

# 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)

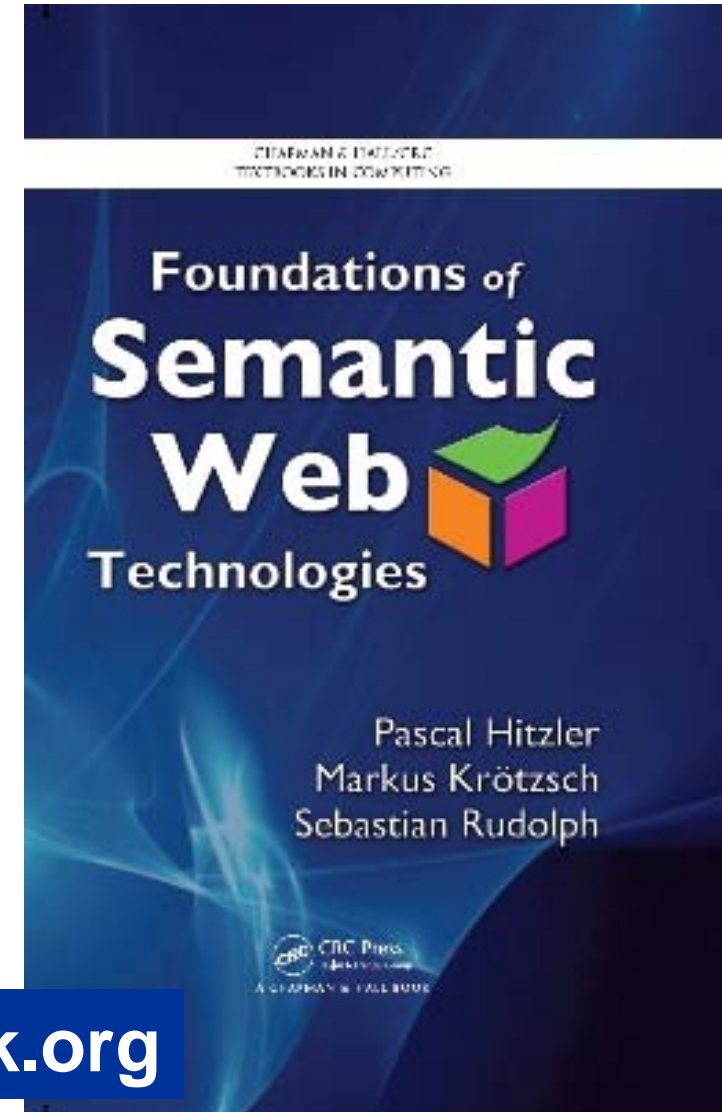
**Pascal Hitzler, Markus Krötzsch,  
Sebastian Rudolph**

**Foundations of Semantic Web  
Technologies**

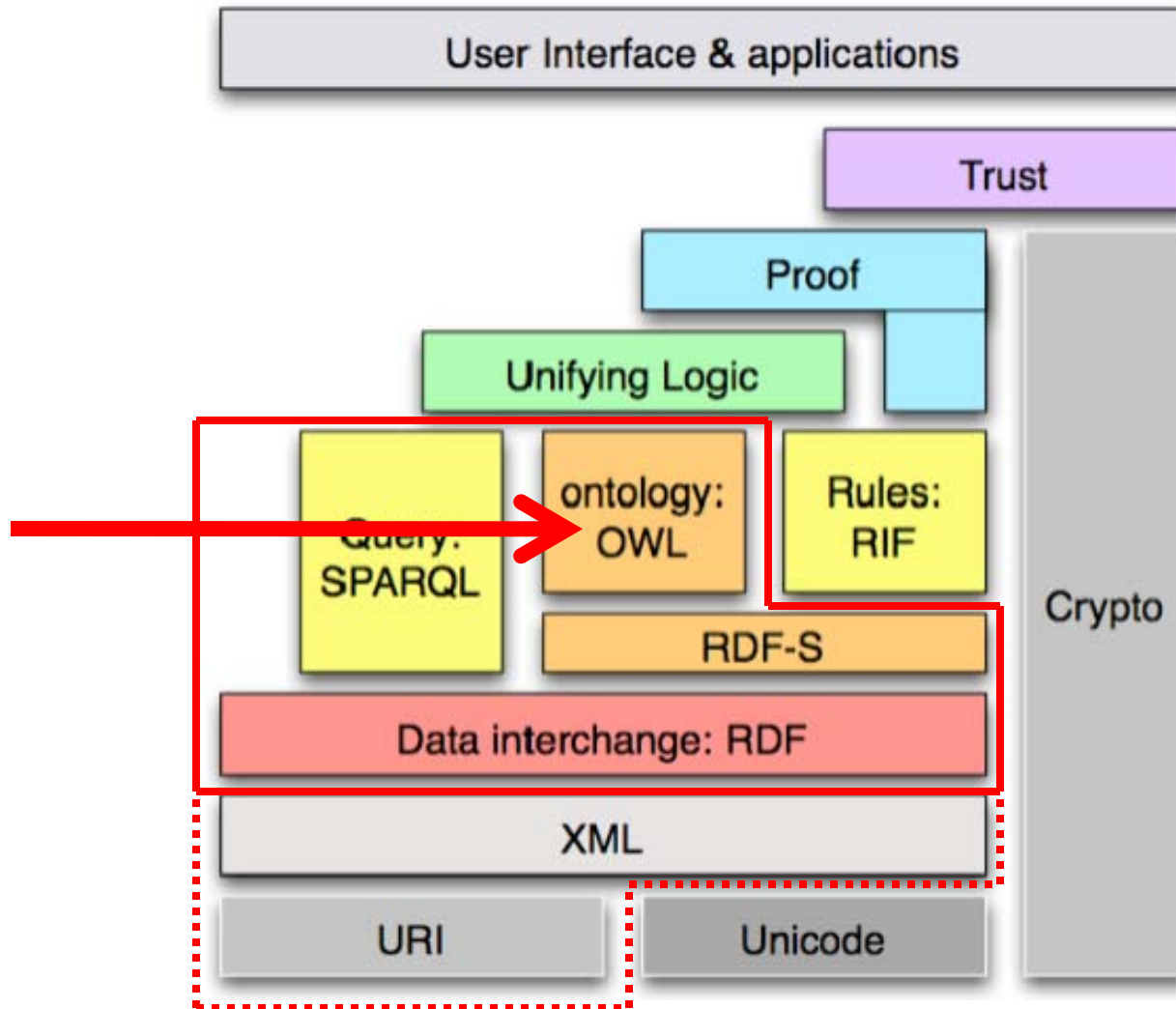
**Chapman & Hall/CRC, 2010**

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

**<http://www.semantic-web-book.org>**



# Today: Description Logics



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

- **individuals (written as URIs)**
  - also: constants (FOL), resources (RDF), instances
  - <http://example.org/sebastianRudolph>
  - <http://www.semantic-web-book.org/>
  - 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"

- **Person(mary)**
  - **:mary rdf:type :Person .**
- **Woman  $\sqsubseteq$  Person**
  - **:Woman rdfs:subClassOf :Person .**
  - **Person  $\equiv$  HumanBeing (class equivalence):**
    - Person  $\sqsubseteq$  HumanBeing      AND**
    - HumanBeing  $\sqsubseteq$  Person**
- **hasWife(john,mary)**
  - **:john :hasWife :mary .**
- **hasWife  $\sqsubseteq$  hasSpouse**
  - **:hasWife rdfs:subPropertyOf :hasSpouse**
  - **hasSpouse  $\equiv$  marriedWith (property equivalence)**

## ABox statements

- **Person(mary)**
- **Person(mary)**
- **Woman  $\sqsubseteq$  Person**
  - **Person  $\equiv$  HumanBeing (class equivalence)**
- **$\forall x$  (Woman(x)  $\rightarrow$  Person(x))**
- **hasWife(john,mary)**
- **hasWife(john,mary)**
- **hasWife  $\sqsubseteq$  hasSpouse**
  - **hasSpouse  $\equiv$  marriedWith (property equivalence)**
- **$\forall x \forall y$  (hasWife(x,y)  $\rightarrow$  hasSpouse(x,y))**

## TBox statements

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



- **conjunction**

$$\forall x (\text{Mother}(x) \leftrightarrow \text{Woman}(x) \wedge \text{Parent}(x))$$

- **Mother  $\equiv$  Woman  $\sqcap$  Parent**

- **„Mothers are exactly those who are women and parents.“**

- **disjunction**

$$\forall x (\text{Parent}(x) \leftrightarrow \text{Mother}(x) \vee \text{Father}(x))$$

- **Parent  $\equiv$  Mother  $\sqcup$  Father**

- **„Parents are exactly those who are mothers or fathers.“**

- **negation**

$$\forall x (\text{ChildlessPerson}(x) \leftrightarrow \text{Person}(x) \wedge \neg \text{Parent}(x))$$

- **ChildlessPerson  $\equiv$  Person  $\sqcap$   $\neg$ Parent**

- **„ChildlessPersons are exactly those who are persons and who are not parents.“**

- **existential quantification**

- only to be used with a role – also called a *property restriction*

- **Parent  $\equiv \exists \text{hasChild.Person}$**   
„Parents are exactly those who have at least one child which is a Person.“

$$\forall x (\text{Parent}(x) \leftrightarrow \exists y (\text{hasChild}(x,y) \wedge \text{Person}(y)))$$

- **universal quantification**

- only to be used with a role – also called a *property restriction*

- **Person  $\sqcap \text{Happy} \equiv \forall \text{hasChild.Happy}$**   
„A (person which is also happy) is exactly (something all children of which are happy).“

$$\forall x (\text{Person}(x) \wedge \text{Happy}(x) \leftrightarrow \forall y (\text{hasChild}(x,y) \rightarrow \text{Happy}(y)))$$

- **Class constructors can be nested arbitrarily**

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

## The description logic ALC

Complexity: ExpTime

- **ABox expressions:**  
Individual assignments  
Property assignments

Father(john)  
hasWife(john,mary)

- **TBox expressions**  
subclass relationships

$\sqsubseteq$

$\equiv$  for equivalence

conjunction

$\sqcap$

disjunction

$\sqcup$

negation

$\neg$

Also:  $\top$ ,  $\perp$

property restrictions

$\forall$

$\exists$

- Set of *individuals*  $a, b, c, \dots$
- Set of *atomic classes (class names)*  $A, B, \dots$
- Set of *role names*  $R, S, \dots$
- **(Complex) class expressions** are constructed as:

$$C, D ::= A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

- A **TBox** is a set of statements of the form  $C \equiv \bar{D}$  or  $C \sqsubseteq D$ , 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.

```
Human  $\sqsubseteq \exists \text{hasParent}.\text{Human}$   
Orphan  $\sqsubseteq \text{Human} \sqcap \forall \text{hasParent}.\neg \text{Alive}$   
Orphan(harrypotter)  
hasParent(harrypotter, jamespotter)
```

## ALC + role chains = SR

- `hasParent o hasBrother  $\sqsubseteq$  hasUncle.`

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

- includes top property and bottom property
- includes **S = ALC + transitivity**
  - `hasAncestor o hasAncestor  $\sqsubseteq$  hasAncestor`
- includes **SH = S + role hierarchies**
  - `hasFather  $\sqsubseteq$  hasParent`

- **O – nominals (closed classes)**
  - **MyBirthdayGuests  $\equiv$  {bill, john, mary}**
  - **Note the difference to**  
**MyBirthdayGuests(bill)**  
**MyBirthdayGuests(john)**  
**MyBirthdayGuests(mary)**
- **Individual equality and inequality (no unique name assumption!)**
  - **bill = john**
    - **{bill}  $\equiv$  {john}**
  - **bill  $\neq$  john**
    - **{bill}  $\sqcap$  {john}  $\equiv \perp$**



- **I – inverse roles**
  - **hasParent  $\equiv$  hasChild<sup>-</sup>**
  - **Orphan  $\equiv \forall \text{hasChild}^-. \text{Dead}$**
- **Q – qualified cardinality restrictions**
  - **$\leq 4$  hasChild.Parent(john)**
  - **HappyFather  $\equiv \geq 2$  hasChild.Female**
  - **Car  $\sqsubseteq =4$ hasTyre.T**
- **Complexity SHIQ, SHOQ, SHIO: ExpTime.**  
**Complexity SHOIQ: NExpTime**  
**Complexity SROIQ: N<sup>2</sup>ExpTime**

Properties can be declared to be

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

called *property characteristics*

## (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`

## (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.**

## further expressive features

- **Self**
  - **PersonCommittingSuicide  $\equiv \exists \text{kills.Self}$**
- **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**

- ABox assignments of individuals to classes or properties
- ALC:  $\sqsubseteq, \equiv$  for classes  
 $\sqcap, \sqcup, \neg, \exists, \forall$   
 $\top, \perp$
- SR: + **property chains, property characteristics, role hierarchies**  $\sqsubseteq$
- 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)

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

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

- **disjoint classes**
  - **Apple  $\sqcap$  Pear  $\sqsubseteq \perp$**
- **disjoint union**
  - **Parent  $\equiv$  Mother  $\sqcup$  Father**  
**Mother  $\sqcap$  Father  $\sqsubseteq \perp$**
- **negative property assignments (also for datatypes)**
  - **$\neg$ hasAge(jack, "53"^^xsd:integer)**

- arbitrary property chain axioms lead to undecidability
- **restriction**: set of property chain axioms has to be *regular*
  - there must be a strict linear order  $<$  on the properties
  - every property chain axiom has to have one of the following forms:

|   |                     |   |
|---|---------------------|---|
| $R \circ R \sqsubseteq R$                                   | $S^- \sqsubseteq R$ | $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$         |
| $R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$ |                     | $S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$ |
  - thereby,  $S_i < R$  for all  $i = 1, 2, \dots, n$ .
- Example 1:  $R \circ S \sqsubseteq R$        $S \circ S \sqsubseteq S$        $R \circ S \circ R \sqsubseteq T$   
→ regular with order  $S < R < T$
- Example 2:  $R \circ T \circ S \sqsubseteq T$   
→ not regular because form not admissible
- Example 3:  $R \circ S \sqsubseteq S$        $S \circ R \sqsubseteq R$   
→ not regular because no adequate order exists



- 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  $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$  with  $n > 1$ ,  $R$  is non-simple
  - for any subproperty axiom  $S \sqsubseteq R$  with  $S$  non-simple,  $R$  is non-simple
  - all other properties are simple
- Example:  $Q \circ P \sqsubseteq R$      $R \circ P \sqsubseteq R$      $R \sqsubseteq S$      $P \sqsubseteq R$      $Q \sqsubseteq S$   
non-simple:  $R, S$     simple:  $P, Q$

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

- **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.**

- **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.**

RDFS semantics is weaker

- `:mary rdf:type :Person .`
- `:Mother rdfs:subClassOf :Woman .`
- `:john :hasWife :Mary .`
- `:hasWife rdfs:subPropertyOf :hasSpouse`
- `:hasWife rdfs:range :Woman .`
- `:hasWife rdfs:domain :Man .`
- `Person(mary)`
- `Mother  $\sqsubseteq$  Woman`
- `hasWife(john,mary)`
- `hasWife  $\sqsubseteq$  hasSpouse`
- `$\top \sqsubseteq \forall \text{hasWife.Woman}$`
- `$\top \sqsubseteq \forall \text{hasWife}^{\neg} .\text{Man}$  or  $\exists \text{hasWife} .\top \sqsubseteq \text{Man}$`

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

- 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.

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

- **none this time.**



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

- <nothing yet>

**Tuesday 10<sup>th</sup> of January: RDF Schema**

**Thursday 12<sup>th</sup> of January: RDF and RDFS Semantics**

**Tuesday 17<sup>th</sup> of January: RDF and RDFS Semantics**

**Thursday 19<sup>th</sup> of January: exercise session 1**

**Tuesday 24<sup>th</sup> of January: OWL part 1 – Description Logics**

**Thursday 26<sup>th</sup> of January: OWL pt 2 – model-theoretic Semantics**

**Tuesday 31<sup>st</sup> of January: Paronomies**

**Thursday 2<sup>nd</sup> of February: OWL pt 3 – Web Syntax**

**Tuesday 7<sup>th</sup> of February: exercise session 2**