Knowledge Representation for the Semantic Web

Winter Quarter 2010

Slides 6 – 02/04/2010

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

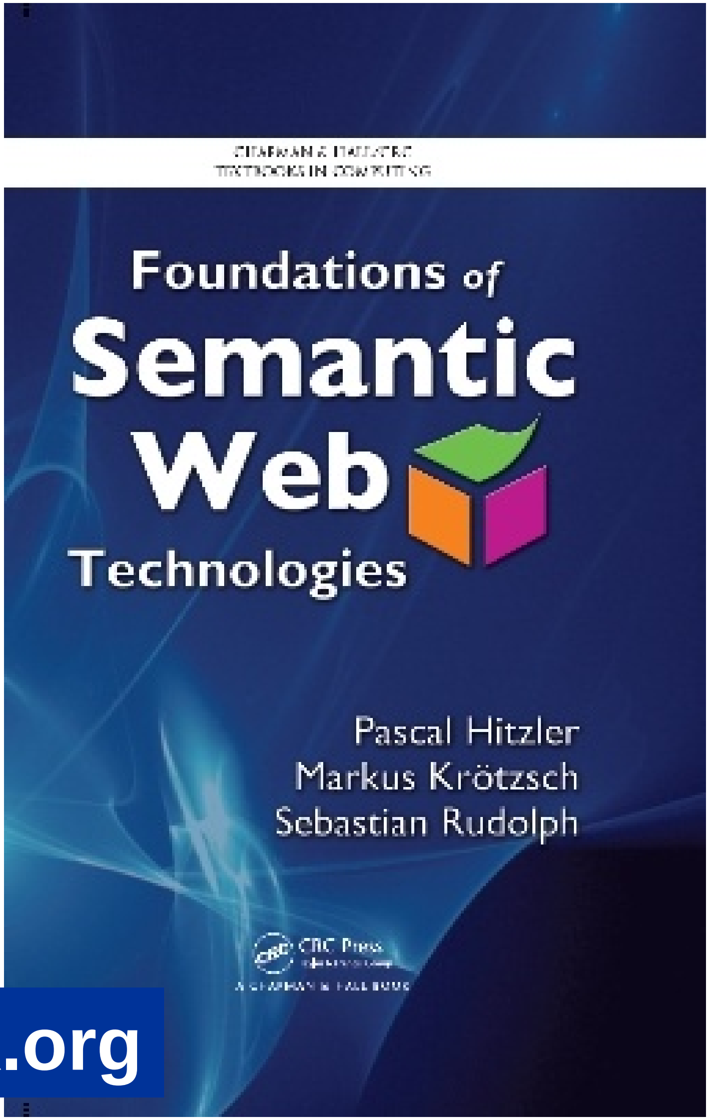
Slides are based on

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

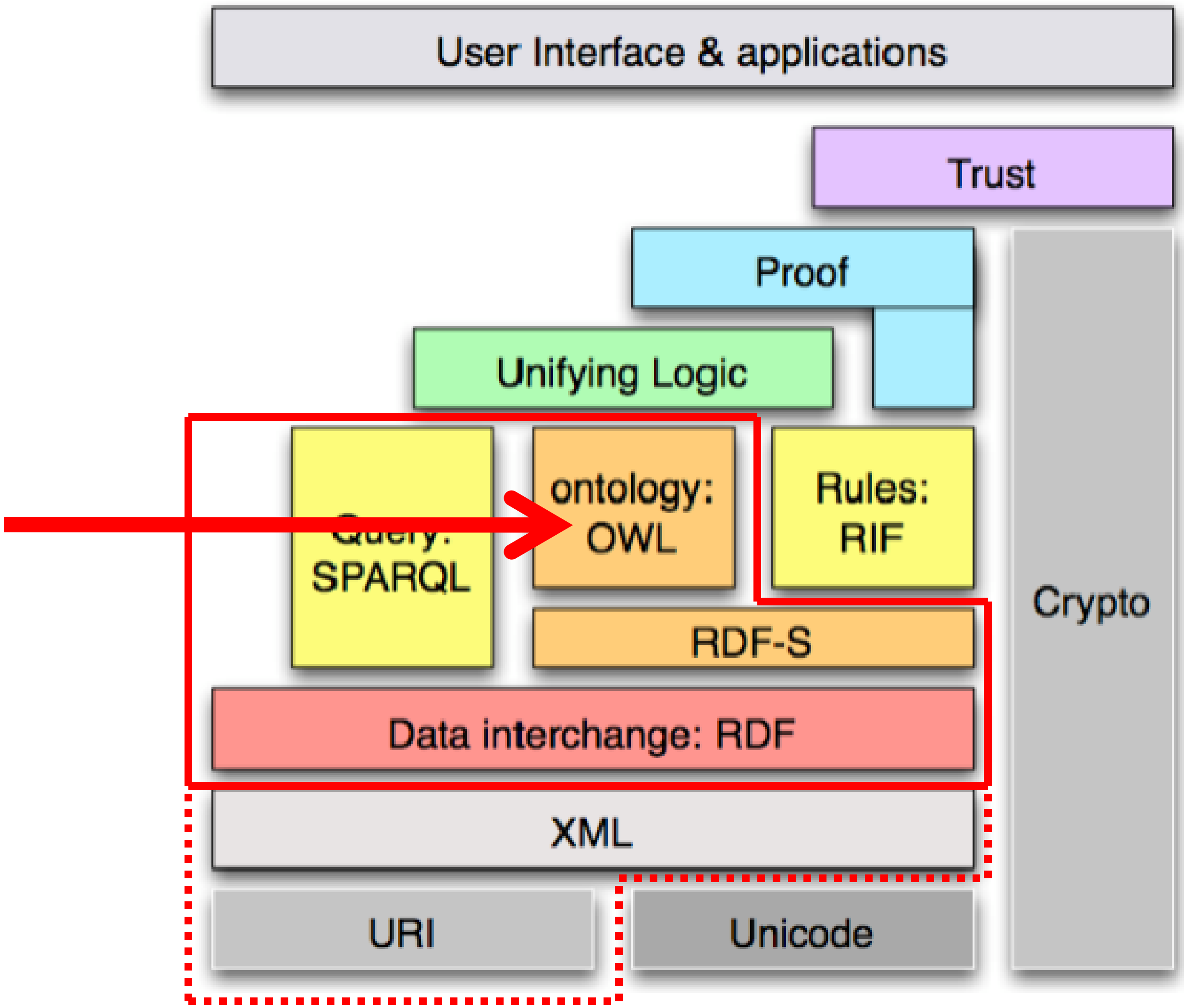
Chapman & Hall/CRC, 2010

Flyer with special offer is available.



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**
  - **: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 (class equivalence)**

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 (class 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**

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

  – **Mother ≡ Woman ⊓ 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 ≡ Mother ⊔ 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 ≡ Person ⊓ ¬Parent**
  „ChildlessPersons are exactly those who are persons and who are not parents.“

Class constructors

- **existential quantification**
  - **only to be used with a role – also called a *property restriction***
  - **Parent $\equiv \exists$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 $\sqcap$ Happy $\equiv \forall$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**

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

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

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

  - **includes top property and bottom property**

- **includes S = ALC + transitivity**
  - **hasAncestor o hasAncestor $\sqsubseteq$ hasAncestor**

- **includes SH = S + role hierarchies**
  - **hasFather $\sqsubseteq$ hasParent**

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 $\equiv$ $\exists$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

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

- **disjoint classes**
  - **Apple $\sqcap$ Pear $\sqsubseteq$ $\bot$**
- **disjoint union**
  - **Parent $\equiv$ Mother $\sqcup$ Father**
    Mother $\sqcap$ 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, . . . , *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 $S_1 \circ S_2 \circ_{\dots} \circ S_n \sqsubseteq R$ 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: $Q \circ P \sqsubseteq R$ $\quad R \circ P \sqsubseteq R$ $\quad R \sqsubseteq S$ $\quad P \sqsubseteq R$ $\quad Q \sqsubseteq S$

  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

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

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

- RDFa – embedding RDF in HTML (W3C standard)
  Pavan, Thursday 28th of January

- Scalable Distributed Reasoning using MapReduce (Urbani, Kotoulas, Oren, van Harmelen, ISWC2009)
  Wenbo, Thursday 28th of January

All remaining presentations will be in the last week

- **Semantic MediaWiki, Vinh, to be scheduled**
- **Linked Open Data, Ashutosh, to be scheduled**
- **FOAF, Hemant, to be scheduled**
- **Virtuoso, Pramod, to be scheduled**
- **Prateek, Conjunctive Queries for OWL**
- **Raghava, DL-Lite**

Class Planning

Thursday 4th of February: OWL Part 1

Tuesday 9th of February: OWL Part 2

Thursday 11th of February: OWL Part 3

Tuesday 23rd of February: Exercise Session

Thursday 25th of February: OWL Part 4

Week from March 8th: Class Presentations

Friday March 12th: most exams

Estimated breakdown of sessions:

Intro + XML: 2

OWL: 4

Class Project Session: 2

Exercise sessions: 2.7

RDF: 3.3

SPARQL: 1

Class Presentations: 3