Multi-agent Semantic Web Systems: Description Logic

Ewan Klein

MASWS — 7 February 2006
1 Background
   - Ontologies Again
   - Limitations of RDFS

2 Anatomy of DL

3 Summary
Shared Meaning

- Ontologies can be used to support communication between agents:
Shared Meaning

- Ontologies can be used to support communication between agents:
- Need to have an agreed vocabulary with shared meaning.
Shared Meaning

- Ontologies can be used to support communication between agents:
- Need to have an agreed vocabulary with shared meaning.

Aspects of meaning

**Denotation**: which individual does myzoo:jerome refer to?
What is meant by the Giraffe?
Ontologies can be used to support communication between agents:

- Need to have an agreed vocabulary with shared meaning.

**Aspects of meaning**

**Denotation**: which individual does `myzoo:jerome` refer to?

What is meant by the *Giraffe*?

**Truth**: `myzoo:jerome rdf:type terms:Giraffe` is true;
Ontologies can be used to support communication between agents:

- Need to have an agreed vocabulary with shared meaning.

**Aspects of meaning**

**Denotation:** which individual does `myzoo:jerome` refer to? What is meant by the **Giraffe**?

**Truth:** `myzoo:jerome rdf:type terms:Giraffe .` is true;

**Inference:** `myzoo:jerome rdf:type terms:Giraffe .` entails

`myzoo:jerome rdf:type terms:Mammal .`
Within Artificial Intelligence, a number of proposals for representing knowledge:
Within Artificial Intelligence, a number of proposals for representing knowledge:

- frames and inheritance
Within Artificial Intelligence, a number of proposals for representing knowledge:
- frames and inheritance
- semantic networks
Within Artificial Intelligence, a number of proposals for representing knowledge:

- frames and inheritance
- semantic networks
- object-oriented languages
Within Artificial Intelligence, a number of proposals for representing knowledge:

- frames and inheritance
- semantic networks
- object-oriented languages

Early approaches to KR lacked explicit treatment of meaning and reasoning.
Evolution of DL

- Within Artificial Intelligence, a number of proposals for representing knowledge:
  - frames and inheritance
  - semantic networks
  - object-oriented languages
- Early approaches to KR lacked explicit treatment of meaning and reasoning.
- Description Logic (DL) was developed as a ‘rational reconstruction’ of these KR systems.
Evolution of DL

- Within Artificial Intelligence, a number of proposals for representing knowledge:
  - frames and inheritance
  - semantic networks
  - object-oriented languages
- Early approaches to KR lacked explicit treatment of meaning and reasoning.
- Description Logic (DL) was developed as a ‘rational reconstruction’ of these KR systems.
- Has been used as the foundation of OWL (Web Ontology Language).
Could mean:
Ambiguity of Diagrams

Could mean:
- giraffes eat only plants;
Ambiguity of Diagrams

Could mean:
- giraffes eat only plants;
- giraffes eat plants and other things too;
Ambiguity of Diagrams

- Could mean:
  - giraffes eat only plants;
  - giraffes eat plants and other things too;
  - there is a specific giraffe which eats only plants (or plants and other things);
Ambiguity of Diagrams

Could mean:

- giraffes eat only plants;
- giraffes eat plants and other things too;
- there is a specific giraffe which eats only plants (or plants and other things);
- giraffes usually eat plants (but some giraffes eat pizza).
Diagrammatic representations potentially very ambiguous.
Explicitness

- Diagrammatic representations potentially very ambiguous.
- Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
Explicitness

- Diagrammatic representations potentially very ambiguous.
- Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
- Set-theoretic approach:
Explicitness

- Diagrammatic representations potentially very ambiguous.
- Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
- Set-theoretic approach:
  - domain of discourse — set of individuals;
Explicitness

- Diagrammatic representations potentially very ambiguous.
- Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
- Set-theoretic approach:
  - domain of discourse — set of individuals;
  - classes are sets of individuals;
Explicitness

- Diagrammatic representations potentially very ambiguous.
- Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
- Set-theoretic approach:
  - domain of discourse — set of individuals;
  - classes are sets of individuals;
  - relations are sets of pairs of individuals.
Explicitness

Diagrammatic representations potentially very ambiguous.
Desirable to have representations which can be grounded in well-understood formal semantic frameworks.
Set-theoretic approach:
- domain of discourse — set of individuals;
- classes are sets of individuals;
- relations are sets of pairs of individuals.
Inference should be justified in terms of the representation.
Sets

Outline
Background
Ontologies
Again
Limitations of RDFS
Anatomy of DL
Summary

Multi-agent Semantic Web Systems: Description Logic
Ewan Klein

Sets

Jerome
Giraffes
Mammals
Lions
Leo

Mammals
Sets
RDFS and Consistency, 1

- RDFS has very weak notion of logic; no way of reasoning with negation.
RDFS and Consistency, 1

- RDFS has very weak notion of logic; no way of reasoning with negation.
- We can’t say that the sets denoted by `terms:Giraffe` and `terms:Lion` are disjoint (i.e., sets with no members in common).
RDFS and Consistency, 1

- RDFS has very weak notion of logic; no way of reasoning with negation.
- We can’t say that the sets denoted by `terms:Giraffe` and `terms:Lion` are disjoint (i.e., sets with no members in common).
- So nothing prevents us from having an RDF store with both of the following triples:
RDFS and Consistency, 1

- RDFS has very weak notion of logic; no way of reasoning with negation.
- We can’t say that the sets denoted by `terms:Giraffe` and `terms:Lion` are disjoint (i.e., sets with no members in common).
- So nothing prevents us from having an RDF store with both of the following triples:

```xml
Disjoint Sets

myzoo:jerome rdf:type terms:Giraffe .
myzoo:jerome rdf:type terms:Lion .
```
We want to say that Mammals eat EdibleThings, and a subclass of Mammals, namely Carnivores, eat Animals.
- We want to say that Mammals eat EdibleThings, and a subclass of Mammals, namely Carnivores, eat Animals.
- In other words,
- We want to say that Mammals eat EdibleThings, and a subclass of Mammals, namely Carnivores, eat Animals.
- In other words,
  - if the domain of terms:eats is terms:Mammal, then its range is terms:EdibleThing;

```
Mammal
  hasLegs                  4
  eats      EdibleThings

Carnivore
  eats      Animal
```
We want to say that Mammals eat EdibleThings, and a subclass of Mammals, namely Carnivores, eat Animals.

In other words,
- if the domain of `terms:eats` is `terms:Mammal`, then its range is `terms:EdibleThing`;
- if the domain of `terms:eats` is `terms:Carnivore`, then its range is `terms:Animal`;

```
Mammal
  hasLegs 4
  eats EdibleThings

isa

Carnivore
  eats Animal
```
We want to say that Mammals eat EdibleThings, and a subclass of Mammals, namely Carnivores, eat Animals.

In other words,
- if the domain of \texttt{terms:eats} is \texttt{terms:Mammal}, then its range is \texttt{terms:EdibleThing};
- if the domain of \texttt{terms:eats} is \texttt{terms:Carnivore}, then its range is \texttt{terms:Animal};

However, RDF(S) could not rule out a set of triples where Leo is a carnivore, but eats some non-Animal thing such as grass.

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Mammal} & \texttt{isa} \\
\hline
\texttt{hasLegs} & 4 \\
\texttt{eats} & \texttt{EdibleThings} \\
\hline
\end{tabular}
\end{center}
All classes in RDFS are primitive; all we can say is that one class is included in another one.
All classes in RDFS are primitive; all we can say is that one class is included in another one.

But sometimes we’d like to define classes; e.g.,
Class Definitions

- All classes in RDFS are primitive; all we can say is that one class is included in another one.
- But sometimes we’d like to define classes; e.g.,
  - **Sibling** is the union of **Brother** and **Sister**;
All classes in RDFS are primitive; all we can say is that one class is included in another one.

But sometimes we’d like to define classes; e.g.,

- **Sibling** is the union of **Brother** and **Sister**;
- **Batchelor** is the intersection of **Male**, **Adult** and **Unmarried**;
Class Definitions

- All classes in RDFS are primitive; all we can say is that one class is included in another one.
- But sometimes we'd like to define classes; e.g.,
  - **Sibling** is the union of **Brother** and **Sister**;
  - **Batchelor** is the intersection of **Male**, **Adult** and **Unmarried**;
  - **Unmarried** is the complement of **Married**,
Description Logics are actually a family of languages;
Description Logics are actually a family of languages; they differ in their expressiveness.
Description Logics are actually a family of languages;
- differ in their expressiveness.
- DLs are restricted fragments of First Order Logic.
Description Logics are actually a family of languages; differ in their expressiveness.

DLs are restricted fragments of First Order Logic.

A set of DL sentences is divided into two parts:
Description Logics are actually a family of languages; they differ in their expressiveness.

DLs are restricted fragments of First Order Logic.

A set of DL sentences is divided into two parts:

**TBox**: Terminological statements
Description Logics are actually a family of languages; differ in their expressiveness.

DLs are restricted fragments of First Order Logic.

A set of DL sentences is divided into two parts:

- **TBox**: Terminological statements
- **ABox**: Assertions

Example Assertions

- jerome: Giraffe (Jerome is a Giraffe)
- ⟨leo, jerome⟩: eats (Leo eats Jerome)
Description Logics are actually a family of languages;
- differ in their expressiveness.
- DLs are restricted fragments of First Order Logic.
- A set of DL sentences is divided into two parts:
  - TBox: Terminological statements
  - ABox: Assertions

Example Assertions

<table>
<thead>
<tr>
<th>jerome: Giraffe</th>
<th>(Jerome is a Giraffe)</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨leo, jerome⟩: eats</td>
<td>(Leo eats Jerome)</td>
</tr>
</tbody>
</table>

- Giraffe could be any DL concept (class expression).
Description Logics are actually a family of languages; differ in their expressiveness.

DLs are restricted fragments of First Order Logic.

A set of DL sentences is divided into two parts:

- **TBox**: Terminological statements
- **ABox**: Assertions

Example Assertions

- jerome: Giraffe (Jerome is a Giraffe)
- ⟨leo, jerome⟩: eats (Leo eats Jerome)

- Giraffe could be any DL concept (class expression).
- All of the TBox is ways of specifying concepts.
**DL Concepts**

- **Concepts** — denote sets of instances.
Concepts — denote sets of instances.

Simple Concepts

Giraffe \{x \mid \text{Giraffe}(x)\}
DL Concepts

- **Concepts** — denote sets of instances.

### Simple Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giraffe</td>
<td>{x</td>
</tr>
</tbody>
</table>

### Composed Concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brother \sqcup Sister</td>
<td>{x</td>
</tr>
<tr>
<td>Adult \sqcap Male</td>
<td>{x</td>
</tr>
<tr>
<td>\neg Married</td>
<td>{x</td>
</tr>
</tbody>
</table>
### DL Concepts

- **Concepts** — denote sets of instances.

<table>
<thead>
<tr>
<th>Simple Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giraffe</td>
</tr>
<tr>
<td>{x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Composed Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brother ⊔ Sister</td>
</tr>
<tr>
<td>{x</td>
</tr>
<tr>
<td>Adult ⊓ Male</td>
</tr>
<tr>
<td>{x</td>
</tr>
<tr>
<td>¬ Married</td>
</tr>
<tr>
<td>{x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Giraffe ⊑ Mammal</td>
</tr>
<tr>
<td>∀x(Giraffe(x) ⇒ Mammal(x))</td>
</tr>
</tbody>
</table>
## DL Concepts

**Concepts** — denote sets of instances.

### Simple Concepts

- **Giraffe**
  \[ \{ x \mid \text{Giraffe}(x) \} \]

### Composed Concepts

- **Brother ⋈ Sister**
  \[ \{ x \mid \text{Brother}(x) \lor \text{Sister}(x) \} \]
- **Adult ⋈ Male**
  \[ \{ x \mid \text{Adult}(x) \land \text{Male}(x) \} \]
- **¬ Married**
  \[ \{ x \mid \neg \text{Married}(x) \} \]

### Subsumption

- **Giraffe ⊑ Mammal**
  \[ \forall x (\text{Giraffe}(x) \Rightarrow \text{Mammal}(x)) \]

### Definitional Equivalence

- **Sibling ≡ Brother ⋈ Sister**
  \[ \forall x (\text{Sibling}(x) \Leftrightarrow \text{Brother}(x) \lor \text{Sister}(x)) \]
Role Expressions

- **Roles** — denote sets of pairs.

**Simple Role**

\[
eats \{ \langle x, y \rangle \mid \text{eats}(x, y) \}\]
Role Expressions

- **Roles** — denote sets of pairs.

### Simple Role

- **eats** \( \{(x, y) \mid \text{eats}(x, y)\} \)

### Restricted Quantification

- \( \forall \text{eats.}\text{Plant} \quad \{x \mid \forall y (\text{eats}(x, y) \Rightarrow \text{Plant}(y))\} \)
- \( \exists \text{eats.}\text{Plant} \quad \{x \mid \exists y (\text{eats}(x, y) \land \text{Plant}(y))\} \)
Role Expressions

- **Roles** — denote sets of pairs.

### Simple Role

\[
eats \{ \langle x, y \rangle | \text{eats}(x, y) \}
\]

### Restricted Quantification

\[
\forall \text{eats.Plant} \{ x | \forall y (\text{eats}(x, y) \Rightarrow \text{Plant}(y)) \}
\]

\[
\exists \text{eats.Plant} \{ x | \exists y (\text{eats}(x, y) \land \text{Plant}(y)) \}
\]

### Manchester OWL Syntax

\[
\forall \text{eats.Plant} \text{ eats only Plant}
\]

\[
\exists \text{eats.Plant} \text{ eats some Plant}
\]

http://www.co-ode.org/resources/reference/manchester_syntax/
A Complex DL Example

Concept of ‘Happy Man’

\[
\text{Man} \sqcap (\exists \text{hasChild}. \text{Boy}) \sqcap \\
(\exists \text{hasChild}. \text{Girl}) \sqcap \\
(\forall \text{hasChild}. \text{Happy} \sqcap \text{Rich})
\]
DL vs. RDF(S)

Disjointness

Giraffe $\sqcap$ Lion $\sqsubseteq \bot$

Restriction of Predicates

Carnivore $\models \forall$ eats.Meat
Inference

Subsumption

\[ C \sqsubseteq D \quad \text{Is } C^I \subseteq D^I \text{ for all interpretations } I? \]

Consistency

\[ \text{Is } C \text{ consistent wrt TBox } I? \quad \text{Is there an interpretation of } I \text{ where } C^I \neq \emptyset \]

Connected Inference Problems

\[ C \sqsubseteq D \text{ iff } C \cap \neg D \text{ is inconsistent wrt to } I \]

- **Classification**: construct a subsumption hierarchy of concepts, given their definitions in a TBox.
- Tableau-based algorithms for classification give good performance.
  - FaCT and RACER both widely used. (RACER can be added as a plug-in to Protégé.)
Next Two Lectures

Friday 10th Feb: Stephen Potter on OWL.
Next Two Lectures

**Friday 10th Feb**: Stephen Potter on OWL.

**Tuesday 14th Feb**: Henry Thompson (W3C Fellow) on Web Standards.