Semantics, knowledge graphs and ontologies in practice

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

Schedule

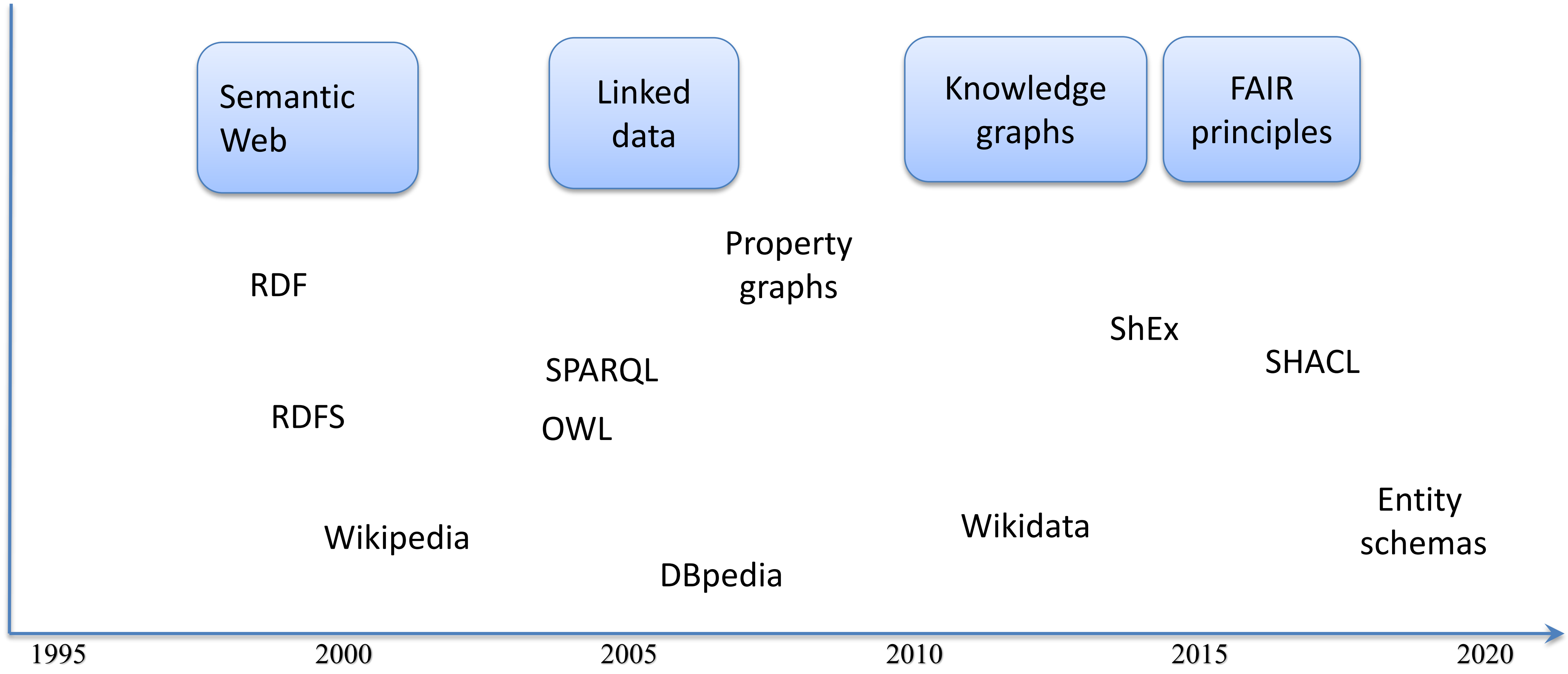
Day	Title	Topics

Day 1.	Semantic Technologies and Knowledge graphs	Semantic Web
		Linked data
		Knowledge graphs
		    RDF data model
		    Property graphs
		    Wikibase graphs
		Examples and applications

Day 2.	RDF data modelling and SPARQL	Data modelling exercises with RDF and turtle SPARQL
Day 3.	Validating RDF data	Shape Expressions (ShEx) SHACL Validating Knowledge Graphs

Day 4.	Advanced topics	ShEx and SHACL compared
		    Shape applicatons and tools
		Reasoning
		    RDFS
		    OWL
		Nanopublications

ROADMAP



Semantic Web
Linked data
Knowledge graphs
FAIR principles
RDF
RDFS
Wikipedia
SPARQL
OWL
Property graphs
DBpedia
Wikidata
ShEx
SHACL
Entity schemas
1995
2000
2005
2010
2015
2020

Session 3. Advanced topics

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

ShEx and SHACL compared

Some similarities

- Similar goal: describe and validate RDF graphs
- Both employ the word "shape"
- Node constraints similar in both languages
- Constraints on incoming/outgoing arcs
- Both allow to define cardinalities
- Both have RDF syntax
- Both have an extension mechanism

ShEx and SHACL compared

Main differences

	ShEx	SHACL
Underlying philosophy	Structure definition	Constraint checking
Syntax	Compact syntax + RDF	RDF
Notion of shape	Only structure	Structure + target decls.
Default cardinalities	{1,1}	{0,*}
Shapes and inference	No	SHACL specific entailment
Recursion	Part of the language	Undefined
Repeated properties	Part of the language	Conjunction by default Requires qualifiedValueShapes
Property paths	Nested shapes	SPARQL like
Property pair comparisons	Unsupported in current version	Part of the language
Extension mechanism	Semantic actions	SHACL-SPARQL
Validation triggering	Query shape map	Target declarations
Result of validation	Result shape map	Validation report

WESO

ShEx: Shape Expressions

Describe RDF data

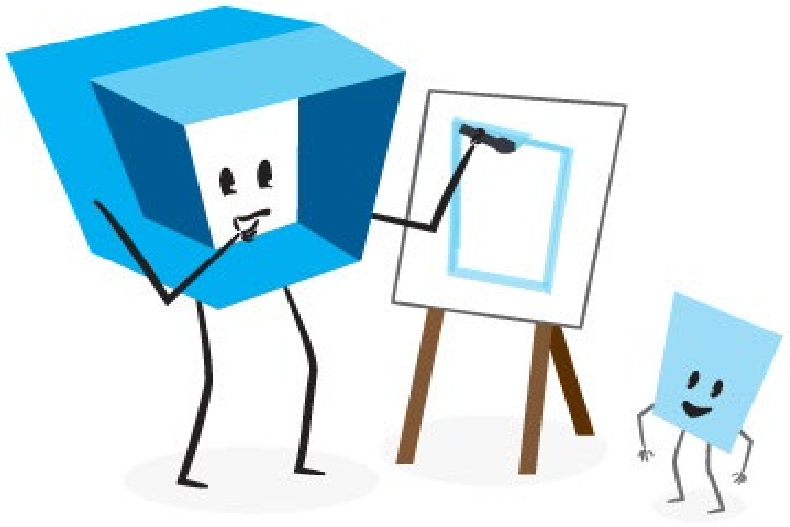
Descriptions focus on what is the shape of the RDF data graph

Focus = RDF graph

Validate = check if data matches the descriptions

Main focus = nodes that match (shape maps)

It can also check nodes that don't match



Description

SHACL: Shapes Constraint Language

Constraints on RDF data

Constraints focus on the RDF data graph and things we don’t allow

Focus = RDF graph (data)

Validate = check if data doesn’t violate the constraints

Focus = nodes that don’t pass the constraints (violations)

But it can also check nodes that pass the constraints

Constraint

ShEx for Property graphs

- Recent paper: "ProGS: Property graph shapes language"
  - https://arxiv.org/abs/2107.05566
  - Extends SHACL to support Property graphs
- In the same way, PShEx has been defined as a ShEx extension to support Property graphs
  - Adds constraints on nodes/property qualifiers
  - Proposal recently published at: https://arxiv.org/abs/2110.11709

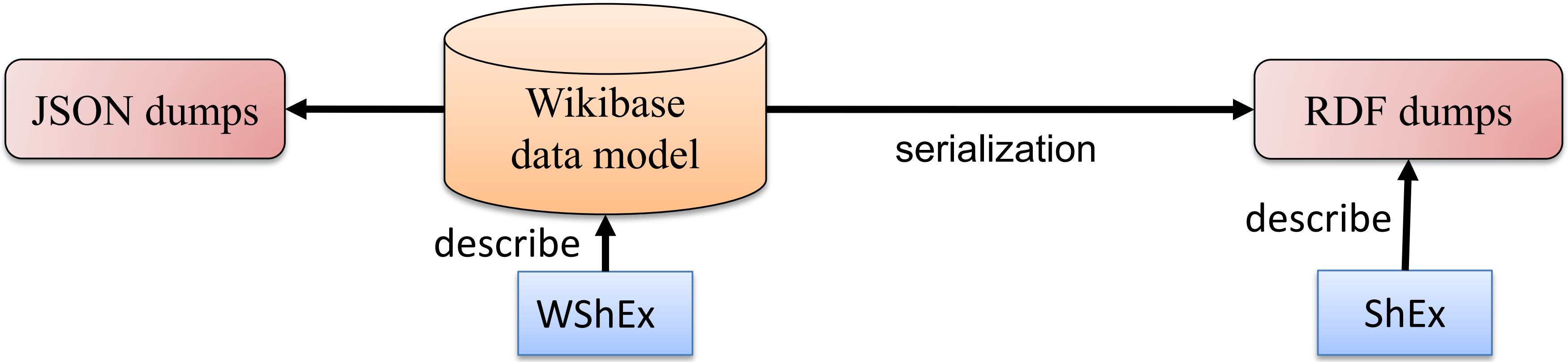
ShEx for Wikibase graphs

Wikidata already added support for ShEx

- Entity schemas extension

WShEx = extension of ShEx that supports qualifiers/references

- Can be used to describe and validate Wikibase graphs
- Defined in: https://arxiv.org/abs/2110.11709

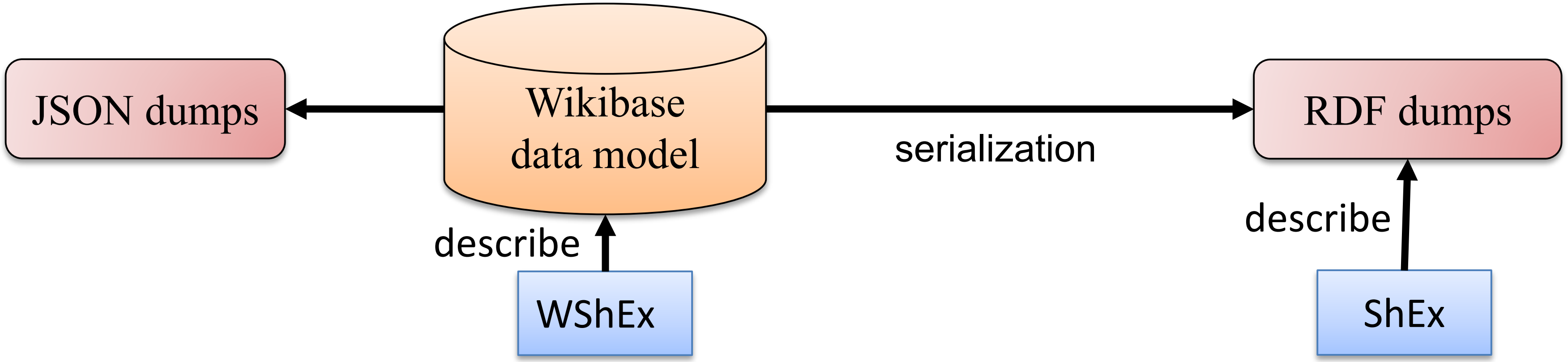


WShEx

```
<Researcher> {
 <birthPlace>       @<Place> ;
 <awarded>          @<Award> {{
    <togetherWith> @<Researcher> *
  }} *
}
<Place> {
 <country>          @<Country>
}
<Country> {}
```

ShEx

```
<Researcher> {
 wdt:birthPlace @<Place> ;
 p:birthPlace    {
    ps:birthPlace   @<Place>
 } ;
 wdt:awarded        @<Awarded> ;
 p:awarded {
   ps:awarded       @<Awarded> ;
   pq:togetherWith @<Researcher> *;
 } *
}
<Place> {
 wdt:country        @<Country> ;
 p:country {
  ps:country        @<Country> }
}
<Country> {}
```



Publishing RDF on the Web

- Publish directly in an RDF serialization
  - Turtle, JSON-LD, RDF/XML...
  - Follow linked data principles
  - Content negotiation
- Embedding in HTML
  - Using a script
  - RDFa
  - Microdata

JSON-LD script in HTML

```html
<html>
 <head>
 <title>Meeting</title>
 <script type="application/ld+json">
   {
     "@context": "http://schema.org",
     "@type": "Event",
     "name": "Meeting",
     "startDate" : "2021-12-04T12:30",
       "location" : {
         "@type" : "Place",
         "name" : "School of Computer Science",
         "address" : "C/Valdés Salas S/N, Oviedo"
       }
     }
 </script>
</head>
<body>
 <h1>Meeting</h1>
 <p>We will have a meeting next saturday at the School</p>
</body>
</html>
```

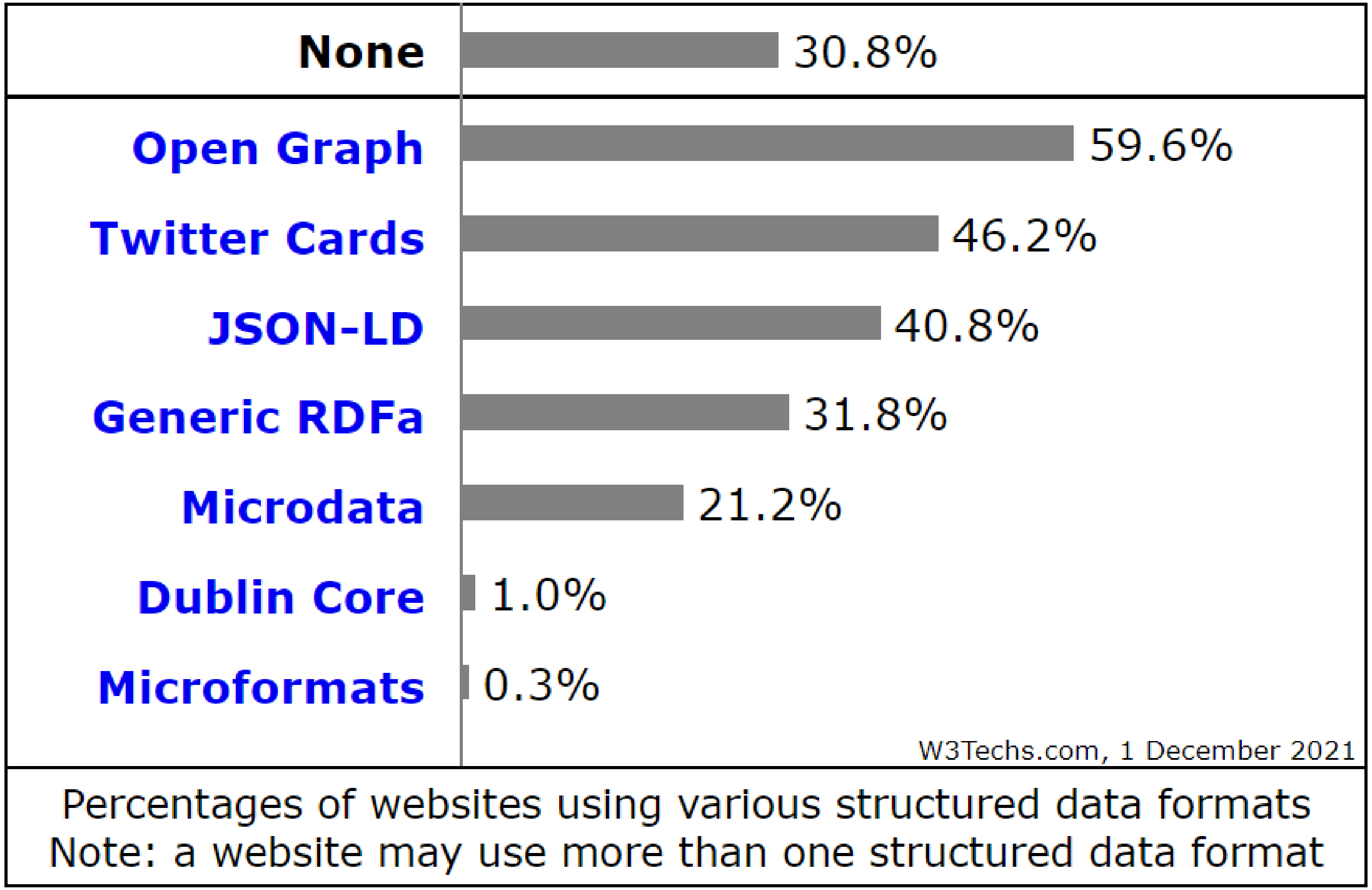
RDFa

```
<html vocab="http://schema.org/">
 <head>
 <title>Meeting</title>
 </head>
<body>
 <h1>Meeting</h1>
 <p>We will have a
  <div typeof="Event" about="http://example.org/event1"
       property="name" value="Meeting">meeting
   <span property="startDate" value="2021-12-04T12:30">next saturday</span>
   at <span property="location" typeof="Place"><span property="name">the School
   </span>
    <meta property="address" content="C/Valdés Salas S/N, Oviedo, Spain" />
   </span>
  </div>
 </p>
</body>
</html>
```

Microdata

```html
<html>
 <head>
 <title>Meeting</title>
 </head>
<body itemscope
        itemid="http://example.org/event1"
        itemtype="http://schema.org/Event">
 <h1>Meeting</h1>
 <p>We will have a
   <span itemprop="name">meeting</span>
   <span itemprop="startDate" content="2021-12-04T12:30">next saturday</span>
   at the
    <span itemprop="location">School
     <meta itemprop="address" content="C/Valdés Salas S/N, Oviedo, Spain"
   </span>
 </p>
</body>
</html>
```

Statistics on structured data



https://w3techs.com/technologies/overview/structured_data

Reasoning

RDFS

- RDFS (RDF Schema) was already created in 1999
  - A language that can be used to describe vocabularies
- It can declare
  - Classes (`rdfs:Class`)
  - Properties (`rdfs:Property`)
  - Resources (`rdfs:Resource`): anything
- Relationships between classes and properties
  - `rdfs:subClassOf`, `rdfs:subPropertyOf`
  - `rdfs:domain`, `rdfs:range`, ...
- Documentation about vocabularies
  - `rdfs:label`, `rdfs:comment`

RDFS: Instances, classes and properties

Difference between

- Instances/individuals: `:alice, :cs101, :Oviedo, ...`
- Classes/concepts: `:Person`, `:Course`, `:City, :Place`
- Properties/relationships: `:teaches`, `:hasCountry`

`rdf:type` declares than an individual has a given type

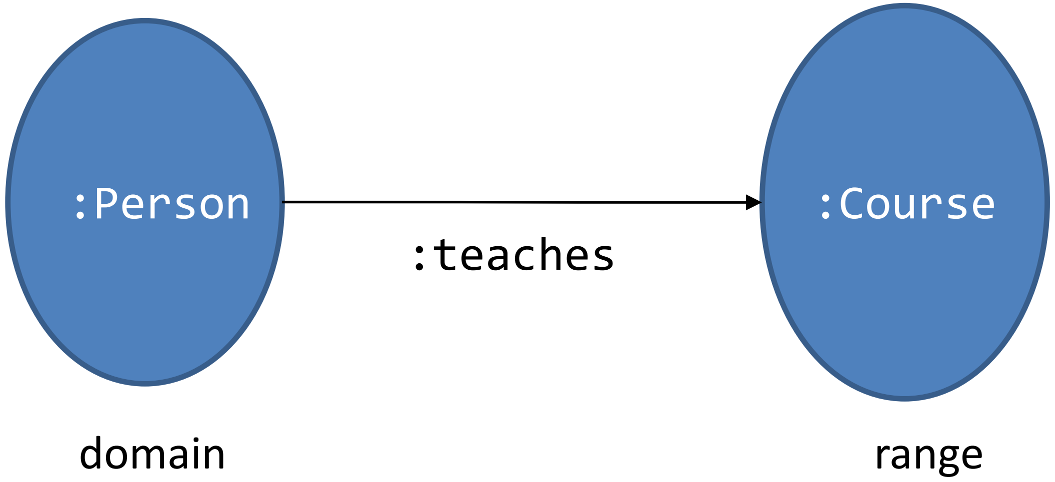
`rdfs:subClassOf` declares than a Class is a subclass of another

RDFS: Domain and range

It is possible to declare domain and range constraints

Example: `:teaches`

- Domain: `:Person`
- Range: `:Course`



RDFS: Class hierarchies

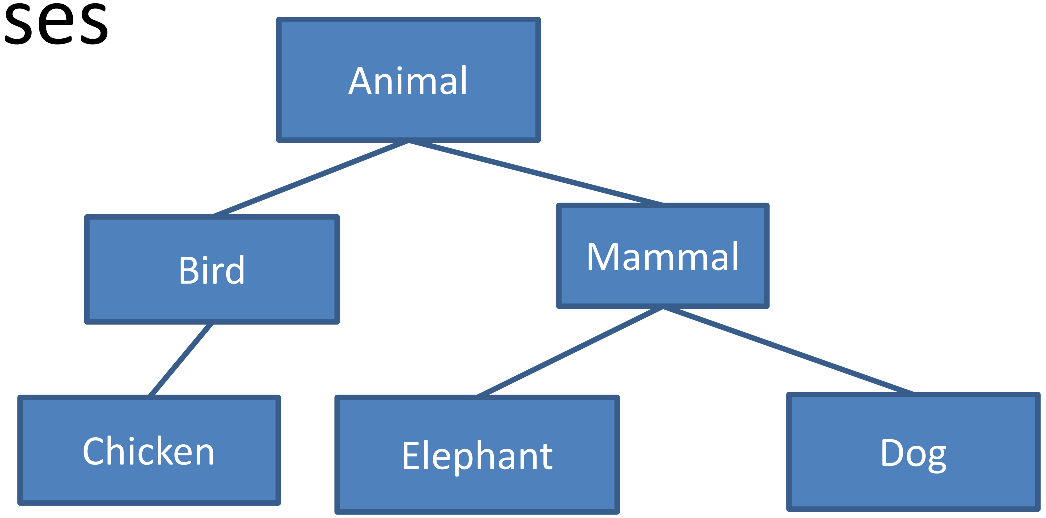
Classes can be organized in hierarchies

`rdfs:subClassOf` defines that a class is a subclass of another

A is a subclass of B if every individual from A is in B

- In that case, B is a superclass of A

A class can have multiple superclasses

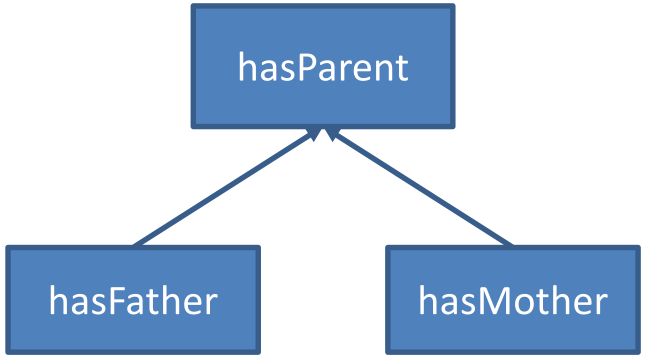


RDFS: Property hierarchies

Properties can also be organized in hierarchies

`rdfs:subPropertyOf` defines that a property is a property of another

p is a subProperty of q if for all `x,y` such that `(x,p,y)` then `(x,q,y)`



`:alice :hasFather :bob ⇒ :alice :hasParent :bob`

RDFS: inferences

RDFS has a built-in semantics which allows to infer new triples

Example rules:

X rdf:type A ∧ A subClassOf B → X rdf:type B

A rdfs:subclassOf B ∧ B rdfs:subClassOf C → A rdfs:subClassOf C

P rdfs:domain A ∧ X P Y → X rdf :type A

P rdfs:range B ∧ X P Y → Y rdf:type B

P rdfs:subPropertyOf Q ∧ Q rdfs:subPropertyOf R → P rdf s:SubpropertyOf R

P rdfs:subPropertyOf Q ∧ X P Y → X Q Y

RDFS expressivity

It can be used as a first step towards vocabulary definitions

But...

- It has limited expressivity to define, for example:
  - Negative information: *A Person is not a Car*
  - Quantifiers: *Every parent must have at least one children*
  - Cardinality: *A good student must have passed at least 3 courses*
  - Attributes about properties
    - Example: Inverse property: If x hasParent y, then y hasChild x
    - Other attributes: symmetry, transitive, etc.
- It can also be too liberal
  - You can declare, in RDFS:
    - `:Person rdf:type :Person`

OWL

OWL = Web Ontology Language

Developed as part of W3C initiative

- Antecedents: DAML, OIL
- Based on Description Logics
- 2004 - OWL 1.0 W3C recommendation
- 2009 - OWL 2.0 W3C recommendation

OWL expressivity

http://www.cs.man.ac.uk/~ezolin/dl/

OWL adds a lot of expressivity

More expressivity $\Rightarrow$ More complexity

Several levels

- OWL Full: No restrictions
  - It can be undecidable
- OWL DL: Decidable but has NPEXPTime Complexity
- Several profiles
  - OWL EL, OWL QL, OWL RL, ...
  - More efficient, less expressive

Expresiviness vs Complexity

OWL: description logics

OWL DL is based on description logics

It corresponds to the family: $\mathcal{SHOIN}(\mathcal{D}n)$

http://dl.kr.org/

Description logics is a subset of first order predicate logic

- Goal: restrict expressivity to obtain decidiblity
- Predicates with one or two arguments
  - Person(alice)
  - Teaches(alice,cs101)
- The use of variables is restricted

OWL: TBox and ABox

TBox (Terminological) definitions

$\text{Parent} \equiv \text{Person} \cap \exists\ \text{hasChild Person}$

$\text{Proud} \equiv \text{Person} \cap \exists\ \text{hasChild NewBorn}$

$\text{NewBorn} \subseteq \text{Person}$

ABox (Assertional) definitions

NewBorn(bob)
hasChild(alice,bob)
Person(alice)

OWL and predicate logic

Description logic definitions can be translated for predicate logic

$\text{Parent} \equiv \text{Person} \cap \exists\ \text{hasChild Person}$

$\forall x(\text{Padre}(x) \leftrightarrow (\text{Person}(x) \wedge \exists y(\text{hasChild}(x,y) \wedge \text{Person}(y)))$

$\text{Proud} \equiv \text{Person} \cap \exists\ \text{hasChild NewBorn}$

$\forall x(\text{Proud}(x) \leftrightarrow (\text{Person}(x) \wedge \exists y(\text{hasChild}(x,y) \wedge \text{NewBorn}(y)))$

$\text{NewBorn} \subseteq \text{Person}$

$\forall x(\text{NewBorn}(x) \rightarrow \text{Person}(x))$

OWL: concept definition

Equivalence: $C \equiv D$

Example: Spaniard $\equiv$ SpanishNationality

Subclass: $C \subseteq D$ (C is a subset of D)

Example: Spaniard $\subseteq$ European

Intersection: $C \cap D$

Example: Woman $\equiv$ Person $\cap$ Female

Union: $C \cup D$

Example: Person $\equiv$ Man $\cup$ Woman

Complement: $\neg C$

Example: Male $\equiv \neg$ Female

Empty concept: $\perp$

Disjoint classes $C \cap D \equiv \perp$

OWL: quantifiers

Existential ($\exists$ R C)

x belongs to $\exists$ R C if there's some y $\in$ C such that R(x,y)

Example: Mother $\equiv$ Woman $\cap$ $\exists$ hasChild Person

Universal ($\forall$ R C)

x belongs to $\forall$ R C if for all y, if R(x,y) then y $\in$ C

Example: HappyWoman $\equiv$ Mother $\cap$ $\forall$ hasChild Healthy

A mother is happy is all her children are healthy

OWL: cardinalities

Cardinality $(P = n)$

x belongs to $(P = n)$ if there are n $y \in C$ such that $R(x,y)$

Example: $Elephant \subseteq Animal \cap hasLegs = 4$

Max. cardinality $(P \leq n)$

x belongs to $(P \leq n)$ if there are n or less $y \in C$ such that $R(x,y)$

Example: $BadStudent \equiv Student \cap passesCourse \leq 3$

Min. cardinality $(P \geq n)$

x belongs to $(P \geq n)$ if there are n or more $y \in C$ such that $R(x,y)$

Example: $GoodStudent \equiv Student \cap passesCourse \geq 3$

$$\forall x(GoodStudent(x) \leftrightarrow Student(x) \wedge \exists y_1 \exists y_2 \exists y_3\, (passesCourse(x, y_1) \wedge passesCourse(x, y_2) \wedge passesCourse(x, y_3) \wedge y_1 \neq y_2 \wedge y_2 \neq y_3 \wedge y_1 \neq y_3\,))$$

OWL: property attributes

Reflexive: P is reflefive $\equiv \forall x P(x,x)$

Example: livesWith is reflexive

Irreflexive: P is irreflexive $\equiv \forall x \neg P(x,x)$

Example: hasParent is irreflexive

Simmetry. If P(x,y) then P(y,x)

Example: sibling

Asimmetry. If P(x,y) then $\neg$P(y,x)

Example: hasParent

Transitivity. If P(x,y) and P(y,z) then P(x,z)

Example: sibling

OWL: property relationships

Inverse: P is inverse of Q $\equiv P(x,y) \Leftrightarrow Q(y,x)$

Example: hasParent is inverse of hasChild

Subproperty: P is a subproperty of Q $\equiv P(x,y) \Rightarrow Q(x,y)$

Example: hasChild is a subproperty of hasDescendant

OWL: Functional properties

Functional property. $P(x,y)$ and $P(x,z) \Rightarrow y = z$

Example: birthPlace

Inverse functional property.

$P(x,y)$ and $P(z,y) \Rightarrow x = z$

Example: ssn

Keys. similar to inverse functional properties but specific for a class

$P(x,y)$ and $P(z,y) \Rightarrow x = z$.

Example: ssn

OWL in practice

- Lots of ontologies have been defined
- Tools: Protégé, TopBraid Composer, ...
- Foundries: OBOFoundry,
- Ontology Engineering development
  - Gradually adopting software engineering techniques like TDD
- OWL and reasoning
  - OWL reasoning can be too complex in practice
  - Ongoing discussions about use cases for OWL, Rules, Shapes,....

Trends in research data

- Share and reuse research data
  - Reproducibility of experiments?
  - Represent workflows and pipelines?
  - Research is more than a PDF paper
- Some initiatives:
  - FAIR principles
  - Nanopublications
  - Research Objects

FAIR principles

https://www.go-fair.org/fair-principles/

(F)indable

F1. (Meta)data are assigned a globally unique and persistent identifier

F2. Data are described with rich metadata (defined by R1 below)

F3. Metadata clearly and explicitly include the identifier of the data they describe

F4. (Meta)data are registered or indexed in a searchable resource

(A)ccesible

A1. (Meta)data are retrievable by their identifier using a standardised communications protocol

A1.1 The protocol is open, free, and universally implementable

A1.2 The protocol allows for an authentication and authorisation procedure, where necessary

A2. Metadata are accessible, even when the data are no longer available

(I)nteroperable

I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.

I2. (Meta)data use vocabularies that follow FAIR principles

I3. (Meta)data include qualified references to other (meta)data

(R)eusable

R1. (Meta)data are richly described with a plurality of accurate and relevant attributes

R1.1. (Meta)data are released with a clear and accessible data usage license

R1.2. (Meta)data are associated with detailed provenance

R1.3. (Meta)data meet domain-relevant community standards

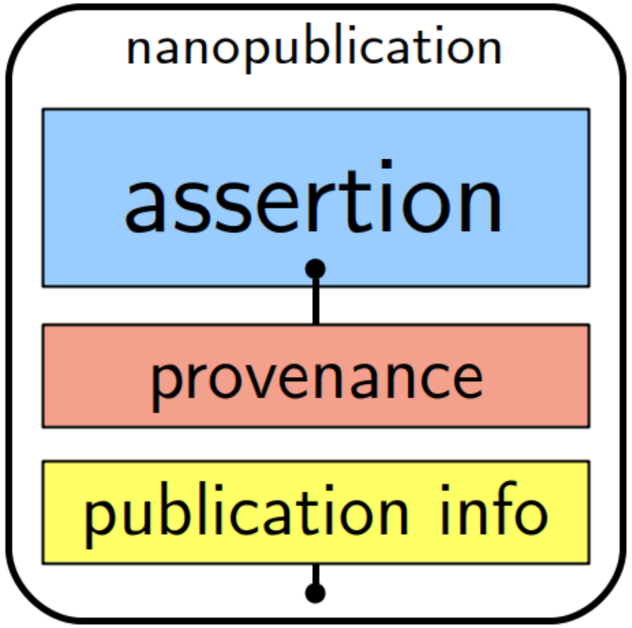
Nanopublications

Smallest unit of publishable information

- About any topic
- Fully expressible in machine-interpretable way (RDF)

3 parts

- Assertion: main content
- Provenance: how the assertion came to be
  - Example: Scientific methods employed
- Publication info: Metadata about nanopublication as a whole
  - Example: When? By whom?



https://nanopub.org/

Nanopublications

Example

using TriG

```
@prefix : <http://example.org/pub1/> .
@prefix ex: <http://example.org/> .
@prefix np: <http://www.nanopub.org/nschema#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix orcid: <https://orcid.org/> .

:Head {
  : a np:Nanopublication              ;
    np:hasAssertion       :assertion  ;
    np:hasProvenance      :provenance ;
    np:hasPublicationInfo :pubinfo
}
:assertion {
  ex:oviedo ex:hasTemperature 36 .
}
:provenance {
  :assertion prov:wasDerivedFrom :slides
}
:pubinfo {
  : dct:creator orcid: 0000-0001-8907-5348 ;
    dct:created "2021-12-01T07:51:00"^^xsd:dateTime  .
}
```

Nanopublications in practice

Nanopub servers: Publish and search nanopublications

Integrity keys can be used to check immutability

- Trusty URIs = include hash values in URIs
- Example: http://server.nanopubs.lod.labs.vu.nl/RAn15vsPJEVdJvjNKtBPo_oadtjeP9oc3Si-69FiJ4poQ

Nanopub monitor:

http://app.tkuhn.eculture.labs.vu.nl/nanopub-monitor/

FAIR principles as a nanopub: https://github.com/peta-pico/FAIR-nanopubs/blob/master/principles.trig

Research Objects

https://www.researchobject.org/

Support publication of:

- More than just PDF, example: data, code, etc.
- Collection of resources
- Annotations about resources

Core principles: identity, aggregation, annotation

Specification: RO-Crate (https://www.researchobject.org/ro-crate)

- Based on JSON-LD

Myths about Semantic Web, linked data, KGs...

Too expensive

It's free

No one will want our data

It is not trendy

Too much openness

Our project will be a success

It is too expensive

Not really

It is possible to learn from previous experiences

Follow the lemma:

- *Separate content from presentation*
  - Content: Information/data
  - Presentation: visual/aesthetic aspects
- Try not to lose semantics

Content

Presentation

It is free

...well, no

It requires to complement with visualizations

- Only data = too restrained

Requires to define data models and URIs

- Stable and cool URIs

Challenge of continuous updates

- Take care of data pipelines

No one will be interested in our data

Just the opposite...our data = our treasure

Search engines index semantic content

- schema.org Project (Google, Bing, Yandex,...)
- If we help them ⇨ better SEO
- Automatic processable data = more valuable data

Promote data culture

- New business opportunities and applications
- Government = catalyzer: Hackathons y similars...

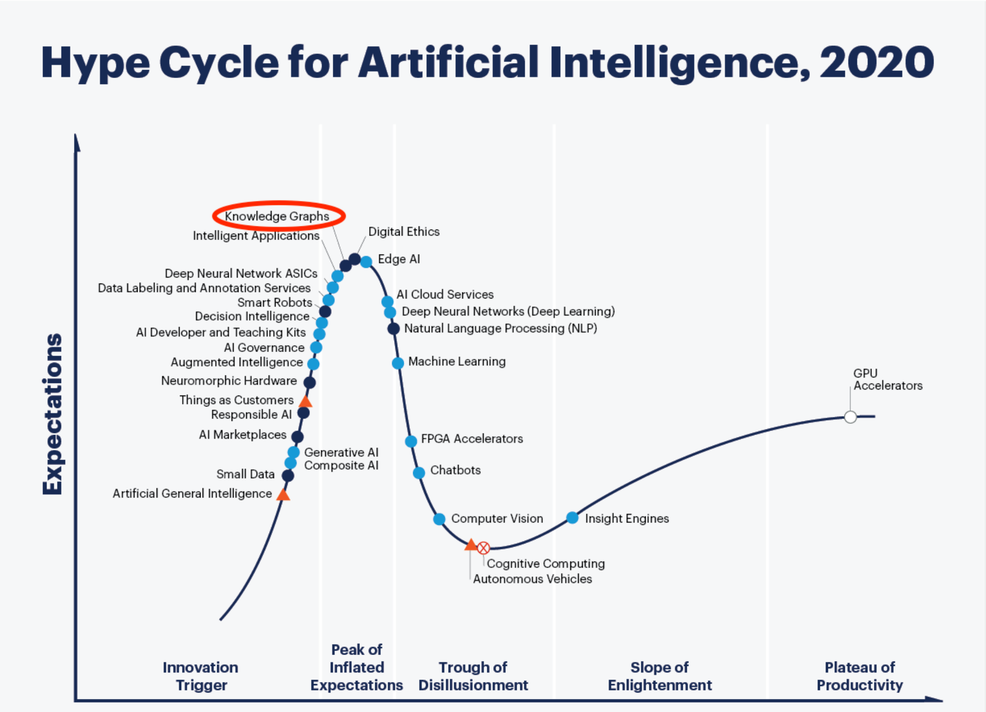
It is not trendy...

Beware of trends in computer science...

Knowledge graphs seem to be trendy

But...lots of technologies appear/dissapear

Is Semantic web trendy? And linked data?



https://allegrograph.com/gartner-hype-cycle-for-ai-knowledge-graphs/

Too much open

If we really believe in transparency...

- We will pursue reusable data

Nevertheless...

- It is useful to distinguish between:
  - Open data
  - Linked data
  - Public/private data
  - Aggregated data
  - Partially open data
  - Linked and closed data
  - FAIR data
  - ...

Backup slides

OWL: Ontologies

Declare classes, properties and entities in a domain

Focus on the domain model

Knowledge representation

Ontologies enable reasoning and classification

Open World Assumption

We assume there are a lot of knowledge we may not have yet

Rules

Declare conditions IF….THEN….

Focus on the domain model

Knowledge representation

Premises/conclusions

Open/Closed World Assumption

Both assumptions can be used depending on the domain