# Semantic Web
# Ontology Languages

## Part II: OWL 2

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Informatik 2009 Lübeck

semantic-web-book.org

# OWL (2) – Overview

- Web Ontology Language
  - W3C Recommendation for the Semantic Web, 2004
  - OWL 2 (currently Proposed W3C Recommendation) forthcoming this October
    - 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

# OWL Rationale

An ontology language for the Web ...

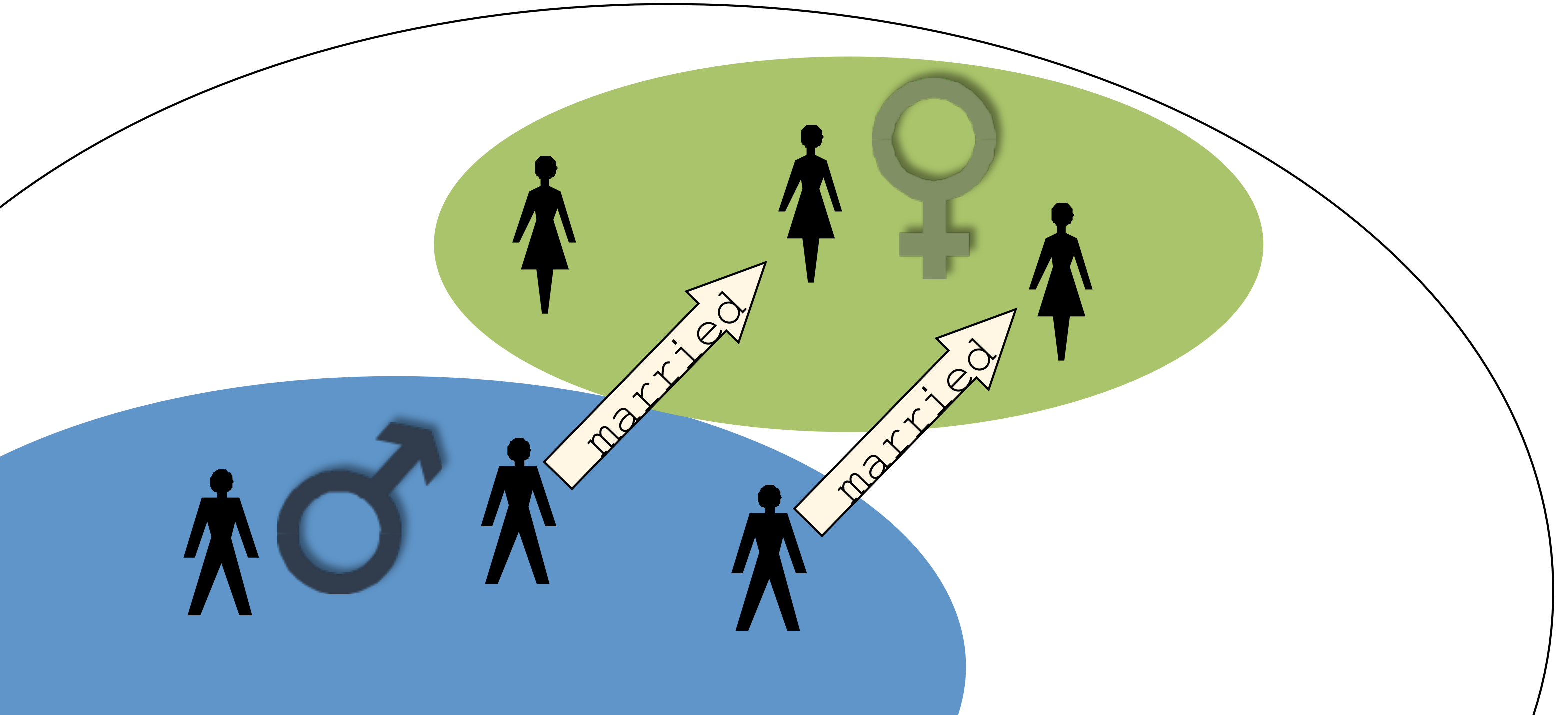
- Open World Assumption
- Reasonable trade-off between expressivity and scalability
- Integrates with RDF and RDF Schema
- Fully declarative semantics

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

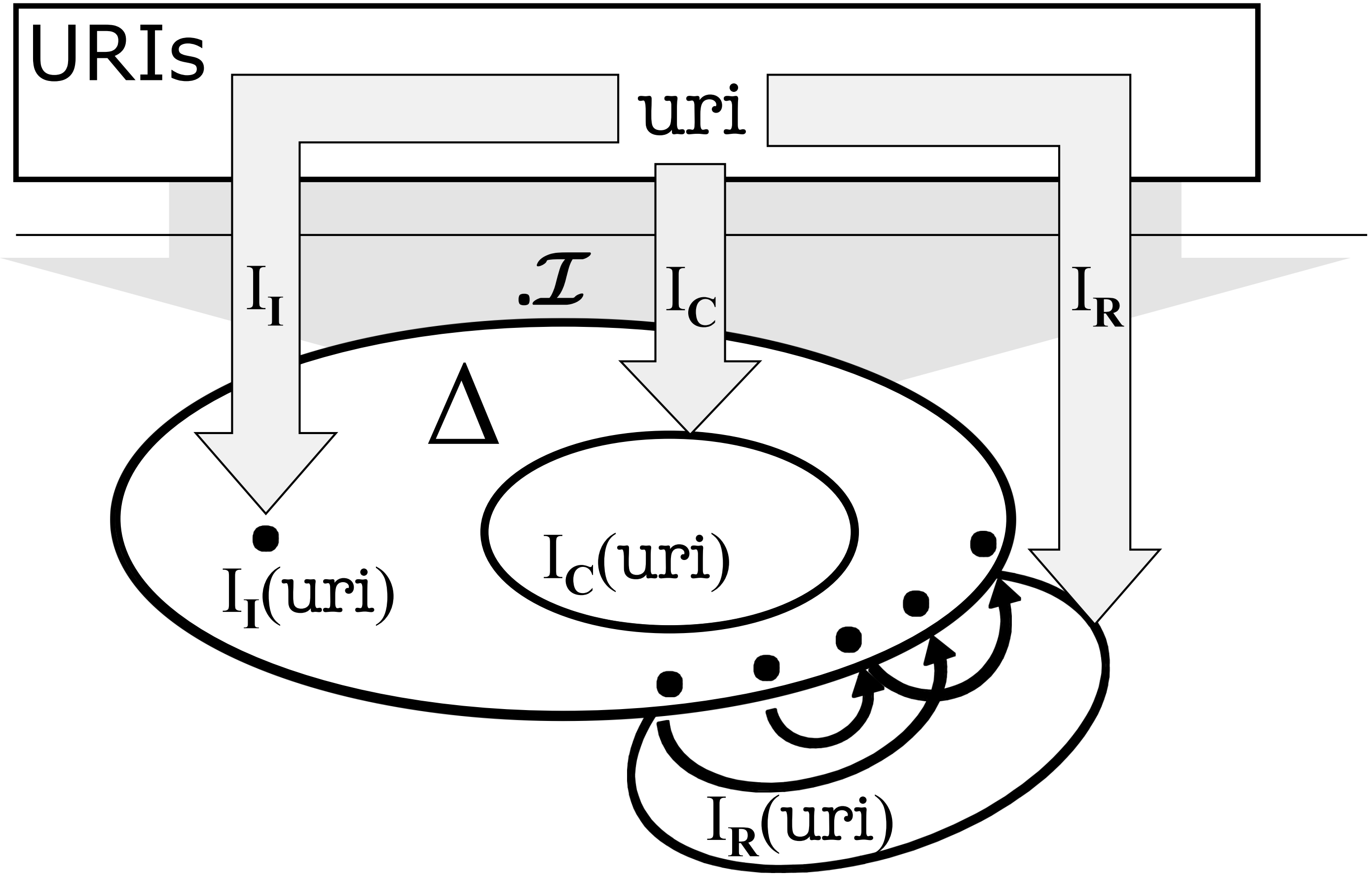
# OWL Building Blocks

- individuals (written as URIs): ex:markus
  - aka: constants (FOL), resources (RDF)
- classes (also written as URIs): ex:Female
  - aka: concepts, unary predicates (FOL)
- properties (also written as URIs): ex:married
  - aka: roles (DL), binary predicates (FOL)



# OWL Direct Semantics

- model theory (aka extensional semantics)
- OWL DL Interpretation $\mathcal{I}$:



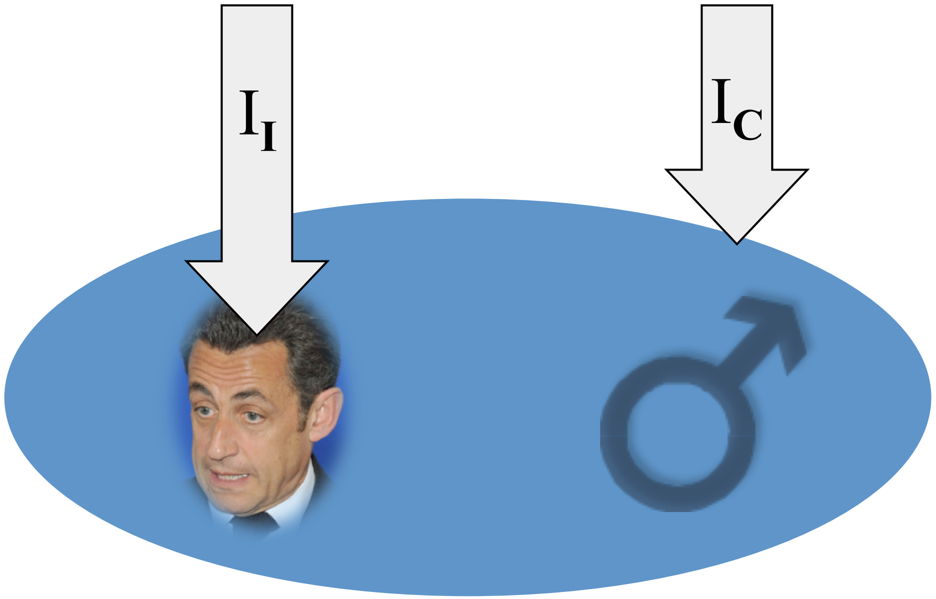
# On the OWL Syntax

- OWL statements are written down as (sets of) RDF triples
- OWL facts (aka: assertions) are written down like in RDF
- some RDF language elements are reused
- new language elements from the OWL namespace
- more complex statements are constructed by using bnodes (we “hide“ them for convenience)

# Class Membership

- induri rdf:type classuri .
- true in $\mathcal{I}$, if $I_I(\text{induri}) \in I_C(\text{classuri})$
- Example:

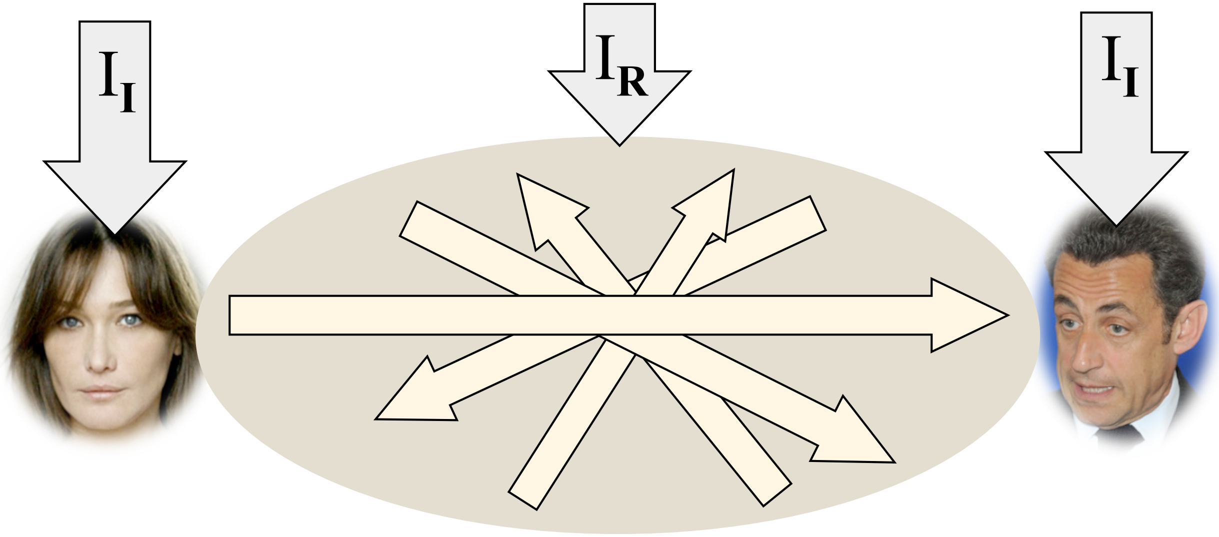
ex:nicolas rdf:type ex:Male



# Property Membership

- induri1 propuri induri2 .
- true in $\mathcal{I}$, if $\langle I_I(\text{induri1}), I_I(\text{induri2})\rangle \in I_P(\text{propuri})$
- Example:

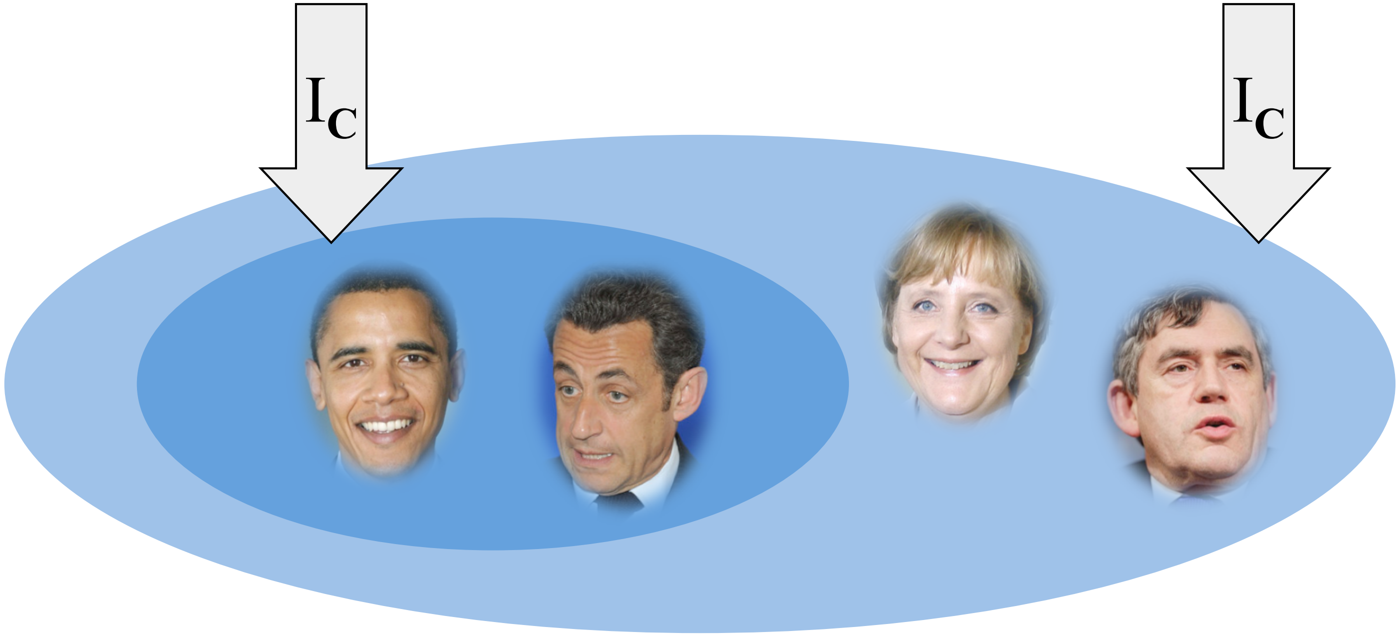
ex:carla ex:marriedWith ex:nicolas



# Class Inclusion

- classuri1 rdfs:subClassOf classuri2 .
- true in $\mathcal{I}$, if $I_C(\text{classuri1}) \subseteq I_C(\text{classuri2})$
- Example:

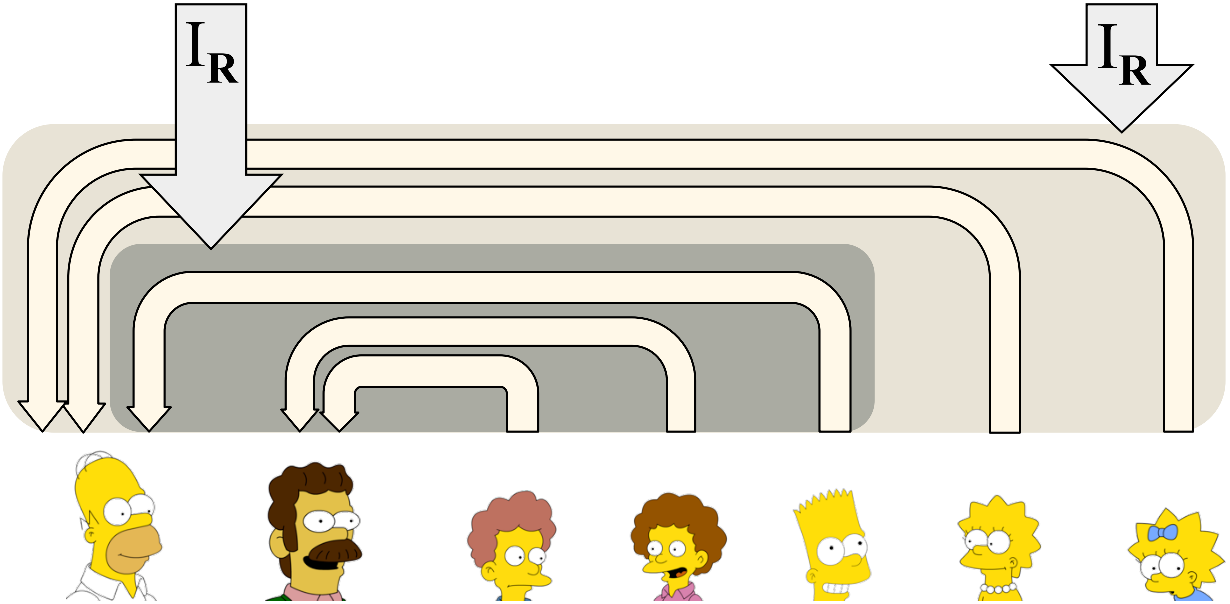
  ex:President rdfs:subClassOf ex:Politician



# Property Inclusion

- propuri1 rdfs:subPropertyOf propuri2 .
- true in $\mathcal{I}$, if $I_R(\text{propuri1}) \subseteq I_R(\text{propuri2})$
- Example:

  ex:sonOf rdfs:subPropertyOf ex:childOf



# Predefined Classes & Properties

- owl:Thing – the class containing everything
  - $I_C$(owl:Thing) = $\Delta$

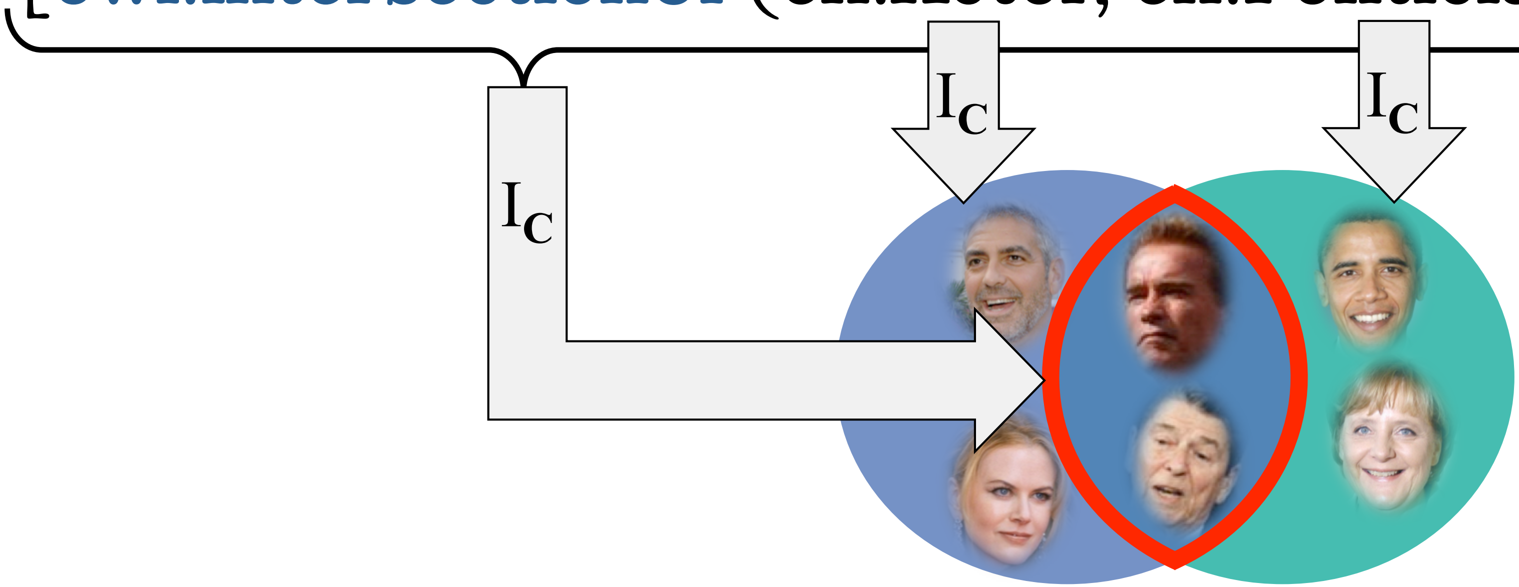
- owl:Nothing – the empty class
  - $I_C$(owl:Nothing) = $\varnothing$

- owl:topProperty – the property connecting everything
  - $I_R$(owl:topProperty) = $\Delta \times \Delta$

- owl:bottomProperty – the empty property
  - $I_R$(owl:bottomProperty) = $\varnothing$

# Complex Classes: Intersection

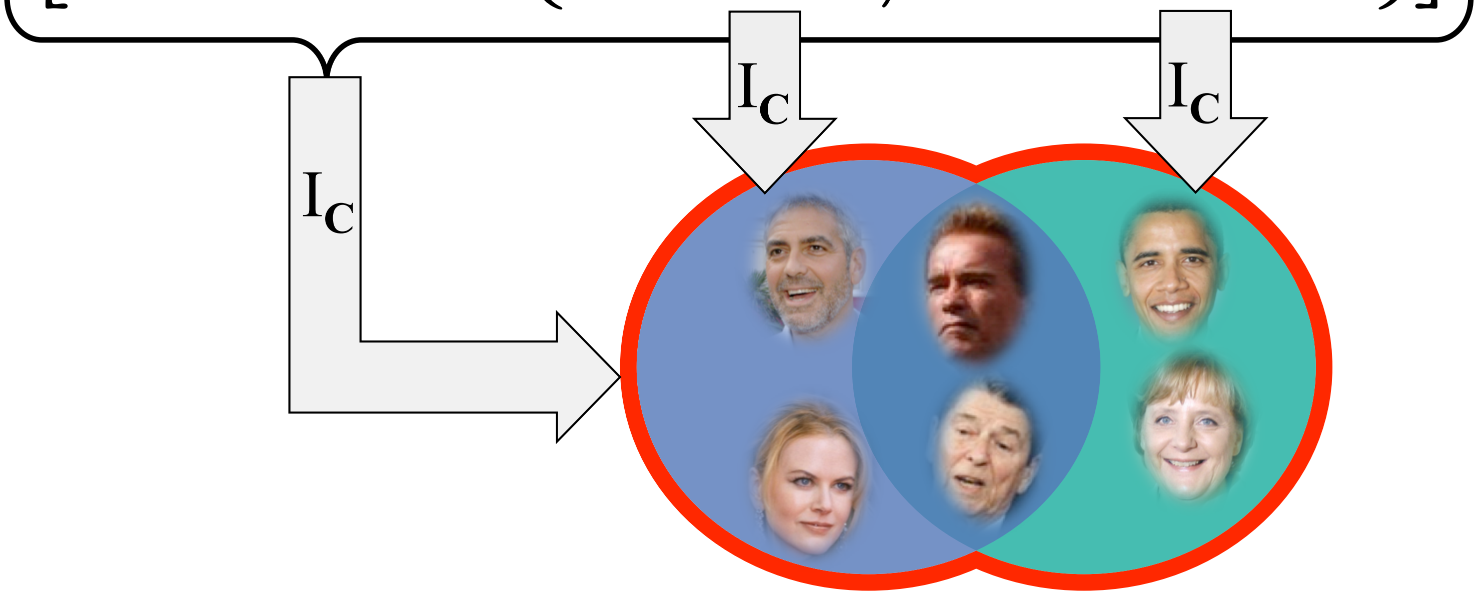
- [owl:intersectionOf (*class1* , … , *classn*)]
- $I_C$([owl:intersectionOf (*class1* , … , *classn*)])
  $= I_C(\mathit{class1}) \cap \ldots \cap I_C(\mathit{classn})$
- Example:
  [owl:intersectionOf (ex:Actor, ex:Politician)]



# Complex Classes: Union

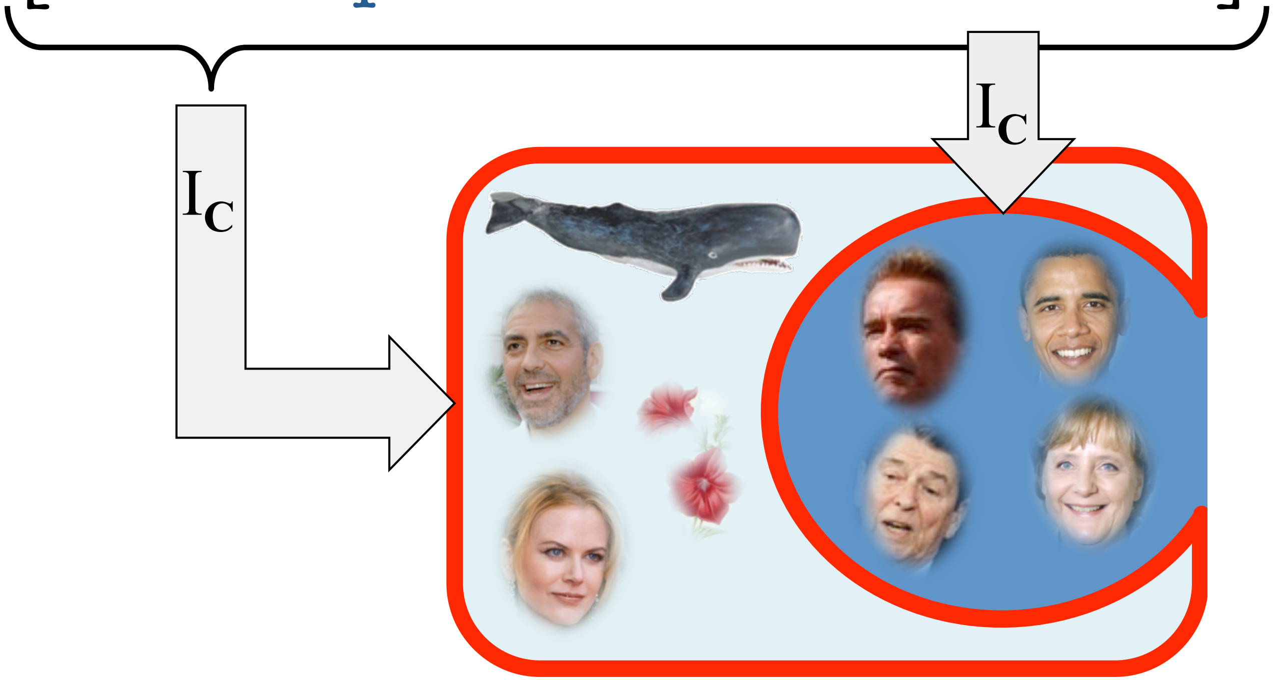
- [owl:unionOf (*class1* , … , *classn*)]
- $I_C$([owl:unionOf (*class1* , … , *classn*)])
  = $I_C$(*class1*) $\cup$ … $\cup$ $I_C$(*classn*)
- Example:

  [owl:unionOf (ex:Actor, ex:Politician)]



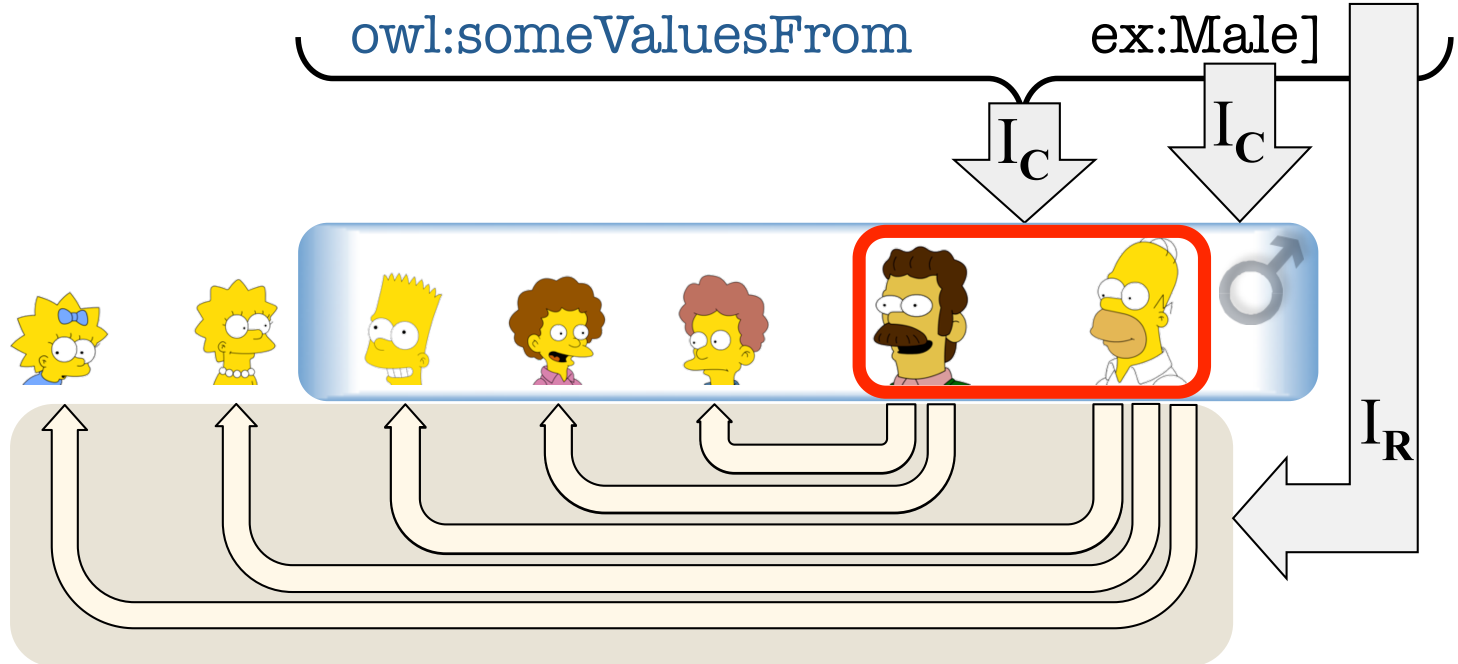
# Complex Classes: Complement

- [owl:complementOf *class*]
- $I_C$([owl:complementOf *class*]) = $\Delta$ - $I_C$(*class*)
- Example:
  [owl:complementOf ex:Politician]



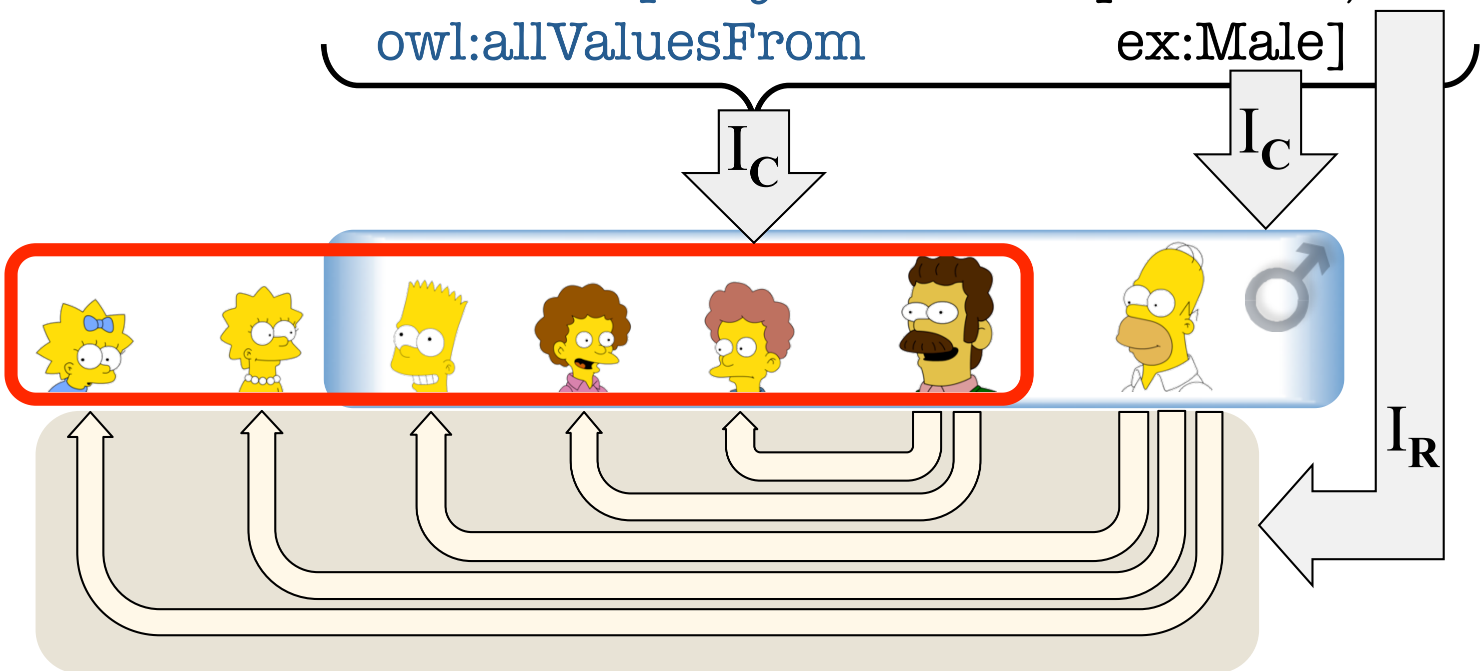
# Complex Classes: Existential Property Restriction

- [ rdf:type owl:Restriction ;
  owl:onProperty *prop* ;
  owl:someValuesFrom *class* ]
- $I_C(\ldots)=\{x \mid \langle x,y\rangle \in I_R(\mathit{prop}) \text{ for some } y \in I_C(\mathit{class})\}$
- Example: [ rdf:type owl:Restriction ;
  owl:onProperty ex:parentOf ;
  owl:someValuesFrom ex:Male]



# Complex Classes: Universal Property Restriction

- [ rdf:type owl:Restriction ;
  owl:onProperty *prop* ;
  owl:allValuesFrom *class* ]
- $I_C(\ldots)=\{x \mid \langle x,y\rangle \in I_R(\textit{prop}) \text{ implies } y \in I_C(\textit{class})\}$
- Example: [ rdf:type owl:Restriction ;
  owl:onProperty ex:parentOf ;
  owl:allValuesFrom ex:Male]



# Syntactic Sugar: Disjointness, Domain & Range Statements

*class1* owl:disjointWith *class2* .

- same as:
  [owl:intersectionOf (*class1* , *class2*)]
  rdfs:subClassOf owl:Nothing .

propuri rdf:domain *class* .

- same as:
  [rdf:type owl:Restriction ;
  owl:onProperty propuri ;
  owl:someValuesFrom owl:Thing ] rdfs:subClassOf *class* .

propuri rdf:range *class* .

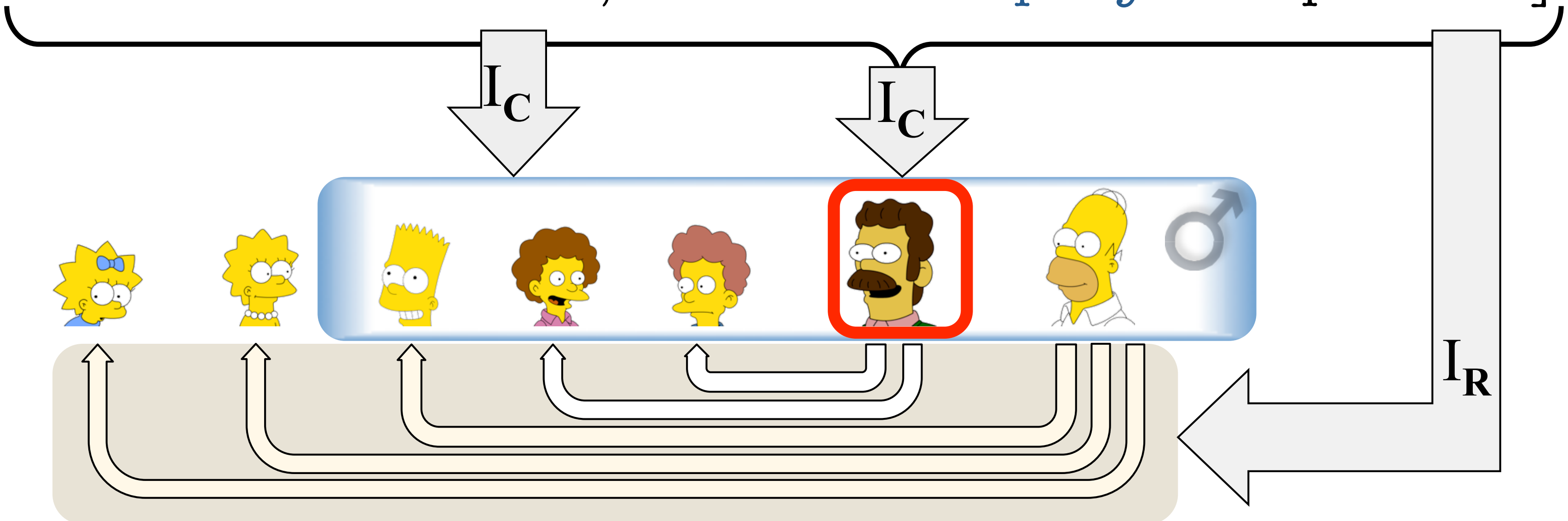
- same as:
  owl:Thing rdfs:subClassOf [rdf:type owl:Restriction ;
  owl:onProperty propuri ;
  owl:allValuesFrom *class* ] .

# Outline

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

# More Complex Classes: Qualified At-Least Restriction

- [ rdf:type owl:Restriction ;
  owl:minQualifiedCardinality "*n*"^^xsd:nonNegativeInteger ;
  owl:onProperty *prop* ; owl:onClass *class* ]
- $I_C(\ldots)=\{x \mid \#\{\langle x,y\rangle \in I_R(\mathit{prop}) \mid y \in I_C(\mathit{class})\} \geq n\}$
- Example:
  [ rdf:type owl:Restriction ; owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;
  owl:onClass ex:Male; owl:onProperty ex:parentOf ]



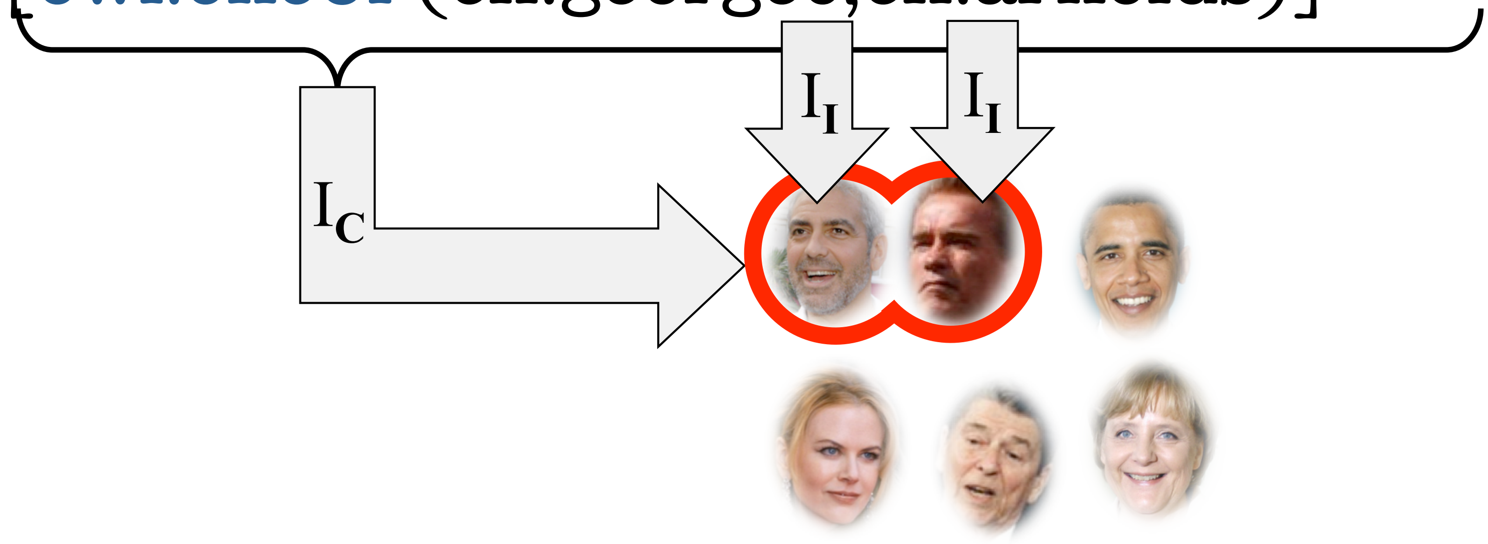
# More Qualified Cardinalities

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

# More Complex Classes: Enumeration of Individuals

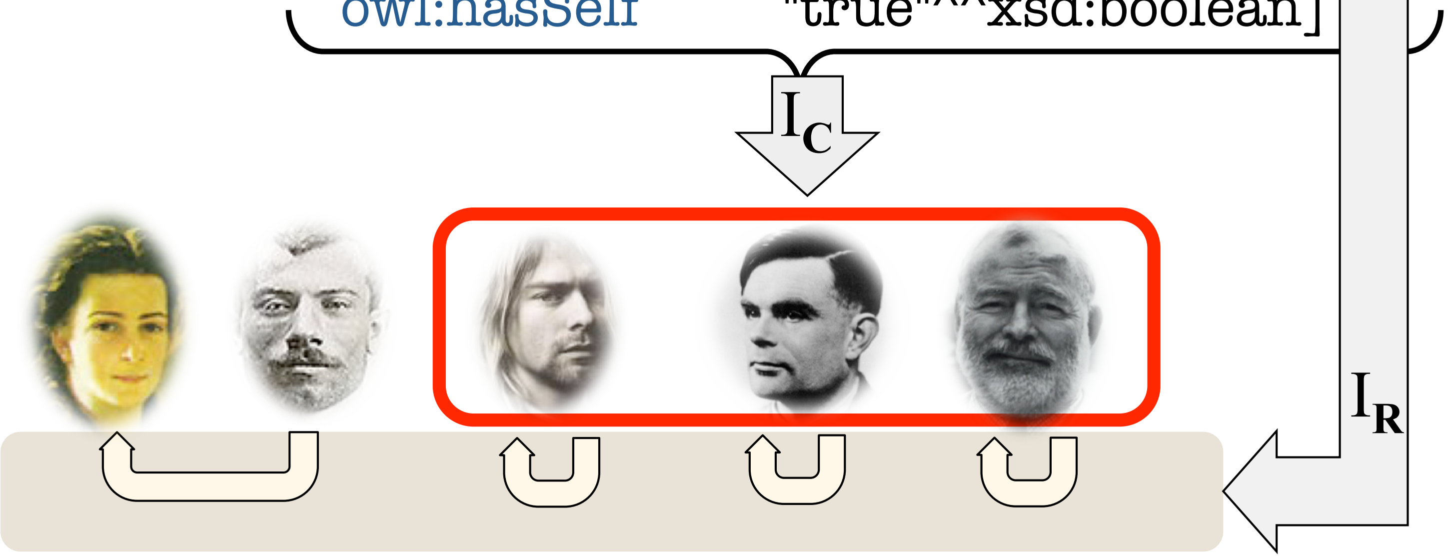
- [owl:oneOf (induri1, ..., indurin)]
- $I_C$([owl:oneOf (induri1, ..., indurin)])
  = {$I_I$(induri1), ... , $I_I$(indurin)}
- Example:

  [owl:oneOf (ex:georgec,ex:arnolds)]



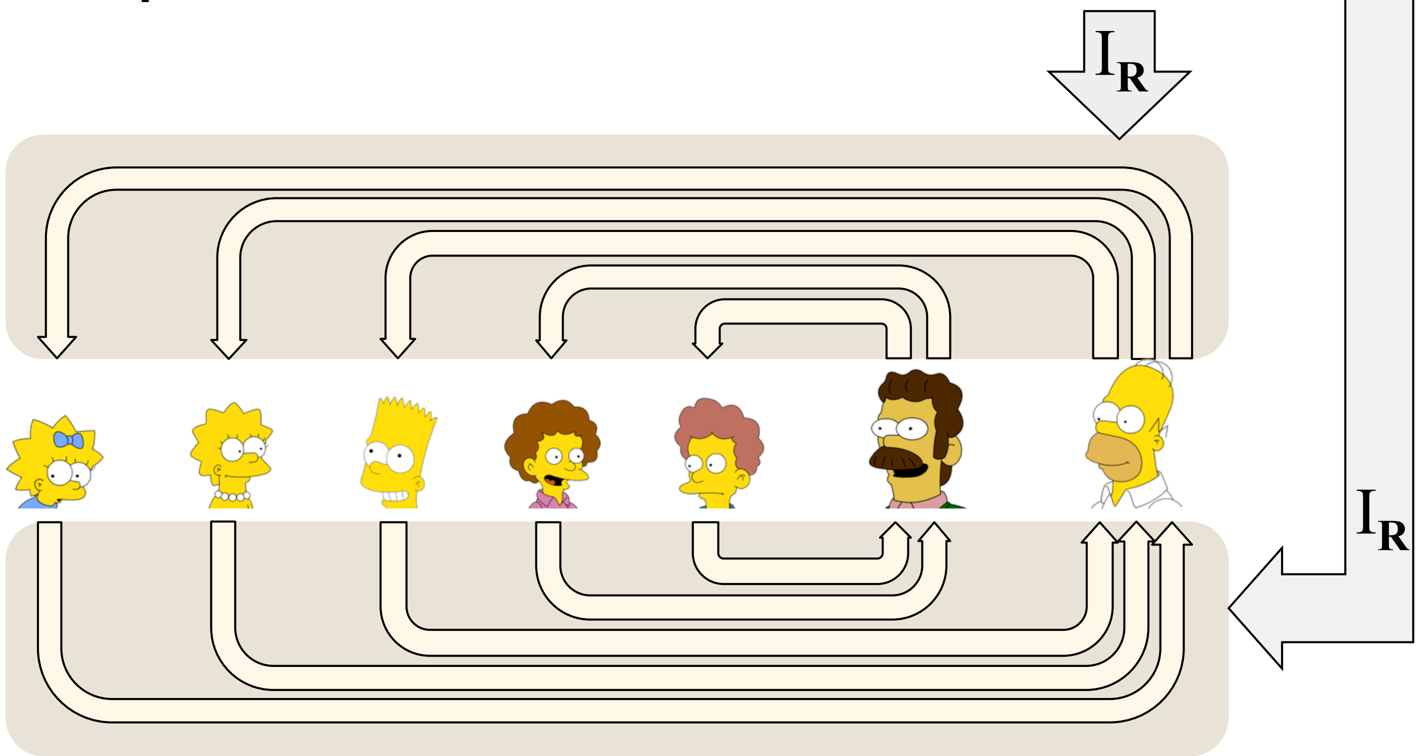
# More Complex Classes: Self Restriction

- [ rdf:type owl:Restriction ;
  owl:onProperty *prop* ;
  owl:hasSelf "true"^^xsd:boolean ]
- $I_C(\ldots)=\{x \mid \langle x,x\rangle \in I_R(\mathit{prop})\}$
- Example: [ rdf:type owl:Restriction ;
  owl:onProperty ex:hasKilled ;
  owl:hasSelf "true"^^xsd:boolean]



# Inverse Properties

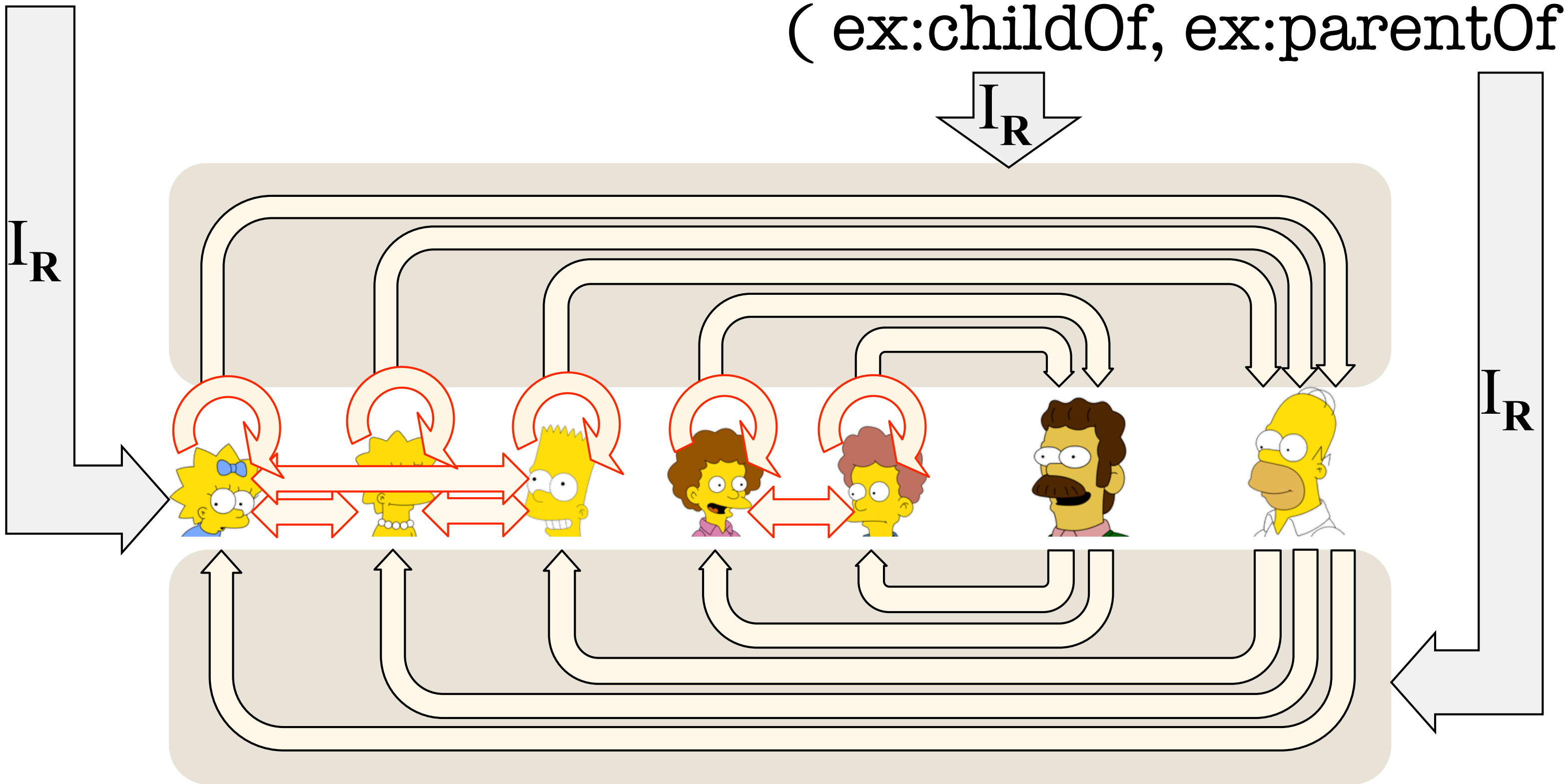
- [owl:inverseOf *prop*]
- $I_R$([owl:inverseOf *prop*])
  $=\{\langle y,x\rangle \mid \langle x,y\rangle \in I_R(prop)\}$
- Example: [owl:inverseOf ex:childOf]



# Property Chain Axioms

- *prop* owl:propertyChainAxiom ( *prop1*, … , *propn* ) .
- true in $\mathcal{I}$, if $I_R(\mathit{prop1}) \circ \ldots \circ I_R(\mathit{propn}) \subseteq I_R(\mathit{prop})$
- Example:

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



# Decidability problems

- property 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
  - 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] $\sqsubseteq R$ $\quad$ $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$

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

- 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: 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:

  $Q \circ P \sqsubseteq R \qquad R \circ P \sqsubseteq R \qquad R \sqsubseteq S \qquad P \sqsubseteq R \qquad Q \sqsubseteq S$

  non-simple: R, S    simple: P, Q

# 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 .

- those *datatype properties* must not be used as individual-interrelating *object properties* at the same time
- datatypes supported by OWL:

  owl:real, owl:rational, xsd:decimal, xsd:integer, xsd:nonNegativeInteger, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:double, xsd:float, xsd:string, xsd:normalizedString, xsd:token, xsd:language, xsd:Name, xsd:NCName, xsd:NMTOKEN, xsd:boolean, xsd:hexBinary, xsd:base64Binary, xsd:anyURI, xsd:dateTime, xsd:dateTimeStamp, rdf:XMLLiteral

# 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 Informatik 2009 participants with the same name and birthday are the same.")

# OWL 2 Keys

OWL 2 provides a way to model
"All Informatik 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!

# 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

Original goal: simpler implementation and usage

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

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

**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

# 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: 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

# OWL 2 QL: Features

- Standard reasoning in OWL 2 QL:
  PTime, instance retrieval even LogSpace (actually $AC_0$) 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

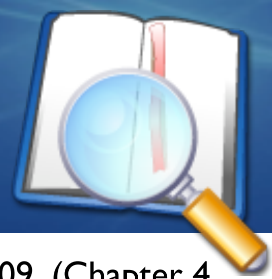
# Summary and Outlook

- 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)