Knowledge Representation for the Semantic Web

Winter Quarter 2012

Slides 5 – 01/24/2012

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
Textbook (required)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)

http://www.semantic-web-book.org
Today: Description Logics
Today’s Session: DLs (towards OWL)

1. Basic Ideas
2. The Description Logic SROIQ(D)
3. Different Perspectives
4. Class Project
5. Class Presentations
OWL Building Blocks

• individuals (written as URIs)
  – also: constants (FOL), resources (RDF), instances
  – http://example.org/sebastianRudolph
  – we write these lowercase and abbreviated, e.g. "sebastianRudolph"

• classes (also written as URIs!)
  – also: concepts, unary predicates (FOL)
  – we write these uppercase, e.g. "Father"

• properties (also written as URIs!)
  – also: roles (DL), binary predicates (FOL)
  – we write these lowercase, e.g. "hasDaughter"
**DL syntax**                    **RDFS syntax**

- \texttt{Person(mary)}
  - \texttt{:mary rdf:type :Person .}

- \texttt{Woman \sqsubseteq \text{Person}}
  - \texttt{:Woman rdfs:subClassOf :Person .}

- \texttt{\text{Person} \equiv \text{HumanBeing} \text{ (class equivalence):}}
  - \texttt{\text{Person} \sqsubseteq \text{HumanBeing} \text{ AND}}
  - \texttt{\text{HumanBeing} \sqsubseteq \text{Person}}

- \texttt{hasWife(john,mary)}
  - \texttt{:john :hasWife :mary .}

- \texttt{hasWife \sqsubseteq \text{hasSpouse}}
  - \texttt{:hasWife rdfs:subPropertyOf :hasSpouse}
  - \texttt{\text{hasSpouse} \equiv \text{marriedWith} \text{ (property equivalence)}}
DL syntax

- Person(mary)
- Woman ⊑ Person
  - Person ≡ HumanBeing (class equivalence)
- hasWife(john,mary)
- hasWife ⊑ hasSpouse
  - hasSpouse ≡ marriedWith (property equivalence)

FOL syntax

- Person(mary)
- ∀x (Woman(x) → Person(x))
- Person ≡ HumanBeing (class equivalence)
- hasWife(john,mary)
- ∀x ∀y (hasWife(x,y) → hasSpouse(x,y))
- TBox statements
- ABox statements
Special classes and properties

- **owl:Thing** (RDF syntax)
  - DL-syntax: \( \top \)
  - contains everything
- **owl:Nothing** (RDF syntax)
  - DL-syntax: \( \bot \)
  - empty class
- **owl:topProperty** (RDF syntax)
  - DL-syntax: \( U \)
  - every pair is in \( U \)
- **owl:bottomProperty** (RDF syntax)
  - empty property
Class constructors

- **conjunction**
  - $\text{Mother} \equiv \text{Woman} \cap \text{Parent}$
  - “Mothers are exactly those who are women and parents.“

- **disjunction**
  - $\text{Parent} \equiv \text{Mother} \cup \text{Father}$
  - “Parents are exactly those who are mothers or fathers.“

- **negation**
  - $\text{ChildlessPerson} \equiv \text{Person} \cap \neg \text{Parent}$
  - “ChildlessPersons are exactly those who are persons and who are not parents.“

\[
\forall x \ (\text{Mother}(x) \leftrightarrow \text{Woman}(x) \land \text{Parent}(x))
\]

\[
\forall x \ (\text{Parent}(x) \leftrightarrow \text{Mother}(x) \lor \text{Father}(x))
\]

\[
\forall x \ (\text{ChildlessPerson}(x) \leftrightarrow \text{Person}(x) \land \neg \text{Parent}(x))
\]
Class constructors

- **existential quantification**
  - only to be used with a role – also called a *property restriction*
  - \( \text{Parent} \equiv \exists \text{hasChild}.\text{Person} \)
    
    “Parents are exactly those who have at least one child which is a Person.“

- **universal quantification**
  - only to be used with a role – also called a *property restriction*
  - \( \text{Person} \sqcap \text{Happy} \equiv \forall \text{hasChild}.\text{Happy} \)
    
    “A (person which is also happy) is exactly (something all children of which are happy).“

- **Class constructors can be nested arbitrarily**
Today’s Session: DLs (towards OWL)

1. Basic Ideas
2. The Description Logic SROIQ(D)
3. Different Perspectives
4. Class Project
5. Class Presentations
The description logic ALC

- **ABox expressions**
  - Individual assignments: `Father(john)`
  - Property assignments: `hasWife(john, mary)`

- **TBox expressions**
  - Subclass relationships: $\sqsubseteq$
  - Equivalence: $\equiv$
  - Conjunction: $\sqcap$
  - Disjunction: $\sqcup$
  - Negation: $\neg$
  - Property restrictions: $\forall$, $\exists$

Complexity: ExpTime

Also: $\top$, $\bot$
The Description Logic ALC

- Set of *individuals* a,b,c,...
- Set of *atomic classes* (*class names*) A,B,...
- Set of *role names* R,S,...

- *(Complex) class expressions* are constructed as:

\[ C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \cap D \mid C \cup D \mid \forall R.C \mid \exists R.C \]

- A *TBox* is a set of statements of the form \( C \equiv D \) or \( C \subseteq D \), where C and D are class expressions. They are called *general inclusion axioms*.

- An *ABox* consists of statements of the form \( C(a) \) or \( R(a,b) \), where C is a class expression, R is a role, and a, b are individuals.
The Description Logic ALC

Human ⊑ ∃ hasParent.Human
Orphan ⊑ Human ∩ ∀ hasParent.¬Alive
Orphan(harrypotter)
hasParent(harrypotter, jamespotter)
Understanding SROIQ(D)

ALC + role chains = SR

- hasParent \circ hasBrother \subseteq hasUncle.

  \forall x \forall y (\exists z ((\text{hasParent}(x,z) \land \text{hasBrother}(z,y)) \rightarrow \text{hasUncle}(x,y)))

  - includes top property and bottom property

- includes S = ALC + transitivity
  - hasAncestor \circ hasAncestor \subseteq hasAncestor

- includes SH = S + role hierarchies
  - hasFather \subseteq hasParent
Understanding SROIQ(D)

- O – nominals (closed classes)
  - MyBirthdayGuests ≡ \{bill, john, mary\}
  - Note the difference to
    MyBirthdayGuests(bill)
    MyBirthdayGuests(john)
    MyBirthdayGuests(mary)

- Individual equality and inequality (no unique name assumption!)
  - bill = john
    - \{bill\} ≡ \{john\}
  - bill ≠ john
    - \{bill\} \cap \{john\} ≡ \bot
Understanding SROIQ(D)

- **I** – inverse roles
  - hasParent \equiv \text{hasChild}^-
  - Orphan \equiv \forall \text{hasChild}^- . \text{Dead}

- **Q** – qualified cardinality restrictions
  - \leq 4 \text{hasChild}.\text{Parent}(\text{john})
  - HappyFather \equiv \geq 2 \text{hasChild}.\text{Female}
  - Car \sqsubseteq =4\text{hasTyre}.\top

- **Complexity** SHIQ, SHOQ, SHIO: ExpTime.
  Complexity SHOIQ: NExpTime
  Complexity SROIQ: N2ExpTime
Properties can be declared to be

- **Transitive** hasAncestor \( R(a,b) \) and \( R(b,c) \) → \( R(a,c) \)
- **Symmetric** hasSpouse \( R(a,b) \) → \( R(b,a) \)
- **Asymmetric** hasChild \( R(a,b) \) → not \( R(b,a) \)
- **Reflexive** hasRelative \( R(a,a) \) for all \( a \)
- **Irreflexive** parentOf not \( R(a,a) \) for any \( a \)
- **Functional** hasHusband \( R(a,b) \) and \( R(a,c) \) → \( b=c \)
- **InverseFunctional** hasHusband \( R(a,b) \) and \( R(c,b) \) → \( a=c \)

called *property characteristics*
Understanding SROIQ(D)

(D) – datatypes

• so far, we have only seen properties with individuals in second argument, called *object properties* or *abstract roles* (DL)

• properties with datatype literals in second argument are called *data properties* or *concrete roles* (DL)

• In OWL allowed are many XML Schema datatypes, including `xsd:integer`, `xsd:string`, `xsd:float`, `xsd:boolean`, `xsd:anyURI`, `xsd:dateTime`

  and also e.g. `owl:real`
Understanding SROIQ(D)

(D) – datatypes

• hasAge(john, "51"^^xsd:integer)

• additional use of constraining facets (from XML Schema)
  – e.g. Teenager ≡ Person ⊓ ∃hasAge.(xsd:integer: ≥12 and ≤19)

note: this is not standard DL notation! It‘s really only used in OWL.
Understanding SROIQ(D)

further expressive features

• Self
  – PersonComittingSuicide ⇔ ∃kills.Self
• Keys (not really in SROIQ(D), but in OWL)
  – set of (object or data) properties whose values uniquely identify an object
• disjoint properties
  – Disjoint(hasParent,hasChild)
• explicit anonymous individuals
  – as in RDF: can be used instead of named individuals
SROIQ(D) constructors – overview

- ABox assignments of individuals to classes or properties
- ALC: $\subseteq, \equiv$ for classes
  $\cap, \cup, \neg, \exists, \forall$
  $\top, \bot$
- SR: + property chains, property characteristics, role hierarchies $\subseteq$
- SRO: + nominals \{o\}
- SROI: + inverse properties
- SROIQ: + qualified cardinality constraints
- SROIQ(D): + datatypes (including facets)

- + top and bottom roles (for objects and datatypes)
- + disjoint properties
- + Self
- + Keys (not in SROIQ(D), but in OWL)
Some Syntactic Sugar in OWL

SROIQ(D) is essentially (semantically) the same as OWL.

Available in OWL (see later) as syntactic sugar for DL axioms:

- disjoint classes
  - $\text{Apple} \cap \text{Pear} \subseteq \bot$

- disjoint union
  - $\text{Parent} \equiv \text{Mother} \cup \text{Father}$
    - $\text{Mother} \cap \text{Father} \subseteq \bot$

- negative property assignments (also for datatypes)
  - $\neg \text{hasAge(jack,"53"^^xsd:integer)}$
Two Global Restrictions

- arbitrary property chain axioms lead to undecidability
- restriction: set of property chain axioms has to be regular
  - there must be a strict linear order $\prec$ on the properties
  - every property chain axiom has to have one of the following forms:
    \[
    \begin{align*}
    R \circ R & \subseteq R & S & \subseteq R & S_1 \circ S_2 \circ \ldots \circ S_n & \subseteq R \\
    R \circ S_1 \circ S_2 \circ \ldots \circ S_n & \subseteq R & S_1 \circ S_2 \circ \ldots \circ S_n \circ R & \subseteq R \\
    
    \end{align*}
    \]
  - thereby, $S_i \prec R$ for all $i = 1, 2, \ldots, n$.

- Example 1: $R \circ S \subseteq R \quad S \circ S \subseteq S \quad R \circ S \circ R \subseteq T$
  $\Rightarrow$ regular with order $S \prec R \prec T$
- Example 2: $R \circ T \circ S \subseteq T$
  $\Rightarrow$ not regular because form not admissible
- Example 3: $R \circ S \subseteq S \quad S \circ R \subseteq R$
  $\Rightarrow$ not regular because no adequate order exists
Two Global Restrictions

- combining property chain axioms and cardinality constraints may lead to undecidability

- **restriction**: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)

- technically:
  - for any property chain axiom $S_1 \circ S_2 \circ \ldots \circ S_n \subseteq R$ with $n > 1$, $R$ is non-simple
  - for any subproperty axiom $S \subseteq R$ with $S$ non-simple, $R$ is non-simple
  - all other properties are simple

- Example: $Q \circ P \subseteq R \quad R \circ P \subseteq R \quad R \subseteq S \quad P \subseteq R \quad Q \subseteq S$

  - non-simple: $R, S$
  - simple: $P, Q$
Today’s Session: DLs (towards OWL)

1. Basic Ideas
2. The Description Logic SROIQ(D)
3. Different Perspectives
4. Class Project
5. Class Presentations
OWL – Extralogical Features

- OWL ontologies have URIs and can be referenced by others via import statements
- Namespace declarations
- Entity declarations (must be done)
- Versioning information etc.

- Annotations
  - Entities and axioms (statements) can be endowed with annotations, e.g. using rdfs:comment.
  - OWL syntax provides annotation properties for this purpose.

Note: We still have to give a syntax for OWL – forthcoming.
The modal logic perspective

- Description logics can be understood from a modal logic perspective.
- Each pair of $\forall R$ and $\exists R$ statements give rise to a pair of modalities.
- Essentially, some description logics are multi-modal logics.
### The RDFS perspective

<table>
<thead>
<tr>
<th>RDFS semantics is weaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>• :mary rdf:type :Person .</td>
</tr>
<tr>
<td>• :Mother rdfs:subClassOf :Woman .</td>
</tr>
<tr>
<td>• :john :hasWife :Mary .</td>
</tr>
<tr>
<td>• :hasWife rdfs:subPropertyOf</td>
</tr>
<tr>
<td>:hasSpouse</td>
</tr>
<tr>
<td>• :hasWife rdfs:range :Woman .</td>
</tr>
<tr>
<td>• :hasWife rdfs:domain :Man .</td>
</tr>
<tr>
<td>• Person(mary)</td>
</tr>
<tr>
<td>• Mother ⊆ Woman</td>
</tr>
<tr>
<td>• hasWife(john,mary)</td>
</tr>
<tr>
<td>• hasWife ⊆ hasSpouse</td>
</tr>
<tr>
<td>• ⊤ ⊆ ∀hasWife.Woman</td>
</tr>
</tbody>
</table>
| • ⊤ ⊆ ∀hasWife⁻.Man or
|  ∃hasWife.⊤ ⊆ ⊆ Man |

RDFS also allows to

- make statements about statements → only possible through annotations in OWL (not present in SROID(D))
- mix class names, individual names, property names (they are all URIs) → *punning* in OWL
Punning

- Description logics impose *type separation*, i.e. names of individuals, classes, and properties must be disjoint.

- In OWL 2 Full, type separation does not apply.

- In OWL 2 DL, type separation is relaxed, but a class X and an individual X are interpreted semantically as if they were different.

- Father(john)
  SocialRole(Father)

- See further below on the two different types/semantics for OWL: OWL DL and OWL Full.
Today’s Session: DLs (towards OWL)

1. Basic Ideas
2. The Description Logic SROIQ(D)
3. Different Perspectives
4. Class Project
5. Class Presentations
Class project: next step

• none this time.
Today’s Session: DLs (towards OWL)

1. Basic Ideas
2. The Description Logic SROIQ(D)
3. Different Perspectives
4. Class Project
5. Class Presentations
Class presentations – first topics

- <nothing yet>
Tuesday 10\textsuperscript{th} of January: RDF Schema
Thursday 12\textsuperscript{th} of January: RDF and RDFS Semantics
Tuesday 17\textsuperscript{th} of January: RDF and RDFS Semantics
Thursday 19\textsuperscript{th} of January: exercise session 1
Tuesday 24\textsuperscript{th} of January: OWL part 1 – Description Logics
Thursday 2\textsuperscript{nd} of February: OWL pt 2 – model-theoretic Semantics
Tuesday 7\textsuperscript{th} of February: Partonomies
Thursday 9\textsuperscript{th} of February: SPARQL
Tuesday 14\textsuperscript{th} of February: OWL part 3 – web syntax
Thursday 16\textsuperscript{th} of February: exercise session 2