Knowledge Representation for the Semantic Web

Part I: OWL 2

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

KI 2009 Paderborn

Most recent versions of all slides available at http://semantic-web-book.org/page/KI2009

OWL (2) – Overview

- Web Ontology Language
  - W3C Recommendation for the Semantic Web, 2004
  - OWL 2 (revised W3C Recommendation) forthcoming in 2009
    - We already present this here
- Semantic Web KR language based on description logics (DLs)
  - OWL DL is essentially DL SROIQ(D)
  - KR for web resources, using URIs as identifiers
  - Using web-enabled syntaxes, e.g. based on XML or RDF
    - We mostly use concise DL syntax, some RDF syntax examples
  - Many technical and extra-logical aspects, e.g. datatypes
    - We focus on the logical core language

(for an OWL tutorial with more emphasis on RDF compatibility and datatypes, see our ESSLLI lecture materials)

OWL Rationale

An ontology language for the Web ...

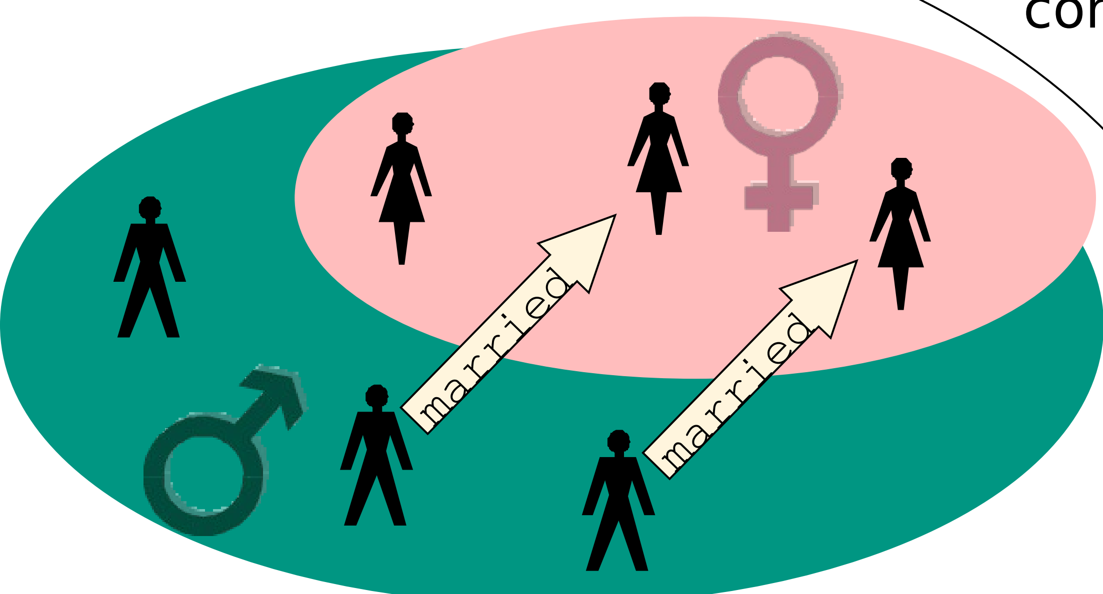
- Open World Assumption
- Reasonable trade-off between expressivity and scalability
- Integrates with RDF and RDF Schema
- Fully declarative semantics

Features (for OWL 2 DL):

- Fragment of first-order predicate logic (FOL)
- Decidable
- Known complexity classes (N2ExpTime for OWL 2 DL)
- Reasonably efficient for real KBs

What OWL Talks About

- OWL DL is based on description logics
- here, we will treat OWL from the “description logic viewpoint“:
  - we use DL syntax
  - we won't talk about datatypes and non-semantic features of OWL
- OWL (DL) ontologies talk about worlds that contain:



individuals

constants: pascal, anne

classes / concepts

unary predicates:
male(_), female(_)

properties / roles

binary predicates:
married(_,_)

Assertional Knowledge

- asserts information about concrete named individuals

  – class membership: Male(pascal)

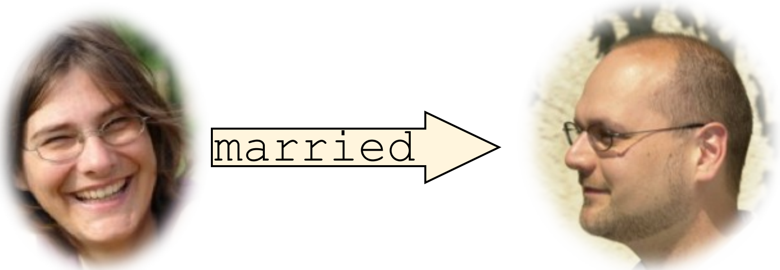
```
<Male rdf:about=“pascal"/>
```

rule version: $\rightarrow$ Male(pascal)

  – property membership: married(anne,pascal)

```
<rdf:Description rdf:about=“anne“>
  <married rdf:resource=“pascal"/>
</rdf:Description>
```

rule version: $\rightarrow$ married (anne,pascal)



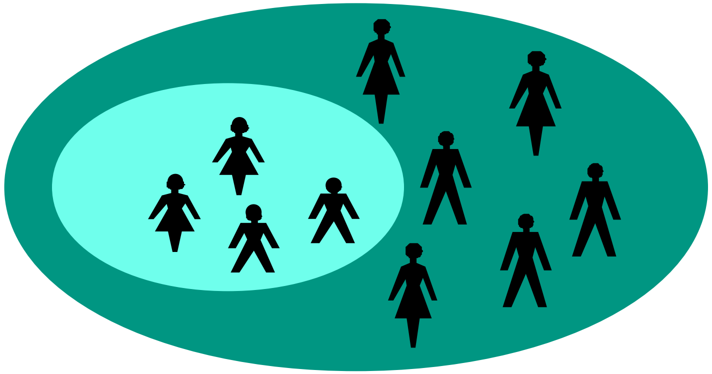
That's all that can be said in RDF.

Terminological Knowledge – Subclasses and Subproperties

- Information about how classes and properties relate in general
  - subclass: Child $\sqsubseteq$ Person

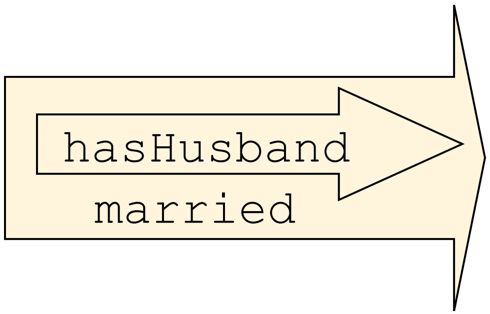
```
<owl:Class rdf:about="Child">
  <rdfs:subClassOf rdf:resource="Person"/>
</owl:Class>
```

rule version: Child(x) $\rightarrow$ Person(x)



  - subproperty: hasHusband $\sqsubseteq$ married

```
<owl:ObjectProperty rdf:about="hasHusband">
  <rdfs:subPropertyOf rdf:resource="married"/>
</owl:ObjectProperty>
```



rule version: hasHusband(x,y) $\rightarrow$ married (x,y)

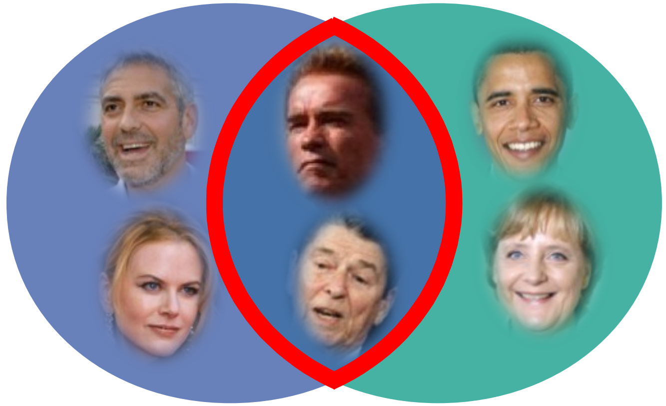
Class Constructors

- build new classes from class, property and individual names
  - union: Actor $\sqcup$ Politician

```
<owl:unionOf  rdf:parseType="Collection">
  <owl:Class rdf:about="Actor"/>
  <owl:Class rdf:about="Politician"/>
</owl:unionOf>
```

  - intersection: Actor $\sqcap$ Politician

```
<owl:intersectionOf  rdf:parseType="Collection">
  <owl:Class rdf:about="Actor"/>
  <owl:Class rdf:about="Politician"/>
</owl:intersectionOf>
```



Class Constructors

- build new classes from class, property and individual names

  - complement: ¬Politician

```
<owl:complementOf
        rdf:resource="Politician">
```

  - closed classes: {anne,merula,pascal}

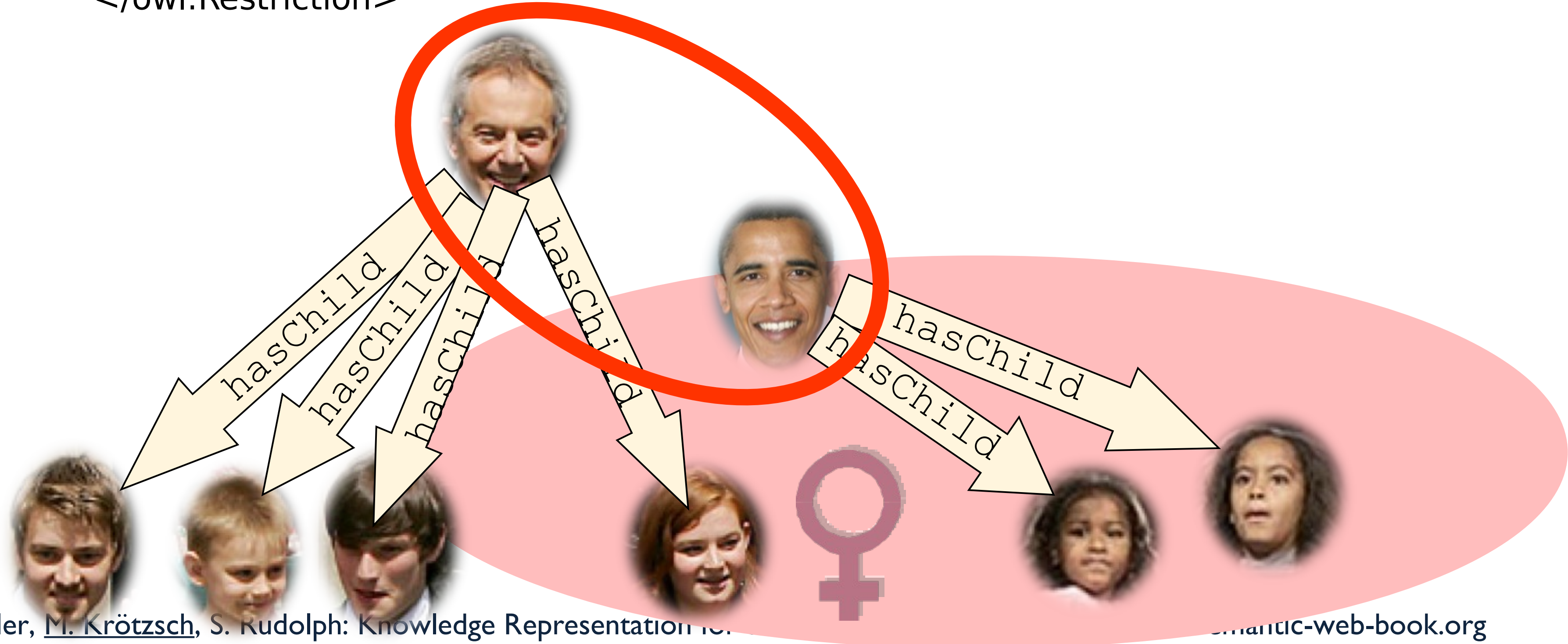
    (singleton closed classes are called *nominals* in DL)

```
<owl:oneOf rdf:parseType="Collection">
   <rdf:Description rdf:about="anne"/>
   <rdf:Description rdf:about="merula"/>
   <rdf:Description rdf:about="pascal"/>
</owl:oneOf>
```

Class Constructors

- build new classes from class, property and individual names
  - existential quantification: ∃hasChild.Female

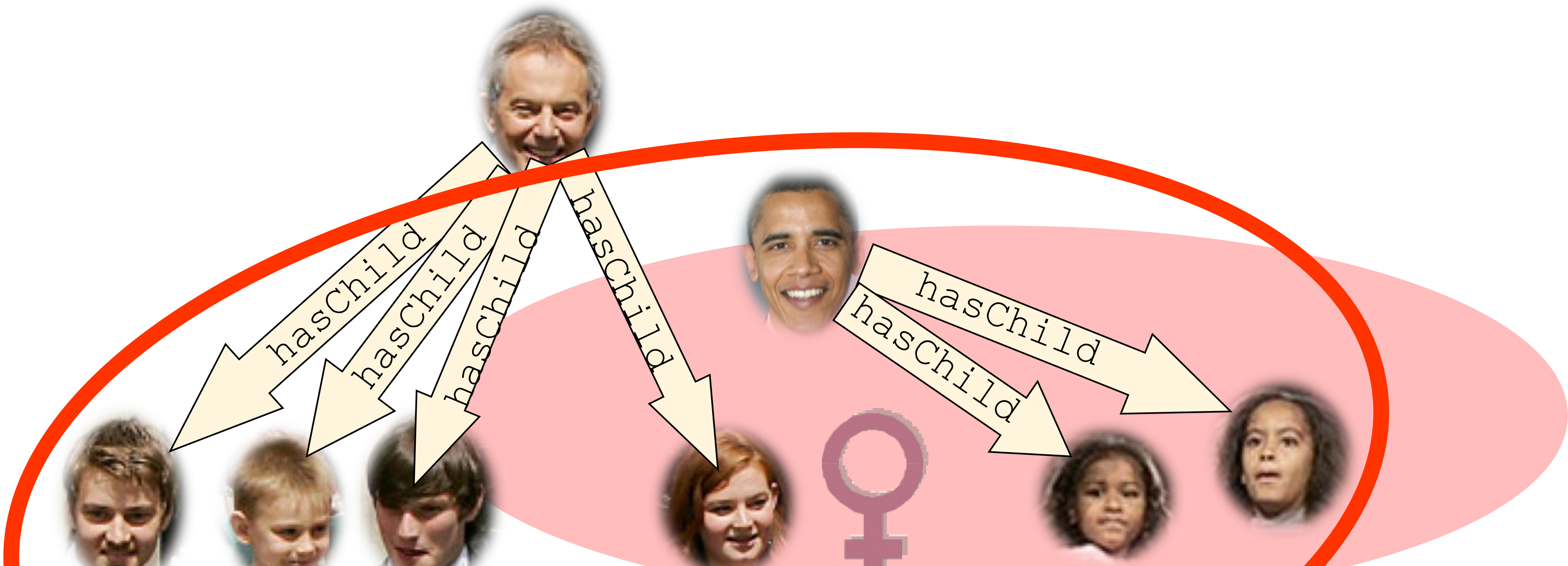
```
<owl:Restriction>
    <owl:onProperty rdf:resource="hasChild"/>
    <owl:someValuesFrom rdf:resource="Female"/>
</owl:Restriction>
```



Class Constructors

- build new classes from class, property and individual names
  - universal quantification: ∀hasChild.Female

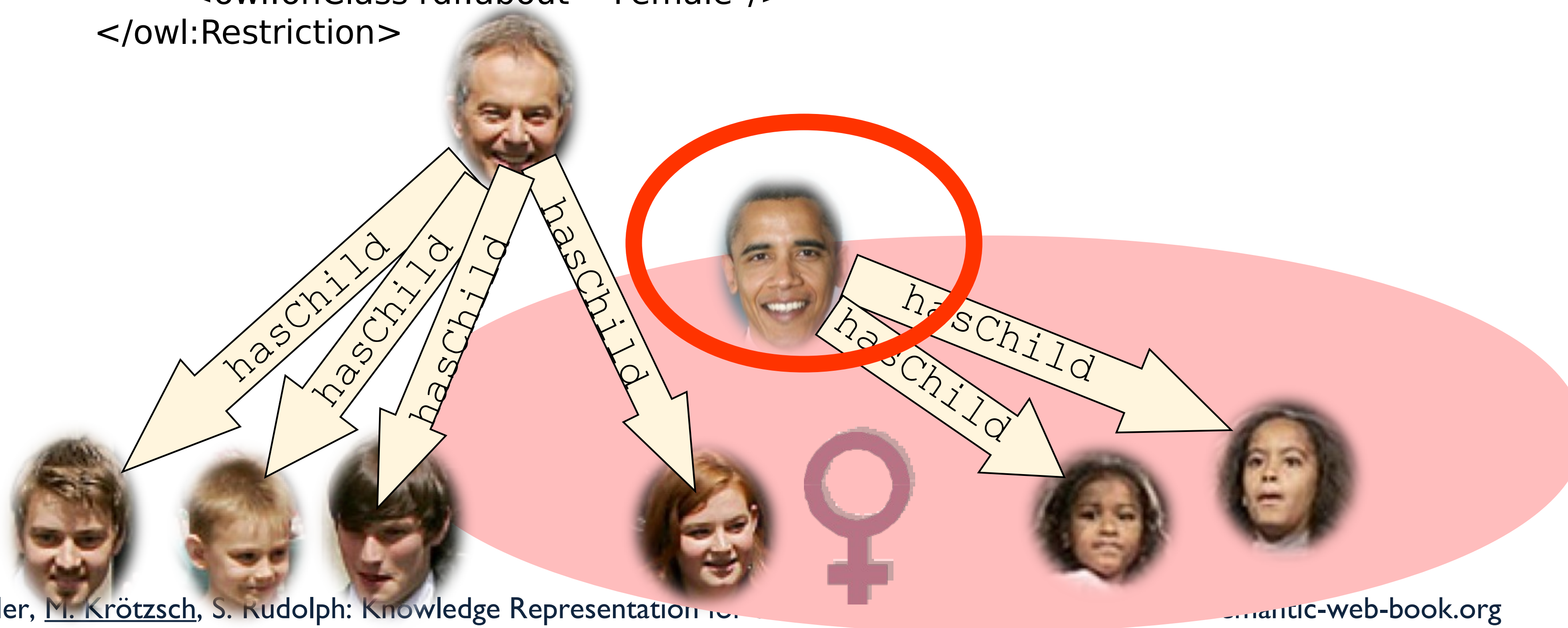
```
<owl:Restriction>
    <owl:onProperty rdf:resource="hasChild"/>
    <owl:allValuesFrom rdf:resource=“Female"/>
</owl:Restriction>
```



Class Constructors

- build new classes from class, property and individual names
  - cardinality restriction: ≥2hasChild.Female

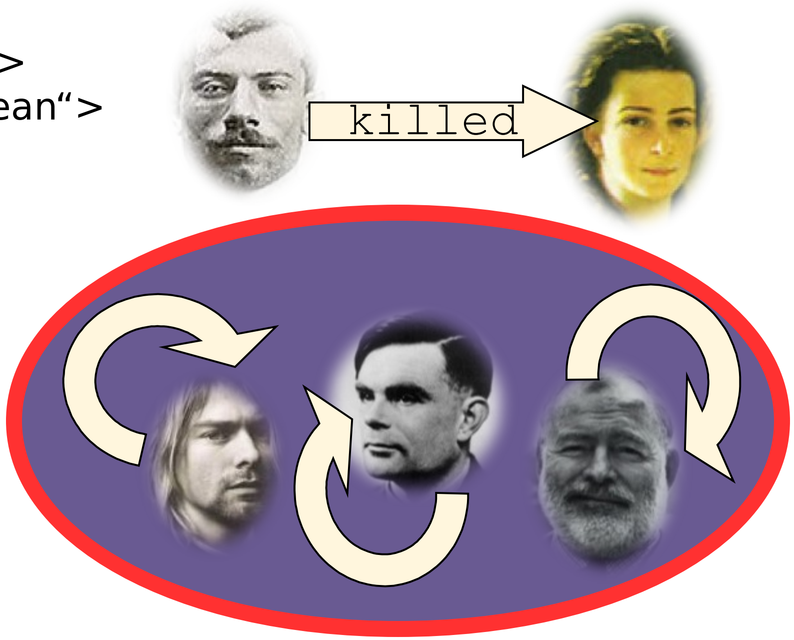
```
<owl:Restriction>
        <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger“>
         2 </owl:minQualifiedCardinality>
        <owl:onProperty rdf:about="hasChild"/>
        <owl:onClass rdf:about=“Female"/>
</owl:Restriction>
```



Class Constructors

- build new classes from class, property and individual names
  - Self-restriction (local reflexivity): ∃killed.Self

```
<owl:Restriction>
      <owl:onProperty rdf:resource=“killed"/>
      <owl:hasSelf rdf:datatype="&xsd;boolean“>
        true
      </owl:hasSelf>
</owl:Restriction>
```



Special Class Constructors

- Special classes:
  - top class: $\top$
    class containing all individuals of the domain

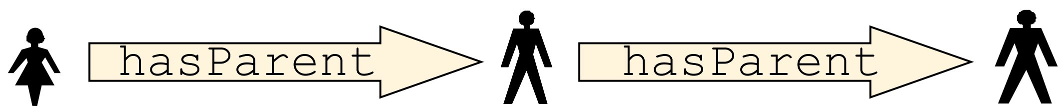
    owl:Thing
  - bottom class: $\bot$
    “empty“ class containing no individuals

    owl:Nothing
- Universal property: U
  property linking every individual to every individual

  owl:topObjectProperty

Property Chain Axioms

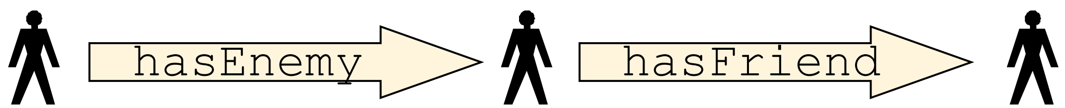
- allow to infer the existence of a property from a chain of properties:
  - $\text{hasParent} \circ \text{hasParent} \sqsubseteq \text{hasGrandparent}$
  - rule version: $\text{hasParent}(x,y) \wedge \text{hasParent}(y,z) \rightarrow \text{hasGrandparent}(x,z)$




```
<rdf:Description rdf:about="hasGrandparent">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasParent"/>
        <owl:ObjectProperty rdf:about="hasParent"/>
    </owl:propertyChainAxiom>
</rdf:Description>
```

Property Chain Axioms

- allow to infer the existence of a property from a chain of properties:
  - $\text{hasEnemy} \circ \text{hasFriend} \sqsubseteq \text{hasEnemy}$
  - rule version: $\text{hasEnemy}(x,y) \wedge \text{hasFriend}(y,z) \rightarrow \text{hasEnemy}(x,z)$




```
<rdf:Description rdf:about="hasEnemy">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasEnemy"/>
        <owl:ObjectProperty rdf:about="hasFriend"/>
    </owl:propertyChainAxiom>
</rdf:Description>
```

Property Chain Axioms: Caution! (1/2)

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:

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

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

  - thereby, $S_i \prec R$ for all i= 1, 2, . . . , *n*.

Property Chain Axioms: Caution! (1/2)

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

Property Chain Axioms: Caution! (2/2)

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

Property Chain Axioms: Caution! (2/2)

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 $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ 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: $Q \circ P \sqsubseteq R$ $\quad R \circ P \sqsubseteq R$ $\quad R \sqsubseteq S$ $\quad P \sqsubseteq R$ $\quad Q \sqsubseteq S$

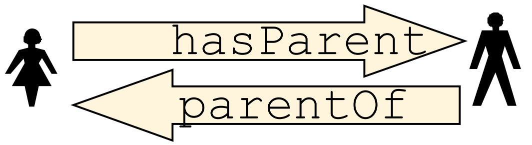
  non-simple: R, S $\qquad$ simple: P, Q

Property Characteristics

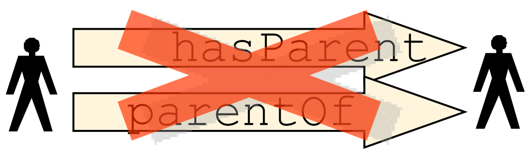
- A property can be
  - the **inverse** of another property: hasParent $\equiv$ parentOf $^{-}$

    rule version:

    hasParent(x,y) $\rightarrow$ parentOf(y,x)
    parentOf(x,y) $\rightarrow$ hasParent(y,x)



  - **disjoint** with another property: Dis(hasParent,parentOf)
    rule version:
    hasParent(x,y), parentOf(x,y) $\rightarrow$

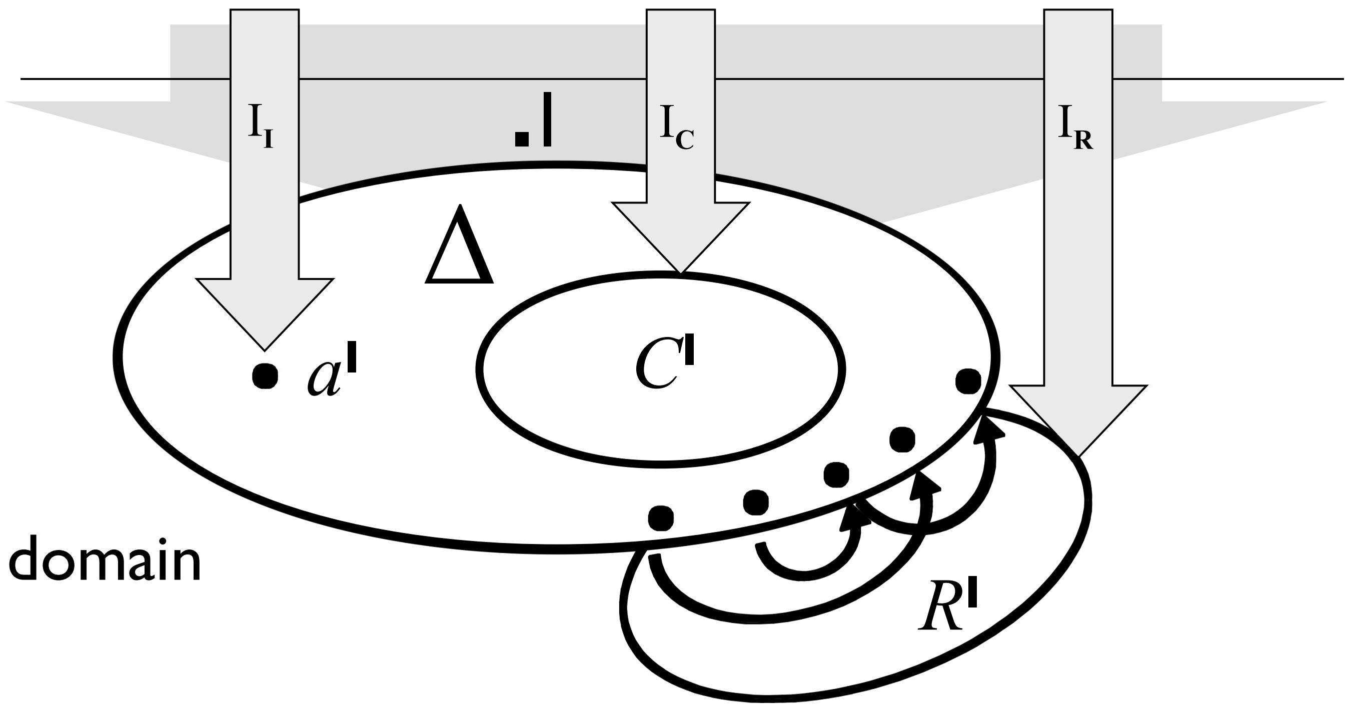


- other property characteristics that can be expressed:
  (inverse) functionality, transitivity, symmetry, asymmetry, reflexivity, irreflexivity

OWL 2 DL – Semantics

- Model-theoretic semantics
- Starts with interpretations
- An interpretation maps

individual names, class names and property names ...



... into a domain

Punning in OWL

- OWL 2 allows the same identifiers (URIs) to denote individuals, classes, and properties
- Interpretation depends on context
- A very simple form of *meta-modelling*
- Here: no punning used

  → we can use I instead of separate mappings I_I, I_C, and I_R

OWL 2 DL – Semantics

- Mapping is extended to complex class expressions:
  - $\top^I = \Delta^I$ $\qquad \bot^I = \emptyset$
  - $(C \sqcap D)^I = C^I \cap D^I$ $\qquad (C \sqcup D)^I = C^I \cup D^I (\neg C)^I = \Delta^I \setminus C^I$
  - $\forall R.C = \{ x \mid \forall (x,y) \in R^I \rightarrow y \in C^I \}$ $\qquad \exists R.C = \{ x \mid \exists (x,y) \in R^I \wedge y \in C^I \}$
  - $\geq nR.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \geq n \}$
  - $\leq nR.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \wedge y \in C^I \} \leq n \}$
- ... and to role expressions:
  - $U^I = \Delta^I \times \Delta^I$ $\qquad (R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$
- ... and to axioms:
  - $C(a)$ holds, if $a^I \in C^I$ $\quad R(a,b)$ holds, if $(a^I,b^I) \in R^I$
  - $C \sqsubseteq D$ holds, if $C^I \subseteq D^I$ $\quad R \sqsubseteq S$ holds, if $R^I \subseteq S^I$
  - $Dis(R,S)$ holds if $R^I \cap S^I = \emptyset$
  - $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ holds if $S_1^I \circ S_2^I \circ \ldots \circ S_n^I \subseteq R^I$

OWL 2 DL – Alternative Semantics

- Isn't OWL 2 DL often said to be a fragment of first-order logic?
- Indeed: there is a translation of OWL 2 DL into FOL ...

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D))$$
$$\pi_x(A) = A(x)$$
$$\pi_x(\neg C) = \neg \pi_x(C)$$
$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$
$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$
$$\pi_x(\forall R.C) = (\forall x_1)(R(x, x_1) \rightarrow \pi_{x_1}(C))$$
$$\pi_x(\exists R.C) = (\exists x_1)(R(x, x_1) \wedge \pi_{x_1}(C))$$
$$\pi_x(\geq nS.C) = (\exists x_1)\ldots(\exists x_n)\left(\bigwedge_{i \neq j}(x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C))\right)$$
$$\pi_x(\leq nS.C) = \neg(\exists x_1)\ldots(\exists x_{n+1})\left(\bigwedge_{i \neq j}(x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C))\right)$$
$$\pi_x(\{a\}) = (x = a)$$
$$\pi_x(\exists S.\mathrm{Self}) = S(x, x)$$

$$\pi(R_1 \sqsubseteq R_2) = (\forall x)(\forall y)(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2))$$
$$\pi_{x,y}(S) = S(x, y)$$
$$\pi_{x,y}(R^-) = \pi_{y,x}(R)$$
$$\pi_{x,y}(R_1 \circ \cdots \circ R_n) = (\exists x_1)\ldots(\exists x_{n-1})\left(\pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i,x_{i+1}}(R_{i+1}) \wedge \pi_{x_{n-1},y}(R_n)\right)$$
$$\pi(\mathrm{Ref}(R)) = (\forall x)\pi_{x,x}(R)$$
$$\pi(\mathrm{Asy}(R)) = (\forall x)(\forall y)(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R))$$
$$\pi(\mathrm{Dis}(R_1, R_2)) = \neg(\exists x)(\exists y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))$$

- ...which (interpreted under FOL semantics) coincides with the definition just given.

Simple Data Integration in OWL

- Practical problem: given ontologies from different sources, which identifiers refer to the same individuals?
- Typical approaches in OWL:
  - Explicitly specify equality (`owl:sameAs`)
  - Use inverse functional properties ("same values → same individual")
- Problems:
  - equality 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 KI 2009 participants with the same name and birthday are the same.")

OWL 2 Keys

OWL 2 provides a way to model
"All KI 2009 participants with same name and birthday are the same."

→ **Keys** (expressed with **`owl:hasKey`**)

- **Restriction:** Keys apply only to named individuals – objects of the interpretation domain to which a constant symbol refers.
- This is not an expressive feature of description logics!
  → see second part of this tutorial for a logical explanation

Other OWLs

- OWL 1 contained three "species" of OWL:
  - **OWL DL:** a DL-based KR language with an RDF syntax
    - not all RDF documents are OWL DL ontologies
  - **OWL Lite:** a restricted version of OWL DL
  - **OWL Full:** an extension of RDF to give semantics to the OWL keywords
    - intended to behave "similar" to OWL DL but applicable to all RDF documents
    - entailment problem undecidable (if the semantics is non-contradictory)
- OWL 2: OWL 2 DL and OWL 2 Full to extend OWL 1 species

Quo Vadis, OWL Lite?

***OWL Lite* as failure:**

- Defined as fragment of OWL 1 DL, intended to be simpler
- However: almost as complex as OWL DL (ExpTime)
- Complex syntax hides real expressive power
- Current usage in ontologies coincidental rather than intentionally

Original goal: simpler implementation and usage

→ approach in OWL 2: three simpler **language profiles:**

- **OWL 2 QL**
- **OWL 2 EL**
- **OWL 2 RL**

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)

- disjunction, or negation + conjunction
- Max. cardinality restrictions
- Combining existentials and universals in superclasses
- Non-unary finite class expressions (nominals) or datatype expressions (not discussed here)

→ features that are not allowed in any OWL 2 profile

Many further features can lead to non-determinism – care needed!

OWL 2 EL

OWL profile based on description logic EL++

- Intuition: focus on terminological expressivity used for light-weight ontologies
- Allow existential but not universal, only `rdfs:range` (special kind of universals) allowed with restrictions
- Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, *Self*, nominals (singleton classes), and keys fully supported
- No inverse or symmetric properties, no disjunctions or negations
- Examples: $\exists \text{has.Sorrow} \sqsubseteq \exists \text{has.Liqueur}$ $\quad \top \sqsubseteq \exists \text{hasParent.Person}$

$\exists \text{married.} \top \sqcap \text{CatholicPriest} \sqsubseteq \bot$ $\quad \text{German} \sqsubseteq \exists \text{knows.}\{\text{angela}\}$

$$\text{hasParent} \circ \text{hasParent} \sqsubseteq \text{hasGrandparent}$$

OWL 2 EL: Features

- 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 <1min;
  real-time responsivity when preprocessed (modules)

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:
  - subclasses can only be class names, nominals, conjunctsions, disjunctions, existentials if applied only to subclass-type expressions
  - superclasses can be class names, universals or nominals; 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 equality

OWL 2 RL Examples

- $\exists$parentOf.$\exists$parentOf.$\top \sqsubseteq$ Grandfather
  rule version: parentOf(x,y) $\wedge$parentOf(y,z) $\rightarrow$ Grandfather(x)
- Orphan$\sqsubseteq$ $\forall$hasParent.Dead
  rule version: Orphan(x) $\wedge$hasParent(x,y) $\rightarrow$ Dead(y)
- Monogamous$\sqsubseteq$ $\leq$1married.Alive
  rule version:
  Monogamous(x)$\wedge$married(x,y)$\wedge$Alive(y) $\wedge$married(x,z)$\wedge$Alive(z)$\rightarrow$ y=z
- childOf $\circ$ childOf$\sqsubseteq$ grandchildOf
  rule version: childOf(x,y) $\wedge$childOf(y,z) $\rightarrow$ grandchildOf(x,z)
- Disj(childOf,parentOf)
  rule version: childOf(x,y) $\wedge$parentOf(x,y) $\rightarrow$

OWL 2 RL: Features

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

Also: possibly useful for combining OWL with rules

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:
  - subclasses can only be class names or existentials with unrestricted ($\top$) filler
  - superclasses can be class names, existentials or conjunctions with superclass filler (recursive), or negations 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 disjunctions, universals, Self, keys, nominals, equality, property chains, transitive properties, cardinalities, or functional properties
- Example: $\exists \text{married}.\top \sqsubseteq \neg\text{Free} \sqcap \exists \text{has}.\text{Sorrow}$

OWL 2 QL: Features

- Standard reasoning in OWL 2 QL:
  PTime, instance retrieval even LogSpace (actually AC0) w.r.t. size of data
- Convenient light-weight interface to legacy data
- Fast implementations on top of legacy database systems (relational or RDF):
  highly scalable to very large datasets

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 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 ($), *ELLY* (extension of IRIS)
  - OWL RL: OWLIM, Jena, Oracle OWL Reasoner (part of O11g) ($),
  - 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
- **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
- …

→ All implemented, tasks on top common in standard tools today

Summary and Outlook

- OWL: an expressive ontology language with practical impact
- Structurally representable in RDF
- 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)

Version 2 of the Web Ontology Language almost complete:

Official specification expected by Oct 2009

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)

Selected research articles:

- I. Horrocks, O. Kutz, U. Sattler: **The even more irresistible SROIQ.** In Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). AAAI Press, 2006.
- F. Baader, S. Brandt, C. Lutz: **Pushing the EL envelope.** In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005. (paper introducing description logic EL++ underlying OWL EL)
- B. Grosof, I. Horrocks, R. Volz, S. Decker: **Description Logic Programs: Combining Logic Programs with Description Logic.** In Proc. of the 12th Int. World Wide Web Conference (WWW 2003), Budapest, Hungary, 2003. (rule-based description logic fragment that influenced OWL RL)
- H. J. ter Horst: **Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary.** J. of Web Semantics 3(2–3):79–115, 2005. (rule-based implementation of parts of OWL Full, considerations that influenced the design of OWL RL)
- D. Calvanese, G. de Giacomo, D. Lembo, M. Lenzerini, R. Rosati: **Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family.** J. of Automated Reasoning 39(3):385–429, 2007 (introduction of DL-Lite, the description logic that inspired OWL QL)