

# The Semantic Web

2 sessions in the Module INF347  
at the École nationale supérieure des  
Télécommunications  
in Paris/France in Summer 2011

by Fabian M. Suchanek



This document is available under a  
[Creative Commons Attribution Non-Commercial License](https://creativecommons.org/licenses/by-nc/4.0/)

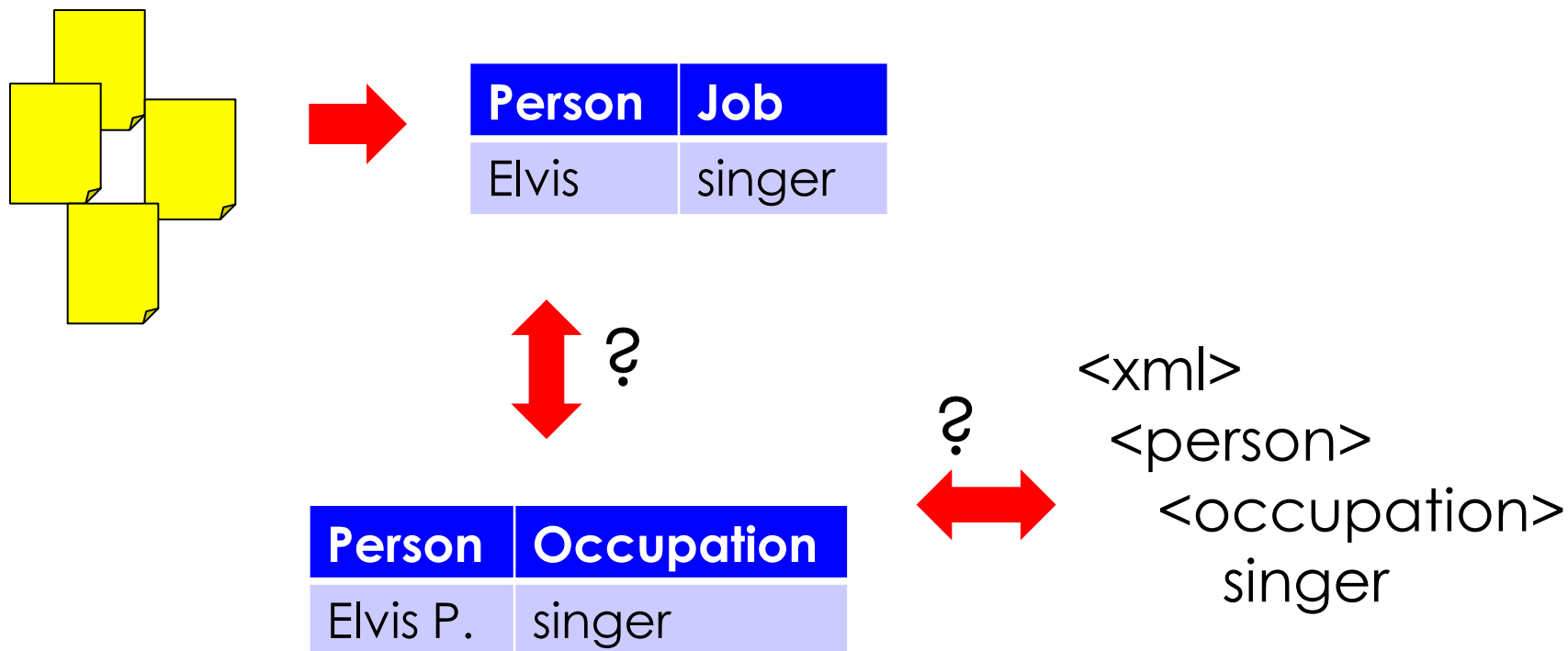
# Organisation

- 2 sessions (each 1.5h) on the Semantic Web
- 1.5h lab session
- Web-sites:  
<http://www.infres.enst.fr/~danzart/INF347/>  
<http://suchanek.name/> → Teaching

# Motivation

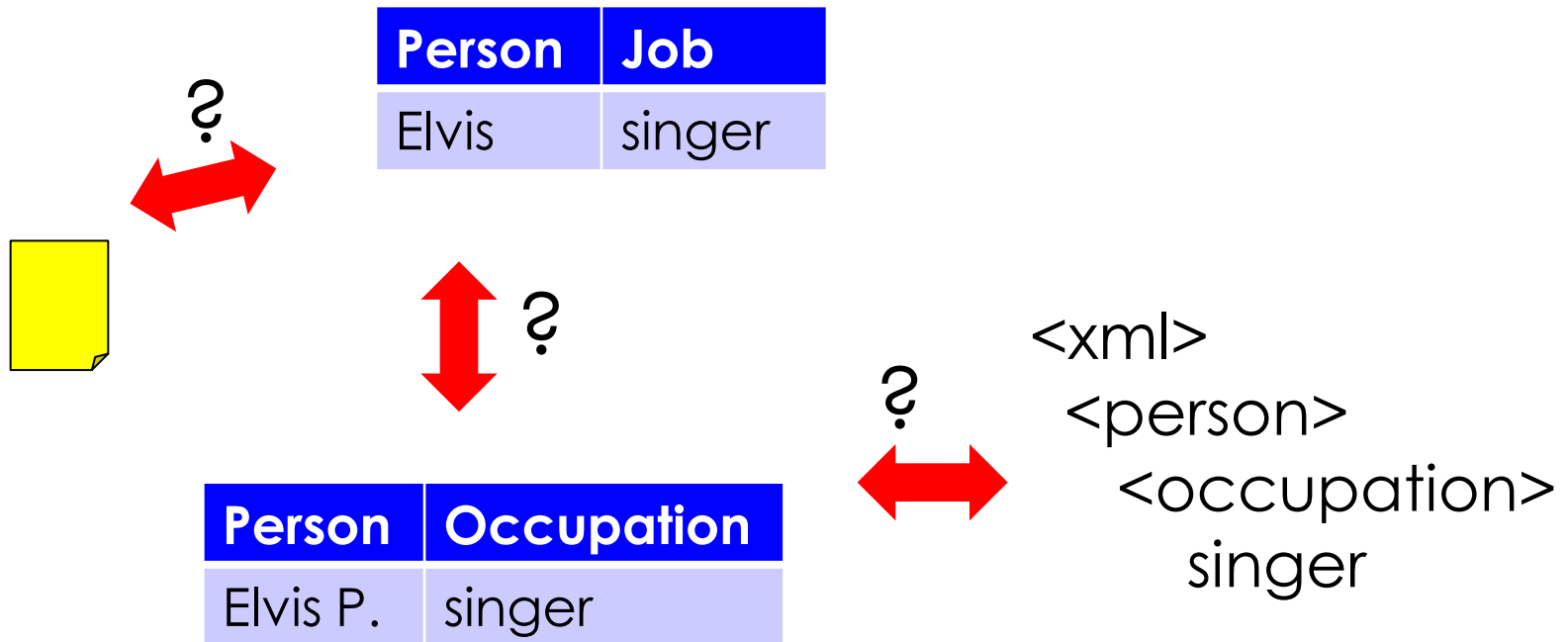
In the last class on Information Extraction, we saw how to move from unstructured data to structured data

But even between structured data, interaction is difficult...



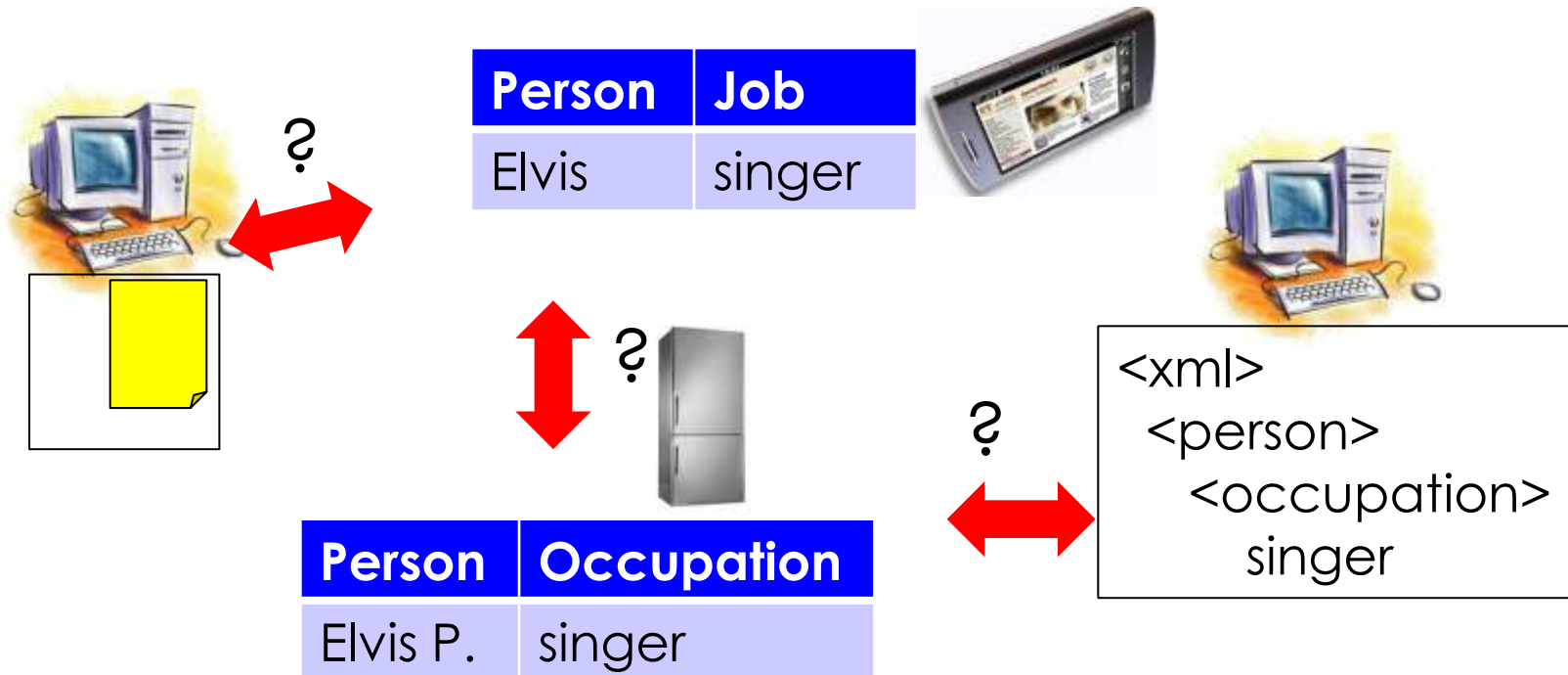
# Motivation

But even between structured data, interaction is difficult, in particular if the data is in different formats



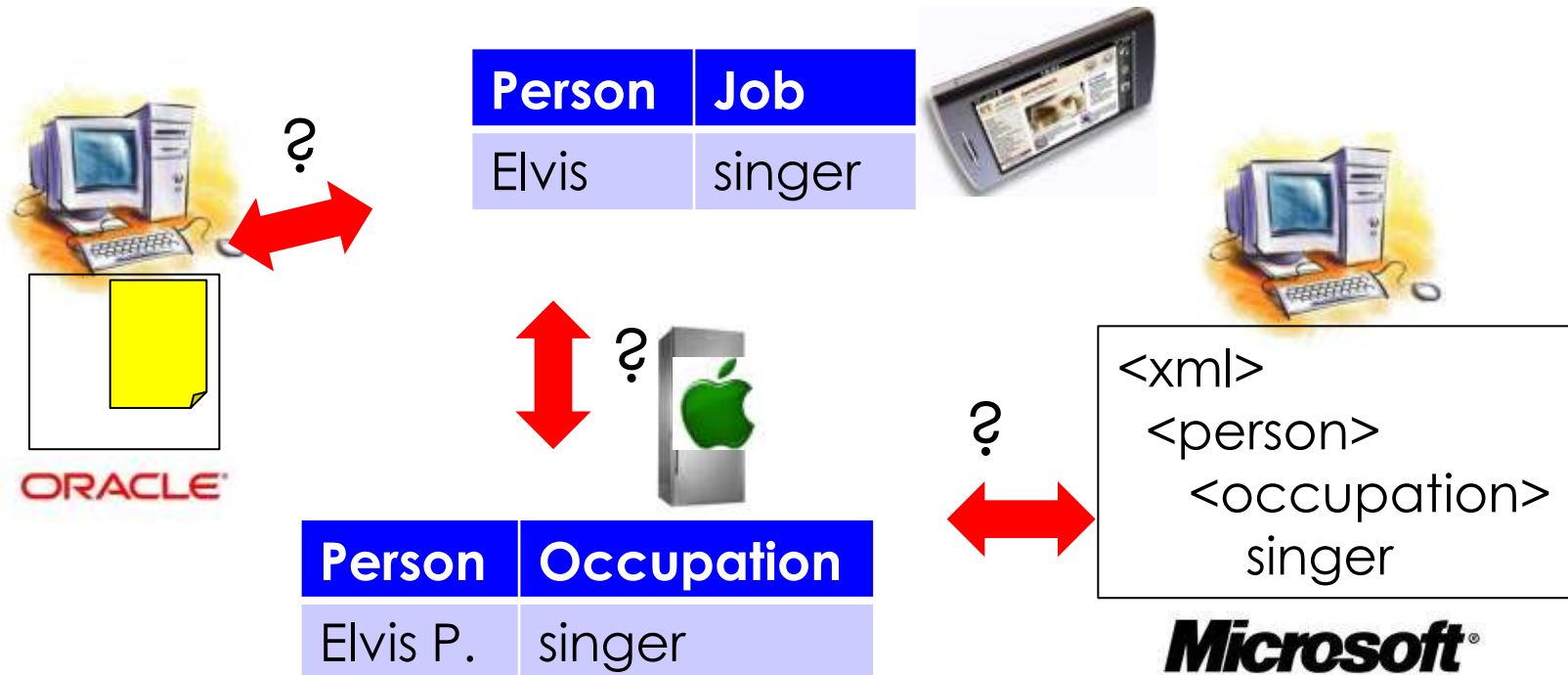
# Motivation

But even between structured data, interaction is difficult, in particular if the data is in different formats, on different machines or devices



# Motivation

But even between structured data, interaction is difficult, in particular if the data is in different formats, on different machines or devices or in different companies



# Use cases

## Examples:

- Booking a flight  
Interaction between office computer, flight company, travel agency, shuttle services, hotel, my calendar
- Finding a restaurant  
Interaction between mobile device, map service, recommendation service, restaurant reservation service
- Web search  
Interaction between client, search service, Web page content provider

# Use cases

- Web service composition  
Interaction between client and Web services and  
Web services themselves
- Intelligent home  
Fridge knows my calendar, orders food  
if I am planning a dinner
- Intelligent cars  
Car knows my schedule, where and when to get gas,  
how not to hit other cars, what are the legal regulations

# Merging

## Examples:

- Adding data to a database  
From XML files, from other databases
- Merging data after company mergers  
(e.g. Apple buys Microsoft)  
Different terminology has to be bridged,  
accounts to be merged
- Merging data in research  
e.g. biochemical, genetic , pharmaceutical research data  
  
(Less exciting, but probably more frequent)

# The Semantic Web

Idea: We need an infrastructure that allows computers to “understand” their data.

This infrastructure shall

- allow machines to process data from other machines
- ensure interoperability between different schemas, devices and organizations
- allow data to describe data
- allow machines to reason on the data
- allow machines to answer semantic queries



This is what the Semantic Web aims at

# The Semantic Web

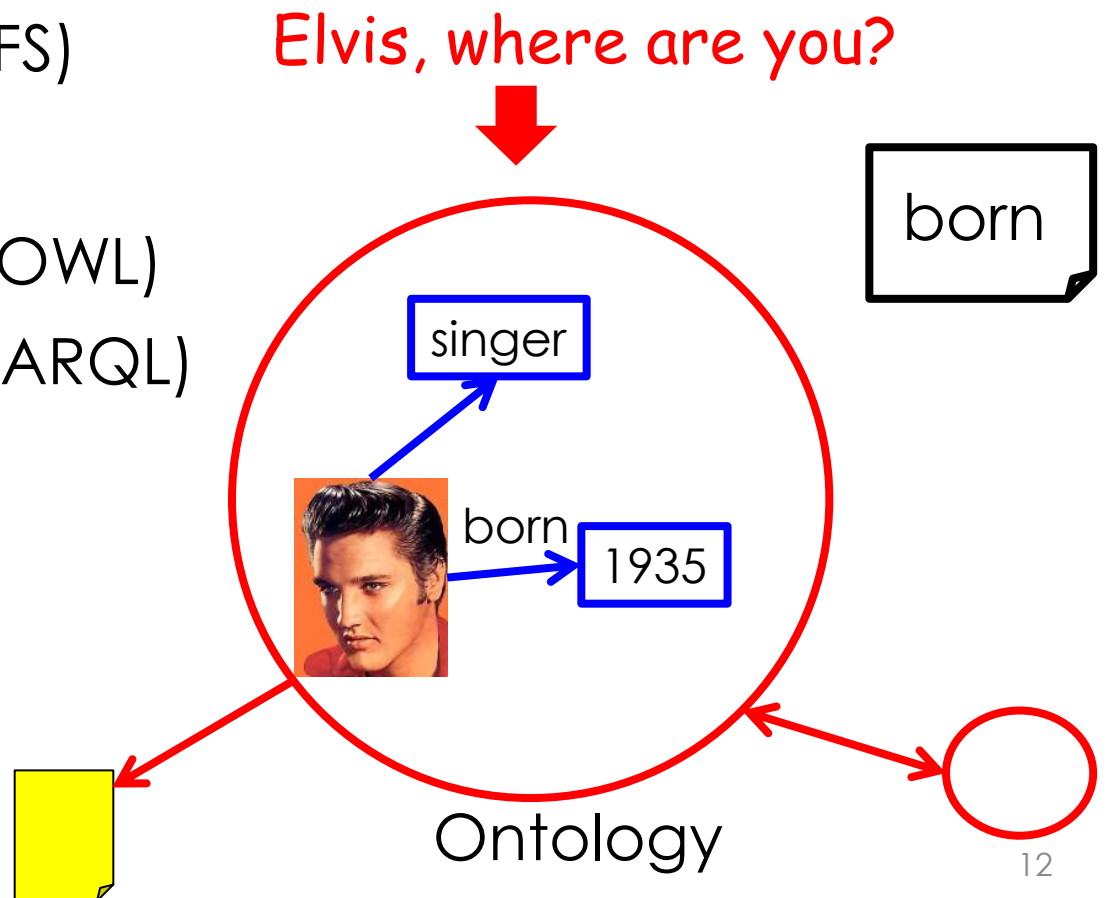
The **Semantic Web** is an evolving extension of the World Wide Web, in which data is made available in one standardized semantic format.

... we will learn more about this format  
in a minute

# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)



# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)

Standards produced by or endorsed by the **World Wide Web Consortium (W3C)**

...represented in Europe by the **European Research Consortium for Informatics and Mathematics (ERCIM)**, with INRIA being a member.

# URIs

Goal: Identify entities uniquely, worldwide  
The same entity can have multiple identifiers,  
but the same identifier shall always mean the same entity.



~~Elvis~~



Elvis



Elvis Presley

# URIs

A **Uniform Resource Identifier** (URI) is a string of characters used to identify a name or a resource on the Internet

A URI can take the form of a URL



<http://elvis.org/me>



<http://imitators.org/Elvis/FG17>

<http://onto.com/people/singers/EP>

# URIs and URLs



<http://elvis.org/me>

Identifies the person,  
not Internet-accessible

Age

76

<http://elvis.org/index.html>

Identifies a file,  
Internet-accessible

5

# Namespaces

<http://imitators.org/Elvis/FG17>

World-wide unique  
mapping to domain  
owner

in the responsibility  
of the domain owner

⇒ The domain provides a “**namespace**”,  
i.e., a range of identifiers that cannot collide with other  
identifiers

{ <http://imitators.org/Elvis>, ..., <http://imitators.org/Madonna> }



{ <http://elvis.org/Elvis> , ... }      { <http://holy.org/Madonna> , ... }

# URI Use Cases

<http://imitators.org/Elvis/FG17>



World-wide unique  
mapping to domain  
owner

in the responsibility  
of the domain owner

⇒ There should be no  
URI with two meanings

⇒ People can invent all kinds of URIs

- a company can create URIs to identify its products
- an organization can assign sub-domains and each sub-domain can define URIs
- individual people can create URIs from their homepage
- people can create URIs from any URL for which they have exclusive rights to create URIs

# URNs

A **Uniform Resource Name** (URN) is a URI that is globally assigned.

A URN takes the form

“urn:” + Namespace + “:” + Identifier

The **IANA** (Internet Assigned Numbers Authority), operated by ICANN, assigns namespaces to specific organizations.

The organization then specifies the identifier.

# Example URNs

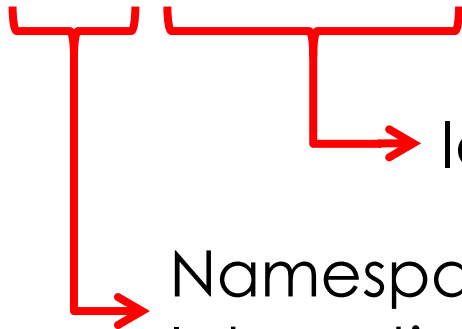
A **Uniform Resource Name** (URN) is a URI that is globally assigned.

A URN takes the form

“urn:” + Namespace + “:” + Identifier

Example:

urn:isbn:123456789



Identifier assigned by ISO to a book

Namespace assigned by IANA to the International Standards Organization (ISO)

# Example URNs

- ISBN: urn:isbn:1234567
- ISAN: urn:isan:0000-1111-2222-3333-4444
- SWIFT: urn:swift:bic:BYLADEM1000
- OID: urn:oid:2.16.840

examples are  
hypothetical

**Object IDs** identifiers form one global tree of identifiers, where sub-trees are administrated by authorities.

For example, the IANA administrates ids for companies

[Example](#)

# UUIDs

- UUID: urn:uuid:6e8bc430-9c3a-11d9-9669-0800200c9a66

A **Universally Unique ID** (UUID) is a software-generated string that is globally unique.

Computed, e.g., by using

- the MAC address (identifier of the computer)
- plus a timestamp

Example: `run ifconfig / getmac`

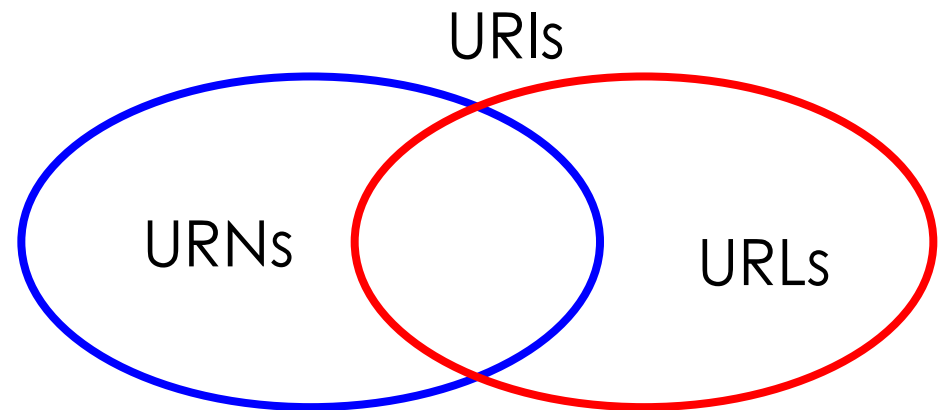
# URIs Summary

A **Uniform Resource Identifier** (URI) is a string of characters used to identify a name or a resource on the Internet.

The goal is to give “all things on Earth” a unique identifier.

Two main approaches:

1. URL-like identifiers
2. Uniform Resource Names (URNs)



*Propose another real-world domain  
where we could establish URNs!*

# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- ➔ Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)

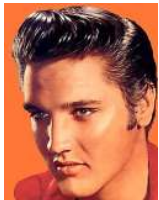
# RDF

The **Resource Description Framework** (RDF) is a standardized knowledge representation model that resembles the entity-relationship model.

An **RDF statement (triple)** is a triple of 3 URIs, called the subject, the predicate and the object.

<http://elvis.org/himself>   <http://inria.fr/rdf/dta#won>   <http://g-a.com/prize>

identifies



Elvis Presley

identifies



the relationship of  
winning something

identifies



Grammy Award<sup>25</sup>

# RDF

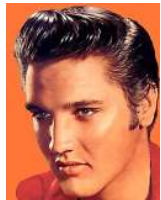
“Elvis won the Grammy Award”

<http://elvis.org/himself>

<http://inria.fr/rdf/dta#won>

<http://g-a.com/prize>

identifies



Elvis Presley

identifies



the relationship of  
winning something

identifies



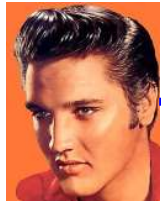
Grammy Award<sup>26</sup>

# RDF Graphs

A set of RDF statements is **isomorphic to a labeled directed multi-graph**, which is the notation we will use here:

The subject and object of a triple correspond to nodes, the predicate corresponds to directed edge from subject to object with a label given by the predicate.

<http://elvis.org/himself> <http://inria.fr/rdf/dta#won> <http://g-a.com/prize>



Elvis Presley

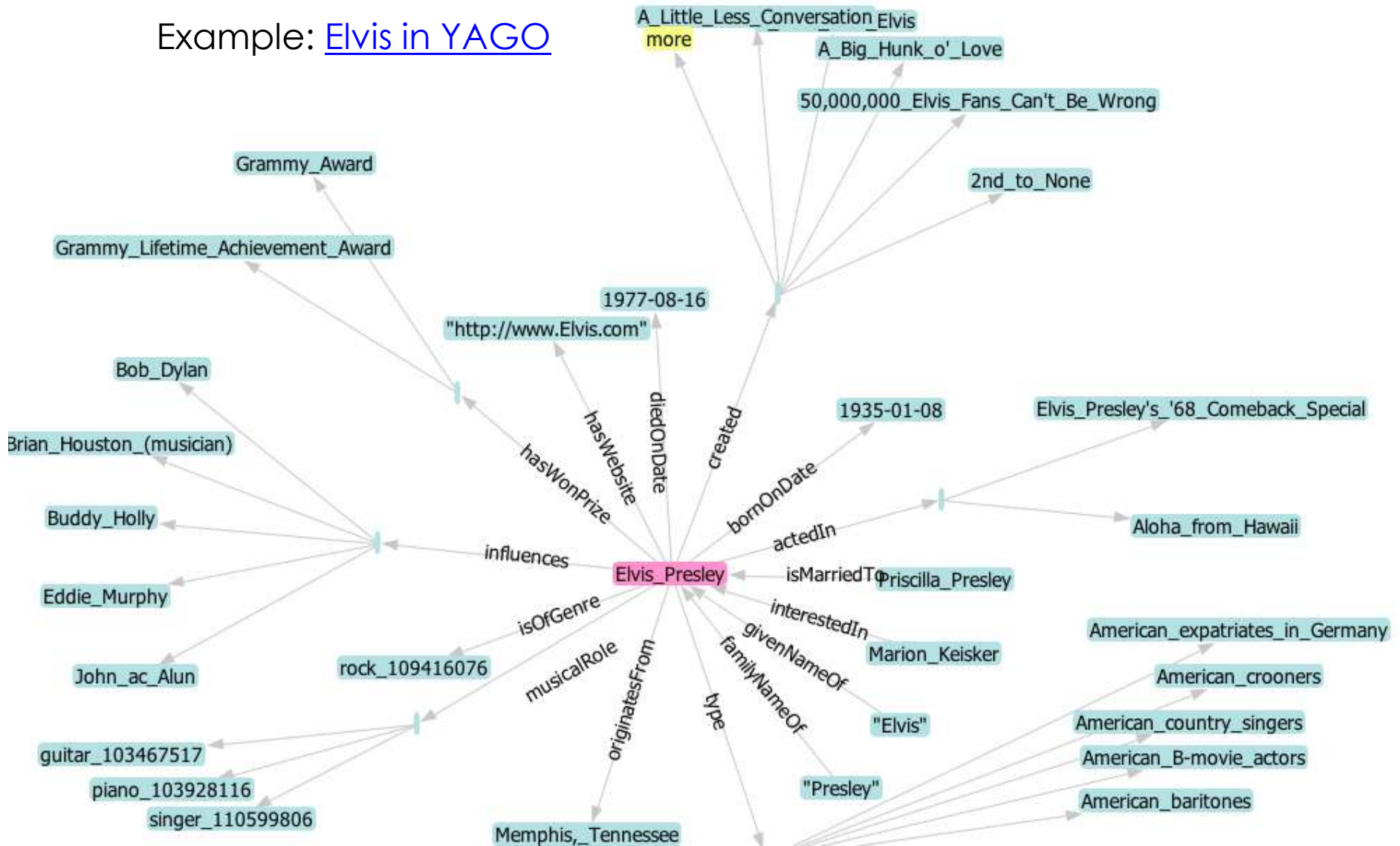
won



Grammy Award<sup>27</sup>

# Sample RDF Graph

Example: [Elvis in YAGO](#)



# Namespace Prefixes

A **namespace prefix** is an abbreviation for the prefix of a URI.

@prefix elvis: <http://elvis.org/>

@prefix inria: <http://inria.fr/rdf/dta#>

@prefix grammy: <http://g-a.com/>

<http://elvis.org/himself>    <http://inria.fr/rdf/dta#won>    <http://g-a.com/prize>



elvis:himself



inria:won



grammy:prize

A URI abbreviated this way is called a **qname**.

# Default Prefix

The default namespace prefix is just a colon. In the following we assume such a prefix.

@prefix : <http://elvis.org/>

@prefix inria: <http://inria.fr/rdf/dta#>

@prefix grammy: <http://g-a.com/>

<http://elvis.org/himself>    <http://inria.fr/rdf/dta#won>    <http://g-a.com/prize>



:himself



inria:won

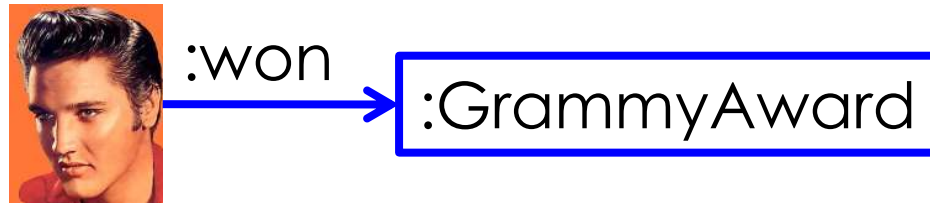


grammy:prize

# Tabular Data in RDF

All tabular data can be expressed in RDF:

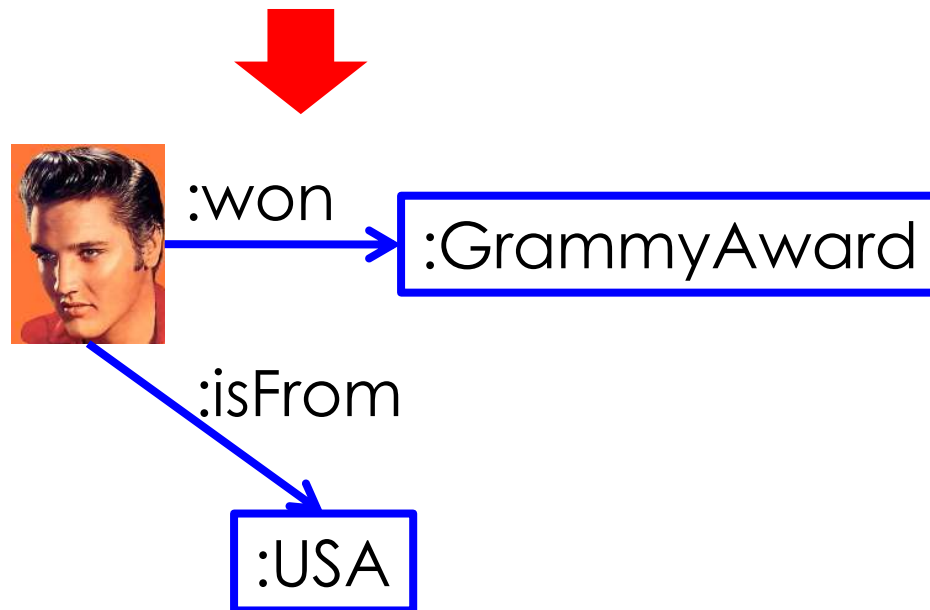
Person	Prize
Elvis	Grammy Award



# Tabular Data in RDF

All tabular data can be expressed in RDF:

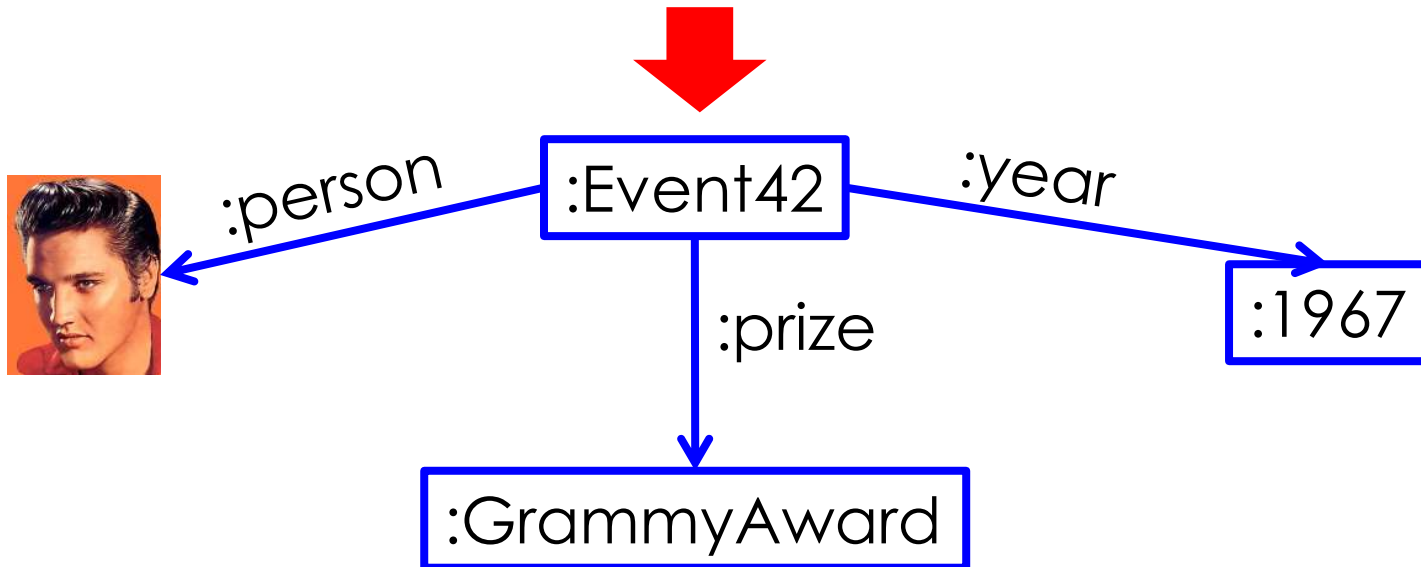
Person	Prize	Country
Elvis	Grammy Award	USA



# Event Entities

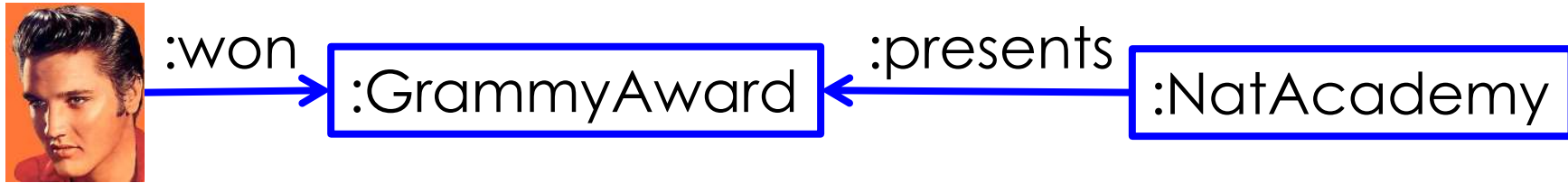
All tabular data can be expressed in RDF:

Person	Prize	Year	
Elvis	Grammy Award	1967	Event42



**Event entities** are artificial entities that represent a complex constellation

# RDF Semantics

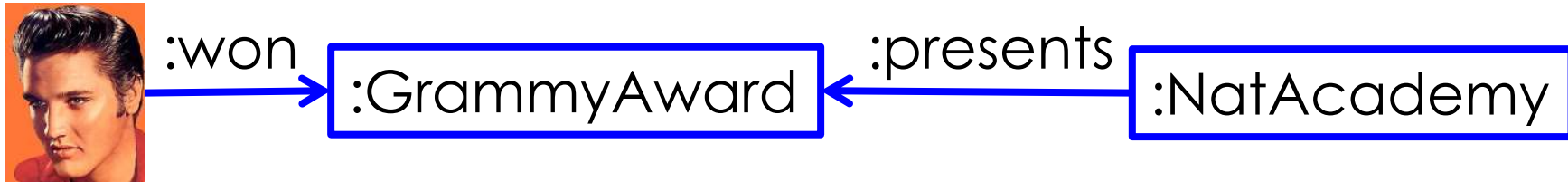


A triple  $\langle s, p, o \rangle$  is interpreted as a First Order Logic fact  $p(s, o)$ .

won(Elvis, GrammyAward)  
presents(NatAcademy, GrammyAward)

The triple of URIs  $\langle s, p, o \rangle$ , the graph and the First Order Logic fact are equivalent representations of the same statement.

# Notation 3



**Notation 3** (N3) is a concrete syntax for RDF.

Qnames or URIs

@prefix : <<http://inria.fr/>>

@prefix elvis: <<http://elvis.org/>>

:NatAcademy :presents <<http://g-a.com/prize>> .

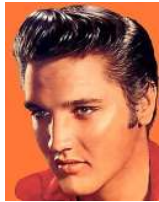
elvis:himself :won <<http://g-a.com/prize>> ;  
:plays <[#guitar](#)> .

Same subject can be abbreviated

Namespaces  
become XML  
namespaces

# RDF XML Syntax

stored in XML format



:won

:GrammyAward

:presents

:NatAcademy

```
<?xml version="1.0"?>
```

```
<rdf:RDF xmlns:rdf="http://www.w3.org/.../ns#"
  xmlns:i="http://inria.fr/rdf/dta#">
```

```
<rdf:Description
```

```
  rdf:about="http://elvis.org/himself">
```

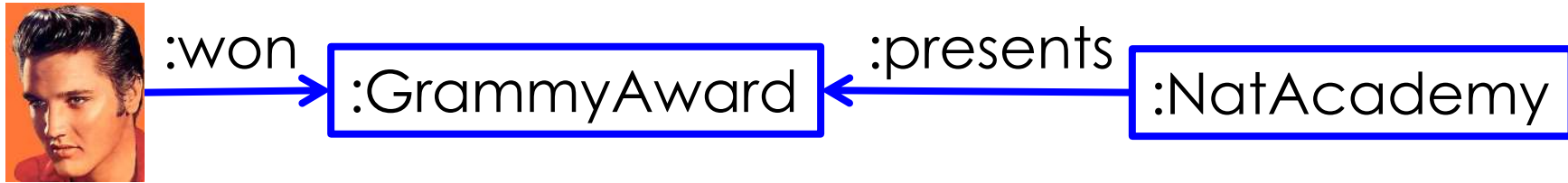
```
    <i:won rdf:resource="http://.../Award" />
```

```
</rdf:Description>
```

Properties of X are listed inside an  
rdf:Description about=X

# RDF: Concrete Syntax

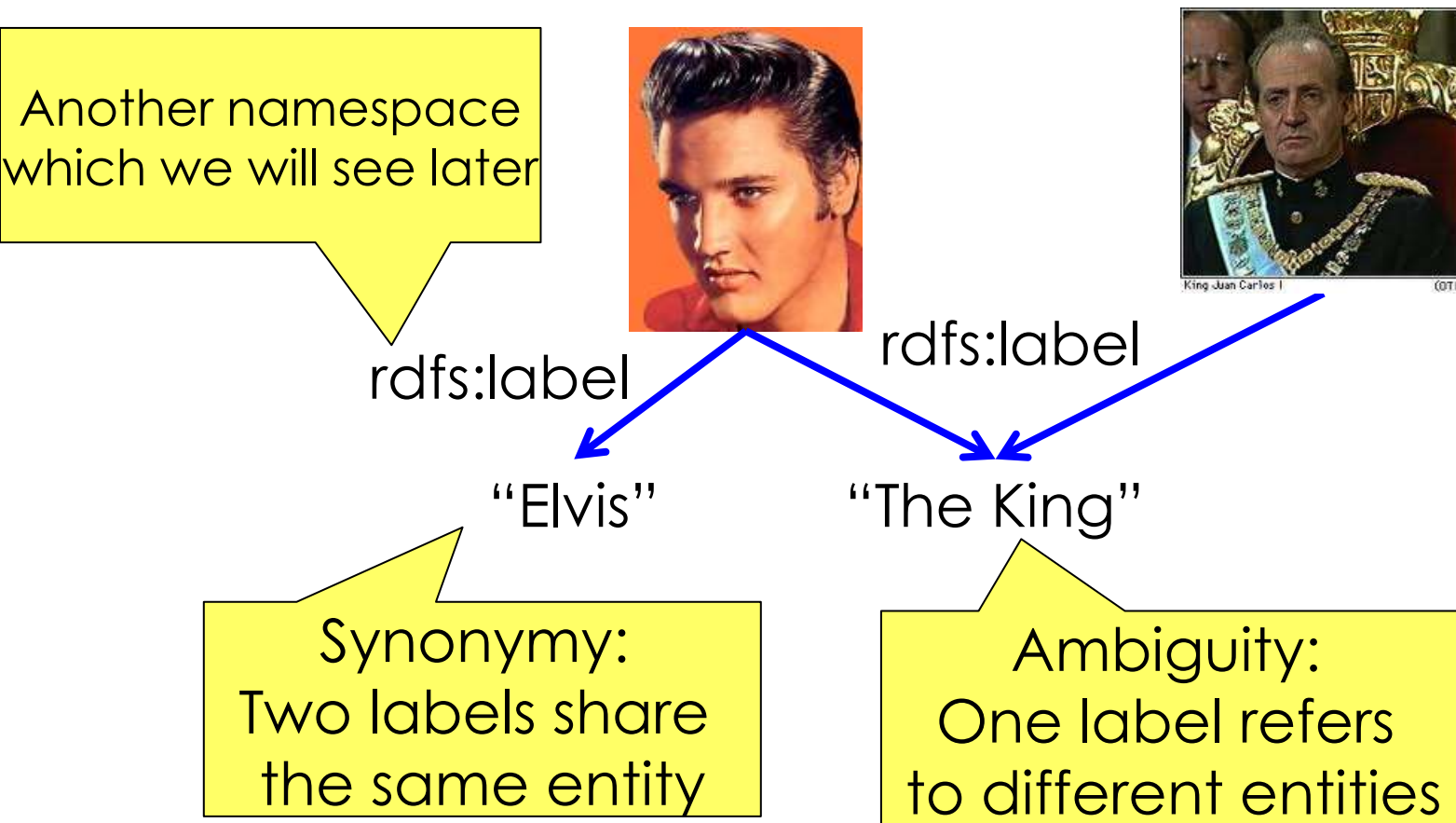
RDF can also in a database



Subject	Predicate	Object
<a href="http://elvis.org/himself">http://elvis.org/himself</a>	won	<a href="http://g-a.com/prize">http://g-a.com/prize</a>
...	...	...

# Labels

A **label** is a human-readable name for an entity.



*Find 1 example for synonymy  
and 1 example for ambiguity.*

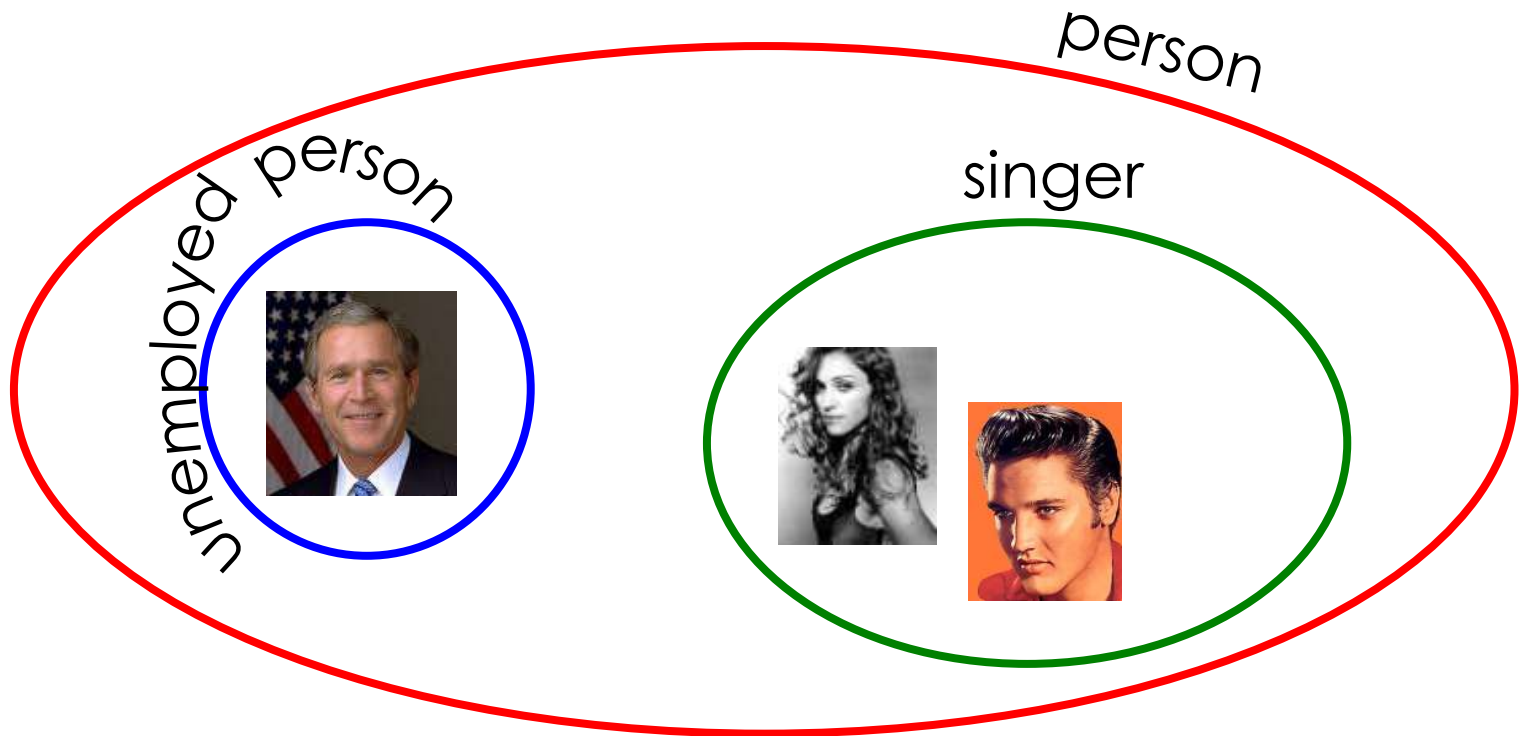
# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- ➔ Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)

# Classes

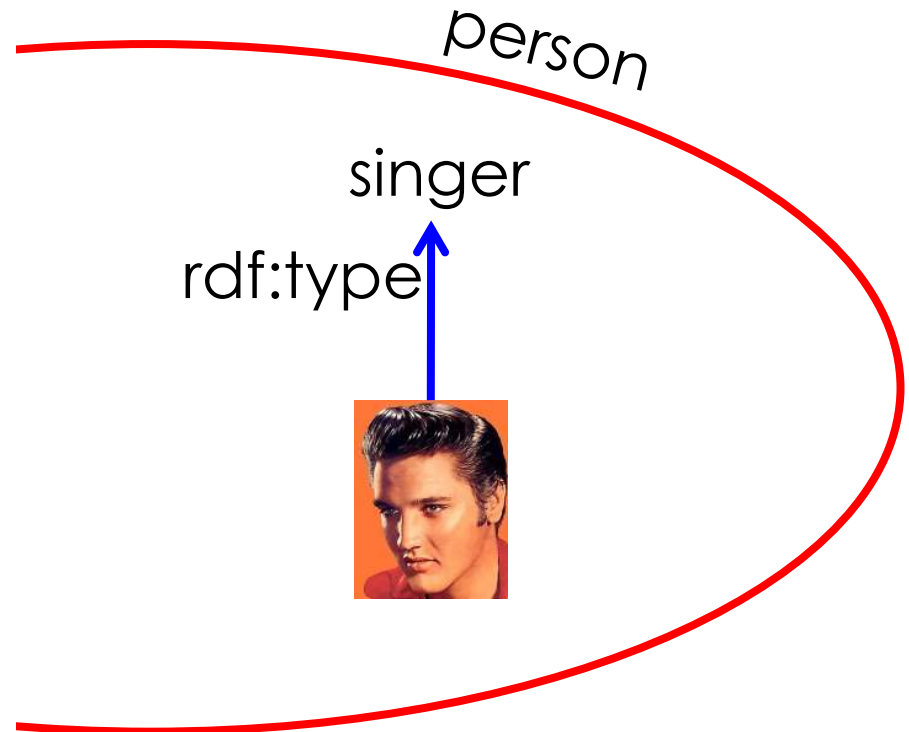
A **class** (also called concept) can be understood as a set of similar entities.



# Classes in RDF

The fact that an entity belongs to a class is expressed by the **type** predicate from the standard namespace rdf (<http://w3c.org/...>).

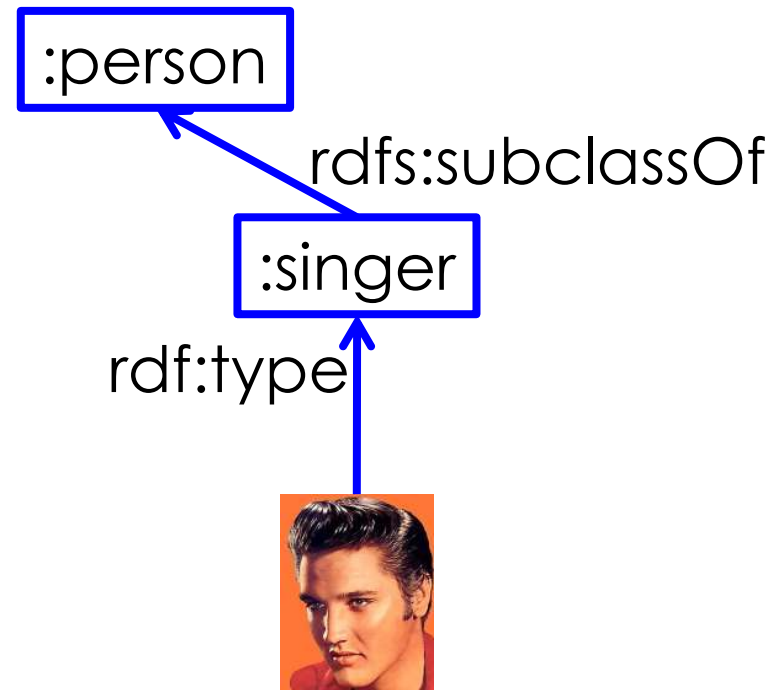
The fact that a class is a sub-class of another class is expressed by the **subclassOf** predicate from the standard namespace rdfs (<http://w3c.org/...>).



# Classes in RDF

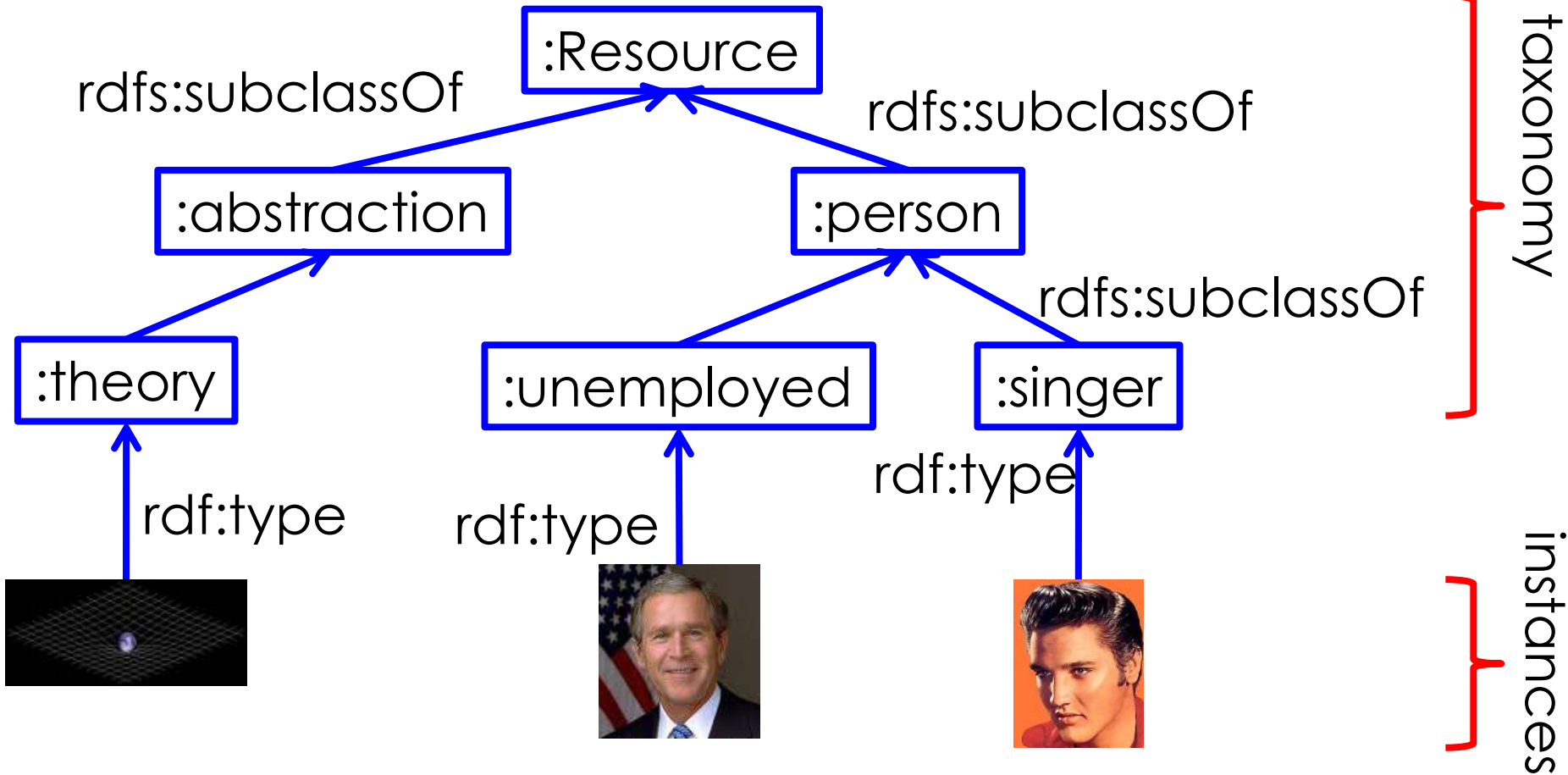
The fact that an entity belongs to a class is expressed by the **type** predicate from the standard namespace rdfs (<http://w3c.org/...> ).

The fact that a class is a sub-class of another class is expressed by the **subclassOf** predicate from the standard namespace rdfs (<http://w3c.org/...> ).



# Taxonomy

A **taxonomy** is a hierarchy of classes



# Taxonomy

The most general class is **rdfs:Resource** – everything is a resource.

More general class

More special class

rdfs:Resource

:person

:singer



rdfs:subclassOf

rdfs:subclassOf

rdf:type

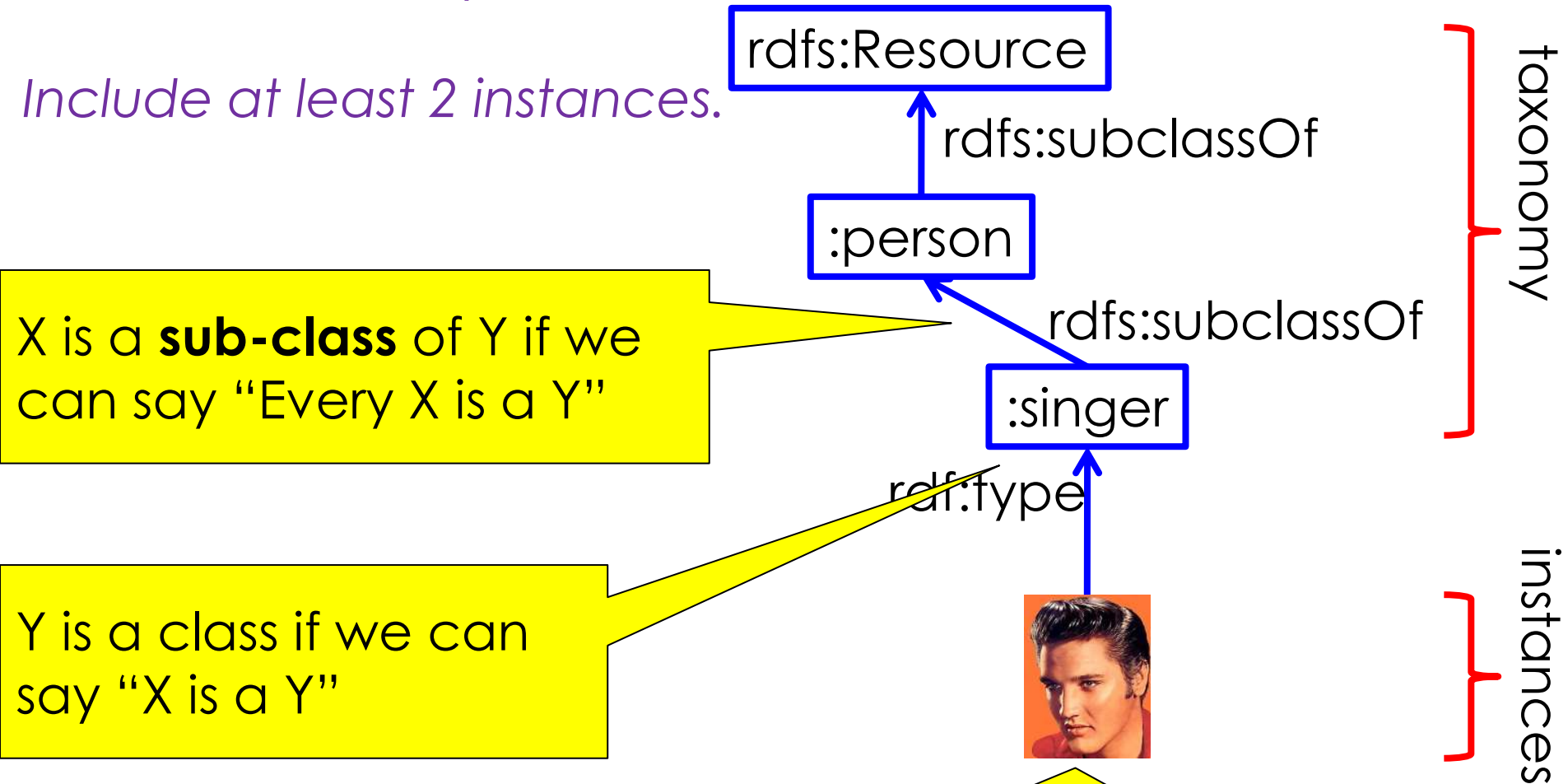
taxonomy

instances

# Taxonomy

*Make a taxonomy of animals.*

*Include at least 2 instances.*



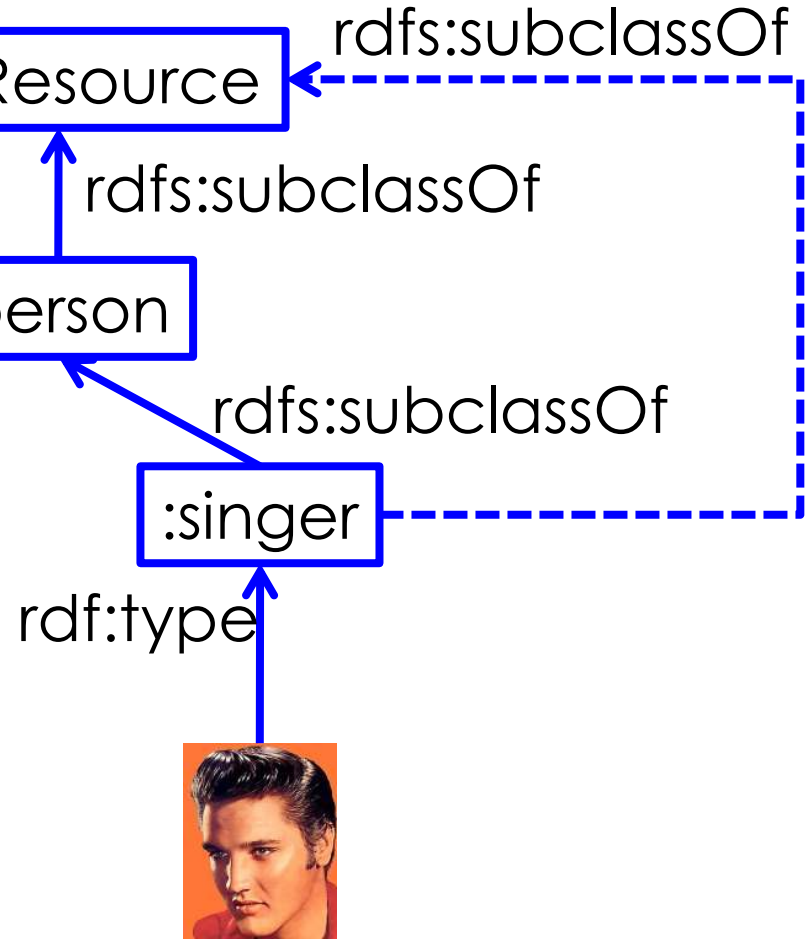
Everything else is an instance (of a particular class).

# SubclassOf Semantics

<X,subclassOf,Y>  
<Y,subclassOf,Z>

---

<X,subclassOf,Z>



Every class is a subclass of all more general classes

# RDFS Semantics

RDFS specifies 44 entailment rules of the form

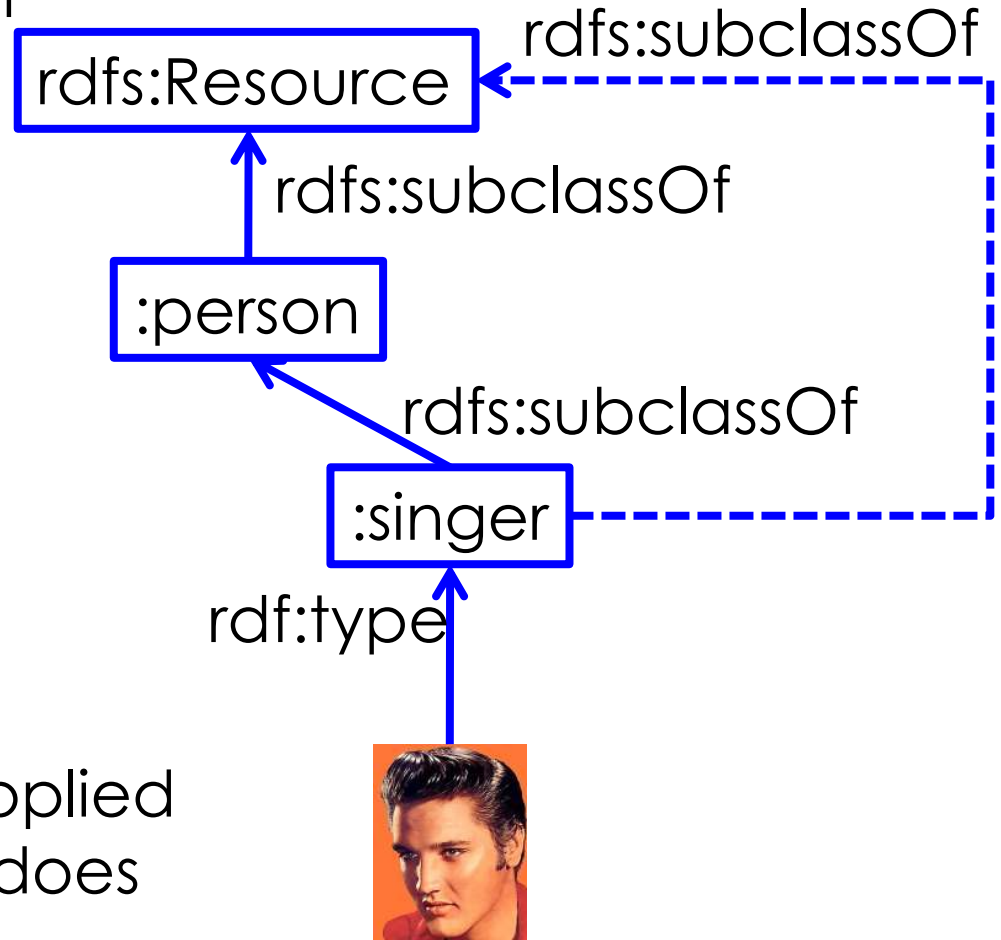
If the graph contains these triples

---

then the graph contains this triple

The entailment rules are applied recursively until the graph does not change any more.

The result is called the **deductive closure**.



# The Semantic Web

The Semantic Web provides standards to

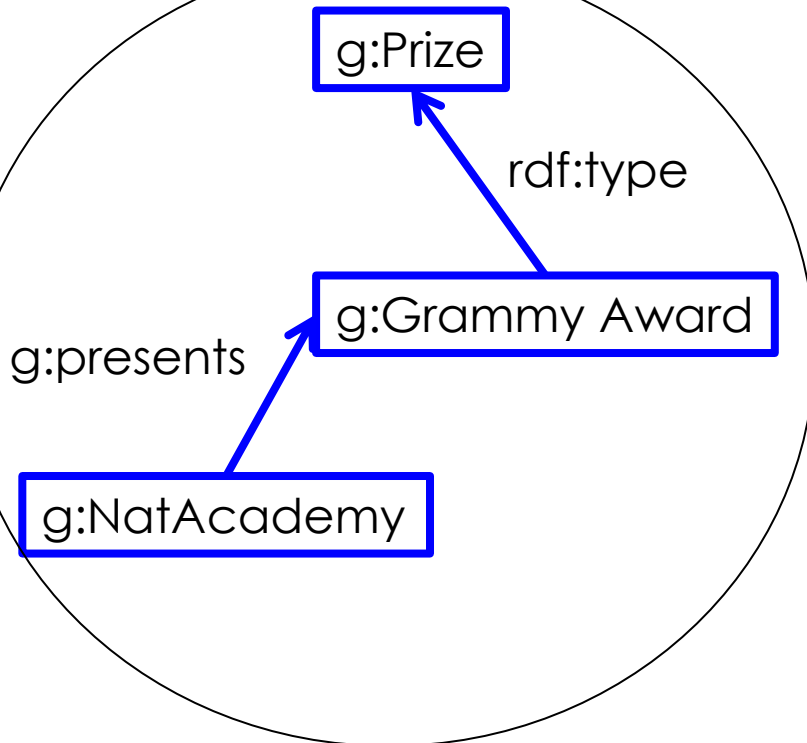
- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- ➔ Share vocabularies
  - Describe constraints (OWL)
  - Query knowledge (SPARQL)
  - Link data
  - Publish data (RDFa)

# Storing data

RDF data is usually stored on a server  
(=internet accessible computer)

Namespace

g = <http://g-a.com>



The server at  
<http://g-a.com>  
stores:

```
@prefix g: http://g-a.com
@prefix rdf: http://www.w3.org/...

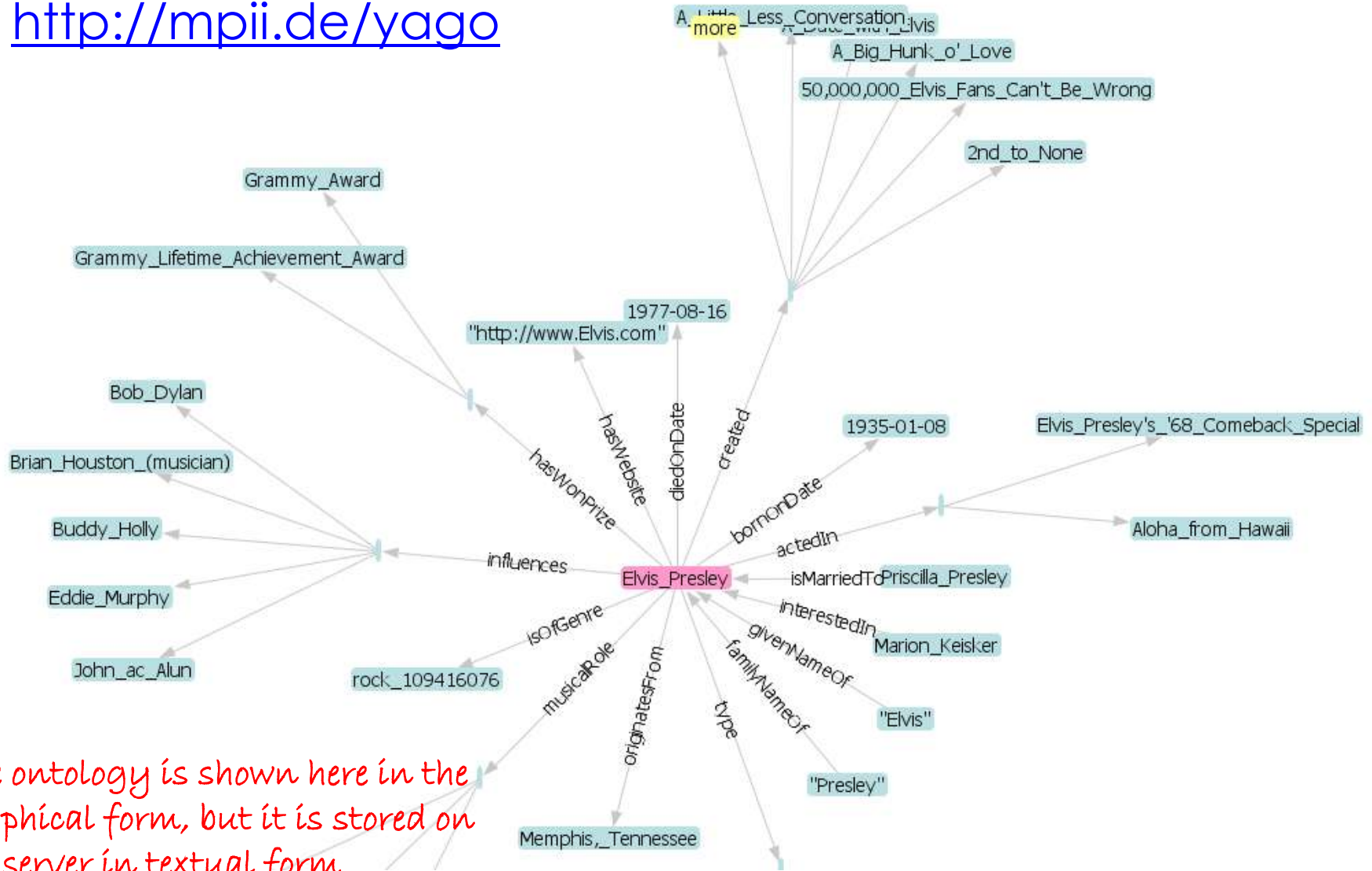
g:GrammyAward
  rdf:type    g:Award

g:NatAcademy
  g:presents g:GrammyAward
```

[Try this](#)

# Storing data

Example: The YAGO ontology is stored at the server at <http://mpii.de/yago>



The ontology is shown here in the graphical form, but it is stored on the server in textual form

# Cool URIs

A URI is not necessarily **dereferenceable** (i.e., it cannot be accessed online)

<http://g-a.com/GrammyAward> => NOT FOUND

... but it *can be* dereferenceable. This means that if I access the URL, the server responds with an RDF snippet:

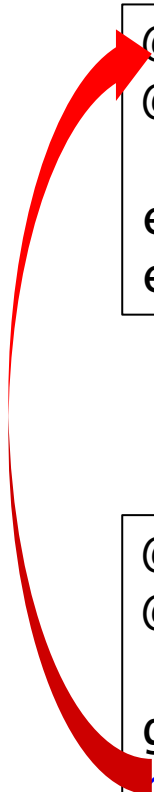
```
@prefix g:      http://g-a.com
@prefix rdf:    http://www.w3.org/1999/02/22-rdf-syntax-ns#

g:GrammyAward  rdf:type      g:Award
http://elvis.com/elvis  g:won      g:GrammyAward
```

Try this out: `rdf:type = http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

⇒ URIs can be “clicked” (followed)

# Cool URIs



```
@prefix e:      http://elvis.com
@prefix rdf:    http://www.w3.org/1999/02/22-rdf-syntax-ns#

e:elvis          rdf:type          e:singer
e:elvis          e:born            1935
```

Server at <http://elvis.com>

```
@prefix g:      http://g-a.com
@prefix rdf:    http://www.w3.org/1999/02/22-rdf-syntax-ns#

g:GrammyAward   rdf:type          g:Award
http://elvis.com/elvis  g:won            g:GrammyAward
```

Server at <http://g-a.com>

⇒ The RDF graph becomes traversable

# We're all one Graph

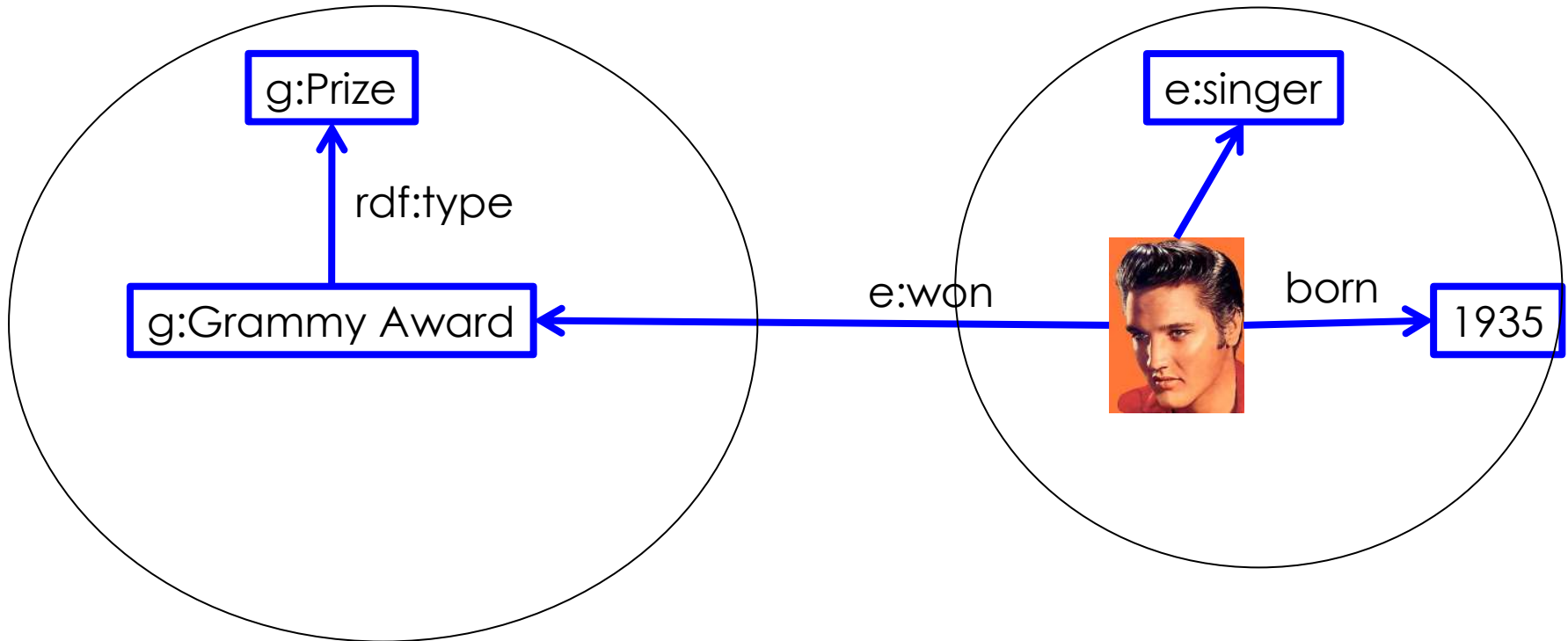
If two RDF graphs share one node, they are actually 1 graph.

Namespace

g = <http://g-a.com>

Namespace

e = <http://example.org>



A machine can follow the links and retrieve more information in the neighboring ontology.

# Standard Vocabulary

A number of standard vocabularies have evolved

rdf: The basic RDF vocabulary

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

rdfs: RDF Schema vocabulary

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

dc: Dublin Core (predicates for describing documents)

<http://purl.org/dc/elements/1.1/>

foaf: Friend Of A Friend (relationships between people)

<http://xmlns.com/foaf/0.1/>

cc: Creative Commons (types of licences)

<http://creativecommons.org/ns#>

# Standard Vocabulary

A number of standard vocabularies have evolved

rdf: The basic RDF vocabulary

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

rdfs: RDF Schema vocabulary

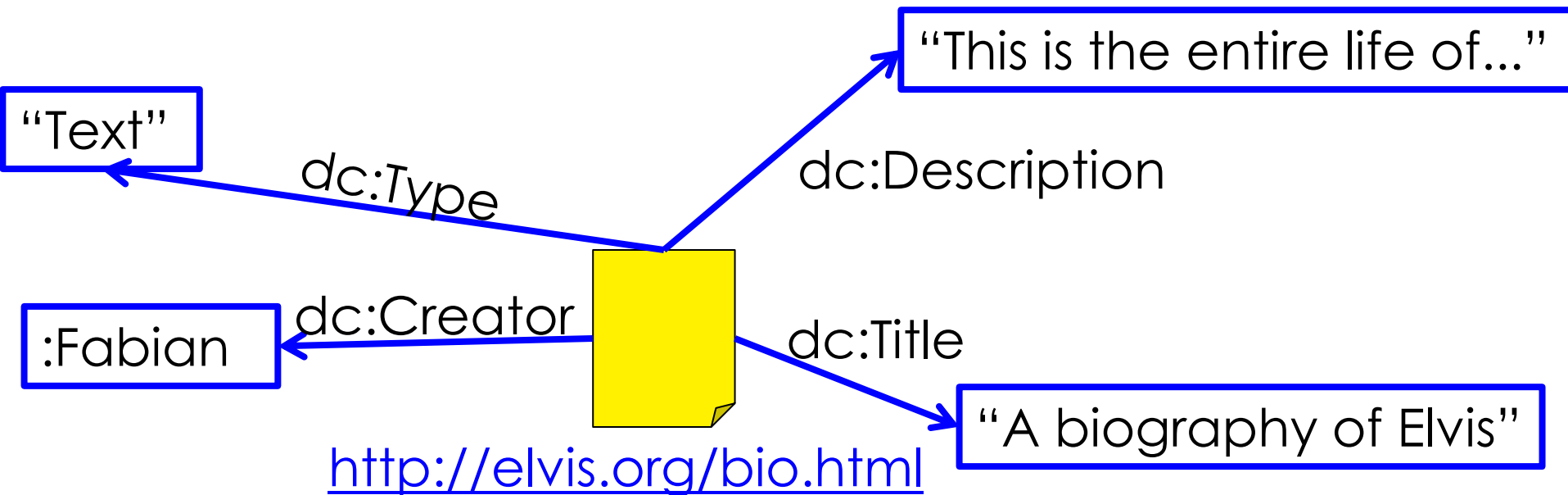
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

Standard vocabulary provided by the W3C:

- type,
- subclassOf,
- Property,
- Class
- label
- ....

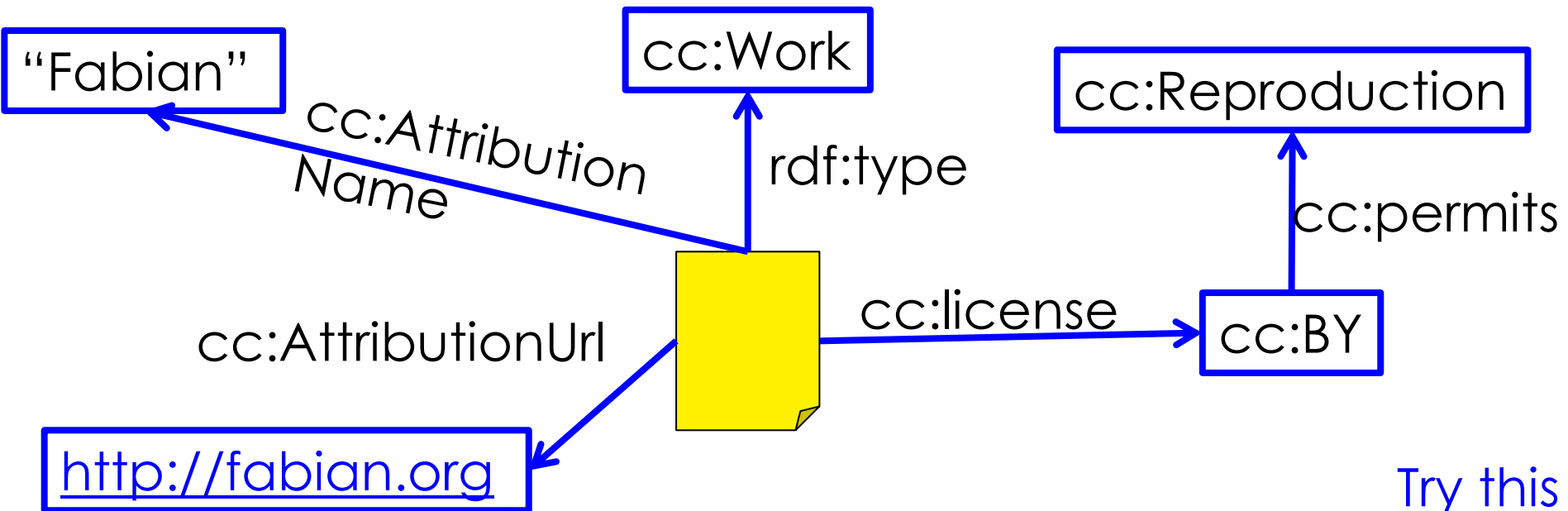
# Dublin Core

dc: Dublin Core (predicates for describing documents)  
<http://purl.org/dc/elements/1.1/>



# Creative Commons

cc: Creative Commons (types of licences)  
<http://creativecommons.org/ns#>



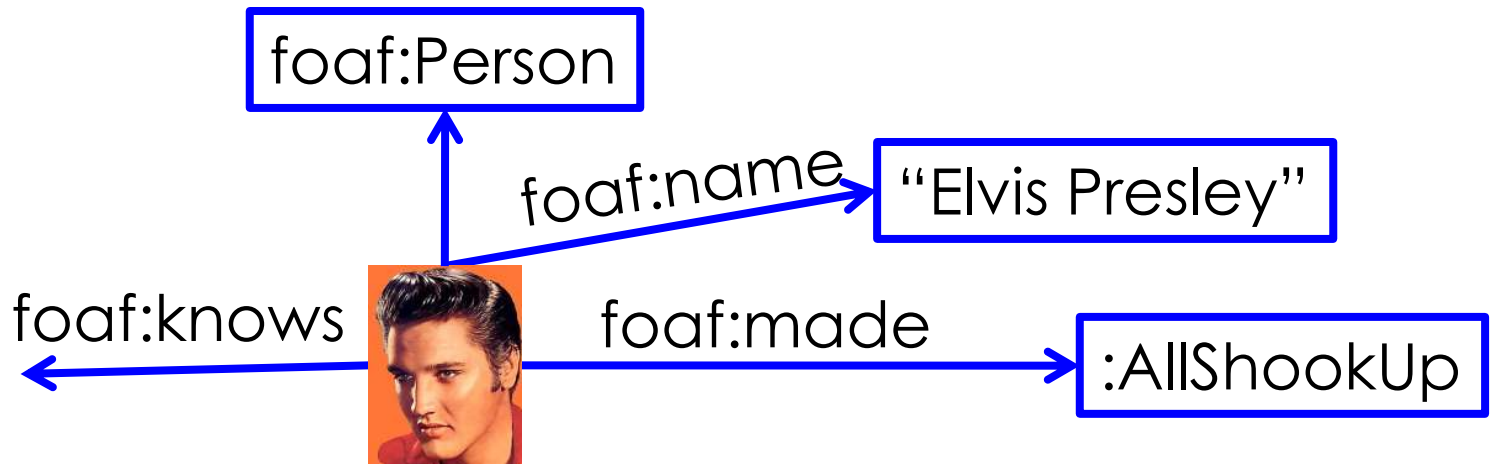
[Try this](#)

**Creative Commons** is a non-profit organization, which defines very popular licenses, notably

- CC-BY: Free for reuse, just give credit to the author
- CC-BY-NC: Free for reuse, give credit, non-commercial use only
- CC-BY-ND: Free for reuse, give credit, do not create derivative works

# Sharing: FOAF

foaf: Friend Of A Friend  
(predicates for relationships between people)  
<http://xmlns.com/foaf/0.1/>



Google launched <http://rdf.data-vocabulary.org> ,  
which provides vocabulary for addresses and other  
personal information.

# The Semantic Web

The Semantic Web provides standards to

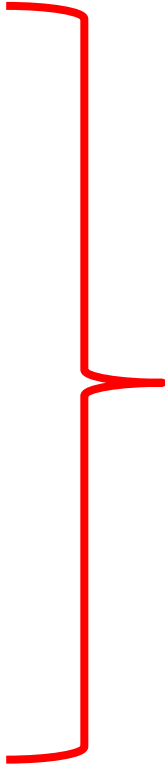
- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- ➔ Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)

# The Goal of OWL

RDFS just allows us to define classes and subclasses with very limited inference.

Can we go further?

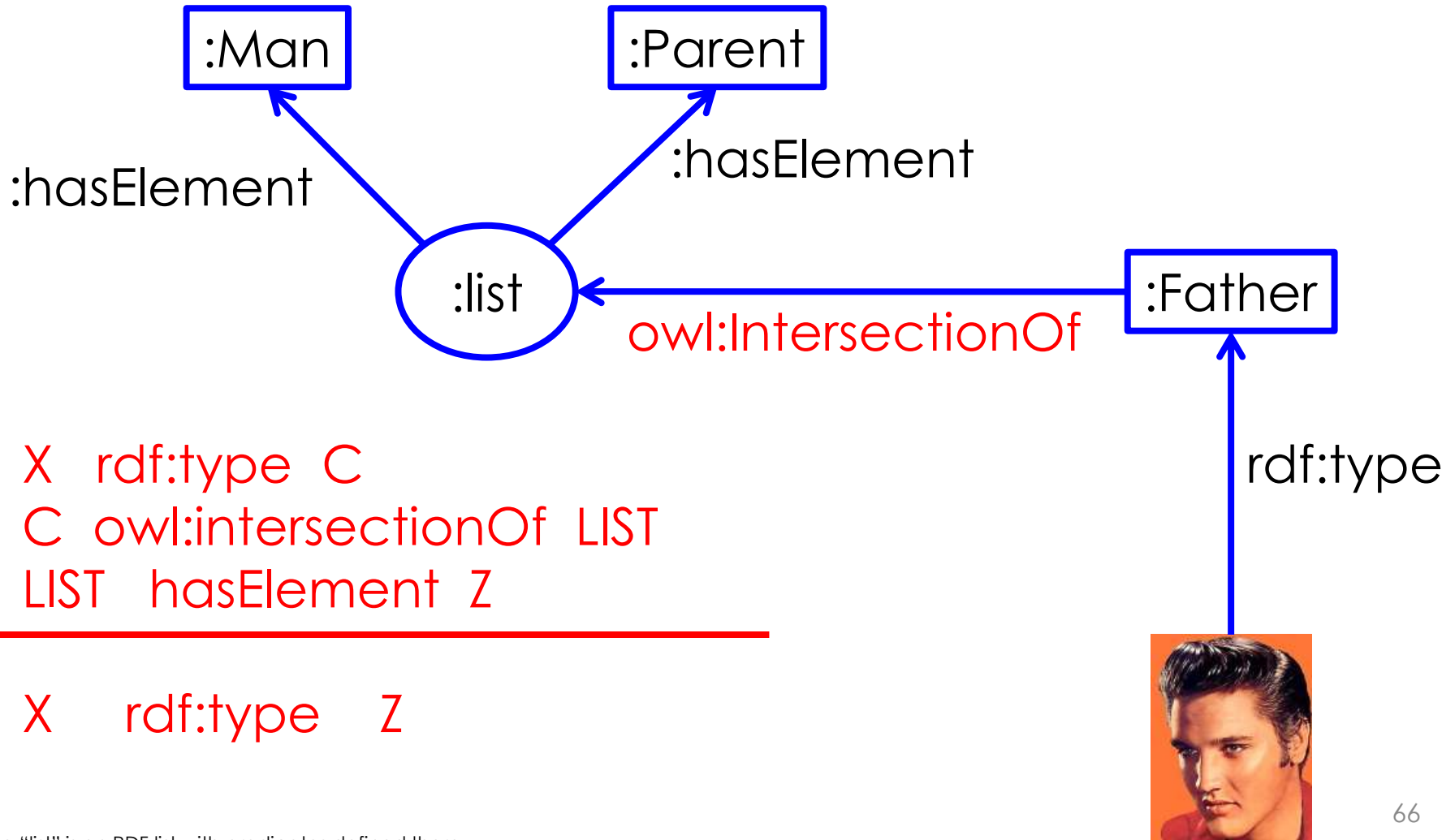
- Reasoning  
If X is left of Y and Y is left of Z  
then X is left of Z
- Class definitions  
The class of husbands is  
the class of married men
- Class properties  
People and tables are two  
disjoint classes



Goal of the  
**Web Ontology  
Language  
(OWL)**

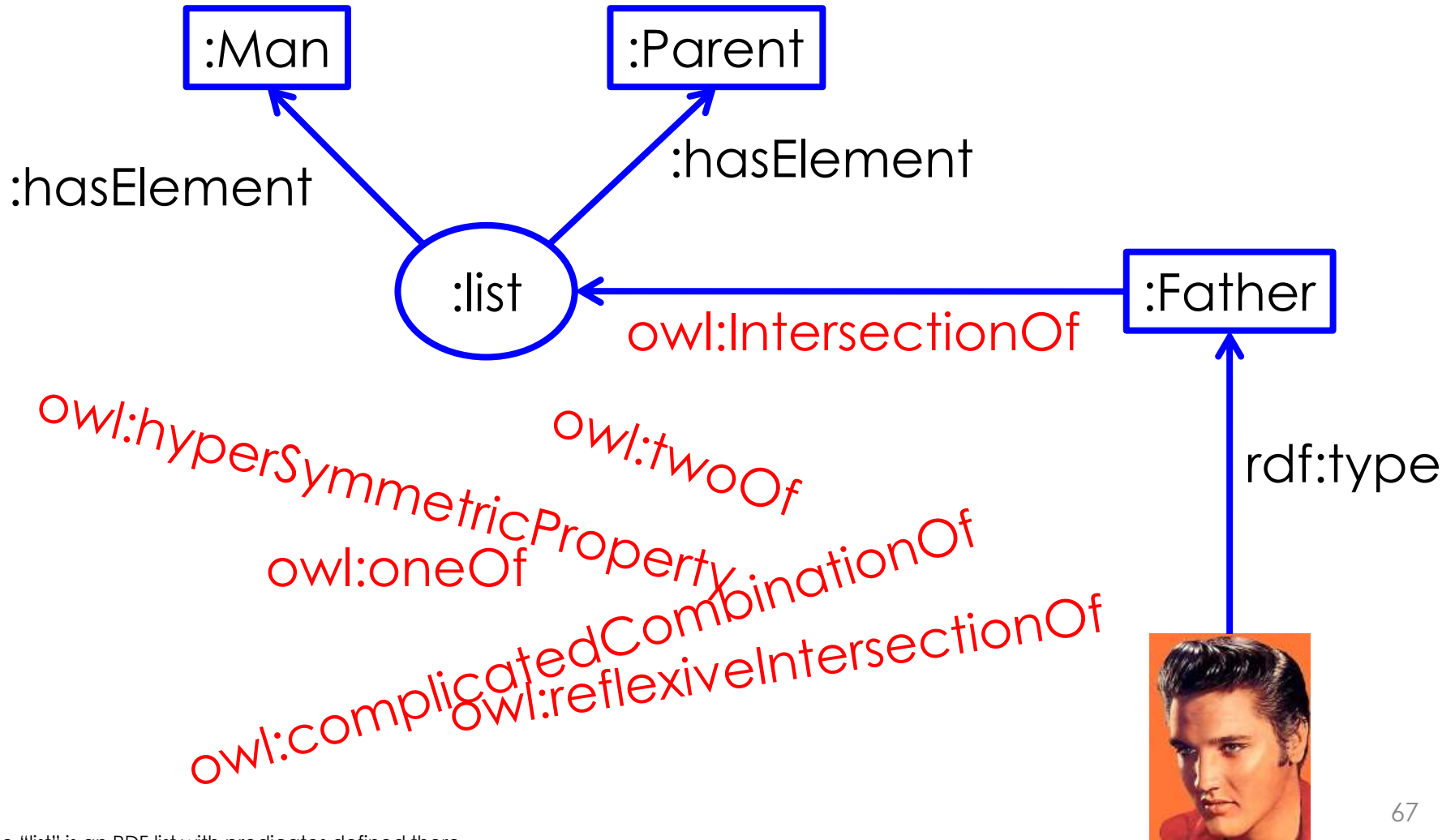
# OWL Vocabulary

OWL is a namespace that defines predicates with certain semantic rules.



# OWL Undecidability

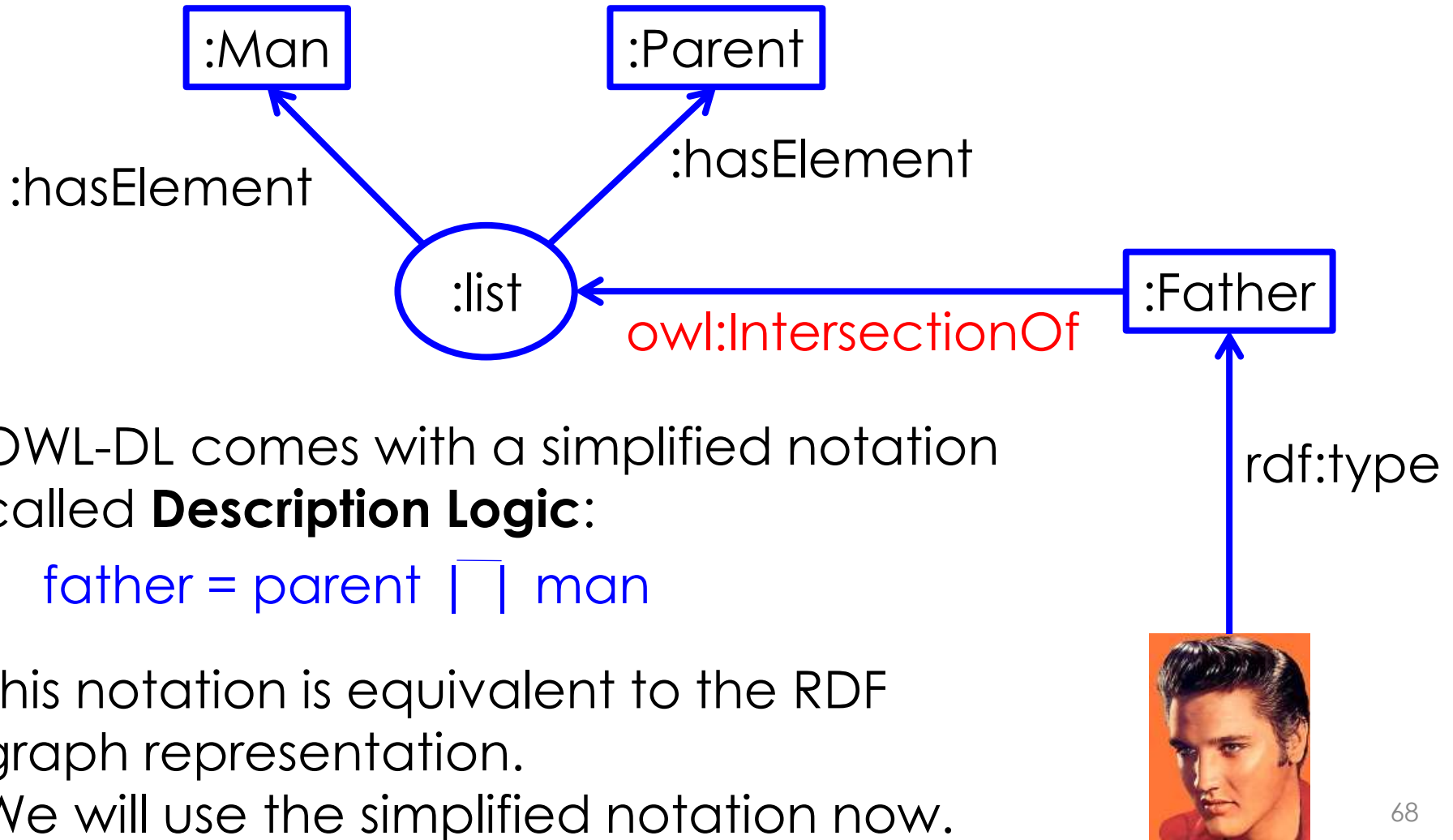
OWL defines so powerful predicates that it is **undecidable**.



The "list" is an RDF list with predicates defined there

# OWL-DL: Goal

OWL-DL is a subset of OWL that is decidable.



OWL-DL comes with a simplified notation called **Description Logic**:

$\text{father} = \text{parent} \sqcap \text{man}$

This notation is equivalent to the RDF graph representation.

We will use the simplified notation now.

# Intersection

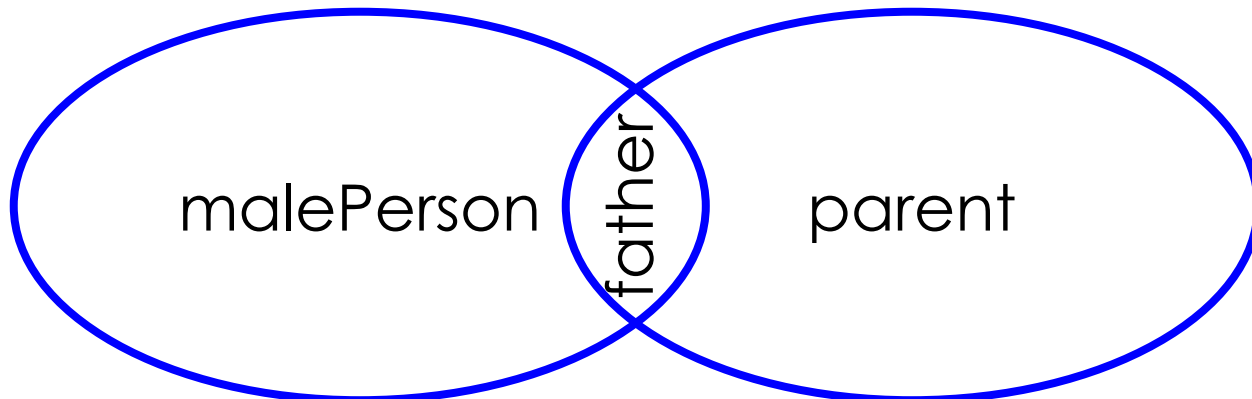
OWL-DL is primarily concerned with describing sets of entities:

$X \sqcap Y$  The class of things that are in both X and Y

$\text{father} = \text{parent} \sqcap \text{malePerson}$

This corresponds to the First-Order Formula

$\forall x: \text{father}(x) \iff \text{parent}(x) \wedge \text{malePerson}(x)$



# Union, Intersection, Negation

- $X \cup Y$  The class of things that are in X or in Y
- $X \cap Y$  The class of things that are in both X and Y
- $\sim X$  The class of things that are not in X

person, parent, hardRockSinger, softRockSinger,  
happyPerson, marriedPerson, malePerson

rockSinger = hardRockSinger  $\cup$  softRockSinger

unmarried-rock-singing-father

= (parent  $\cap$  man)  $\cap$  (hardRockSinger  $\cup$  softRockSinger)  
 $\cap$   $\sim$ married

non-rock-singing-person

= person  $\cap$   $\sim$ (hardRockSinger  $\cup$  softRockSinger)

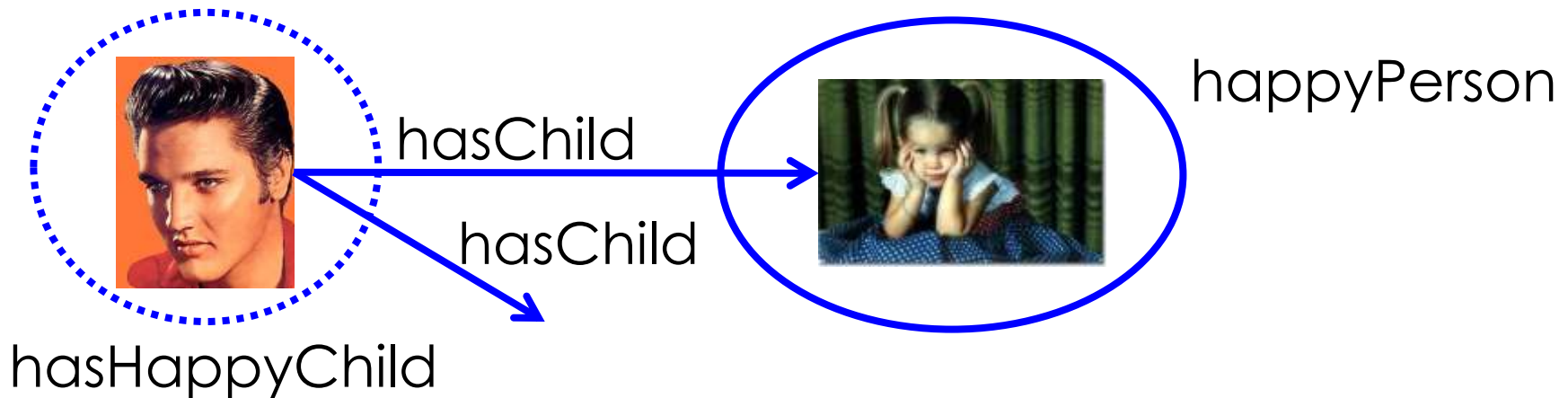
# Restrict

R: A predicate/role

C: a class

- $\forall$  R.C     The class of things where all R-links lead to a C
- $\exists$  R.C     The class of things where there is a R-link to a C

has-happy-child =  $\exists$  hasChild.happyPerson



This corresponds to the First Order Logic formula:

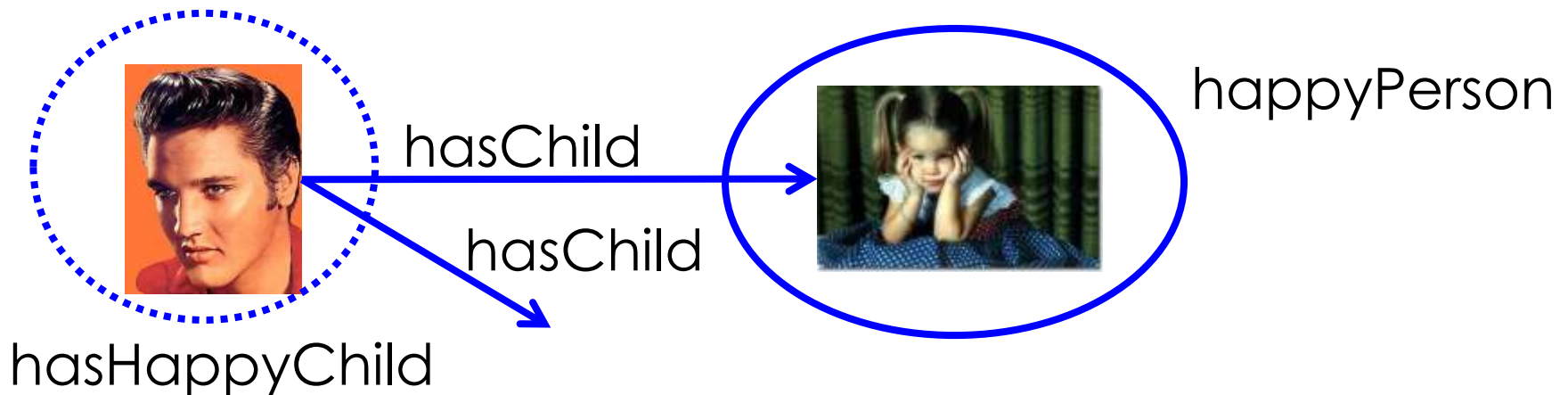
$\forall x: \text{has-happy-child}(x)$

$\Leftrightarrow \exists y: \text{hasChild}(x,y) \wedge \text{happyPerson}(y)$

# Restrictions

- $\forall$  R.C     The class of things where all R-links lead to a C
- $\exists$  R.C     The class of things where there is a R-link to a C

has-only-happy-children =  $\forall$  hasChild.happyPerson



This corresponds to the First Order Logic formula:

$\forall x: \text{has-only-happy-children}(x)$

$\Leftrightarrow \forall y: \text{hasChild}(x,y) \Rightarrow \text{happyPerson}(y)$

# Restrictions

- $\forall$  R.C      The class of things where all R-links lead to a C
- $\exists$  R.C      The class of things where there is a R-link to a C

singer-with-happy-child =

singer  $\sqcap$   $\exists$  hasChild.happyPerson

singer-with-only-happy-children =

singer  $\sqcap$   $\forall$  hasChild.happyPerson

# Subclass Assertions

$X \sqsubseteq Y$     X is a subclass of Y (everything in X is also in Y)

singer  $\sqsubseteq$  person

This corresponds to the First Order Logic formula:

$\forall x: \text{singer}(x) \Rightarrow \text{person}(x)$

happysingers  $\sqsubseteq$  singer  $\sqcap$   $\forall$  hasChild.happyPerson

# Type Assertions

$a: C$      $a$  is of type  $C$  ( $a$  is in  $C$ )

elvis: singer

This corresponds to the First Order Logic formula:

singer(elvis)

elvis : singer  $\sqcap$   $\forall$  hasChild.happyPerson

# Fact Assertions

$(X,Y): R$       X and Y stand in relationship R

$(\text{elvis}, \text{lisa}): \text{hasChild}$

This corresponds to:  $\text{hasChild}(\text{elvis}, \text{lisa})$

$(\text{elvis}, \text{priscilla}): \text{marriedTo}$

This corresponds to:  $\text{marriedTo}(\text{elvis}, \text{priscilla})$

# Exercise

Class constructors:

$X \sqcap Y$

The class of things that are in both X and Y

$X \sqcup Y$

The class of things that are in X or in Y

$\sim X$

The class of things that are not in X

$\forall R.C$

The class of things where all R-links lead to a C

$\exists R.C$

The class of things where there is a R-link to a C

Assertions:

$X \sqsubseteq Y$

X is a subclass of Y (everything in X is also in Y)

$a:C$

a is a thing in the class C

$(a,b):R$

a and b stand in the relation R, i.e.,  $R(a,b)$

*Assume the classes: male, person, happyPerson  
and the predicates: marriedTo, hasChild*

- build the class of married people*
- build the class of people married to at least 1 happy pers*
- build the class of happy male married people*
- say that married people are happy*

# Cardinality Restrictions

$\geq n R$  The class of those who have more than  $n$  outgoing  $R$  links

$\leq n R$  The class of those who have less than  $n$  outgoing  $R$  links

singers with more than 10 children:

$\text{singer} \mid \bar{\mid} \geq 10 \text{ hasChild}$

people with exactly 10 children



# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class

$\exists$  hasChild.female

“Those that have  
a female child”



# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class
- having all links of one type leading to a certain class

“Those that  
have only female  
children”

$\forall$  hasChild.female



# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class
- having all links of one type leading to a certain class
- having at most or at least  $n$  links of a certain type

$\geq 2$  hasChild

“Those that  
have more than  
1 child”

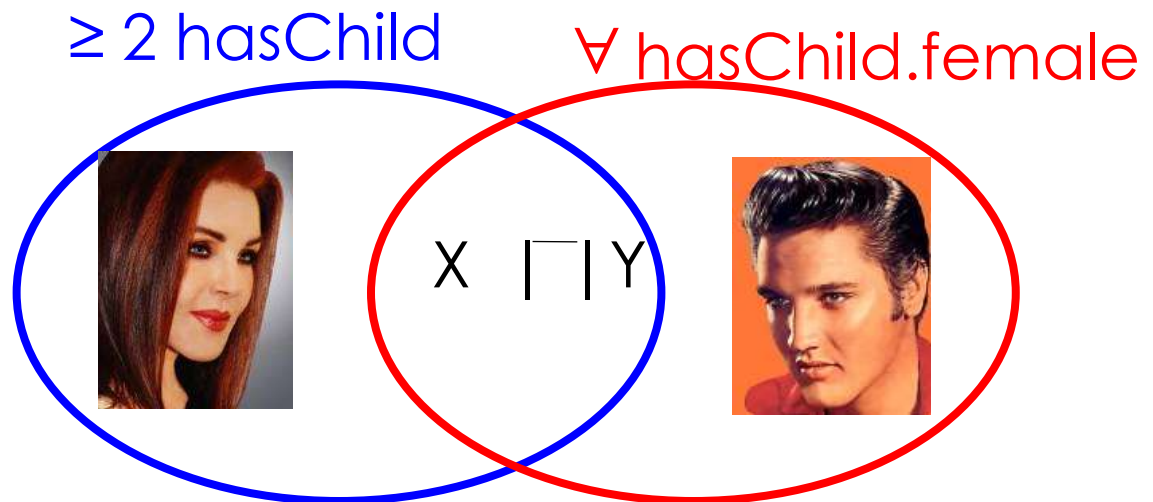


# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class
- having all links of one type leading to a certain class
- having at most or at least n links of a certain type
- being in two classes at the same time

“Those that have more than 1 child and have only female children”



# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class
- having all links of one type leading to a certain class
- having at most or at least n links of a certain type
- being in two classes at the same time
- belonging to such a class

elvis:  $\forall$  hasChild.female

$\forall$  hasChild.female



# OWL-DL

OWL-DL basically reasons about properties such as

- having a link to an element of a class
- having all links of one type leading to a certain class
- having at most or at least n links of a certain type
- being in two classes at the same time
- belonging to such a class
- standing in a relationship



(elvis, pricilla): marriedTo

# OWL-DL

OWL-DL infer properties to make the knowledge base consistent:

(priscilla, navarone):hasChild

priscilla:  $\forall$  hasChild.female

$\Rightarrow$  navarone:female

This may fail, of course:

navarone:male

male  $\sqcap$  female  $\sqsubseteq \perp$

$\forall$  hasChild.female



$\Rightarrow$  Whatever other children Priscilla has will become girls.

# Reasoning Tasks

Classical reasoning tasks in OWL-DL:

- Is the knowledge base consistent?

singer  $\sqsubseteq$  person

dog  $\sqsubseteq$   $\sim$ person

bello:singer

bello:dog

X

# Reasoning Tasks

Classical reasoning tasks in OWL-DL:

- Is the knowledge base consistent?
- Is one class a subclass of another class?

$\exists \text{ hasChild.}\{Lisa\} \sqsubseteq \text{singer} \quad ?$

(Are all parents of Lisa singers?)

can be reduced to the consistency problem:

$(bob, Lisa) : \text{hasChild}$   
 $bob : \sim \text{singer}$

More precisely, we ask: Does it follow necessarily from the KB that one class is a subclass of the other?  
If so, we get a contradiction here.

# Reasoning Tasks

Classical reasoning tasks in OWL-DL:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?

elvis:singer ?

can be reduced to the consistency problem:

elvis:~singer

More precisely, we ask: Does it follow necessarily from the KB that Elvis is a singer? If so, this assertion will cause a contradiction.

# OWL: OWL-DL

Classical reasoning tasks in OWL-DL:

- Is the knowledge base consistent?
- Is one class a subclass of another class?
- Is an individual an instance of a class?
- Does an individual have a certain property?

$\text{elvis: } \exists \text{sings. goodSong} \quad ?$

can be reduced to the consistency problem:

$\text{elvis: } \sim \exists \text{sings. goodSong}$

More precisely, we ask: Does it follow necessarily from the KB that Elvis sings at least one good song? If so, this assertion will cause a contradiction.

# OWL-DL Summary

**OWL-DL** is a decidable subset of OWL.

It is based on the **description logic**  $\mathcal{SHOIN}^{(D)}$ , a formalism that allows describing properties of objects in a manner inspired by set theory.

There are a number of free OWL DL reasoners available online:

- Pellet
- FaCT++
- Prova

$\forall$  hasChild.female



# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- ➔ Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)

# SPARQL

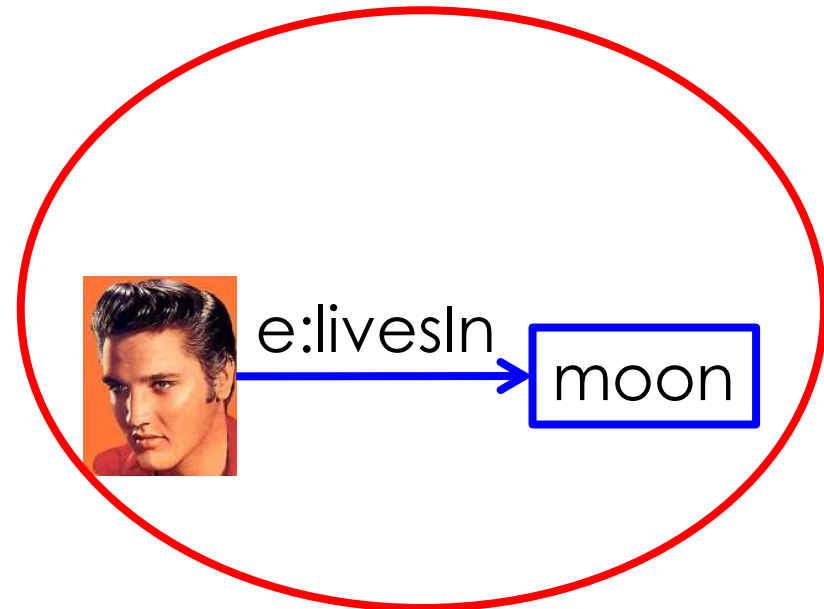
**SPARQL** (SPARQL Protocol and RDF Query Language) is the query language of the Semantic Web.

```
PREFIX e: <http://elvis.org/>
```

```
SELECT ?loc  
WHERE {  
  e:elvis e:livesIn ?loc  
}
```

Find me all the values for ?loc such that the triple is true.

Elvis, where are you?



# SPARQL Matching

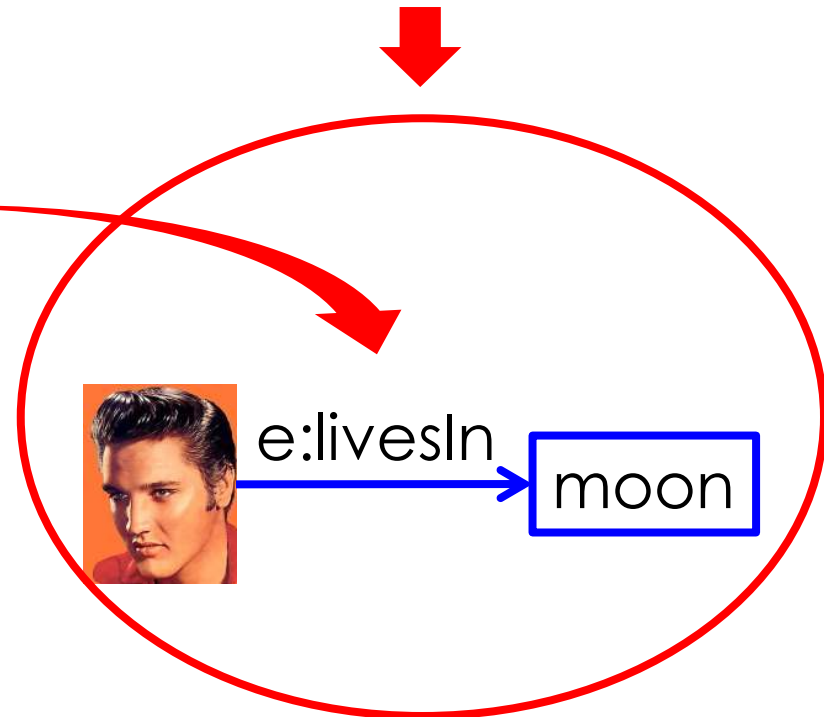
**SPARQL** (SPARQL Protocol and RDF Query Language) is the query language of the Semantic Web.

PREFIX e: <<http://elvis.org/>>

```
SELECT ?loc  
WHERE {  
  e:elvis e:livesIn ?loc  
}
```



Elvis, where are you?



SPARQL queries can be seen as sub-graph matching.

?loc = e:moon <sup>96</sup>

# SPARQL Matching

PREFIX e: <<http://elvis.org/>>

PREFIX rdf: <[http://w3c.org/...](http://w3c.org/)>

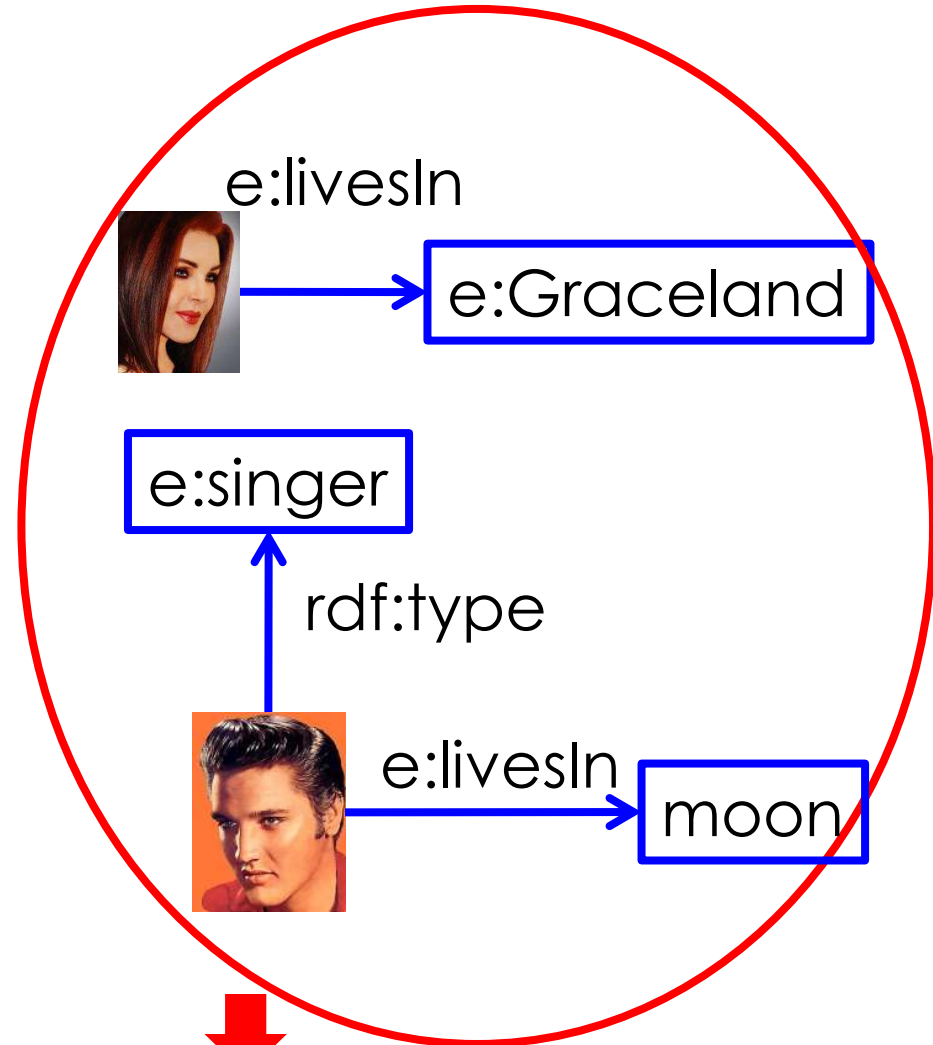
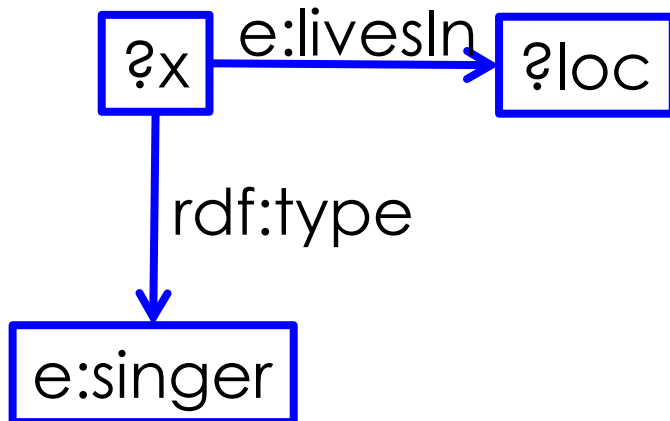
SELECT ?loc, ?x

WHERE {

?x e:livesIn ?loc.

?x rdf:type e:singer

}

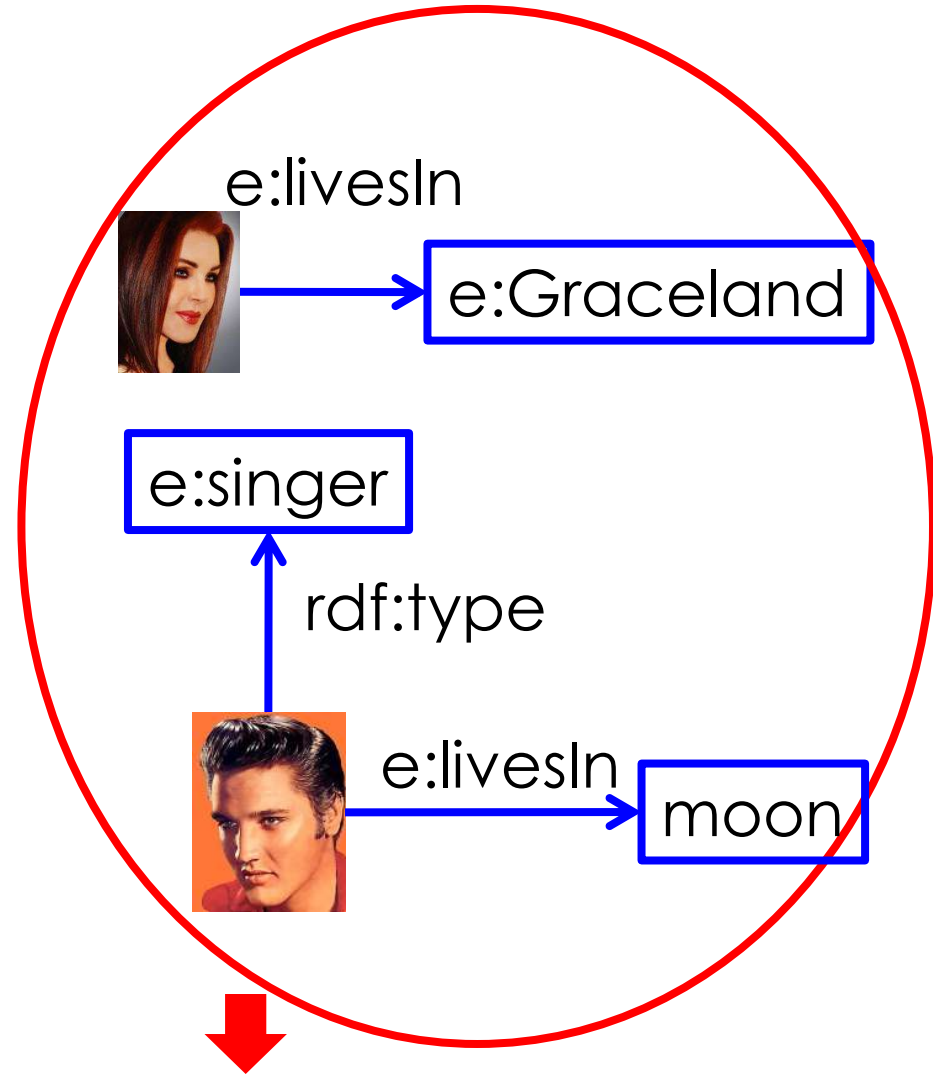
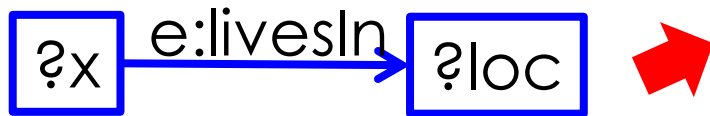


?loc = e:moon  
?x = e:Elvis

# SPARQL Result Sets

PREFIX e: <<http://elvis.org/>>  
PREFIX rdf: <[http://w3c.org/...](http://w3c.org/)>

```
SELECT ?x  
WHERE {  
  ?x e:livesIn ?loc.  
}
```



?x = e:Priscilla  
?x = e:Elvis

# SPARQL Endpoints

Many ontologies provide a “SPARQL endpoint”, i.e. a service that can

- receive SPARQL queries sent by a machine
- receive SPARQL queries typed by a human in a Web interface

<http://esw.w3.org/SparqlEndpoints>

## Currently Alive SPARQL Endpoints

(alphabetical. let's avoid [PoorMansHypertext](#) and in-your-face URIs, please)

Project	status	SPARQL endpoint	Webform	c
<a href="#">BBC Programmes and Music</a>	(2010-06-29) alive	<a href="#">endpoint</a>	<a href="#">Ajax based Visual Query Builder</a>	P E
<a href="#">Bio2RDF</a>	(2010-01-07) alive	<a href="#">List of 40 SPARQL endpoints</a>	n/a	u
<a href="#">BioGateway</a>	(2010-01-07)	<a href="#">endpoint</a>	<a href="#">webform</a>	B b

# SPARQL Example

Example at <http://dbpedia-live.openlinksw.com/sparql/> :

```
select distinct ?x {  
  <http://dbpedia.org/resource/Elvis_Presley>  
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
  ?x  
}  
limit 100
```



x
<a href="http://dbpedia.org/class/yago/Person100007846">http://dbpedia.org/class/yago/Person100007846</a>
<a href="http://dbpedia.org/class/yago/AmericanMaleSingers">http://dbpedia.org/class/yago/AmericanMaleSingers</a>
<a href="http://dbpedia.org/class/yago/UnitedStatesArmySoldiers">http://dbpedia.org/class/yago/UnitedStatesArmySoldiers</a>
<a href="http://dbpedia.org/class/yago/SunRecordsArtists">http://dbpedia.org/class/yago/SunRecordsArtists</a>
<a href="http://dbpedia.org/class/yago/LasVegasMusicians">http://dbpedia.org/class/yago/LasVegasMusicians</a>
<a href="http://www.w3.org/2002/07/owl#Thing">http://www.w3.org/2002/07/owl#Thing</a>
<a href="http://sw.opencyc.org/2008/06/10/concept/Mx4rvVjWoZwpEbGdrcN5Y29ycA">http://sw.opencyc.org/2008/06/10/concept/Mx4rvVjWoZwpEbGdrcN5Y29ycA</a>
<a href="http://sw.opencyc.org/2008/06/10/concept/Mx4rwJ45PHS7EdaAAACgyZzFrg">http://sw.opencyc.org/2008/06/10/concept/Mx4rwJ45PHS7EdaAAACgyZzFrg</a>
<a href="http://dbpedia.org/class/yago/AmericanFilmActors">http://dbpedia.org/class/yago/AmericanFilmActors</a>

# SPARQL Summary

**SPARQL** (SPARQL Protocol and RDF Query Language) is the query language of the Semantic Web.

```
PREFIX e: <http://elvis.org/>  
PREFIX rdf: http://w3c.org/...  
PREFIX g:<http://g-a.com...>
```

```
SELECT ?x  
WHERE {  
  ?x e:won g:Grammy  
}
```

SPARQL borrows concepts from SQL, but is specially adapted to, distributed RDF graphs.

Many ontologies provide “SPARQL endpoints”, i.e. a service that can

- receive SPARQL queries sent by a machine
- receive SPARQL queries typed by a human in a Web interface

# The Semantic Web

The Semantic Web provides standards to

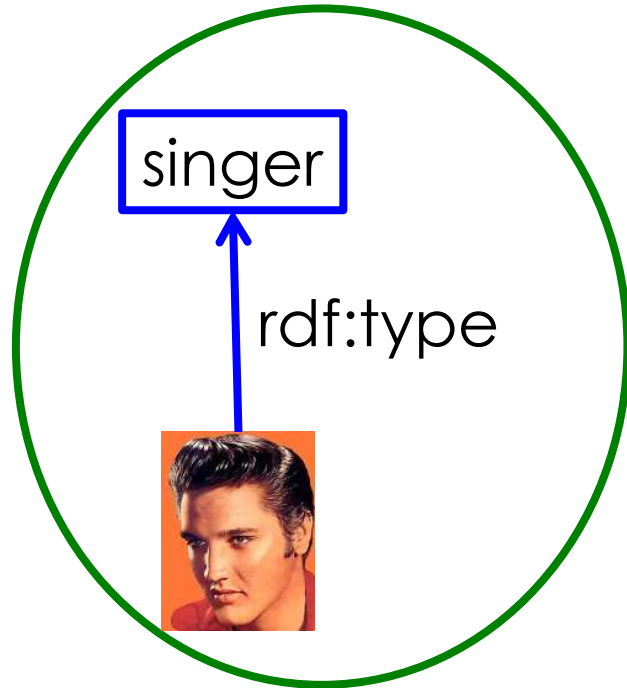
- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)

 Link data

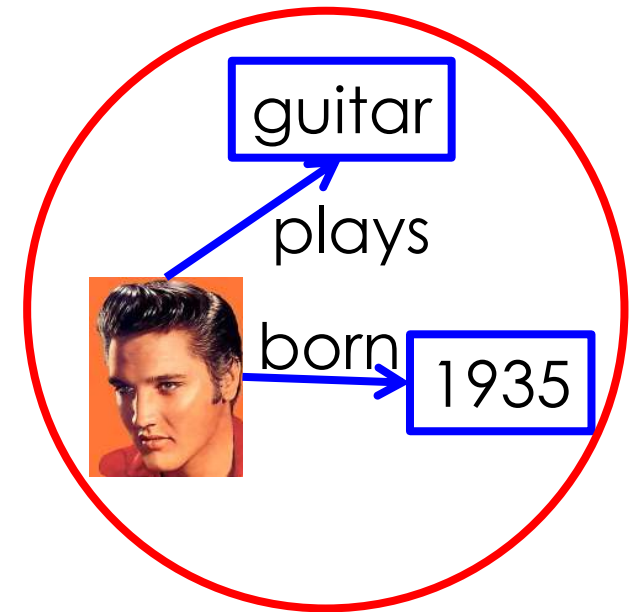
- Publish data (RDFa)

# Linked Data Problem

Many ontologies talk about the same entity with different URIs.



Elvisopedia  
(<http://elvisopedia.org/>)

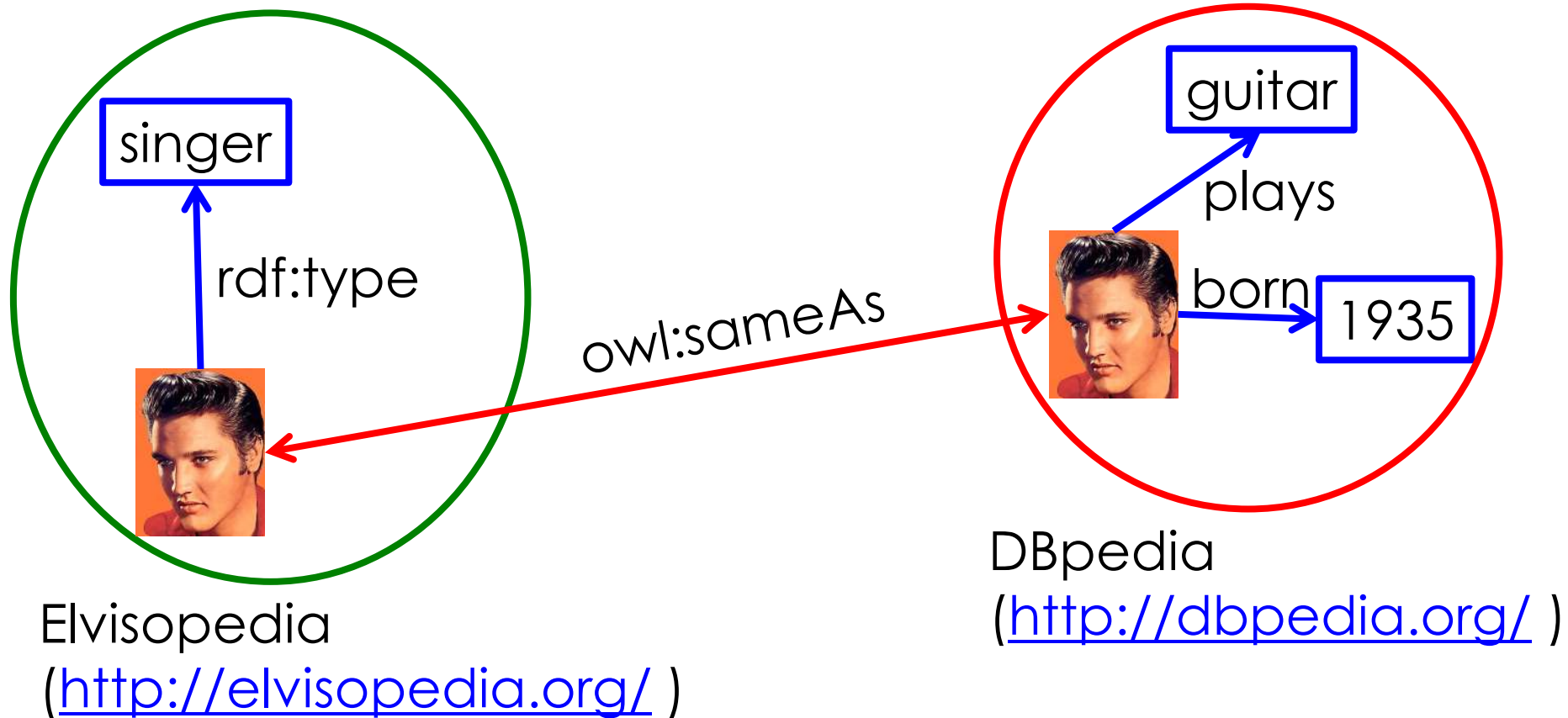


DBpedia  
(<http://dbpedia.org/>)

This is bad, because we cannot join the information.

# Linked Data Solution

OWL provides vocabulary to link equivalent entities



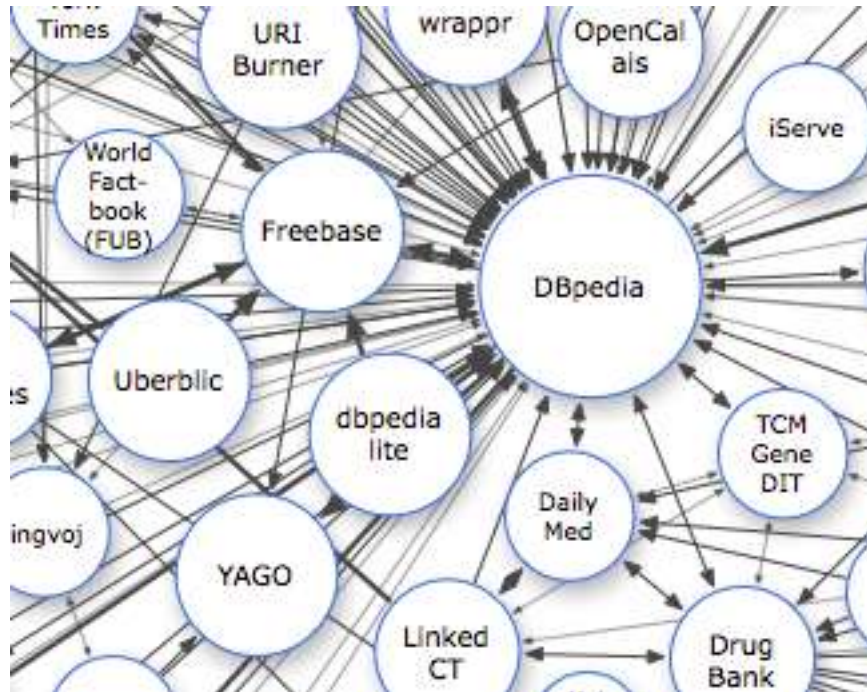
<http://elvisopedia.org/Elvis>

owl:sameAs

<http://dbpedia.org/Elvis>

# The Linking Data Project

The **Linking Open Data Project** aims to interlink all open RDF data sources into one gigantic RDF graph ([link](#)).





# Existing Ontologies

The existing ontologies in the Linked Data Cloud include ( <http://www4.wiwiss.fu-berlin.de/lodcloud/> )

- US census data
- BBC music database
- Gene ontologies
- DBpedia general knowledge, + YAGO, + Cyc etc.
- UK government data
- geographical data in abundance
- national library catalogs (USA, Germany etc.)
- publications (DBLP)
- commercial products
- all Pokemons
- ...and many more

# The Semantic Web


The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data

 Publish data (RDFa)

# And the rest of the Web?

Homepage



**Gerhard Weikum**  
[Max-Planck-Institut für Informatik](#)  
[Department 5: Databases and In](#)  
[Building E1.4, Room 402](#)  
[Campus E1.4](#)  
[66123 Saarbrücken](#)  
[Germany](#)

**Email:** [Get my email address via](#)  
**Phone:** +49 681 9325 500  
**Fax:** +49 681 9325 599



## Nikon - Coolpix 12.1-Megapixel Digital Camera - Black

Model: L110 Black | SKU: 9758692

15x optical/4x digital zoom; 3" HVGA TFT-LCD display; Hybrid VR image stabilization; PictBridge compatible

Compare

★★★★☆ 4.3 of 5 (102 reviews)

[Check Shipping & Availability](#) ▶

## Le 13 juillet place de la Bastille

Le 13 juillet, plus de 15 artistes d'exception vous attendent à partir de 20h30, place de la Bastille. (transmis en direct sur France Ô).

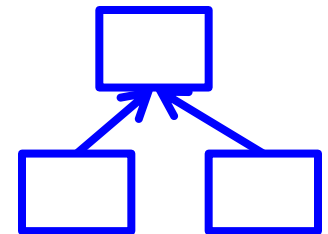


La Mairie de Paris en partenariat avec France Télévisions et Electron Libre, présente le **Concert de la diversité.**

Ce concert **gratuit**, placé sous le signe de l'éclectisme, du divertissement et du partage,



?



# RDFa

**RDFa** is a W3C standard to annotate HTML pages with RDF data.

```
<div>
```

```
  Martin Thunderbird<br>
```

```
  Researcher in Rock'N'Roll Music of 1935-1977<br>
```

```
  3764 Presley Boulevard<br>
```

```
  Memphis, Tennessee
```

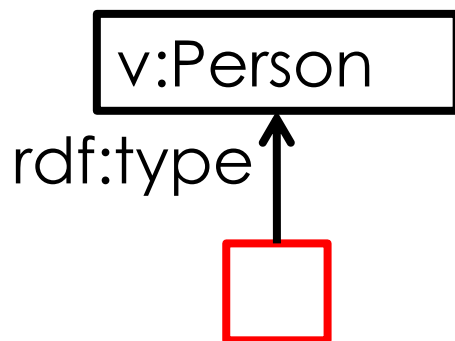
```
</div>
```

Namespace declaration  
(here: a namespace  
provided by Google)

RDFo

Everything inside this  
<div> becomes a node  
of type v:Person

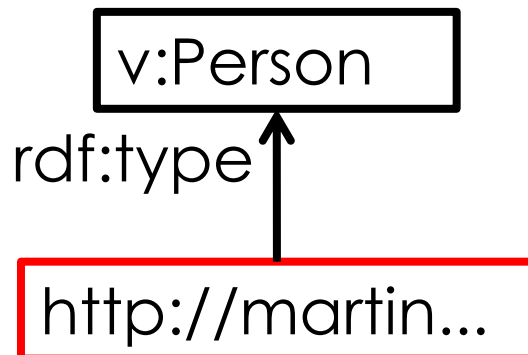
```
<div xmlns:v="http://rdf.data-vocabulary.org/#"  
      typeof="v:Person">  
  Martin Thunderbird<br>  
  Researcher in Rock'N'Roll Music of 1935-1977<br>  
  3764 Presley Boulevard<br>  
  Memphis, Tennessee  
</div>
```



# Naming a node

We can also give the URI of the node.

```
<div xmlns="http://rdf.data-vocabulary.org/#"
      typeof="v:Person"
      about="http://martin-thunderbird.com/me">
  Martin Thunderbird<br>
  Researcher in Rock'N'Roll Music of 1935-1977<br>
  3764 Presley Boulevard<br>
  Memphis, Tennessee
</div>
```

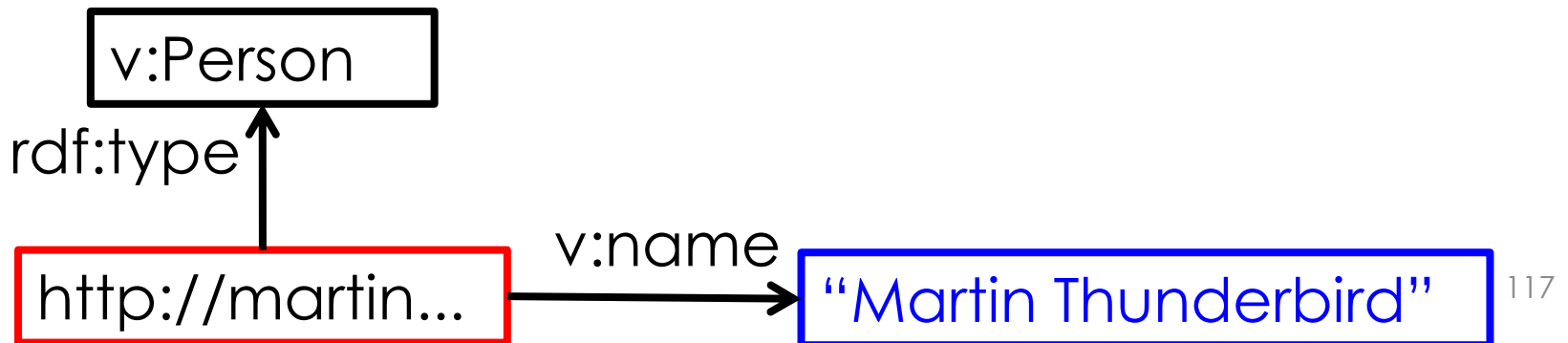


# Statements

Statements are constructed with "property="

Text becomes a string node in RDF

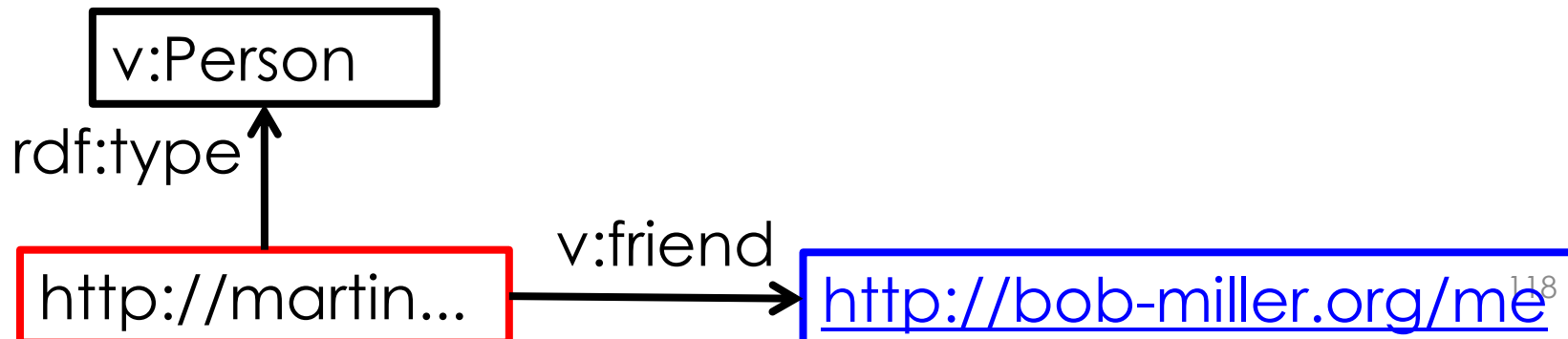
```
<div xmlns:v="http://rdf.data-vocabulary.org/#"
      typeof="v:Person"
      about="http://martin-thunderbird.com/me">
  <span property="v:name">Martin Thunderbird</span>
  Researcher in Rock'N'Roll Music of 1935-1977<br>
  3764 Presley Boulevard<br>
  Memphis, Tennessee
</div>
```



Links given by "rel" +  
"resource" become  
a URI node in RDF

# Statements

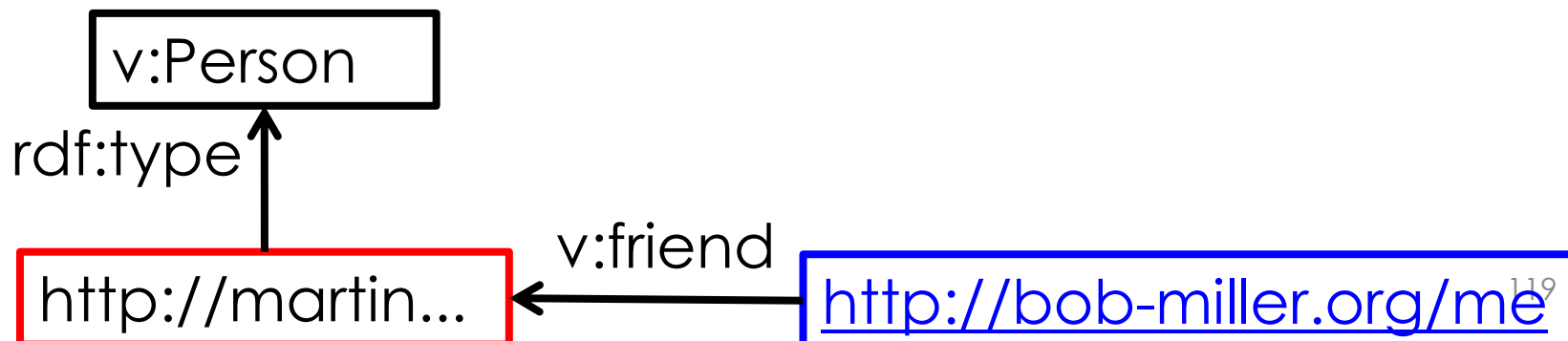
```
<div xmlns:v="http://rdf.data-vocabulary.org/#"  
      typeof="v:Person"  
      about="http://martin-thunderbird.com/me">  
  <span rel="v:friend"  
        resource="http://bob-miller.org/me">  
    Bob Miller  
  </span>  
</div>
```



# Statements

Links given by "rev"  
become statements in the  
other direction

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"  
      typeof="v:Person"  
      about="http://martin-thunderbird.com/me">  
  <span rev="v:friend"  
        resource="http://bob-miller.org/me">  
    Bob Miller  
  </span>  
</div>
```

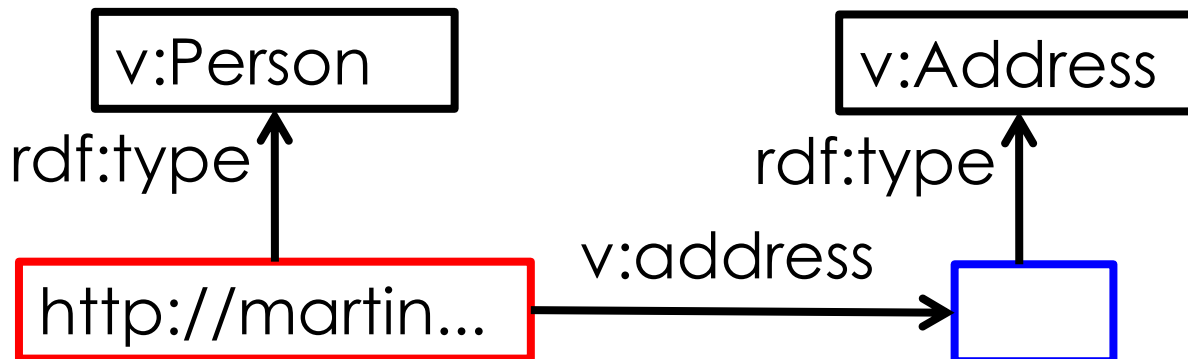


# Nodes

“rel” creates a link between the outer node and the inner node.

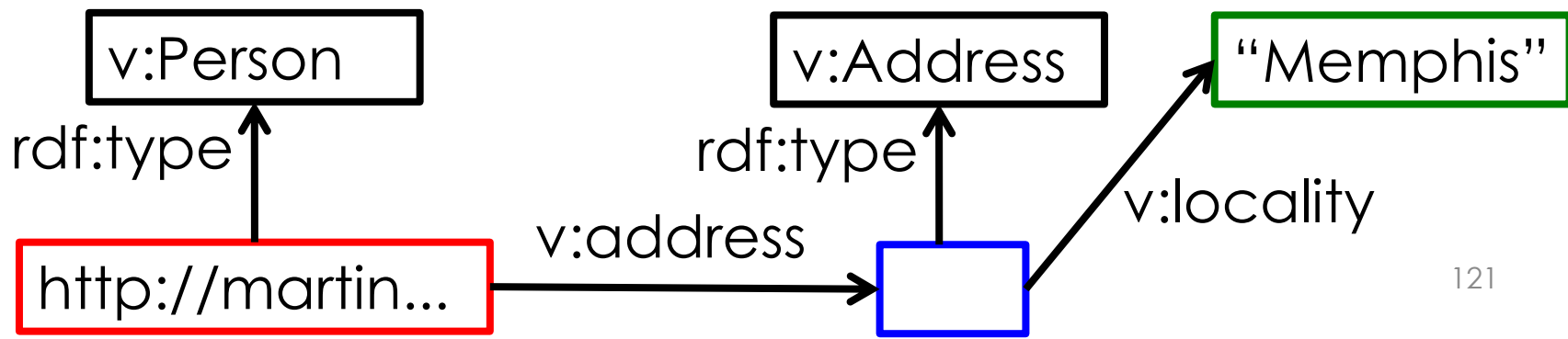
As before, typeof creates a new node

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"  
  typeof="v:Person"  
  <span rel="v:address">  
    <span typeof="v:Address">  
    </span>  
  </span>  
</div>
```



# Inner Nodes

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"  
      typeof="v:Person"  
  <span rel="v:address">  
    <span typeof="v:Address">  
      <span property="v:locality">Memphis</span>  
    </span>  
  </span>  
</div>
```



# RDFa Vocabulary

<tag xmlns:xxxx=<http://...>>...</tag>

<tag typeof=class>

<tag about=uri>

<tag property=predicate>value</tag>

<tag rel=predicate resource=uri />

<tag rev=predicate resource=uri />

<tag rel=predicate> new node </tag>

defines a namespace

defines a new blank node

defines a new node with URI

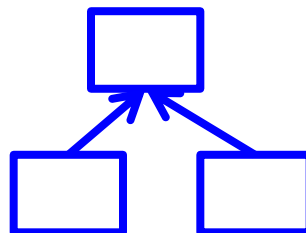
defines a link of the outer node  
to a literal

defines a link of the outer node  
to a URI

defines a link of the URI to the  
outer node

defines a link of the outer node  
to the new node

... which is isomorphic to an RDF graph...

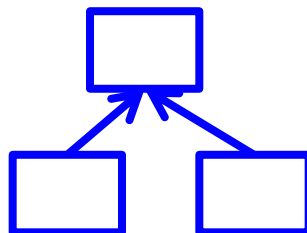


# RDFa Summary

**RDFa** is a W3C standard to annotate HTML pages with RDF data.

Advantages:

- Grass root appeal  
(everybody can start annotating pages)
- No data duplication  
(all data in one file)
- Publisher independence  
(everybody can use his own attributes)



# RDFa Example

## Contact

Fabian M. Suchanek  
[INRIA](#), bâtiment G, bureau 116  
4, rue Jacques Monod  
91893 Orsay Cedex  
France  
E-Mail: *firstName@lastName.name*  
URL: <http://suchanek.name>

Example with the [Ubiquity RDFa parser](#)  
on this [page](#)

```
<H2>Contact</H2>
<div class=indented>
  <a target=_top property="v:name">Fabian M. Suchanek</a><br>
  <a href=http://www.inria.fr/saclay target="_blank" property="v:affiliation">INRIA</a>,
  <span rel="v:address">
    <span property="v:street-address">4, rue Jacques Monod</span><br>
    <span property="v:postal-code">91893</span> <span property="v:locality">Orsay</span> C
    <span property="v:country-name">France</span><br>
  </span>
```

```
http://suchanek.name/fabian
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://rdf.data-vocabulary.org/#Person>
<http://rdf.data-vocabulary.org/#role> "researcher"
<http://rdf.data-vocabulary.org/#name> "Fabian M. Suchanek"
<http://rdf.data-vocabulary.org/#address> <bnode:http://suchanek.name/about/index_e.htmspan1>
```

[www.imdb.com/title/tt0268978/](http://www.imdb.com/title/tt0268978/)

```
<html xmlns:og=http://ogp.me/ns# >
```

...

```
<meta property='og:type' content='movie' />
```

```
<meta property='fb:app_id' content='123' />
```

...

```
</html>
```



ogp:type

Beautiful  
mind

ogp:siteName

IMDb

RDF data following the Open Graph Protocol is often embedded in HTML pages, thus allowing the Facebook LIKE button to work.

# Google & RDFa

Google has defined its own namespace, which allows annotating HTML pages with meta-information that will show up in “rich snippets”.

## [Nikon D3100 review - Digital Camera reviews -](#)

★★★★☆ Review by Gavin Stoker - Jan 10, 2011

10 Jan 2011 ... Following its release, **Nikon** proudly claim digital SLR in Europe. Its successor therefore, the **D3100**,  
[www.trustedreviews.com](http://www.trustedreviews.com) › Digital Cameras - Cached

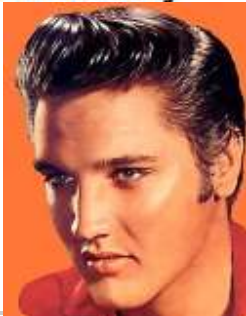
[Try it out](#)

# Sigma

**Sigma** is a Semantic Web search engine developed at the DERI Ireland. It scraped RDFa and follows owl:sameAs (<http://sig.ma>)



**Elvis Presley**



---

**fbmlpage id: 81914997331 [4]**

---

**genre: [http://dbpedia.org/resource/Rock\\_music](http://dbpedia.org/resource/Rock_music) [1]**

# Sigma

Sigma also allows the user to correct factually wrong information (such as the urban legend that Elvis would be dead).

death: 1977-08-16 [7]

death date: **hide value**

died: Memphis, Tennessee, [7]

1977-08-16 [7]

United States [7]

# The Semantic Web

The Semantic Web provides standards to

- Identify entities (URIs)
- Express facts (RDF)
- Express concepts (RDFS)
- Share vocabularies
- Describe constraints (OWL)
- Query knowledge (SPARQL)
- Link data
- Publish data (RDFa)