedia (http://dbpedia.org/About).

- By parsing text documents and extracting information from them. For instance, TEXTRUNEER was created by reading a large corpus of Web Pages (http://openie.cs.washington.edu/).

- By enticing unskilled amateurs to enter common knowledge. For instance, the OPENMIND system was build by volunteers (a project from MIT and IBM).
An example of a real application

**Inquire: An Intelligent Textbook**, form the Artificial Intelligence Center of SRI (http://www.ai.sri.com/):

- http://www.aaaivideos.org/2012/inquire_intelligent_textbook/
Ontologies and Reasoning

- Ontologies are logical theories, and several interpretations may exist that satisfy them (incomplete information)

- Reasoning over ontologies amounts to make logical inference over them
  - Intensional reasoning: concept/relationship satisfiability, concept/relationship subsumption, etc.
  - Ontology reasoning: ontology satisfiability, instance checking, query answering.
What are Description Logics?

Description Logics are logics specifically designed to represent and reasoning on structured knowledge.

- First-order logic is designed to make it ease to say things about objects.
- Description logics are notations that are designed to make easier to describe definitions and properties of categories (sets of objects).

The domain is composed of objects and is structured into:

- concepts, which correspond to classes, and denote sets of objects,
- roles, which correspond to (binary) relations, and denote binary relations on objects.

The knowledge is asserted through so-called assertions, i.e., logical axioms.
An example of a concept description

“A happy man that is married to a doctor, and all of whose children are either doctors or professors”

This concept can be expressed as follows:

\[ \text{HappyMan} \equiv \text{Human} \cap \neg \text{Female} \cap (\exists \text{married.Dr}) \cap (\forall \text{hasChild.}(\text{Doctor} \cup \text{Professor})) \]
## Concept constructors

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>A</td>
<td>Doctor</td>
<td>$A_i \subseteq \Delta_i$</td>
</tr>
<tr>
<td>atomic role</td>
<td>P</td>
<td>hasChild</td>
<td>$P_i \subseteq \Delta_i \times \Delta_i$</td>
</tr>
<tr>
<td>atomic negation</td>
<td>$\neg A$</td>
<td>$\neg$Doctor</td>
<td>$\Delta_i \setminus A_i$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
<td>Hum $\sqcap$ Male</td>
<td>$C_i \cap D_i$</td>
</tr>
<tr>
<td>(unqual.) exist. res.</td>
<td>$\exists R$</td>
<td>$\exists$hasChild</td>
<td>${ o \mid \exists o'. (o, o') \in R_i }$</td>
</tr>
<tr>
<td>value restriction</td>
<td>$\forall R.C$</td>
<td>$\forall$hasChild.Male</td>
<td>${ o \mid \forall o'. (o, o') \in R_i \rightarrow o' \in C_i }$</td>
</tr>
<tr>
<td>bottom</td>
<td>$\bot$</td>
<td></td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

($C, D$ denote arbitrary concepts and $R$ an arbitrary role)

The above constructs form the basic language $\mathcal{AL}$ of the family of $\mathcal{AL}$ Languages
Description Logics ontology (or knowledge base)

An ontology is a pair $O = \langle T, A \rangle$, where $T$ is a TBox and $A$ is an ABox:

### Description Logics TBox

Consists of a set of **assertions** on concepts and roles:

- **Inclusion assertions on concepts**: $C_1 \sqsubseteq C_2$
- **Inclusion assertions on roles**: $R_1 \sqsubseteq R_2$
- **Property assertions on (atomic) roles**:
  - (transitive $P$)
  - (symmetric $P$)
  - (functional $P$)
  - (reflexive $P$)

### Description Logics ABox

Consists of a set of **membership assertions** on individuals:

- **for concepts**: $A(C)$
- **for roles**: $P(c_1, c_2)$

(we use $c_i$ to denote individuals)
An example of a Description Logic Knowledge Base

NOTE: We use $C_1 \equiv C_2$ as an abbreviation for $C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1$

**Tbox assertions:**
- Inclusion assertions on concepts:
  \[
  \begin{align*}
  \text{Father} & \equiv \text{Human} \sqcap \text{Male} \sqcap \exists \text{hasChild} \\
  \text{HappyFather} & \sqsubseteq \text{Father} \sqcap \forall \text{hasChild.HappyPerson} \\
  \text{HappyAnc} & \sqsubseteq \forall \text{descendant.HappyFather} \\
  \text{Teacher} & \sqsubseteq \lnot \text{Doctor} \sqcap \lnot \text{Lawyer}
  \end{align*}
  \]
- Inclusion assertions on roles: $\text{hasChild} \sqsubseteq \text{descendant}$
- Property assertions on roles:
  \[
  (\text{transitive descendant})(\text{reflexive descendant})(\text{functional hasFather})
  \]

**Abox membership assertions:**
\[
\begin{align*}
\text{Teacher(mary)} & \quad \text{hasFather(mary, john)} \\
\text{HappyAnc(john)} &
\end{align*}
\]
Complexity of reasoning over DL ontologies

• **Bad news**
  • *Without restriction* on the form of Tbox assertions, reasoning over DL ontologies is already \textit{ExpTime-Hard}, even for every simple DLs.

• **Good news:**
  • We can add a lot of \textit{expressivity} (i.e., essentially all DL constructs seen so far), while still staying within the ExpTime upper bound.
  • There are DL reasoners that perform reasonably well in practice for such DLs (e.g., Racer, Pellet, Fact++, . . . )
Relationship between DLs and ontology formalisms

- Description Logics are nowadays advocated to provide the foundations for ontology languages.
- Different versions of the Web Ontology Language (OWL) have been defined as syntactic variants of certain Description Logics.
- DLs are also ideally suited to capture the fundamental features of conceptual modeling formalisms used in information systems design:
  - **Entity-Relationship diagrams**, used in database conceptual modeling.
  - **UML Class Diagrams**, used in the design phase of software applications
Description Logics vs. OWL

DLs provide the foundations for standard ontology languages.

Different versions of the W3C standard Web Ontology Language (OWL) have been defined as syntactic variants of certain DLs:

- **OWL Lite** is a variant of the DL SHIF (D), where:
  - S stands for ALC extended with transitive roles,
  - H stands for role hierarchies (i.e., role inclusion assertions),
  - I stands for inverse roles,
  - F stands for functionality of roles,
  - (D) stand for data types, which are necessary in any practical knowledge representation language.

- **OWL DL** is a variant of SHOIN (D), where:
  - O stands for nominals, which means the possibility of using individuals in the TBox (i.e., the intensional part of the ontology),
  - N stands for (unqualified) number restrictions.
Description Logics vs. OWL

There is another version of OWL which is called: OWL Full. However the semantics of OWL Full is given by an extension of the RDF model theory.

An example in OWL syntax of the concept expression: \( Human \sqcap Male \)

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:class rdf:about="#Human"/>
    <owl:class rdf:about="#Male"/>
  </owl:intersectionOf>
</owl:Class>
```

while \( (\geq 2 \ hasChild.\ Thing) \) would be written as:

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasChild"/>
  <owl:minCardinality>
    <owl:datatypedefinition rdf:datatype="&xsd;NonNegativeInteger" > 2
  </owl:minCardinality>
</owl:Restriction>
```
## DL constructs vs. OWL constructs

<table>
<thead>
<tr>
<th>OWL construct</th>
<th>DL construct</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectIntersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
</tr>
<tr>
<td>ObjectUnionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
</tr>
<tr>
<td>ObjectComplementOf</td>
<td>$\neg C$</td>
<td>$\neg$Male</td>
</tr>
<tr>
<td>ObjectOneOf</td>
<td>${a_1} \sqcup \ldots \sqcup {a_n}$</td>
<td>${john} \sqcup {mary}$</td>
</tr>
<tr>
<td>ObjectAllValuesFrom</td>
<td>$\forall P . C$</td>
<td>$\forall$hasChild.Doctor</td>
</tr>
<tr>
<td>ObjectSomeValuesFrom</td>
<td>$\exists P . C$</td>
<td>$\exists$hasChild.Lawyer</td>
</tr>
<tr>
<td>ObjectMaxCardinality</td>
<td>$(\leq n P)$</td>
<td>$(\leq 1$ hasChild$)$</td>
</tr>
<tr>
<td>ObjectMinCardinality</td>
<td>$(\geq n P)$</td>
<td>$(\geq 2$ hasChild$)$</td>
</tr>
</tbody>
</table>

... 

**Note:** all constructs come also in the Data... instead of Object...
## DL axioms vs. OWL axioms

<table>
<thead>
<tr>
<th>OWL axiom</th>
<th>DL syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubClassOf</td>
<td>C1 ⊑ C2</td>
<td>Human ⊑ Animal ⊓ Biped</td>
</tr>
<tr>
<td>EquivalentClasses</td>
<td>C1 ≡ C2</td>
<td>Man ≡ Human ⊓ Male</td>
</tr>
<tr>
<td>DisjointClasses</td>
<td>C1 ⊑ ¬C2</td>
<td>Man ⊑ ¬Female</td>
</tr>
<tr>
<td>SameIndividual</td>
<td>{a1} ≡ {a2}</td>
<td>{presBush} ≡ {G.W.Bush}</td>
</tr>
<tr>
<td>DifferentIndividuals</td>
<td>{a1} ⊑ ¬{a2}</td>
<td>{john} ⊑ ¬{peter}</td>
</tr>
<tr>
<td>SubObjectPropertyOf</td>
<td>P1 ⊑ P2</td>
<td>hasDaughter ⊑ hasChild</td>
</tr>
<tr>
<td>EquivalentObjectProperties</td>
<td>P1 ≡ P2</td>
<td>hasCost ≡ hasPrice</td>
</tr>
<tr>
<td>InverseObjectProperties</td>
<td>P1 ⊑ P− 2</td>
<td>hasChild ≡ hasParent−</td>
</tr>
<tr>
<td>TransitiveObjectProperty</td>
<td>P+ ⊑ P</td>
<td>ancestor+ ⊑ ancestor</td>
</tr>
<tr>
<td>FunctionalObjectProperty</td>
<td>(functional P )</td>
<td>(functional hasFather)</td>
</tr>
</tbody>
</table>

...
Ontologies in Semantic Web

User interface and applications

Trust

Proof

Unifying logic

Ontologies: OWL

Rules: RIF/SWRL

Taxonomies: RDFS

Data interchange: RDF

Syntax: XML

Identifiers: URI

Character set: UNICODE

Cryptography

Semantic Web

Normal Web
SPARQL and Rule Languages

- **Query Language for RDF (SPARQL)**
  - Query language for RDF triples
  - A protocol for querying RDF data over the Web
  - E.g. language used to query the repository from the user interface
  - Can also be used for Updates

- **Rule languages (e.g. Rule Interchange Format (RIF), Semantic Web Rule Language (SWRL))**
  - W3C recommendation for exchanging rule sets between rule engines
  - Extend ontology languages with proprietary axioms
  - Based on different types of logics
    - Description Logic
    - Logic Programming
  - E.g. used to enable reasoning over data to infer new knowledge
Data Interoperability with semantic models: An example

Data Interoperability with semantic models: An example

- Instance of a knowledge base profile
  - Conform to CIM and IEC 61850
  - Conform to semantics
  - Inference queries: SPARQL, SWRL, Description Logic Inference, Description Logic Programs, etc.

- Knowledge base profile
  - Conform to Syntax
  - Restriction of vocabulary

- Harmonized semantic data model between CIM and IEC 61850
  - Subset of CIM and 61850

- Software artifacts, e.g. electric models, XML messages, SQL schemas, SCL files, FIPA SCL messages, etc.
Some Tools for Ontological Engineering

Ontology design:

- **Protégé** (Stanford University: [http://protege.stanford.edu/](http://protege.stanford.edu/)).
- **SWOOP** (University of Maryland): [https://code.google.com/p/swoop/](https://code.google.com/p/swoop/)
- **OilEd** (University of Manchester) [http://oiled.semanticweb.org/index.shtml](http://oiled.semanticweb.org/index.shtml)
- **TopBraid Composer** [http://www.topquadrant.com](http://www.topquadrant.com)

Reasoning support:

- **FaCT++** OWL-DL Reasoner ([http://owl.man.ac.uk/factplusplus/](http://owl.man.ac.uk/factplusplus/))
- **RACER** OWL -DL Reasoner ([http://racer.sts.tuhh.de/](http://racer.sts.tuhh.de/))
Sources of this Lecture

• F. Baader, I. Horrocks, U. Sattler, Description Logic (Chapter), Handbook of Knowledge Representation, 2008