Semantic Web Technologies I

Lehrveranstaltung im WS10/11

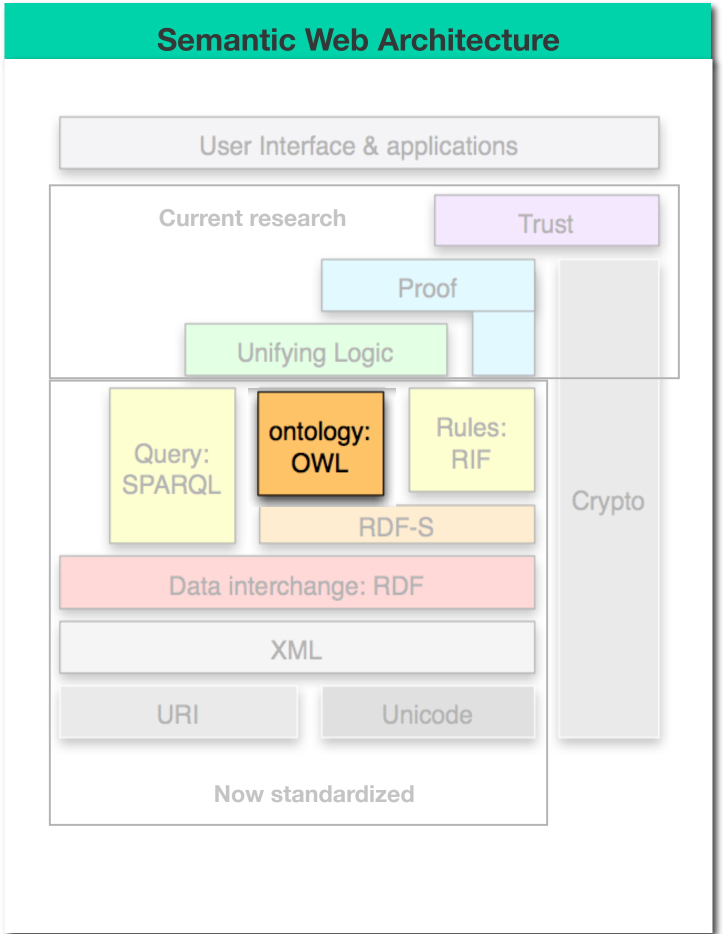
Dr. Andreas Harth
Dr. Sebastian Rudolph

AIFB

OWL – Syntax & Intuition

2/2

Dr. Sebastian Rudolph



entspricht Kapitel 5 des Buches „Semantic Web - Grundlagen" - siehe auch http://www.semantic-web-grundlagen.de/index.php/Kapitel_5

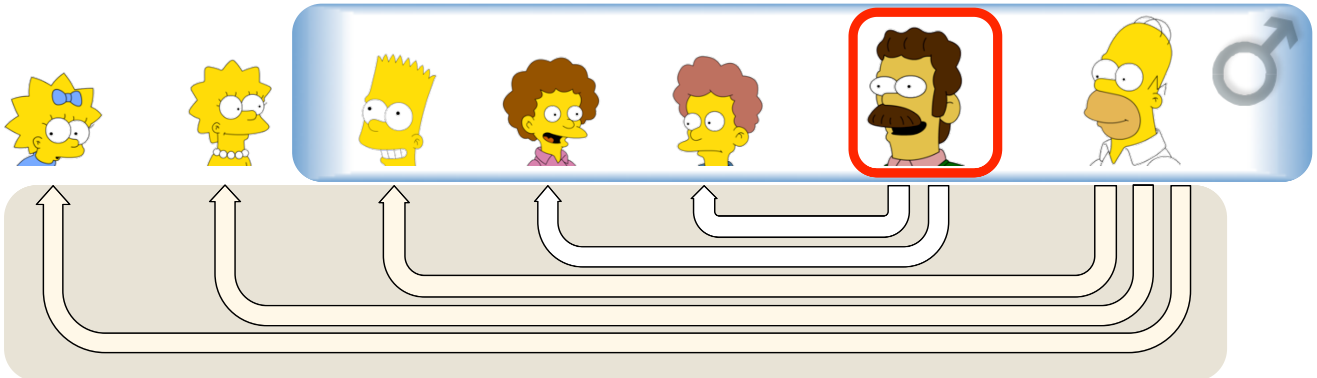
Outline

- Advanced Features of OWL
  - more class constructors
  - extended property modeling
  - handling of data values
  - OWL Profiles

More Complex Classes: Qualified At-Least Restriction

- [rdf:type owl:Restriction ;
  owl:minQualifiedCardinality
  "*n*"^^xsd:nonNegativeInteger ;
  owl:onProperty *prop* ; owl:onClass *class*]

- Example:
  [rdf:type owl:Restriction ; owl:minQualifiedCardinality
  "2"^^xsd:nonNegativeInteger ;
  owl:onClass ex:Male; owl:onProperty ex:parentOf]



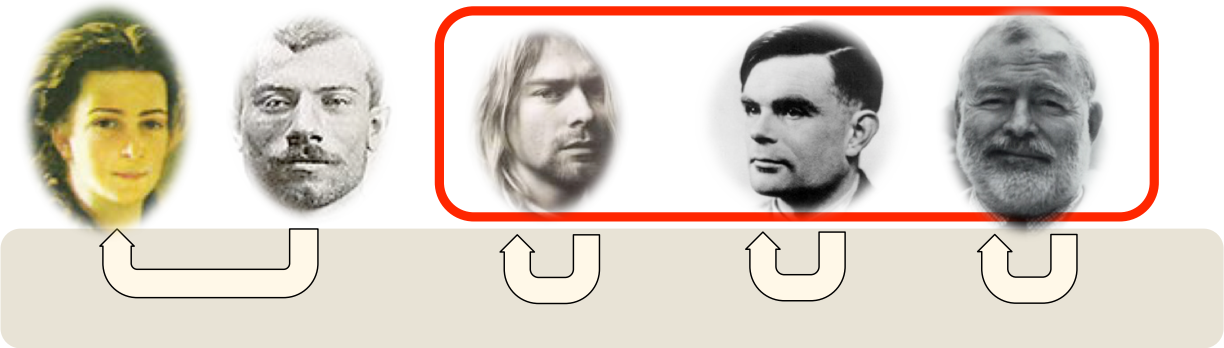
More Qualified Cardinalities

- in analogy to at-least restrictions:
  - at-most:
    owl:maxQualifiedCardinality
  - exact cardinality:
    owl:QualifiedCardinality

More Complex Classes: Self Restriction

- [rdf:type owl:Restriction ;
  owl:onProperty *prop* ;
  owl:hasSelf "true"^^xsd:boolean]

- Example: [rdf:type owl:Restriction ;
  owl:onProperty ex:hasKilled ;
  owl:hasSelf "true"^^xsd:boolean]

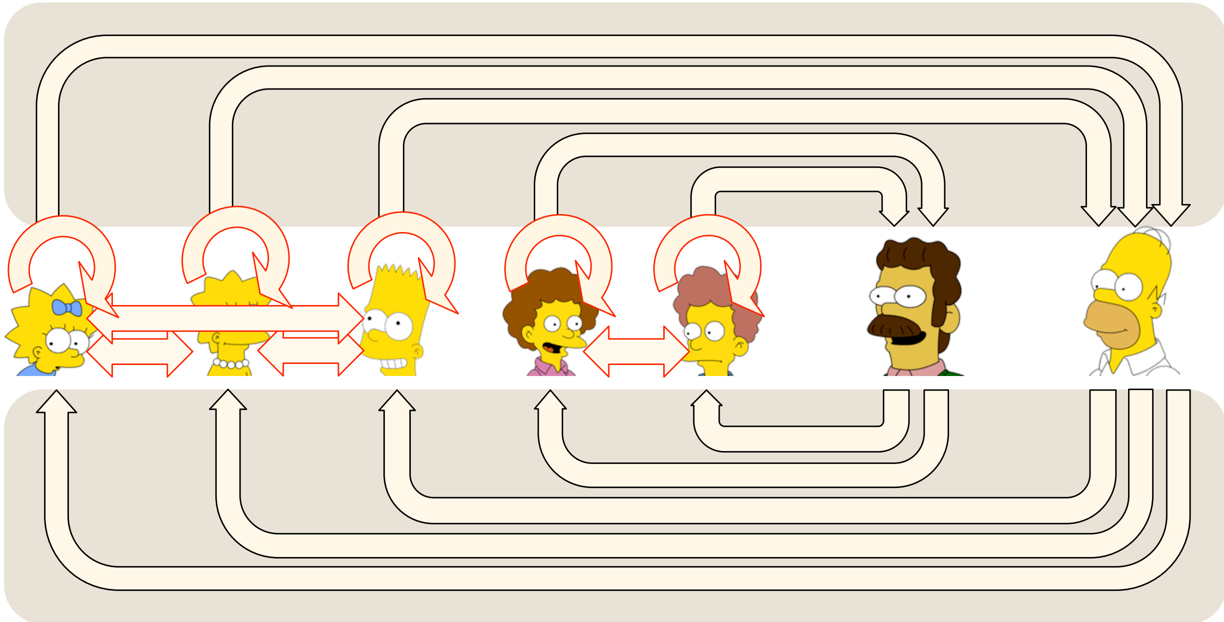


Property Chain Axioms

- *prop* owl:propertyChainAxiom (*prop1*, … , *propn*) .

- Example:

  ex:siblingOf owl:propertyChainAxiom
  (ex:childOf, ex:parentOf) .



Decidability problems

- role chain axioms can easily lead to undecidability
- in order to retain decidability, two global constraints are imposed on OWL DL ontologies:
  - the set of property chain axioms and subproperty statements must be *regular*
  - properties used in cardinality and self restrictions must be *simple* properties

Property Chain Axioms: Regularity

- in the following , we abbreviate
  R owl:propertyChainAxiom (S_1 … S_n). by $S_1 \circ \ldots \circ S_n \sqsubseteq R$
  S owlrdfs:subPropertyOf R. by $S \sqsubseteq R$

- regularity restriction: there must be a strict linear order $\prec$ on the properties such that every property chain or subproperty axiom has to have one of the following forms where $S_i \prec R$ for all i= 1, 2, . . . , *n*:

  $R \circ R \sqsubseteq R$ [owl:inverseOf R] $\sqsubseteq R$ $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$

  $R \circ S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ $S_1 \circ S_2 \circ \ldots \circ S_n \circ R \sqsubseteq R$

- Example 1: $R \circ S \sqsubseteq R$ $S \circ S \sqsubseteq S$ $R \circ S \circ R \sqsubseteq T$
  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: $R \circ S \sqsubseteq S$ $S \circ R \sqsubseteq R$
  not regular because no adequate order exists

Property Chain Axioms: Simplicity

- combining property chain axioms and cardinality or self restrictions 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 $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ 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

Semantic Web TECHNOLOGIES

AIFB

Property Characteristics

- OWL also allows for specifying that properties are:
  - disjoint from another
  - functional
  - inverse functional
  - transitive
  - symmetric
  - asymmetric
  - reflexive
  - irreflexive

syntactic sugar w.r.t. already introduced modeling features

Datatypes in OWL

- like in RDF, properties can also be used to associate individuals with data values:

  ex:john ex:hasAge “42“^^xsd:integer .

Datatype Ranges

- Property ranges for datatype properties:
  Datatypes (e.g. from XML Schema)

- Example:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
...
ex:hasAge  rdfs:range  xsd:integer .
```

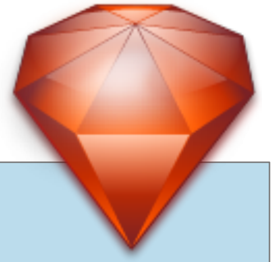
- Interpretation of datatypes defined in XML Schema (OWL adds some clarifications, e.g. “Do floating point and integer numbers overlap?”)

- Attention: datatypes still have to be explicitly specified in RDF and OWL! Given the above axiom, we find:

```
ex:jean    ex:hasAge  "17"^^xsd:integer .    ← Correct
ex:paul    ex:hasAge  "23"^^xsd:decimal .    ← Correct
ex:claire  ex:hasAge  "42" .              ← Inconsistent!
```

Defining New Datatypes

- XML Schema has ways of restricting datatypes → **datatype facets**
- Example:



```
ex:personAge  owl:equivalentClass
  [ rdf:type  rdfs:Datatype;
    owl:onDatatype  xsd:integer;
    owl:withRestrictions (
        [ xsd:minInclusive  "0"^^xsd:integer ]
        [ xsd:maxInclusive  "150"^^xsd:integer ]
    )
  ] .
```

- Possible facets depend on datatype, some facets restricted in OWL → see specs for details

Simple Data Integration in OWL

- Practical problem: given ontologies from different sources, which URIs refer to the same individuals?

- Typical approaches in OWL:
  - Explicitly specify equality with `owl:sameAs`
  - Use inverse functional properties ("same values → same individual")

- Problems:
  - `owl:sameAs` requires explicit mappings (rare on the Web)
  - OWL DL disallows inverse functional datatype properties (complicated interplay with datatype definitions!)
  - Only one property used globally for identification, no property combinations (Example: "All ESSLLI participants with the same name and birthday are the same.")

OWL 2 Keys

- OWL 2 provides a way to model
  “All ESSLLI students with same name and birthday are the same.“

- → **Keys**

```
ex:ESSLLIStudent  owl:hasKey  (ex:name, ex:birthday) .
```

- **Restriction:** Keys apply only to named individuals – objects of the interpretation domain to which a URI refers.

- More explicitly:

- If there are two URIs ***u*** and ***v***, and there is some name ***n*** and birthday ***b*** such that

```
u rdf:type ex:ESSLLIStudent; ex:name n ; ex:birthday b .
v rdf:type ex:ESSLLIStudent; ex:name n ; ex:birthday b .
```

then we conclude: **`u owl:sameAs v .`**

OWL 2 Profiles

- **Design principle for profiles:**
  Identify maximal OWL 2 sublanguages that are still implementable in PTime.

- Main source of intractability: **non-determinism** (requires guessing/backtracking)

- `owl:unionOf`, or `owl:complementOf` + `owl:intersectionOf`
- Max. cardinality restrictions
- Combining existentials (`owl:someValuesFrom`) and universals (`owl:allValuesFrom`) in superclasses
- Non-unary finite class expressions (`owl:oneOf`) or datatype expressions

→ features that are not allowed in any OWL 2 profile

OWL 2 EL

- **OWL profile based on description logic EL++**
- Intuition: focus on terminological expressivity used for light-weight ontologies
- Allow `owl:someValuesFrom` (existential) but not `owl:allvaluesFrom` (universal)
- Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, `owl:hasSelf`, `owl:hasValue`, and keys fully supported
- No inverse or symmetric properties
- `rdfs:range` allowed but with some restrictions
- No `owl:unionOf` or `owl:complementOf`
- Various restrictions on available datatypes

Semantic Web
TECHNOLOGIES

OWL 2 EL: Features

AIFB

- Standard reasoning in OWL 2 EL:
  PTime-complete

- Used by practically relevant ontologies:
  Prime example is SNOMED CT
  (clinical terms ontology with classes and properties in the order of 10^5)

- Fast implementations available:
  full classification of SNOMED-CT in <10min;
  real-time responsivity when preprocessed (modules)

OWL 2 QL

- **OWL profile that can be used to query data-rich applications:**
- Intuition: use OWL concepts as light-weight queries, allow query answering using rewriting in SQL on top of relational DBs
- Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:
  - subclasses can only be class names or `owl:someValuesFrom` (existential) with unrestricted (`owl:Thing`) filler
  - superclasses can be class names, `owl:someValuesFrom` or `owl:intersectionOf` with superclass filler (recursive), or `owl:complementOf` with subclass filler
- Property hierarchies, disjointness, inverses, (a)symmetry supported, restrictions on range and domain
- Disjoint or equivalence of classes only for subclass-type expressions
- No `owl:unionOf`, `owl:allValuesFrom`, `owl:hasSelf`, `owl:hasKey`, `owl:hasValue`, `owl:oneOf`, `owl:sameAs`, `owl:propertyChainAxiom`, `owl:TransitiveProperty`, cardinalities, functional properties

OWL 2 QL: Features

- Standard reasoning in OWL 2 QL:
  PTime, for some cases even LogSpace (<PTime)
- Convenient light-weight interface to legacy data
- Fast implementations on top of legacy database systems (relational or RDF):
  highly scalable to very large datasets

OWL 2 RL

- **OWL profile that resembles an OWL-based rule language:**
- Intuition: subclass axioms in OWL RL can be understood as rule-like implications with head (superclass) and body (subclass)
- Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:
  - subclasses can only be class names, `owl:oneOf`, `owl:hasValue`, `owl:intersectionOf`, `owl:unionOf`, `owl:someValuesFrom` if applied only to subclass-type expressions
  - superclasses can be class names, `owl:allValuesFrom` or `owl:hasValue`; also max. cardinalities of 0 or 1 are allowed, all with superclass-type filler expressions only
- Property domains and ranges only for subclass-type expressions; property hierarchies, disjointness, inverses, (a)symmetry, transitivity, chains, (inverse) functionality, irreflexivity fully supported
- Disjoint classes and classes in keys need subclass-type expressions, equivalence only for expressions that are sub- and superclass-type, no restrictions on `owl:sameAs`
- Some restrictions on available datatypes

OWL 2 RL: Features

- Standard reasoning in OWL 2 RL: PTime-complete
- Rule-based reading simplifies modeling and implementation:
  even naïve implementations can be useful
- Fast and scalable implementations exist

Do We Really Need So Many OWLs?

- **Three new OWL profiles with somewhat complex descriptions … why not just one?**
- The union of any two of the profiles is no longer light-weight!
  QL+RL, QL+EL, RL+EL all ExpTime-hard
- Restricting to fewer profiles = giving up potentially useful feature combinations
- Rationale: profiles are "maximal" (well, not quite) well-behaved fragments of OWL 2
  → Pick suitable feature set for applications
- In particular, nobody is forced to implement *all* of a profile

OWL in Practice: Tools

- Editors (http://semanticweb.org/wiki/Editors)
  - Most common editor: Protégé 4
  - Other tools: TopBraid Composer ($), *NeOn toolkit*
  - Special purpose apps, esp. for light-weight ontologies (e.g. FOAF editors)
- Reasoners (http://semanticweb.org/wiki/Reasoners)
  - OWL DL: Pellet, HermiT, FaCT++, RacerPro ($)
  - OWL EL: CEL, SHER, snorocket ($), *ELLY (*extension of IRIS*)*
  - OWL RL: OWLIM, Jena, Oracle Prime (part of O 11g) ($),
  - OWL QL: Owlgres, QuOnto, Quill
- Many tools use the **OWL API** library (Java)
- Note: many other Semantic Web tools are found online

Non-standard Reasoning in OWL

- There is more to do than editing and inferencing:
- **Explanation:** reasoning task of providing axiom sets to explain a conclusion (important for editing and debugging)
- **Conjunctive querying:** check entailment of complex query patterns (cf. Lecture 5)
- **Modularisation:** extract sub-ontologies that suffice for (dis)proving a certain conclusion
- **Repair:** determine ways to repair inconsistencies (related to explanation)
- **Least Common Subsumer:** assuming that class unions are not available, find the smallest class expression that subsumes two given classes
- **Abduction:** given an observed conclusion, derive possible input facts that would lead to this conclusion

Overview: Essential OWL Features

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	`owl:Thing`/`owl:Nothing`	(axiomatise)	$\top / \bot$
Class intersection	`owl:intersectionOf`	$\wedge$	$\sqcap$
Class union	`owl:unionOf`	$\vee$	$\sqcup$
Class complement	`owl:complementOf`	$\neg$	$\neg$
Enumerated class	`owl:oneOf`	(ax. with $\approx$)	$\{a\}$
Property restrictions	`owl:onProperty`		
Existential	`owl:someValueFrom`	$\exists y \ldots$	$\exists R.C$
Universal	`owl:allValuesFrom`	$\forall y \ldots$	$\forall R.C$
Min. cardinality	`owl:minQualifiedCardinality` `owl:onClass`	$\exists y1 \ldots yn. \ldots$	$\geq n\ S.C$
Max. cardinality	`owl:maxQualifiedCardinality` `owl:onClass`	$\forall y1 \ldots yn{+}1. \ldots \rightarrow \ldots$	$\leq n\ S.C$
Local reflexivity	`owl:hasSelf`	$R(x,x)$	$\exists R.Self$

Semantic Web TECHNOLOGIES

AIFB

Overview: Essential OWL Features

Feature	Related OWL vocabulary	DL
Property chain	`owl:propertyChainAxiom`	$\circ$
Inverse	`owl:inverseOf`	R^{-}
Key	`owl:hasKey`	rule, see Lecture 5
Property disjointness	`owl:propertyDisjointWith`	Dis(R,S)
Property characteristics	**`rdf:hasType`**	
Symmetric	`owl:SymmetricProperty`	Sym(R)
Asymmetric	`owl:AsymmetricProperty`	Asy(R)
Reflexive	`owl:ReflexiveProperty`	Ref(R)
Irreflexive	`owl:IrreflexiveProperty`	Irr(R)
Transitivity	`owl:TransitiveProperty`	Tra(R)

Subclass	`rdfs:subClassOf`	$\forall x.C(x) \rightarrow D(x)$	$C \sqsubseteq D$
Subproperty	`rdfs:subPropertyOf`	$\forall x,y.R(x,y) \rightarrow S(x,y)$	$R \sqsubseteq S$

Summary and Outlook

- OWL: expressive ontology language with practical impact
- Structurally representable in RDF (e.g. using Turtle syntax)
- Reasoning typical based on extensional (“direct”) semantics:
  - closely related to description logics and first-order logic (with equality)
  - different from RDF semantics, but compatible for many purposes
- Various flavours for different applications:
  - OWL Full provides RDF-based semantics (undecidable)
  - OWL DL decidable but complex (N2ExpTime)
  - OWL profiles for light-weight reasoning (in Ptime)

Further Reading

- P. Hitzler, S. Rudolph, M. Krötzsch: **Foundations of Semantic Web Technologies.** CRC Press, 2009. (Chapter 4 and 5 closely related to this lecture)
- W3C OWL Working Group: **OWL 2 Web Ontology LanguageDocument Overview.** See http://www.w3.org/TR/owl2-overview/. W3C Working Draft, Jun 11 2009. (overview of official OWL 2 documents)
- P. Hitzler, M. Krötzsch, B. Parsia, P.F. Patel-Schneider, S. Rudolph (editors): **OWL 2 Web Ontology Language Primer.** See http://www.w3.org/TR/owl2-primer/. W3C Working Draft, Jun 11 2009. (informative introduction to OWL 2)
- B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz: **OWL 2 Web Ontology Language Profiles.** See http://www.w3.org/TR/owl2-profiles/.W3C Candidate Recommendation, Jun 11 2009. (definition of OWL 2 profiles)