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)

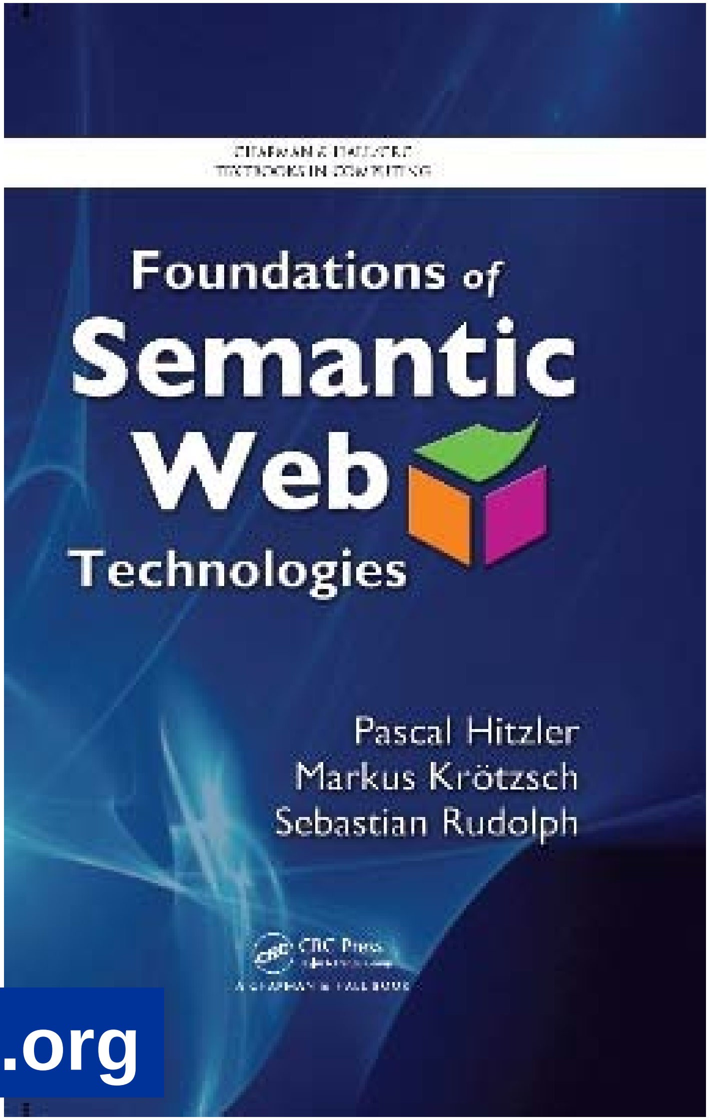
Kno.e.sis

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

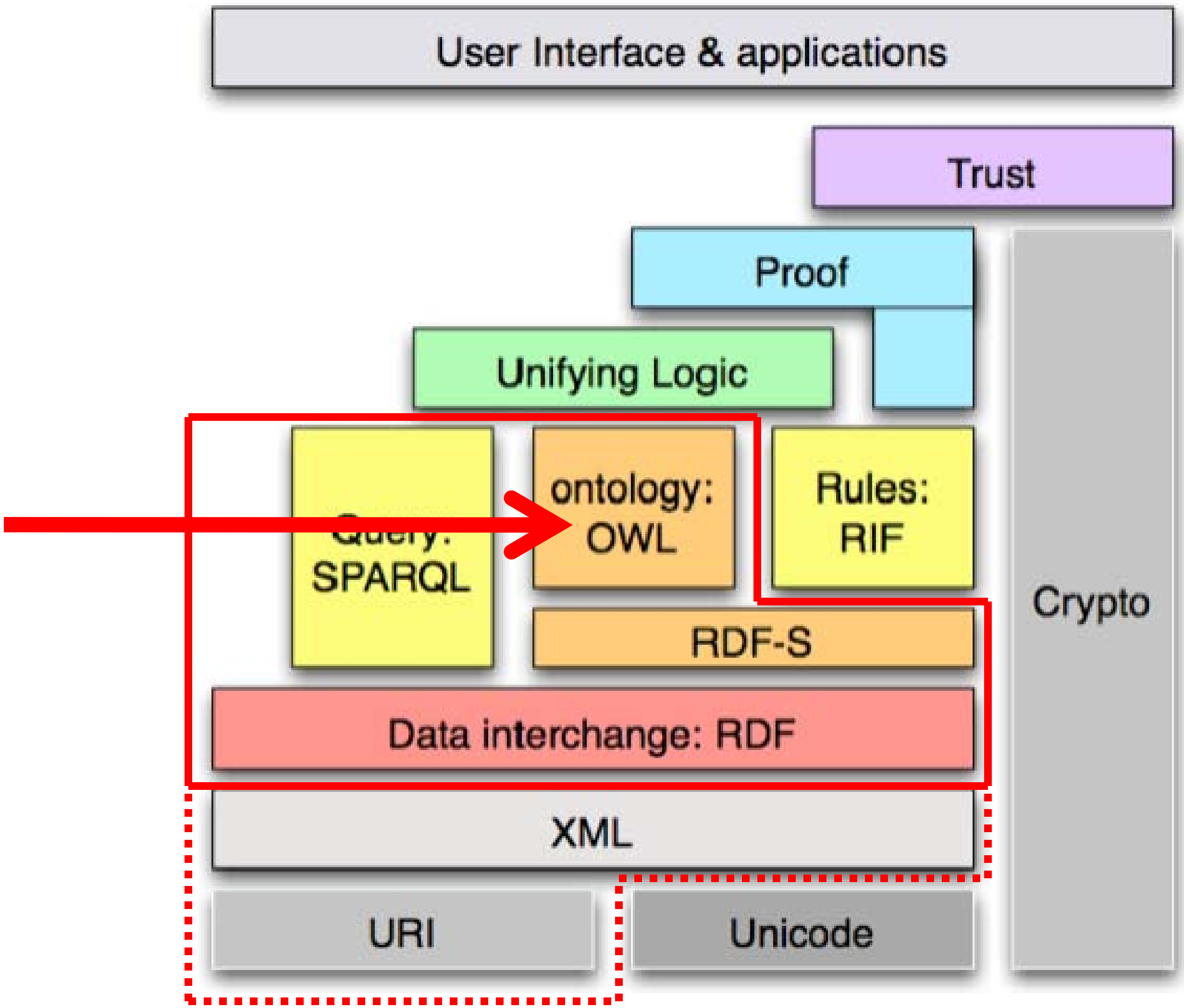
Chapman & Hall/CRC, 2010

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
  - 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"

DL syntax RDFS syntax

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

DL syntax FOL syntax

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

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**
  - **Mother ≡ Woman ⊓ Parent**
    „Mothers are exactly those who are women and parents.“

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

- **disjunction**
  - **Parent ≡ Mother ⊔ Father**
    „Parents are exactly those who are mothers or fathers.“

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

- **negation**
  - **ChildlessPerson ≡ Person ⊓ ¬Parent**
    „ChildlessPersons are exactly those who are persons and who are not parents.“

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

Class constructors

- **existential quantification**
  - **only to be used with a role – also called a *property restriction***
  - **Parent ≡ ∃hasChild.Person**
    „Parents are exactly those who have at least one child which is a Person.“

∀x (Parent(x) ↔
    ∃y (hasChild(x,y) ∧ Person(y)))

- **universal quantification**
  - **only to be used with a role – also called a *property restriction***
  - **Person ⊓ Happy ≡ ∀hasChild.Happy**
    „A (person which is also happy) is exactly (something all children of which are happy).“

∀x (Person(x) ∧ Happy(x) ↔
    ∀y (hasChild(x,y) → Happy(y)))

- **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

The description logic ALC

Complexity: ExpTime

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

- **TBox expressions**
  subclass relationships $\sqsubseteq$ $\equiv$ **for equivalence**

  conjunction $\sqcap$
  disjunction $\sqcup$
  negation $\neg$ Also: $\top$, $\bot$

  property restrictions $\forall$
  $\exists$

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:**

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

- **A *TBox* is a set of statements of the form $C \equiv D$ 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.**

The Description Logic ALC

$$\mathtt{Human} \sqsubseteq \exists \mathtt{hasParent}.\mathtt{Human}$$

$$\mathtt{Orphan} \sqsubseteq \mathtt{Human} \sqcap \forall \mathtt{hasParent}.\neg \mathtt{Alive}$$

$$\mathtt{Orphan}(\mathtt{harrypotter})$$

$$\mathtt{hasParent}(\mathtt{harrypotter}, \mathtt{jamespotter})$$

Understanding SROIQ(D)

ALC + role chains = SR

- $\mathtt{hasParent} \circ \mathtt{hasBrother} \sqsubseteq \mathtt{hasUncle}.$

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

  – **includes top property and bottom property**

- **includes S = ALC + transitivity**
  – **hasAncestor o hasAncestor ⊑ hasAncestor**

- **includes SH = S + role hierarchies**
  – **hasFather ⊑ hasParent**

Understanding SROIQ(D)

- **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} $\equiv$ $\bot$**

Understanding SROIQ(D)

- **I – inverse roles**
  - **hasParent $\equiv$ hasChild$^-$**
  - **Orphan $\equiv$ $\forall$hasChild$^-$.Dead**
- **Q – qualified cardinality restrictions**
  - **$\leq$4 hasChild.Parent(john)**
  - **HappyFather $\equiv$ $\geq$2 hasChild.Female**
  - **Car $\sqsubseteq$ =4hasTyre.$\top$**
- **Complexity SHIQ, SHOQ, SHIO: ExpTime.**
  Complexity SHOIQ: NExpTime
  Complexity SROIQ: N2ExpTime

Understanding SROIQ(D)

Properties can be declared to be

• Transitive	hasAncestor	R(a,b) and R(b,c) $\rightarrow$ R(a,c)
• Symmetric	hasSpouse	R(a,b) $\rightarrow$ R(b,a)
• Asymmetric	hasChild	R(a,b) $\rightarrow$ 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) $\rightarrow$ b=c
• InverseFunctional	hasHusband	R(a,b) and R(c,b) $\rightarrow$ 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:booelan, 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 $\equiv$ Person $\sqcap$ $\exists$hasAge.(xsd:integer: $\geq$12 and $\leq$19)**

    note: this is not standard DL notation! It‘s really only used in OWL.

Understanding SROIQ(D)

further expressive features

- **Self**
  - **PersonCommittingSuicide ≡ ∃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: $\sqsubseteq$, $\equiv$ for classes
  $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$
  $\top$, $\bot$
- 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)

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
  - $\text{Apple} \sqcap \text{Pear} \sqsubseteq \bot$
- disjoint union
  - $\text{Parent} \equiv \text{Mother} \sqcup \text{Father}$
    $\text{Mother} \sqcap \text{Father} \sqsubseteq \bot$
- negative property assignments (also for datatypes)
  - ¬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:

    $R \circ R \sqsubseteq R$ $\quad$ $S^{-} \sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$

    $R \circ S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \ldots \circ S_n \circ R \sqsubseteq R$
  - thereby, $S_i \prec R$ for all $i = 1, 2, \ldots, n$.

- Example 1: $R \circ S \sqsubseteq R$ $\quad$ $S \circ S \sqsubseteq S$ $\quad$ $R \circ S \circ R \sqsubseteq T$

  → regular with order $S \prec R \prec T$
- Example 2: $R \circ T \circ S \sqsubseteq T$

  → not regular because form not admissible
- Example 3: $R \circ S \sqsubseteq S$ $\quad$ $S \circ R \sqsubseteq R$

  → 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 o S_2 o $_{...}$ o S_n ⊑ R with n>1, R is non-simple
  - for any subproperty axiom S ⊑ R with S non-simple, R is non-simple
  - all other properties are simple
- Example: Q o P ⊑ R R o P ⊑ R R ⊑ S P ⊑ R Q ⊑ 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 ∀R and ∃R statements give rise to a pair of modalities.
- Essentially, some description logics are multi-modal logics.

The RDFS perspective

RDFS semantics is weaker

- :mary rdf:type :Person .
- :Mother rdfs:subClassOf :Woman .
- :john :hasWife :Mary .
- :hasWife rdfs:subPropertyOf :hasSpouse

- Person(mary)
- Mother $\sqsubseteq$ Woman
- hasWife(john,mary)
- hasWife $\sqsubseteq$ hasSpouse

- :hasWife rdfs:range :Woman .
- :hasWife rdfs:domain :Man .

- $\top \sqsubseteq \forall \text{hasWife}.\text{Woman}$
- $\top \sqsubseteq \forall \text{hasWife}^{-}.\text{Man}$ or $\exists \text{hasWife}.\top \sqsubseteq \text{Man}$

RDFS also allows to

- make statements about statements
  $\rightarrow$ only possible through annotations in OWL (not present in SROIQ(D))
- mix class names, individual names, property names (they are all URIs)
  $\rightarrow$ *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>

Class planning (tentative)

Tuesday 10th of January: RDF Schema
Thursday 12th of January: RDF and RDFS Semantics
Tuesday 17th of January: RDF and RDFS Semantics

Thursday 19th of January: exercise session 1

Tuesday 24th of January: OWL part 1 – Description Logics

Thursday 2nd of February: OWL pt 2 – model-theoretic Semantics

Tuesday 7th of February: Partonomies

Thursday 9th of February: SPARQL

Tuesday 14th of February: OWL part 3 – web syntax

Thursday 16th of February: exercise session 2