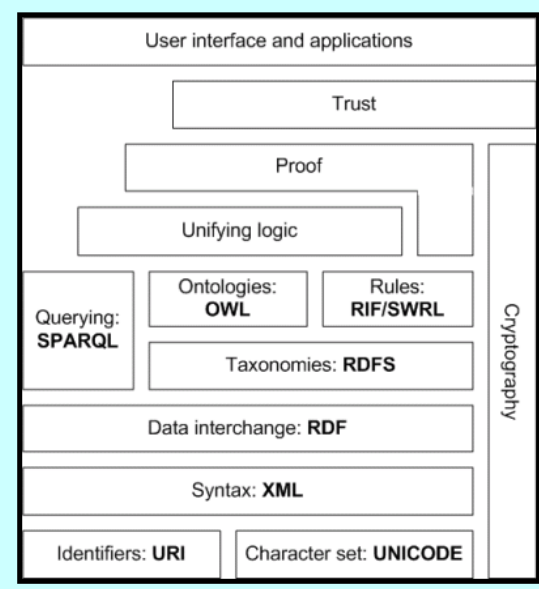


Semantic Web

AI and the future of the World Wide Web



Meaning on the WWW

Logic Protege XML-Schema Meaning RDF-Schema XML

OWL

RDF

Language

Ontologies

Taxonomies

Rules

Annotation

Queries

Starting Point: The Un-Semantic Web

The World-Wide-Web (WWW) is a dense network of interconnected texts / resources thru which data can be linked presented, tagged and retrieved

Key Elements of the current WWW:

- 1. Hypertext document structure (non-linear document structure)**
- 2. Integration of data (text/images/sources) from multiple sources**
- 3. Unique identification of resources (URIs and URLs)**
- 4. Linking of documents/resources across different hosts**
- 5. Mark-up of content for structural / presentational purposes (HTML)**

What's Wrong with the Current Web

The WWW comprises a network of interconnected documents and data

Web documents (and presentation of data) are mostly organized via HTML

BUT, HTML attempts to provide two orthogonal services at once:

1. To Mark-up the presentational appearance/structure of a document
(e.g., what's the title, where are the breaks, what are list items)
2. To Mark-up the information content of a document
(e.g., via tables of rows/columns with headings, values) lists, etc.

These are irreconcilable requirements of a simple mark-up language

Meaning (semantics) inevitably takes a back seat to form (presentation)

Meta Tags

One solution is to use a special tag for the semantic content in a web page.

The META tag in HTML:

- The `<meta>` element provides non-presentational-information about a web page, such as descriptions and keywords of its content
- These keywords/descriptions are primarily aimed at search engines

```
<meta name="description" content="Advanced Software Eng" />
```

```
<meta name="keywords" content="Web, AI, Semantics, Logic" />
```

```
<meta http-equiv="refresh" content="5" />
```

Can these be trusted?



Separating Style from Content

This would allow a machine to obtain some hint at the meaning of a document element from how it is marked-up.

CSS: Cascading Style Sheets

The content of the web document is marked up in XHTML or XML

The presentational style of this content is provided by a CSS file

XML can be used to represent the logical structure of the document

Appropriate XML tags can be used to mark the semantic-status of entities

The XML structure (not the CSS) can be used for semantic processing

```
h1 { color: white;
background: orange;
border: 1px solid black;
padding: 0 0 0 0;
font-weight: bold;
}
/* begin: seaside-theme */

body {
background-color: white;
color: black;
font-family: Arial, sans-serif;
margin: 0 4px 0 0;
border: 12px solid;
}
```

Going Further: The Semantic Web

The content of Web is largely built from Natural Language, requiring human-level intelligence (AI, NLP) to *generate, interpret and reason about*

Even non-linguistic content requires human intelligence to understand:

Numbers in table cells may be ages, prices, quantities or spatial dimensions

Proper names may be the names of People or the names of Places

Non-verbal text strings may be product labels, machine codes, etc.

A price may be a sale price, a purchase price, a bid-price, an offer price, ...

We therefore need to tag information items with meanings, not keywords

We thus need a language of meanings (*unambiguous logical vocabulary*)

Keywords **versus** Meanings

Simple keywords alone will not suffice, for the following reasons:

Keywords are linguistic terms (*words, phrases*) and may be ambiguous

Even unambiguous terms have different meanings in different contexts

Different producers/consumers may interpret the same term differently

Complex meanings can require complex forms, with no unique phrasing

Compare keywords to mark content (in XML) with consensus meanings:

```
<item>box</item>
```

```
<item rdf:about="http://dbpedia.org/resource/Box">box</item>
```

What's the Difference? **Meanings** versus **Keywords**

Remember how an Ontology is defined in Computer Science / AI:

Gruber: “The Formal Specification of a Conceptualization.”

Guarino: “An engineering artifact, constituted by a specific vocabulary, that describes a certain reality.”

Keywords are neither formal, specific, or part of an engineered solution:

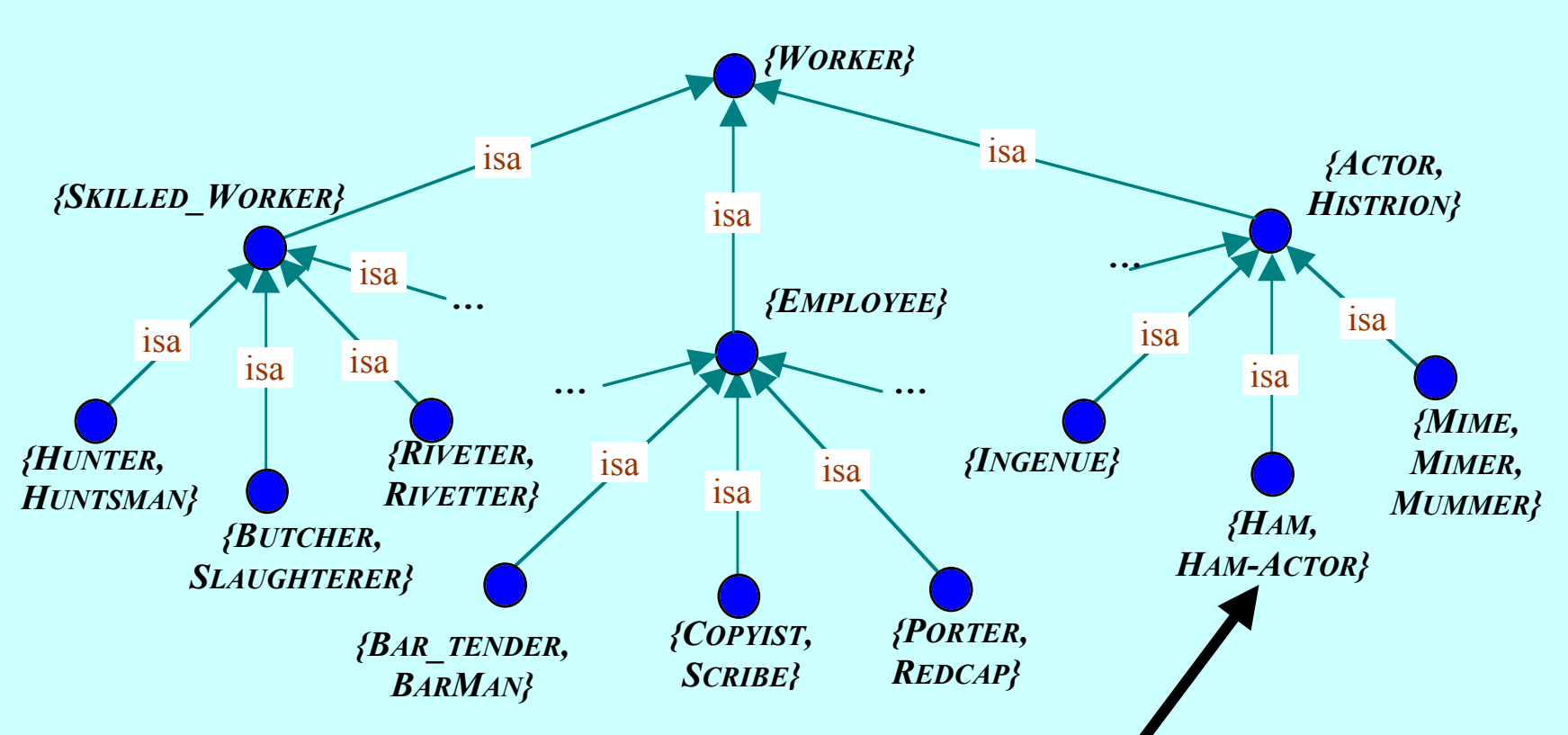
```
<item>box</item>
```

Ontological terms are formally grounded in a shared specification of reality:

```
<item rdf:about="http://dbpedia.org/resource/Box">box</item>
```

Here the term “box” means what the dbpedia specifies it to mean, and nothing else. To reason about this “box”, a machine must do so using this ontology

Lexical **Ontologies**: WordNet on the Semantic Web



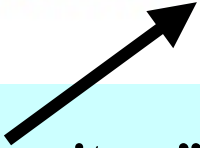
```
<item      rdf:about = "http://www.w3.org/2006/03/wn/wn20/instances/
                                     wordsense-ham-noun-4">
  ham
</item>
```

Senses on the Semantic Web: WordNet RDF Schema

Princeton WordNet is represented with its own RDF schema on the web

Documents can unambiguously use specific word senses via this namespace

```
<rdf:RDF>
- <rdf:Description
  rdf:about="http://www.w3.org/2006/03/wn/wn20/instances/wordsense-ham-noun-4">
    <wn20schema:word
      rdf:resource="http://www.w3.org/2006/03/wn/wn20/instances/word-ham"/>
    <wn20schema:tagCount>0</wn20schema:tagCount>
    <rdfs:label>ham</rdfs:label>
    <rdf:type
      rdf:resource="http://www.w3.org/2006/03/wn/wn20/schema/NounWordSense"/>
    <wn20schema:derivationallyRelated
      rdf:resource="http://www.w3.org/2006/03/wn/wn20/instances/wordsense-ham-verb-1"/>
    </rdf:Description>
</rdf:RDF>
```



Note the link to the schema instance for verb sense of “to ham it up”

Structure of the Semantic Web: Semantic Web Stack

This is the *classic* W3C blueprint

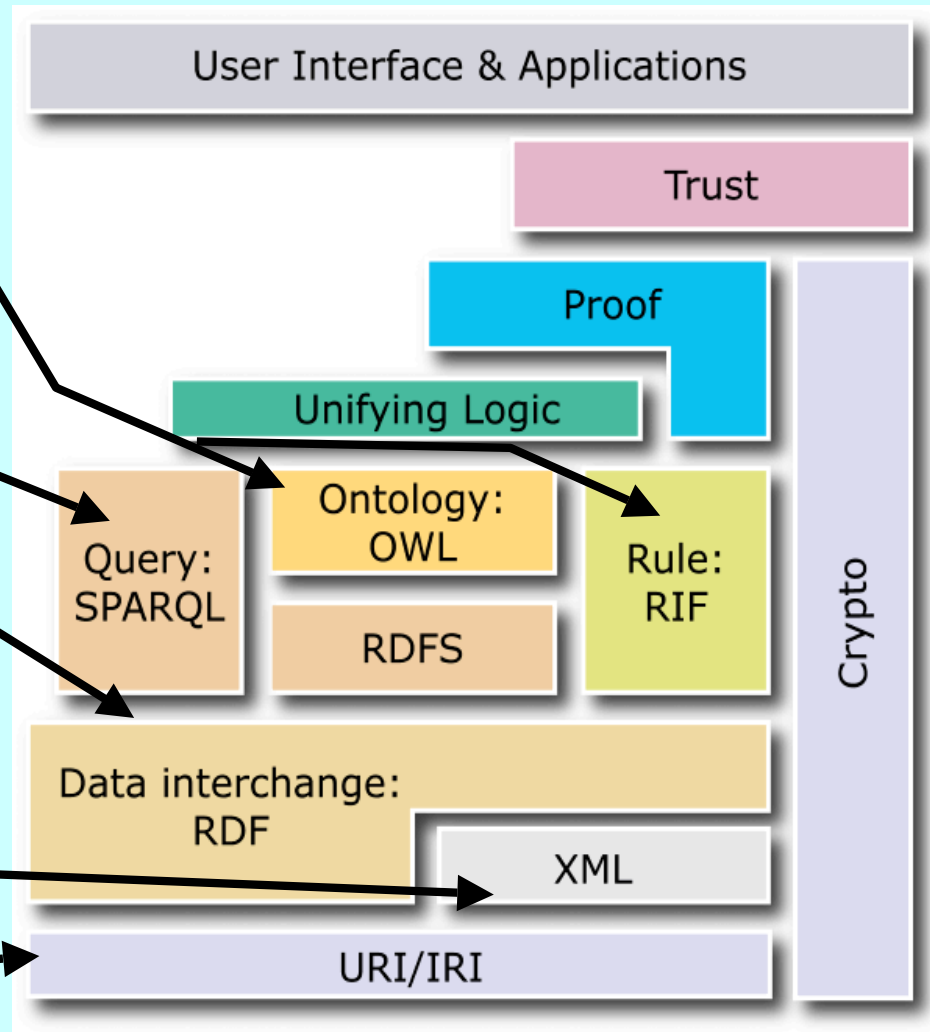
Note the *central* role of ontologies

Support for “Semantic” queries:

A data-model for this structure:

Providing structure to content:

The foundational WWW:



Levels of the Semantic Web Stack #1: XML

XML : eXtensible Mark-up Language

A nested-tagging system for document content that employs user-defined tags

```
<?xml version="1.0"
<quiz>
  <question>
    who was the forty-second
    president of the U.S.A.?
  </question>
  <answer>
    William Jefferson Clinton
  </answer>
<!-- Note: We need to add
      more questions later.-->
</quiz>
```

XML is intended to capture the logical structure of a document / datastore

An XML document may obey a DTD (Document Type Definition)

A DTD defines the allowable structure of an XML document (a schema)

XML documents may be Well-Formed and/or Valid (*conforming to DTD*)

XML provides syntax for content of documents on the Semantic Web

At this level of the stack, XML imposes just structure, not meaning!

Meaning is imposed/associated at successively higher levels of the stack.

Levels of the Semantic Web Stack #1.5: XML-Schema

A Schema is a Frame Structure that imposes *structure/order* on experience

XML-Schema shapes and restricts the data that can appear in XML docs

A DTD is a special schema that serves as a grammar for an XML doc

XMLS allows us to define:

C-style Structs

Java-like Objects

AI-like Frames

For Example:

```
<!ELEMENT ADDRESSBOOK (CONTACT)>
<!ELEMENT CONTACT (NAME, ADDRESS,
CITY, PIN, PHONE)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT ADDRESS (#PCDATA)>
<!ELEMENT CITY (#PCDATA)>
<!ELEMENT PIN (#PCDATA)>
<!ELEMENT PHONE (#PCDATA)>
```

Well, in form, at least!

*Inference, process-triggering,
or rule-firing with XML
schemas requires another
level of representation.*

Levels of the Semantic Web Stack #2: RDF

XML provides *a way to structure data*, but doesn't provide a data-model

The same content might be modeled very differently by different people.

RDF (*Resource Description Framework*) provides a specific data-model.

RDF contains statements that describe resources, often named via an URI

RDF statements use triples to capture *subject-predicate-object* relationships

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
    <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Barack_Obama">
      <dc:title>Barack Obama</dc:title>
      <dc:publisher>Wikipedia</dc:publisher>
    </rdf:Description>
</rdf:RDF>
```

<Article on Barack Obama, title, Barack Obama>

<Article on Barack Obama, publisher, Wikipedia>

Semantic Web Stack #2: **RDF** Triple-Stores

RDF triples are like Frame slots, providing *<frame, slot, filler>* data

RDF triples are like binary logical axioms of form *pred(subject, object)*

Triples form a semantic network, *linking subjects to objects via predicates*.

A complete RDF knowledge-base is thus *known as/stored as* a **Triple-Store**

A **Triple Store** is a specialized relational database, optimized for triples.

A special Query Language is used to *query / retrieve* triples from the store.

SPARQL (Simple Protocol and RDF Query Language), *pron.* “sparkle”

SPARQL has an SQL-like syntax for selecting data from triple-stores:

E.g., `Select ?var Where { ?subject relation ?var }`

Levels of the Semantic Web Stack #3: RDFS

XML is a *data-tagging* language, RDF is a *data-modeling* language,

RDFS (RDF-Schema) is a true Knowledge-Representation language

RDFS uses RDF (& *schemas*) to provide the basic elements of an ontology

RDFS allows the creation of Classes of entities and Subclasses of Classes

E.g., the *FOAF namespace* defines the class foaf:Person

We can define new subclasses of Person, such as Politicians, as follows:

```
ex:Politician rdfs:subClassOf foaf:Person
```

We can define Bertie as an instance of this class as follows:

```
ex:Bertie rdf:type foaf:Person
```


Namespaces and classes in RDFS

namespace: **foaf**

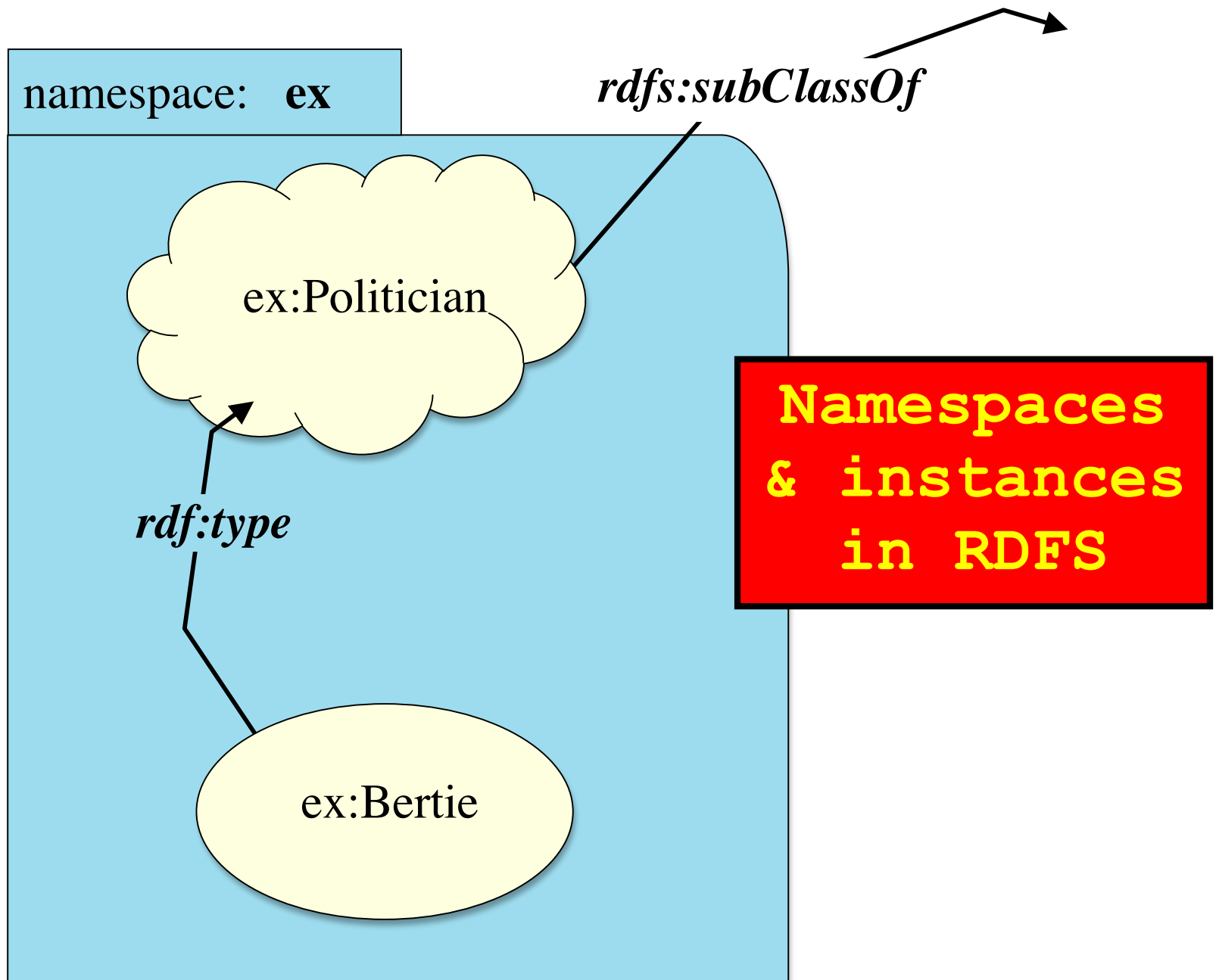
foaf:Person

namespace: **ex**

ex:Politician

rdfs:subClassOf

The diagram illustrates a subClassOf relationship in RDFS. It features two namespace containers. The top container, labeled 'namespace: foaf', contains a yellow cloud labeled 'foaf:Person'. The bottom container, labeled 'namespace: ex', contains a yellow cloud labeled 'ex:Politician'. An arrow points from the 'ex:Politician' cloud to the 'foaf:Person' cloud, with the label 'rdfs:subClassOf' written above the arrow.



Semantic Web Stack #3: **RDFS** Properties

RDFS allows us to place semantic constraints on the elements of a relation

Every RDF triple links a subject (the *domain*) to an object (the *range*)

We can *constrain* the domain and range of the relation in a triple thusly:

`ex:votesFor rdfs:domain foaf:Person`

`ex:votesFor rdfs:range ex:Politician`

This means that the *votesFor* relation can only link Persons to Politicians

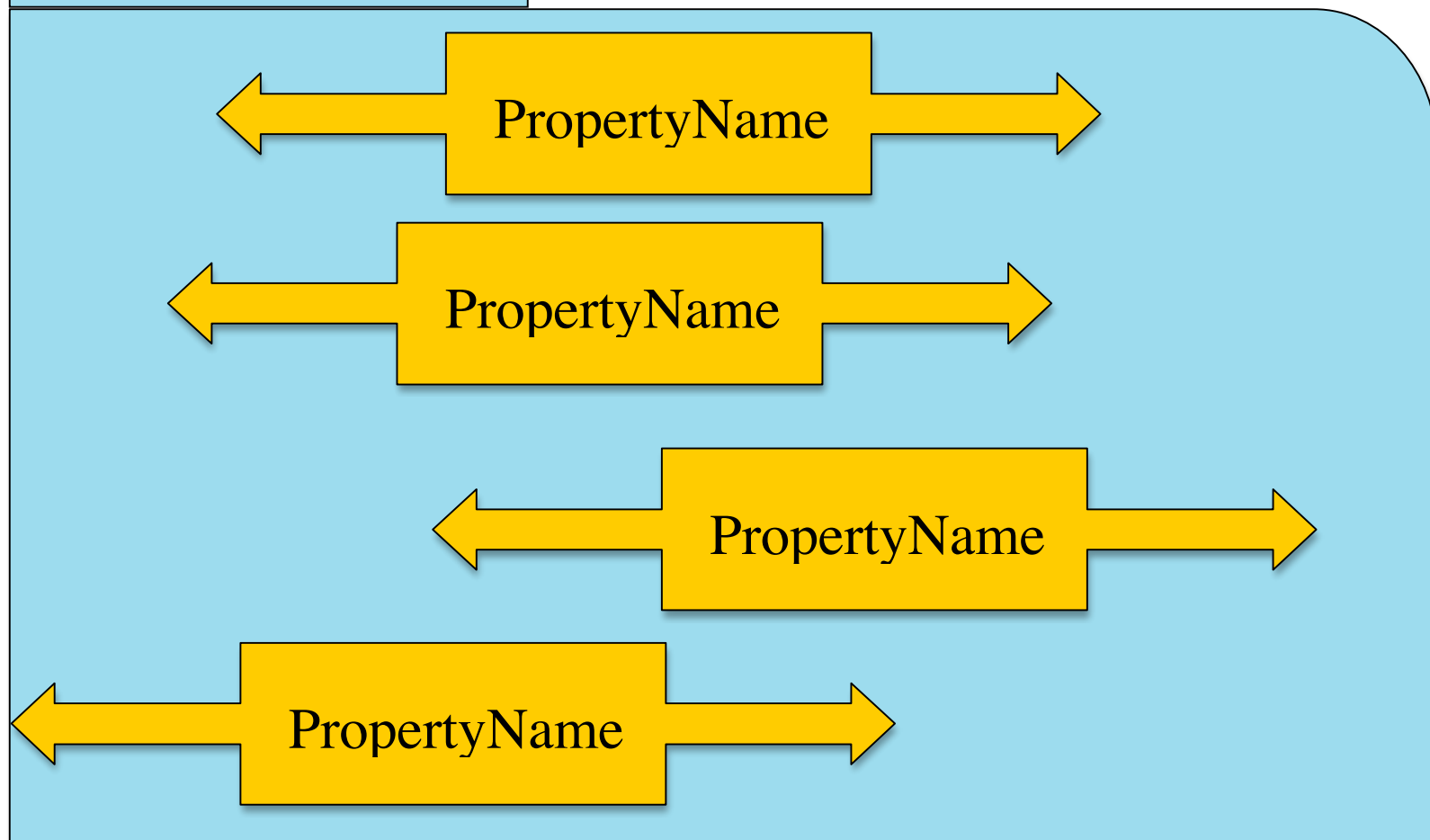
E.g., `ex:John ex:votesFor ex:Bertie`

Notice how these semantic constraints are themselves expressed as triples

Notice also that the domain and range of a property is specified as a class

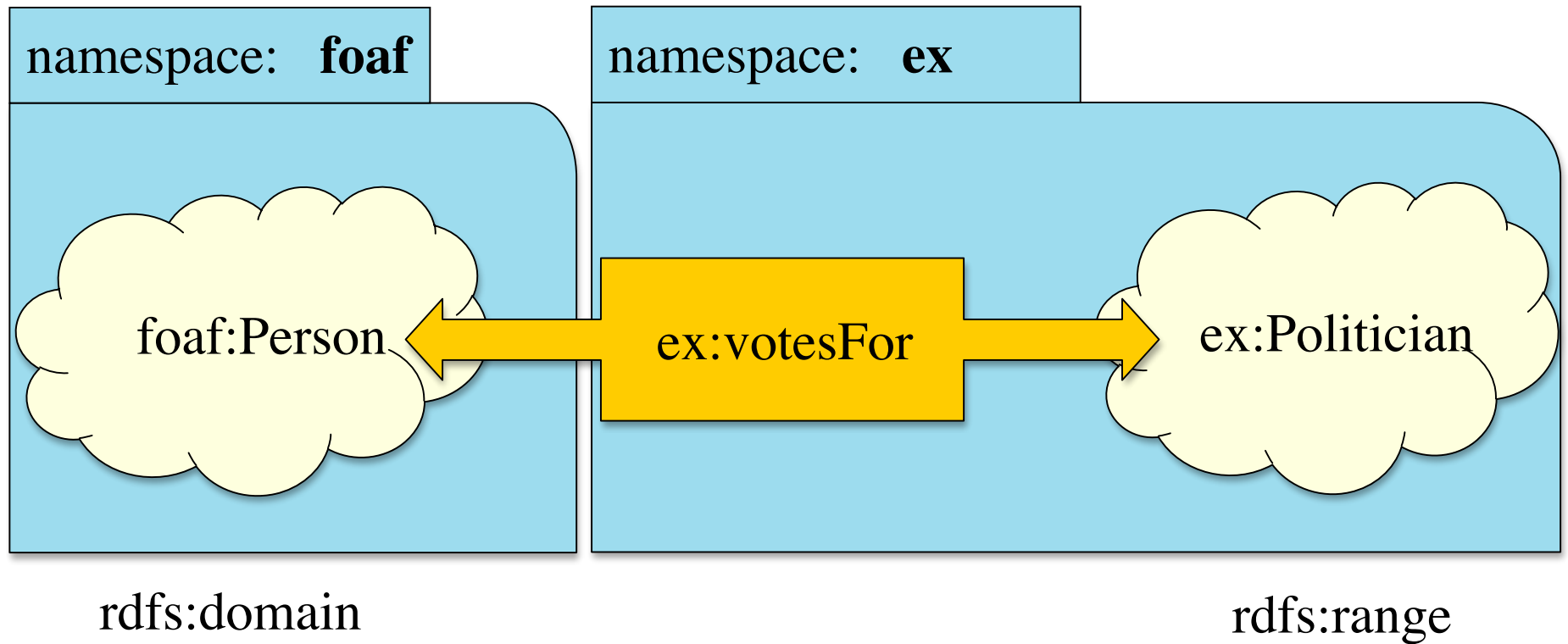
Properties in RDFS

namespace

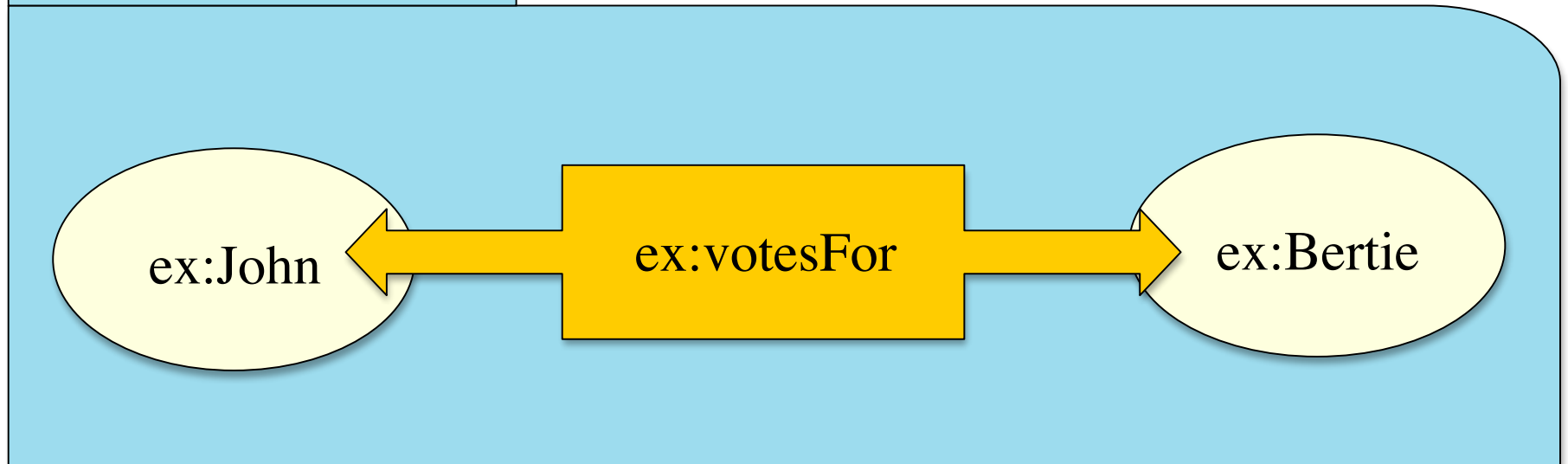


Properties in RDFS





namespace: **ex**



rdfs:domain foaf:Person

rdfs:range ex:Politician

Levels of the Semantic Web Stack #4: OWL

OWL (Web Ontology Language) is actually a *family* of ontology languages

OWL DL (Description Language) based on *AI terminology logics*

OWL lite lightweight DL language, computationally efficient

OWL full (compatible with RDF Schema)

An OWL ontology comprises axioms about classes and relations

Axioms allow a machine to infer facts that are explicitly stated in system

Some axioms require a linking or chaining of properties via SWRL:

X hasParent Y and Y hasBrother Z then X hasUncle Z

X hasParent Y and Y hasSister Z then X hasAunt Z

X hasChild Y and Y hasChild Z then X hasGrandchild Y

Levels of the Semantic Web Stack #4: SPARQL

SPARQL is a query language for finding matches in RDF *triple-stores*

OWL ontologies stored in RDF format can be *queried* using SPARQL

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital;
    abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
    abc:isInContinent abc:Africa .
}
```

Joins two RDF relation types:

<City isCapitalOf Country>

<Country isInContinent Continent>

Retrieve all matches for Capitals of African Countries (?capital, ?country)

Layers of Form and Meaning: A Summary

XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.

XML Schema is a language for restricting the structure of XML documents and also extends XML with **datatypes**.

RDF is a **datamodel** for objects ("resources") and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax.

RDF Schema is a vocabulary for describing properties and classes of RDF resources, with semantics for hierarchies of **properties** and **classes**.

OWL adds more vocabulary for describing properties and classes: e.g., *relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.*

Web Ontology Language: What do we Want and Need?

Computational logics / reasoning frameworks are judged by the following:

Soundness: *the system does not and cannot contain a contradiction*

Completeness: *the system guarantees all true conclusions to be provable*

Computational Completeness: *all true conclusions guaranteed computable*

Decidability: *all proofs / computations guaranteed to finish in finite time*

A System will also be judged by the following *pragmatic* considerations:

Practicality: *how practical is the system to use and maintain?*

Efficiency: *is the system computationally efficient (fast, low overhead)?*

Web Ontology Languages: The OWL Hierarchy

OWL-lite, OWL-DL and OWL-full form a hierarchy of expressiveness:

[Simplicity: \rightarrow *OWL-lite* \rightarrow *OWL-DL* \rightarrow *OWL-full* \rightarrow :Complexity]

Every legal OWL-lite ontology is a legal OWL-DL ontology.

Every legal OWL-DL ontology is a legal OWL-full ontology.

Every valid OWL-lite conclusion is a valid OWL-DL conclusion.

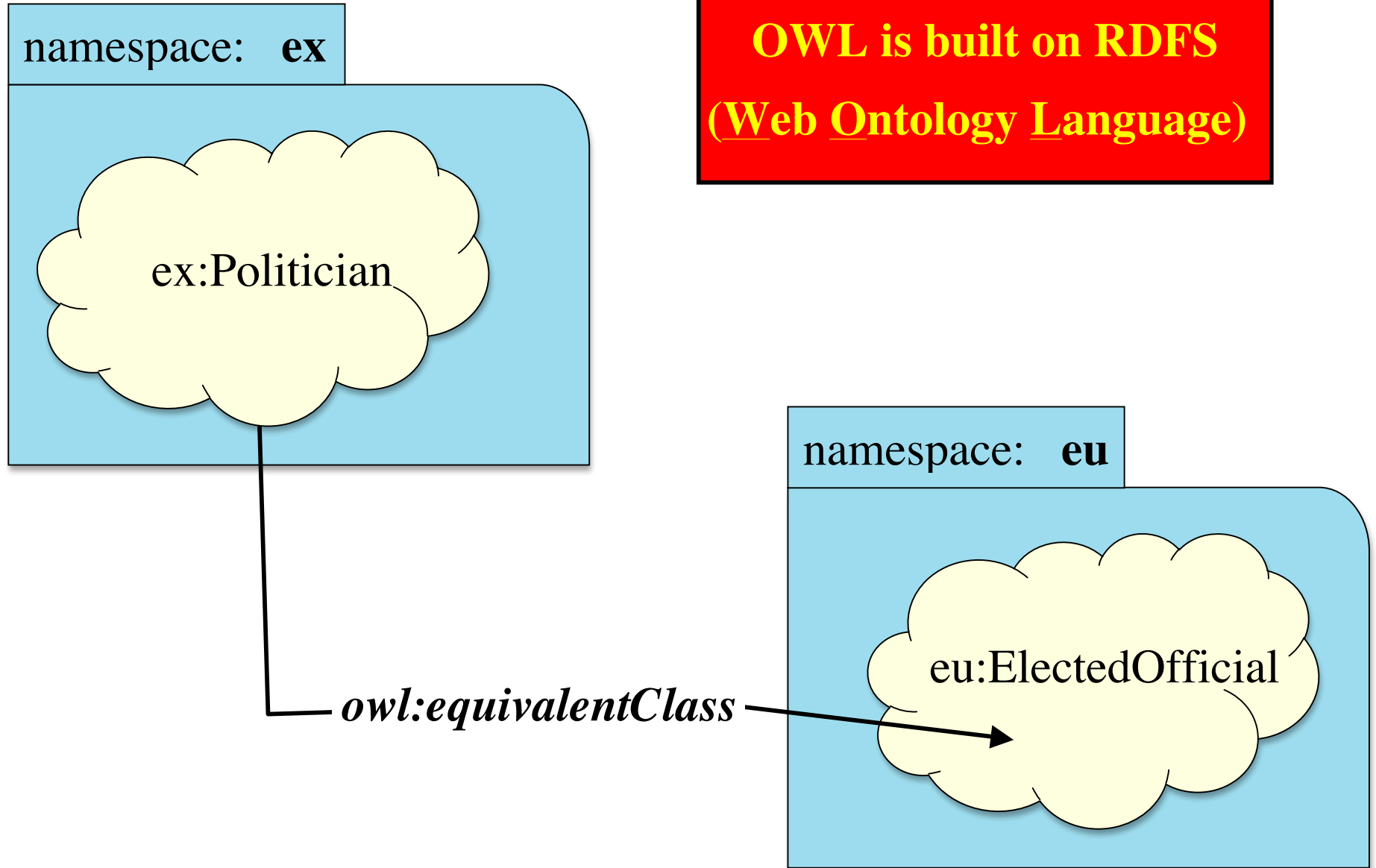
Every valid OWL-DL conclusion is a valid OWL-full conclusion.

Ontologists who want simple taxonomies / thesauri choose OWL-lite.

Ontologists who want more complexity (+ efficiency) choose OWL-DL

Ontologists who want maximal expressiveness choose OWL-full

OWL is built on RDFS
(Web Ontology Language)



namespace: **ex**

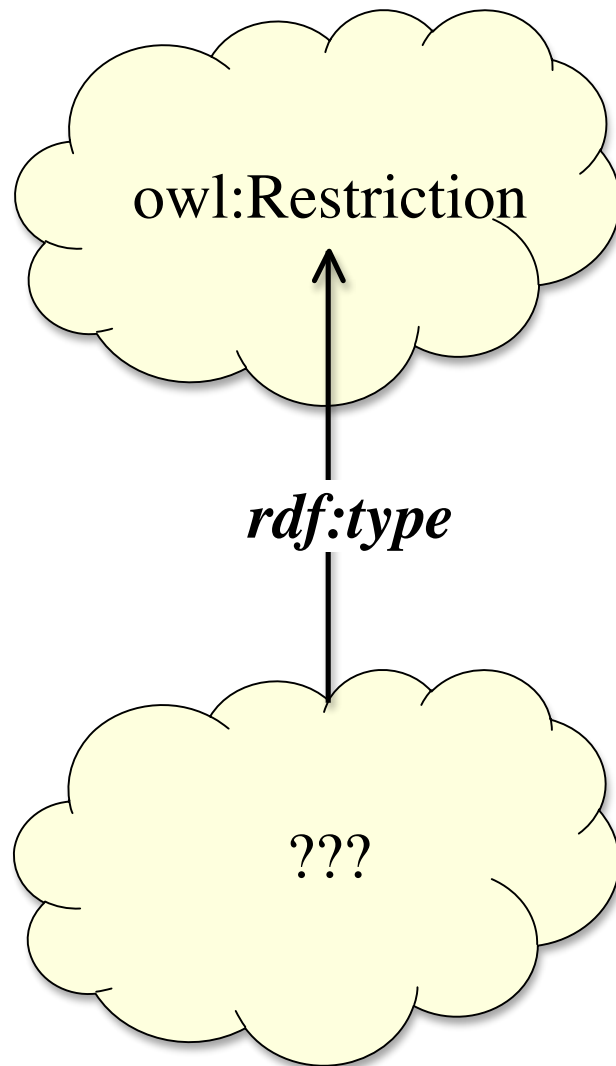
**OWL classes can be disjoint
(no common instances, ever)**

ex:Priest

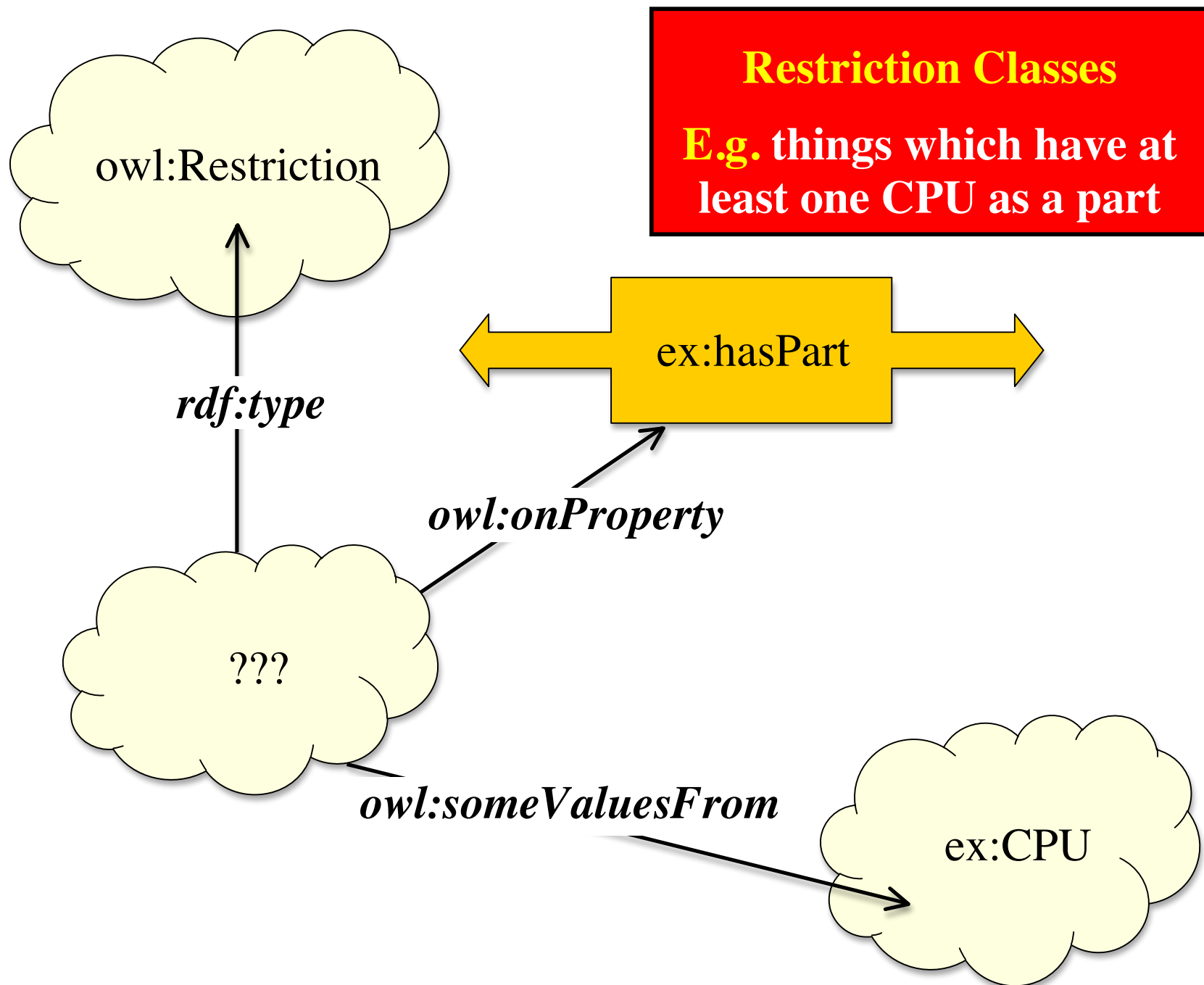
owl:disjointWith

ex:Woman

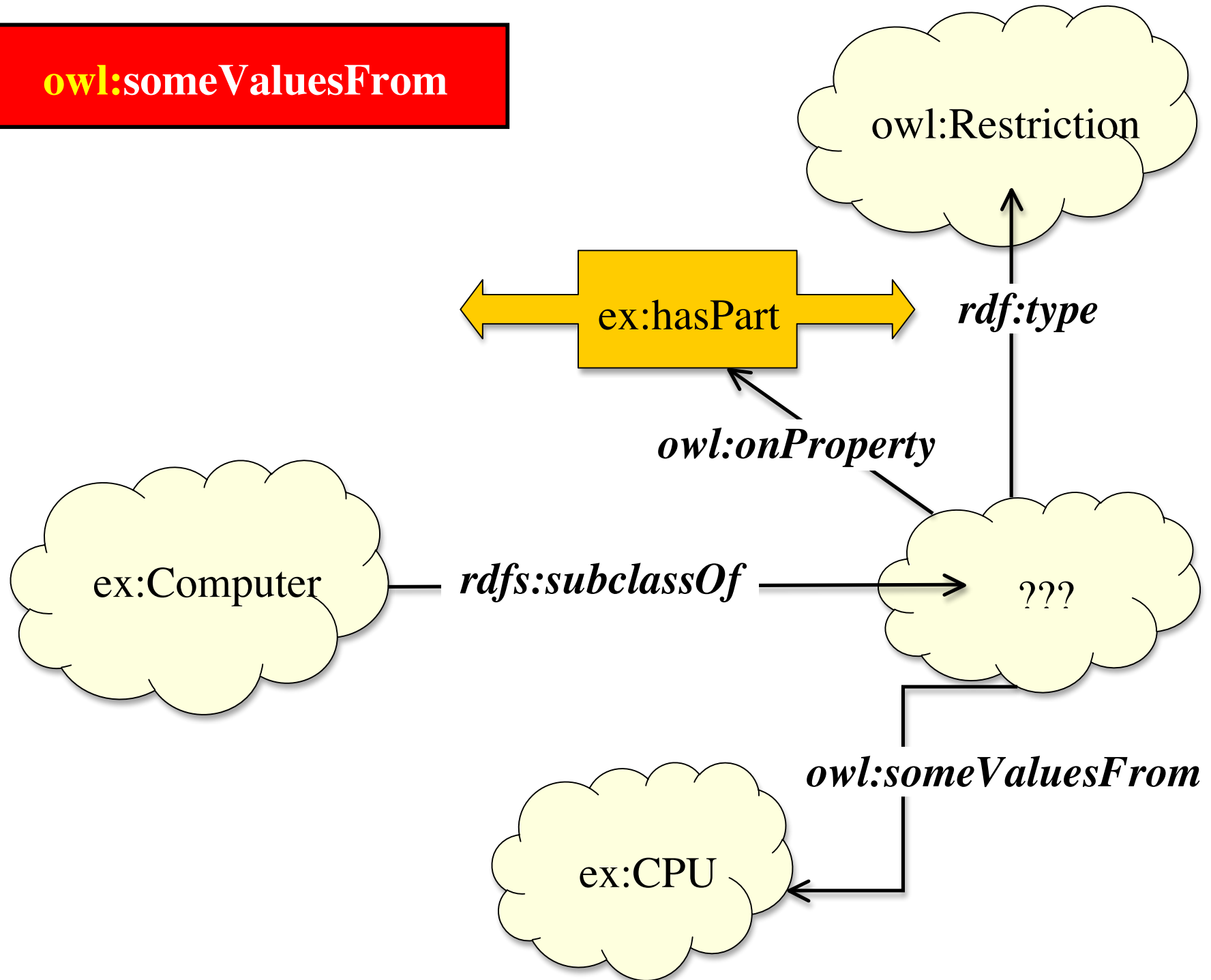
The diagram illustrates a disjoint relationship between two OWL classes. A light blue rectangular frame contains two yellow cloud shapes. The left cloud is labeled 'ex:Priest' and the right cloud is labeled 'ex:Woman'. A black line connects the bottom of the 'ex:Priest' cloud to the 'ex:Woman' cloud, with the label 'owl:disjointWith' written above the line. In the top right corner of the frame, a red box contains the text 'OWL classes can be disjoint (no common instances, ever)' in yellow. In the top left corner, a small blue box contains the text 'namespace: ex'.



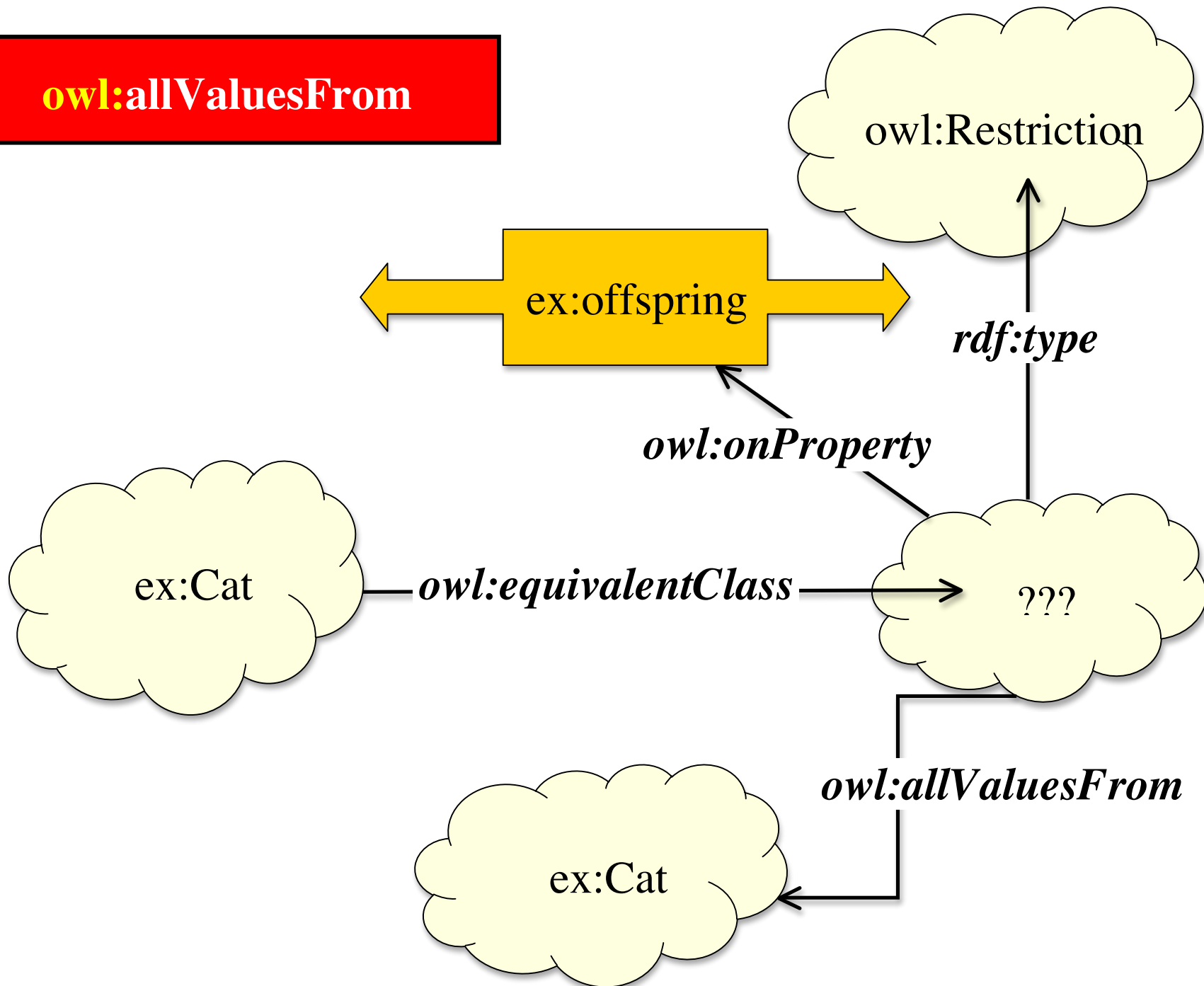
**OWL supports
Anonymous Classes**
(Classes that we don't need
to explicitly name)



owl:someValuesFrom



owl:allValuesFrom



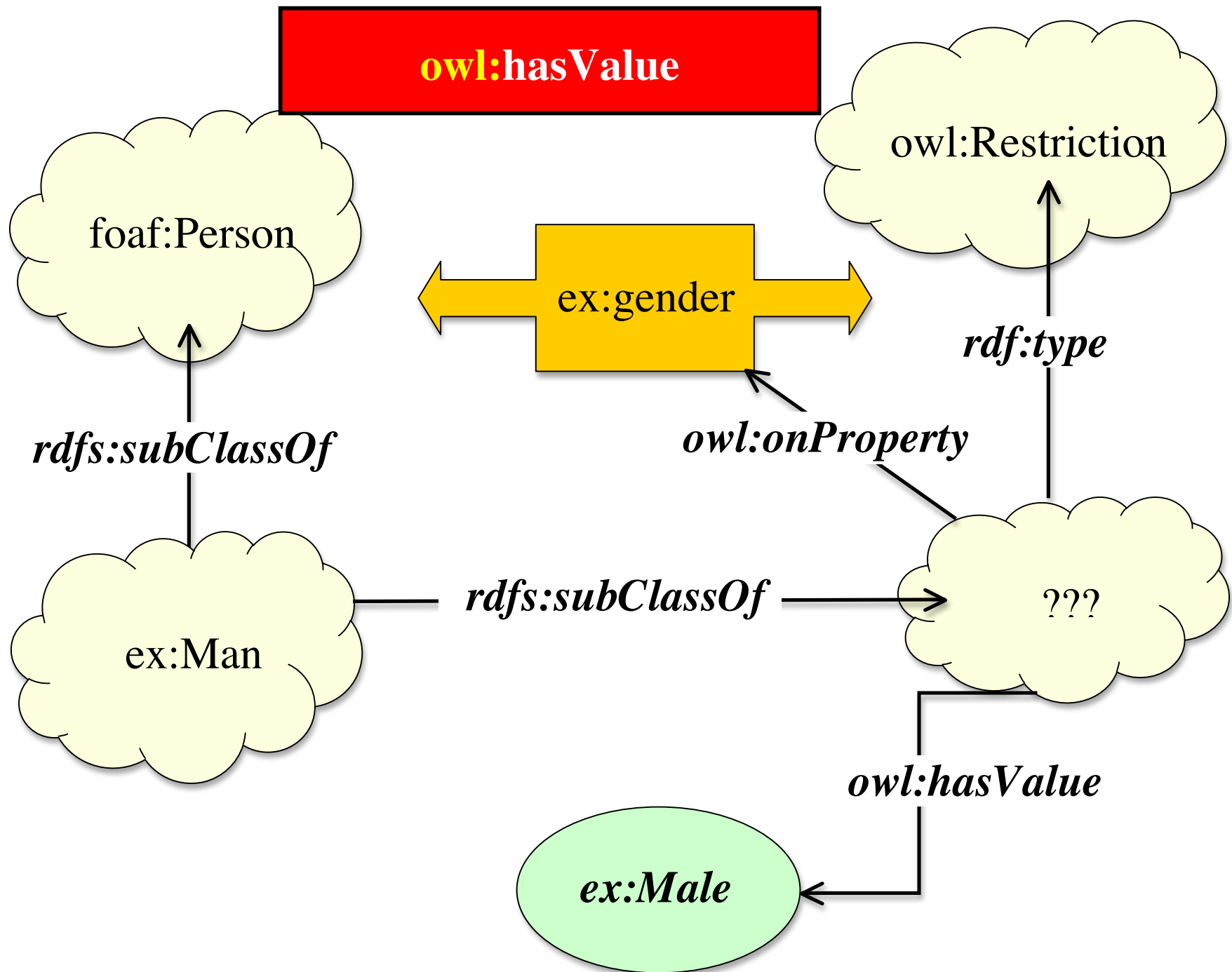
Web Ontology Language: Constraining Properties

OWL allows an engineer to constrain the range of a property in specific contexts. Use *owl:onProperty* to specific the restricted property.

OWL has two *useful but limited* property constraints. In each case, a constraint applies to a property as it is used in a given class or subclasses:

allValuesFrom: a property is constrained, when used with a given class or its subclasses, to a given range of classes. E.g., in the class Person the *hasChild* property is limited so all values are Person also; in the class Cat, the *hasChild* property is limited so all values are Cats also.

someValuesFrom: a property is constrained, when used in a given class, so that it has at least one value from the specified range. E.g., in the class Computer, *hasPart* is limited so that instances have at least one value of *hasPart* that is an instance of the class CPU.



Reasoning Systems: Open and Closed Worlds

What happens if we cannot prove that a claim is either true or false?

Suppose we ask a system whether Mick Fassbender is Irish

Also suppose the system has insufficient knowledge to decide either way?

The Closed World Assumption

(used in Prolog)

1. Assume that the system knows or can prove all true facts.
2. By 1, failure to prove a claim must mean it is false (*Mick is not Irish*)

The Open World Assumption

(used in Description Logic)

1. Do not assume that the system knows or can prove all true facts.
2. By 1, failure to prove a claim does not imply false (*Mick may be Irish*)

Web Ontology Languages: Unique Names Assumption

The Unique Names Assumption: “if two individuals in a logical system have distinct names, then they are assumed to be distinct individuals”

OWL does NOT apply the Unique Names Assumption

Making claims about Classes and Individuals in OWL:

equivalentClass: two classes are equivalent, and have the same instances

disjointWith: two classes are mutually exclusive, no common instances

equivalentProperty: two properties are equivalent, and thus *synonymous*

sameAs: two individuals are actually the same individual

differentFrom: converse of *sameAs*, explicitly state individuals are distinct

Web Ontology Languages: Cardinality

In Mathematics, *Cardinality* refers to the size of a set.

In OWL, *Cardinality* refers to the number of different values that a property can validly be assigned. *Cardinality* can vary from class to class.

E.g., Cardinality constraints on properties are constrained by:

maxCardinality – the maximum number of values that can be assigned

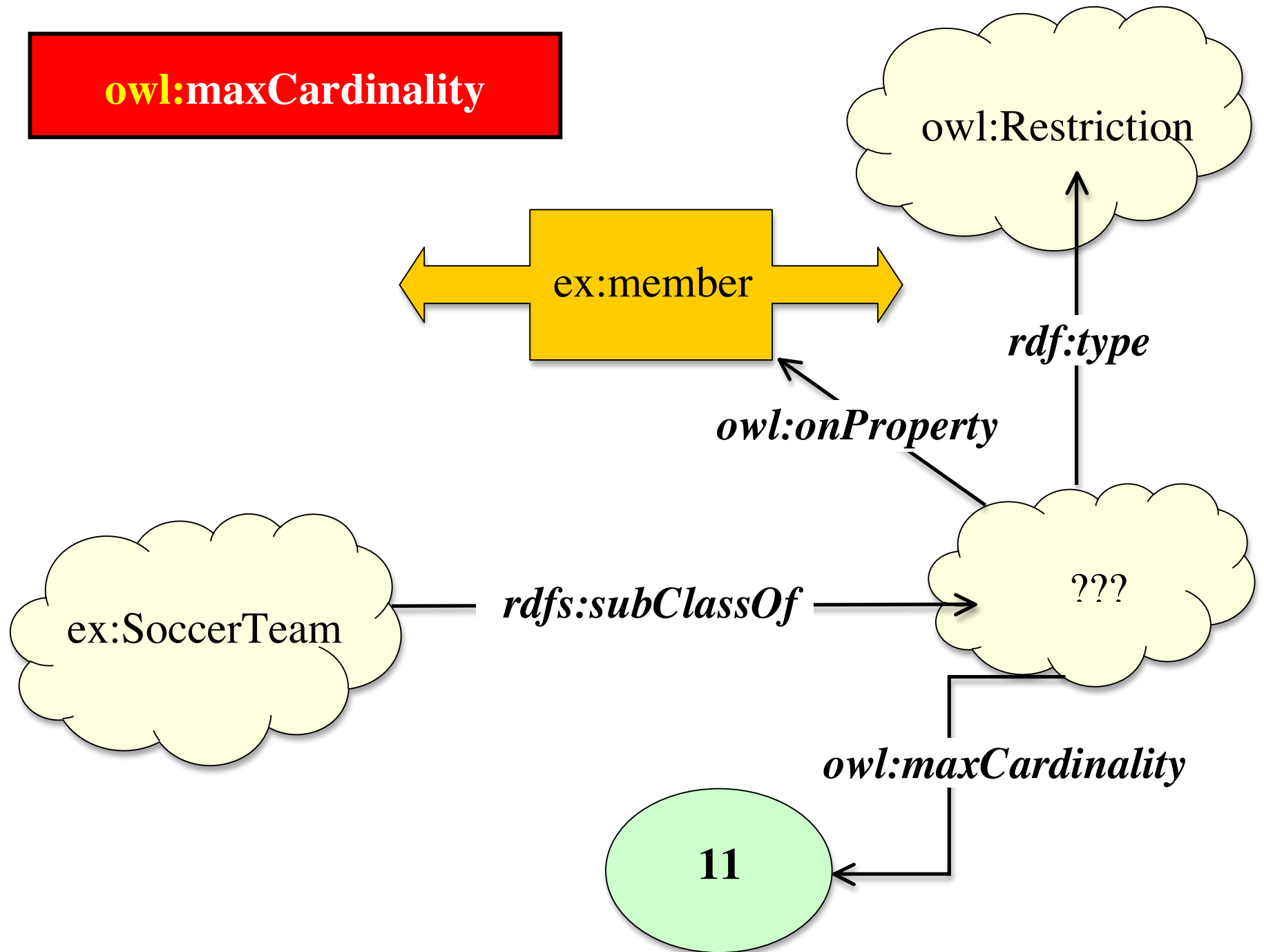
minCardinality – the minimum number of values that can be assigned

When *maxCardinality* = *minCardinality* then no wiggle room!

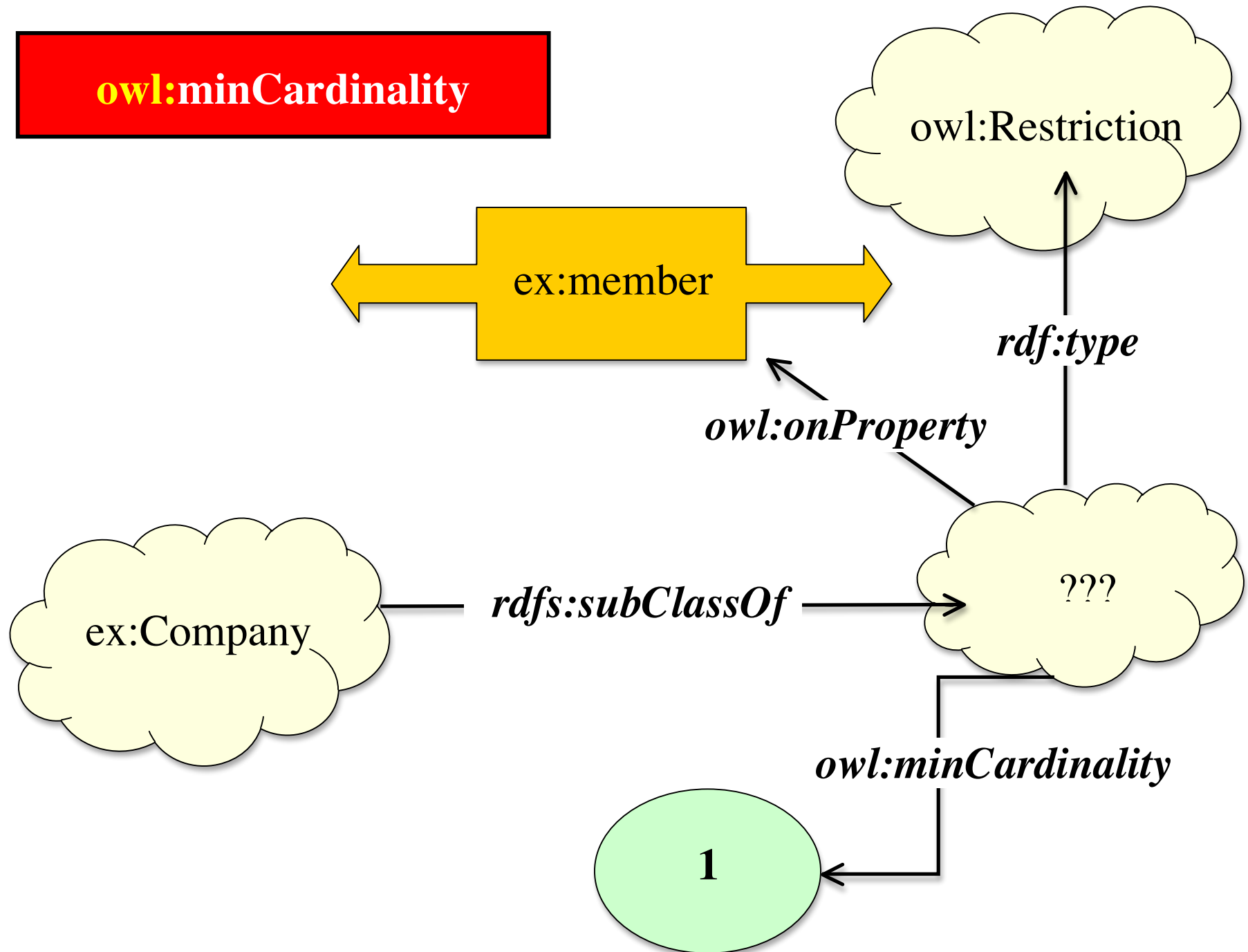
Cardinality is a kind of restriction that can be used to defined classes

Thus, we can use an owl:Restriction class to specify cardinality constraints

owl:maxCardinality



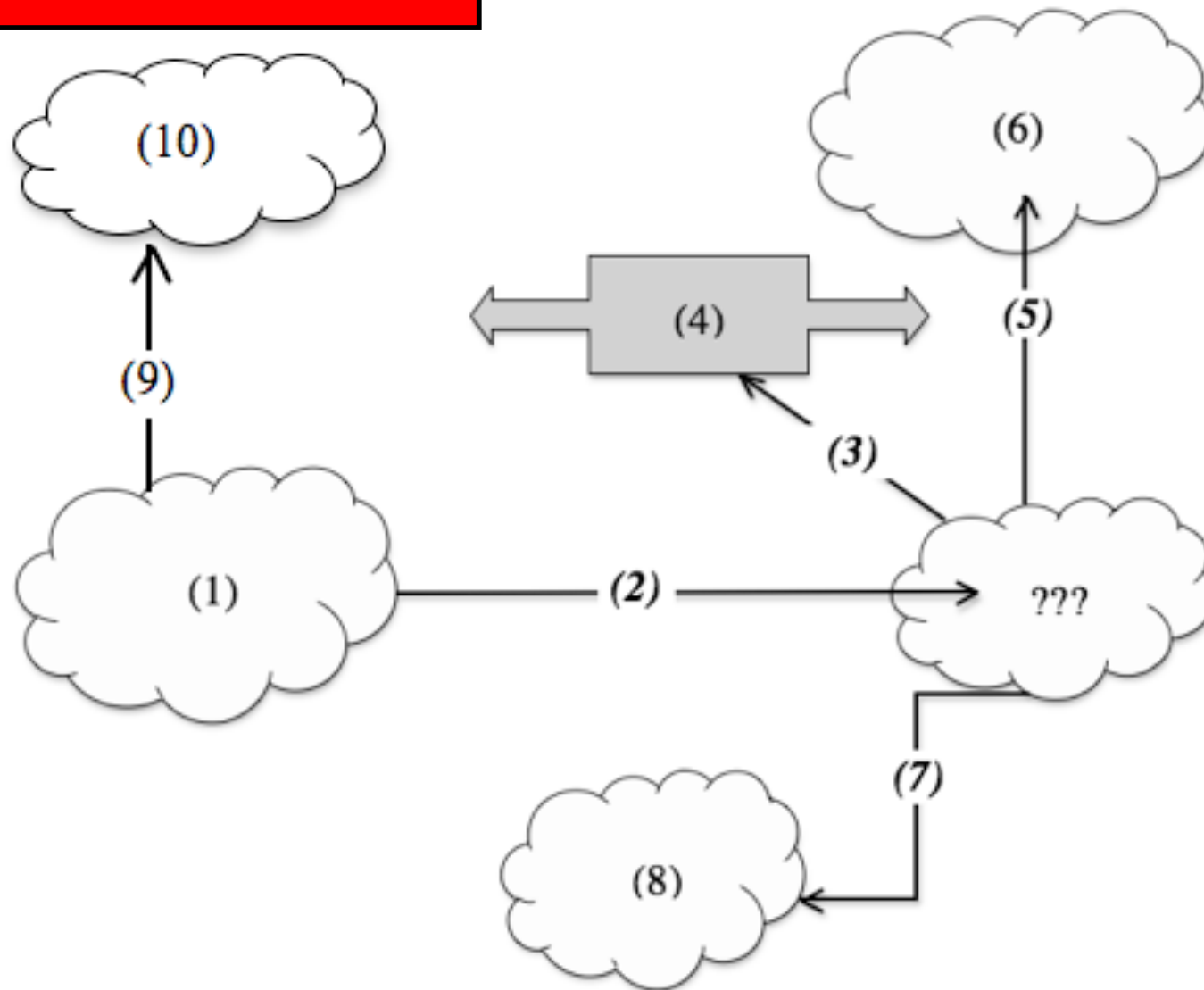
owl:minCardinality



Owl Exercise

- A **Superhero** is a kind of **Person**
- Every **Superhero** has a *secret identity*. The secret identity of a **Superhero** is the **Person** underneath the mask (e.g., Bruce Wayne is the secret identity of Batman, Clark Kent is the secret identity of Superman).
- When a **Person** (like Bruce Wayne) is secretly a **Superhero**, we say that the **Superhero** identity is the *alter-ego* of that **Person**. So, e.g. Batman is the alter-ego of Bruce Wayne.
- When a **Person** has an alter-ego that is a **Superhero**, or a **Superhero** has a secret identity that is a **Person**, the **Superhero** and the **Person** are the *same* individual.
- Every **Person** can have one or more **Abilities**.
- A **Superpower** is a kind of **Ability**.
- A **Superhero** is a **Person** for which one or more of their abilities is a **Superpower**

Owl Exercise



Replace the numbers with the correct symbols

Web Ontology Language: Property Types

OWL allows a knowledge-engineer to specify some interesting characteristics of properties in an ontology, reminiscent of AI Frames

NB: a property *linking two individuals* is an ObjectProperty in OWL
a property that *links an individual to a value* is a DataTypeProperty

Characterizing properties in OWL:

inverseOf: one property is the inverse of another (e.g., *parent* and *child*)

TransitiveProperty: two properties are transitive, allows closure inference

SymmetricProperty: a property is its own inverse (e.g., *friend*)

FunctionalProperty: the property corresponds to a well-formed function, so that no individual has more than one value.

Web Ontology Languages: OWL DL (Description Logic)

OWL is a form of Description Logic. It is complete and decidable.

OWL offers *maximal expressiveness* for a *complete, decidable* system

OWL-DL is named for Description Logics in AI:

DL *concepts* are OWL *classes* DL *roles* are OWL *properties*

Does not make either *Unique-Names* or the *Closed-World* assumptions

Designed as *first-order* extensions to Frames / Semantic Networks in AI

to support declarative rule-based reasoning (no *opaque procedures*)

Web Ontology Languages: Description Logics

A fragment of First-Order Logic for defining concepts and their relations.

DL distinguishes between *general* facts about concepts only (terminology) and *specific* facts about the instances of these concepts (assertions).

The T-Box (terminology-box) contains all general facts about concepts:

Every politician is a person *a fact about all politicians, in the T-Box*

Every Senator is a person *a fact about all senators, in the T-Box*

The A-Box (assertion-box) contains all ground facts about instances:

Bertie is a politician *a fact just about Bertie, in the A-Box*

John is a Senator *a fact just about John, in the A-Box*

Ontology Languages: Types of Description Logics

The family of distinct DLs is large, based on the following criteria:

- \mathcal{F} Allows Functional properties
- \mathcal{E} Allows full, qualified Existential restriction of properties
- \mathcal{O} Allows nOminals – enumerated classes of object value restrictions
- \mathcal{Q} Qualified Cardinality restrictions (*specify numbers of types*)
- \mathcal{S} (abbrev) Simple Negation, Concept Intersection, Univ. restrictions
- \mathcal{H} Allows Hierarchical ordering of roles (e.g., *subPropertyOf* in RDF)
- \mathcal{R} Allows complex Role (Property) definitions, disjointness, etc.
- \mathcal{I} Allows Inverse properties / roles
- \mathcal{N} Allows cardinality / Number restrictions on roles

Ontology Languages: Naming Description Logics

Description Logics are named for the characteristics they exhibit:

$SHOIN$ Exhibits S, H, O, I and N characteristics
(e.g., *atomic negation, hierarchical roles, inverses, nominal values, cardinality restrictions, etc.*)

$SHIF$ Exhibits S, H, I and F characteristics

$SHOIN^{(D)}$ Exhibits S, H, O, I, N and allows datatype properties
(i.e., *+ properties that link an individual to a value*)

$SHIF^{(D)}$ Exhibits S, H, I, F and allows datatype properties

OWL-lite is $SHIF^{(D)}$, OWL-DL is $SHOIN^{(D)}$, OWL-full is $SROIQ^{(D)}$

Semantic Web: Conclusions

The Semantic Web is an attempt to formalize the contents of the WWW

As such, it builds on the proven, formal foundations of AI:

Ontologies (class/ISA hierarchies, frames, semantic networks)

Description Logics (a.k.a Terminology Logics / Concept Logics in AI)

Theorem Provers / Reasoners

The success of the Semantic Web (or Web 3.0) depends on several factors:

The availability of good tools (e.g., Protégé)

The development of rich, shared and inter-operable ontologies

The willingness to use these to provide semantically-marked content