An introduction to ontologies

Amedeo Napoli

LORIA – UMR 7503 – BP 239, 54506 Vandœuvre-lès-Nancy

Email : Amedeo.Napoli@loria.fr

http://www.loria.fr/˜napoli/

http://www.loria.fr/LORIA/EXT/equipes/ORPAILLEUR/

Thematic meetings

Workshop on ontologies, Strasbourg, October, 25th

ACI " Masse de données "

Summary of the tutorial

- A general introduction to knowledge representation and ontologies.

Summary of the tutorial

- A general introduction to knowledge representation and ontologies.
- Inside Ontologies and ontology engineering.

Summary of the tutorial

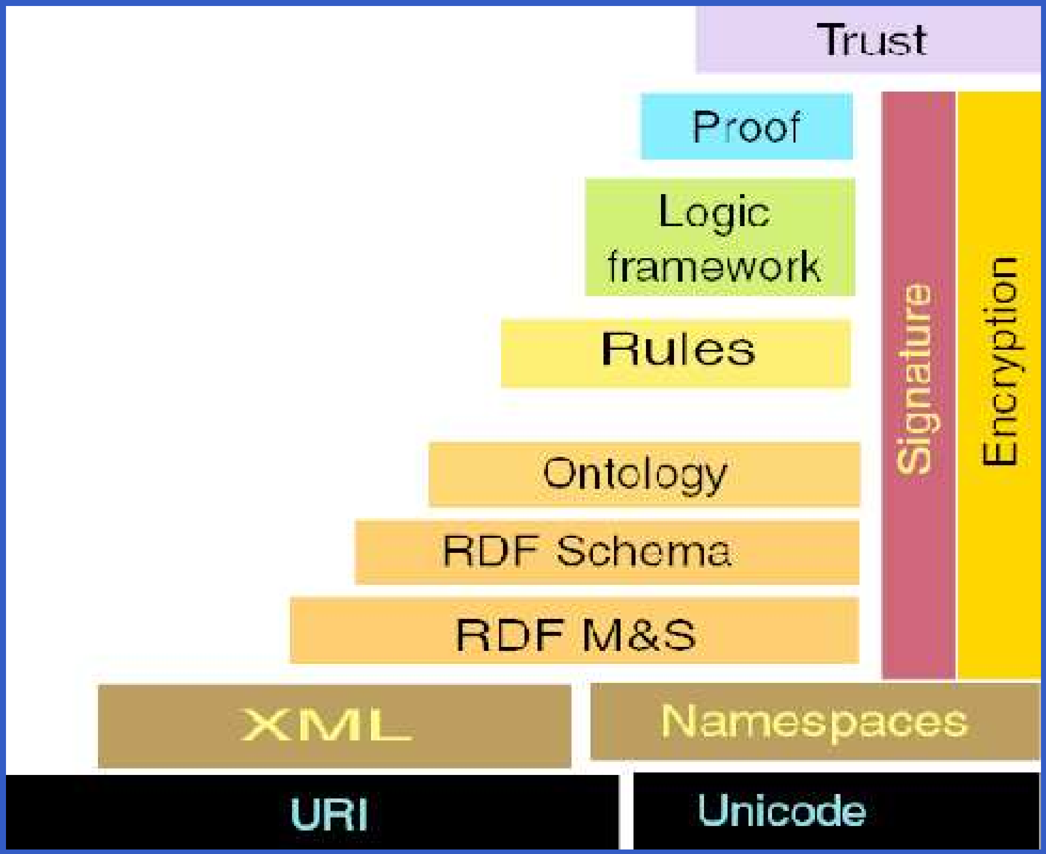
- A general introduction to knowledge representation and ontologies.
- Inside Ontologies and ontology engineering.
- Description logics as ontology language ($\mathcal{SHIQ}$ and OWL).

Summary of the tutorial

- A general introduction to knowledge representation and ontologies.
- Inside Ontologies and ontology engineering.
- Description logics as ontology language ($\mathcal{SHIQ}$ and OWL).
- An example of ontology and reasoning within an ontology.

The Semantic Web “cake”

The Semantic Web "cake"



The Semantic Web "cake"

The manipulation of documents for the Semantic Web

- There is a need for structures for recording, disseminating, and exchanging information and knowledge units.
- For being accessible and processable by machines in an intelligent way, the semantics of documents has to be explicitly given: this is exactly the purpose of **knowledge representation** languages, of **ontologies**, and of document **content annotations**.
- An intelligent manipulation of documents is based on the exploitation of the content and of the **semantics** of the documents, with respect to the knowledge on the domain of documents.

The Semantic Web "cake"

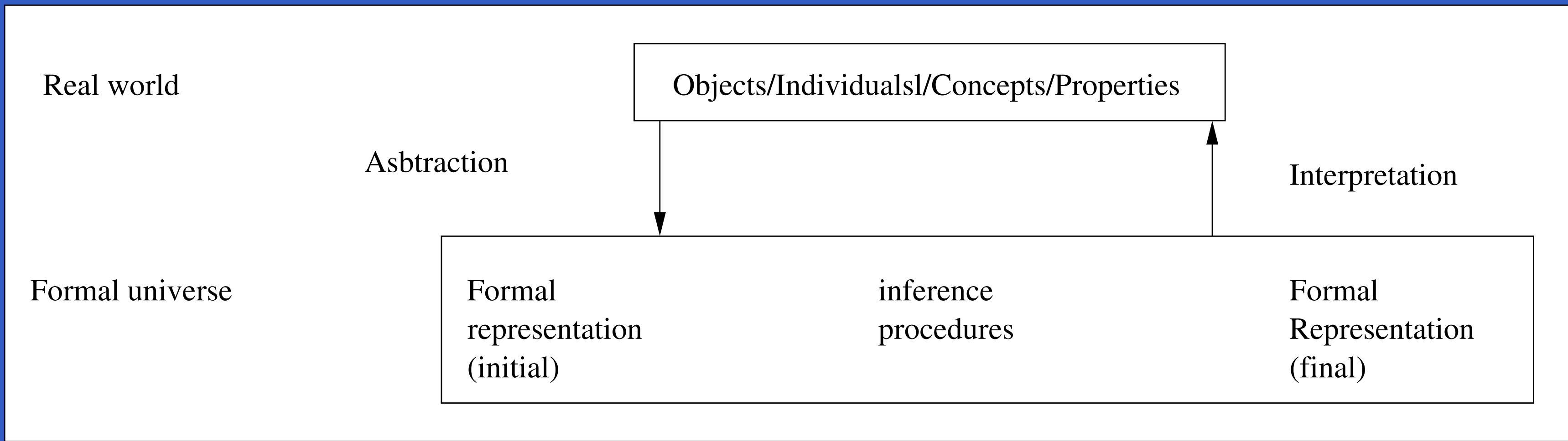
The interpretation and the annotation of documents has to be guided by domain knowledge

- The description of a document in a given context must rely on elements of the content of the document, metadata (*Dublin core*), and **annotations** (built according to domain knowledge).
- A semantics can be attached to documents –and their content– using XML, RDF(S), and knowledge representation languages, e.g. description logics.
- Information extraction, i.e. extraction of key terms from documents, and data mining –especially text mining– may be used for analyzing and classifying documents with respect to their content.

The purpose of knowledge representation

The purpose of knowledge representation

A view of knowledge representation



The purpose of knowledge representation

- **Data: uninterpreted, raw**.
  ...—... E !
- **Information: meaning attached to data**.
  - SOS, a letter (or the notation of a scale), a symbol mark...
- **Knowledge: attach purpose and competence to information, generate actions**.
  - if emergency alert then start rescue operations,
  - in a musical context, if E is attached to a score line, then play the E scale,
  - the sentence that precedes ! has to be interpreted as an interjection.

The purpose of knowledge representation

- Knowledge units rely on expertise, experiences, explanations, strategies...
- Knowledge units can be made explicit by asking an expert or may be implicit in databases on a given domain. In this case, tools must be made available for extracting knowledge units from databases.
- Reasoning must be formalized in accordance with the struccture of knowledge units for carrying out inferences on a sound and complete basis.

A first view of ontologies

A first view of ontologies

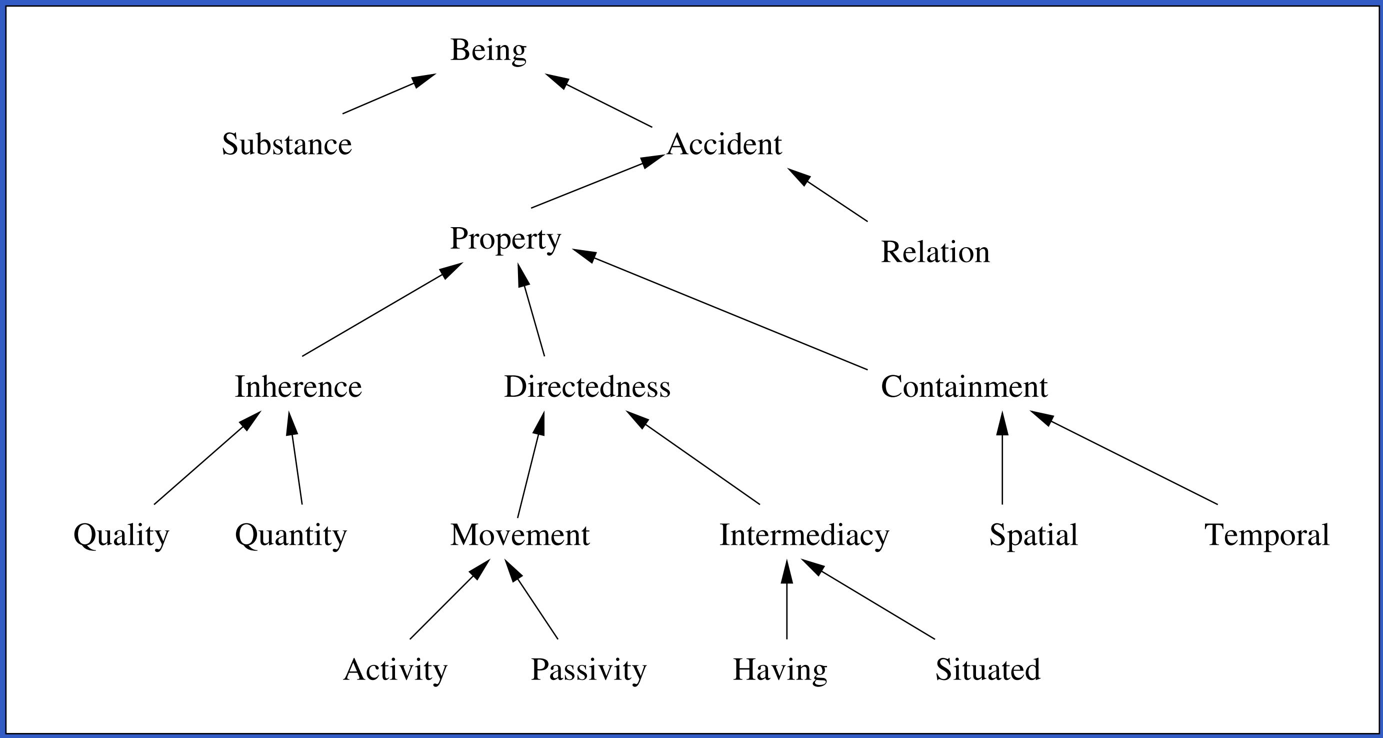
- There have been many attempts to defined what constitutes an ontology, and perhaps the best known (in computer science) being due to Gruber **"an ontology is an explicit specification of a conceptualization"**.
- In this concept, a **conceptualization** means an abstract model of some aspect of the world, taking the form of a definition of the properties of important concepts and relationships.
- An **explicit specification** means that the model should be specified in some unambiguous language, making it amenable to processing by machines as well as by humans.

A first view of ontologies

- Ontologies are becoming of increasing importance in fields such as knowledge management, information integration, cooperative information systems, information retrieval, and electronic commerce.
- The application area which has recently seen an explosion of interest is the Semantic Web, where ontologies are set to play a key role in establishing a **common terminology** between agents, thus ensuring that different agents have a shared understanding of terms using in semantic markup.
- The effective use of ontologies requires not only a well-designed and well-defined ontology language, but also support from reasoning tools.

A first view of ontologies

An example: a part of the ontology of Aristotle



A first view of ontologies

Formally, an ontology $\mathcal{O}$ is a symbol system consisting of:

- A set $\mathtt{S_C}$ of concepts, and a set $\mathtt{S_R}$ of binary relations specifying pairs $(\mathtt{D}, \mathtt{R})$ of domains and ranges (in $\mathtt{S_C}$).
- A hierarchy $\mathtt{H}$ where concepts and relations are hierarchically related by a **subsumption** relation $\sqsubseteq$ (a partial ordering), where $\mathtt{c}_1 \sqsubseteq \mathtt{c}_2$ ($\mathtt{r}_1 \sqsubseteq \mathtt{r}_2$) means that $\mathtt{c}_1$ is a subconcept of $\mathtt{c}_2$ ($\mathtt{r}_1$ is a subrelation of $\mathtt{r}_2$).
- A set $\mathtt{A}$ of ontology axioms including introduction of concepts and of relations.

Elements on ontology engineering

Elements on ontology engineering

- **Kickoff**: ontology requirement specification.
- **Refinement**: produce a mature and application-oriented target ontology according to the specification (knowledge elicitation and formalization).
- **Evaluation**: prove the usefulness of the developed ontology and the associated software environment.
- **Maintenance**: as things are constantly changing so do the specification for an ontology, and these changes must be reflected in the developed ontology, with the guarantee of coherence and compatible upgrade.

Elements on ontology engineering

- **Reasoning** is important to ensure the quality of an ontology, and it can be used in different phases of the ontology life cycle.
- During the ontology design, reasoning can be used to test whether concepts are non-contradictory, and to derive implied relations.
- For example, one usually wants to compute the concept hierarchy, i.e. the partial ordering of named concepts based on subsumption relationship.

Elements on ontology engineering

- Information on which concept is a specialization of another, and which concepts are synonyms, can be used in the design phase to test whether the concept definitions in the ontology have the intended consequences or not.
- Reasoning may also be used when ontology is deployed, for determining the consistency of facts stated in annotations, or infer relationships between annotations instances and ontology classes.
- Interoperability and integration of different ontologies is an important issue.
  For example, after asserting some inter-ontology relationships, the integrated concept hierarchy is computed and the concepts are checked for consistency.

Languages for representing ontologies

Languages for representing ontologies

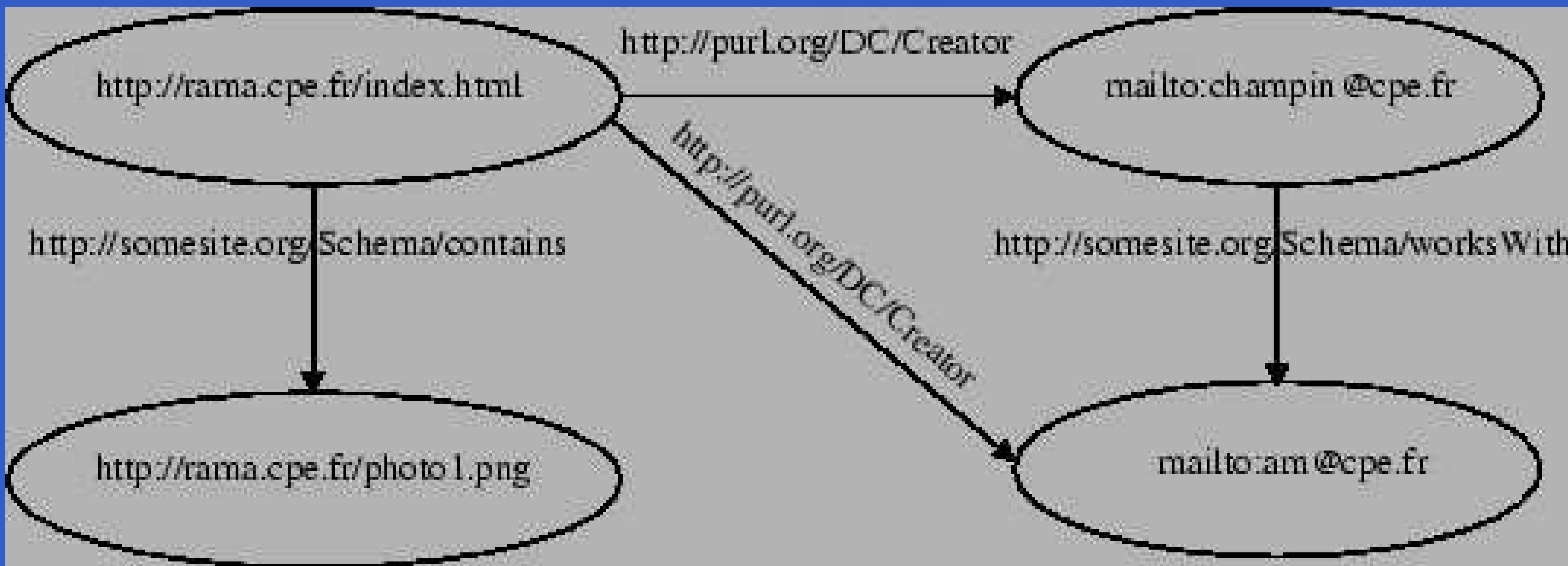
- XML is a language for describing documents.
- RDF and RDFS are languages for descrbing the organization of resources on the Web.
- Description logics (and OWL) are knowledge representation languages, that are well-founded, useful and efficient enough for being the basis of knowledge representation languages for the Semantic Web, and thus for representing ontologies...

Languages for representing ontologies

- RDF identifies resources with qualified **uniform resource identifiers** or URI.
- A resource –the **subject**– is linked to another resource –the **object**– through an arc labeled with a third resource, the **predicate**.
- The **"subject"** has a *property* –the **"predicate"**– *valued* by the **"object"**: **Champin is the creator of index.html**

Languages for representing ontologies

- All the triples may be combined to form a directed graph whose nodes and arcs are lebeled with qualified URIs.
- Moreover, a resource may have more than one value for a given property.



Languages for representing ontologies

- Hierarchies of triples can be represented in RDFS (the `rdfs:subClassOf` property holds between resources of type `rdfs:Class`).
- The expressivity of RDF and RDF Schema is limited: RDF is (roughly) limited to binary ground predicates, and RDF Schema is (roughly) limited to a subclass hierarchy and a property hierarchy, with domain and range definitions of these properties.

Requirements for an ontology language

Requirements for an ontology language

Ontology languages allow users to write explicit, formal conceptualization of domain models. The main requirements are:

- a well-defined syntax, and a well-defined semantics,
- an efficient reasoning support,
- a sufficient expressive power, and a convenience of expression.
- Semantics is a prerequisite for reasoning support, which allows to: (1) check the consistency of the ontology and the knowledge, (2) check the consistency of the ontology and the knowledge, (3) automatically classify instances in classes...

Requirements for an ontology language

Reasoning tasks on ontological knowledge

- Class membership: if $\mathtt{x}$ is an instance of a class $\mathtt{C}$, and $\mathtt{C}$ is a subclass of $\mathtt{D}$, then we can infer that $\mathtt{x}$ is an instance of $\mathtt{D}$.
- Equivalence of classes: is $\mathtt{C}$ is equivalent to $\mathtt{D}$, and $\mathtt{D}$ to $\mathtt{E}$, then $\mathtt{C}$ is equivalent to $\mathtt{E}$.
- Consistency: if $\mathtt{x}$ is an instance of $\mathtt{C}$, and if $\mathtt{C}$ is a subclass of $\mathtt{D} \sqcap \mathtt{E}$, $\mathtt{C}$ is a subclass of $\mathtt{F}$, with $\mathtt{D} \sqcap \mathtt{F} \sqsubseteq \perp$ (i.e. $\mathtt{D}$ and $\mathtt{F}$ are incompatible), then there must exist an inconsistency, and the class $\mathtt{C}$ should be empty.
- Classification: if certain property-value pairs have been declared as sufficient conditions for membership to a class $\mathtt{C}$, then if an individual $\mathtt{x}$ satisfies such conditions, it can be concluded that $\mathtt{x}$ is an instance of $\mathtt{C}$.

Requirements for an ontology language

DL as Ontology languages

- The suitability of Description Logics as ontology languages has been highlighted by their roles as the foundations for several Web ontology languages, including OIL, DAML+OIL, and finally OWL (Web Ontology Language).
- All these languages have a syntax based on RDF Schema, but the basis for their design is the expressive DL $\mathcal{SHIQ}$.
- The DL $\mathcal{SHIQ}$ is decidable, but it has a rather high worst-case complexity (Exptime): nevertheless, highly optimized $\mathcal{SHIQ}$ reasoners such as FaCT and RACER behave very well in practice.

The features of $\mathcal{SHIQ}$

The features of $\mathcal{SHIQ}$

- $\mathcal{SHIQ} = \mathcal{ALC} \cup \mathcal{H} \cup \mathcal{I} \cup \mathcal{Q}$
  $\mathcal{ALC} = \{\top, \bot, \mathtt{C} \sqcap \mathtt{D}, \mathtt{C} \sqcup \mathtt{D}, \neg\mathtt{C}, \forall\mathtt{r.C}, \exists\mathtt{r.C}\}$
- *Qualified number restrictions*:
  $\mathcal{Q}$ for $\{\geq\mathtt{n\,r.C}, \leq\mathtt{n\,r.C}\}$.
  $\geq 1\,\mathtt{hasChild}.\neg\mathtt{Female} \sqcap \geq 1\,\mathtt{hasChild.Female}$
- $\mathcal{SHIQ}$ allows the formulation of complex terminological axioms (with terminological cycles):
  $\mathtt{Human} \sqsubseteq \exists\mathtt{hasParent.Human}$
- $\mathcal{SHIQ}$ allows inverse roles ($\mathcal{I}$), subroles or role hierarchy ($\mathcal{H}$), and transitive roles.
  The role `hasChild` has for inverse `hasParent`, `hasAncestor` is a transitive role, and `hasParent` is a subrole of `hasAncestor`.

The features of $\mathcal{SHIQ}$

- *Concrete domains* (datatypes) integrate DLs with concrete sets such as real numbers, or strings, and built-in predicates such as comparisons $\leq$, $\geq$, `isPrefixOf`,...
- A *general concept inclusion*, or GCI, is of the form $\mathtt{C} \sqsubseteq \mathtt{D}$, where $\mathtt{C}$, $\mathtt{D}$ are $\mathcal{SHIQ}$ concepts.
  A finite set of GCIs is called a Tbox.
- A concept definition is of the form $\mathtt{A} \equiv \mathtt{C}$, where $\mathtt{A}$ is a concept name.
  It can be seen as an abbreviation for the two GCIs $\mathtt{A} \sqsubseteq \mathtt{C}$ and $\mathtt{C} \sqsubseteq \mathtt{A}$.

The features of $\mathcal{SHIQ}$

- Human are either muggle or sorcerer, and a muggle is not sorcerer, and vice versa:
  $\texttt{Human} \sqsubseteq \texttt{Muggle} \sqcup \texttt{Sorcerer}$ and $\texttt{Muggle} \sqsubseteq \neg\texttt{Sorcerer}$
- Humans have exactly two parents, and all parents and children of humans are human:
  $\texttt{Human} \sqsubseteq \forall\texttt{hasParent}.\texttt{Human} \sqcap (\leq 2\, \texttt{hasParent}.\top) \sqcap (\geq 2\, \texttt{hasParent}.\top) \sqcap \forall\texttt{hasParent}^{-}.\texttt{Human}$
- The role $\texttt{hasAncestor}$ is transitive and has a subrole:
  $\texttt{hasParent} \sqsubseteq \texttt{hasAncestor}$
- Humans having an ancestor sorcerer are themselves sorcerers:
  $\texttt{Human} \sqcap \exists\texttt{hasAncestor}.\texttt{Sorcerer} \sqsubseteq \texttt{Sorcerer}$

The features of $\mathcal{SHIQ}$

- $\mathtt{Human} \sqsubseteq \mathtt{Muggle} \sqcup \mathtt{Sorcerer}$ *and* $\mathtt{Muggle} \sqsubseteq \neg\mathtt{Sorcerer}$
- $\mathtt{Human} \sqsubseteq \forall\mathtt{hasParent}.\mathtt{Human} \sqcap (\leq 2\,\mathtt{hasParent}.\top) \sqcap (\geq 2\,\mathtt{hasParent}.\top) \sqcap \forall\mathtt{hasParent}^{-}.\mathtt{Human}$
- $\mathtt{hasParent} \sqsubseteq \mathtt{hasAncestor}$ ($\mathtt{hasAncestor}$ transitive)
- $\mathtt{Human} \sqcap \exists\mathtt{hasAncestor}.\mathtt{Sorcerer} \sqsubseteq \mathtt{Sorcerer}$
- From the above definitions and the CGIs, it can be deduced that: $\mathtt{Grandparent} \sqcap \mathtt{Sorcerer} \sqsubseteq \exists\mathtt{hasParent}^{-}.\exists\mathtt{hasParent}^{-}.\mathtt{Sorcerer}$
  i.e. grandparents that are sorcerers have a grandchild that is a sorcerer.

From $\mathcal{SHIQ}$ to OWL

From $\mathcal{SHIQ}$ to OWL

Constructor	DL syntax	Example
intersectionOf	$C_1 \sqcap C_2$	$\mathtt{Human} \sqcap \mathtt{Male}$
unionOf	$C_1 \sqcup C_2$	$\mathtt{Doctor} \sqcup \mathtt{Professor}$
complementOf	$\neg C$	$\neg\mathtt{Male}$
oneOf	$\{x_1, x_2, \ldots, x_n\}$	$\{\mathtt{Paolo}, \mathtt{Maria}\}$
allValuesFrom	$\forall r.C$	$\forall\mathtt{hasChild}.\mathtt{Male}$
someValuesFrom	$\exists r.C$	$\exists\mathtt{hasChild}.\mathtt{Female}$
hasValue	$\exists r.\{x\}$	$\exists\mathtt{citizenOf}.\{\mathtt{Europe}\}$
minCardinality	$\geq n\, r$	$(\geq 2\, \mathtt{hasChild})$
maxCardinality	$\leq n\, r$	$(\leq 1\, \mathtt{hasChild})$
inverseOf	r^-	$\mathtt{hasChild}^-$

From $\mathcal{SHIQ}$ to OWL

Axiom	DL syntax	Example
subClassOf	$\mathtt{C_1 \sqsubseteq C_2}$	$\mathtt{Man \sqsubseteq Human}$
equivalentClass	$\mathtt{C_1 \equiv C_2}$	$\mathtt{Man \equiv Human \sqcap Ma}$
subPropertyOf	$\mathtt{r_1 \sqsubseteq r_2}$	$\mathtt{hasSon \sqsubseteq hasCh}$
equivalentProperty	$\mathtt{r_1 \equiv r_2}$	$\mathtt{cost \equiv price}$
disjointWith	$\mathtt{C_1 \sqsubseteq \neg C_2}$	$\mathtt{Male \sqsubseteq \neg Femal}$
sameAs	$\mathtt{\{x_1\} \equiv \{x_2\}}$	$\mathtt{\{Parigi\} \equiv \{Par}$
differentFrom	$\mathtt{\{x_1\} \sqsubseteq \neg\{x_2\}}$	$\mathtt{\{Paolo\} \sqsubseteq \neg\{Mar}$
transitiveProperty	$\mathtt{r \in R_+}$	$\mathtt{hasAncestor^+ \in}$
functionalProperty	$\top \sqsubseteq (\leq 1\, \mathtt{r})$	$\top \sqsubseteq (\leq 1\, \mathtt{hasMoth}$
inverseFunctionalProperty	$\top \sqsubseteq (\leq 1\, \mathtt{r^-})$	$\top \sqsubseteq (\leq 1\, \mathtt{isMother}$
symmetricProperty	$\mathtt{r \equiv r^-}$	$\mathtt{isSibingOf \equiv isSibl}$

From $\mathcal{SHIQ}$ to OWL

- OWL Full: OWL language primitives + RDF + RDF Schema, with the possibility of changing the predefined primitives in RDF or OWL, syntactically and semantically compatible with RDF(S), and undecidability...
- OWL DL: a sublanguage of OWL Full based on the DL language $\mathcal{SHOQ}(\mathcal{D})$, with efficient reasoning support, and not full compatibility with RDF(S), i.e. an RDF document is not necessarily a legal OWL DL document, but an OWL DL document is a legal RDF document.
- OWL Lite: a sublanguage of OWL DL based on the DL language $\mathcal{SHIQ}$ (without enumerated classes, disjointness statements, and arbitrary cardinality), easier to use or to implement but restricted expressivity.

A small example

A small example

```
ObjectProperty(hasMember inverseOf(isMemberOf))
ObjectProperty(isMemberOf inverseOf(hasMember))
ObjectProperty(isMarriedTo inverseOf(isMarriedTo)
    domain(Person) range(Person))

Class(Female partial Person
    restriction(isMarriedTo allValuesFrom(Male)))
Class(Male partial Person
    restriction(isMarriedTo allValuesFrom(Female)))
Class(MarriedPerson complete intersectionOf(Person
    restriction(isMarriedTo someValuesFrom(owl:Thing))))
Class(Person partial owl:Thing unionOf(Female Male))
```

A small example

```
Class(MixedTeam complete intersectionOf(Team
    restriction(hasMember someValuesFrom(Male))
    restriction(hasMember someValuesFrom(Female))))
Class(NonSingletonTeam complete intersectionOf(Team
    restriction(hasMember minCardinality(2))))
Class(SingletonTeam complete intersectionOf(Team
    restriction(hasMember cardinality(1))))
Class(Team partial)
Class(owl:Thing partial)

Individual(Chris type(Person) value(isMarriedTo Sam)
value(isMemberOf OntologyMDA))
Individual(OntologyMDA type(Team))
Individual(Sam type(Person) value(isMarriedTo Chris)
value(isMemberOf OntologyMDA))
```

A small example

The ontology in DL syntax:

$$
\begin{array}{l}
\mathtt{Female} \sqsubseteq \mathtt{Person} \sqcap \forall \mathtt{isMarriedTo.Male} \\
\mathtt{Male} \sqsubseteq \mathtt{Person} \sqcap \forall \mathtt{isMarriedTo.Female} \\
\mathtt{MarriedPerson} \equiv \mathtt{Person} \sqcap \exists \mathtt{isMarriedTo}.\top \\
\mathtt{Person} \sqsubseteq \mathtt{Female} \sqcup \mathtt{Male} \\
\mathtt{MixedTeam} \equiv \mathtt{Team} \sqcap \exists \mathtt{hasMember.Male} \sqcap \exists \mathtt{hasMember.Female} \\
\mathtt{NonSingletonTeam} \equiv \mathtt{Team} \sqcap (\geq 2\, \mathtt{hasMember}) \\
\mathtt{SingletonTeam} \equiv \mathtt{Team} \sqcap (\geq 1\, \mathtt{hasMember}) \sqcap (\leq 1\, \mathtt{hasMember}) \\
\mathtt{Team} \sqsubseteq \top \\
\mathtt{Person}(\mathtt{Chris}) \sqcap \mathtt{isMarriedTo}(\mathtt{Chris}, \mathtt{Sam}) \sqcap \\
\mathtt{isMemberOf}(\mathtt{Chris}, \mathtt{OntologyMDA}) \\
\mathtt{Person}(\mathtt{Sam}) \sqcap \mathtt{isMarriedTo}(\mathtt{Sam}, \mathtt{Chris}) \sqcap \\
\mathtt{isMemberOf}(\mathtt{Sam}, \mathtt{OntologyMDA})
\end{array}
$$

A small example

Facts that can be deduced from the ontology:

- `OntologyMDA` is a `MixedTeam`, even though we don't know anything specific about the sex of `Chris` and `Sam`.
- Reasoning on a case by case basis: either `Chris` is `Male`, in which case `Sam` is `Female`, or `Chris` is `Female` and `Sam` is `Male`. In both cases, `OntologyMDA` has both `Male` and `Female` members. However, we still don't know whether `Chris` (or `Sam`) is `Male` or `Female`!

A small example

Facts that can be deduced from the ontology:

- `OntologyMDA` is not a `NonSingletonTeam`: we might expect this to be the case as both `Sam` and `Chris` are members, but it is not.
- By default, OWL makes no assumptions about whether primitive classes are disjoint, and the open world assumtion holds: an unknown fact is not considered as false unless specified.
- A perfectly acceptable interpretation here is that `Sam` and `Chris` are the same person, and thus `OntologyMDA` is only known to have at least one member.

A small example

Facts that can be deduced from the ontology:

- The new statement $\texttt{Female} \sqcap \texttt{Male} \sqsubseteq \bot$ is added.
- The reasoner will be able to determine that the sets of instances of `Male` and `Female` must be distinct.
- Thus any team that has a `Male` member and a `Female` member must have at least 2 members, and thus is a `NonSingletonTeam`, and thus that any `MixedTeam` must be a `NonSingletonTeam`.

Conclusion

Conclusion

- OWL is the proposed standard for Web ontologies. It allows us to describe the semantics of knowledge in a machine-accessible way.
- OWL builds upon RDF and RDF Schema.
- Formal semantics and reasoning support is provided through the mapping of OWL logics (mainly description logics).
- While OWL is sufficiently rich to be used in practice, extensions are in the making: they will provide further logical features, including rules.

Conclusion

- G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. van de Velde, and B. Wielinga. Knowledge Engineering and Management: the CommonKADS Methodoloy, The MIT Press, Cambridge, MA 1999.
- D. Fensel, J. Hendler, H. Lieberman and W. Wahlster editors, Spinning the Semantic Web, The MIT Press, Cambridge, Massachusetts, 2003.
- S. Staab and R. Studer editors, Handbook on Ontologies, Springer, Berlin, 2004.
- M. Stefik, Introduction to Knowledge Systems, Morgan Kaufmann Publishers, San Francisco (CA), 1995.