Semantic Web

OWL

Acknowledgements to Pascal Hitzler, York Sure

OWL – General

- W3C Recommendation since 2004
- Semantic fragment of FOL
- Three variants: OWL Lite \( \subseteq \) OWL DL \( \subseteq \) OWL Full
- No reification in OWL DL
- RDFS is fragment of OWL Full
- OWL DL is decidable
- OWL DL = SHOIN(D) (description logics)
- W3C-Documents contain many more details that we cannot talk about here

Content

OWL – Syntax and model theoretic semantics
  a. Description logics: SHOIN(D)
  b. OWL as SHOIN(D)
  c. Serializations
  d. Knowledge modelling in OWL

Description Logics (Terminological Logics, DLs)

- Fragments of FOL
- Most often decidable
- Moderately expressive
- Stem from semantic networks
- Close relationship to propositional modal logics
- W3C Standard OWL DL corresponds to SHOIN(D)
General DL Architecture

Knowledge Base

- Tbox (schema)
  - Man = Human \land Male
  - Happy-Father = Man \land \exists has-child.Female \land ...

- Abox (data)
  - Happy-Father(John)
  - has-child(John, Mary)

Inference System

Interface

DLs – general structure

- DLs are a Family of logic-based formalism for knowledge representation
- Special language characterized by:
  - Constructors to define complex concepts and roles based on simpler ones.
  - Set of axiom to express facts using concepts, roles and individuals.
- ALC is the smallest DL, which is propositionally closed:
  - \land, \lor, \neg are constructors, noted by \land, \lor, \neg.
  - Quantors define how roles are to be interpreted:
    - Man \land \exists hasChild.Female \land \exists hasChild.Male
    - \forall hasChild.(Rich \lor Happy)

Further DL concepts and role constructors

- Number restrictions (cardinality constraints) for roles:
  - \geq 3 hasChild, \leq 1 hasMother
- Qualified number restrictions:
  - \geq 2 hasChild.Female, \leq 1 hasParent.Male
- Nominals (definition by extension):
  - \{Italy, France, Spain\}
- Concrete domains (datatypes):
  - hasAge.(\geq 21)
- Inverse roles:
  - hasChild' \equiv hasParent
- Transitive roles:
  - hasAncestor* (descendant)
- Role composition:
  - hasParent.hasBrother (uncle)

DL Knowledge Bases

- DL Knowledge Bases consist of two parts (in general):
  - TBox: Axioms, describing the structure of a modelled domain (conceptual schema):
    - HappyFather = Man \land \exists hasChild.Female \land ...
    - Elephant \subseteq Animal \land Large \land Grey
    - transitive(hasAncestor)
  - Abox: Axiome describing concrete situations (data, facts):
    - HappyFather(John)
    - hasChild(John, Mary)
- The distinction between TBox/ABox does not have a deep logical distinction ...
  but it is common useful modelling practice.
Syntax for DLs (ohne concrete domains)

<table>
<thead>
<tr>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic: A, B</td>
</tr>
<tr>
<td>Not: ¬C</td>
</tr>
<tr>
<td>And: C ∧ D</td>
</tr>
<tr>
<td>Or: C ∨ D</td>
</tr>
<tr>
<td>Exists: ∃R.C</td>
</tr>
<tr>
<td>For all: ∀R.C</td>
</tr>
<tr>
<td>At least: ≥n R.C (≥n R)</td>
</tr>
<tr>
<td>At most: ≤n R.C (≤n R)</td>
</tr>
<tr>
<td>Nominal: {i_1, ..., i_n}</td>
</tr>
</tbody>
</table>

Ontology (=Knowledge Base)

Concept Axioms (TBox)
- Subclass: C ⊒ D
- Equivalent: C ≡ D

Role Axioms (RBox)
- Subrole: R ⊒ S
- Transitivity: Trans(S)

Assertional Axioms (ABox)
- Instance: C(a)
- Role: R(a, b)
- Same: a = b
- Different: a ≠ b

S = ALC + Transitivity

OWL DL = SHOIN(D) (D: concrete domain)

Content

OWL – Syntax and model theoretic semantics
a. Description logics: SHOIN(D)
b. OWL as SHOIN(D)
c. Serializations
d. Knowledge modelling in OWL

OWL DL as DL: Class constructors

Constructor | DL Syntax | Example | FOL Syntax |
-------------|-----------|---------|-----------|
intersectionOf | C_1 ∩ ... ∩ C_n | Human ∩ Male | C_1(x) ∧ ... ∧ C_n(x) |
unionOf | C_1 ∪ ... ∪ C_n | Doctor ∪ Lawyer | C_1(x) ∨ ... ∨ C_n(x) |
complementOf | ¬C | ¬Male | ¬C(x) |
oneOf | \{x_1 \cup ... \cup x_n\} | \{John \cup \{Mary\}} | x = x_1 \lor ... \lor x = x_n |
\forall\text{valuesFrom} | \exists R.C | \text{hasChild.Lawyer} | \exists R.P(x, y) \rightarrow C(y) |
\exists\text{valuesFrom} | \exists R.C | \text{hasChild.Doctor} | \exists R.P(x, y) \land C(y) |
minCardinality | ≤n R.C | ≤n \text{hasChild} | ≤n R.P(x, y) |
maxCardinality | ≥n R.C | ≥n \text{hasChild} | ≥n R.P(x, y) |

Nesting of expression is allowed at arbitrary depth:

Person ⊒ ∀hasChild.(Doctor ∪ ∃hasChild.Doctor)

OWL DL as DL: Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subclassOf</td>
<td>C_1 ⊒ C_2</td>
<td>Human ⊒ Animal ⊒ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>C_1 ≡ C_2</td>
<td>Man ≡ Human ∩ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>C_1 ⊖ C_2</td>
<td>Male ⊖ Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>{x_1} ≡ {x_2}</td>
<td>{Bush} ≡ {GW Bush}</td>
</tr>
<tr>
<td>differentFrom</td>
<td>{x_1} \neq {x_2}</td>
<td>{John} \neq {Peter}</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>P_1 ⊑ P_2</td>
<td>hasDaughter ⊑ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>P_1 ≡ P_2</td>
<td>cost ≡ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>P_1 \equiv P_2</td>
<td>hasChild ≡ hasParent</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>T ⊒ T</td>
<td>ancestor \rightarrow ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>T ⊒ T \quad</td>
<td>T \subseteq hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>T \subseteq T^-</td>
<td>T \subseteq hasSon</td>
</tr>
</tbody>
</table>

- General Class Inclusion (≡): C ≡ D  gdw (C ⊒ D und D ⊒ C)
- Obvious equivalences with FOL:
  C ⊒ D ⇔ (∀x) (C(x) ⇔ D(x))
  C ⊒ D ⇔ (∃x) (C(x) ⇔ D(x))
OWL/DL Example

Terminological Knowledge (TBox):
- Human ⊆ ∃parentOf.Human
- Orphan ≡ Human ∩ ¬∃childOf.Alive

Knowledge about Individuals (ABox):
- Orphan(harrypotter)
- ParentOf(jamespotter,harrypotter)

Semantics and logical consequences may be derived by translation to FOL

Model theoretical Semantics – direct

Concept expressions

| A | Subset of Δ₁ |
| ¬C | Δ₁ \ C₂ |
| C ∪ D | {x| x ∈ C₁ and x ∈ D₁} |
| C ∩ D | {x| x ∈ C₁ and x ∈ D₁} |
| ∃R.C | {x| (x,y) ∈ R² and y ∈ C₁} |
| ∀R.C | {x| if (x,y) ∈ R² then y ∈ C₁} |
| ≥ n R.C | {x| #(x,y) ∈ R² and y ∈ C₁} ≥ n |
| ≤ n R.C | {x| #(x,y) ∈ R² and y ∈ C₁} ≤ n |
| {i₁,...,iₙ} | (i₁,...,iₙ) |

Role expressions

| R | Subset of Δ x Δ |
| R⁻¹ | {(y,x) | (x,y) ∈ R²} |

Concrete Domains

- Strings and Integers (required by W3C OWL rec)
- Further datatypes may be supported.
- Restricted to decidable predicates over the concrete domain
- Each concrete domain must be implemented separately and then included into the reasoner (weak analogy: built-ins – but no procedural semantics!)

OWL Lite

- Simple fragment, comparatively low complexity
- Some constructors may not be used:
  - owl:unionOf
  - owl:complementOf
  - owl:oneOf
  - owl:hasValue
  - owl:disjointWith
- The applicability of some operators is restricted:
  - owl:intersectionOf
  - owl:minCardinality
  - owl:maxCardinality
  - owl:cardinality
OWL Full

- equals OWL DL union RDFS
- RDF is contained in OWL Full, not in OWL DL

Intuition:
- OWL Full allows reification.
- OWL Full is no “nice” fragment of FOL
- OWL Full ist not decidable

Reification – Example

Sometimes one might state a proposition about a class name:

Class: father.
(concrete) Rolls: germanClassName(father, “Vater“)
frenchClassName(father, „père“)
englishClassName(father, „father“)

→ Cannot be expressed in OWL DL!

Complexity (worst-case)

<table>
<thead>
<tr>
<th>OWL variant</th>
<th>Datenkomplexität</th>
<th>Combined complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL Full</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
<tr>
<td>OWL DL</td>
<td>not known</td>
<td>NExptime</td>
</tr>
<tr>
<td>OWL DL without nominals</td>
<td>NP (neues Resultat IJCAI 2005!)</td>
<td>Exptime</td>
</tr>
<tr>
<td>OWL Lite</td>
<td>NP</td>
<td>Exptime</td>
</tr>
</tbody>
</table>

Data complexity: assume fixed T-Box, complexit wrt size of ABox
Combined complexity: wrt combined size of ABox and TBox

Content

OWL – Syntax and model theoretic semantics

a. Description logics: SHOIN(D)
b. OWL as SHOIN(D)
c. Serializations
d. Knowledge modelling in OWL
Serializations/different Syntaxes

- OWL RDF Syntax: W3C recommendation
- OWL Abstract Syntax: W3C recommendation; See next section
- OWL XML Syntax: W3C document
- DL Schreibweise: widely used in scientific contexts
- FOL Schreibweise: uncommon

For the implementation and the testing of KAON2 a functional syntax has been developed. It is Lisp-like.

Example: RDF Syntax

Person $\forall$ hasChild. (Doctor $\exists$ hasChild. Doctor):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Content

OWL – Syntax and model theoretic semantics

- Description logics: SHOIN(D)
- OWL as SHOIN(D)
- Serializations
- Knowledge modelling in OWL

Knowledge modelling in OWL

Example ontology and conclusion from http://owl.man.ac.uk/2003/why/latest/#2

- Also an example for OWL Abstract Syntax.

```xml
Namespace(a = <http://cohse.semanticweb.org/ontologies/people#>)

Ontology(
  ObjectProperty(a:drives)
  ObjectProperty(a:eaten_by)
  ObjectProperty(a:eats inverseOf(a:eaten_by) domain(a:animal))
  ...)

Class(a:adult partial annotation(rdfs:comment "Things that are adult.")
Class(a:animal partial restriction(a:eats someValuesFrom (owl:Thing)))
Class(a:animal_lover complete intersectionOf(restriction(a:has_pet
minCardinality(3)) a:person))
...)
```
Knowledge modelling: examples

Class(a:bus_driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:bus))))

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle))))

Class(a:bus partial a:vehicle)

• A bus driver is a person that drives a bus.
• A bus is a vehicle.
• A bus driver drives a vehicle, so must be a driver.

The subclass is inferred due to subclasses being used in existential quantification.

Knowledge modelling: examples

Class(a:driver complete intersectionOf(a:person restriction(a:drives someValuesFrom (a:vehicle))))

Class(a:driver partial a:adult)

Class(a:grownup complete intersectionOf(a:adult a:person))

• Drivers are defined as persons that drive cars (complete definition)
• We also know that drivers are adults (partial definition)
• So all drivers must be adult persons (e.g. grownups)

An example of axioms being used to assert additional necessary information about a class. We do not need to know that a driver is an adult in order to recognize one, but once we have recognized a driver, we know that they must be adult.

Knowledge modelling: Examples

Class(a:cow partial a:vegetarian)

DisjointClasses(unionOf(restriction(a:part_of someValuesFrom (a:animal)) a:animal) unionOf(a:plant restriction(a:part_of someValuesFrom (a:plant))))

Class(a:vegetarian complete intersectionOf( restriction(a:eats allValuesFrom (complementOf(restriction(a:part_of someValuesFrom (a:animal))) restriction(a:eats allValuesFrom (complementOf(a:animal))) a:animal)))

Class(a:mad_cow complete intersectionOf(a:cow restriction(a:eats someValuesFrom (intersectionOf(restriction(a:part_of someValuesFrom (a:sheep)) a:brain)))))

Class(a:sheep partial a:animal restriction(a:eats allValuesFrom (a:grass))

• Cows are naturally vegetarians
• A mad cow is one that has been eating sheeps brains
• Sheep are animals

Thus a mad cow has been eating part of an animal, which is inconsistent with the definition of a vegetarian

Knowledge modelling: Example

Individual(a:Walt type(a:person) value(a:has_pet a:Huey) value(a:has_pet a:Louie) value(a:has_pet a:Dewey))

Individual(a:Huey type(a:duck))

Individual(a:Dewey type(a:duck))

Individual(a:Louie type(a:duck))

DifferentIndividuals(a:Huey a:Dewey a:Louie)

Class(a:animal_lover complete intersectionOf(a:person restriction(a:has_pet minCardinality(3)))))

ObjectProperty(a:has_pet domain(a:person) range(a:animal))

• Walt has pets Huey, Dewey and Louie.
• Huey, Dewey and Louie are all distinct individuals.
• Walt has at least three pets and is thus an animal lover.

Note that in this case, we don’t actually need to include person in the definition of animal lover (as the domain restriction will allow us to draw this inference).
Knowledge modelling: OWA vs. CWA

OWA: Open World Assumption
The existence of further individuals is possible if it is not explicitly excluded.

OWL uses OWA!

CWA: Closed World Assumption
One assumes that the knowledge base contains all known individuals and all known facts.

No idea, since we do not know all children of Bill.

If we assume that we know everything about Bill, then all of his children are male.

Know everything about Bill's children.

Knowledge modelling: Some Research Challenges

- Concluding with
  - uncertainty (fuzzy, probabilistic)
  - Inconsistencies (paraconsistent)
  - Rules
  - Further AI-Paradigms (nonmonotonic reasoning, preferences …)
- Maintenance (updates, infrastructure, etc)
- Scalability of reasoning
- …

Knowledge modelling: Domain and Range

- ObjectProperty(xyz:has_topping
domain(xyz:Pizza)
range(xyz:Pizza_topping))

- Class(xyz:Ice_cream_cone partial
  restriction(xyz:has_topping someValuesFrom (xyz:Ice_cream)))

- If Ice_cream_cone and Pizza not disjunct:
  - Ice_cream_cone is classified as Pizza
  - …but: Ice_cream is not classified as Pizza_topping
  - Consequences: all Ice_cream_cones are Pizzas, and some Ice_cream is a Pizza_topping